Introduction to *cthmm* (Continuous-time hidden Markov models) package

Abstract

A disease process refers to a patient's traversal over time through a disease with multiple discrete states. Multistate models are tools used to describe the dynamics of disease processes. Clinical study settings present modeling challenges, as patients' disease trajectories are only partially observed, and patients' disease statuses are only assessed at discrete clinic visit times. Furthermore, imperfect diagnostic tests may yield misclassification error in disease observations. Observational data, such as that available in electronic medical records, present additional challenges, since patients initiate visits based on symptoms, and these times are informative about patients' disease histories.

The *cthmm* package fits multistate models to discretely observed disease process data, allowing for misclassification in disease observations and informative patient visit times. The basic approach is to model the disease process via a latent continuous time Markov chain (CTMCs), allowing for approximation of arbitrary sojourn time distributions with non-constant hazard functions. Moreover, these models can be readily expanded to accomodate information observation times, by specifying the informative observation process as a Markov-modulated Poisson process where patient-initated visit times depend on the patient's disease status.

We illustrate the use of the *cthmm* package in these two discrete observation contexts

1. A panel observed reversible disease process, in which observation times are non-informative.
2. A disease process with a competing risks structure, observed at a combination patient-initiated (informative) and scheduled (non-informative) times.

Using these examples, we describe the core functionality of the package, including

1. Specifying the structure for the data and the model components
2. Obtaining maximum likelihood estimates for model parameters
3. Estimating the variance of the parameter estimates
4. Summarizing result by plotting estimated hazard and cumulative distribution functions for disease state sojourn times.

Contents

1 Background on Latent CTMCs **2**
- 1.1 Latent CTMC model parameterization . . . 2
- 1.2 Accomodating informative observation times . . . 3
- 1.3 Incorporating covariates into the latent CTMC models . . . 3

2 Specifying the data and model structure: panel data example **4**
- 2.1 Data-generating model . . . 4
  - 2.1.1 Covariate effects . . . 4
- 2.2 Data structure . . . 4
  - 2.2.1 Accomodating known times of absorption . . . 5
  - 2.2.2 Baseline covariate data . . . 5
- 2.3 Model specification: transition intensity matrix . . . 5
  - 2.3.1 Fixed transition intensity matrix . . . 7
- 2.4 Model specification: initial distribution . . . 7
  - 2.4.1 Fixed initial distribution . . . 8
- 2.5 Model specification: Emission matrix . . . 8
  - 2.5.1 Covariate-parameterized emission matrix . . . 8
  - 2.5.2 Emission distribution when some states are observed without error . . . 9
  - 2.5.3 Fixed emission matrix . . . 10

3 Specifying the data and model structure: informative sampling time example 11
3.1 Data-generating model . . . 11
3.2 Data structure . . . 11
3.3 Model specification: transition intensity matrix . . . 12
3.4 Model specification: initial distribution . . . 13
3.5 Model specification: emission distribution . . . 13
3.6 Model specification: informative sampling time model . . . 15

4 Obtaining maximum likelihood estimates via the EM algorithm 16
4.1 EM for panel data example . . . 16
4.2 EM for informative sampling time example . . . 17

5 Fitted intensity matrices, initial distributions, and emission matrices 17

6 Variance of parameter estimates 18

7 Estimating CDFs and Hazard functions for sojourn time distributions 19
7.1 Estimating CDFs: panel data example . . . 19
7.1.1 Transition intensity matrices for first passage times . . . 20
7.1.2 Standard errors of CDFs . . . 20
7.2 Estimating hazard functions: informative sampling time example . . . 23

1 Background on Latent CTMCs

1.1 Latent CTMC model parameterization

Discrete observation scenarios for multistate processes present modeling challenges, as many of the methods available for fully observed trajectories are no longer computationally tractable. For this reason, standard CTMCs are frequently used to model discretely observed disease processes. Standard CTMCs describe transitions in a multistate process where sojourn time distributions are exponentially distributed and depend on the process history only via the currently occupied state.

Despite their tractability, the assumptions of standard CTMCs are often unrealisitc. *Cthmm*'s approach of using latent CTMCs allows for more flexibility. We assume that the disease process $W(t)$, with state space G, is based on an underlying CTMC $X(t)$, with state space S. These models retain the basic framework of standard CTMCs but assume that multiple latent states in S map to each observable disease state in G, thereby allowing for non-exponentially distributed sojourn times for $W(t)$.

Latent CTMCs are parameterized via an intensity matrix $\mathbf{\Lambda}$, describing rates of transitions between latent states; an initial distribution $\boldsymbol{\pi}$; and an emission matrix $\mathbf{E}$ assigning mappings between the latent states in S and observed state space G. In this sense, we can view a discretely observed latent CTMC as a hidden Markov model based on a time-inhomogeneous transition matrix. From this perspective, we can modify the transition matrix to incorporate misclassified disease observations, by assuming that the observed states at each observation time are probablistically, rather than deterministically, related to the underlying latent states.

Notationally, let $O(t)$ be the observed data at time t, which refers to a possibly misclassified observation of the disease process. The emission matrix $\{E[i,j]\}$ describes the probability of observing a value of $O(t)$ given the underlying value of the latent CTMC $X(t)$ and has entries has entries $E[i,j] = P(O(t) = j|X(t) = i)$. If the process is observed without error, these entries are either 0 or 1.

Latent CTMC models provide a means of approximating disease processes with arbitrary sojourn time distributions. In most cases, the latent structure is not inherently meaningful. Furthermore, certain parameterizations may yield models that may not be identifiable. To alleviate these problems, we recommend use of Coxian phase-type structures to characterize the

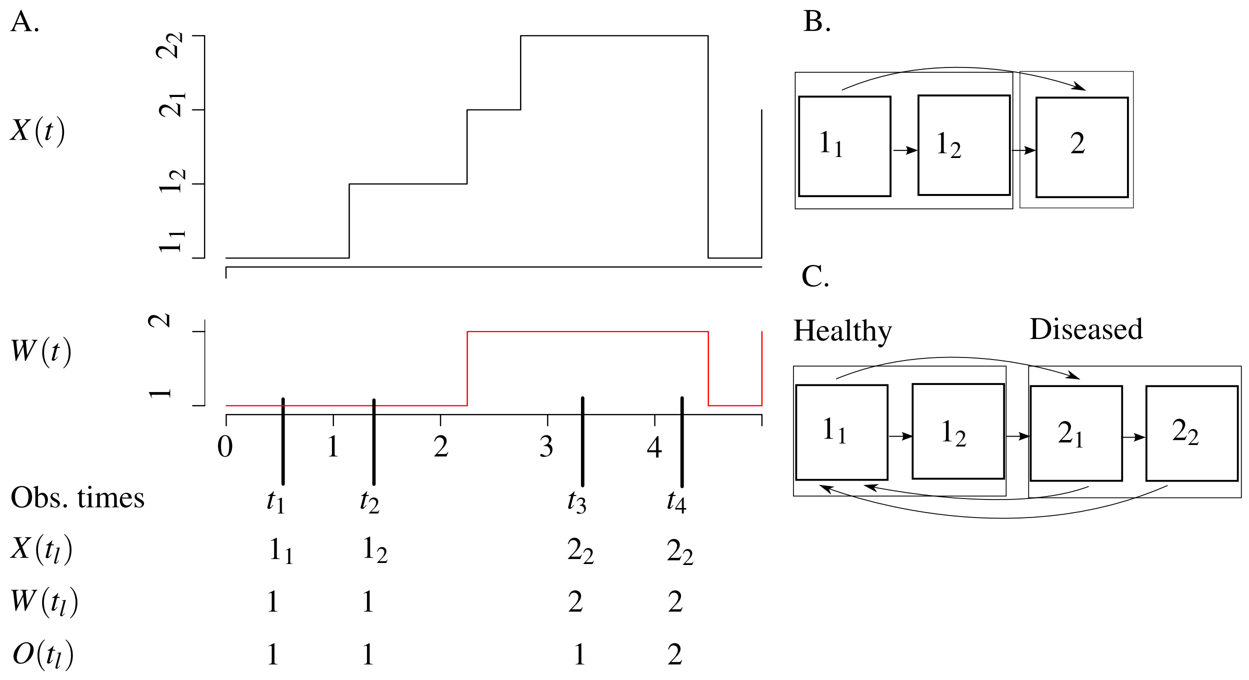


Figure 1: A. Example of latent trajectory $X(t)$, disease trajectory $W(t)$, and observed data $O(t_l)$ at discrete observation times for model in subfigure C, assuming possible misclassification error. B. 2-state survival model of W(t) assuming $G = \{1, 2\}$ and $S = \{\{1_1, 1_2\}, \{2\}\}$, where disease state 2 is absorbing. The Coxian PH structure implies X(t) starts in 1_1. C. 2-state reversible model of W(t), with state space $R = \{1 = \mathit{Healthy}, 2 = \mathit{Diseased}\}$ and $S = \{\{1_1, 1_2\}, \{2_1, 2_2\}\})$. $X(t)$ starts in 1_1 or 2_1.

sojourn times in the latent space (Figure 1C). Such models characterize limit the number of allowable transitions in the latent space, but still provide for flexible models that approximate arbitrary sojourn time distributions .

Figure 1A provides an illustration of a complete, but unobserved trajectory $W(t)$, and its relation to the latent trajectory $X(t)$ and the observed data consisting of discrete observations, $O(t_1), \ldots, O(t_n)$. The underlying structure of $X(t)$ allows for Coxian phasetype distributions for the sojourn times in each of the two disease states.

1.2 Accomodating informative observation times

Patient-initiated, disease-driven observations (DDOs) are frequently present in clinical datasets. To accomodate such times within the latent CTMC framework, we can model DDO times according to a Markov-modulated Poisson process where rates of informative observation events depend on the underlying disease state. Formally, the rates of DDOs are a piecewise function $q(t) = q(X(t))$ that depend on the underlying disease state. In particular, rates of DDOs corresponding to disease states $\{1, \ldots, s\}$ are denoted $\mathbf{q} = (q_1, \ldots, q_s)'$.

1.3 Incorporating covariates into the latent CTMC models

To incorporate baseline subject-level covariates $\mathbf{w}^{(k)}$ in the disease model, we relate log-rates to a linear predictor, $\log(\lambda_{ij}^{(k)}) = \boldsymbol{\zeta}_{ij}^T \mathbf{w}^k$, where k denotes the individual. In latent CTMCs, different constraints on covariate effects provide different interpretations. Adding the same covariate parameter to all latent transitions originating from disease state p, i.e. $\left\{\lambda_{ij} : i \in \{p_1, \ldots, p_{s_p}\}\right\}$, implies a multiplicative effect on the sojourn time in state p. To represent covariate effects on cause-specific hazard functions, one can add a separate covariate parameter for each transition out of disease state p to disease state r, i.e. $\left\{\lambda_{ij} : i \in \{p_1, \ldots, p_{s_p}, j \in \{r_1, \ldots, r_{s_r}\}\right\}$. This specification does not, however, represent a proportional hazards parameterization without additional non-linear constraints.

One can also add covariates to DDO, emission, and initial distribution parameterizations. This is achieved by relating log rates of DDOs to a linear predictor; i.e. $\log(q_i^{(k)}) = \boldsymbol{\nu}_i^T \mathbf{w}^{(k)}$. Initial distributions and emission distributions are multinomial. Assuming S has s total states, the initial distribution $\boldsymbol{\pi}$ has natural parameters $\{\eta_i = \log\left(\pi_i/\pi_1\right) : i = 2, ..., s\}$, and the emission distribution $\mathbf{e}_i$ has natural parameters $\{\eta_{ij} = \log\left(E[i,j]/E[i,1]\right) : j = 2, ..., g\}$. Subject-level covariates $\vec{w}^{(k)}$ are added to the multinomial models via a linear predictor, e.g., specifying $\eta_{ij}^{(k)} = \boldsymbol{\gamma}_{ij}\mathbf{w}^{(k)}$.

2 Specifying the data and model structure: panel data example

2.1 Data-generating model

First, we illustrate *cthmm's* use for panel data with non-informative observation times. The disease model used to generate the disease process has two states $\{A, B\}$ and reversible transitions between these states (Figure 2). The sojourn times in states A and B are generated, respectively, according to Weibull(shape=1.5 and scale=1) and Weibull(shape=.75, scale=10) distributions, which correspond to sojourn time distributions with increasing and discresing hazard functions, respectively. There are 200 independent individuals in the sample, each observed at fixed times $0, 1, 2, \ldots, 10$ years. Disease states A and B are observed with misclassification error.

2.1.1 Covariate effects

For each individual, there are two baseline covariates: a binary variable X (half of the sample have X=0, and half have X=1) and a continous variable Y (generated independently from a Normal(0,1) distribution). The covariate X affects occupancy times, such that for those with X=1 (versus X=0), sojourns in state A are multiplied by 1.5 and in state, B, by .75.

The covariate X also affects the misclassification probabilites. The probability of observing B given A is .028 in those with X=0, and the odds of missification increase by a factor of 3.7 in individuals with X=1. The probability of observing A given B is .05, and this probability is not affected by covariates.

The state initially occupied by an individual is governed by an initial probability distribution that is related to the covariate Y. Those with Y=0 occupy state B with probability .094. An increase in Y by one unit increases the odds of occupying state B by 2.55.

2.2 Data structure

Data are stored in a list, with an entry for every subject. The list elements include the entries "obs.data" and "obs.times" correpsponding to the observed data at each of the discrete observation times. Observations of state A are coded by 1 and B, by 2.

```
> the.data[[1]]$obs.data

 [1] 2 2 2 2 2 1 2 2 2 2 2

> the.data[[1]]$obs.times

 [1] 0.000000 1.488500 2.112423 2.969151 3.860416 4.960169 5.622864 7.234300
 [9] 8.186855 9.080619 9.672877
```

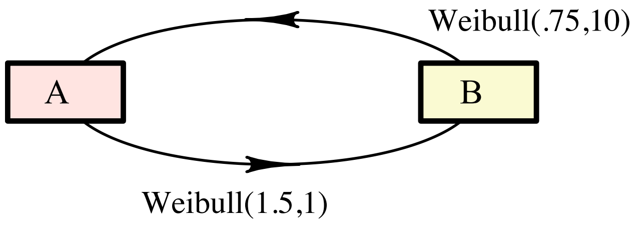


Figure 2: Data-generating model for the two-state reversible disease process. Sojourn times in state A have a Weibull(1.5,1) distribution and in state B, a Weibull(.75,10) distribution.

2.2.1 Accomodating known times of absorption

Note: this model has no absorbing states. In situations where the disease has an absorbing state, indivduals may have a final observations correpond to a known time of absorption, such as a known time of death. In this situation, we need to specify which observation corresponds to the the exact time. If there are 10 observations for person 1, and the last time corresponds to a known absorbtion time, then we would also include the item

```
>  the.data[[1]]$exact.times.ranks=10.
```

2.2.2 Baseline covariate data

The baseline covariate data is in a data frame called `cov.data.` Each subject is identified by an id in the same order as `the.data` list.

```
> cov.data[1:10,]
```

```
   id X          Y
1   1 1 -0.5982105
2   2 1  1.5467532
3   3 1  0.7701050
4   4 1  0.2851026
5   5 1  2.2890916
6   6 1  1.6181584
7   7 1  0.3472898
8   8 1  0.2726347
9   9 1 -0.4469496
10 10 1 -1.1877430
```

2.3 Model specification: transition intensity matrix

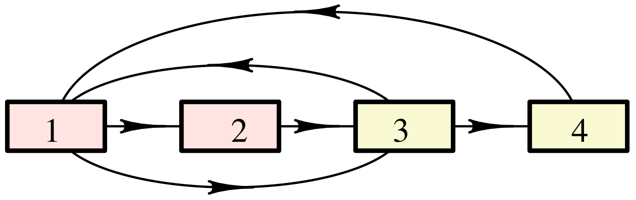


Figure 3: Latent continuous time model for two state disease process. Latent states $\{1, 2\}$ map to disease state A and $\{3, 4\}$ to disease state 2. The transitions between latent states in each disease state have Coxian phase type structure, in at each transition, it is only possible to go forward or exit to the other disease state.

We will fit the data with latent CTMC model that has a total of 4 latent states, 2 per each disease state. The latent state space is $S = \{1, 2, 3, 4\}$ where states $\{1, 2\}$ correspond to disease state A and states $\{3, 4\}$ to disease state B. The latent CTMC has transition intensity matrix $\mathbf{\Lambda}$. The model is characterized by Coxian phase type sojourn distributions for times spent in the disease states, and the latent structure has transitions implied by Figure 3. The $\mathbf{\Lambda}$ matrix that corresponds to this structure is

$$\Lambda = \begin{bmatrix} -(\lambda_{12}+\lambda_{13}) & \lambda_{12} & \lambda_{13} & 0 \\ 0 & -\lambda_{23} & \lambda_{23} & 0 \\ \lambda_{31} & 0 & -(\lambda_{31}+\lambda_{34}) & \lambda_{34} \\ \lambda_{41} & 0 & 0 & -\lambda_{41} \end{bmatrix}.$$

We incorporate covariates in the transition matrix model as follows. Since the covariate X scales multiplicatively the sojourn time distributions in A and B, we can assume that the covariate effect of X is the same for $\lambda_{12}, \lambda_{13}, \lambda_{23}$ and likewise for $\lambda_{34}, \lambda_{31}, \lambda_{41}$. The model is specified in terms of log-rates and has parameters $\{r_1, \ldots, r_8\}$ as follows:

$$\begin{aligned} log(\lambda_{12}) &= r_1 + r_7 X \\ log(\lambda_{13}) &= r_2 + r_7 X \\ log(\lambda_{23}) &= r_3 + r_7 X \\ log(\lambda_{34}) &= r_4 + r_8 X \\ log(\lambda_{31}) &= r_5 + r_8 X \\ log(\lambda_{41}) &= r_6 + r_8 X. \end{aligned}$$

We specify the design matrix for the rate intensity matrix via a list called `rates.setup`. The list as several elements, which need to be set by the user before fitting the EM. We specify the number of latent states in the model

```
> rates.setup$num.states
```

```
[1] 4
```

We specify the non-zero transitions between the latent states with a matrix

```
>  rates.setup$transition.codes
```

```
  i j
1 1 2
2 1 3
3 2 3
4 3 4
5 3 1
6 4 1
```

We specify the design matrix for the model with the matrix

```
>  rates.setup$design.matrix
```

```
  r1 r2 r3 r4 r5 r6 r7 r8
1  1 NA NA NA NA NA  X
2 NA  1 NA NA NA NA  X
3 NA NA  1 NA NA NA  X
4 NA NA NA  1 NA NA     X
5 NA NA NA NA  1 NA     X
6 NA NA NA NA NA  1     X
```

Each row in the design matrix corresponds to a single $i \to j$ transition in the order specified by the `rates.setup$transition.codes`. Intercepts are represented by 1 entries. Named covariates are specified with the name of the covariate (here, X). The setup allows us to easily specify that parameters are common across multiple transitions, e.g. r_7 and r_8. We also need to specify for each parameter whether it is fixed or will be estimated. if it is fixed, its value is that designated in `rates.setup$param.values`. If we set param.types=0, then we are going to estimate it, if 1, it is fixed. In this example we will be estimating all parameters.

```
> rates.setup$param.types
```

```
[1] 0 0 0 0 0 0 0 0
```

We specify intial values for $r_1, \ldots r_8$ (for the EM) with

```
>  rates.setup$param.values
```

```
[1]  0.21905279 -1.54918429  0.51695069 -2.33470966 -1.78216158 -3.73949033
[7] -0.39584792  0.03575836
```

In the case we are not estimating a parameter, the value is fixed at the initial value.

2.3.1 Fixed transition intensity matrix

For certain models, we may want to specify a fixed transition intensity matrix, with no unknown entries. In this case, we have the option to specify by setting the list item

```
> rates.setup$fixed.rates
```

to be equal to the fixed rate matrix for all individuals. This option may be most useful in the debugging stage. Note that all of the entries in the fixed rate matrix are on the intensity, not log-intensity, scale.

2.4 Model specification: initial distribution

The model for the initial distribution of the latent CTMC is a multinomial logit model. In the example, the initial distribution is $(\pi_1, ..\pi_4)$. According to the Coxian specification, an individual allows starts in the first latent state of the corresponding disease state. So, for our model, an individual either starts in latent state 1 or latent state 3. The probability of the initial state occupancy is related to the covariate Y. With state 1 as the reference state, the model is

$$\log \frac{\pi_3}{\pi_1} = b1 + b2Y.$$

To specify the initial distribution model, we have an `init.setup` list. We have an entry for the number of states in the model.

```
> init.setup$num.states
```

```
[1] 4
```

We identify the reference state via

```
> init.setup$ref
```

```
[1] 1
```

We then specify all of the states, other than the reference state, that can be occupied at the initial time point. States that cannot be initially occupied are not listed. Therefore, we do not list states 2 and 4, just 3.

```
> init.setup$states
```

```
[1] 3
```

We then specify the design matrix for the initial distribution. Each row corresponds to the state listed in `init.setup$states`.

```
> init.setup$design.matrix

  b1 b2
1  1  Y
```

We specify which of the parameters are fixed versus unknown. The code is 0 for unknown, 1 for fixed.

```
> init.setup$param.types

[1] 0 0
```

We specify the initial values of the parameters–or their fixed value if will not be estimating them in the model.

```
> init.setup$param.values

[1] -2.3604716  0.8488266
```

2.4.1 Fixed initial distribution

Alternatively, we can specify the initial distiribution as a fixed value for everyone. For example, assume that we knew that the individual started in latent state 1. Then we would specify

```
> init.setup$fixed.dist=c(1,0,0,0)
```

We note that we use the probability scale for specifying a fixed initial distribution.

2.5 Model specification: Emission matrix

In this example the latent CTMC model we are fitting has state space $\{1, 2, 3, 4\}$, and the observed data state space is $\{A, B\}$. Thus, the emission matrix has four rows, one for latent state and 3 columns, one for each observable state.

The emission setup object `emission.setup` contains the information about the emission matrix parameterization and is analogous to `rates.setup` and `init.setup`.

2.5.1 Covariate-parameterized emission matrix

The emission distribution model, and relationship to covariates, are specified according to separate multinomial logistic regression models, one for each latent state. In our model we will assume that $E[1, 2] = E[2, 2]$ and that $E[3, 1] = E[4, 1]$, and that only $E[1, 2] = E[2, 2]$ are related to the covariate X. The multinomial logistic regression models conveying the covariate effects can be written as

$$log(\frac{E[1,2]}{E[1,1]}) = g_1 + g_2X$$
$$log(\frac{E[2,2]}{E[2,1]}) = g_1 + g_2X$$
$$log(\frac{E[3,1]}{E[3,2]}) = g_3$$
$$log(\frac{E[4,1]}{E[4,2]}) = g_3$$

To specify the model in `emission.setup`, first we designate the reference categories in multinomial models via a matrix. In this example, our reference categories are $E[1, 1]$, $E[2, 1]$, $E[3, 2]$, and $E[4, 2]$.

```
>  emission.setup$ref.states
```

```
  i j
1 1 1
2 2 1
3 3 2
4 4 2
```

Then we specify all other categories with non-zero emission probabilities that are not reference states.

```
>  emission.setup$emission.states
```

```
  i j
1 1 2
2 2 2
3 3 1
4 4 1
```

The `design.matrix` encodes the parameterization of the model, with each row corresponding to a row in the `emission.states` matrix.

```
>  emission.setup$design.matrix
```

```
  g1 g2 g3
1  1  X NA
2  1  X NA
3 NA     1
4 NA     1
```

We specify which of the parameters will be estimated (0) and which are fixed (1).

```
>  emission.setup$param.types
```

```
[1] 0 0 0
```

We set the initial values of the parameters, or the values of the parameters if they are fixed.

```
>  emission.setup$param.values
```

```
[1] -4.224486  1.802472 -3.179071
```

2.5.2 Emission distribution when some states are observed without error

In some models, some of the states may be observed with error and others observed without error. In this situation, we have the option of specifying this directly. For example, if $E[1,1] = 1$ we set

```
>  emission.setup$exact.states=matrix(c(1,1),nrow=1,dimnames=list(1,c("i","j")))
>  emission.setup$exact.states
```

```
  i j
1 1 1
```

This means that we will not estimate parameters for $E[1,2], E[1,3], etc$, since all other entries in row 1 of the emission matrix will be 0. This does not prevent us from estimating covariates related to other emission probabilities.

2.5.3 Fixed emission matrix

In situations where there is no misclassification error or the misclassification probabilities are known, we specify an emission matrix that is common across indivduals.

For instance, if we assumed that the latent CTMC model we are fitting was observed without misclassification error, the emission matrix would be a 4×2 matrix

$$\mathbf{E} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix},$$

indicating the mapping of the latent states onto observed disease states.

Suppose we have a 5% misclassification error probability. In this case, the emission distribution is

$$\mathbf{E} = \begin{bmatrix} .95 & .05 \\ .95 & .05 \\ .05 & .95 \\ .05 & .95 \end{bmatrix}.$$

In general, to specify a fixed emission distribution, we set

```
>  emission.setup$fixed.dist=matrix(c(1,0,1,0,0,1,0,1),byrow=T,nrow=4)
```

```
>  emission.setup$fixed.dist
```

```
     [,1] [,2]
[1,]    1    0
[2,]    1    0
[3,]    0    1
[4,]    0    1
```

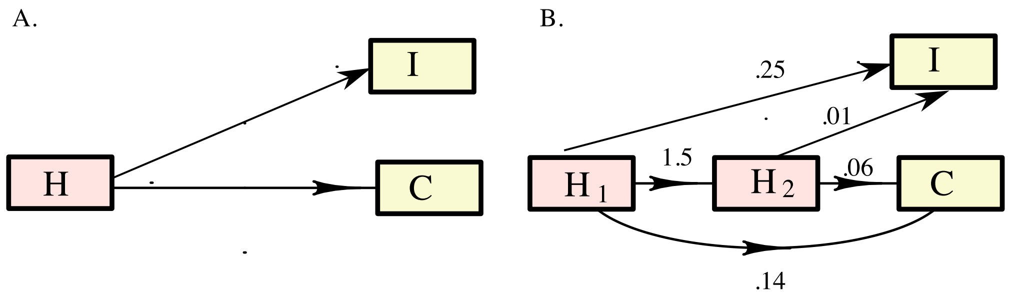


Figure 4: A.Disease model with competing risks structure. State H is a healthy state and states I and C are diseased states. B. Latent CTMC structure for the disease model. States H_1 and H_2 map to the healthy state. The two latent healthy state enable non-constant hazard functions for sojourn times in the disease states. The transition intensities (for covariate X=0) are shown next to the arrows for each transition in the latent model.

3 Specifying the data and model structure: informative sampling time example

3.1 Data-generating model

Our next example will involve a discretely observed disease process with a competing risks structure and a combination of patient-initated (informative) and non-informative observation times. In this example, we assume that the data-generating model is a latent CTMC with a competing risks structure (Figure 4). States H_1 and H_2 map to two healthy states, and I, and C, to separate disease states. States I and C are absorbing, and cannot be exited once the invidual enters them.

The simulated dataset consists of 500 indivuals. For each individual, there is one baseline covariate, X, generated as a random draw from a Normal(0,1) distribution, that affects the transition intensities in the model. Specifically a unit change in X increases the rates of $H_1 \to I$ and $H_2 \to I$ transitions by 4.5 and the $H_1 \to C$ and $H_2 \to C$ transitions by 2.0. There is no covariate effect on the $H_1 \to H_2$ transition.

We assume that all individuals start in state H_1. All individuals have fixed observations at times $t = 0$ years and $t = 8$ years, and at random patient-initiated times in that time frame. The informative observation times occur according to a Markov-modulated Poisson process, with rates that depend on the underlying disease state. In the healthy states H_1 and H_2, informative, disease-driven observation times occur .25 times/year. In the diseased states, they occur 2 times/year.

We assume that the disease process is observed with misclassification error. When an individual is in either of the healthy states, one observes that they are healthy 98% of the time. In contrast, individuals in states I or C are only correctly identified as diseased 70% of the time.

3.2 Data structure

The structure for the data is similar to the panel data structure. All of the data is represented as a list, with one sublist per indivdual. Individual entries specify the discrete obervation times and the observed data at those times. In addition, the sublist specifies which of the times are informative observation times versus panel times.

Given there are 3 observable states in the model (H, I, C), the observed data is coded, correspondingly, by values 1,2, and 3. The type of each observation time (informative versus non-informative) is coded by a vector h. When the observation time is informative, the entry is 1, otherwise it is 0. For these data, all individuals have fixed times at $t = 0$ and $t = 8$. Let's look at the entry for individual 1.

```
> DDO_data[[1]]$obs.data

[1] 1 1 1 1

> DDO_data[[1]]$obs.times

[1] 0.000000 5.391065 7.095911 8.000000

> DDO_data[[1]]$h

[1] 0 1 1 0
```

As we the previous example, the baseline covariate data is in a data frame called `cov.data`. Each subject is identified by an id in the same order as the `DDO_data` list.

```
> cov.data[1:10,]

   id          X
1   1  0.8951574
2   2 -1.3936490
3   3  0.2752159
4   4 -1.8418686
5   5  1.1064556
6   6 -0.4110485
7   7 -0.1435747
8   8 -0.5545592
9   9  0.4188015
10 10  0.7665758
```

3.3 Model specification: transition intensity matrix

We will fit the data with an appropriately specified latent CTMC disease model (Figure 4B.) The latent state space is $S = \{1, 2, 3, 4\}$ where states $\{1, 2\}$ correspond to states H_1 and H_2 and states $\{3, 4\}$ to disease states I and C, respectively. The latent CTMC has transition intensity matrix Λ. As in our previous example, the model is characterized by Coxian phase type sojourn distributions for times spent in the disease states, and the latent structure has transitions implied by Figure 4. The Λ matrix that corresponds to this structure is

$$\Lambda = \begin{bmatrix} -(\lambda_{12}+\lambda_{13}+\lambda_{14}) & \lambda_{12} & \lambda_{13} & \lambda_{14} \\ 0 & -(\lambda_{23}+\lambda_{24}) & \lambda_{23} & \lambda_{24} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

According to the data-generating model we incorporate covariates in the transition matrix model as follows:

$$\begin{aligned} log(\lambda_{12}) &= r_1 \\ log(\lambda_{13}) &= r_2 + r_6 X \\ log(\lambda_{14}) &= r_3 + r_6 X \\ log(\lambda_{23}) &= r_4 + r_7 X \\ log(\lambda_{24}) &= r_5 + r_7 X \end{aligned}$$

As with the panel data example, the transition intensity design matrix information is stored in a list called `rates.setup`. We specify the number of latent states in the model

```
> rates.setup$num.states

[1] 4
```

We specify the non-zero transitions between the latent states via the matrix

```
> rates.setup$transition.codes
```

```
  i j
1 1 2
2 1 3
3 1 4
4 2 3
5 2 4
```

We specify the design matrix for the model via

```
> rates.setup$design.matrix
```

```
  r1 r2 r3 r4 r5 r6 r7
1  1 NA NA NA NA
2 NA  1 NA NA NA  X
3 NA NA  1 NA NA     X
4 NA NA NA  1 NA  X
5 NA NA NA NA  1     X
```

Each row in the design matrix corresponds to a single $i \to j$ transition in the order specified by `rates.setup$transition.codes`. Intercepts are represented by 1 entries. Named covariates are specified with the name of the covariate (here, X). The setup allows us to easily specify that parameters are common across multiple transitions, e.g. r_6 and r_7.

We specify intial values for the coefficients $r_1, \ldots, r_7$ with

```
>  rates.setup$param.values
```

```
[1]  0.405465100 -1.391402000 -1.953979000 -4.527186000 -2.865628000
[6]  0.005176596  1.778063566
```

We also need to specify for each parameter whether it is fixed or will be estimated. If it is fixed, its value is designated in `rates.setup$param.values`. If we set param.types=0, then we are going to estimate it, if 1, it is fixed. In this example we will estimate all parameters.

```
>  rates.setup$param.types
```

```
[1] 0 0 0 0 0 0 0
```

3.4 Model specification: initial distribution

In this example, we assume that all individuals start in state H_1, so we will not be estimating the initial distribution. Thus we specify

```
> init.setup$fixed.dist=c(1,0,0,0)
```

which implies that all individuals start in latent state H_1.

3.5 Model specification: emission distribution

The emission distribution in our model has 4 columns, corresponding to each of the latent states, and 3 rows, corresponding to the possible observable states. The data-generating emission matrix is

$$\begin{bmatrix} .98 & .01 & .01 \\ .98 & .01 & .01 \\ .3 & .7 & 0 \\ .3 & 0 & .7 \end{bmatrix}.$$

We can specify the model using multinomial regression via only intercept terms. In our model we will assume that $E[1,2] = E[1,3] = E[2,2] = E[2,3]$, $E[3,1] = E[4,1]$, and $E[3,3] = E[4,3] = 0$. Assuming that the reference categories are $E[1,1]$, $E[2,1]$, $E[3,2]$, and $E[4,3]$, he multinomial logistic regression models can be written as

$$log(\frac{E[1,2]}{E[1,1]}) = b_1$$
$$log(\frac{E[1,3]}{E[1,1]}) = b_1$$
$$log(\frac{E[2,2]}{E[2,1]}) = b_1$$
$$log(\frac{E[2,3]}{E[2,1]}) = b_1$$
$$log(\frac{E[3,1]}{E[3,2]}) = b_2$$
$$log(\frac{E[3,3]}{E[3,2]}) = b_3$$
$$log(\frac{E[4,1]}{E[4,3]}) = b_2$$
$$log(\frac{E[4,2]}{E[4,3]}) = b_3$$

To specify the emission distribution model, we create a list object `emission.setup`.

First we designate the reference categories in multinomial models via a matrix. In this example.

```
>  emission.setup$ref.states
```

```
     i j
[1,] 1 1
[2,] 2 1
[3,] 3 2
[4,] 4 3
```

Then we specify all other categories with non-zero emission probabilities that are not reference states.

```
>  emission.setup$emission.states
```

```
  i j
1 1 2
2 1 3
3 2 2
4 2 3
5 3 1
6 3 3
7 4 1
8 4 2
```

The `design.matrix` encodes the parameterization of the model, with each row corresponding to a row in the `emission.states` matrix.

```
>  emission.setup$design.matrix
```

```
  b1 b2 b3
1  1 NA NA
```

```
2  1 NA NA
3  1 NA NA
4  1 NA NA
5 NA  1 NA
6 NA NA  1
7 NA  1 NA
8 NA NA  1
```

We specify which of the parameters will be estimated (0) and which are fixed (1).

```
>  emission.setup$param.types

[1] 0 0 1
```

Here, we will not estimate b_3, but rather set it to -50, yielding 0 probabilities for $E[3,3]$ and $E[4,3]$.

We set the initial values of the parameters, or the values of the parameters if they are fixed.

```
>  emission.setup$param.values

[1]  -4.50  -0.84 -50.00
```

3.6 Model specification: informative sampling time model

The informative sampling time model in this example has no covariates, and the rates in H_1 and H_2 are set to be equal, as are the rates in latent states I and C. The DDO model can be expressed as

$$log(q_1) = a$$
$$log(q_2) = a$$
$$log(q_3) = b$$
$$log(q_4) = b$$

where $H_1 = 1$, $H_2 = 2$, $I = 3$ and $C = 4$.

The informative sampling time model is specified via a `DDO.setup` list object, which is similar in form to the `rates.setup` objects. That is, we specify the design for the DDO rates as if we were specifying the design of matrix with diagonal entries $\{q_1, \ldots, q_4\}$, denoted $\mathbf{Q}$.

We first identify the number of latent states in the model

```
> DDO.setup$num.states

[1] 4
```

We then specify that we are modeling the diagonal entries of Q via

```
> DDO.setup$transition.codes

  ni nj
1  1  1
2  2  2
3  3  3
4  4  4
```

Finally, the design matrix is given by

```
> DDO.setup$design.matrix

   a  b
1  1 NA
2  1 NA
3 NA  1
4 NA  1
```

where each row specifies the model for the corresponding entry of `DDO.setup$transition.codes`, or corresponding rate parameters.

We assign which values of parameters will be fixed (1) and which will be estimated (0).

```
> DDO.setup$param.types
```

```
[1] 0 0
```

And we set the intial values for the parameters, or their fixed value if it will be estimated.

```
> DDO.setup$param.values
```

```
[1]  0.3556693 -0.2655589
```

4 Obtaining maximum likelihood estimates via the EM algorithm

Cthmm uses an EM algorithm to obtain maximum likelihood estimates using starting values from the model setup objects. As is typical for EM optimization, global convergence is not guaranteed, and we recommend using a number of randomly selected starting values before assumng that the MLEs have been obtained.

Depending on the model complexity, and convergence tolerance, the EM algorithm may take several minutes to complete. Generally, the latent structure of the models leads to slow rates of convergence. Simplifications to the model, such as fixing certain parameters or reducing the complexity of the latent structure may be necessary in the debugging stages.

4.1 EM for panel data example

Arguments to the EM function include the setup objects for the rate matrix, initial distribution, and emission distribution. We also need to specify the the number of sujbects, the number of latent states, number of observed states, the tolerance for convergence, and the maximum number of EM iterations, and which if any of the latent states are absorbing.

```
fit.4state=EM(rates.setup=rates.setup,
   init.setup=init.setup,
   emission.setup=emission.setup,
   the.data=the.data,
   num.subjects=length(the.data),
   num.states=4,
   num.obs.states=2,
   tol=1e-7,
   absorb.state=NULL,
   maxiter=500)
```

The object returned by the EM algorithm is a list that contains the parameter estimates, the final log-likelihood, the run-time, and other information, including the values of the parameter estimates at each iteration. Estimates of rate parameters are listed first, followed by emission parameters, followed by initial distribution parameters, followed by informative observation parameters. The order in which the parameters appear is the same order of the columns of the design matrix in the rate, emission, and initial distribution setup objects. Note that if parameter was specified as fixed in the setup, it will still appear in the vector of estimates, and will be the same as its specified starting value.

```
>  fit.4state$LL
```

```
[1] -927.9087
```

```
>  fit.4state$param
```

```
         r           r           r           r           r           r 
 0.7554409  -1.6249891   0.8439478  -2.9315493  -1.6741693 -16.0000000 
         r           r           e           e           e           i 
-0.3557577  -0.2533592  -4.1987910   1.7829274  -3.1790904  -2.3723878 
         i 
 0.7381510 
```

```
> fit.4state$time
```

```
   user  system elapsed 
363.406   1.543 368.321 
```

In some cases, the MLEs for log-rate parameters may in fact be $-\infty$. In this case, the EM algorithm cuts off estimation at -16. We see that this is the case for one of the estimates.

4.2 EM for informative sampling time example

The EM algorithm call for the informative sampling model is

```
DDO_EM=EM(rates.setup=rates.setup,
   init.setup=init.setup,
   emission.setup=emission.setup,
   the.data=DDO_data,
   num.subjects=500,
   num.states=4,
   num.obs.states=3,
   tol = 1e-07,
   absorb.state=c(3,4),
   maxiter = 500,
   DDO.setup = DDO.setup)
```

The argument `absorb.state=c(3,4)` refers to the fact that latent states 3 and 4 are absorbing states in the model.

The output conveying the results of fitting the EM for this example are

```
> DDO_EM$LL
```

```
[1] -5418.382
```

```
> DDO_EM$param
```

```
         r           r           r           r           r           r 
 0.7466216  -1.0087643  -1.8307406  -4.4153603  -3.0527381   1.2191924 
         r           e           e           e           d           d 
 0.7881650  -4.6833853  -0.8288690 -50.0000000  -1.3725479   0.7249369 
```

```
> DDO_EM$time
```

```
   user  system elapsed 
 79.997   0.421  80.403 
```

5 Fitted intensity matrices, initial distributions, and emission matrices

After fitting the model, it may be desirable to obtain individual-specific estimates for transition intensity matrice, initial distributions and emission matrices. This may be useful for individual level predictions or simply to interpret model parameters/covariate effects on an intuitive scale.

We will use the panel data example as an illustration of the functions used for getting fitted values.

First we obtain a list of fitted rate matrices, via the function call

```
>  fit.4state$param[1:8]

          r           r           r           r           r           r
  0.7554409  -1.6249891   0.8439478  -2.9315493  -1.6741693 -16.0000000
          r           r
 -0.3557577  -0.2533592

>  rates.list=get.rate.matrix.list(current.params=fit.4state$param[1:8],
+ rate.setup=rates.setup, do.list=T)
>  rates.list[[1]]

              [,1]      [,2]       [,3]          [,4]
[1,] -1.629318e+00  1.491352  0.1379662  0.000000e+00
[2,]  0.000000e+00 -1.629365  1.6293645  0.000000e+00
[3,]  1.455074e-01  0.000000 -0.1868894  4.138203e-02
[4,]  8.734857e-08  0.000000  0.0000000 -8.734857e-08
```

The `current.params` argument is set to the MLEs for the rate parameters obtained from fitting the model. The first entry corresponds to the estimated rate matrix for the first individual in the sample.

Similarly, for the emission matrix,

```
>   emission.list=get.emission.matrix.list(current.params=fit.4state$param[9:11],
+                      emission.setup=emission.setup, num.states=4,
+                      num.obs.states=2,
+                      num.subjects=200, do.list = T)
> emission.list[[1]]

           [,1]       [,2]
[1,] 0.91802902 0.08197098
[2,] 0.91802902 0.08197098
[3,] 0.03996021 0.96003979
[4,] 0.03996021 0.96003979
```

Finally, to get fitted initial distributions

```
>   init.list=get.init.matrix.list(current.params  =fit.4state$param[12:13],
+                         init.setup=init.setup, do.list = T)
> init.list[[1]]

[1] 0.94342542 0.00000000 0.05657458 0.00000000
```

Again, the `param.values` argument is set to the MLEs for the intial distribution obtained from fitting the model.

6 Variance of parameter estimates

Running the EM only provides maximum likelihood estimates for model parameters. For inference, we will also need the estimated variance of the parameter estimates. We use a numeric estimation of the hessian of the observed data log-likelihood to estimate the variance.

The function call to get the variance for the panel data example is

```
covariance_4state=get_covariance(par=fit.4state$param,
the.data=the.data,
num.subjects=length(the.data),
num.states=4,
num.obs.states=2,
rates.setup=rates.setup,
emission.setup=emission.setup,
init.setup=init.setup,
DDO.setup=NULL,
do.DDO=F)
```

The estimated standard error for the parameters (rates, emission, and initial distribution, in order) is

```
> round(sqrt(diag(covariance_4state$covariance)),digits=3)

 [1]    0.684    3.324    0.510    0.870    0.361 2816.621    0.161    0.371
 [9]    3.191    3.006    0.427    0.570    0.369
```

The very high variance of the sixth parameter estimates corresponds to the log-rate parameter whose MLE was essentially $-\infty$.

For the informative sampling time example, the function call to get the variance is

```
covariance_DDO=get_covariance(par=DDO_EM$param,
            the.data=DDO_data,
            num.subjects=500,
            num.states=4,
            num.obs.states=3,
            rates.setup=rates.setup,
            emission.setup=emission.setup,
            init.setup=init.setup,
            DDO.setup=DDO.setup,
            do.DDO=T)
```

The estimated standard errors for the parameters (rates, emission, and initial distribution, DDO, in order) is

```
> round(sqrt(diag(covariance_DDO$covariance)),digits=3)

 [1] 0.308 0.281 0.294 0.258 0.112 0.133 0.107 0.223 0.038 0.000 0.042 0.019
```

We note that when parameters are fixed in the model (as was the case for one of the emission parameters) the estimated standard error will be zero.

7 Estimating CDFs and Hazard functions for sojourn time distributions

It will often be the case that functions of the latent parameters, such as hazard or cumulative distribution functions for sojourn times are of scientific interest. *Cthmm* provides functions for estimating these quantities and their point wise standard errors.

7.1 Estimating CDFs: panel data example

In our panel data example, the disease process can be described in terms of the distribution of sojourn times in states A and B. Our objective here will be to estimate the cumulative distribution functions for sojourn times for the covariate levels X=0 and X=1 and to plot the point estimates along with pointwise standard errors.

7.1.1 Transition intensity matrices for first passage times

The sojourn time in state A has the same distribution as time to absorption in a process that starts in A and treats B as an absorbing state. Thus in order to calculate the CDF of interest, we first need to construct the rate matrix for the process that treats B as an absorbing state.

To do so, we set the rows of the original matrix to zero for those states correspond to the latent states of B. In this case, these are states (and rows) 3 and 4. We will transform the estimated rate matrices for an individual with covariate value X=1, and one for X=0. In the dataset these correspond to, for example, individuals 1 and 200.

```
>   ####################################################################
>   #covariate X=0
> rate.firstpassage.AB_0=rates.list[[200]]
> rate.firstpassage.AB_0[3:4,]=0
> rate.firstpassage.AB_0

          [,1]     [,2]      [,3] [,4]
[1,] -2.325464  2.12855 0.1969138    0
[2,]  0.000000 -2.32553 2.3255296    0
[3,]  0.000000  0.00000 0.0000000    0
[4,]  0.000000  0.00000 0.0000000    0

> #covariate X=1
> rate.firstpassage.AB_1=rates.list[[1]]
> rate.firstpassage.AB_1[3:4,]=0
> rate.firstpassage.AB_1

          [,1]      [,2]      [,3] [,4]
[1,] -1.629318  1.491352 0.1379662    0
[2,]  0.000000 -1.629365 1.6293645    0
[3,]  0.000000  0.000000 0.0000000    0
[4,]  0.000000  0.000000 0.0000000    0

>
```

To obtain estimates of the CDF at different time points, we use the function `sub_dist_times`. The `start` and `end` arguments refer to the starting and ending times over which we will calculate the CDF. The `alpha` argument refers to the initial distribution (here we assume the individual starts in latent state 1). The `states` argument refers to the destination state (or states) we are tracking.

```
> cdfA.B_0=sub_dist_times(start=.01,end=10,states=3,
+                           alpha=c(1,0,0,0),rate.firstpassage.AB_0)
> cdfA.B_1=sub_dist_times(start=.01,end=10,states=3,
+                           alpha=c(1,0,0,0),rate.firstpassage.AB_1)
```

7.1.2 Standard errors of CDFs

We can obtain pointwise delta-method based standard errors using the covariance matrix for the parameters.

```
> covar=covariance_4state$covariance[1:8,1:8]
> num.transitions=dim(rates.setup$transition.codes)[1]
> num.params=8
> transitions=rates.setup$transition.codes
> num.states=4
> param.deriv=rates.setup$deriv.array[,,1]
>   se_AB_0=se.dist.times(start=.01,end=10,covar[1:8,1:8],rate.firstpassage.AB_0,
+                         states=3,alpha=c(1,0,0,0), param.deriv,num.transitions,
+                         num.params,transitions,
+                         num.states,length.out=1000)
>   se_AB_1=se.dist.times(start=.01,end=10,covar[1:8,1:8],rate.firstpassage.AB_1,
+                         states=3,alpha=c(1,0,0,0), param.deriv,num.transitions,
+                         num.params,transitions,
+                         num.states,length.out=1000)

> se_AB_1_high=se_AB_1+cdfA.B_1$dist
> se_AB_1_low=-se_AB_1+cdfA.B_1$dist
> se_AB_0_high=se_AB_0+cdfA.B_0$dist
> se_AB_0_low=-se_AB_0+cdfA.B_0$dist
> plot(cdfA.B_1$times,cdfA.B_1$dist,type="l",
+      xlab="sojourn time", ylab="probablility",col="blue",lty=1,lwd=2,
+      ylim=c(0,1),xlim=c(0,5), main="A to B")
> lines(cdfA.B_1$times,se_AB_1_low,type="l",lwd=2,col="lightblue",lty=2)
> lines(cdfA.B_1$times,se_AB_1_high,type="l",lwd=2,col="lightblue",lty=2)
> lines(cdfA.B_0$times,cdfA.B_0$dist,type="l",col="maroon",lty=1,lwd=2,
+       ylim=c(0,1),xlim=c(0,5))
> lines(cdfA.B_0$times,se_AB_0_low,type="l",lwd=2,col="pink",lty=2)
> lines(cdfA.B_0$times,se_AB_0_high,type="l",lwd=2,col="pink",lty=2)
> legend("bottomright",legend=c("X=0","X=1"),col=c("maroon", "blue"), lty=1, lwd=2)
```

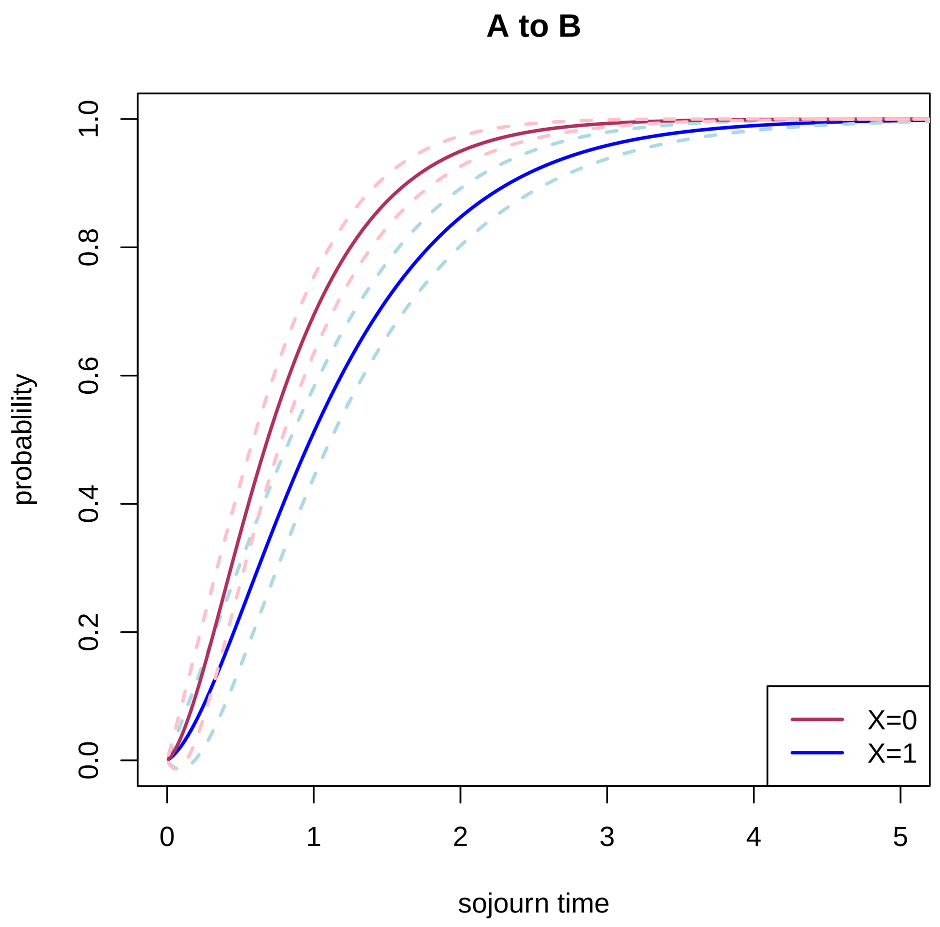


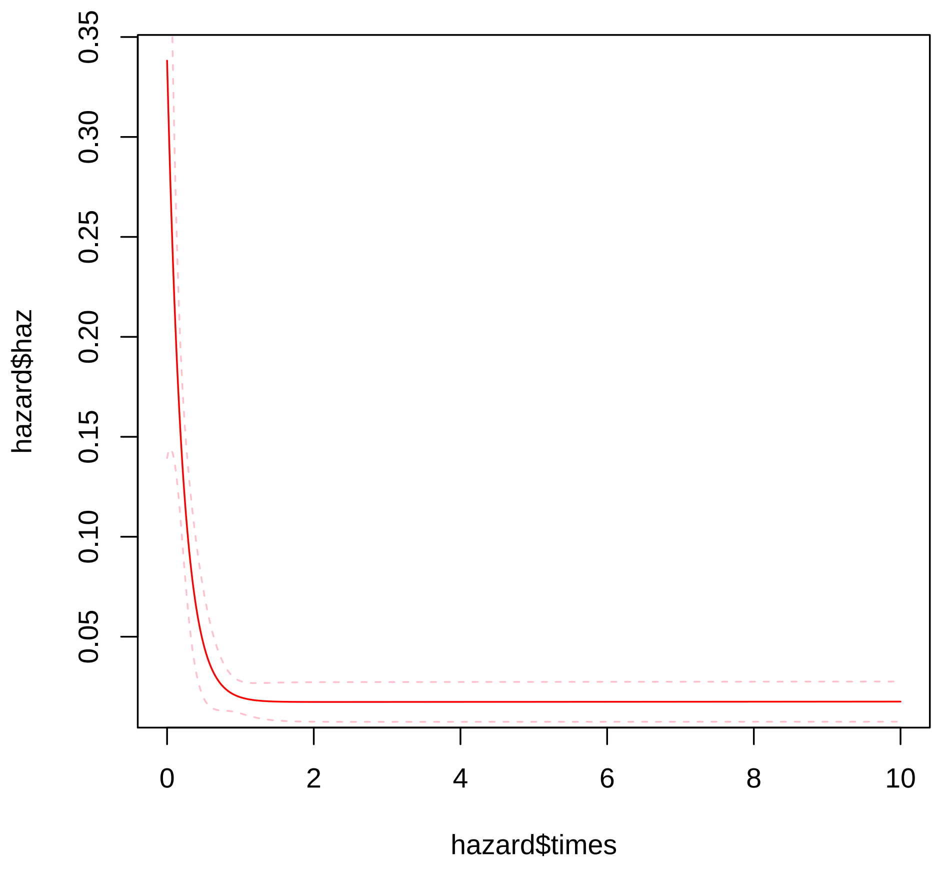
Figure 5: Estimated cumulative distribution function for sojourn times in state A for covariate levels X=0 and X=1, with standard errors (dotted lines)

7.2 Estimating hazard functions: informative sampling time example

Finally, we will estimate hazard functions for the informative sample time example for a selected individual.

```
> #get estimated rate matrices for each individual
> rates.list=get.rate.matrix.list(current.params=DDO_EM$param[1:7],
+ rate.setup=rates.setup, do.list=T)
```

```
> ################################################
> #plot hazard estimates for individual
> #################################
> hazard=hazard_times(start = 0, end=10, state_of_interest=c(3), at_risk_states=c(1,2),
+           alpha=c(1,0,0,0), rate=rates.list[[1]], length.out = 1000)
> se=se.haz.times(start=0, end=10,
+              covar=covariance_DDO$covariance[1:7,1:7],
+              rate=rates.list[[1]],
+              at_risk_states=c(1,2),
+              state_of_interest=3,
+              alpha=c(1,0,0,0),
+              param.deriv=rates.setup$deriv.array[,,1],
+              num.transitions=dim(rates.setup$transition.codes)[1],
+              num.params=7,
+              transitions=rates.setup$transition.codes,
+              num.states=4,
+              length.out = 1000)
> se.high=hazard$hazard+1.96*se
> se.low=hazard$hazard-1.96*se
> plot(hazard$times,hazard$haz,type="l",col="red")
> lines(hazard$times,se.high,lty=2,col="pink")
> lines(hazard$times,se.low, lty=2,col="pink")
```


hazard$haz
hazard$times