8-2011

Methods and software for nonparametric estimation in multistate models.

Amanda Nicole Ferguson
University of Louisville

METHODS AND SOFTWARE FOR NONPARAMETRIC ESTIMATION IN MULTISTATE MODELS

By

Amanda Nicole Ferguson
M.S., University of Georgia, 2004
B.S., University of Georgia, 2002

A Dissertation
Submitted to the Faculty of the
Graduate School of the University of Louisville
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy

Department of Physics
University of Louisville
Louisville, Kentucky

August 2011

METHODS AND SOFTWARE FOR NONPARAMETRIC ESTIMATION IN MULTISTATE MODELS

By

Amanda Nicole Ferguson
M.S., University of Georgia, 2004
B.S., University of Georgia, 2002

A Dissertation Approved on

7/12/11
Date

by the following Dissertation Committee:

Dissertation Director (Somnath Datta)

Co-Dissertation Director (Guy Brock)

Maiying Kong

Shesh Rai

Mahendra Sunkara

ACKNOWLEDGMENTS

I would like to thank my dissertation directors, Dr. Somnath Datta and Dr. Guy Brock, for their wisdom, direction, and support these past two years. They have been my mentors and set an amazing example of the type of professor I hope to be to my students in the future. Words cannot properly express how much I appreciate their investment in my life. I have been truly blessed by both of them.

I would like to thank my committee members, Dr. Maiying Kong, Dr. Shesh Rai, and Dr. Mahendra Sunkara, for their involvement and suggestions. Many thanks to all the faculty members in the Department of Bioinformatics and Biostatistics for their dedication to education and research. Their informative instruction greatly contributed to my academic growth and success. I want to thank the other students in the department for their friendship and encouragement these past four years.

Additional thanks go to Dr. Michael R. Marvin for permission to use the UNOS data and his assistance in understanding important clinical levels. This work was supported in part by Health Resources and Services Administration contract 234-2005-37011C. The content is the responsibility of the authors alone and does not necessarily reflect the views or policies of the Department of Health and Human Services, nor does mention of trade names, commercial products, or organizations imply endorsement by the U.S. Government.

This dissertation work represents four years of hard work, sacrifice, dedication, and perseverance. I would like to thank above all the Lord, for His protection, provision, and presence during my time in Louisville. I also want to thank my wonderful family for their amazing support. I could not have pursued this

dream without the love and support of my parents. They believed in me and offered an endless supply of encouragement and prayer that were invaluable to me. I love you! To my grandmother and uncle who both offered an incredible amount of support, I love you and I'm thankful for both of you. I want to thank my dear friend, Neal Heatherly, for his impact on my life these past 'almost' four years. I could not have made it without you. I want to also thank my friends who encouraged me through this season in my life. Finally, I'd like to thank my Louisville Bulldawg Family for adopting me these last four years and being my family away from home.

ABSTRACT

METHODS AND SOFTWARE FOR NONPARAMETRIC ESTIMATION IN MULTISTATE MODELS

Amanda Nicole Ferguson

July 12, 2011

Multistate models are a type of multi-variate survival data which provide a framework for describing a complex system where individuals transition through a series of distinct states. This research focuses on nonparametric inference for general multistate models with directed tree topology.

In this dissertation, we developed an R package, **msSurv**, which calculates the marginal stage occupation probabilities and stage entry and exit time distributions for a general, possibly non-Markov, multistage system under left-truncation and right censoring. Dependent censoring is handled via modeling the censoring hazard through observable covariates. Pointwise confidence intervals for the above mentioned quantities are obtained and returned for independent censoring from closed-form variance estimators and for dependent censoring using the bootstrap.

We also develop novel nonparametric estimators of state occupation probabilities, state entry time distributions and state exit time distributions for interval censored data using a combination of weighted isotonic regression and kernel smoothing with product limit estimation. Structural assumptions about the multistate system are avoided when possible. We evaluate the performance of our

estimators through simulation studies and real data analysis of a UNOS (United Network for Organ Sharing) data set.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

AN OVERVIEW OF NONPARAMETRIC ESTIMATION IN MULTISTATE MODELS

A Introduction

Multistate models are a type of multi-variate survival data which provide a framework for describing a complex system where individuals transition through a series of distinct states. This framework, which is often represented with a directed graph, illustrates the different states (or events) individuals may experience, as well as the possible transitions between states. Transitions between states may be reversible or irreversible while states can be either absorbing (meaning further transitions cannot occur) or transient. Multistate models have a range of applications including event history data, epidemiology, clinical trials where individuals progress through the different stages of a disease such as cancer and AIDS, and in systems engineering where a machine may experience various systems conditions with age.

Standard survival analysis models measure the time span from some time origin (e.g., birth) until the occurrence of the event of interest (e.g., death). This corresponds to the simplest multistate model, the two-state model with one transient root state (alive) and one absorbing state (dead). This could be expanded to include several absorbing states corresponding to different causes of death and is called the competing risk or multiple decrement model. Another simple example of a multistate model, which allows for a branching event, is the so called illness-death model. In this model, individuals start in the well state. Some individuals

subsequently move to the illness state and the rest of the individuals eventually experience death without ever visiting the illness state. In the irreversible version of the model all such individuals eventually move to the "dead" state without any possible recovery from the illness while in the reversible version, an individual in the illness state may recover and thus makes a transition back to the well state. All these simple models are represented by directed graphs or flowgraphs (Huzurbazar, 2005) in Figure 1. Multistate models can offer various degrees of complexities where individuals can pass through multiple transient states before entering a number of possible absorbing states.

There are several key questions which arise in studying multistate models. What is the probability that a subject is in a specific state j at a time t? What is the hazard (rate) at which a subject in a given state j at time t transitions to a future stage j'? What is the distribution of the time spent (waited) in a state j? More formally, these questions ask what are the state occupation probabilities, the state transition intensities (or transitional hazards), and the state waiting time distributions, respectively. Distribution functions for the state entry and exit times are also of interest. Estimators of these quantities have been proposed in the recent past under a variety of parametric and nonparametric assumptions as well as structural assumptions on the system (such as, progressive, Markov, semi-Markov etc.). In this paper we restrict ourselves to the nonparametric methods. Moreover, we concern ourselves with the estimation questions in a marginal model and not a conditional (e.g., regression) model. Thus, we do not discuss the semiparametric models in this paper. Generally speaking, results for the survival setup (e.g., a two state progressive model) are widely available in the literature and are not discussed in this dissertation.

In the standard survival analysis setting, especially with right censored data, the nonparametric likelihood type methods have been the usual choice. As for example, the classical Kaplan-Meier estimator for the survival function can be

obtained as a nonparametric maximum likelihood estimator. It is possible to apply this technique to certain multistate models such as a Markov or a semi-Markov which simplifies the likelihood formulation (Aalen, 1978; Aalen and Johansen, 1978; Frydman, 1992; Satten and Sternberg, 1999, etc). However, in absence of such additional structural assumptions the likelihood of an event may depend on all past events (state occupation) and event times. Thus, a likelihood approach in general is not feasible. In addition, there are additional challenges brought on by inherent incompleteness in the observed data due to various form of censoring. As we shall see, a combination of nonparametric functional estimation techniques, mostly various forms of averaging or smoothing are needed to form the estimators in multistate models.

The following general notations will be used throughout the paper. A multistate process is a stochastic process $\mathcal{S} = \{S(t) : t \geq 0\}$, where t denotes time and $S(t)$ denotes the state occupied at time t. We can think of $S(t-) = \lim_{s \to t-} S(s)$ as the state occupied just before time t. We assume a finite state space $\mathcal{X} = \{0, 1, \cdots, M\}$. Under the marginal model, we will assume that the multistate processes for n individuals $\mathcal{S}_i = \{S_i(t) : t \geq 0\}$, $1 \leq i \leq n$, are independent and identical (i.i.d., hereafter) realizations of $\mathcal{S}$.

For many applications, it is reasonable to assume that the system is progressive in which case the directed graph will have a tree structure and we will denote the root node by 0. For a given state j, $P_j(t) = Pr\{S(t) = j\}$ is the state occupation probability of state j as a function of time. In a multistate model representation of the standard survival analysis setup, we let state 0 = "alive" and state 1 = "dead". Then $P_0(t)$ is the survival function and $P_1(t)$ is the distribution function of the failure time. For simplicity of exposition, throughout the paper, we will assume that the process has at most one jump in an infinitesimal time interval $[t, t + dt)$ leading to (marginal) hazard rates of transitions from states j and j', $\alpha_{jj'}(t) = \lim_{dt \to 0} Pr\{S(s) = j', \text{ for some } s \in [t, t + dt) \,|S(t-) = j\}$, and integrated

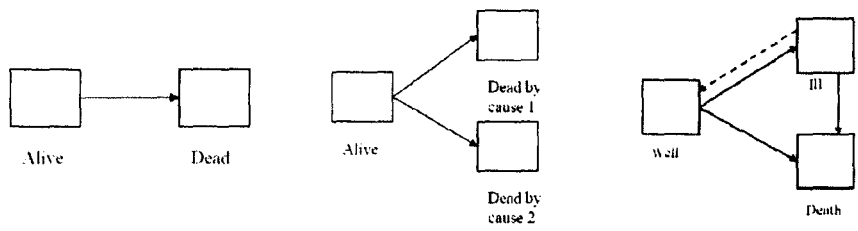

Figure 1. Flowgraphs of multistate models; (a) survival, (b) competing risk, (c) illness-death.

(or cumulative) hazard rates $A_{jj'}(t) = \int_0^t \alpha_{jj'}(s)ds$. Similarly, the (marginal) rates of entry to and exit out of state j are given by $\alpha_{\bullet j}(t) = \lim_{dt\to 0} Pr\{S(s) = j, \text{ for some } s \in [t, t+dt) \,|S(t-) \neq j\}$ and $\alpha_{j\bullet}(t) = \lim_{dt\to 0} Pr\{S(s) \neq j, \text{ for some } s \in [t, t+dt) \,|S(t-) = j\}$. For defining the state waiting times, we need to impose the restriction that a given state j can be entered at most once. For handling situations with repeated events, one would therefore add additional states to the system such as first entry, second entry and so on; this would mean that we can keep track of the occurrence of multiple entries to a given state. In this case, we can define the state entry, exit and waiting (sojourn) times by $U^j = \inf\{t : S(t) = j\}$ and $V^j = \sup\{t : t > U_j,\ S(t) \neq j\}$, $W^j = V^j - U^j$, when $U^j < \infty$. Note that by convention, $U^j = \infty$, if state j is never entered and $V^j = \infty$, if either state j is never entered or j is an absorbing state (in which case it is never left). The (marginal) state entry, exit and waiting time distributions will be denoted by $F^j(t) = Pr\{U^j \leq t | U_j < \infty\}$, $G^j(t) = Pr\{V^j \leq t | V_j < \infty\}$ and $H^j(t) = Pr\{W^j \leq t | V_j < \infty\}$, respectively.

The rest of the paper is organized as follows. The next section of the paper introduces various estimation methodologies to handle right censored multistate data. Right censoring is perhaps the prevalent form of censoring in time to event studies. Section C considers more severe forms of censoring when individuals are not constantly monitored. The paper ends with a discussion section (Section D).

B Estimation Under Right Censoring

There are a number of reasons why right censoring is often, if not always, present in time to event data including multistate models. Generally, studies have a finite duration and the event of interest may not take place during the study interval leading to right censoring of the event. More generally, in a multistate model framework, an individual may still be at a transient state at the end of the study or follow-up time which means there are potential future transitions whose exact times will be unknown. Mathematically speaking, a multistate process $\mathcal{S}$ that is right censored by a censoring variable C is given by the stochastic process, $\mathcal{S}^c = \{S(t \wedge C) : t \geq 0\}$. Basically, it means that we observe all the transition times and the state occupation up to time C and nothing beyond that. Thus, the right censored data will be i.i.d. realizations of $\mathcal{S}^c$ given by $\mathcal{S}_1^c, \cdots, \mathcal{S}_n^c$ together with the censoring times $C_1, \cdots, C_n$. The most common assumption on the censoring times is that they are i.i.d. and are independent of the original multistate processes $\mathcal{S}_1, \cdots, \mathcal{S}_n$. This is the so called "random censoring" assumption and will be assumed for subsection 1 and 2.

1 Nelson-Aalen Estimators

The Nelson-Aalen estimators (Aalen, 1978; Andersen *et al.*, 1993) are obtained on the basis of rate calculations. Using the independent censoring assumption, one can establish that the observed rates of transitions between states in a censored experiment is the same as that in an uncensored experiment. The former rate can be empirically estimated based on available data and it leads to the Nelson-Aalen estimators of integrated (marginal) transition hazards. More formally, let for states j and j', $N_{jj'}$ and $N_{jj'}^c$ be counting processes with jumps given by $\Delta N_{jj'}(t) = \sum_{i=1}^n I\{S_i(t-) = j,\, S_i(t) = j'\}$ and $\Delta N_{jj'}^c(t) = \sum_{i=1}^n I\{S_i(t-) = j,\, S_i(t) = j',\, C_i \geq t\}$, respectively, recording the transition counts from states j to j' in the uncensored and censored experiments,

respectively. Also, let $Y_j(t) = \sum_{i=1}^n I(S_i(t-) = j)$ and $Y_j^c(t) = \sum_{i=1}^n I(S_i(t-) = j, C_i \geq t)$ be the number of individuals at state j just before time t in the uncensored and censored experiments, respectively. Then as, $n \to \infty$, the two instantaneous rates $dN_{jj'}(t)/Y_j(t)$ and $dN_{jj'}^c(t)/Y_j^c(t)$converges (in probability) to $\alpha_{jj'}(t)dt$ and $\alpha_{jj'}^c(t)dt$, respectively, where $\alpha_{jj'}$ is defined earlier and

$$\begin{aligned}
\alpha_{jj'}^c(t) &= \lim_{dt\to 0} \frac{Pr\{S(s) = j', \text{ for some } s \in [t, t+dt), C \geq s\}}{Pr\{S(t-) = j, C \geq t\}} \\
&= \lim_{dt\to 0} \frac{Pr\{S(s) = j', \text{ for some } s \in [t, t+dt), C \geq t\}}{Pr\{S(t-) = j, C \geq t\}} \\
&= \lim_{dt\to 0} \frac{Pr\{S(s) = j', \text{ for some } s \in [t, t+dt)\}Pr\{C \geq t\}}{Pr\{S(t-) = j\}Pr\{C \geq t\}},
\end{aligned}$$

using independence of S and C. The last expression, however equals to $\alpha_{jj'}(t)$. In other words, the two hazard rates $\alpha_{jj'}(t)$ and $\alpha_{jj'}^c(t)$ are equal at all time points t. Therefore the integrated hazard rate $A_{jj'}(t)$ in the marginal model can be estimated by the integrated empirical hazard rate from the right censored multistate data leading to the Nelson-Aalen estimator

$$\widehat{A}_{jj'}(t) = \int_0^t I(Y_j^c(s) > 0) \frac{dN_{jj'}^c(s)}{Y_j^c(s)}. \tag{1}$$

Since this estimator is a step function, in order to obtain a legitimate estimator of the hazard rate $\alpha_{jj'}$, one needs to apply kernel smoothing to it. To that end, let K be a symmetric kernel (e.g., a symmetric density function) and let $0 < h = h(n) \downarrow 0$ be a bandwidth sequence. Then a non-parametric estimator of the marginal hazard rate of transition from state j to j' is given by

$$\widehat{\alpha}_{jj'}(t) = \frac{1}{h} \int K(\frac{t-s}{h}) d\widehat{A}_{jj'}(s).$$

2 Aalen-Johansen Estimators

For a Markov multistate process, the transition probabilities $p_{jj'}(s,t) = Pr\{S(t) = j' | S(s) = j\}$ can be computed by product integration of the

marginal hazard function $\boldsymbol{P}(s,t) = \prod_{(s,t]}(I + d\boldsymbol{A}(u))$, where $\boldsymbol{P}(s,t)$ is a matrix with (j,j')th entry $p_{jj'}(s,t)$ and $\boldsymbol{A}$ is a matrix with (j,j')th entry $A_{jj'}$, if $j' \neq j$, and $= -\sum_{k \neq j} A_{jk}$, if $j' = j$. This leads to the construction of Aalen-Johansen estimator (Aalen and Johansen 1978) of transition probabilities of a Markov multistate model obtained by substituting the Nelson-Aalen estimators of $\boldsymbol{A}$ into this formula

$$\widehat{\boldsymbol{P}}(s,t) = \prod_{(s,t]}(I + d\widehat{\boldsymbol{A}}(u)). \tag{2}$$

For multistate models with only one transient state, such as classical survival analysis and the competing risk model, the assumption of Markovity holds trivially and thus the Aalen-Johansen estimators are valid. In particular, for the survival setting it is just the Kaplan-Meier estimator. As mentioned earlier, the Aalen-Johansen estimator can also be obtained as a non-parametric maximum likelihood estimator under the Markov assumption. Valid estimators for the three state progressive non-Markov illness-death model are proposed by Meira-Machado *et al.* (2006). Nonparametric estimators of transition probabilities for general multistate models without the Markovity assumption are not currently available.

One can set the initial time $s = 0$ in the Aalen-Johansen estimator and combine it with the initial state occupation to obtain the following natural estimators of state occupation probabilities $p_j(t) = Pr\{S(t) = j\}$,

$$\widehat{p}_j(t) = n^{-1} \sum_{k=0}^{M} \widehat{P}_{kj}(0,t) Y_k(0+). \tag{3}$$

Interestingly, Datta and Satten (2001) noted that this estimator remains valid (e.g., consistent) even without the Markov assumption; also, see Glidden (2002) for a different proof of the same result. In other words, the Markov assumption which is often unverified but routinely assumed is not really needed if one is only interested in estimation of state occupation probabilities as a function of time. Unfortunately, this fact still remains relatively unknown amongst practitioners even till date.

3 Datta-Satten Estimators

Datta and Satten (2002) extended the Nelson-Aalen and Aalen-Johansen estimators to situations where the censoring random variable is not necessarily independent of the multistate process but rather only conditionally independent given an observable time varying covariate $\boldsymbol{Z} = \{Z(t) : t \geq 0\}$. In their treatment, they estimate the two processes $N_{jj'}$ and Y_j separately using the principle of inverse probability of censoring weights (Koul *et al.*, 1981; Robins and Rotnitzky, 1992; Robins, 1993; Satten *et al.*, 2001) rather than their ratio. The estimates are not equal to the censored data versions defined earlier; however, under the model of independent censoring these estimated processes are proportional to the respective censored data processes defined before and so the Datta-Satten estimators under the independent censoring hazard assumption reduce to Nelson-Aalen and Aalen-Johansen estimators. State occupation probability estimators in an illness-death model using a different reweighting scheme to handle a specific type of dependent censoring was considered in Datta *et al.* (2000b) but the present treatment is more general.

In general, to construct these estimators, a model for the censoring hazard $\lambda_c(t|\overline{Z}(t)) = \lim_{dt\to 0} Pr\{C_i \in [t, t+dt) | C_i \geq t, Z(s), 0 \leq s \leq t, \mathcal{S}\} = \lim_{dt\to 0} Pr\{C_i \in [t, t+dt) | C_i \geq t, Z(s), 0 \leq s \leq t\}$ given the time dependent covariates Z is needed to obtain an estimate $\widehat{K}(t) = \exp\{-\widehat{\Lambda}_c(t|\overline{Z}(t)\}$. In particular, Datta and Satten advocated the use of Aalen's linear hazards model (Aalen, 1980) for this purpose. Using the reweighting principle (see Datta and Satten (2002), for a formal argument) one can construct the following estimators of the complete data counting and at risk processes

$$\Delta\widehat{N}_{jj'}(t) = \sum_{i=1}^{n} \frac{I\{S_i(t-) = j,\, S_i(t) = j',\, C_i \geq t\}}{\widehat{K}_i(t-)} \tag{4}$$

and

$$\widehat{Y}_j(t) = Y_j(t) = \sum_{i=1}^{n} \frac{I(S_i(t-) = j, C_i \geq t)}{\widehat{K}_i(t-)}. \tag{5}$$

Substituting these expressions (4-5) into the formula (1) in places of $\widehat{N}^c_{jj'}$ and $\widehat{Y}^c_j$, we obtain the Datta-Satten estimators of integrated (marginal) transition hazards. Using the Datta-Satten estimator of $\widehat{\boldsymbol{A}}$ and the at risk set $\widehat{Y}_j$ in formulas (2) and (3), we in turn get the Datta-Satten estimators of transition probabilities (for Markov systems) and state occupation probabilities (for possibly non-Markov systems) under dependent censoring. See Cook *et al.* (2009) for an application of the Datta-Satten estimator to bone cancer data.

For some applications, state entry and exit time distribution functions are of interest. The estimators of state occupation probabilities constructed above can be used to estimate these distributions by state pooling as follows. For this purpose, we assume that the model can be expanded into a progressive tree-like structure with a root node 0 so that each state can be entered and exited at most once. For cyclic models (such as a reversible illness-death and recurrent events data), each entry of a given state needs to be interpreted as a new state. After the state occupation probabilities of these expanded system are calculated they can be pooled (e.g., summed) to obtain estimators in the original system.

Let $\mathcal{S}^j$ denote the collection of all states $j' \neq j$ such that state j appears on the path connecting states 0 and j'. In other words, $\mathcal{S}^j$ is the collection of all states which proceeds state j. Then estimators of entry and exit time distributions of state j are given by

$$\widehat{F}^j(t) = \frac{\sum_{k\in\{j\}\cup\mathcal{S}^j} \widehat{p}_k(t)}{\sum_{k\in\{j\}\cup\mathcal{S}^j} \widehat{p}_k(\infty)} \quad \text{and} \quad \widehat{G}^j(t) = \frac{\sum_{k\in\mathcal{S}^j} \widehat{p}_k(t)}{\sum_{k\in\mathcal{S}^j} \widehat{p}_k(\infty)}. \tag{6}$$

We end this subsection with an introduction of the Satten and Datta (2002) estimators of state waiting time distributions that are valid without the Markovity assumption. Furthermore, these estimators use reweighting based on the censoring

hazard and thus available covariate information can be incorporated that might be related to the censoring mechanism. The form of the reweighting reflects the fact that waiting time distributions are measured since state entry and not in calendar times. Once again, assume that a transient state j can be entered at most once.

Let $\widehat{K}_i(t) = \exp\{-\widehat{\Lambda}_c(t|\overline{Z}_i(t)\}$ be as before. Then, the estimated counting processes for waiting times in a give state j is a jump process with jump size equal to

$$\Delta\widehat{N}_j^W(t) = \sum_{i=1}^{n} \frac{I\{W_i^j = t, C_i \geq V_i^j\}}{\widehat{K}_i(V_i^j -)}$$

which can be computed based on the available right censored data since if $C_i \geq V_i^j$ then the state j waiting time W_i^j is available. The inverse weighting factor is essentially the estimated conditional probability of the event $\{C_i \geq V_i^j\}$, given $\{V_i^j, W_i^j\}$. Next, the size of the "at risk" set of state j waiting times is estimated by

$$\widehat{Y}_j^W(t) = \sum_{i=1}^{n} \frac{I\{W_i^j \geq t, C_i \geq t + U_i^j\}}{\widehat{K}_i((t + U_i^j)-)}$$

Note that, once again, this quantity can be computed based on the available data and in particular, even if the exit time is right censored. Finally, a nonparametric estimator of state j waiting time distribution is obtained by a Kaplan-Meier type product limit formula using these two sets

$$\widehat{H}^j(t) = 1 - \prod_{s \leq t} \left(1 - \frac{d\widehat{N}_j^W(ds)}{\widehat{Y}_j^W(s)}\right).$$

These estimators are valid even when the censoring is not independent. Other versions of nonparametric estimators of state waiting times for certain types of multistate models under independent censoring assumption were obtained in Wang and Wells (1998) and Wang (2003).

4 The Pepe Estimator and Its Extensions

Pepe (1991) suggested using the difference of two Kaplan-Meier estimators to estimate the state occupation probability of a transient state in a four-state leukemia progression model. Another nonparametric estimator of the state occupation probability was proposed by Datta *et al.* (2000a) and involved using a "fractional size at risk set" and a reweighting approach in the three-state irreversible illness-death model. The fractional weights representing the probabilities of traversing a future path in a more general multistate model with a tree structure were considered by Datta and Satten (2000). These weights can be combined with right censored entry and exit times to calculate marginal estimators of state entry and exit time distributions. A Pepe type subtraction estimator can also be constructed using these in a general multistate model with a tree structure.

Suppose we have a progressive model that can be expanded into a rooted directed tree with the root node 0. Let $N^c_{\bullet j}$ and $N^c_{j\bullet}$ be the counting processes of observed entry and exits to state j with jumps given by

$$\Delta N^c_{\bullet j}(t) = \sum_{i=1}^{n} I(S_i(t) = j,\, S_i(t-) = j_*, C_i \geq t) \tag{7}$$

and

$$\Delta N^c_{j\bullet}(t) = \sum_{i=1}^{n} I(S_i(t) \in \mathcal{S}^j,\, S_i(t-) = j,\, C_i \geq t), \tag{8}$$

where j_* is the state that precedes j in the path from 0 to j. The corresponding "numbers at risk" processes at time t in the censored experiment are given by

$$Y^c_{\bullet j}(t) = \sum_{i=1}^{n} I(C_i \geq t,\, S_i(t-) \in \{j \cup S^j\}^c,\, S_i(u) = j, \text{ for some } u \geq 0)$$

and

$$Y^c_{j\bullet}(t) = \sum_{i=1}^{n} I(C_i \geq t,\, S_i(t-) \in \{S^j\}^c,\, S_i(u) = j, \text{ for some } u \geq 0).$$

However, these later two processes cannot be evaluated from the observed data if certain individuals are censored at a state, say $S(C)$ from which eventual passage through state j is possible but not guaranteed. Such individuals should contribute a fractional count ϕ_j to the "at risk sets" which represents their probability of passing through state j in the future had there been no censoring. This idea leads to the following "fractional at risk sets":

$$\widehat{Y}^c_{\bullet j}(t) = \sum_{i=1}^{n} \widehat{\phi}_{ij}\, I(C_i \geq t,\, S_i(t-) \in \{j \cup S^j\}^c) \tag{9}$$

and

$$\widehat{Y}^c_{j\bullet}(t) = \sum_{i=1}^{n} \widehat{\phi}_{ij} I(C_i \geq t,\, S_i(t-) \in \{S^j\}^c) \tag{10}$$

that can be computed from the available data. Here $\widehat{\phi}_{ij}$ is an estimate of $Pr\{S_i(u) = j, \text{for some } u \geq 0 \,|\, C_i, S^c_i\}$. These fractional weights are recursively calculated based on the distance of the state $S(C_i)$ from the root node 0 (Datta and Satten, 2000).

For the time being, we drop the index i to keep the notation simple. First consider the case when $S(C) = 0$ and j can be reached from 0 in one step. Let $N^c_{0\bullet}$ be the counting process of transitions out of state 0 defined as above. Then, $\widehat{\phi}_j$ can be calculated using the Aalen-Johansen transition probability estimates in a competing risk model

$$\widehat{\phi}_j = \int_{(C,\infty)} \left\{ \prod_{(C,u)} \left(1 - \frac{dN^c_{0\bullet}(u)}{\widehat{Y}^c_{0\bullet}(u)} \right) \right\} \frac{dN^c_{0j}(u)}{\widehat{Y}^c_{0\bullet}(u)}. \tag{11}$$

since one can view $\widehat{\phi}_j$ as an eventual occupation probability $\widehat{P}_{0,j}(C, \infty)$ of stage j in a collapsed network (where all future states beyond j are equated with j and so on).

Next, let $S(C) = k$ $(\neq 0)$ and j can be reached from 0 in two steps with k as the intermediate step. Observe that $\widehat{\phi}_k$ can be calculated by the above formula (11), so $\widehat{Y}^c_{k\bullet}(t)$ is now well defined. Now, define $\widehat{\phi}_j$ by the above formula (11) with 0's replaced by k throughout.

Finally, for the general case when j can be reached from 0 in m steps with $S(C) = k$ on the path to j. Let $k = j_1 \to j_2 \to \cdots \to j_{m'} = j$ be the path from k to j for some $m' \leq m$. Assume that by induction, we have calculated $\widehat{\phi}_{\tilde{j}}$ whenever 0 and $\tilde{j}$ are separated by less than m stepS. Note that in this case $\widehat{Y}_{j_l\bullet}$, for $l < m'$, are all well defined and hence are

$$\widehat{\psi}_l = \int_{(C,\infty)} \left\{ \prod_{(C,u)} \left(1 - \frac{dN^c_{j_l\bullet}(u)}{\widehat{Y}^c_{j_l\bullet}(u)} \right) \right\} \frac{dN^c_{j_l j_{l+1}}(u)}{\widehat{Y}^c_{j_l\bullet}(u)}$$

for $l = 1, \cdots, m' - 1$. Finally, let $\widehat{\phi}_j = \prod_{l=1}^{m'-1} \widehat{\psi}_l$.

The counting and fractional size at risk processes given by (7-10) can be used to compute alternative estimators of the state entry and exit time distributions using the product-limit formulas

$$\widehat{F}^j(t) = \frac{\prod_{s \leq t} \left(1 - \frac{dN^c_{\bullet j}(s)}{\widehat{Y}^c_{\bullet j}(s)} \right)}{\prod_{s \geq 0} \left(1 - \frac{dN^c_{\bullet j}(s)}{\widehat{Y}^c_{\bullet j}(s)} \right)} \quad \text{and} \quad \widehat{G}^j(t) = \frac{\prod_{s \leq t} \left(1 - \frac{dN^c_{j\bullet}(s)}{\widehat{Y}^c_{j\bullet}(s)} \right)}{\prod_{s \geq 0} \left(1 - \frac{dN^c_{j\bullet}(s)}{\widehat{Y}^c_{j\bullet}(s)} \right)}. \tag{12}$$

Unlike (6), these estimators are guaranteed to be monotonic. These estimators, in turn, can be combined to obtain an estimator of the state occupation probabilities which are extensions of Pepe's (1991) estimators to more complex multistate models

$$\widehat{p}_j(t) = (n^{-1} \sum_{i=1}^{n} \widehat{\phi}_{ij}) \{ \widehat{F}_j(t) - \widehat{G}_j(t) \}. \tag{13}$$

However, since these are based on a subtraction formula, unlike the Aalen-Johansen (or the Datta-Satten) estimators (3), these estimators may sometimes assume negative values which is not desirable of probability estimates.

Martingale representations of all the estimators reviewed in this section are available even though these could be quite complex, especially, when dependent censoring is present. Bootstrap resampling is an attractive alternative to large sample calculations for these estimators leading to variance estimates and pointwise confidence intervals.

C Estimation Under Current Status Data

Marginal nonparametric estimation for multistate current status data was undertaken in Datta and Sundaram (2006), Datta *et al.* (2009), and Lan and Datta (2010b); the special case of competing risk models was investigated by Jewell *et al.* (2003) and Groeneboom *et al.* (2008).

As before, for an individual i and a time $t \geq 0$, $S_i(t)$ denotes the state individual i is in at time t; C_i denotes the random time at which the individual i gets inspected. The censoring times and the state occupation processes $\{C_i, S_i(t),\ t \geq 0\}$ for the individuals are assumed to be independent and identically distributed. For simplicity of development, we will make the assumption of random censoring, which means C_i is independent of $\mathcal{S}_i = \{S_i(t) : t \geq 0\}$. We further assume that all transition and censoring times are continuous and that the allowable transitions give rise to a rooted directed tree structure, in which every state $j \in \mathcal{S}$ can be reached from an initial state 0 (the root node) by a unique path $\pi(j) : 0 = s_1 \to s_2 \cdots \to s_{j+1} = j$. The observed data consist of $\{C_i, S_i(C_i)\}$ for $i = 1, \cdots, n$.

1 Estimators of State Occupation Probabilities

Consider two states j and j'. Let $U^{jj'}$ denote the (unobserved) transition time of an individual from state j to j' (define it to be ∞, if this transition is not made by the individual). Let $N_{jj'}(t)$ denote the usual counting process counting the number of j to j' transitions in $[0, t]$ with the complete data. By the laws of large

numbers,

$$n^{-1}N_{jj'}(t) \xrightarrow{P} n^{-1}EN_{jj'}(t) = P\{U^{jj'} \leq t\} = n_{jj'}(t), \text{ say.}$$

Consider the indicator function $I\left(U^{jj'} \leq C\right)$ of the event that the j to j' transition has taken place by time C. Then, for any $t \geq 0$,

$$E(I\left(U^{jj'} \leq C\right)|C = t) = Pr\{U^{jj'} \leq t\}.$$

Therefore, $n^{-1}\widehat{N}_{jj'}(\cdot)$ can be obtained by a nonparametric regression estimator of $I\left(U^{jj'} \leq C\right)$ given C. Since $Pr\{U^{jj'} \leq t\}$ is monotonic in t, $n^{-1}\widehat{N}_{jj'}(\cdot)$ can be constructed by an isotonic regression of $I\left(U^{jj'} \leq C\right)$ on C, based on the pairs $\left(C_i, I(U_i^{jj'} \leq C_i)\right)$.

Next, note that $P_j(t-) = Pr\{S(t-) = j\}$ is the (in probability) limit of $n^{-1}Y_j(t)$, where $Y_j(t)$ denotes the size of the "at risk" set of transitions out of state j with the complete data. However, unlike the counting process of transition counts, the Y_j process does not have to be monotonic for a transient state j. Therefore, one can use kernel smoothing rather than isotonic regression to estimate this process leading to

$$\widehat{Y}_j(t) = \frac{\sum_{i=1}^{n} I(S_i(C_i) = j)K_h(C_i - t)}{n^{-1}\sum_{i=1}^{n} K_h(C_i - t)},$$

where K is a density kernel, $h = h(n)$ is a bandwidth sequence, and $K_h(\cdot) = h^{-1}K(\cdot/h)$.

With the above estimators in place, the class of state occupation probabilities will be computed as in Section B using the relationship (3),where $\widehat{P}(0,t) = \prod_{(0,t]}(I + d\widehat{\boldsymbol{A}}(u))$. However, the integrated conditional transition hazards are now calculated using $\widehat{N}$ and $\widehat{Y}$ defined in this subsection.

$$\widehat{A}_{jj'}(t) = \begin{cases} \int_0^t J_j(u)\widehat{Y}_j(u)^{-1} d\widehat{N}_{jj'}(u) & j \neq j' \\ -\sum_{j' \neq j} \widehat{A}_{jj'}(t) & j = j', \end{cases}$$

where $J_j(u) = I(\widehat{Y}_j(u) > 0)$.

2 State Entry and Exit Time Distributions

A similar approach as in Section 3 can be followed to obtain these. However, for current status data, the basic ingredients, namely, the counting processes of transition counts and the size of the at risk sets in and out of a given state j are computed using a different machinery.

Since the indicators $I\left(U_i^j \leq C_i\right)$ and $I(V_i^j \leq C_i)$ are calculable from the available current status information (along with the topological knowledge of the system) one could regress them (say, by isotonic regression) to obtain $\widehat{N}_{\bullet j}(\cdot)$ and $\widehat{N}_{j\bullet}(\cdot)$. More precisely, $n^{-1}\widehat{N}_{j\bullet}(\cdot)$ is a step function for taking values $n^{-1}\widehat{N}_{j\bullet}(C_{(i)})$ $= R_i$, say, that minimizes the sum of squares $\sum_{i=1}^n \{R_i - I(U_{[i]}^j \leq C_{(i)})\}^2$ subject to $R_1 \leq \cdots \leq R_n$, where $[i]$ denotes the index corresponding to the ith largest C; $n^{-1}\widehat{N}_{\bullet j}(\cdot)$ is computed the same way with U's replaced by V's. These can be obtained using the well known pooled adjacent violator algorithm (Barlow *et al.*, 1972).

The "size at risk" sets will be computed by antitonic regression but with fractional weights representing the probability of ever making it to state j. Thus, $n^{-1}\widehat{Y}_{\bullet j}(\cdot)$ is a step function taking values $n^{-1}\widehat{Y}_{\bullet j}(C_{(i)}) = R_i$, say, that minimize the sum of squares $\sum_{i=1}^n \{R_i - \widehat{\phi}_{[i]j} I(U_{[i],j} \geq C_{(i)})\}^2$ subject to $R_1 \geq \cdots \geq R_n$; $n^{-1}\widehat{Y}_{j\bullet}(\cdot)$ is computed the same way with U's replaced by V's.

The fractional weights are successively (recursively) calculated from the root node to the distant states as before $\widehat{\phi}_j = \prod_l \widehat{\psi}_l$, where

$$\widehat{\psi}_l = \int_{(C,\infty)} \left\{ \prod_{(C,u)} \left(1 - \frac{d\widehat{N}_{j_l \bullet}(v)}{\widehat{Y}_{j_l \bullet}(v)} \right) \right\} \frac{d\widehat{N}_{j_l j_{l+1}}(u)}{\widehat{Y}_{j_l \bullet}(u)}.$$

In the above formula, $\widehat{N}_{j_l j_{l+1}}$ are calculated by isotonic regression of the pairs $(C_i, I\left(U_i^j \leq C_i\right))$, $1 \leq i \leq n$. Estimators of F^j, G^j and the Pepe type alternative estimator of p_j can be formed using these processes by formulas ((12)) and ((13)) where we use $\widehat{N}_{\bullet j}$ and $\widehat{N}_{j \bullet}$ instead of $N^c_{\bullet j}$ and $N^c_{j \bullet}$, respectively; similarly, $\widehat{Y}_{\bullet j}$ and $\widehat{Y}_{j \bullet}$ are used in places of $\widehat{Y}^c_{\bullet j}$ and $\widehat{Y}^c_{j \bullet}$, respectively.

3 State Waiting Time Distributions

Calculation of state waiting time distributions with current status data poses additional difficulty since we cannot directly regress the indicators of events involving the waiting times because the state entry times are also unknown. Some progress can be made with additional structural assumptions. As for example, under the Markov assumption (Datta et al, 2009), we could obtain the following identity

$$H^j(t) = 1 - \int_0^\infty \prod_{u<s\leq u+t} \left(1 - d\Lambda_{j\bullet}(s)\right) dF^j(u),\ t \geq 0,$$

where $\Lambda_{j\bullet}$ is integrate transition hazard out of state j. Using this and the quantities defined earlier we obtain a non-parametric regression estimator of the state waiting time survival function

$$H^j(t) = 1 - \int_0^\infty \left\{ \prod_{u<s\leq u+t} \left(1 - \frac{d\widehat{N}_{j\bullet}(s)}{\widehat{Y}_j(s)} \right) \right\} d\widehat{F}^j(u),\ t \geq 0.$$

D Discussion

Generally speaking, while parametric (and semiparametric) methods produce relatively precise inference for various model characteristics and the effects of

covariates under the correct model, their performance under incorrect model assumptions is questionable. This is one compelling reason why a fully nonparametric approach is preferable even though such a formulation is often difficult with time to event data. A large sample size may be necessary to derive the full utility of nonparametric methods; in addition, in dealing with time to event data, one faces additional difficulty and loss of information due to various forms of censoring. The situation with multi-state models that generalize the traditional survival setup is even more challenging. Nevertheless, only nonparametric answers represent truly empirical (or evidence based) calculations. They can at least serve as a guideline to the shape of the various marginal aspects of the system even if a semiparametric or parametric calculation is ultimately performed. Doksum and Yandell (1982) made similar points with compelling comparative illustrations of nonparametric calculations versus semiparametric calculations using the well known Stanford heart transplant data. We hope that this paper serves as an overview of nonparametric approaches to study certain marginal temporal characteristics of a broad class of multi-state models. There is scope of future work in these areas including bivariate estimation such as that of transition probabilities without the Markov assumption and the joint distribution estimation of two waiting times. Estimation of related functionals such as measures of association are of interest too. Estimation of sojourn time distribution under current status and intervals censored data in non-Markov models remain an open problem as well.

CHAPTER II

msSurv, AN R PACKAGE FOR NONPARAMETRIC ESTIMATION OF MULTISTATE MODELS

A Introduction

Multistate models are systems of multivariate survival data where individuals transition through a series of distinct states following certain paths of possible transitions. These systems are illustrated by a directed graph, where distinct states are treated as nodes and possible transitions are considered directed edges. Transitions between states may be reversible or irreversible while states can be either absorbing or transient. Multistate models have a wide range of application including epidemiology, dentistry, clinical trials, reliability studies in engineering, and medicine where individuals progress through the different states of a disease such as cancer and AIDS. Data in these applications are often subject to right censoring and possibly left truncation.

Aalen (1978, see also Nelson, 1972) proposed an estimator for the integrated hazard under a broad class of counting process models. Aalen and Johansen (1978) obtained an estimator for the transition probability matrix and subsequently state occupation probabilities through product limit integration of the Nelson-Aalen estimator. Datta and Satten (2001) established that the resulting estimators of state occupation probabilities remained valid even when the process is non-Markovian. Datta and Satten (2002) also proposed an estimator for state occupation probabilities that can handle state dependent censoring and other flexible models through a weighting function based on the censoring scheme.

Estimation of state entry and exit distribution functions are also of interest (Pepe, 1991; Datta and Ferguson, 2011), and can be calculated through normalized sums of state occupation probabilities.

Several R packages are available on the Comprehensive R Archive Network (CRAN, http://www.r-project.org) for use with multistate models. Recently, *The Journal of Statistical Software* published a special volume on Competing Risks and Multistate Models featuring papers on the **msm**, **mstate**, **etm**, and **3state.msm** packages (Jackson, 2011; de Wreede *et al.*, 2011; Allignol *et al.*, 2011; Meira-Machado and Roca-Pardinas, 2011). Other packages currently available include **changeLOS** and **mvna**. The **msm** package provides functions for fitting multistate Markov models to panel count data and offers extensions to hidden Markov multistate models and possibly inhomogeneous Markov models (Jackson, 2011). The **mstate** package can be applied to right censored and left truncated data in semiparametric or nonparamertric multistate models with or without covariates and it may also be applied to competing risk models. The package offers functions which calculate transition probabilities and standard errors. It also uses Cox regression models to estimate different types of covariate effects (de Wreede *et al.*, 2011). The packages **changeLOS**, **mvna**, and **etm** all provide methods for nonparametric estimation in multistate models. The most specialized package available is **changeLOS**, which is based on methods described in Schulgen and Schumacher (1996) and computes changes in length of hospital stay (Wrangler *et al.*, 2006). It does offer a function to compute the Aalen-Johansen estimator, but it does not provide variance estimates and cannot be applied to left truncated data. The **mvna** package computes the Nelson-Aalen estimator of the cumulative transition hazard for any multistate model with right censored, left truncated data, but does not compute transition probability matrices. The **etm** package calculates the transition probability matrices and corresponding variance estimates for any finite-state multistate model (Allignol *et al.*, 2011) with data subject to right

censoring and left truncation. State occupation probabilities may be indirectly found using an initial time of 0 in the **etm** package. No packages currently available estimate state entry and exit time distributions, nor do they estimate desired quantities like transition probabilities for dependent censoring. These missing methods are addressed in the **msSurv** package available from CRAN which we introduce in this paper. **msSurv** calculates nonparametric estimates of marginal quantities for time to event multistate data subject to right censoring and possibly left truncation. We assume left truncation occurs with respect to the total time of an individual in the multistate system, so that individuals are not considered for the estimation of transitions prior to their left truncation time. The main function `msSurv()` calculates and returns the marginal state occupation probabilities and state entry and exit time distributions for a general, possibly non-Markov, multistate system, and provides additional features not currently found in other R packages. The function also calculates and returns the marginal integrated transition hazards and the hazard rate functions, and for a Markov model, the transition probability matrix between any two times. Users specify whether the censoring is independent or state dependent and then appropriate calculations of variance and confidence intervals are performed. Pointwise confidence intervals for the above mentioned quantities are obtained and returned for independent censoring from closed-form variance estimators and from bootstrapping for dependent censoring. The function returns an object of S4 class `msSurv` which includes state and transition information, event times, estimates, variance estimates, confidence intervals, and counting process information. The `msSurv` object can then be summarized or plotted using methods from the package.

The rest of this chapter is organized as follows. Section B describes nonparametric estimation methods which are used in the **msSurv** package. Section C describes the implementation of **msSurv** and illustrates available functions through examples. Possible future extensions of our software package are discussed in D.

B The estimators

Consider a multistate model with a finite state space $\mathcal{S} = \{1, \ldots, M\}$ and set of possible transitions between the states in $\mathcal{S}$. The quantities of interest (like state occupation probabilities and state entry and exit distributions) can be calculated for complete data, when available, and also using estimators obtained from censored data when complete data is not known. It is useful to keep track of all the transitions an individual makes before ending in an absorbing state. Let T_{ik}^* represent the time of the kth transition for individual i ($= \infty$ if the ith individual enters the absorbing state before the kth transition is made), where $T_{i0}^* = 0$. Let C_i be the right censoring time for the ith individual, L_i be the left truncation time for the ith individual, and s_{ik} be the state occupied by the ith individual between times T_{ik-1}^* and T_{ik}^*. Let $T_i^* = sup_k \{T_{ik}^* : T_{ik}^* < \infty\}$ be the time for the last transition for individual i. The collection of all transition times and states occupied by individual i can be denoted as $\boldsymbol{T}_i^* = (T_{ik}^* : k \geq 1)$ and $\boldsymbol{s}_i^* = (s_{ik} : k \geq 1)$, respectively. Define an indicator of whether the ith individual was never censored, e.g., $\delta_i = I(C_i > T_i^*)$, and let $T_i = min(T_i^*, C_i)$.

The censoring hazard is independent of the multistate system when

$$\lim_{dt \to 0} \frac{Pr\left(C_i \epsilon [t, t+dt), \delta_i = 0 | T_i \geq t > L_i, \boldsymbol{T}_i^*, s_{ik}^*\right)}{dt} = \lim_{dt \to 0} \frac{Pr\left(C_i \epsilon [t, t+dt), \delta_i = 0 | T_i \geq t > L_i\right)}{dt}$$

(Datta and Satten, 2001, 2002). There are times when censoring and hazards of future transitions are affected by time varying covariates $Z = Z(t)$. In these cases, the covariate variables Z explain the dependence and thus future transitions and censoring events behave conditionally independent given the covariate process Z (Datta and Satten, 2002). Modeling of this censoring hazard using the covariate process is discussed in Section 3.

1 Nelson-Aalen and Aalen-Johansen Estimators

Andersen *et al.* (1993) presented formulas for the Nelson-Aalen estimator for

the integrated hazard matrix $\boldsymbol{A}$ and the Aalen-Johansen estimator of the state occupation probability matrix of a Markov system. The counting process and the number at risk for data subject to left truncation and right censoring are estimated as

$$\widehat{N}_{jj'}(t) = \sum_{i=1}^{n} \sum_{k \geq 1} I\left(T_{ik}^{*} \leq t, C_i \geq T_{ik}^{*}, L_i < t, s_{ik} = j, s_{ik+1} = j'\right) \tag{12}$$

and

$$\widehat{Y}_j(t) = \sum_{i=1}^{n} \sum_{k \geq 1} I\left(T_{ik-1}^{*} < t \leq T_{ik}^{*}, C_i \geq t, L_i < t, s_{ik} = j\right). \tag{13}$$

The Nelson-Aalen estimator of the cumulative hazard is given by

$$\widehat{A}_{jj'}(t) = \begin{cases} \int_0^t \frac{I\left(\widehat{Y}_j(s) > 0\right)}{\widehat{Y}_j(s)} d\widehat{N}_{jj'}(s) & j \neq j' \\ -\sum_{j \neq j'} \widehat{A}_{jj'}(t) & j = j', \end{cases} \tag{14}$$

The Aalen-Johansen estimator of a transition probabilities matrix of a Markov multistate system is obtained by product integration of $\widehat{A}_{jj'}(t)$, i.e.,

$$\widehat{P}(s,t) = \prod_{(s,t]} \left(I + d\widehat{\boldsymbol{A}}(u)\right), \tag{15}$$

where $\widehat{\boldsymbol{A}} = \left\{\widehat{\boldsymbol{A}}_{jj'}\right\}$ which reduces to simple empirical proportions for the complete data.

A recursive formula for computing the variance of transition probability can be found in Andersen *et al.* (1993)(see formula 4.4.19 on p. 295 for details). The resulting estimator is of the Greenwood-type.

2 State occupation probabilities

The state occupation probability answers the marginal question: "What is the probability that an individual is in state j at time t?" Let $p_j(t) = Pr\{s(t) = j\}$ denote the state occupation probabilities where $s(t)$ is the state occupied by an

individual at time t, $j \in \{1, ..., M\}$. For incomplete data, the state occupation probabilities are estimated as

$$\widehat{p}_j(t) = \sum_{k=1}^{M} \widehat{p}_k(0)\, \widehat{p}_{kj}(0,t), \tag{16}$$

where $\widehat{p}_{jk}(0,t)$ is the kjth element of the matrix $\widehat{P}(0,t) = \prod_{(0,t)} \left(I + d\widehat{A}(u)\right)$ and $\widehat{p}_k(0)$ is the initial state occupation proportions for state k. This estimator is in fact the Aalen-Johansen estimator of state occupation and holds its validity regardless of Markovian assumptions (Datta and Satten, 2001). Estimation of state occupation probabilities can be extended to dependent censoring and other flexible models by explicitly modeling the censoring process (Datta and Satten, 2000, 2002).

3 Datta-Satten estimators

Datta and Satten (2002) use the principle of inverse probability of censoring weights (Robins and Rotnitzky, 1992) to extend the Nelson-Aalen and Aalen-Johansen estimators to data subject to dependent censoring. The resulting Datta-Satten estimators use a weighting function, denoted by K, which is based on a fitted model of the censoring hazards using Aalen's linear hazards model with possibly time-dependent covariates Z (Aalen, 1980).

Reweighted estimators for the counting process and the size of the at risk sets of complete data are defined as

$$\widehat{N}_{jj'}(t) = \sum_{i=1}^{n} \sum_{k \geq 1} I\left(T_{ik}^{*} \leq t, C_i \geq T_{ik}^{*}, L_i < t, s_{ik} = j, s_{ik+1} = j'\right) / K_i\left(T_{ik}^{*}-\right) \tag{17}$$

and

$$\widehat{Y}_j(t) = \sum_{i=1}^{n} \sum_{k \geq 1} I\left(T_{ik-1}^{*} < t \leq T_{ik}^{*},\, C_i \geq t,\, L_i < t, s_{ik} = j\right) / K_i(t-) \tag{18}$$

where $\widehat{K}(t) = exp\left\{-\widehat{\Lambda}_C\left(t|\boldsymbol{Z}(t)\right)\right\}$ with

$\lambda_C(t|\boldsymbol{Z}(t)) = \lim_{dt\to 0} \frac{Pr\left\{C_i \epsilon [t,dt), \delta_i | T_i \geq t > L_i, \boldsymbol{Z}_i(t), \boldsymbol{T}_i^*, \boldsymbol{s}_i^*\right\}}{dt}$, and $\boldsymbol{Z}_i(t) = \{Z_i(s) : 0 \leq s < t\}$. See Datta and Satten (2002) for a formal argument.

For state dependent right censoring (i.e., when $Z(t) = s(t)$), this is equivalent to estimating the censoring hazard by a state specific Nelson-Aalen estimator of censoring; see Datta and Satten (2002) for details.

Substituting the formulas for the estimated counting process and the number at risk into equations 14, 15, and 16 yields the Datta-Satten estimators of integrated transition hazards, transition probabilities for Markov Systems and state occupation probabilities for non-Markov systems under dependent censoring. Variance estimates for the Datta-Satten estimators of transition probabilities are obtained using the bootstrap.

4 State entry and exit distributions

An important application of state occupation probabilities is computing the state entry and exit time distribution functions. We assume an acyclic system. Suppose X_j denotes the possibly unobserved indicator of an individual ever entering state j. Suppose F_j and G_j denote the state entry and exit time distribution functions, respectively, for the individuals who ever enter state j (i.e., $X_j = 1$). Let $\mathcal{S}^j$ denote the collection of all states which come after state j in the progressive model. The entry time distribution to state j is estimated by taking the normalized sum of the estimated state occupation probabilities of state j and all the other states that come after j in the system, i.e.,

$$\widehat{F}_j(t) = \frac{\sum_{k\epsilon \mathcal{S}^j \cup j} \widehat{p}_k(t)}{\sum_{k\epsilon \mathcal{S}^j \cup j} \widehat{p}_k(\infty)},$$

where $\widehat{p}_k(\infty) = \lim_{t\to\infty} \widehat{p}_k(t)$.

The exit time distribution from a transient state j is estimated by taking the normalized sum of estimated state occupation probabilities of all states that come after state j in the progressive system, i.e.,

$$\widehat{G}_j(t) = \frac{\sum_{k \epsilon \delta^j} \widehat{p}_k(t)}{\sum_{k \epsilon \delta^j} \widehat{p}_k(\infty)}.$$

Variance for state entry and exit time distributions are obtained using the bootstrap.

5 Confidence intervals

Pointwise confidence intervals for estimators of transition probabilities and state occupation probabilities follow methods described in Andersen *et al.* (1993). Let $\widehat{P}(s,t)$ be the transition probability between two states in the system (the subscripts are omitted to simply the notation) between times s and t and let $\widehat{\sigma}(s,t)$ be the corresponding variance estimate. Then the linear confidence interval for $\widehat{P}(s,t)$ is defined as

$$\widehat{P}(s,t) \pm c_{\alpha/2}\widehat{\sigma}(s,t),$$

where $c_{(\alpha/2)}$ is the upper $\alpha/2$ percentile of the standard normal distribution. It may be beneficial to consider transformations to improve estimation especially in the case of small sample sizes (Bie *et al.*, 1987; Thomas and Grunkemeier, 1975). Borgan and Liestol (1990) suggested a log transformation to improve small sample properties. The resulting formula for the confidence interval is

$$\widehat{P}(s,t)\, exp\left\{\frac{\pm c_{\alpha/2}\widehat{\sigma}(s,t)}{\widehat{P}(s,t)}\right\},$$

Other transformations to improve small sample properties include the log-log transformation, proposed by Kalbfleisch and Prentice (1980) and defined as

$$\widehat{P}(s,t)^{exp\left\{\frac{\pm c_{\alpha/2}\widehat{\sigma}(s,t)}{\widehat{P}(s,t)log(\widehat{P}(s,t))}\right\}},$$

and the complementary log-log transformation

$$1-\left(1-\widehat{P}(s,t)\right)^{exp\left\{\frac{\pm c_{\alpha/2}\widehat{\sigma}(s,t)}{(1-\widehat{P}(s,t))log(1-\widehat{P}(s,t))}\right\}}.$$

Confidence intervals for state occupation probabilities are calculated using $s = 0$ in the formulas above.

Confidence intervals for state entry and exit time distributions for state i at time t are calculated using $\widehat{F}_i(t)$ and $\widehat{G}_i(t)$ instead of $\widehat{P}(s,t)$, with corresponding variance estimate $\widehat{\sigma}(t)$ obtained using the bootstrap.

C msSurv package implementation

The **msSurv** package may be applied to any general multistate model with data subject to right censoring and possibly left truncation. **msSurv** is written entirely in the R programming language using S4 classes and methods, and is available for download from CRAN (http://www.r-project.org). It can be installed on all operating systems for which the R software is installed. Other package dependencies include the **graph** package (Gentleman *et al.*, 2010) and the **lattice** package (Sarkar, 2008).

The main function of the package, `msSurv()`, calculates the counting processes and risk sets according to both Andersen *et al.* (1993) and Datta and Satten (2001), as well as the state occupation probabilities and transition probabilities described in the previous section. The **msSurv** package contains a function to calculate the transition probabilities between two specific times (`Pst`), a function to display the state occupation probabilities at a specific time t (`st.t`), a function to display the state entry and exit time distributions at a specific time t (`EntryExit`), as well as `print`, `plot`,and `summary` methods for `msSurv` objects.

We will illustrate the application of the **msSurv** package through 3 examples. The first example uses simulated data with independent right censoring, the second example uses a simulated data set with left truncation and independent right censoring, and the third example uses a bone marrow transplant data set from Klein and Moeschberger (1997) with state dependent right censoring.

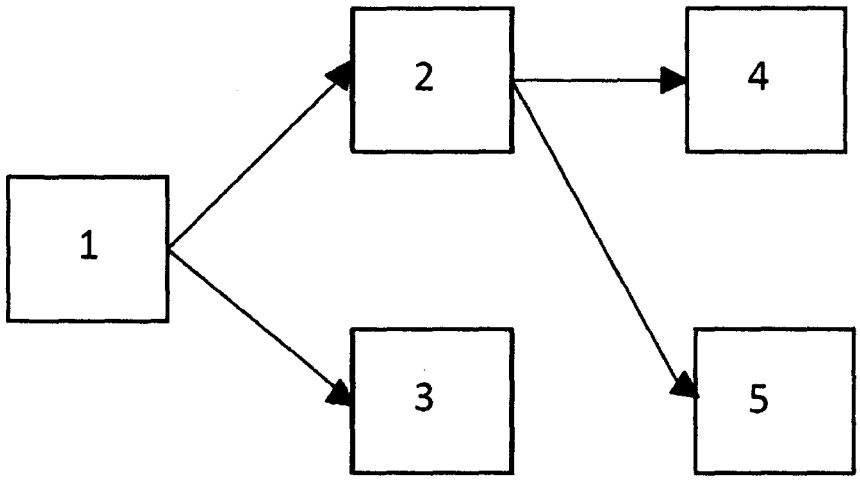

Figure 2. A five state model for simulated multistate data.

1 A 5 state example

We consider a five-state progressive model with the tree structure illustrated in Figure 2. We simulated a data set of 1000 individuals subject to independent right censoring with 60% of individuals starting in state 1 at time 0 and 40% starting in state 2. Those in state 1 remain there until they transition to the transient state 2 or the terminal state 3. Individuals in state 2 remain there until they transition to either terminal state 4 or 5.

A right censoring time is generated for each of the 1000 individuals using the log normal distribution with log mean -0.5 and log standard deviation 2. For individuals starting in state 1, two times are generated using the Weibull distribution using a sample size of 600 and shape parameter of 2 to reflect transition times between states 1 and 2 and states 1 and 3. Times are compared and the minimum time is kept as the event time and the corresponding state is recorded. Then, two additional times are generated to reflect transitions between states 2 and 4 and states 2 and 5. These times are generated using the formula $T_2 = D^{-1}\left(D\left(T_1\right) + R_2\left\{1 - D\left(T_1\right)\right\}\right)$ where T_1 is the first transition time, R_2 is a random number generated from $U\left(0,1\right)$ independent of T_1, $D\left(\cdot\right)$ denotes the

distribution function for the Weibull distribution with shape parameter 2, $D^{-1}(\cdot)$ denotes the corresponding quantile function. The minimum of these two times and the censoring times are compared and the minimum time is taken as the event time and the corresponding state is recorded. For individuals starting in state 2, two times are generated using the Weibull distribution with a sample size of 400 and shape parameter of 2 to reflect transitions between states 2 and 4 and states 2 and 5. These times are then compared with the corresponding censoring times and the minimum time is taken as the event time and the state information is recorded. All times were rounded to the fourth decimal place for clarity of presentation. The simulated data is available as `RCdata` in the package.

We begin by loading the package.

```
R> library("msSurv")
```

Now we load the data.

```
R> data("RCdata")
```

Data should be in a data frame with column names "id", "stop", "st.stage", and "stage" where "id" is the individual's identification number, "stop" is the transition time from state j to j', "st.stage" is the state the individual is transitioning from (i.e., j), and "stage" is the state the individual is transitioning to (i.e., j') and equals 0 if right censored.

```
R> RCdata[70:76,]
```

```
    id   stop st.stage stage
57  57 0.3086        1     3
58  58 0.5322        1     2
614 58 0.6333        2     5
59  59 0.3330        1     2
```

```
615 59 0.5824        2     5
60  60 0.7722        1     0
61  61 0.4096        1     3
```

Now we specify the tree structure for this multistate system. First, input the states in the multistate system as a list or character vector of the state names. Then, store the transition information as a list of possible states with allowed transitions being lists of edges. For terminal states, the lists of edges will be `NULL`. Nodes correspond to the states in the model and edges refer to the allowed transitions.

```
R> Nodes <- c("1","2","3","4","5")
R> Edges <- list("1"=list(edges=c("2","3")),
+                "2"=list(edges=c("4","5")),
+                "3"=list(edges=NULL),
+                "4"=list(edges=NULL),
+                "5"=list(edges=NULL))
```

The tree structure is then specified using the **graph** package (Gentleman *et al.*, 2010) by creating a `graphNEL` object as below with nodes and edges defined above.

```
R> treeobj <- new("graphNEL",nodes=Nodes,edgeL=Edges,
+             edgemode="directed")
```

Now we will call the `msSurv` function to perform nonparametric estimation for this simulated example.

```
R> ex1 <- msSurv(RCdata,treeobj)
```

Results of the analysis can be viewed using the `print`, `plot`, and `summary` methods, as well as the `Pst`, `st.t` and `EntryExit` functions, available for the

msSurv object ex1. The print method will be discussed in this section while the summary and plot methods will be discussed in examples 2 and 3, respectively.

The print method gives an overview of the model, specifying the number of transient and absorbing states and identifying the states and possible transitions in the system. It also provides the state occupation probabilities, state entry time distributions, state exit time distributions, and transition probability matrix $P(0,t)$ for the largest event time in the data set.

```
R> print(ex1)

The specified multistate model has 2 transient state(s) and
 3 absorbing state(s)

Possible States in this Model:
[1] "1" "2" "3" "4" "5"

Possible Transitions for this Model:
[1] "1 2" "1 3" "2 4" "2 5"

State Occupation Information at time 1.7345:

Estimates of State Occupation Probabilities
   p 1    p 2    p 3    p 4    p 5
0.0000 0.0000 0.2802 0.3784 0.3414

Estimates of State Entry Time Distribution
F 1 F 2 F 3 F 4 F 5
 NA   1   1   1   1

Estimates of State Exit Time Distribution
```

```
G 1 G 2 G 3 G 4 G 5
  1   1  NA  NA  NA
```

```
Transition Probability Information:
```

```
Estimate of P(0,1.7345)
    cols
rows 1 2     3      4      5
   1 0 0 0.467 0.2935 0.2395
   2 0 0 0.000 0.5056 0.4944
   3 0 0 1.000 0.0000 0.0000
   4 0 0 0.000 1.0000 0.0000
   5 0 0 0.000 0.0000 1.0000
```

Variance estimates are omitted by default, but users may specify `covar=TRUE` and variance estimates will be provided for each estimated quantity. For example,

```
R> print(ex1,covar=TRUE)
```

The **msSurv** package contains 2 functions for the user to easily access estimators of transition and state occupation probabilities at specific time points. The package also contains a function for the user to access state entry and exit time distribution estimators at specific time points.

The transition probabilities between any two times s and t are computed using the `Pst` function. The `Pst` function takes an `msSurv` object, a starting time s, and an ending time t and prints the transition probability matrix $P(s,t)$. For example, to find the transition probability matrix P(1,3.1), we would enter

```
R> Pst(ex1,s=1,t=3.1)
```

```
Estimate of P(1,3.1)
   cols
    1 2      3     4      5
  1 0 0 0.4805 0.307 0.2124
  2 0 0 0.0000 0.586 0.4140
  3 0 0 1.0000 0.000 0.0000
  4 0 0 0.0000 1.000 0.0000
  5 0 0 0.0000 0.000 1.0000
```

The corresponding covariance matrix for this transition probability is printed by adding the argument `covar=TRUE`, i.e.,:

```
R> Pst(ex1,s=1,t=3.1,covar=TRUE)
```

State occupation probabilities at a specific time t are given using the `st.t` function. This function takes a `msSurv` object as well as time t as arguments. The default time t is the maximum event time in the data set. Individuals may start in any state in the system at time 0. The function prints the state occupation probabilities for all states in the system at time t. For example, call the function `st.t` to find the state occupation probabilities at time `t=0.85`.

```
R> st.t(ex1,t=0.85)

The state occupation probabilities at time 0.85 are:
State 1: 0.1415
State 2: 0.2179
State 3: 0.2127
State 4: 0.2028
State 5: 0.2252
```

The corresponding variance estimates are provided using the argument `covar=TRUE`. Variance estimates are found by evaluating the formula in Andersen

et al. (1993) at $\widehat{P}(0,t)$ for data where all the individuals start in the initial state at time 0. The bootstrap is used to find variance estimates of state occupation for data where individuals start in different states of the system.

```
R> st.t(ex1,t=0.85,covar=TRUE)
```

The `EntryExit` function in **msSurv** displays the state entry and exit time distributions at a specific time t. This function takes a `msSurv` object and time t as arguments and displays the state entry distributions for non-initial states and state exit distributions for non-terminal states. Estimates are rounded to four decimal places by default, but the user may specify a different number through the `deci` argument. For example, the state entry and exit distributions at time 1 are displayed by

```
R> EntryExit(ex1,t=1)

The state entry distributions at time 1 are:
State 2: 0.9587
State 3: 0.9018
State 4: 0.7172
State 5: 0.7794
State entry distributions for state  1 is omitted
since there are no transitions into that state.


The state exit distributions at time 1 are:
State 1: 0.9427
State 2: 0.7467
State exit distributions for state(s)  3 4 5 is (are) omitted
since there are no transitions into that (those) state(s).
```

To display the bootstrap variance estimates for state entry and exit time distributions, the user would need to include the `d.var=TRUE` argument in the call of the `msSurv` function. Then, include the `covar=TRUE` argument in the call of the `EntryExit` function.

```
R> ex1a <- msSurv(data,treeobj,d.var=TRUE)
R> EntryExit(ex1a,t=1,covar=TRUE)
```

2 Left truncation and right censoring example

We consider an irreversible three-state illness-death model with data subject to independent right censoring and left truncation (Andersen *et al.*, 1993). We simulated a data set of 1000 individuals starting in state 1 at time 0. Individuals remain in state 1 until they transition to the transient state 2 (ill) or the terminal state 3 (death). Individuals in state 2 remain there until they transition to the terminal state 3 (death).

Two times are generated using the Weibull distribution with a sample size of 1000 and shape parameter of 2 to reflect transition times for either illness or death. Right censoring and left truncation times are generated using the log normal distribution with mean -0.5 and standard deviation 2 on the log scale. We assume 20% of individuals have a left truncation time of 0. Only individuals whose left truncation times were less than the terminal event times or censored event times were kept. The left truncation time was taken to be the time the individual entered the study. Individuals were not included in the at risk set before their left truncation time. Times for these individuals were compared and the minimum time was kept as the event time and the corresponding state was recorded. Then, another time was generated reflecting the transition between states 2 and 3 using the formula $T_2 = D^{-1}\left(D\left(T_1\right) + R_2\left\{1 - D\left(T_1\right)\right\}\right)$ where T_1 is the first transition time, R_2 is a random number generated from $U(0,1)$ independent of T_1, $D\left(\cdot\right)$ denotes the distribution function for the Weibull distribution with shape parameter 2, $D^{-1}\left(\cdot\right)$

denotes the corresponding quantile function. These "death" times are then compared to the censoring times and the minimum time is kept as the event time and the corresponding state information is recorded. All times were rounded to the fourth decimal place for clarity of presentation. The simulated data is available as `LTRCdata` in the package.

Begin by loading the data

```
R> data("LTRCdata")
```

Data should be in a data frame with column names "id", "start", "stop", "st.stage", and "stage" where "id" is the individual's identification number, "start" is the start time for the period of observation after the individual enters state j (and is the left truncation time for the first observed transition), "stop" is the transition time from state j to j', "st.stage" is the state the individual is transitioning from (i.e., j), and "stage" is the state the individual is transitioning to (and equals 0 if right censored).

```
R> LTRCdata[489:494,]
```

```
     id  start   stop st.stage stage
468 468 0.0000 0.9229        1     3
3     3 0.5851 0.9231        1     3
65   65 0.5944 0.9237        1     3
534 222 0.7367 0.9239        2     3
547 262 0.0886 0.9305        2     0
47   47 0.5488 0.9313        1     2
```

Now we specify the tree structure.

```
R> Nodes <- c("1","2","3")
R> Edges <- list("1"=list(edges=c("2","3")),
```

```
+                  "2"=list(edges=c("3")),
+                  "3"=list(edges=NULL))
R> treeobj2 <- new("graphNEL",nodes=Nodes,edgeL=Edges,
+              edgemode="directed")
```

The **msSurv** function is called to perform nonparametric estimation for this simulated example. Since the data is subject to left truncation, we must add the argument `LT=TRUE` to the call of **msSurv**.

```
R> ex2 <- msSurv(LTRCdata,treeobj2,LT=TRUE)
```

We assume individuals subject to left truncation start in state 1 at time 0 unless otherwise specified. The user may enter a vector of starting states for each individual in the data through the **start.states** argument.

Results of the analysis will be illustrated through the **summary** method available for the **msSurv** object **ex2**. The **summary** method displays information for both state occupation probabilities and transition probabilities. Estimates of state occupation probability are displayed with corresponding variance estimates, confidence intervals (denoted "lower.ci" and "upper.ci" in the output), state entry time distributions ("entry.d") and state exit time distributions ("exit.d") are shown for each state in the system. The default settings give these estimates to three decimal places for key percentile event times (minimum, maximum, 25th percentile, median, and 75th percentile). The **summary** method also provides summary information for each allowed transition in the system. Estimates of the transition probabilities are given with estimates of variance, confidence intervals ("lower.ci and "upper.ci"), the risk sets calculated according to Andersen *et al.* (1993) ("n.risk") and Datta and Satten (2001) ("n.risk.K") at the key percentile event times mentioned above. For transitions from one state to a different state, both counting processes ("n.event" for Andersen *et al.* (1993) and "n.event.K" for Datta and Satten (2001)) are displayed. For transitions into the same state, the number remaining at risk are displayed ("n.event" and "n.event.K").

```
R> summary(ex2,digits=2)

State Occupation Information:

State  1
 time estimate variance lower.ci upper.ci entry.d exit.d
 0.07     1.00  1.5e-05     0.99     1.00      NA 0.0038
 0.42     0.72  7.0e-04     0.67     0.77      NA 0.2811
 0.66     0.45  8.8e-04     0.39     0.51      NA 0.5479
 0.92     0.20  5.8e-04     0.15     0.24      NA 0.8048
 2.01     0.00  0.0e+00     0.00     0.00      NA 1.0000

State  2
 time estimate variance lower.ci upper.ci entry.d exit.d
 0.07   0.0038  1.5e-05    0.000    0.011  0.0038   0.00
 0.42   0.1175  3.5e-04    0.081    0.154  0.2811   0.18
 0.66   0.1992  5.6e-04    0.153    0.245  0.5479   0.38
 0.92   0.2480  6.8e-04    0.197    0.299  0.8048   0.60
 2.01   0.0787  7.5e-04    0.025    0.132  1.0000   1.00

State  3
 time estimate variance lower.ci upper.ci entry.d exit.d
 0.07     0.00  0.00000     0.00     0.00    0.00     NA
 0.42     0.16  0.00047     0.12     0.21    0.18     NA
 0.66     0.35  0.00079     0.29     0.40    0.38     NA
 0.92     0.56  0.00090     0.50     0.62    0.60     NA
 2.01     0.92  0.00076     0.87     0.98    1.00     NA

Transition Probability Information:
```

Transition 1 -> 1

time	estimate	variance	lower.ci	upper.ci	n.risk	n.remain	n.risk.K	n.remain.K
0.07	1.00	1.5e-05	0.99	1.00	261	260	304	303
0.42	0.72	7.0e-04	0.67	0.77	210	210	361	361
0.66	0.45	8.8e-04	0.39	0.51	125	124	270	267
0.92	0.20	5.8e-04	0.15	0.24	52	52	135	135
2.01	0.00	0.0e+00	0.00	0.00	0	0	0	0

Transition 1 -> 2

time	estimate	variance	lower.ci	upper.ci	n.risk	n.event	n.risk.K	n.event.K
0.07	0.0038	1.5e-05	0.000	0.011	261	1	304	1.2
0.42	0.1175	3.5e-04	0.081	0.154	210	0	361	0.0
0.66	0.1992	5.6e-04	0.153	0.245	125	0	270	0.0
0.92	0.2480	6.8e-04	0.197	0.299	52	0	135	0.0
2.01	0.0787	7.5e-04	0.025	0.132	0	0	0	0.0

Transition 1 -> 3

time	estimate	variance	lower.ci	upper.ci	n.risk	n.event	n.risk.K	n.event.K
0.07	0.00	0.00000	0.00	0.00	261	0	304	0.0
0.42	0.16	0.00047	0.12	0.21	210	0	361	0.0
0.66	0.35	0.00079	0.29	0.40	125	1	270	2.2
0.92	0.56	0.00090	0.50	0.62	52	0	135	0.0
2.01	0.92	0.00076	0.87	0.98	0	0	0	0.0

Transition 2 -> 2

time	estimate	variance	lower.ci	upper.ci	n.risk	n.remain	n.risk.K	n.remain.K
0.07	1.000	0.0000	1.0000	1.00	0	0	0	0

0.42	0.554	0.0279	0.2269	0.88	37	36	64	62
0.66	0.412	0.0165	0.1601	0.66	58	58	125	125
0.92	0.299	0.0091	0.1117	0.49	63	62	163	161
2.01	0.068	0.0010	0.0058	0.13	5	4	70	56

```
Transition 2 -> 3
```

time	estimate	variance	lower.ci	upper.ci	n.risk	n.event	n.risk.K	n.event.K
0.07	0.00	0.0000	0.00	0.00	0	0	0	0.0
0.42	0.45	0.0279	0.12	0.77	37	1	64	1.7
0.66	0.59	0.0165	0.34	0.84	58	0	125	0.0
0.92	0.70	0.0091	0.51	0.89	63	1	163	2.6
2.01	0.93	0.0010	0.87	0.99	5	1	70	14.0

Confidence intervals provided by default are 95% linear confidence intervals, but the user may change the confidence level to either 90% or 99% by changing the `ci.level` argument or apply a transformation of "log", "cloglog" or "log-log" by changing the `ci.trans` argument. The user may change the number of significant digits through the `digits` argument.

Summary information for all event times in the data set are displayed using the `all=TRUE` argument.

```
R> summary(ex2,all=TRUE)
```

The user also has the option to display information about only the state occupation probabilities (by inserting the argument `trans.pr=FALSE`) or information only about transition probabilities (`stateocc=FALSE`).

3 An example of state dependent censoring

State dependent censoring in **msSurv** will be illustrated using data on 136 cancer patients who received bone marrow transplants found in Klein and

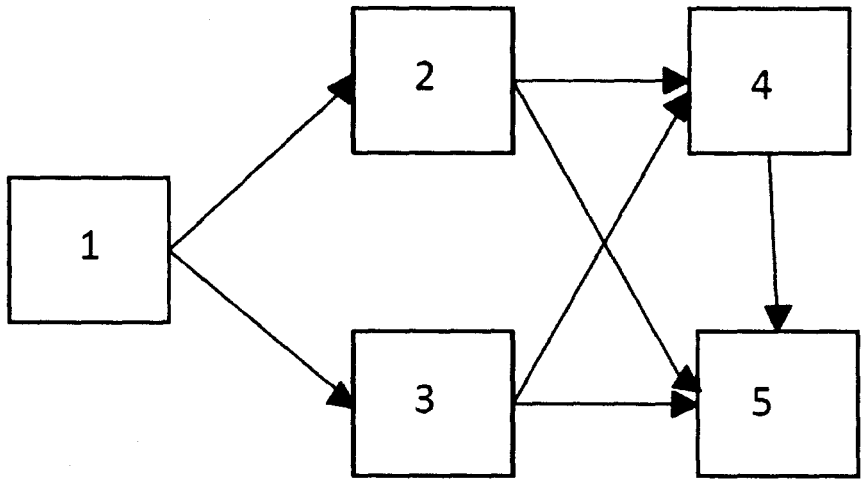

Figure 3. A five state model for cancer patients who received bone marrow transplants.

Moeschberger (1997). Following Datta and Satten (2002), we defined five states based on platelets returning to a normal level, presence of acute graft-versus-host disease (GVHD), and onset of chronic GVHD.

State 2 is entered when acute GVHD develops before the patients platelet level returns to normal. State 3 is entered if the patient's platelet level returns to normal before acute GVHD develops. State 4 is for patients who have both normal platelet levels and acute GVHD while State 5 is for those patients who have chronic GVHD. Patients transition to either state 2 or state 3 and then either state 4 or state 5, as depicted in Figure 3. Data from one patient was dropped for this analysis since his/her platelet levels never returned to normal and he/she did not develop acute GVHD. Patients do not necessarily progress to state 5 as they may remain in any state for any amount of time. Those patients who died or experienced relapse were considered censored for this example.

Our analysis allows for the censoring hazards to vary by state, since relapse or death can be predicted by the different (immunologic) states defined in this system. The cumulative censoring hazard for each state is estimated as the Nelson-Aalen estimator for censoring at each state and used to weigh the counting processes.

We open the data set **bmt** from the **KMsurv** package and form a data frame with column names "id", "stop", "st.stage", and "stage". (Code is suppressed here but is available in the accompanying R file.)

The first few lines of data are

```
R> head(data.sdc)
```

```
    id stop st.stage stage
20   1   13        1     3
198  1   67        3     4
272  1  121        4     5
21   2   18        1     3
217  2  139        3     5
22   3   12        1     3
```

Now we specify the tree structure.

```
R> Nodes <- c("1","2","3","4","5")
R> Edges <- list("1"=list(edges=c("2","3")),"2"=list(edges=c("4","5")),
+                "3"=list(edges=c("4","5")), "4"=list(edges=c("5")),
+                "5"=list(edges=NULL))
R> deptree <- new("graphNEL",nodes=Nodes,edgeL=Edges,
+                 edgemode="directed")
```

Next we will call the `msSurv` function to perform nonparametric estimation on the data. Since the data is subject to state dependent censoring, we must add the argument `cens.type="dep"` to the call of `msSurv`. Bootstrapping is used to estimate the variance for transition probabilities for state dependent censored data and may be specified using the `B` argument. The default number of bootstraps is 200 and that is used for this example.

```
R> DepEx <- msSurv(data.sdc,deptree,cens.type="dep",d.var=TRUE)
```

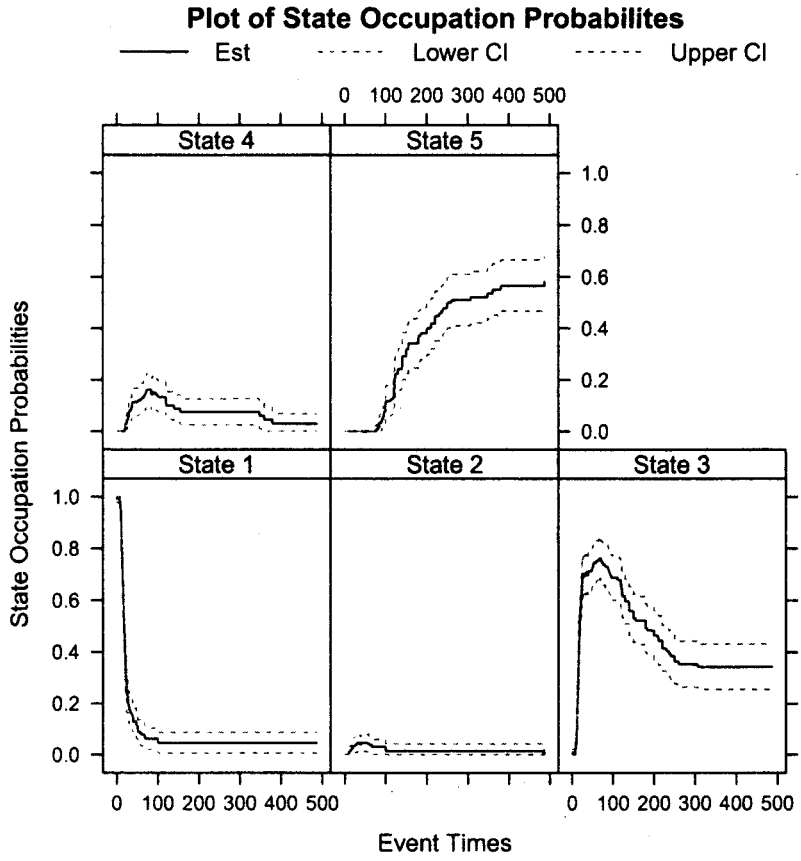

Figure 4. Plot of state occupation probability estimates and corresponding confidence intervals for each state in the system for data subject to dependent censoring.

The results are displayed graphically using the `plot` method. The `plot` method takes `msSurv` objects and produces plots of estimated quantities using functions available in the **lattice** package (Sarkar, 2008). The plots of the state occupation probabilities for every state in the system are produced by default with their corresponding 95% linear confidence intervals. For state dependent censoring, these confidence intervals are based on variances obtained through bootstrapping. Each state is plotted separately in a single panel. The results are given in Figure 4.

```
R> plot(DepEx)
```

Users may also specify specific states to be plotted using the `states`

argument. For example, to plot the state occupation probabilities for state 2, the user would type

```
R> plot(DepEx,state="2")
```

To plot the state occupation probabilities for states 2 and 3, use

```
R> plot(DepEx,state=c("2","3"))
```

Confidence intervals may be altered using the `ci.level` and `ci.trans` arguments which allow the user to specify a different confidence level ("90%" or "99%") or a different transformation ("log", "log-log", "cloglog" as described in the previous section). Confidence intervals are omitted from the plots using the `CI=FALSE` argument.

The `plot.type` argument is used to change the plotted estimators. As previously mentioned, the default is "stateocc" for the state occupation probabilities, but the user may create plots for the transition probabilities ("transprob"), state entry distributions ("entry.d"), and state exit distributions ("exit.d"). The state entry time distribution plots entry time distributions for all states except the initial state by default, but users may specify specific states using the `states` argument. The state exit time distributions are plotted for all non-terminal states by default, but specific non-terminal states may be requested by the user. We included the argument `d.var=TRUE` in the call of `msSurv` so that confidence interval estimates of the state entry and exit distributions are calculated and then plotted when `plot.type="entry.d"` or `plot.type="exit.d"`.

For example, the user may plot the state entry time distributions for all the non-initial states in the model

```
R> plot(DepEx,plot.type="entry.d",states="ALL")
```

or they may choose to plot specific state entry distributions

```
R> plot(DepEx,plot.type="entry.d",states=c("2","3"))
```

The resulting plot for all non-initial states is given in Figure 5.

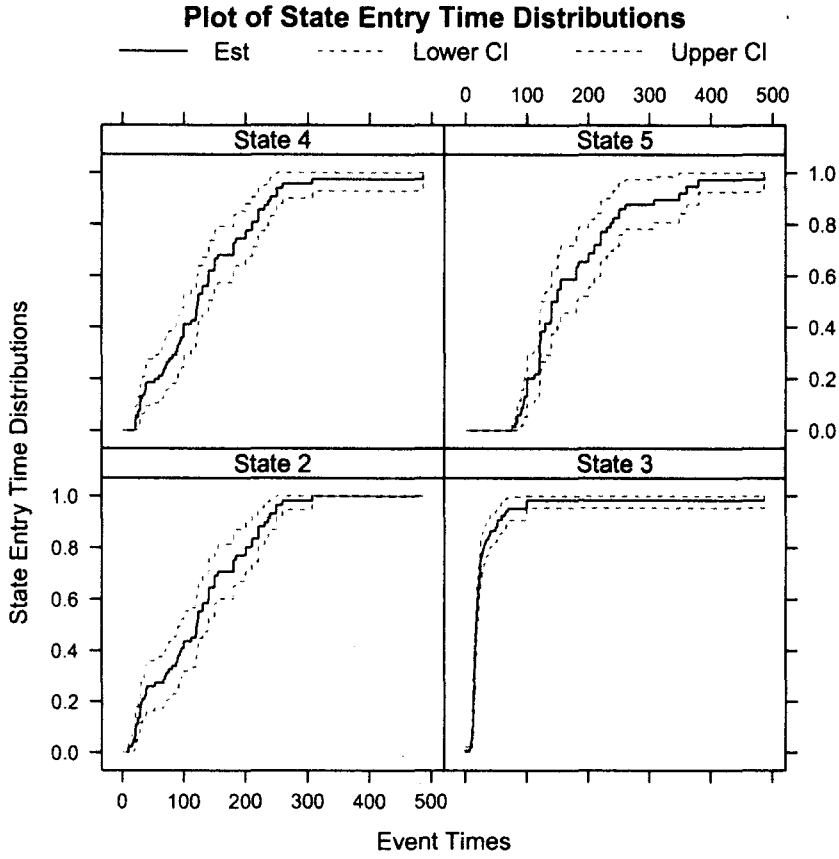

Figure 5. Plot of state entry time distribution estimates and corresponding confidence intervals for the state dependent censoring example.

The user may instead plot the state exit time distributions for all the non-terminal states in the model

```
R> plot(DepEx,plot.type="exit.d")
```

or instead plot specific state exit distributions

```
R> plot(DepEx,plot.type="exit.d",states="1")
```

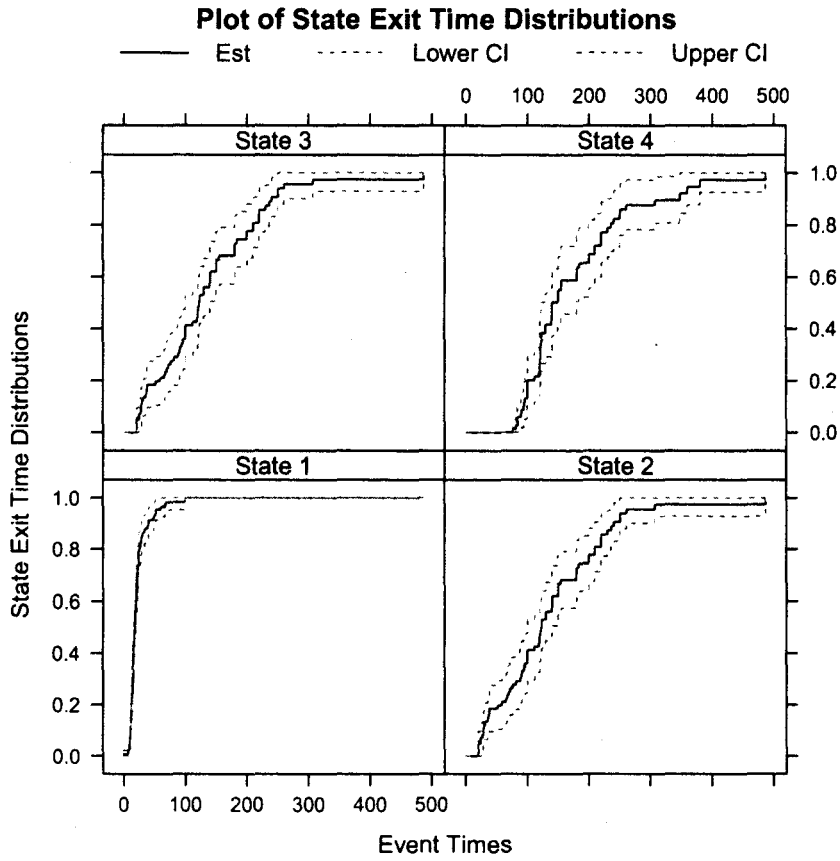

Figure 6. Plot of state exit time distribution estimates for all non-terminal states and corresponding confidence intervals for the state dependent censoring example.

The resulting plot for all non-initial states is given in Figure 6.

Default plots produced for transition probabilities plots estimates and confidence intervals for all possible transitions in the system.

```
R> plot(DepEx,plot.type="transprob")
```

The resulting plot is given in Figure 7.

Users may specify specific transitions using the **trans** argument. For example, if the user wants to plot the transition probabilities for the transitions out of state 1, say the "12" and "13" transitions, they would type

```
R> plot(DepEx,plot.type="transprob",trans=c("12","13"))
```

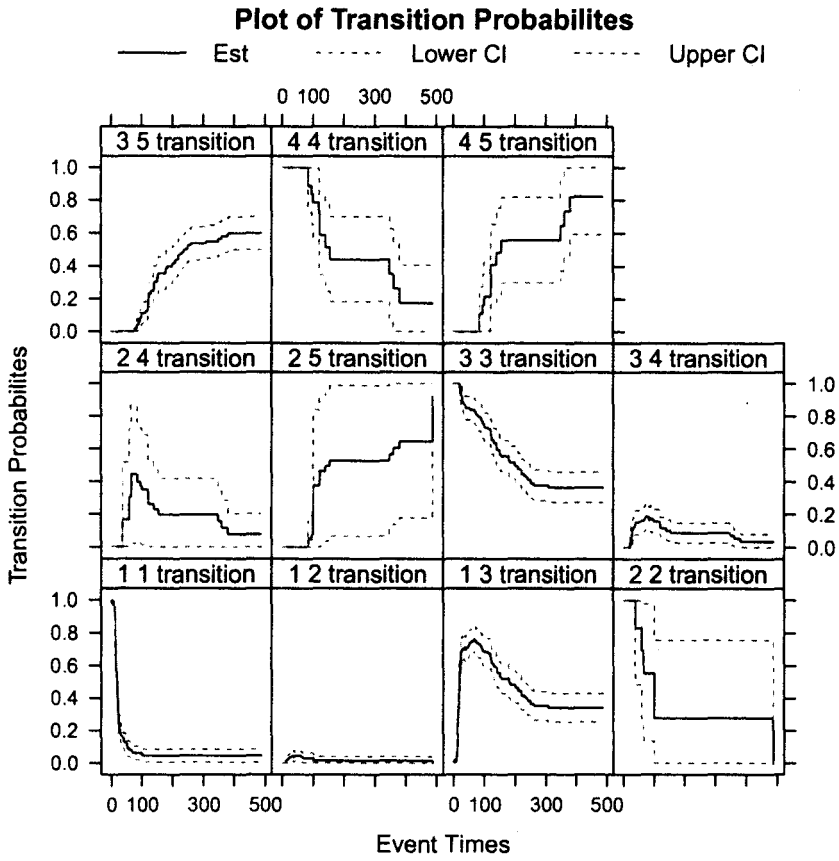

Figure 7. Plot of transition probability estimates and their corresponding confidence intervals for the state dependent censoring example.

To illustrate how to integrate the information in the various estimates, we describe a clinical interpretation of the data. Starting from state 1, nearly all of the patients transition to either state 2 (acute GVHD) or state 3 (normal platelets) by day 100. By 70 days post-transplant, roughly 75% of the patients have had their platelet levels return to normal before development of acute GVHD. The entry time distribution for state 3 reaches 1.0 by 100 days, indicating patients whose platelet levels return to normal prior to development of acute GVHD tend to do so within the first 100 days. About 40% of these patients never develop GVHD (remain in state 3), with a small percentage subsequently developing acute GHVD and 60% eventually going on to develop chronic GVHD. A much smaller percentage of

patients first develop acute GVHD, with about 40% of these eventually having their platelet levels returning to normal. All of the patients who develop acute GVHD first eventually go on to develop chronic GVHD.

D Discussion

We present a comprehensive R package for nonparametric estimation of general multistate models subject to independent right censoring and possibly left truncation. This package computes the marginal state occupation probabilities and state entry and exit time distributions for possibly non-Markov models. For a Markov model, the R package, **msSurv**, also calculates and returns the marginal integrated transition hazard and the hazard rate functions, as well as the transition probability matrix between any two states. Motivated by Datta and Satten (2001), **msSurv** also performs nonparametric estimation for state dependent right censored and possibly left truncated data. Currently no other packages available on CRAN (http://www.r-project.org) calculate the state entry and exit time distributions for censored multistate data or calculate nonparametric estimates for data subject to state dependent censoring. Pointwise confidence intervals for state occupation probability and transition probability matrices are obtained and returned for independent censoring from closed-form variance estimators and for state dependent censoring using the bootstrap. Pointwise confidence intervals for state entry and exit time distributions are obtained using the bootstrap. The bootstrap is also used to find variance estimators for state occupation probabilities of data subject to state dependent censoring. The **msSurv** package provides functions to find the state occupation probabilities at a specific time t and to find the transition probabilities between any two times s and t ($s \leq t$). Package **msSurv** is written using S4 classes and methods. Methods are available to print, plot, and summarize the `msSurv` objects.

The **msSurv** package has a number of limitations that will be improved upon

in future releases. Currently state dependent censoring is the only censoring scheme incorporated in **msSurv**, future expansion could include incorporating general covariates into the (dependent) censoring mechanism. The package could also be extended to include estimation for current status data and eventually interval censored data (Datta and Sundaram, 2006; Lan and Datta, 2010b). Other extensions include calculations of the state waiting time distributions (Satten and Datta, 2002) and allowing estimation for recurrent event models.

There are two types of interval censored data. In the context of multistate models, type I interval censored data, referred to as current status data henceforth, occurs when individuals are inspected at a single random time and the corresponding state information is recorded. This type of censoring is often found in reliability studies and cross-sectional studies. Nonparametric estimators of key marginal quantities for current status data were developed in Datta and Sundaram (2006), Datta *et al.* (2009), and Lan and Datta (2010b) for possibly non-Markov models with directed tree structures. Datta and Sundaram (2006) obtained product limit estimators (PLE) of state occupation probabilities for data in this setup. Lan and Datta (2010b) extended the estimation to state entry and exit time distributions.

Type II interval censored data, simply referred to as interval censored data henceforth, occurs when individuals are inspected at multiple random inspection times and their corresponding state information is recorded. Often only intervals where a state change has taken place are kept. The exact transition times are not observed but known to have taken place in an interval. This type of censoring often arises in clinical trials and longitudinal studies where individuals are subject to periodic follow-up and the event of interest is repeatedly observed. Inspection times for different individuals are typically assumed to be independent while the different inspection times for each individual are dependent. The efficient use of possibly non-independent information coming from the same individual causes an additional methodological challenge for us. A fully nonparametric approach to this problem is not currently available in the literature unless one is considering specific models, e.g., competing risks (Hudgens *et al.*, 2001) or Markov illness-death (Frydman, 1995).

In this chapter, we construct fully nonparametric inference procedures to estimate the state occupation probabilities as well as state entry and exit time distributions for interval censored data from a multistate system with a directed tree structure. Structural assumptions about the model, such as Markovity, are avoided whenever possible. Motivated by Datta and Sundaram (2006) and Lan and

Datta (2010b), we construct nonparametric estimators through weighted isotonic regression with possibly dependent indicator terms for the event of two transitions occurring by a specified time. We seek to improve efficiency through use of a weight matrix W taken as the identity matrix and the diagonal variance matrix estimated from the data.

A simulation study in Section C illustrates the performance of the new estimators for both a tracking multistate model and a branching multistate model. We also illustrate the new method using a liver waiting list and transplantion dataset obtained from the UNOS (United Network for Organ Sharing) Standard Transplant Analysis and Research (STAR) files for liver registrations and transplants. Data analysis is performed and state occupation probabilities are calculated for states based on the MELD (Model for End-Stage Liver Disease) scores of the patients on the waiting list and whether the patient was eventually transplanted or removed from the waiting list.

B Estimation

Consider a finite state space $\mathcal{S} = \{1, \ldots, M\}$ for a multistate model with directed tree topology to model the interval censored data where every state $j \in \mathcal{S}$ can be reached from the initial state 1 according to
$\pi(j) = 1 = s_1 \to s_2 \to \ldots \to s_j = j$. This model allows the possibility that not all individuals need to be at state 1 at time 0.

Individuals progressing through this multistate system are inspected at multiple inspection times and the corresponding state information is recorded. Exact transition times are not observed but known to have taken place in an interval. Data is represented as $\{C_{ik}, S_i(C_{ik})\}$ for $1 \leq k \leq n_i$; $1 \leq i \leq n$, where n denotes the total number of individuals, n_i denotes the number of inspection times retained for the ith individual, C_{ik} are the inspection times for the ith individual and the corresponding state information is denoted $S_i(C_{ik})$. The data for each

individual is dependent while individuals are independent. We assume that all transition times and censoring variables are continuous. We further assume random censoring of the inspection times so that the inspection times are independent of the corresponding state information. Typically only intervals ending with inspection times where state changes have occurred are kept.

1 State Occupation Probability

Datta and Sundaram (2006) presented state occupation probability estimators for current status data based on the product limit formula for state occupation presented by Datta and Satten (2001). Let $U_{jj'}$ denote the (unobserved) transition time of an individual i making a transition from state j to another state j' ($= \infty$ if this transition is never made by the individual). For the counting process,the pooled adjacent violators (PAV) algorithm is applied to perform isotonic regression of $I(U_{jj'} \leq C)$ on C based on the pairs of data $\{C_i, I(U_{jj',i} \leq C_i))\}$ (Barlow *et al.*, 1972) and then kernel smoothing is applied to the resulting estimates from PAV to smooth the jumps while maintaining monotonicity. The at risk estimators are obtained through application of the Nadaraya-Watson estimator, with a PAV step added to only the estimation of the risk set out of the initial state.

Interval censored data contains more information and also introduces dependency between inspection times for each individual, which makes computation more complicated. For this type of dependent data, one can ignore the dependency of inspection times for each individual, pool the data, and apply the same methodology used by Datta and Sundaram (2006) to obtain valid (e.g., consistent) but inefficient nonparametric estimators of various marginal quantities. To improve efficiency, we use weighted regression techniques involving a matrix W combined with a monotonicity constraint to develop estimators. We take W as the identity matrix and diagonal variance matrix for estimation in this dissertation research.

Consider two states j and j' in a multistate system. It is necessary to keep

track of all transitions made by individuals progressing through such a system to calculate marginal estimators in a non-Markov system. Let X_j denote the (unobserved) indicator of the event that an individual would ever enter state j. Let $N^*_{jj'}(t)$ denote the usual counting process which counts the number of j to j' transitions [0,t] for the complete data. Note that this can be expressed as $N^*_{jj'}(t) = \sum_{i=1}^{n} I(U_{i,jj'} \leq t)$. By the law of large numbers,

$$n^{-1}N^*_{jj'}(t) \xrightarrow{P} n^{-1}EN^*_{jj'}(t) = P\{U_{jj'} \leq t\} = n_{jj'}(t), \text{ say.}$$

Furthermore, for any $t \geq 0$, $E(I(U_{jj'} \leq C_k | C_k = t)) = P\{U_{jj'} \leq t\}$. Therefore, $n^{-1}\widehat{N}_{jj'}(\cdot)$ can be obtained by a nonparametric regression estimator of $I(U_{jj'} \leq C_k)$ given C_k. $Pr\{U_{jj'} \leq t\}$ is monotonic in t, however the indicator terms $I(U_{jj',i} \leq C_{ik})$ of the event that the j to j' transition has taken place by time C_k for $1 \leq k \leq n_i$ are dependent with a non-diagonal variance-covariance matrix ($k \leq k'$). Thus, the counting process, $n^{-1}\widehat{N}(\cdot)$, is estimated through a weighted isotonic regression on the indicator terms using a weight matrix W.

Initially the dependence between the data is ignored and all the data is pooled $\{C_{ik}, I(U_{jj',i} \leq C_{ik}), 1 \leq k \leq n_i; 1 \leq i \leq n\}$ and isotonic regression without weights is calculated and smoothed by kernel smoothing. The resulting state occupation probabilities are then used to compute an estimate of the variance.

Then, a weighted isotonic regression is run to improve efficiency, e.g. minimize $\sum_{i=1}^{n} \Delta_i^T \widehat{W}_i \Delta_i$ subject to $\widehat{P}\{U_{jj'} \leq c_{(i)}\} \leq \ldots \leq \widehat{P}\{U_{jj'} \leq c_{(n)}\}$ where $c_{(i)} \leq \ldots \leq c_{(n)}$ are the ordered inspection times in the pooled sample where $\Delta_i = (\widehat{P}\{U_{jj'} \leq c_{ik}\} - I\{U_{jj'} \leq c_{ik}\}, 1 \leq k \leq n_i)$. Weighted isotonic regression of $I(U_{jj',i} \leq C_{ik})$ on C_{ik} is then performed using the matrix W as weights. For this dissertation research we use two different selections of W for estimation – the identity matrix and the inverse of the diagonal variance matrix whose entries are estimated as $P\{U_{jj'} \leq C_{ik}\}(1 - P\{U_{jj'} \leq C_{ik}\})$ from the data. Note that the result of using the identity matrix is analogous with ignoring the dependence in the data

The resulting estimates, denoted $\widehat{N}_{jj'}^{P}(\cdot)$, are then smoothed to reduce flat parts while maintaining monotonicity. We apply kernel smoothing using a log-concave density, $K > 0$, which leads to

$$\widehat{N}_{jj'}(t) = \frac{\sum_{i=1}^{n}\sum_{k=1}^{n_i}\widehat{N}_{jj'}^{P}(C_{ik})\,K_h(C_{ik}-t)}{\sum_{i=1}^{n}\sum_{k=1}^{n_i}K_h(C_{ik}-t)}$$

as the final estimate of $N_{jj'}^{*}(t)$.

Further efficiency may be obtained through weighted regression using the the full variance-covariance matrix as W (see discussion in Section C of IV: Future Research).

For the "at risk" set, let $Y_j^*(t)$ denote the number of individuals "at risk" of transitioning out of state j by time t for the complete data. Note that this can be expressed as $Y_j^*(t) = \sum_{i=1}^{n} I(S_i(t-))$ where $(S_i(t-))$ is the state occupied just before time t. Analogous to the counting process arguments, $P_j(t-) = Pr\{S(t-) = j\}$ is the limit of $n^{-1}Y_j^*(t)$ in probability, where $Y_j^*(t)$ denotes the "at risk" set of transitions out of state j with the complete data.

For this research the "at risk" set is estimated as $\widehat{Y}_j(t) = n_j + \widehat{N}_{\cdot j}(t) - \widehat{N}_{j\cdot}(t)$, where n_j denotes the number of individuals in state j by time 0, $\widehat{N}_{\cdot j}(t)$ denotes the total number of transitions into state j at time t from the estimators above and $\widehat{N}_{j\cdot}(t)$ denotes the total number of transitions out of state j by time j in the counting process estimator defined above.

The estimators of the marginal integrated transition hazard are defined as

$$\widehat{A}_{jj'}(t) = \begin{cases} \int_0^t \widehat{Y}_j(u)^{-1}\, d\widehat{N}_{jj'}(u) & j \neq j' \\ -\sum_{j\neq j'}\widehat{A}_{jj'}(t) & j = j', \end{cases}$$

where $\widehat{Y}$ and $\widehat{N}$ are estimators of the at risk sets and counting processes, respectively.

Then, the class of state occupation probabilities are computed using the identity

$$\widehat{p}_j(t) = \sum_{k=1}^{M} \frac{\widehat{Y}_k(0+)}{n} \left(\widehat{P}(0,t)\right)_{kj},$$

where $\left(\widehat{P}(0,t)\right)_{kj}$ is the kjth element of $\widehat{P}(0,t) = \prod_{(0,t]} \left(I + d\widehat{\boldsymbol{A}}(u)\right)$ and $\frac{\widehat{Y}_k(0+)}{n}$ is the relative proportion of individuals in various states at time 0.

2 State Entry and Exit Distributions

State entry and exit distributions will be estimated through normalized sums of estimated state occupation probabilities for progressive systems. For state j, let U_{ij}^* denote the time the ith person enters state j ($= \infty$ if the ith person never enters state j) and let V_{ij}^* denote the time the ith person leaves state j ($= \infty$ if state j is never entered or if state j is never left). Let $X_{ij}^* = I\left(U_{ij}^* < \infty\right)$ be the indicator function that takes the value 1 if the ith person ever enters state j and 0 otherwise.

Let F_j denote the state entry distribution function for the individuals who ever enter state j (i.e., $X_j = 1$) defined as $F_j(t) = P\{U_j \leq t | X_j = 1\}$, where $F_0(t) = 1$ for all $t \geq 0$. Any state will be reached from the root node by a unique path. Let S^j denote the set of states ℓ such that state j is on the path from the root node to ℓ. The entry time distribution to state j is estimated by taking the normalized sum of the estimated state occupation probabilities of state j and all the other states that come after j in the progressive system, i.e.,

$$\widehat{F}_j(t) = \frac{\sum_{k \epsilon S^j \cup j} \widehat{p}_k(t)}{\sum_{k \epsilon S^j \cup j} \widehat{p}_k(\infty)},$$

where $\widehat{p}_k(\infty) = \lim_{t \to \infty} \widehat{p}_k(t)$.

Similar to the description above, let V_j denote the departure time for state j of individuals who ever enter state j. Let G_j denote the state exit time distribution functions, $G_j(t) = P\{V_j \leq t | X_j = 1\}$, where $G_j(t) = 0$ for all $t \geq 0$ if j is a terminal node in the directed tree structure. When j is a transient state, $\widehat{G}_j(t)$ is estimated by taking the normalized sum of estimated state occupation probabilities

of all states that come after state j in the progressive system, i.e.,

$$\widehat{G}_j(t) = \frac{\sum_{k \epsilon \delta j} \widehat{p}_k(t)}{\sum_{k \epsilon \delta j} \widehat{p}_k(\infty)}.$$

C A Simulation Study

We performed a Monte Carlo simulation study to compare the performance of the state occupation probabilities, entry and exit time distributions described in the previous section to the empirical estimates obtained from the full data. We consider two common models: a 3 state tracking model (Figure 8) and a 5 state branching model (Figure 2).

Simulated data was generated for both tree structures under the Markov setup and the semi-Markov setup. For the semi-Markov setup, times were generated for each state and then added progressively to generate transition times for the successive states. In the Markov setup, times were randomly generated for the initial state from the distributions and then successive state times were generated using the formula $T_j = D^{-1}\left(D\left(T_{j-1}\right) + R_j\left\{1 - D\left(T_{j-1}\right)\right\}\right)$ where T_{j-1} is the previous transition time, R_j is a random number generated from $U(0,1)$ independent of T_{j-1}, $D(\cdot)$ denotes the distribution function of the sampling distribution used, and $D^{-1}(\cdot)$ denotes the corresponding quantile function. Transition times were generated from either a lognormal distribution or Weibull distribution. Under each data setup, censoring times were generated from either a uniform or Weibull distribution. Thus, eight simulation settings were run for each of the two multistate models. Sample sizes of 100, 200, 500, and 1000 were considered. 100 iterations were completed for the 1000 sample setups and 1000 iterations were completed for the other sample sizes.

The gpava function in the **isotone** package in R was used to perform the weighted and non-weighted least squares regression for the counting process estimators (de Leeuw *et al.*, 2009). Kernel smoothing of the estimators was

performed using the `ksmooth` function in the **stats** package. Bandwidths for the initial, non-weighted least squares fit were found using the function `bw.SJ` of the inspection times, but changed to 0.4 to balance variation in the simulated data. For the subsequent smoothing of the weighted least squares fit, the bandwidth was selected by the `bw.SJ` function using the inspection times.

For complete data, estimates of the state occupation probability and state entry and exit time distributions were computed using the **msSurv** package. The L_1 distances were calculated to assess the performance of our estimators according to the formula

$$\Delta = E \int \left| \widehat{\theta}(x) - \widehat{\theta}_E(x) \right| dF_n(x),$$

where $\widehat{\theta}$ and $\widehat{\theta}_E$ denote the estimators of θ based on the complete data and interval censored data, respectively, and dF_n denotes the distribution function of the inspection times. The function θ is taken as the state occupation probability, state entry time distribution, or the state exit time distribution. The L_1 distances, denoted Δ, were estimated by averaging the Monte Carlo estimates obtained by the process described above.

The results of the L_1 distance computations of both the weighted and non-weighted nonparametric estimates for the three-state tracking model are provided in Tables 1-8. The L_1 results for the five-state branching model are shown in Tables 10-12.

1 A 3 state example

For the three-state tracking model, we simulated data sets with individuals starting in state 1 at time 0. State exit times were generated for states 1 and 2. For the initial state 1 in both the Markov and semi-Markov setup, state exit times were generated from either the lognormal distribution with log mean -1.5 and log standard deviation 1 or the Weibull distribution with a shape parameter or 3 and a scale parameter of 1. The number of censoring times were randomly generated to be

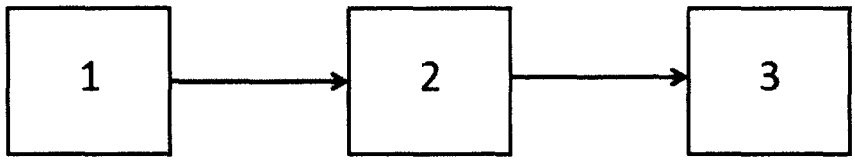

Figure 8. A three state model for simulated multistate data subject to interval censoring.

between 2 and 4 and the censoring times were generated from either the uniform distribution with minimum 0 and 2 median absolute deviations above the maximum time as the maximum or the Weibull distribution with shape parameter 2 and scale parameter 1.05.

For the Markov setup, the exit times from state 2 were generated according to the formula $T_2 = D^{-1}\left(D\left(T_1\right) + R_2\left\{1 - D\left(T_1\right)\right\}\right)$ where T_1 is the first inspection time, R_2 is a random number generated from $U\left(0,1\right)$ independent of T_1, $D\left(\cdot\right)$ denotes the distribution function for either the lognormal distribution with log mean -1.5 and log standard deviation 1 or the Weibull distribution with shape parameter 3, and $D^{-1}\left(\cdot\right)$ denotes the corresponding quantile function. Tables displaying the L_1 results for the uniformly censored Weibull and log-normal simulations are given in TableS 1 and 2. The L_1 results for Weibull censored log-normal and Weibull simulations are provided in Tables 3 and 4.

In the Markov setup, the L_1 values decrease as the sample size increases for both the weighted and unweighted least squares fits in all four scenarios. The new weighted estimators offer lower $L1$ values for all estimated quantities (state occupation probabilities and state entry time distributions) for both simulations with lognormal waiting times. The L_1 distances for the Weibull waiting times with uniform censoring show that the weighted estimators offer only slightly lowered L_1 values while the L_1 values for the new estimators are slightly higher than those of the unweighted estimators for the simulation with Weibull waiting and censoring times.

Results are the same for the semi-Markov setup where exit times from the transient state 2 are generated by adding the exit times from state 1 to times generated from either a lognormal distribution with mean log 0 and mean standard deviation of 0.1 or from the Weibull distribution with shape 3 and scale 1. The L_1 values decrease as sample size increases and 3 scenarios have smaller L_1 values for the weighted estimators(lognormal waiting times with uniform censoring or Weibull censoring and Weibull waiting times with uniform censoring), while the L_1 values of the unweighted estimators are lower in the Weibull-Weibull simulation. The L_1 distance results are in Tables 5, 6, 7, and 8

Plots of the log L_1 values of state occupation probabilities by log sample size of the semi-Markov setup are provided in Figures 10 and 11. Similar plots for the state entry time distribution for the same setups are provided in Figures 12 and 13. These plots illustrate an approximate linear relationship between the logarithms of the mean L_1 distances and the log of the sample size suggesting that the difference between the estimates will converge to zero as n approaches infinity. Further, the slopes of the lines provide an estimate of the rate of convergence. The plots for the weighted least squares estimates for the interval censored data appear to do as well if not better than those of the non-weighted estimates, suggesting that the weighting offers improved efficiency over the non-weighted estimators for interval censored data. Plots for the Markov setting are also provided in Figures 14, 15, and 16 and illustrate the same trends as those for the semi-Markov setting.

2 A 5 state example

For the five-state branching model (Figure 2), we assumed individuals started in state 1 at time 0. Individuals had a 60% chance of transitioning to state 2 (transient) or a 40% chance of transitioning to state 3 (terminal). State 1 waiting times were generated from either a lognormal distribution with log mean -1.5 and log standard deviation 1 or a Weibull distribution with shape parameter 3 and scale

parameter 1. State information was randomly assigned through a random Bernoulli variable that is independent of the waiting times. For individuals who transitioned to state 2, they had a 40% chance of transitioning to state 4 and a 60% chance of transitioning to state 5. For the semi-Markov setting, a second waiting time was generated from the corresponding lognormal or Weibull distribution for those individuals who transitioned to state 2 and the (total) waiting time for state 2 was taken as the sum of the two waiting times. In the Markov setting, the formula $T_2 = D^{-1}\left(D\left(T_1\right) + R_2\left\{1 - D\left(T_1\right)\right\}\right)$ where T_1 is the first transition time, R_2 is a random number generated from $U(0,1)$ independent of T_1, $D\left(\cdot\right)$ denotes the distribution function for the Weibull distribution with shape parameter 2, $D^{-1}\left(\cdot\right)$ denotes the corresponding quantile function. State information is controlled through a second Bernoulli random variable that is independent of the waiting times generated for state 2. Individuals were inspected a random number of times from inspection times generated from the uniform distribution or the Weibull distribution with shape parameter 2 and scale parameter 1.05. State information was assigned based on these inspection times.

As before, the empirical estimates based on complete data were calculated using the **msSurv** package, and the L_1 distances were calculated to assess the performance of our estimators.

Tables displaying the L_1 distance results for the uniformly censored log-normal and Weibull simulations in the Markov setting are given in Tables 9 and 10. The L_1 results for the Markov model with Weibull censored log-normal and Weibull simulations are provided in Tables 11 and 12. Tables displaying the L_1 distance results for the semi-Markov setup are in Tables 13, 14, 15, and 16.

Plots of the log L_1 values of state occupation probabilities by log sample size for the Markov setup are provided in Figures 19 and 20. Plots for the state entry time distributions are provided in Figures 21 and 22.

These scatter plots illustrate an approximate linear relationship between the

logarithms of the mean L_1 distances and the log of the sample size, suggesting that the difference between these estimates will converge to zero as $N \to \infty$. The plots comparing the weighted least squares estimates (using the diagonal variance as a weight) against the nonweighted least squares (using the identity matrix as a weight) show some mixed results. The weighted method offers improvement for the state 1 occupation probability and some improvement (especially for larger sample size) for state 2. The weighted estimators appear to have slightly larger L_1 values for states 4 and 5 than the nonweighted estimates. This may be due to the cumulative effect of product limit estimation where the later states (in this case 4 and 5) depend on estimates from earlier states (1 and 2) causing misestimation of the state occupation probabilities to compound through the model. For state entry time distributions, the weighted estimation offered improvements for states 1 and 2 and performed as well as the nonweighted estimates for states 4 and 5 for simulations with lognormal waiting times and Weibull censoring times. For the simulation with Weibull waiting times and censoring times, the weighted estimator offered improvement for the state 2 entry time distribution, comparable results for the state 3 entry distributions, and actually performed worse for the later states.

D A Real Data Example

We now present estimates of the state occupation probabilities, state entry time distributions, and state exit time distributions for liver registration and transplant data. The liver data is from the United Network for Organ Sharing (UNOS) Standard Transplant Analysis and Research (STAR) files for liver registrations and transplants.

The number of patients waiting for a liver transplant has increased since liver transplantation has become a universally accepted treatment for end-stage liver disease. The UNOS data set contains information on all waiting list registrations and transplants of livers that have been listed or performed in the U.S. and reported

to the Organ Procurement and Transplantation Network (OPTN).

The Model for End-Stage Liver Disease (MELD) score has become the standard allocation policy for liver transplants, as it has proven to be an effective predictor of pre-transplantation mortality and post-transplantation outcome (Martin *et al.*, 2007). MELD scores are calculated based on a combination of bilirubin, international normalized ratio for prothrombin time (INR), and creatinine lab values, and determine the urgency with which a patient needs a liver transplant within the next three months. MELD scores range from 6 (less ill) to 40 (gravely ill).

Patients are followed through-out the process and each follow-up event is recorded, leading to multiple records per patient on the waiting list and thus creating interval censored data. A change in MELD score is only known to occur between inspection times while transplant and removal times are known exactly. Inspection times for patient i are taken as the number of days since the patient was added to the waitlist. The corresponding state information $S_k(C_k)$ is assigned based on their MELD scores and waitlist status (either transplanted, still on the list, or removed from the list) at the inspection times.

We define a multistate model illustrated in Figure 9 based on the MELD scores and waitlist status. Patients with a MELD score of <15 are assigned to state 1, MELD scores of 15-22 to state 2, 23-30 to state 3, and 31 or above are assigned to state 4. Patients who received transplants and were deleted from the list are assumed to be in state 5, while patients who are removed without transplant are in state 6. Individuals are added to the wait list with a variety of MELD scores and therefore individuals may not necessarily enter the system in state 1.

Some patients receive MELD exceptions due to cases where MELD scores may not reflect the urgency of their need for a transplant (i.e., patients with hepatocellular carcinoma) and they are exempted from our analysis. We restrict our analysis to only adults 18 years old and older. We restricted our analysis to those who were added to the waitlist between 2-27-2002 and 2-27-2003.

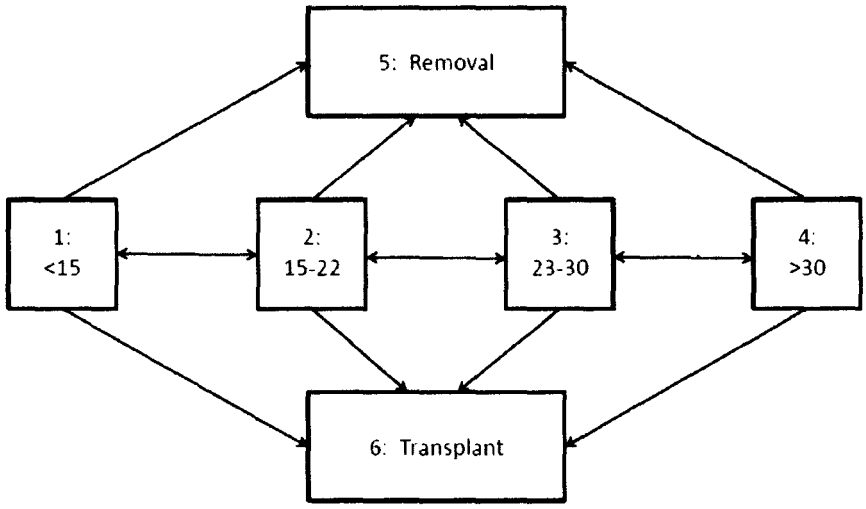

Figure 9. A six state model illustrating the states for liver transplants based on the UNOS data set.

We took a random sample of 1000 individuals from the data set for our analysis. To handle the recurrence in the model, we expanded the model for our analysis to include additional states to handle the multiple occurrence of transitions so that, for example, the second "12" transition represents a transition into a state representing "second visit to state 2". Since actual event times are known for patients who are removed from the wait list or transplanted the random censoring assumption for interval censored data is violated for these event times. Hence,f the counting processes for transitions into those states will be estimated directly from their indicators without isotonic regression. These transitions were used in the ultimate computation of state occupation probabilities.

The distribution of the inspection times is displayed in Figure 23. The estimated state occupation probabilites are show in Figure 24. State entry and exit distributions are not estimated for this example since the reoccurrence violates the assumption that individuals pass through a state only one time.

Individuals enter the liver transplant waitlist with MELD scores between 6 and 40. 55.8% of individuals enter the waitlist with a MELD score less than 15

(state 1), 32.2% have a MELD score between 15 and 22 (state 2), 7% have a MELD score between 23 and 30 (state 3), and 5% have a MELD score of 31 or greater (state 4). The state 1 occupation probability increases slightly between 0 and 176 days on the waiting list before decreasing and eventually getting very close to 10% after 2,753 days on the waiting list. Not every patient on the waiting list will progress out of state 1 and some patients will be transplanted or removed from the waiting list with a MELD score less than 15. The state 2 occupation probability drops to almost 0 around 175 days after entry on the waitlist before increasing to 45% 2,800 days on the waitlist. The state 3 occupation probability peaks at 10% around 55 days on the waitlist before eventually reaching 0 at almost 2,900 days on the waitlist. Note that the drop in state 2 occupation probabilities corresponds to the increase of the state 1 and 3 occupation probabilities. The state 4 occupation probability drops to 0 after 905 days on the wait list before slightly increasing after 2,667 days on the waitlist. About 6% of patients are removed from the waitlist (state 5) by 2,473 days after entry on the waitlist while 33.4% of patients receive a liver transplants at 2,734 days on the waitlist.

E Discussion

In this chapter we consider multistate models with directed tree structure subject to interval censored data. Structural assumptions are not required to obtain valid nonparametric state occupation probability estimates. We introduced a novel fully nonparametric estimation method for general multistate model subject to interval censoring where there were were no methods previously available. The methods presented produce reasonable estimates for a variety of complex settings.

We compared the weighted and smoothed isotonic regression to a non-weighted version and found that the overall performance offered significant improvement in the three state tracking model and offered some improvement in the five-state branching model. Estimation seems very sensitive to bandwidth as we

used a fixed bandwidth for the three-state model and used a more generalized, time based bandwidth for the branching model.

In this research we estimated the at risk set as $\widehat{Y}_j(t) = n_j + \widehat{N}_{\cdot j}(t) - \widehat{N}_{j\cdot}(t)$ since $\widehat{N}_{jj'}$ estimates the cumulative value of the counting processes and it seemed redundant to calculate the Y_js through the regression and kernel smoothing process. However, one potential problem is that computations are cumulative and the cumulative effects may propagate as computation continues and cause errors later in estimation. Therefore, a more local computation of the number at risk set may help improve estimation. One approach we may try in the future is local smoothing since the $Y_j^*(t)$ process does not have to be monotonic for a transient state j. For progressive models the number at risk set for transition out of the initial step is monotonic, so a step may be added to account for monotonic constraints. Note that $\widehat{Y} = \widehat{N}$ after monotonization.

Estimation may also be improved through the use of the full variance-covariance matrix as weights in estimating the counting processes. We investigated this scenario but had problems with singularity and current available R packages for isotonic regression can only handle nonsingular matrices. Methods are currently under construction for this case and will be reported elsewhere.

The resulting estimators of the counting process and number at risk may be used to calculate state waiting time distributions or perform hypothesis testing to compare two (or more) groups. A more in-depth discussion of future work can be found in Chapter IV.

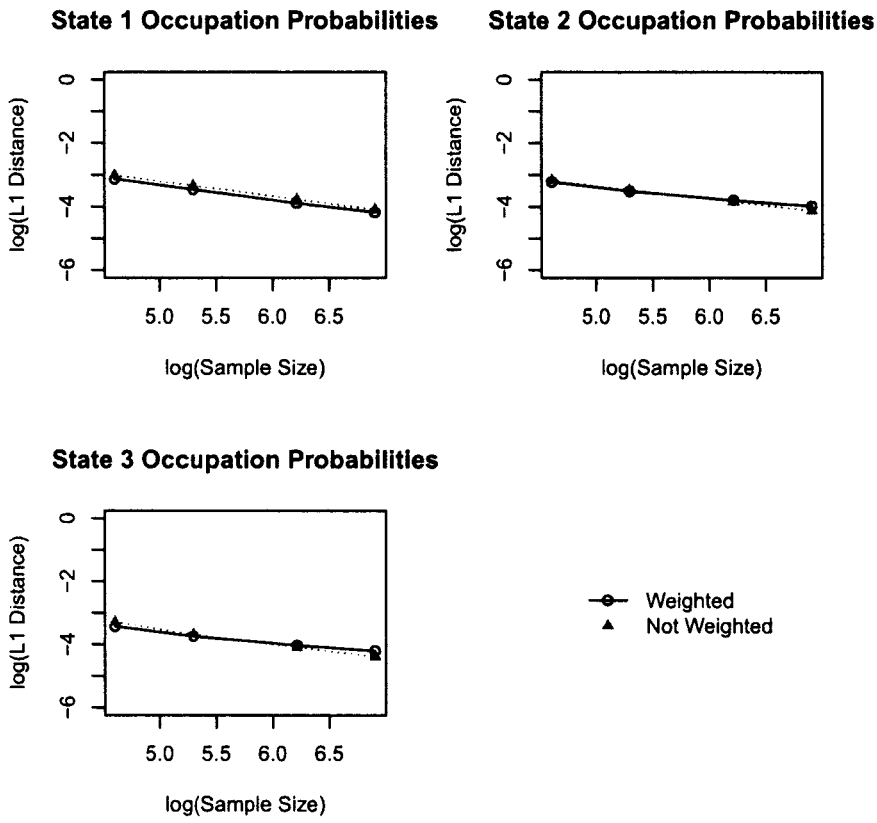

Figure 10. The log mean L_1 values of state occupation probabilities for the three-state tracking semi-Markov model with Weibull waiting times and uniform censoring times.

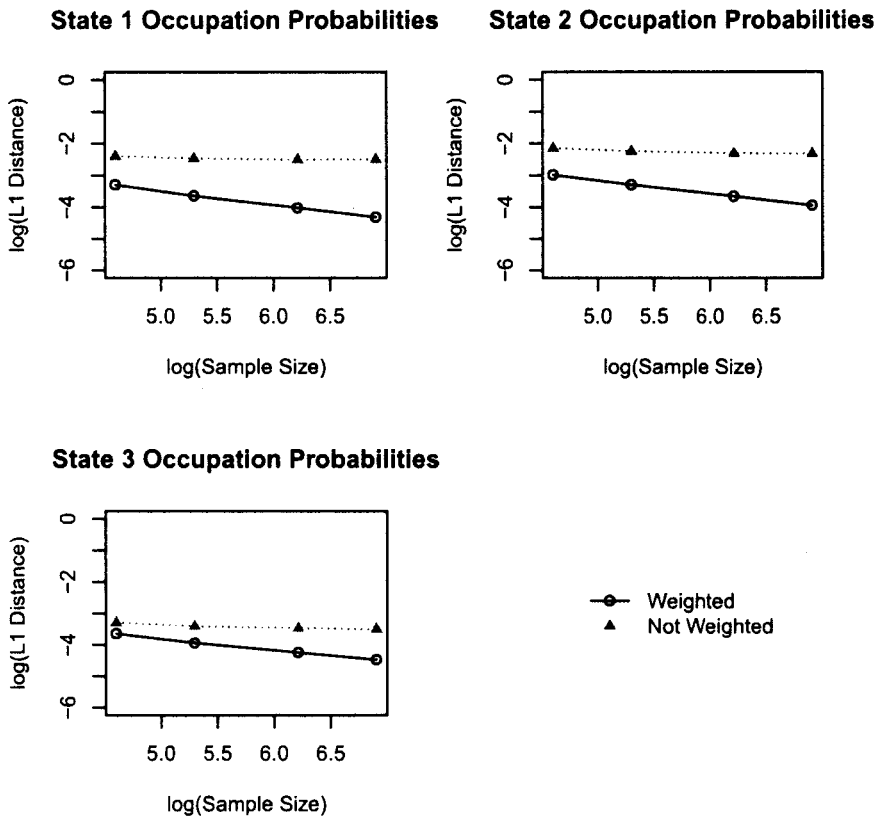

Figure 11. The log mean L_1 values of state occupation probabilities for the three-state tracking semi-Markov model with lognormal waiting times and Weibull censoring times.

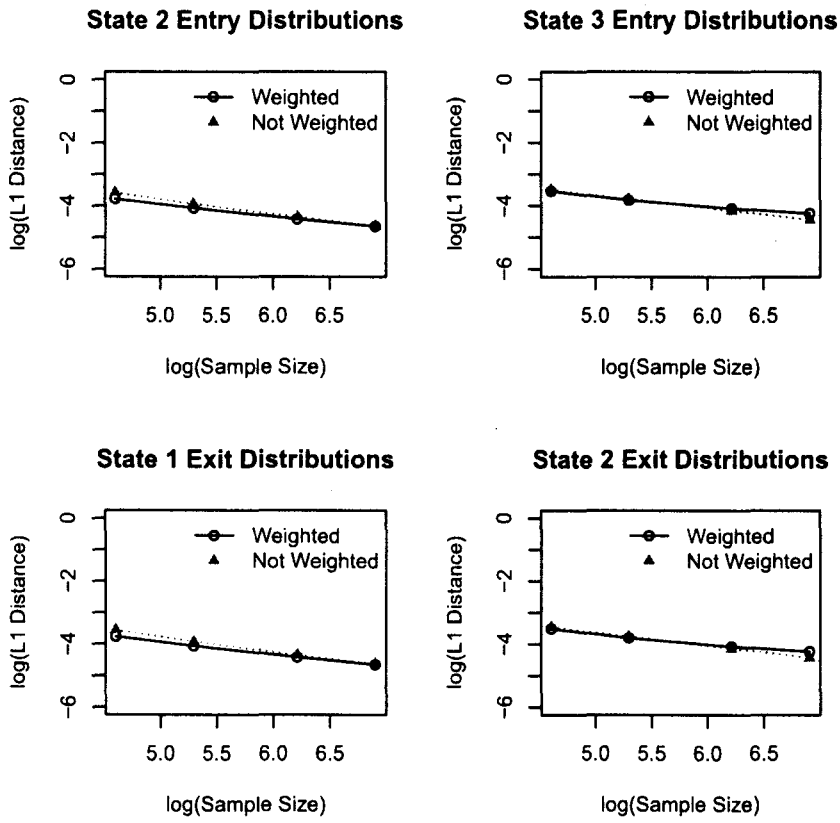

Figure 12. The log mean L_1 values of state entry and exit time distributions for the three-state tracking semi-Markov model with Weibull waiting times and uniform censoring times.

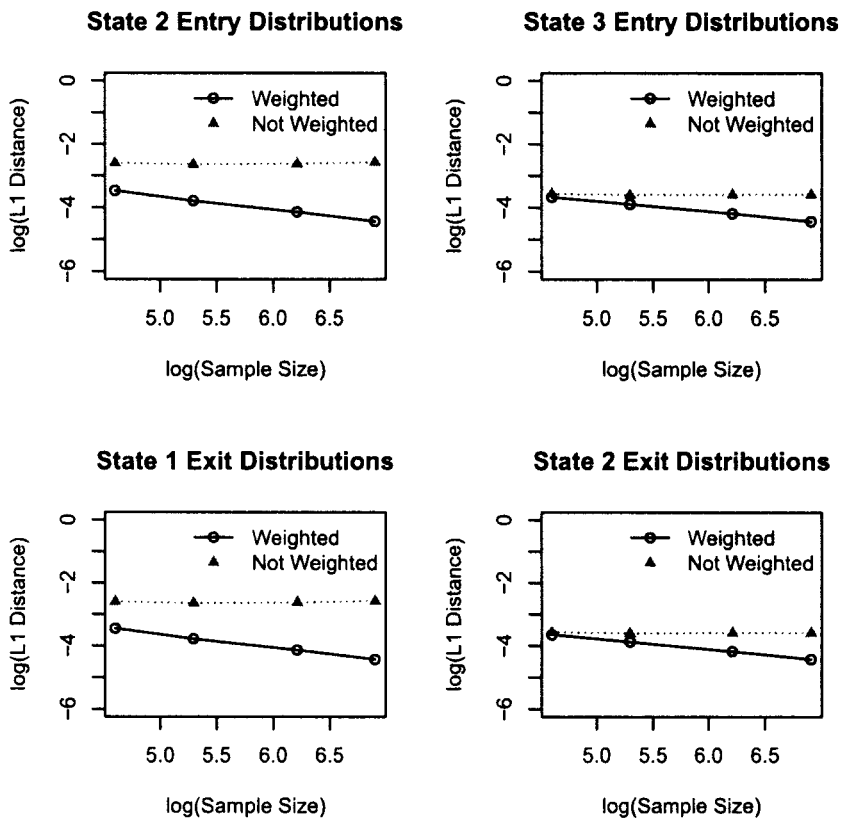

Figure 13. The log mean L_1 values of state entry and exit time distributions for the three-state tracking Markov model with lognormal waiting times and Weibull censoring times.

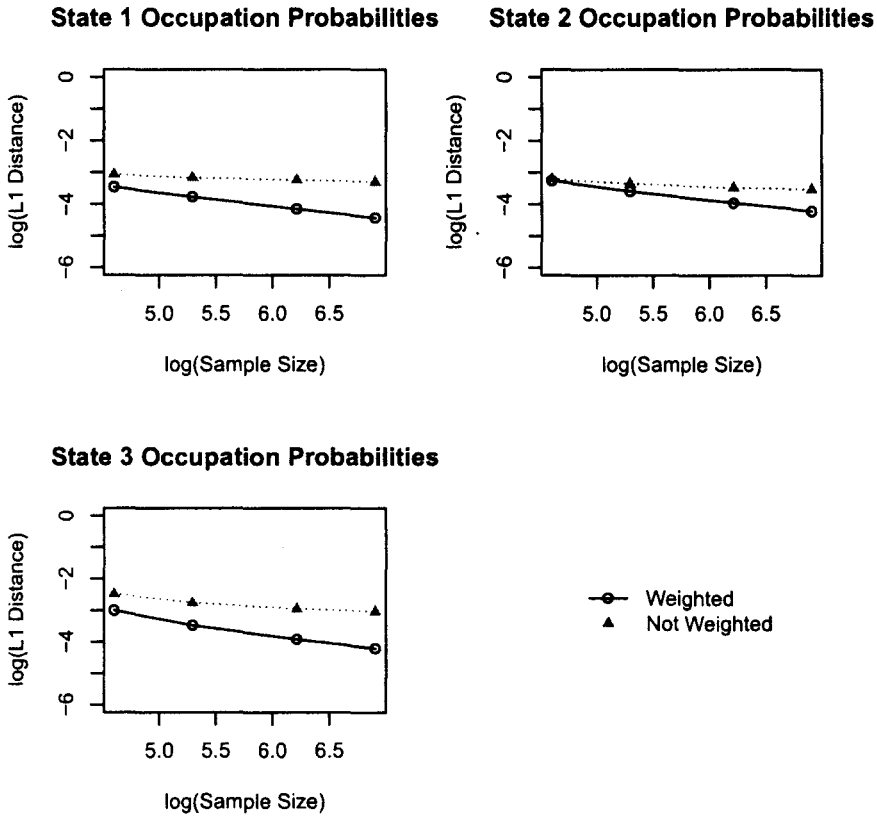

Figure 14. The log mean L_1 values of state occupation probabilities for the three-state tracking Markov model with lognormal waiting times and uniform censoring times.

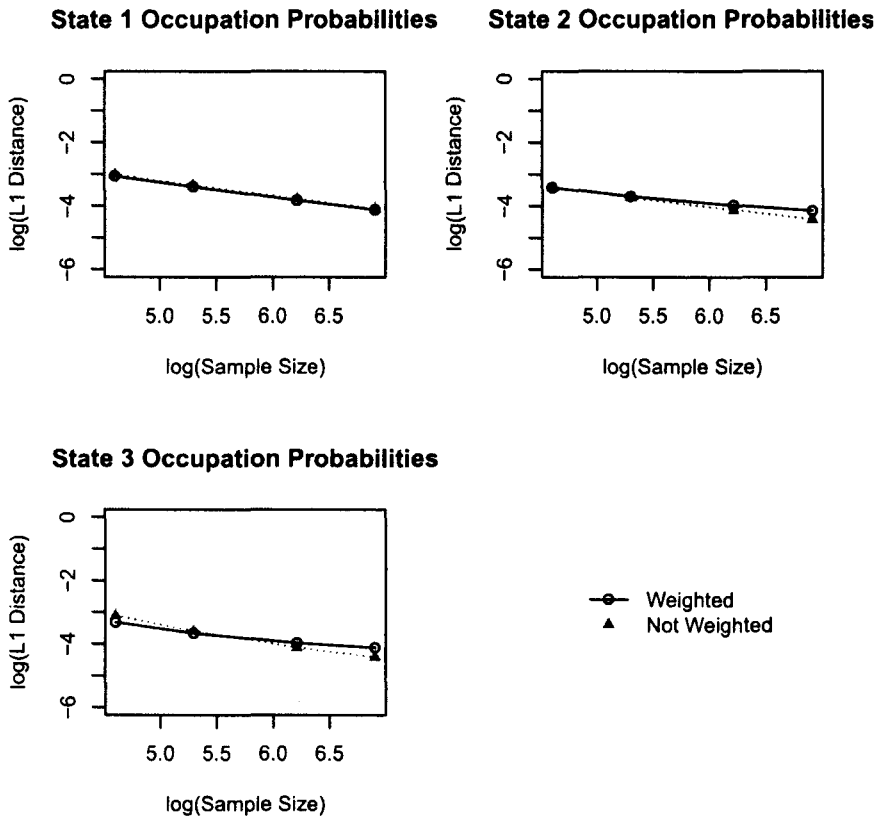

Figure 15. The log mean L_1 values of state occupation probabilities for the three-state tracking Markov model with Weibull waiting times and uniform censoring times.

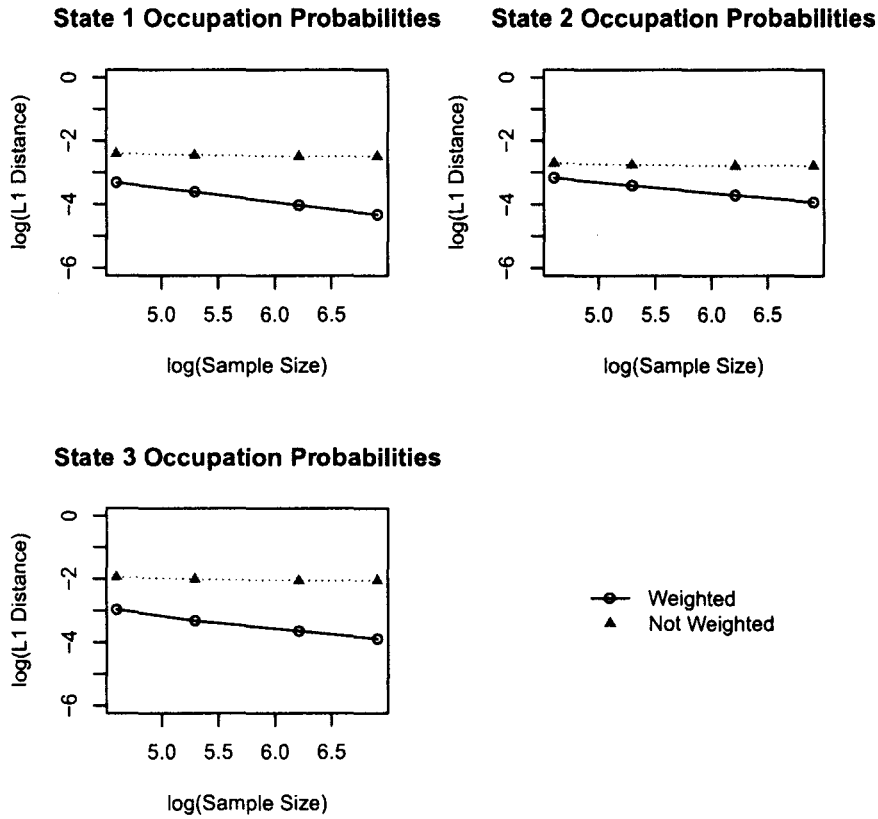

Figure 16. The log mean L_1 values of state occupation probabilities for the three-state tracking Markov model with lognormal waiting times and Weibull censoring times.

TABLE 1

The L_1 distances between estimators based on interval censored data and complete data in a three state tracking Markov model with Weibull state waiting times and Uniform censoring times. The estimates are based on a Monte Carlo sample size of 1000 for N = 100, 200, and 500. A Monte Carlo sample size of 100 was used for N = 1000. All standard errors were less than 0.0018.

	N=100		N=200		N=500		N=1000	
	W	NW	W	NW	W	NW	W	NW
P1	0.0467	0.0493	0.0335	0.0354	0.0218	0.0229	0.0163	0.0167
P2	0.0327	0.0320	0.0247	0.0236	0.0188	0.0160	0.0159	0.0121
P3	0.0359	0.0445	0.0255	0.0272	0.0188	0.0162	0.0162	0.0120
F2	0.0248	0.0269	0.0183	0.0189	0.0128	0.0127	0.0101	0.0096
F3	0.0275	0.0259	0.0212	0.0191	0.0170	0.0131	0.0152	0.0103

TABLE 2

The L_1 distances between estimators based on interval censored data and complete data in a three state tracking Markov model with Lognormal state waiting times and Uniform censoring times. The estimates are based on a Monte Carlo sample size of 1000 for N = 100, 200, and 500. A Monte Carlo sample size of 100 was used for N = 1000. All standard errors were less than 0.004.

	N=100		N=200		N=500		N=1000	
	W	NW	W	NW	W	NW	W	NW
P1	0.0312	0.0466	0.0229	0.0417	0.0157	0.0390	0.0119	0.0365
P2	0.0383	0.0405	0.0274	0.0347	0.0190	0.0306	0.0147	0.0292
P3	0.0495	0.0832	0.0308	0.0629	0.0198	0.0519	0.0148	0.0479
F2	0.0243	0.0378	0.0185	0.0358	0.0129	0.0351	0.0097	0.0334
F3	0.0376	0.0489	0.0273	0.0462	0.0192	0.0444	0.0148	0.0422

TABLE 3

The L_1 distances between estimators based on interval censored data and complete data in a three state tracking Markov model with Lognormal state waiting times and Weibull censoring time. The estimates are based on a Monte Carlo sample size of 1000 for N = 100, 200, and 500. A Monte Carlo sample size of 100 was used for N = 1000. All standard errors were less than 0.004.

	N=100		N=200		N=500		N=1000	
	W	NW	W	NW	W	NW	W	NW
P1	0.0360	0.0899	0.0269	0.0855	0.0177	0.0814	0.0132	0.0819
P2	0.0418	0.0661	0.0328	0.0628	0.0242	0.0607	0.0195	0.0612
P3	0.0509	0.1420	0.0358	0.1326	0.0258	0.1270	0.0202	0.1273
F2	0.0304	0.0777	0.0235	0.0756	0.0159	0.0737	0.0119	0.0748
F3	0.0460	0.1094	0.0357	0.1107	0.0265	0.1120	0.0210	0.1137

TABLE 4

The L_1 distances between estimators based on interval censored data and complete data in a three state tracking Markov model with Weibull state waiting times and Weibull censoring time. The estimates are based on a Monte Carlo sample size of 1000 for N = 100, 200, and 500. A Monte Carlo sample size of 100 was used for N = 1000. All standard errors were less than 0.0026.

	N=100		N=200		N=500		N=1000	
	W	NW	W	NW	W	NW	W	NW
P1	0.0738	0.0617	0.0589	0.0465	0.0451	0.0342	0.0386	0.0293
P2	0.0489	0.0374	0.0436	0.0293	0.0390	0.0234	0.0370	0.0208
P3	0.0596	0.0374	0.0536	0.0294	0.0485	0.0239	0.0465	0.0224
F2	0.0335	0.0320	0.0299	0.0256	0.0267	0.0228	0.0250	0.0222
F3	0.0521	0.0282	0.0499	0.0236	0.0470	0.0213	0.0457	0.0210

TABLE 5

The L_1 distances between estimators based on interval censored data and complete data in a three state tracking Semi-Markov model with Lognormal state waiting times and Uniform censoring time. The estimates are based on a Monte Carlo sample size of 1000 for N = 100, 200, and 500. A Monte Carlo sample size of 100 was used for N = 1000. All standard errors were less than 0.0028.

	N=100		N=200		N=500		N=1000	
	W	NW	W	NW	W	NW	W	NW
P1	0.0339	0.0467	0.0244	0.0417	0.0157	0.0390	0.0119	0.0365
P2	0.0284	0.0359	0.0196	0.0316	0.0190	0.0306	0.0147	0.0292
P3	0.0690	0.1041	0.0382	0.0657	0.0198	0.0519	0.0148	0.0479
G1	0.0250	0.0347	0.0190	0.0336	0.0129	0.0352	0.0097	0.0334
G2	0.0278	0.0352	0.0199	0.0324	0.0193	0.0447	0.0148	0.0423
F2	0.0249	0.0340	0.0189	0.0334	0.0129	0.0351	0.0097	0.0334
F3	0.0270	0.0345	0.0195	0.0318	0.0192	0.0444	0.0148	0.0422

TABLE 6

The L_1 distances between estimators based on interval censored data and complete data in a three state tracking Semi-Markov model with Weibull state waiting times and Uniform censoring time. The estimates are based on a Monte Carlo sample size of 1000 for N = 100, 200, and 500. A Monte Carlo sample size of 100 was used for N = 1000. All standard errors were less than 0.0018.

	N=100		N=200		N=500		N=1000	
	W	NW	W	NW	W	NW	W	NW
P1	0.0434	0.0493	0.0312	0.0354	0.0204	0.0229	0.0153	0.0167
P2	0.0395	0.0415	0.0297	0.0306	0.0223	0.0213	0.0185	0.0161
P3	0.0324	0.0374	0.0236	0.0251	0.0177	0.0166	0.0149	0.0124
G1	0.0230	0.0282	0.0170	0.0195	0.0120	0.0129	0.0095	0.0097
G2	0.0297	0.0317	0.0221	0.0232	0.0169	0.0158	0.0144	0.0119
F2	0.0227	0.0277	0.0169	0.0193	0.0119	0.0128	0.0095	0.0096
F3	0.0291	0.0304	0.0218	0.0226	0.0168	0.0156	0.0143	0.0117

TABLE 7

The L_1 distances between estimators based on interval censored data and complete data in a three state tracking Semi-Markov model with Lognormal state waiting times and Weibull censoring time. The estimates are based on a Monte Carlo sample size of 1000 for N = 100, 200, and 500. A Monte Carlo sample size of 100 was used for N = 1000. All standard errors were less than 0.0035.

	N=100		N=200		N=500		N=1000	
	W	NW	W	NW	W	NW	W	NW
P1	0.0368	0.0901	0.0260	0.0841	0.0178	0.0812	0.0133	0.0818
P2	0.0501	0.1158	0.0366	0.1047	0.0256	0.0985	0.0192	0.0972
P3	0.0261	0.0367	0.0194	0.0329	0.0144	0.0310	0.0114	0.0300
G1	0.0315	0.0738	0.0225	0.0708	0.0159	0.0722	0.0119	0.0755
G2	0.0259	0.0280	0.0207	0.0271	0.0153	0.0274	0.0119	0.0276
F2	0.0309	0.0738	0.0225	0.0708	0.0159	0.0722	0.0118	0.0755
F3	0.0253	0.0280	0.0203	0.0271	0.0152	0.0274	0.0119	0.0276

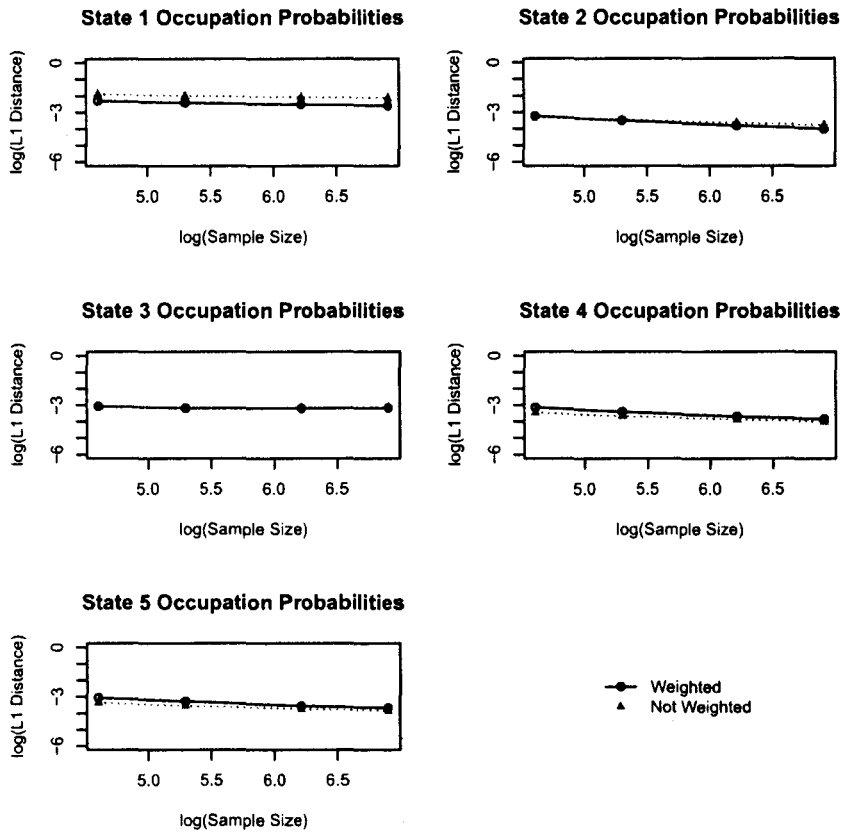

Figure 17. The log mean L_1 values of state occupation probabilities for the five-state branching Markov model with lognormal waiting times and uniform censoring times.

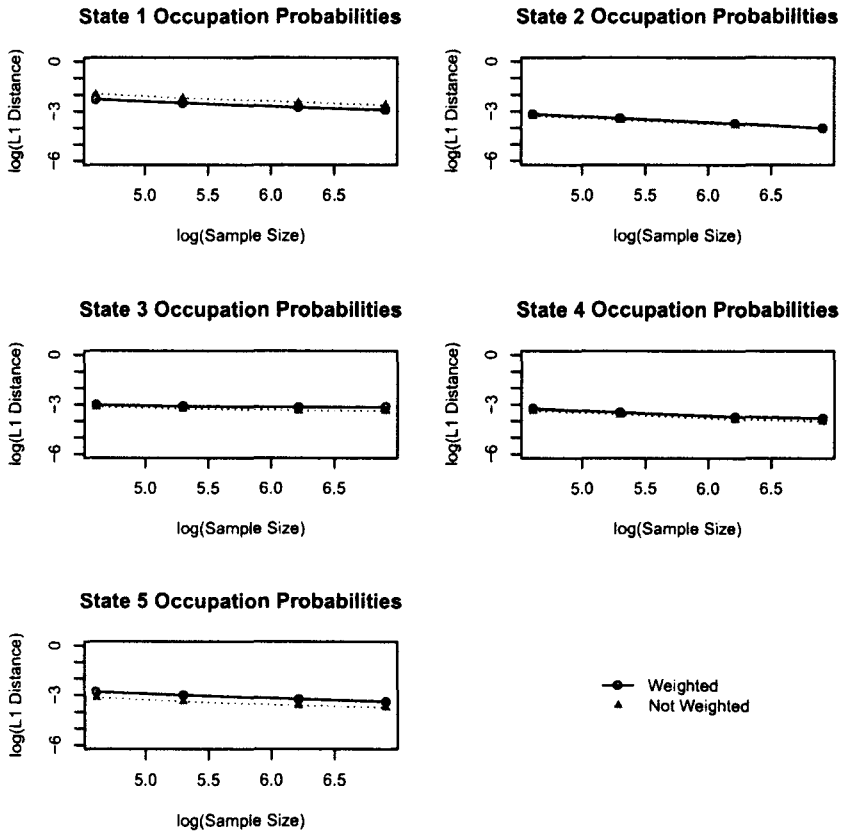

Figure 18. The log mean L_1 values of state occupation probabilities for the five-state branching Markov model with Weibull waiting times and uniform censoring times.

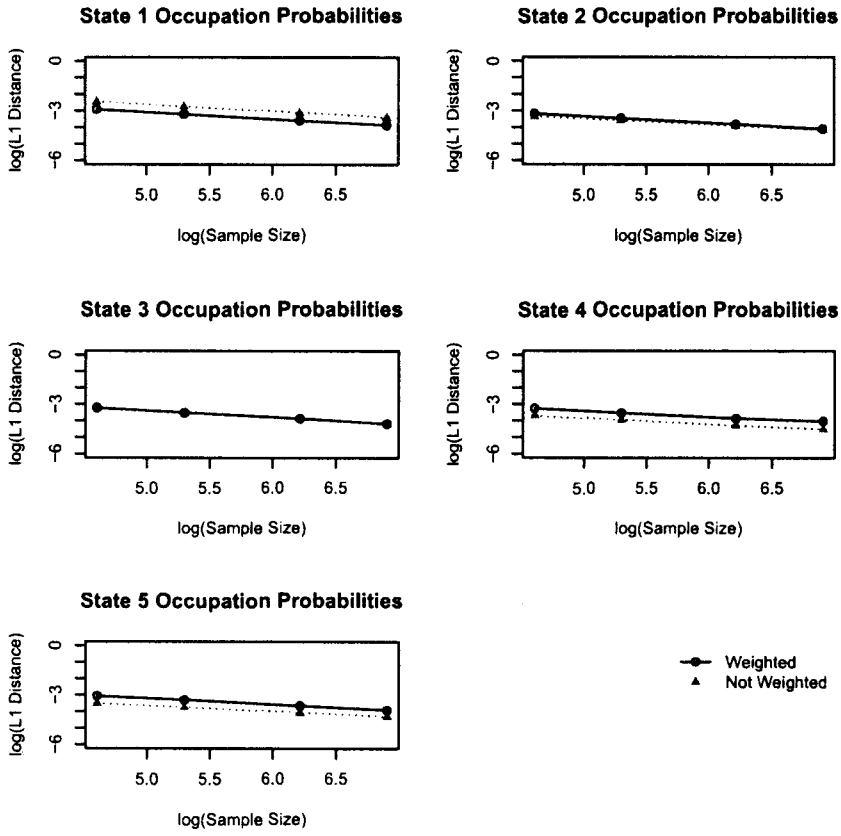

Figure 19. The log mean L_1 values of state occupation probabilities for the five-state branching Markov model with lognormal waiting times and Weibull censoring times.

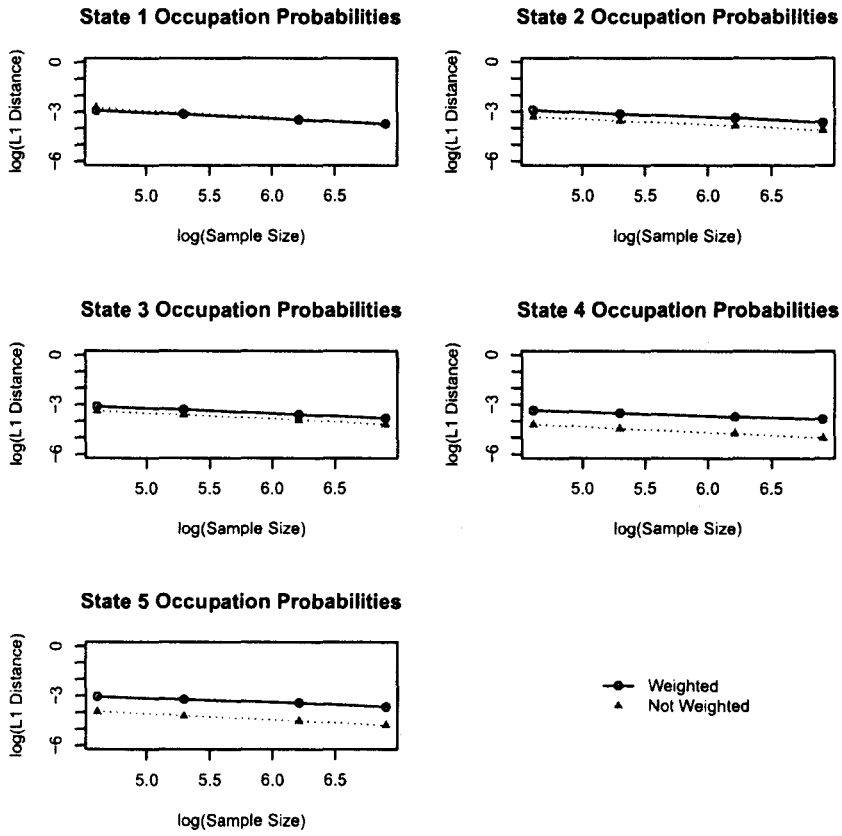

Figure 20. The log mean L_1 values of state occupation probabilities for the five-state branching Markov model with Weibull waiting times and Weibull censoring times.

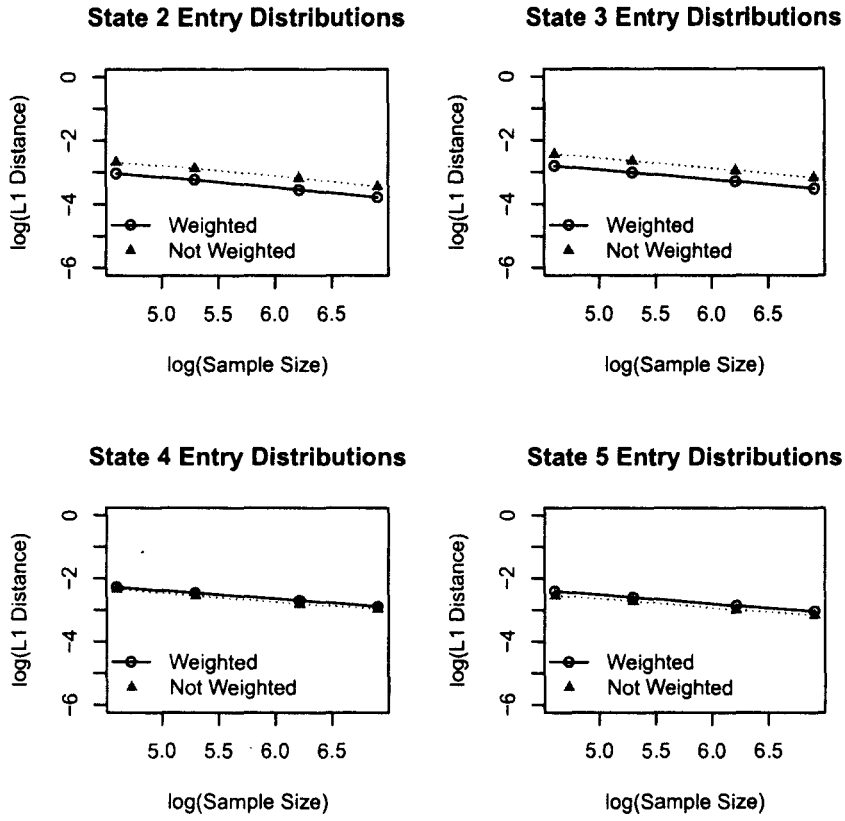

Figure 21. The log mean L_1 values of state entry time distributions for the five-state branching Markov model with lognormal waiting times and Weibull censoring times.

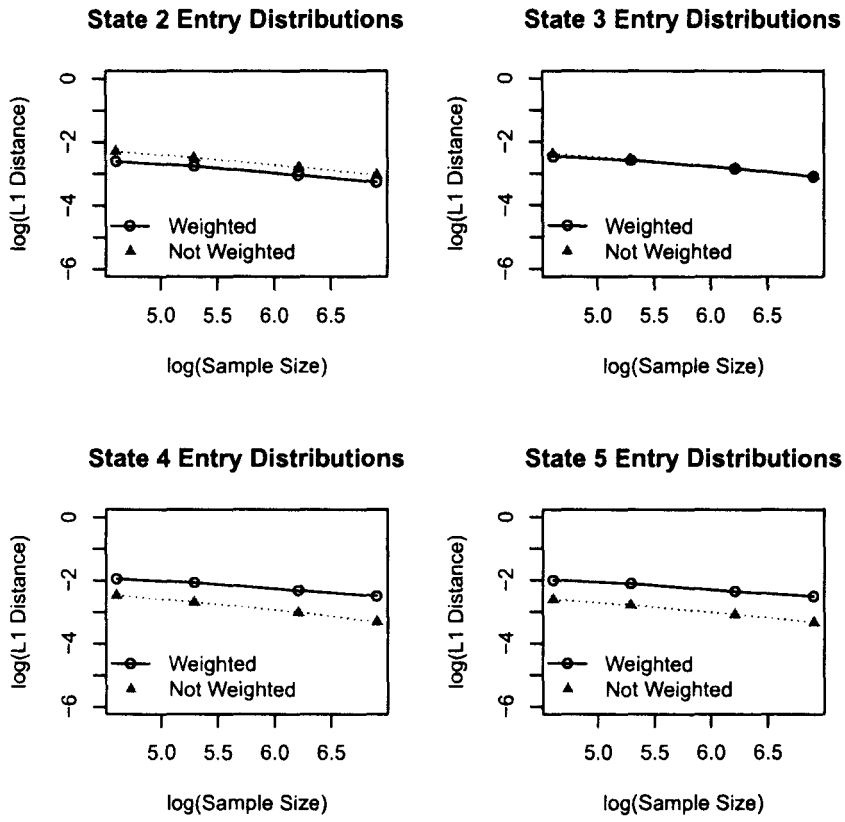

Figure 22. The log mean L_1 values of state entry time distributions for the five-state branching Markov model with Weibull waiting times and Weibull censoring times.

TABLE 8

The L_1 distances between estimators based on interval censored data and complete data in a three state tracking Semi-Markov model with Weibull state waiting times and Weibull censoring time. The estimates are based on a Monte Carlo sample size of 1000 for N = 100, 200, and 500. A Monte Carlo sample size of 100 was used for N = 1000. All standard errors were less than 0.0026.

	N=100		N=200		N=500		N=1000	
	W	NW	W	NW	W	NW	W	NW
P1	0.0701	0.0617	0.0556	0.0465	0.0422	0.0342	0.0359	0.0293
P2	0.0571	0.0422	0.0505	0.0341	0.0465	0.0283	0.0439	0.0257
P3	0.0419	0.0232	0.0367	0.0171	0.0344	0.0125	0.0328	0.0104
G1	0.0318	0.0376	0.0282	0.0277	0.0251	0.0208	0.0236	0.0188
G2	0.0359	0.0266	0.0329	0.0177	0.0327	0.0111	0.0320	0.0086
F2	0.0316	0.0371	0.0281	0.0275	0.0251	0.0208	0.0235	0.0188
F3	0.0352	0.0254	0.0326	0.0170	0.0326	0.0109	0.0320	0.0085

TABLE 9

The L_1 distances between estimators based on interval censored data and complete data in a five state branching Markov model with lognormal state waiting times and uniform censoring time. The estimates are based on a Monte Carlo sample size of 1000 for N = 100, 200, and 500. A Monte Carlo sample size of 100 was used for N = 1000. All standard errors were less than 0.0036.

	N=100		N=200		N=500		N=1000	
	W	NW	W	NW	W	NW	W	NW
P1	0.0995	0.1490	0.0885	0.1343	0.0784	0.1228	0.0724	0.1164
P2	0.0387	0.0380	0.0299	0.0312	0.0218	0.0259	0.0178	0.0224
P3	0.0452	0.0446	0.0409	0.0421	0.0402	0.0422	0.0412	0.0437
P4	0.0426	0.0308	0.0328	0.0250	0.0245	0.0203	0.0211	0.0177
P5	0.0465	0.0343	0.0376	0.0281	0.0284	0.0231	0.0253	0.0207
F2	0.0409	0.0568	0.0346	0.0477	0.0294	0.0410	0.0268	0.0383
F3	0.0678	0.0942	0.0694	0.0994	0.0738	0.1112	0.0770	0.1202
F4	0.0706	0.0756	0.0600	0.0653	0.0470	0.0548	0.0406	0.0471
F5	0.0624	0.0652	0.0531	0.0558	0.0405	0.0469	0.0338	0.0412

TABLE 10

The L_1 distances between estimators based on interval censored data and complete data in a five state branching Markov model with Weibull state waiting times and uniform censoring time. The estimates are based on a Monte Carlo sample size of 1000 for N = 100, 200, and 500. A Monte Carlo sample size of 100 was used for N = 1000. All standard errors were less than 0.0041.

	N=100		N=200		N=500		N=1000	
	W	NW	W	NW	W	NW	W	NW
P1	0.1003	0.1384	0.0808	0.1094	0.0627	0.0842	0.0534	0.0716
P2	0.0414	0.0370	0.0324	0.0287	0.0234	0.0213	0.0176	0.0177
P3	0.0481	0.0422	0.0448	0.0373	0.0420	0.0337	0.0427	0.0333
P4	0.0388	0.0336	0.0309	0.0267	0.0237	0.0200	0.0217	0.0176
P5	0.0600	0.0423	0.0497	0.0335	0.0389	0.0260	0.0334	0.0233
F2	0.0422	0.0561	0.0364	0.0475	0.0306	0.0399	0.0296	0.0386
F3	0.0570	0.0739	0.0530	0.0678	0.0478	0.0606	0.0458	0.0581
F4	0.0669	0.0700	0.0561	0.0582	0.0466	0.0471	0.0399	0.0416
F5	0.0571	0.0584	0.0484	0.0501	0.0402	0.0418	0.0359	0.0397

TABLE 11

The L_1 distances between estimators based on interval censored data and complete data in a five state branching Markov model with lognormal state waiting times and Weibull censoring time. The estimates are based on a Monte Carlo sample size of 1000 for N = 100, 200, and 500. A Monte Carlo sample size of 100 was used for N = 1000. All standard errors were less than 0.0042.

	N=100		N=200		N=500		N=1000	
	W	NW	W	NW	W	NW	W	NW
P1	0.0533	0.0855	0.0402	0.0639	0.0273	0.0428	0.0208	0.0324
P2	0.0408	0.0346	0.0311	0.0276	0.0215	0.0198	0.0164	0.0155
P3	0.0395	0.0400	0.0295	0.0294	0.0209	0.0203	0.0152	0.0152
P4	0.0381	0.0250	0.0289	0.0191	0.0209	0.0136	0.0178	0.0109
P5	0.0468	0.0296	0.0366	0.0232	0.0255	0.0166	0.0198	0.0131
F2	0.0478	0.0682	0.0392	0.0560	0.0287	0.0412	0.0227	0.0316
F3	0.0603	0.0872	0.0488	0.0697	0.0369	0.0521	0.0296	0.0413
F4	0.1020	0.0945	0.0851	0.0770	0.0668	0.0592	0.0551	0.0504
F5	0.0896	0.0775	0.0741	0.0647	0.0567	0.0494	0.0478	0.0418

TABLE 12

The L_1 distances between estimators based on interval censored data and complete data in a five state branching Markov model with Weibull state waiting times and Weibull censoring time. The estimates are based on a Monte Carlo sample size of 1000 for N = 100, 200, and 500. A Monte Carlo sample size of 100 was used for N = 1000. All standard errors were less than 0.0072.

	N=100		N=200		N=500		N=1000	
	W	NW	W	NW	W	NW	W	NW
P1	0.0537	0.0648	0.0424	0.0471	0.0301	0.0321	0.0235	0.0245
P2	0.0539	0.0356	0.0428	0.0277	0.0338	0.0208	0.0256	0.0161
P3	0.0439	0.0334	0.0364	0.0265	0.0271	0.0193	0.0218	0.0149
P4	0.0351	0.0148	0.0297	0.0116	0.0237	0.0086	0.0209	0.0067
P5	0.0467	0.0187	0.0393	0.0146	0.0317	0.0107	0.0250	0.0084
F2	0.0724	0.0996	0.0628	0.0822	0.0471	0.0604	0.0380	0.0473
F3	0.0842	0.0915	0.0751	0.0774	0.0576	0.0588	0.0449	0.0456
F4	0.1396	0.0826	0.1244	0.0667	0.0973	0.0484	0.0814	0.0358
F5	0.1342	0.0728	0.1209	0.0612	0.0944	0.0452	0.0807	0.0354

TABLE 13

The L_1 distances between estimators based on interval censored data and complete data in a five state branching semi-Markov model with lognormal state waiting times and uniform censoring time. The estimates are based on a Monte Carlo sample size of 1000 for N = 100, 200, and 500. A Monte Carlo sample size of 100 was used for N = 1000. All standard errors were less than 0.003.

	N=100		N=200		N=500		N=1000	
	W	NW	W	NW	W	NW	W	NW
P1	0.0835	0.1243	0.0750	0.1118	0.0671	0.1021	0.0630	0.0974
P2	0.0340	0.0447	0.0282	0.0390	0.0232	0.0341	0.0219	0.0325
P3	0.0866	0.0759	0.0858	0.0749	0.0866	0.0758	0.0877	0.0770
P4	0.0303	0.0278	0.0272	0.0267	0.0284	0.0273	0.0315	0.0280
P5	0.0526	0.0352	0.0503	0.0365	0.0501	0.0385	0.0493	0.0388
G1	0.0326	0.0452	0.0308	0.0430	0.0290	0.0418	0.0751	0.1153
G2	0.0304	0.0389	0.0252	0.0346	0.0231	0.0328	0.0225	0.0318
F2	0.0469	0.0669	0.0442	0.0645	0.0403	0.0607	0.0394	0.0601
F3	0.0943	0.1363	0.1022	0.1506	0.1063	0.1611	0.1078	0.1660
F4	0.0507	0.0578	0.0402	0.0474	0.0323	0.0408	0.0287	0.0386
F5	0.0388	0.0491	0.0317	0.0421	0.0261	0.0378	0.0230	0.0335

TABLE 14

The L_1 distances between estimators based on interval censored data and complete data in a five state branching semi-Markov model with Weibull state waiting times and uniform censoring time. The estimates are based on a Monte Carlo sample size of 1000 for N = 100, 200, and 500. A Monte Carlo sample size of 100 was used for N = 1000. All standard errors were less than 0.0041.

	N=100		N=200		N=500		N=1000	
	W	NW	W	NW	W	NW	W	NW
P1	0.1003	0.1384	0.0808	0.1094	0.0627	0.0842	0.0534	0.0716
P2	0.0414	0.0370	0.0324	0.0287	0.0234	0.0213	0.0176	0.0177
P3	0.0481	0.0422	0.0448	0.0373	0.0420	0.0337	0.0427	0.0333
P4	0.0388	0.0336	0.0309	0.0267	0.0237	0.0200	0.0217	0.0176
P5	0.0600	0.0423	0.0497	0.0335	0.0389	0.0260	0.0334	0.0233
G1	0.0355	0.0432	0.0312	0.0371	0.0614	0.0809	0.0239	0.0290
G2	0.0474	0.0481	0.0399	0.0401	0.0337	0.0337	0.0314	0.0332
F2	0.0422	0.0561	0.0364	0.0475	0.0306	0.0399	0.0296	0.0386
F3	0.0570	0.0739	0.0530	0.0678	0.0478	0.0606	0.0458	0.0581
F4	0.0669	0.0700	0.0561	0.0582	0.0466	0.0471	0.0399	0.0416
F5	0.0571	0.0584	0.0484	0.0501	0.0402	0.0418	0.0359	0.0397

TABLE 15

The L_1 distances between estimators based on interval censored data and complete data in a five state branching semi-Markov model with lognormal state waiting times and Weibull censoring time. The estimates are based on a Monte Carlo sample size of 1000 for N = 100, 200, and 500. A Monte Carlo sample size of 100 was used for N = 1000. All standard errors were less than 0.0042.

	N=100		N=200		N=500		N=1000	
	W	NW	W	NW	W	NW	W	NW
P1	0.0533	0.0855	0.0402	0.0639	0.0273	0.0428	0.0208	0.0324
P2	0.0408	0.0346	0.0311	0.0276	0.0215	0.0198	0.0164	0.0155
P3	0.0395	0.0400	0.0295	0.0294	0.0209	0.0203	0.0152	0.0152
P4	0.0381	0.0250	0.0289	0.0191	0.0209	0.0136	0.0178	0.0109
P5	0.0468	0.0296	0.0366	0.0232	0.0255	0.0166	0.0198	0.0131
G1	0.0340	0.0501	0.0261	0.0381	0.0176	0.0254	0.0687	0.1086
G2	0.0579	0.0574	0.0451	0.0454	0.0337	0.0334	0.0273	0.0263
F2	0.0478	0.0682	0.0392	0.0560	0.0287	0.0412	0.0227	0.0316
F3	0.0603	0.0872	0.0488	0.0697	0.0369	0.0521	0.0296	0.0413
F4	0.1020	0.0945	0.0851	0.0770	0.0668	0.0592	0.0551	0.0504
F5	0.0896	0.0775	0.0741	0.0647	0.0567	0.0494	0.0478	0.0418

TABLE 16

The L_1 distances between estimators based on interval censored data and complete data in a five state branching semi-Markov model with Weibull state waiting times and Weibull censoring time. The estimates are based on a Monte Carlo sample size of 1000 for N = 100, 200, and 500. A Monte Carlo sample size of 100 was used for N = 1000. All standard errors were less than NA.

	W	NW	W	NW	W	NW	W	NW
P1	0.0537	0.0648	0.0424	0.0471	0.0301	0.0322	0.0219	0.0242
P2	0.0523	0.0352	0.0428	0.0277	0.0332	0.0206	0.0260	0.0161
P3	0.0439	0.0334	0.0364	0.0265	0.0271	0.0193	0.0218	0.0150
P4	0.0354	0.0149	0.0300	0.0117	0.0240	0.0086	0.0192	0.0068
P5	0.0459	0.0186	0.0387	0.0144	0.0319	0.0107	0.0252	0.0084
G1	0.0471	0.0633	0.0396	0.0499	0.0293	0.0364	0.0218	0.0279
G2	0.0892	0.0551	0.0761	0.0427	0.0570	0.0304	0.0420	0.0241
F2	0.0724	0.0996	0.0626	0.0804	0.0472	0.0604	0.0362	0.0471
F3	0.0842	0.0915	0.0751	0.0774	0.0576	0.0587	0.0431	0.0456
F4		0.0815	0.1231	0.0659	0.0960	0.0487	0.0716	0.0337
F5	0.1348	0.0754	0.1171	0.0597	0.0922	0.0452	0.0745	0.0357

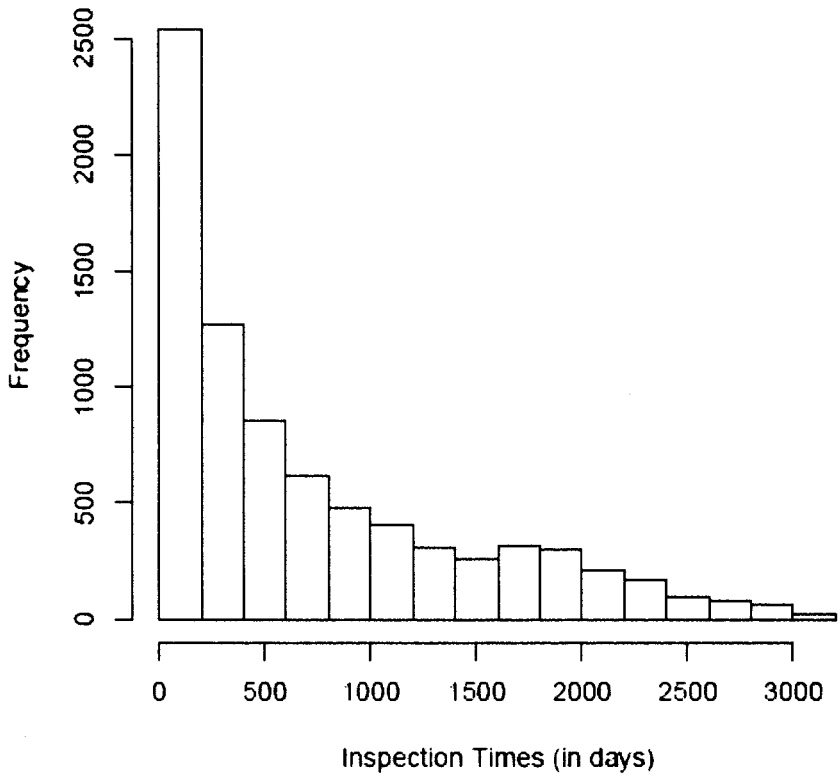

Figure 23. Distribution of inspection times for patients on the UNOS liver transplant waitlist.

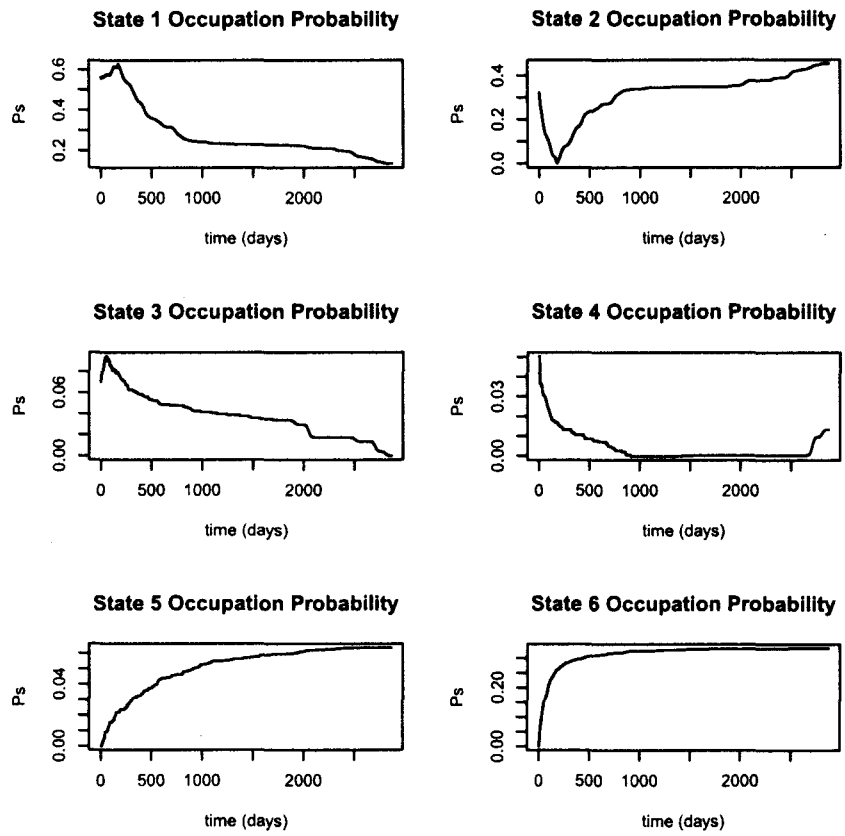

Figure 24. State occupation probabilitiess for levels of MELD scores for patients on the UNOS liver transplant waiting list.

CHAPTER IV

CONCLUSIONS AND FUTURE WORK

The primary focus of this dissertation has been on nonaparametric estimation of multistate data subject to right censoring and interval censoring. We created an R package, **msSurv**, to calculate and display key marginal estimators for data subject to independent or dependent censoring. For interval censored data, we developed nonparametric estimators of state occupation probabilities, state entry time distributions, and state exit time distributions based on product limit estimation. Simulations and real data analysis were performed in both cases and showed that the methods proposed are reasonable and that they can be implemented.

Our future research for the **msSurv** package includes incorporating waiting time distribution computations for right censored data, as well as modifying the current methods to accurately estimate state entry and exit time distributions for recurrent event data. We will also conduct research to extend **msSurv** to incorporate nonparametric estimation for current status data. Implementation of estimation methods for current status data is considerably more complicated than that of the right censored data since actual transition times are not known.

Future research for interval censored data includes developing estimators for state waiting time distributions and L_1 testing methods using these proposed procudes and estimators, as well as investigating a more general weight matrix to further improve efficiency of the estimation. The computations using the general weighting matrix are more complicated than those for the diagonal variance matrix. Ultimately, we would like to also include the interval censored estimation in the **msSurv** package as well, which will require an investigation into an appropriate

general bandwidth.

Future research plans for the **msSurv** package are presented in more detail in section A. We describe the future research plans for estimating the waiting time distribution for interval censored data in section B and performing L_1 tests in section D. Then we discuss further generalization of the proposed procedures through a general weighting matrix in section C.

A msSurv package expansion

msSurv is a comprehensive R package for nonparametric estimation of a general multistate model subject to right censoring and possibly left truncation. The package computes the transition probabilities for a Markov model and offers estimates of state occupation probabilities and stat entry and exit time distributions which were previously unavailable in any R package. **msSurv** produces accurate estiamtes for both independent and state dependent censoring, the latter of which was previously unavailable in other packages. **msSurv** provides functions that print, summarize, and display plots of the estimates and corresponding estimates.

Though the package is very thorough for right censored data, extension to include state waiting time distributions as described in Satten and Datta (2002) is desireable and currently unavailable. State waiting (sojourn) times can be defined as $W_i^j = V_i^j - U_i^j$, where U_i^j and V_i^j represent the state entry and exit times respectively. Waiting times W_i^j are calculated from right censored data when the censoring time is larger than the state exit time ($C_i \geq V_i^j$).

Satten and Datta (2002) estimate the counting processes for waiting times in state j as a jump process with jump size equal to

$$\Delta \widehat{N}_j^W(t) = \sum_{i=1}^{n} \frac{I\{W_i^j = t, C_i \geq V_i^j\}}{\widehat{K}_i(V_i^j -)} \tag{19}$$

where $\widehat{K}(t) = exp\{-\widehat{\Lambda}_c(t|\overline{Z}(t)\}$ is estimated as before. The estimated "at risk" set

for state j waiting times is defined by

$$\widehat{Y}_j^W(t) = \sum_{i=1}^{n} \frac{I\{W_i^j \geq t, C_i \geq t + U_i^j\}}{\widehat{K}_i((t + U_i^j)-)} \tag{20}$$

Then, the state waiting time distribution, denoted as $H^j(t) = Pr\{W^j \leq t | V_j < \infty\}$, is estimated by

$$\widehat{H}^j(t) = 1 - \prod_{s \leq t} \left(1 - \frac{\Delta \widehat{N}_j^W(ds)}{\widehat{Y}_j^W(s)}\right).$$

which is essentially a Kaplan-Meier type product limit formula using the estimators in 19 and 20.

These estimators are valid without the Markovity assumption and may include state dependent censoring through the use of reweighting based on estimation of the censoring hazard.

Waiting time distributions are more challenging than state entry or exit time distributions in that state waiting times are measured in time since state entry instead of calendar time, so we will need to incorporate functions to measure waiting times, compute the waiting time counting processes and "at risk" sets, as well as a function to compute the waiting time distribution for any model.

Another useful extension to **msSurv** for right censored data is incorporating estimation for cyclic models where individuals pass through state j more than one time. For handling situations with repeated events, we will add code to internally expand the system to include additional states to track the different recurrent transitions into a given state j. We will need to incorporate a method for combining these expanded counting processes and "at risk" set to accurately calculate the state entry, exit, and waiting time distributions for any general recurrent model. This process will involve a lot of so called bookkeeping and properly indexing for general models.

We will also conduct future research to extend **msSurv** to include nonparametric estimation of state occupation probabilities and state entry and exit

time distributions for current status data found in Datta and Sundaram (2006), Datta *et al.* (2009), and Lan and Datta (2010b). Estimation for current status data is much more difficult than that for right censored data because actual transition (event) times are not known. Let C_i denote the random inspection time for individual i and S_i denote the corresponding state information at time C_i. Datta and Sundaram (2006) defined the counting process of j to j' transitions for current status data as

$$\widehat{N}_{jj'}(t) = \frac{\sum_{i=1}^{n} \widehat{N}_{jj'}^{P}(C_i) K_h(C_i - t)}{\sum_{i=1}^{n} K_h(C_i - t)}$$

where $\widehat{N}_{jj'}^{P}(C_i)$ denotes the smoothed PAV estimator of the counting process and K is a density kernel defined as $K_h(\cdot) = h^{-1}K(\cdot/h)$ with bandwidth $h = h(n)$. Note that $\widehat{N}_{jj'}^{P}(C_i)$ is obtained by performing isotonic regression on the pairs $I(U_{jj'} \leq C)$ on C based on the pairs $(C_i, I(U_{jj',i} \leq C_i))$ using the PAV algorithm followed by kernel smoothing where $U_{jj'}$ denotes the (unobserved) transition time of an individual from state j to j'.

The "at risk" set of transitions out of state j does not have to be monotonic and therefore can be estimated using kernel smoothing through the function K previously described. Therefore, the "at risk" set is defined as

$$\widehat{Y}(t) = \frac{\sum_{i=1}^{n} I(S_i(C_i) = j) K_h(C_i - t)}{n^{-1}\sum_{i=1}^{n} K_h(C_i - t)}$$

where K is described above.

State occupation probabilities will be computed using the special case $\widehat{P}(0,t) = \prod_{(0,t]}(I + d\widehat{\boldsymbol{A}}(u))$ of the Aalen-Johansen estimator formula

$$\widehat{A}_{jj'}(t) = \begin{cases} \int_0^t J_j(u)\widehat{Y}_j(u)^{-1} d\widehat{N}_{jj'}(u) & j \neq j' \\ -\sum_{j' \neq j} \widehat{A}_{jj'}(t) & j = j', \end{cases}$$

where $J_j(u) = I(\widehat{Y}_j(u) > 0)$ and the integrated conditional transition hazards are now calculated using $\widehat{N}$ and $\widehat{Y}$ defined by Datta and Sundaram (2006).

We will update the **msSurv** package so that a user specifies the type of censoring, e.g., "current status data", and then the package with calculate the appropriate counting process. We will investigate using an available R packages to perform the isotonic regression (e.g., **isotone**) and kernel smoothing (e.g., **KernSmooth**) to estimate the counting process and "at risk" sets. One potential problem is finding an appropriate bandwidth for any general framework, as the estimate may be very sensitive to bandwidths. We will update the current framework of state occupation probability estimation in **msSurv** to use the appropriate counting process and "at risk" set estimators for the user specified censoring scheme.

We would like to ultimately extend the package to include nonparametric estimation of interval censored data. In fact, we had this in mind as we coded for the interval censored estimation for this dissertation research. The initial non-weighted isotonic regression fit and subsequent smoothing are already generalized. We will face challenges in efficiently generalizing the inversion of the variance-covariance matrix for all the transitions in the system, as the computation can be extremely time consuming and require a lot of memory usage.

B Interval censored data

In this dissertation research we extended the methods of Datta and Sundaram (2006) and Lan and Datta (2010b) to interval censored data where inspection times for individuals may be dependent. We ignored the dependencies, pooled the times, and then performed isotonic regression followed by kernel smoothing to get initial estimates of the counting process. We then calculated the diagonal variance estimates using those initial state occupation probabilities to use as a weight for a weighted isotonic regression to improve efficiency of our estimates. We applied

kernel smoothing to the resulting estimates to obtain our counting process estimates.

Future research will be conducted to find the state waiting time distributions for interval censored data. Estimation of these distributions are difficult since the exact entry and exit times of an individual are never observed. We plan to extend the work of Datta *et al.* (2009), who obtained estimates of state waiting time distributions for any acyclic Markov multistate model subject to current status data. For the sake of completeness we will present a brief description of their estimators.

Let C_i denote the inspection time for individual i and let S_i denote the corresponding state information. Suppose X_j denotes the (possiby unobserved) indicator that individual i ever enters state j. Then, let U_j, V_j, and $W_j = V_j - U_j$ denote the entry, exit, and waiting times, respectively, for individuals who ever enter state j. Then, denote the corresponding distribution functions as F_j, G_j, and F^{W_j}. Datta *et al.* (2009) define the state waiting time distribution function for a state j as

$$\widehat{F}^{W_j}(t) = 1 - \int_0^{\infty} \left\{ exp\left(- \int_n^{u+t} \frac{d\widehat{N}_{j\cdot}(s)}{\widehat{Y}_j(s)} \right) \right\},$$

where $\widehat{F}_j(u) = \frac{1-exp\left\{ - \int_0^u \frac{d\widehat{N}_{\cdot j}(s)}{\widehat{Y}_j(s)} \right\}}{1-exp\left\{ - \int_0^{\infty} \frac{d\widehat{N}_{\cdot j}(s)}{\widehat{Y}_j(s)} \right\}}$. Note that $\widehat{N}_{\cdot j}(t)$, $\widehat{N}_{j\cdot}(t)$, $\widehat{Y}_{\cdot j}(t)$, and $\widehat{Y}_{\cdot j}(t)$ denote estimators based on current status data.

Calculation of state waiting time distributions with current status data poses additional difficulty, since we cannot directly regress the indicators of events involving the waiting times because the state entry times are also unknown. Some progress can be made with additional structural assumptions. As for example, under the Markov assumption (Datta *et al.*, 2009), we could obtain the following identity

$$H^j(t) = 1 - \int_0^{\infty} \prod_{u<s\leq u+t} (1 - d\Lambda_{j\bullet}(s))\, dF^j(u),\ t \geq 0,$$

where $\Lambda_{j\bullet}$ is integrate transition hazard out of state j. Using this and the quantities defined earlier we obtain a non-parametric regression estimator of the state waiting

time survival function

$$H^j(t) = 1 - \int_0^\infty \left\{ \prod_{u<s\leq u+t} \left(1 - \frac{d\widehat{N}_{j\bullet}(s)}{\widehat{Y}_j(s)} \right) \right\} d\widehat{F}^j(u),\ t \geq 0.$$

For interval censored data, we will assume a Markov model since an individual's entry and exit times are only known to fall within a certain interval. We will investigate counting and at risk processes in terms of both entry and exit times, specifically calculating $\widehat{N}_{\cdot j}(t)$, $\widehat{N}_{j\cdot}(t)$, $\widehat{Y}_{\cdot j}(t)$, and $\widehat{Y}_{\cdot j}(t)$ based on the estimators proposed in this research. We will investigate how to effectively measure the waiting time distributions since they are typically measured in time since state entry and those entry times are not known.

C General weighting matrix

In this dissertation research we ran a weighted isotonic regression on indicators of whether an individual made a transition by some time C using the diagonal variance matrix as weights. In future research we would like to find a general weighted matrix, say W, to further improve efficiency, e.g. minimize $\sum_{i=1}^n \Delta_i^T \widehat{W}_i \Delta_i$ subject to $\widehat{P}\left\{U_{jj'} \leq c_{(i)}\right\} \leq \ldots \leq \widehat{P}\left\{U_{jj'} \leq c_{(n)}\right\}$ where $c_{(i)} \leq \ldots \leq c_{(n)}$ are the ordered inspection times in the pooled sample where $\Delta_i = (\widehat{P}\left\{U_{jj'} \leq c_{ik}\right\} - I\left\{U_{jj'} \leq c_{ik}\right\}, 1 \leq k \leq n_i)$.

One possible choice for W is Σ^{-1} where Σ denotes the full variance-covariance matrix defined as

$$\sigma_{i,kk'}^{jj'} = \begin{cases} P\left\{U_{jj'} \leq C_{ik}\right\}\left(1 - P\left\{U_{jj'} \leq C_{ik}\right\}\right) & k = k' \\ P\left\{U_{jj'} \leq C_{ik}\right\} - P\left\{U_{jj'} \leq C_{ik}\right\} P\left\{U_{jj'} \leq C_{ik'}\right\} & k \neq k' \end{cases}$$

The inverse of the resulting W could then be an isotonic weighted regression performed using an R package such as **isotone**. Results of the regression would then be smoothed using a kernel smoother as before. In preliminary work, we were able

to successfully estimate the variance-covariance matrix, but had some problems with some fits of the regression becoming negative. Research will be done on adding constraints so that the resulting probabilities remain between 0 and 1.

State occupation probabilities and state entry, exit, and waiting time distributions will then be computed using the new estimators and their performace will be evaluated through simulation studies and computation of L_1 distances.

D L_1 tests

Another future research area will be constructing L_1 hypothesis testing procedures for comparing two (or more) groups using the nonparametric estimators developed in this dissertation for interval censored data. These types of tests would be useful in practice as investigators seek to compare the state occupation, entry and exit times in two or more groups (e.g., comparing the state occupation probabilities between genders).

Lan and Datta (2010a) obtained generalized testing procedures for current status data in multistate models with a Markovian framework using a distance-based bootstrap test. They assumed the multistate system had a directed tree structure so that every state j in the system is reached by a unique path. They assume that inspection times and state occupation status for are independently and identically distributed within each group and that censoring times are random in each group so that the the censoring time C_i is independent of the state occupied at that time $S_i(C_i)$ for individual i. For sake of completeness, we will provide a description of the testing process.

Suppose two groups are independent and $\theta = \theta(t)$ are the marginal function quantities to estimate, e.g., $\theta = P_j(t)$ for state occupation probabilities, $\theta = F_j$ for state entry time distribution. etc. The null hypothesis for testing is of the form

$$H_0 : \theta^1(t) = \theta^2(t), 0 \leq t \leq \tau,$$

where $\tau(\leq \infty)$ is a user specified limit and the superscript represents the groups being compared. Lan and Datta (2010a) defined a test statistic based on L_1 distance for comparing the marginal estimates θ^j each group $j = 1, 2$ as

$$\Delta := \int_{[0,\tau]} \left| \widehat{\theta}^1(x) - \widehat{\theta}^2(x) \right| dF_n(x)$$

where F_n is the empirical cumulative distribution function of the pooled collection of inspection times C_i and $\widehat{\theta}^k$ is the nonparametric estimator of θ^k using samples from the kth group $(k = 1, 2)$. Let n_1 and n_2 denote the sample sizes for the samples from groups 1 and 2, respectively, then the test statistic becomes

$$\Delta = \frac{1}{n_1 + n_2} \sum_{l=1}^{n_1+n_2} \left| \widehat{\theta}^1(C_i) - \widehat{\theta}^2(C_i) \right| I(C_i \leq \tau) \tag{21}$$

Lan and Datta (2010a) use bootstrap resampling to compute the p-value by assuming that the two multistate processes are identical and that the censoring mechanism in the two groups are idential. They pool inspection times C_i and then sample from the pool to get times C_i^* and their corresponding state information is taken as S_i^*. The bootstrap sample is then split in half with the first half taken as group 1 and those remaining are in group 2. The test statistic in Equation (21) is then computed for each bootstrap sample as

$$\widehat{\Delta}^* = \frac{1}{n_1 + n_2} \sum_{l=1}^{n_1+n_2} \left| \widehat{\theta}^{1*}(C_i^*) - \widehat{\theta}^{2*}(C_i^*) \right| I(C_i^* \leq \tau) \tag{22}$$

The p-value for B bootstrap replicated is then estimates as

$$\widehat{p} = B^{-1} \sum I(\widehat{\Delta}^* \geq \widehat{\Delta}) \tag{23}$$

Note that B is typically at least 500. The null hypothesis H_0 is rejected when $\widehat{p} \leq \alpha$ where α is a nominal level of significance.

We will apply these general hypothesis testing methods to interval censored data. We will use the estimators of state occupation probability, as well as state

entry and exit time distributions, as marginal estimates for comparison. Interval censored data consist of pairs of data $\{C_{ik}, S_i(C_{ik})\}$ for $1 \leq k \leq n_i$; $1 \leq i \leq n$, where n denotes the total number of individuals, n_i denotes the number of inspection times retained for the ith individual, C_{ik} are the inspection times for the ith individual and the corresponding state information is denoted $S_i(C_{ik})$. We will generate the bootstrap inspection times, say C^*_{ik}, by taking a random sample of pooled inspection times C_{ik}, initially ignoring the dependency for pooling. The bootstrap state information S^*_{ik} will be taken as the S_{ik} associated with C^*_{ik} in the original data. The resulting bootstrap sample will then be split into two groups with the first n_1 pairs taken as group 1 and the next n_2 individuals are taken as group 2. We will then estimate the test statistic found in 22 and compute the test statistic in 23.

REFERENCES

Aalen, O. O. (1978). Nonparametric inference for a family of counting processes. *The Annals of Statistics*, **6**(4), 701–726.

Aalen, O. O. (1980). A model for non-parametric regression analysis of counting processes. In W. Klonecki, A. Kozek, and J. Rosiskipp, editors, *Lecture Notes in Statistics-2: Mathematical Statistics and Probability Theory*, pages 1–25. Springer-Verlag, New York.

Aalen, O. O. and Johansen, S. (1978). An empirical transition matrix for nonhomogeneous markov chains based on censored observations. *Scandinavian Journal of Statistics*, **5**(3), 141–150.

Allignol, A., Schumacher, M., and Beyersmann, J. (2011). Empirical transition matrix of multi-state models: The **etm** package. *Journal of Statistical Software*, **38**(4), 1–15.

Andersen, P. K., Borgan, Ø., Gill, R. D., and Keiding, N. (1993). *Statistical Models Based on Counting Processes*. Springer-Verlag, New York.

Barlow, R. E., Bartholomew, J. M., Bremner, J. M., and Brunk, H. D. (1972). *Statistical Inference Under Order Restrictions*. John Wiley, New York.

Bie, O., Borgan, O., and Liestol, K. (1987). Confidence intervals and confidence bands for the cumulative hazard rate function and their small sample properties. *Scandinavian Journal of Statistics*, **14**(3), 221–233.

Borgan, O. and Liestol, K. (1990). A note on confidence intervals and bands for the

survival curve based on transformations. *Scandinavian Journal of Statistics*, **17**(1), 35–41.

Cook, R. J., Lawless, J. F., Lakhal-Chaieb, L., and Lee, K. (2009). Robust estimation of mean functions and treatment effects for recurrent events under event-dependent censoring and termination: application to skeletal complications in cancer metastic to bone. *Journal of the American Statistical Association*, **104**, 60–75.

Datta, S. and Ferguson, A. N. (2011). Nonparametric estimation of marginal temporal functionals in a multistate model. In A. Lisnianski and I. Frenkel, editors, *Recent Advances in System Reliability: Signatures, Multi-state Systems and Statistical Inference*. Springer.

Datta, S. and Satten, G. A. (2000). Estimating future stage entry and occupation probabilities in a multistage model based on randomly right-censored data. *Statistics and Probability Letters*, **50**(1), 89–95.

Datta, S. and Satten, G. A. (2001). Validity of the aalen-johansen estimators of stage occupation probabilities and nelson-aalen estimators of integrated transition hazards for non-markov models. *Statistics and Probability Letters*, **55**(4), 403 – 411.

Datta, S. and Satten, G. A. (2002). Estimation of integrated transition hazards and stage occupation probabilities for non-markov systems under dependent censoring. *Biometrics*, **58**(4), 792–802.

Datta, S. and Sundaram, R. (2006). Nonparametric estimation of state occupation probabilities in a multistate model with current status data. *Biometrics*, **62**, 829–837.

Datta, S., Satten, G. A., and Datta, S. (2000a). Estimation of stage occupation probabilities in multi-state models. In N. Balakrishnan, editor, *Advances on*

Theoretical and Methodological Aspects of Probability and Statistics. Gordon and Breach, New York.

Datta, S., Satten, G. A., and Datta, S. (2000b). Nonparametric estimation for the three-stage irreversible illness-death model. *Biometrics*, **56**, 841–847.

Datta, S., Lan, L., and Sundaram, R. (2009). Nonparametric estimation of waiting time distributions in a markov model based on current status data. *Journal of Statistical Planning and Inference*, **139**(9), 2885–2897.

de Leeuw, J., Hornik, K., and Mair, P. (2009). Isotone optimization in r: Pool-adjacent-violators algorithm (pava) and active set methods. *Journal of Statistical Software*, **32**(5), 1–24.

de Wreede, L. C., Fiocco, M., and Putter, H. (2011). **mstate**: An R package for the analysis of competing risks and multistate models. *Journal of Statistical Software*, **38**(7), 1–30.

Doksum, K. A. and Yandell, B. S. (1982). Properties of regression estimates based on censored survival data. In P. Bickel, K. Doksum, and J. Hodges, editors, *A Festschrifl for Erich L. Lehmann*, pages 140–56. Wadsworth, Belmont, California.

Frydman, H. (1992). A non-parametric estimation procedure for a periodically observed threestate markov process, with application to aids. *Journal of the Royal Statistical Society. Ser. B*, **54**(3), 853âĂŞ66.

Frydman, H. (1995). Semi-parametric estimation in a three-state duration-dependent markov model from interval-censored observations with application to aids. *Biometrics*, **51**(2), 502âĂŞ11.

Gentleman, R., Whalen, E., Huber, W., and Falcon, S. (2010). **graph**: *A Package to Handle Graph Data Structures*. R package version 1.28.0.

Glidden, D. V. (2002). Robust inference for event probabilities with non-markov data. *Biometrics*, **58**, 361–368.

Groeneboom, P., Maathuis, M. H., and Wellner, J. A. (2008). Current status data with competing risks: Limiting distribution of the mle. *The Annals of Statistics*, **36**(3), 1064–1089.

Hudgens, M. G., Satten, G. A., and Longini, I. M. (2001). Nonparametric maximum likelihood estimation for competing risks survival data subject to interval censoring and truncation. *Biometrics*, **57**(1), 74âĂŞ80.

Huzurbazar, A. V. (2005). *Flowgraph models for multi-state time-to-event data*. Wiley-Interscience, Hoboken, New Jersey.

Jackson, C. (2011). Multi-state models for panel data: The **msm** package for R. *Journal of Statistical Software*, **38**(8), 1–28.

Jewell, N., van der Laan, M., and Henneman, T. (2003). Nonparametric estimation from current status data with competing risks. *Biometrika*, **90**(1), 183–197.

Kalbfleisch, J. D. and Prentice, R. L. (1980). *The Statistical Analysis of Failure Time Data*. John Wiley and Sons, New York.

Klein, J. P. and Moeschberger, M. L. (1997). *Survival Analysis: Techniques for Censored and Truncated Data*. Springer, New York.

Koul, H., Susarla, V., and Van Ryzin, J. (1981). Regression analysis of randomly right censored data. *Annals of Statistics*, **9**, 1276–1288.

Lan, L. and Datta, S. (2010a). Comparison of state occupation, entry, exit and waiting times in two or more groups based on current status data in a multistate model. *Statistics in Medicine*, **29**(7-8), 906–14.

Lan, L. and Datta, S. (2010b). Nonparametric estimation of state occupation, entry and exit times with multistate current status data. *Statistical Methods in Medical Research*, **19**, 147–165.

Martin, A. P., Bartels, M., Hauss, J., and Fangmann, J. (2007). Overview of the meld score and the unos adult liver allocation system. *Transplantation Proceedings*, **39**(10), 3169–3174.

Meira-Machado, L. and Roca-Pardinas, J. (2011). **3state.msm**: Analyzing survival data from an illness-death model. *Journal of Statistical Software*, **38**(3), 1–18.

Meira-Machado, L., de UÃśa-Alvarez, J., and Cadarso-Su'arez, C. (2006). Nonparametric estimation of transition probabilities in a non-markov illness-death model. *Lifetime Data Anal.*, **12**, 325–344.

Nelson, W. (1972). Theory and applications of hazard plotting for censored failure data. *Technometrics*, **14**(4), 945–965.

Pepe, M. S. (1991). Inference for events with dependent risks in multiple end point studies. *Journal of American Statistical Association*, **86**(415), 770–778.

Robins, J. and Rotnitzky, A. (1992). Recovery of information and adjustment for dependent censoring using surrogate markers. In N. Jewell, K. Dietz, and V. Farewell, editors, *AIDS Epidemiology - Methodological Issues*, pages 297–331. Birkhauser, Boston.

Robins, J. M. (1993). Information recovery and bias adjustment in proportional hazards regression analysis of randomized trials using surrogate markers. In *Proceedings of the American Statistical Association, Biopharmaceutical*, pages 24–33.

Sarkar, D. (2008). *Lattice: Multivariate Data Visualization with R*. Springer.

Satten, G. A. and Datta, S. (2002). Marginal estimation for multistage nodels: Waiting time distributions and competing risk analyses. *Statistics in Medicine*, **21**(1), 3–19.

Satten, G. A. and Sternberg, M. R. (1999). Fitting semi-markov models to interval-censored data with unknown initiation times. *Biometrics*, **55**, 507–513.

Satten, G. A., Datta, S., and J., R. (2001). Estimating the marginal survival function in the presence of time dependent covariates. *Statistics and Probability Letters*, **54**, 397–403.

Schulgen, G. and Schumacher, M. (1996). Estimation of prolongation of hospital stay attributable to nosocomial infections: New approaches based on multistate models. *Lifetime Data Anal.*, **2**(3), 219–40.

Thomas, D. R. and Grunkemeier, G. L. (1975). Confidence interval estimation of survival probabilities for censored data. *Journal of American Statistical Association*, **70**(352), 865–871.

Wang, W. (2003). Nonparametric estimation of the sojourn time distributions for a multipath model. *Journal of the Royal Statistical Society. Ser. B*, **65**, 921–935.

Wang, W. and Wells, M. T. (1998). Nonparametric estimation of successive duration times under dependent censoring. *Biometrika*, **85**, 561–572.

Wrangler, M., Beyersmann, J., and Schumacher, M. (2006). **changeLOS**: An R -package for change in length of hospital stay based on the aalen-johansen estimator. *R News*, **6**(2), 31–35.

A Functions in the msSurv package

Key internal functions in the R package **msSurv** for nonparametric estimation of right censored and possibly left truncated data.

```
## Adding Start Times ##
Add.start <- function(Data){
Data$start <- 0
idx <- which(table(Data$id)>1)

for(i in idx){
   ab <-Data[which(Data$id==i),]
   ab<-with(ab,ab[order(ab$stop),])
   ab2<-which(Data$id==i) #row numbers in Data
   start2<-vector(length=length(ab2))
   start2[1]<-0
   start2[2:length(ab2)]<-ab$stop[1:length(ab2)-1]
   Data$start[ab2]<-start2
} #end of for loop

new.data <- data.frame(id=Data$id,start=Data$start,stop=Data$stop,
st.stage=Data$st.stage,stage=Data$stage)
res<-new.data
}

## Converting for Censoring ##

Add.States <- function(tree){
```

```
   ##Adding censoring state to Nodes & Edges
   Nodes <- c("0",nodes(tree))
   Edges <- edgeL(tree)
   Edges[["0"]] <- list(edges=numeric(0))

   nt.states <- which(sapply(Edges, function(x) length(x$edges)>0))

  for(stage in nt.states) {
    Edges[[stage]]$edges <- c("0",Edges[[stage]]$edges)
  }

 ##tree for censored data
 tree0 <- new("graphNEL",nodes=Nodes,edgeL=Edges,edgemode="directed")

## Adding "Left Truncated" State
Nodes<- c("LT",nodes(tree0))
Edges[["LT"]] <- list(edges=nodes(tree)[nodes(tree)%in%names(nt.states)])
  nt.states.LT <- which(sapply(Edges, function(x) length(x$edges)>0))


treeLT <-new("graphNEL",nodes=Nodes,edgeL=Edges,edgemode="directed")

 list(tree0=tree0,nt.states=nt.states,nt.states.LT=nt.states.LT,
  treeLT=treeLT)

}
```

```
## Adding Dummy "LT" obs to Data set ##

LT.Data <- function(Data){
  ## NOTE: Below assumes all the variables in Data have the names 'id',
   ## 'start', 'stop', etc.,
   Data <- Data[order(Data$id), ]  ## make sure id's line up below
   ids <- unique(Data$id)
   stop.time <- with(Data, tapply(start, id,min))
   enter.st<- with(Data, tapply(st.stage, id,min))
   dummy <- data.frame(id = ids, start = -1, stop = stop.time,
    st.stage="LT", stage=enter.st) #dummy initial stage
   Data <- rbind(Data, dummy)
   Data <- with(Data, Data[order(id,stop), ])
   return(Data=Data)
}

## Counting Process & At Risk ##
CP <- function(tree,tree0,Data,nt.states){

   times <- sort(unique(Data$stop))
   lng <- sapply(edges(tree0)[nodes(tree0)%in%names(nt.states)],
    length)
   ds <- paste("dN", rep(nodes(tree0)[nodes(tree0)%in%names(nt.states)],
     lng),unlist(edges(tree0)[nodes(tree0)%in%names(nt.states)]))
   ys <- paste("y",unlist(nodes(tree0)))

   ## index of obs in each stage/state/node
   indy <- vector(length=length(ys),mode="list")
```

```
names(indy) <- ys

indD <- vector(length=length(ds),mode="list")

# matrix of # of transitions, initialize to zeros
dNs <- matrix(0, nrow=length(times), ncol=length(ds))

# matrix of total # of transitions from a state, initialize to zeros
sum.dNs <- matrix(0, nrow=length(times), ncol=length(nt.states))

# matrix of at-risk sets for each stage at each time
Ys <- matrix(NA, nrow=length(times), ncol=length(ys))

#names of rows/columns for vectors/matrices
rownames(dNs) <- rownames(sum.dNs) <- rownames(Ys) <- times
names(indD) <- colnames(dNs) <- ds
colnames(Ys) <- ys
colnames(sum.dNs) <- paste("dN",names(nt.states),".")

n.vec<-vector(length=length(nodes(tree0)))

for(i in nodes(tree0)){ #loop through nodes

  nam <- strsplit(names(indy)," ")
  idx <- which(sapply(nam, function(x) x[2]==i))
  indy[[ys[idx]]] <- which(Data$stage==i)

  if(length(inEdges(tree0)[[i]])==0) next
```

```
    ld <- length(inEdges(tree0)[[i]])

    for(j in 1:ld){ #Fill-in no. transitioning between stages at each time

       nam2 <- paste("dN", inEdges(tree0)[[i]][j], i)
     indD[[nam2]] <- indy[[idx]][Data$st.stage[indy[[idx]]]
      ==inEdges(tree0)[[i]][j]]

       tmp.tab <- table(Data$stop[indD[[nam2]]])
       dNs[names(tmp.tab),nam2] <- tmp.tab
    }
  } #end of outer loop for dNs

res <- by(Data, Data$id, function(x) x$st.stage[which.min(x$stop)])
res <- factor(res, levels=nodes(tree), labels=nodes(tree))
start.probs <- table(res)/length(res)

  ### starting at risk computations ###
  for(i in nodes(tree0)){ #loop through nodes to find Ys


    n <- length(which(res==i))

    nam <- strsplit(names(indy)," ")
    idx <- which(sapply(nam, function(x) x[2]==i))

     if (length(inEdges(tree0)[[i]])>0)
```

```
        into.node <- paste("dN", inEdges(tree0)[[i]], i)
        else into.node <- NULL
      if (length(edges(tree0)[[i]])>0)
        from.node <- paste("dN", i, edges(tree0)[[i]])
        else from.node <- NULL

  initial <- which(sapply(inEdges(tree0), function(x) !length(x)>0))
  transient <- which(sapply(edges(tree0),function(x) length(x)>0)
   & sapply(inEdges(tree0),function(x) length(x)>0))

if (i==names(initial)){
        Ys[,idx] <- c(n, n + cumsum(rowSums(dNs[,into.node, drop=FALSE]))
           - cumsum(rowSums(dNs[,from.node, drop=FALSE])))[-(nrow(Ys)+1)]
} else if(i==names(transient) && !n==0){
        Ys[,idx] <- c(n, n + cumsum(rowSums(dNs[,into.node, drop=FALSE]))
           - cumsum(rowSums(dNs[,from.node, drop=FALSE])))[-(nrow(Ys)+1)]
} else Ys[,idx] <- c(0, cumsum(rowSums(dNs[,into.node, drop=FALSE]))
         - cumsum(rowSums(dNs[,from.node, drop=FALSE])))[-(nrow(Ys)+1)]

   } #end of loop for Ys

    ## Counting transitions from different stages (ie: stage sums)
    sum.dNs <- matrix(nrow=nrow(dNs),ncol=length(nt.states))
    rownames(sum.dNs) <- rownames(dNs) #
    colnames(sum.dNs) <- paste("dN",names(nt.states),".")
    a <- strsplit(colnames(sum.dNs), " ")
    a2 <- strsplit(colnames(dNs), " ")
    uni <- unique(sapply(a,function(x) x[2]))
```

```
    for(i in uni){ #calculating the dNi.s
      b <- which(sapply(a,function(x) x[2]==i))
      b2 <- which(sapply(a2,function(x) x[2]==i))
 sum.dNs[,b] <- rowSums(dNs[,b2])
    } #end of for loop for calculating dNi.s

    list(dNs=dNs,Ys=Ys,sum.dNs=sum.dNs,res=res,start.probs=start.probs)

} #end of function

## Datta-Satten Estimation ##

DS <- function(LT="LT",nt.states,dNs,sum.dNs,Ys,Cens="0",cens.type){
   ## Calculating dNs, sum.dNs, and Y from D-S(2001) paper
       res <- strsplit(colnames(dNs), " ") #string splits names
       res2 <- strsplit(colnames(Ys)," ") #string split names of Ys
       res3 <- strsplit(colnames(sum.dNs)," ") #string splits names of dNs

       DS.col.idx <- which(sapply(res, function(x) x[3]==Cens))
       DS2.col.idx <- which(sapply(res2, function(x) x[2]%in%names(nt.states)))
       DS3.col.idx <- which(sapply(res3, function(x) x[2]%in%names(nt.states)))

   if(cens.type=="ind"){ ## for INDEPENDENT censoring

K <- vector(length=nrow(dNs))
dN0 <- rowSums(dNs[,DS.col.idx])
Y0 <- rowSums(Ys[,DS2.col.idx]) #those at risk of being censored
```

```
N.Y <- ifelse(dN0/Y0=="NaN",0,dN0/Y0)
colnames(N.Y) <- NULL
H.t <- cumsum(N.Y) #calculating the hazard
k <- exp(-H.t)
K <- c(1, k[-length(k)])

dNs.K <- dNs/K  #D-S dNs
Ys.K <- Ys/K  #D-S Ys
        sum.dNs.K <- sum.dNs/K
    } #end of ind censoring if

## Dependent censoring
    if(cens.type=="dep"){

      dN0 <- dNs[,DS.col.idx]
      Y0 <- Ys[,DS2.col.idx] #those at risk of being censored

      N.Y <- ifelse(dN0/Y0=="NaN",0,dN0/Y0)
colnames(N.Y) <- paste(colnames(dN0),"/",colnames(Y0))

      H.t <- apply(N.Y, 2, function(x) cumsum(x))
      K <- exp(-H.t)
      ## K <- apply(k, 2, function(x) c(1, x[-length(x)]))

    ab <- which(sapply(res,function(x) x[2]%in%nt.states))
      ac <- which(sapply(res3,function(x) x[2]%in%nt.states))
dNs.K <-dNs; Ys.K <- Ys; sum.dNs.K <- sum.dNs
for(i in names(nt.states)){
```

```
K.idx <- which(sapply(strsplit(colnames(N.Y)," "),function(x) x[2]==i))
dN.idx <- which(sapply(res,function(x) x[2]==i))
                sum.dNs.idx <- which(sapply(res3,function(x) x[2]==i))
Ys.idx <- which(sapply(res2,function(x) x[2]==i))
dNs.K[,dN.idx] <- dNs[,dN.idx]/K[,K.idx]
                sum.dNs.K[,sum.dNs.idx] <- sum.dNs[,sum.dNs.idx]/K[,K.idx]
Ys.K[,Ys.idx] <- Ys[,Ys.idx]/K[,K.idx]
}
    } #end of if dependent censoring

      res <- list(dNs.K=dNs.K,Ys.K=Ys.K,sum.dNs.K=sum.dNs.K)
      return(res)

} ## end of D-S function

## Reducing dNs & Ys to event times ##

Red <- function(tree,dNs,Ys,sum.dNs,dNs.K,Ys.K,sum.dNs.K){
    res <- strsplit(colnames(dNs), " ") #string splits names
    col.idx <- which(sapply(res, function(x) x[2]%in%nodes(tree)
     & x[3]%in%nodes(tree)))
    row.idx <- which(apply(dNs[,col.idx], 1, function(x) any(x>0)))
    dNs.et <- dNs[row.idx,col.idx] ## reduces dNs
    res2 <- strsplit(colnames(Ys)," ") #string split names of Ys
    nt.states.f <- which(sapply(edges(tree), function(x) length(x)>0))
    col2.idx <- which(sapply(res2,function(x) x[2]%in%names(nt.states.f)))
    Ys.et <- Ys[row.idx,col2.idx] ## reduces Ys
    col3.idx <- which(sapply(strsplit(colnames(sum.dNs)," "),
```

```
     function(x) x[2]%in%nodes(tree)))
    sum.dNs.et <- sum.dNs[row.idx,col3.idx]
    dNs.K.et <- dNs.K[row.idx,col.idx]
    Ys.K.et <- Ys.K[row.idx,col2.idx]
    sum.dNs.K.et <- sum.dNs.K[row.idx,col3.idx]
    ans <- list(dNs=dNs.et,Ys=Ys.et,sum.dNs=sum.dNs.et,dNs.K=dNs.K.et,
     Ys.K=Ys.K.et,sum.dNs.K=sum.dNs.K.et)
    return(ans)
}

## State Occupation Probabilities ##
stocc <- function(ns,tree,dNs.et,Ys.et,start.probs){
   cum.tm <- diag(ns)
   colnames(cum.tm) <- rownames(cum.tm) <- nodes(tree)

   ps <- matrix(NA, nrow=nrow(dNs.et), ncol=length(nodes(tree)))
      rownames(ps) <- rownames(dNs.et); colnames(ps) <- paste("p",nodes(tree))
   all.dA <- all.I_dA <- all.ajs <- array(dim=c(ns,ns,nrow(dNs.et)),
   dimnames=list(rows=nodes(tree),cols=nodes(tree),dim=rownames(dNs.et)))

   for(i in 1:nrow(dNs.et)){ ##loop through times

I_dA <- diag(ns) #creates trans matrix for current time
dA <- matrix(0,nrow=ns,ncol=ns)
colnames(I_dA) <- rownames(I_dA) <- colnames(dA) <- rownames(dA) <- nodes(tree)

idx <- which(dNs.et[i,]>0)  ## transition time i
t.nam <- colnames(dNs.et)[idx] ## gets names of transitions (ie:  dN##)
```

```
tmp <- strsplit(t.nam," ") ## splits title of dN##
start <- sapply(tmp, function(x) x[2])
end <- sapply(tmp, function(x) x[3])
idxs <- matrix(as.numeric(c(start, end)), ncol=2)
idxs2 <- matrix(as.numeric(c(start, start)), ncol=2)

dA[idxs] <- dNs.et[i,idx]/Ys.et[i,paste("y",start)]
      if(length(idx)==1)
       dA[start,start] <- -dNs.et[i,idx]/Ys.et[i,paste("y",start)]
else dA[idxs2] <- -rowSums(dA[start, ])

I_dA <- I_dA + dA #I+dA matrix

all.dA[,,i] <- dA  #stores all dA matrices
all.I_dA[,,i] <- I_dA

cum.tm <- cum.tm %*% I_dA
all.ajs[,,i] <- cum.tm

ps[i,] <- start.probs%*%all.ajs[,,i] #just the state occupation probabilities

    } #end of loop

    list(ps=ps,all.ajs=all.ajs,all.I_dA=all.I_dA)
} #end of function

## State Entry/Exit Distributions ##
Dist <- function(ps,ns,tree){
```

```
   initial <- which(sapply(inEdges(tree), function(x) !length(x)>0))
   terminal <- which(sapply(edges(tree), function(x) !length(x)>0))

   Fs <- matrix(0, nrow=nrow(ps), ncol=ns) #entry distn
     rownames(Fs) <- rownames(ps)
     colnames(Fs) <- paste("F",nodes(tree))

   Gs <- matrix(0, nrow=nrow(ps), ncol=ns) #exit distn
     rownames(Gs) <- rownames(ps)
     colnames(Gs) <- paste("G",nodes(tree))

   for(i in 1:ns){#looping through nodes
 node <- nodes(tree)[i]
 later.stages <- names(acc(tree, node)[[1]])
 stages <- c(node, later.stages)

       f.numer <- rowSums(ps[,paste("p", stages),drop=FALSE])
       Fs[,i] <- f.numer/f.numer[length(f.numer)]

 if(length(stages)==1) next

 g.numer <- rowSums(ps[,paste("p", later.stages),drop=FALSE])
 Gs[,i] <- g.numer/g.numer[length(g.numer)]

    } #end of for loop

Fr <- strsplit(colnames(Fs)," ")
Fs.idx <- which(sapply(Fr,function(x) x[2]%in%names(initial)))
```

```
Fs[,Fs.idx]<-NA

Gr <- strsplit(colnames(Gs)," ")
Gs.idx <- which(sapply(Gr,function(x) x[2]%in%names(terminal)))
Gs[,Gs.idx]<-NA

    list(Fs=Fs,Gs=Gs)
} #end of function

## Variance ##
var.fn <- function(tree,ns,nt.states,dNs.et,Ys.et,sum.dNs,
all.ajs,all.I_dA,ps){

    #elements needed for computation
    varcov <- array(0, dim = c(ns^2,ns^2,nrow(dNs.et)))
      colnames(varcov) <- rownames(varcov) <-
      paste(rep(nodes(tree),ns),sort(rep(nodes(tree),ns)))
    bl.Id <- diag(1,(ns)^2) #Ident matrix for Kronecker product
    tm <- matrix(0,nrow=ns,ncol=ns) #tmp matrix to col var est
    res.array <- array(0,dim(tm)^2)
      colnames(res.array) <- rownames(res.array) <-
      paste(rep(nodes(tree),ns),sort(rep(nodes(tree),ns)))
    out <- array(0, dim=c(dim(all.I_dA)[c(1, 2)]^2,nrow(dNs.et)))
      colnames(out) <- rownames(out) <- paste(rep(nodes(tree),ns),
      sort(rep(nodes(tree),ns)))
    Id <- diag(1,ns)
    cov.p <- matrix(0,nrow=nrow(dNs.et),ncol=ns)
      colnames(cov.p) <- paste("Var", "p",nodes(tree))
```

```
      rownames(cov.p) <- rownames(ps)
    v.p <- matrix(0,ns,ns)

    for(i in 1:nrow(dNs.et)){ ##loop through times

        #VARIANCE OF A-J (TRANS PROB MATRIX P(0,t))
for(outer in 1:length(nt.states)){ #loop on the blocks (g)

  tm <- matrix(0,nrow=ns,ncol=ns)
  for(j in 1:ns){ #loop in the blocks

     for(k in j:ns){

        if(Ys.et[i,outer]==0){  ## if Y_g = 0 the covariance = 0
            tm[j,k] <- 0
         next
          } #end of if

          if (j == outer & k == outer) {  ## 3rd formula
tm[j,k] <- (Ys.et[i,outer]-sum.dNs[i,outer])*
sum.dNs[i,outer]/Ys.et[i,outer]^3
        }  else if (j == outer & k != outer) {  ## 2nd formula
            name <- paste("dN", outer, k)
if (!name%in%colnames(dNs.et)) next
tm[j,k] <- -(Ys.et[i,outer]-sum.dNs[i,outer])
*dNs.et[i,name]/Ys.et[i,outer]^3
  } else if (j != outer & k == outer) {
            name <- paste("dN", outer, j)
```

```
if (!name%in%colnames(dNs.et)) next
tm[j,k] <- -(Ys.et[i,outer]-sum.dNs[i,outer])*
dNs.et[i,name]/Ys.et[i,outer]^3
  } else { ## 1st formula
namek <- paste("dN", outer, k)
namej <- paste("dN", outer, j)
if (!(namej%in%colnames(dNs.et) & namek%in%colnames(dNs.et))) next
tm[j,k] <- (ifelse(j==k, 1, 0)*Ys.et[i,outer]-dNs.et[i,namej])*
dNs.et[i,namek]/Ys.et[i,outer]^3
              } #end of if/else statements
     } ## end of k loop
  } ## end of j loop

      tm[lower.tri(tm)] <- t(tm)[lower.tri(tm)]

      res.array[(seq(1, ns*(ns-1)+1, by=ns)+outer-1),
       (seq(1, ns*(ns-1)+1, by=ns)+outer-1)] <- tm

}#end of outer loop

varcov[,,i] <- res.array

if(i==1) out[, , i] <- bl.Id%*% varcov[,,i] %*% bl.Id
  else out[, , i] <- (t(all.I_dA[, , i]) %x% Id)
   %*% out[, , i-1] %*%((all.I_dA[, , i]) %x% Id) +
      (Id %x% all.ajs[, , i-1]) %*% varcov[,,i]  %*%
      (Id%x% t(all.ajs[, , i-1]))
      ## calculating the variance of state occupation prob
```

```
for (j in nodes(tree)){ #loop through states

        st.nam <- paste("1",j)
   part1 <- var.pkj0t <- out[st.nam,st.nam,i]

   res3 <- strsplit(colnames(ps)," ")
   col.idx3 <- which(sapply(res3, function(x) x[2]== j))
   b.t <- all.ajs[,col.idx3,i]

   part2 <- t(b.t)%*%v.p%*%b.t #should be 0 when P(0,t)
   res.varp <- part1+part2
   cov.p[i,as.numeric(j)] <- res.varp

} #closes states loop
    } ## end of time loop

    list(out=out,varcov=varcov,cov.p=cov.p)

}#end of function

## BS Variance for Dep Cens ##
BS.var <- function(Data,tree,ns,et,cens.type,B,LT,start.states){

n <- length(unique(Data$id)) # sample size
ids <- unique(Data$id)

bs.est <- array(dim=c(length(nodes(tree)),length(nodes(tree)),
length(et),B),
```

```
dimnames=list(rows=nodes(tree),cols=nodes(tree),dim=et))
bs.ps <- array(dim=c(length(et),ns,B))
        rownames(bs.ps) <- et
        colnames(bs.ps) <- paste("p",nodes(tree))

## For entry / exit distributions
bs.Fs <- bs.ps; bs.Gs <- bs.ps #storage for BS Fs/Gs
colnames(bs.Fs) <- paste("F",nodes(tree))
colnames(bs.Gs) <- paste("G",nodes(tree))
      initial <- which(sapply(inEdges(tree),
      function(x) !length(x)>0)) #initial states, no Fs
      terminal <- which(sapply(edges(tree),
      function(x) !length(x)>0)) #terminal states, no Gs

      bs.cov.p <- matrix(0,nrow=length(et),ncol=ns)
      colnames(bs.cov.p) <- paste("Var", "p",nodes(tree))
      rownames(bs.cov.p) <- et

      res.array <- array(0,dim=c(ns^2,ns^2,length(et)),dimnames=list
      (rows=paste(rep(nodes(tree),ns),sort(rep(nodes(tree),ns))),
      cols=paste(rep(nodes(tree),ns),sort(rep(nodes(tree),ns))),dim=et))

for(b in 1:B){ #randomly selects the indices

## Find the bootstrap sample
bs=sample(ids, n, replace=TRUE)
bs=factor(bs, levels=ids)
bs.tab=data.frame(table(bs)) ##table with the frequencies
```

```
Data.bs=merge(Data, bs.tab, by.x="id", by.y="bs")
bs.id=unlist(apply(Data.bs[Data.bs$Freq>0,], 1,
function(x) paste(x["id"], 1:x["Freq"], sep=".")))
idx=rep(1:nrow(Data.bs), Data.bs$Freq)
Data.bs=Data.bs[idx,]
Data.bs.originalID=Data.bs$id
Data.bs$id=bs.id
Data.bs=Data.bs[order(Data.bs$stop),]

Cens <- Add.States(tree)
  if(LT) {
Data.bs = LT.Data(Data.bs)
cp <- CP(tree,Cens$treeLT,Data.bs,Cens$nt.states.LT)
res <- factor(start.states, levels=nodes(tree), labels=nodes(tree))
start.probs <- table(res)/length(res)
   }

   if(!LT) {
cp <- CP(tree,Cens$tree0,Data.bs,Cens$nt.states)
start.probs <- cp$start.probs
  }
ds.est<-DS(LT="LT",Cens$nt.states,cp$dNs,cp$sum.dNs, cp$Ys,Cens="0",
cens.type)
cp.red <- Red(tree,cp$dNs,cp$Ys,cp$sum.dNs,ds.est$dNs.K,
ds.est$Ys.K,ds.est$sum.dNs.K)
stateoccfn <- stocc(ns,tree,cp.red$dNs.K,cp.red$Ys.K,start.probs)

idx <- which(dimnames(bs.est)[[3]] %in% dimnames(stateoccfn$all.I_dA)[[3]])
```

```
idx2 <- which(!(dimnames(bs.est)[[3]] %in% dimnames(stateoccfn$all.I_dA)[[3]]))
bs.IA <- bs.est

bs.IA[,,idx,b] <- stateoccfn$all.I_dA
bs.IA[,,idx2,b] <- diag(ns)

bs.est[,,1,b] <- bs.IA[,,1,b]
bs.ps[1,,b] <- start.probs%*%bs.est[,,1,b]

for(j in 2:length(et)){
bs.est[,,j,b] <- bs.est[,,j-1,b] %*% bs.IA[,,j,b]
bs.ps[j,,b] <- start.probs%*%bs.est[,,j,b]
} ## end of j for loop

  ## Entry / Exit variance as well
  for(i in 1:ns){#looping through nodes
 node <- nodes(tree)[i]
 later.stages <- names(acc(tree, node)[[1]])
 stages <- c(node, later.stages)

       bs.f.numer <- rowSums(bs.ps[,paste("p", stages),b,drop=FALSE])
if(sum(bs.f.numer)==0)  bs.Fs[,i,b]<-0
          else bs.Fs[,i,b] <- bs.f.numer/bs.f.numer[length(bs.f.numer)]


 if(length(stages)==1) next

 bs.g.numer <- rowSums(bs.ps[,paste("p", later.stages),b,drop=FALSE])
```

```
if(sum(bs.g.numer)==0)  bs.Gs[,i,b]<-0
 else bs.Gs[,i,b] <- bs.g.numer/bs.g.numer[length(bs.g.numer)]

  } #end of for loop

} ## end of bs loop

Fs.var <- apply(bs.Fs,c(1,2),var)
Fs.var[,initial]<-NA
Gs.var <- apply(bs.Gs,c(1,2),var)
Gs.var[,terminal] <- NA

bs.var <- apply(bs.est, c(1,2,3), var)
bs.cov.p <- apply(bs.ps,c(1,2),var)
        colnames(bs.cov.p) <- paste("Var", "p",nodes(tree))
        rownames(bs.cov.p) <- et

bs.cov <- array(dim=c(ns^2,ns^2,length(et)),dimnames=list(rows=
paste(rep(1:ns,ns), rep(1:ns, each=ns)),cols=paste(rep(1:ns,ns),
rep(1:ns, each=ns)),dim=et))
for(i in 1:length(et)){
bs.est2 <- matrix(bs.est[,,i,],nrow=B, ncol=ns^2, byrow=TRUE)
bs.cov[,,i] <- cov(bs.est2)
}  ##this for loop creates a B x (# of states)^2 x (# of event times)

list(out=bs.cov,cov.p=bs.cov.p, Fs.var=Fs.var,Gs.var=Gs.var)

} ## end of function
```

```
## BS Variance for Entry/Exit Distn ##
Dist.BS.var <- function(Data,tree,ns,et,dNs.K,cens.type,B,LT,start.probs){

n <- length(unique(Data$id)) # sample size
ids <- unique(Data$id)
        initial <- which(sapply(inEdges(tree), function(x) !length(x)>0))
        terminal <- which(sapply(edges(tree), function(x) !length(x)>0))
bs.est <- array(dim=c(length(nodes(tree)),length(nodes(tree)),length(et),B),
dimnames=list(rows=nodes(tree),cols=nodes(tree),dim=rownames(dNs.K)))
bs.ps <- array(dim=c(length(et),ns,B))
        rownames(bs.ps) <- et
        colnames(bs.ps) <- paste("p",nodes(tree))

bs.Fs <- bs.ps; bs.Gs <- bs.ps #storage for BS Fs/Gs
colnames(bs.Fs) <- paste("F",nodes(tree))
colnames(bs.Gs) <- paste("G",nodes(tree))


for(b in 1:B){ #randomly selects the indices
bs=sample(ids, n, replace=TRUE)
bs=factor(bs, levels=ids)
bs.tab=data.frame(table(bs))
Data.bs=merge(Data, bs.tab, by.x="id", by.y="bs")
bs.id=unlist(apply(Data.bs[Data.bs$Freq>0,], 1,
function(x) paste(x["id"], 1:x["Freq"], sep=".")))
idx=rep(1:nrow(Data.bs), Data.bs$Freq)
```

```
Data.bs=Data.bs[idx,] ##creating a bs dataset
Data.bs.originalID=Data.bs$id
Data.bs$id=bs.id
Data.bs=Data.bs[order(Data.bs$stop),] #ordered bs dataset
## Calling functions using bs dataset
Cens <- Add.States(tree)
  if(LT) {
Data.bs = LT.Data(Data.bs)
cp <- CP(tree,Cens$treeLT,Data.bs,Cens$nt.states.LT)
   }
   if(!LT) cp <- CP(tree,Cens$tree0,Data.bs,Cens$nt.states)
ds.est<-DS(LT="LT",Cens$nt.states,cp$dNs,cp$sum.dNs,cp$Ys,
Cens="0",cens.type)
cp.red <- Red(tree,cp$dNs,cp$Ys,cp$sum.dNs,ds.est$dNs.K,
ds.est$Ys.K,ds.est$sum.dNs.K)
stateoccfn <- stocc(ns,tree,cp.red$dNs.K,cp.red$Ys.K)
idx <- which(dimnames(bs.est)[[3]] %in% dimnames(stateoccfn$all.I_dA)[[3]])
idx2 <- which(!(dimnames(bs.est)[[3]] %in% dimnames(stateoccfn$all.I_dA)[[3]]))
bs.IA <- bs.est
bs.IA[,,idx,b] <- stateoccfn$all.I_dA
bs.IA[,,idx2,b] <- diag(ns)
bs.est[,,1,b] <- bs.IA[,,1,b]
bs.ps[1,,b]<-start.probs%*%bs.est[,,1,b]

for(j in 2:length(et)){
bs.est[,,j,b] <- bs.est[,,j-1,b] %*% bs.IA[,,j,b]
bs.ps[j,,b]<-start.probs%*%bs.est[,,j,b]
} ## end of j for loop
```

```
 for(i in 1:ns){#looping through nodes
node <- nodes(tree)[i]
later.stages <- names(acc(tree, node)[[1]])
stages <- c(node, later.stages)

      bs.f.numer <- rowSums(bs.ps[,paste("p", stages),b,drop=FALSE])
if(sum(bs.f.numer)==0)  bs.Fs[,i,b]<-0
         else bs.Fs[,i,b] <- bs.f.numer/bs.f.numer[length(bs.f.numer)]


if(length(stages)==1) next

bs.g.numer <- rowSums(bs.ps[,paste("p", later.stages),b,drop=FALSE])
if(sum(bs.g.numer)==0)  bs.Gs[,i,b]<-0
 else bs.Gs[,i,b] <- bs.g.numer/bs.g.numer[length(bs.g.numer)]

 } #end of for loop

} ## end of bs loop

Fs.var <- apply(bs.Fs,c(1,2),var)
Fs.var[,initial]<-NA #setting the initial state variances = NA
Gs.var <- apply(bs.Gs,c(1,2),var)
Gs.var[,terminal] <- NA
list(Fs.var=Fs.var,Gs.var=Gs.var)
} ## end of function
```

```
## CONFIDENCE INTERVALS for p(t) & P(s,t) ##
MSM.CIs <- function(x,ci.level=0.95,ci.trans="linear"){
#default ci.level is 0.95, default CI type (ie: ci.trans) is linear
    if(ci.level < 0 || ci.level > 1)
      stop("confidence level must be between 0 and 1")

    z.alpha <- qnorm(ci.level + (1 - ci.level) / 2)
    ci.trans <- match.arg(ci.trans,c("linear","log","cloglog","log-log"))
    CI.p <- array(0,dim=c(nrow(x@dNs),3,length(nodes(x@tree))),dimnames=list(rows=

    for(i in 1:nrow(x@dNs)){ ##loop through times
      for (j in as.numeric(nodes(x@tree))){ #loop through states

        res.ci <- strsplit(colnames(x@ps), " ") #string splits names
        col.idx <- which(sapply(res.ci, function(x) x[2]== j))
        res.ci2 <- strsplit(colnames(x@cov.p), " ")
        col.idx2 <- which(sapply(res.ci2, function(x) x[3]== j))

        CI.p[i,1,j]<- PE.p <- x@all.ajs[1,col.idx,i]
        var.p <- x@cov.p[i,col.idx2]

        switch(ci.trans[1],
"linear" = {
        CI.p[i,2,j] <- PE.p - z.alpha * sqrt(var.p)
              CI.p[i,3,j] <- PE.p + z.alpha * sqrt(var.p)},
"log" = {
        CI.p[i,2,j] <- exp(log(PE.p) - z.alpha * sqrt(var.p) / PE.p)
          CI.p[i,3,j] <- exp(log(PE.p) + z.alpha * sqrt(var.p) / PE.p)},
```

```
          "cloglog" = {
         CI.p[i,2,j] <- 1 - (1 - PE.p)^(exp(z.alpha * (sqrt(var.p) /
          ((1 - PE.p) * log(1 - PE.p)))))
          CI.p[i,3,j] <- 1 - (1 - PE.p)^(exp(-z.alpha * (sqrt(var.p) /
           ((1 - PE.p) * log(1 - PE.p)))))},
"log-log" = {
         CI.p[i,2,j] <- PE.p^(exp(-z.alpha * (sqrt(var.p) /
          (PE.p * log(PE.p)))))
          CI.p[i,3,j] <- PE.p^(exp(z.alpha * (sqrt(var.p) /
          (PE.p * log(PE.p)))))})


    CI.p[i,2,j] <- pmax(CI.p[i,2,j],0)
  CI.p[i,3,j] <- pmin(CI.p[i,3,j],1)
     } #end states loop
    } #end times loop for CI.p

## CIs on transition probability matrices##
    CI.trans <- array(0,dim=c(nrow(x@dNs),4,length(x@pos.trans)),
    dimnames=list(rows=rownames(x@dNs),cols=c("est","lower limit",
    "upper limit","var.tp"),dim=paste(x@pos.trans,"transition")))

    for(i in 1:nrow(x@dNs)){ ##loop through times

     for(j in 1:length(x@pos.trans)){ #loop through possible transitions

        idx <- as.numeric(unlist(strsplit(x@pos.trans[j], " ")))
        CI.trans[i,1,j] <- PE <- x@all.ajs[idx[1], idx[2] ,i]
        CI.trans[i,4,j] <- var <- x@out[x@pos.trans[j], x@pos.trans[j], i]
```

```
        switch(ci.trans[1],
"linear" = {
        CI.trans[i,2,j] <- PE - z.alpha * sqrt(var)
         CI.trans[i,3,j] <- PE + z.alpha * sqrt(var)},
"log" = {
        CI.trans[i,2,j] <- exp(log(PE) - z.alpha *
         sqrt(var) / PE)
         CI.trans[i,3,j] <- exp(log(PE) + z.alpha *
          sqrt(var) / PE)},
        "cloglog" = {
        CI.trans[i,2,j] <- 1 - (1 - PE)^(exp(z.alpha * (sqrt(var) /
         ((1 - PE) * log(1 - PE)))))
         CI.trans[i,3,j] <- 1 - (1 - PE)^(exp(-z.alpha * (sqrt(var) /
          ((1 - PE) * log(1 - PE))))))},
"log-log" = {
        CI.trans[i,2,j] <- PE^(exp(-z.alpha * (sqrt(var) /
         (PE * log(PE)))))
         CI.trans[i,3,j] <- PE^(exp(z.alpha * (sqrt(var) /
          (PE * log(PE))))))})

    CI.trans[i,2,j] <- pmax(CI.trans[i,2,j],0)
  CI.trans[i,3,j] <- pmin(CI.trans[i,3,j],1)

     } #end j loop
    } #end times loop
```

```
    list(CI.p=CI.p,CI.trans=CI.trans)
} #end of function

## CIS for distributions ##
Dist.CIs <- function(x,ci.level=0.95,ci.trans="linear"){
z.alpha <- qnorm(ci.level + (1 - ci.level) / 2)
ci.trans <- match.arg(ci.trans,c("linear","log","cloglog","log-log"))
CI.Fs <- array(0,dim=c(nrow(x@Fs),3,length(nodes(x@tree))),
dimnames=list(rows=rownames(x@Fs),cols=c("est","lower limit","upper limit"),
dim=paste("F",nodes(x@tree))))
CI.Gs <- array(0,dim=c(nrow(x@Gs),3,length(nodes(x@tree))),
dimnames=list(rows=rownames(x@Gs),cols=c("est","lower limit","upper limit"),
dim=paste("G",nodes(x@tree))))

    for(i in 1:nrow(x@Fs)){ ##loop through times

      for (j in as.numeric(nodes(x@tree))){ #loop through states

        res.ci.F <- strsplit(colnames(x@Fs), " ")
        col.idx.F <- which(sapply(res.ci.F, function(x) x[2]== j))
        res.ci2.F <- strsplit(colnames(x@Fs.var), " ")
        col.idx2.F <- which(sapply(res.ci2.F, function(x) x[2]== j))

        res.ci.G <- strsplit(colnames(x@Gs), " ")
        col.idx.G <- which(sapply(res.ci.G, function(x) x[2]== j))
        res.ci2.G <- strsplit(colnames(x@Gs.var), " ")
        col.idx2.G <- which(sapply(res.ci2.G, function(x) x[2]== j))
```

```
        CI.Fs[i,1,j]<- PE.F <- x@Fs[i,col.idx.F]
        varestF <- x@Fs.var[i,col.idx2.F]

        CI.Gs[i,1,j]<- PE.G <- x@Gs[i,col.idx.G]
        varestG <- x@Gs.var[i,col.idx2.G]

        switch(ci.trans[1],
"linear" = {
        CI.Fs[i,2,j] <- PE.F - z.alpha * sqrt(varestF)
               CI.Fs[i,3,j] <- PE.F + z.alpha * sqrt(varestF)
        CI.Gs[i,2,j] <- PE.G - z.alpha * sqrt(varestG)
               CI.Gs[i,3,j] <- PE.G + z.alpha * sqrt(varestG)},
"log" = {
        CI.Fs[i,2,j] <- exp(log(PE.F) - z.alpha * sqrt(varestF) / PE.F)
          CI.Fs[i,3,j] <- exp(log(PE.F) + z.alpha * sqrt(varestF) / PE.F)
CI.Gs[i,2,j] <- exp(log(PE.G) - z.alpha * sqrt(varestG) / PE.G)
          CI.Gs[i,3,j] <- exp(log(PE.G) + z.alpha * sqrt(varestG) / PE.G)},
         "cloglog" = {
        CI.Fs[i,2,j] <- 1 - (1 - PE.F)^(exp(z.alpha * (sqrt(varestF) /
          ((1 - PE.F) * log(1 - PE.F)))))
         CI.Fs[i,3,j] <- 1 - (1 - PE.F)^(exp(-z.alpha * (sqrt(varestF) /
           ((1 - PE.F) * log(1 - PE.F)))))
CI.Gs[i,2,j] <- 1 - (1 - PE.G)^(exp(z.alpha * (sqrt(varestG) /
 ((1 - PE.G) * log(1 - PE.G)))))
          CI.Gs[i,3,j] <- 1 - (1 - PE.G)^(exp(-z.alpha * (sqrt(varestG) /
           ((1 - PE.G) * log(1 - PE.G)))))},
"log-log" = {
        CI.Fs[i,2,j] <- PE.F^(exp(-z.alpha * (sqrt(varestF) /
```

```
        (PE.F * log(PE.F)))))
        CI.Fs[i,3,j] <- PE.F^(exp(z.alpha * (sqrt(varestF) /
         (PE.F * log(PE.F)))))
       CI.Gs[i,2,j] <- PE.G^(exp(-z.alpha * (sqrt(varestG) /
        (PE.G * log(PE.G)))))
        CI.Gs[i,3,j] <- PE.G^(exp(z.alpha * (sqrt(varestG) /
         (PE.G * log(PE.G)))))})

    CI.Fs[i,2,j] <- pmax(CI.Fs[i,2,j],0)
  CI.Fs[i,3,j] <- pmin(CI.Fs[i,3,j],1)

    CI.Gs[i,2,j] <- pmax(CI.Gs[i,2,j],0)
  CI.Gs[i,3,j] <- pmin(CI.Gs[i,3,j],1)

     } #end states loop
    } #end times loop for CI

    list(CI.Fs=CI.Fs,CI.Gs=CI.Gs)

} #end of function

## Main Function ##
msSurv <- function(Data,tree,cens.type="ind",LT=FALSE,
d.var=FALSE,B=200,start.states){
   if (any(!(c("id", "stop", "st.stage", "stage")%in%colnames(Data))))
       stop("'Incorrect column names for 'Data'.
       Column names should be 'id','stop','st.stage', or 'stage'.")
```

```
   if(!("start" %in% colnames(Data)) & LT==TRUE)
stop("The 'start' times must be specified for left truncated data.")

   if(!("start" %in% colnames(Data)) & LT==FALSE)  Data=Add.start(Data)


  if (missing(start.states) & LT == TRUE) {
start.probs <- numeric(length(nodes(tree)))
names(start.probs) <- nodes(tree)
start.probs[names(start.probs)== nodes(tree)[which(sapply(inEdges(tree),
function(x) !length(x)>0))]] <- which(sapply(inEdges(tree),
function(x) !length(x)>0))
      warning("'start.states' not specified.  Assuming all individuals
      start in the initial state at time 0.")
  }

   if(!missing(start.states) & LT==TRUE){
start.probs <- numeric(length(nodes(tree)))
names(start.probs) <- nodes(tree)
start.probs[names(start.probs)== names(table(start.states))] <-
table(start.states)/length(start.states)
   }

   n <- length(unique(Data$id)) ## number of individuals in sample
   ns <- length(nodes(tree)) ## number of states

   Cens <- Add.States(tree)
   if(LT) {
```

```
Data = LT.Data(Data)
cp <- CP(tree,Cens$treeLT,Data,Cens$nt.states.LT)
   }

   if(!LT) cp <- CP(tree,Cens$tree0,Data,Cens$nt.states)

    ds.est<-DS(LT="LT",Cens$nt.states,cp$dNs,cp$sum.dNs,cp$Ys,
     Cens="0",cens.type)
    cp.red <- Red(tree,cp$dNs,cp$Ys,cp$sum.dNs,ds.est$dNs.K,
     ds.est$Ys.K,ds.est$sum.dNs.K)

    if(missing(start.states)){
 if(!LT)  start.probs=cp$start.probs
    }

    et <- as.numeric(rownames(cp.red$dNs))

    res.ci2 <- strsplit(colnames(cp.red$dNs), " ")
    a <- sapply(res.ci2, function(x) x[2])
    b <- sapply(res.ci2, function(x) x[3])
    pos.trans <- paste(a,b)
    stay <- paste(Cens$nt.states,Cens$nt.states)
    pos.trans <- sort(c(stay,pos.trans))

   stateoccfn <- stocc(ns,tree,cp.red$dNs.K,cp.red$Ys.K,start.probs)
   ent.exit <- Dist(stateoccfn$ps,ns,tree)

   variances <- var.fn(tree,ns,Cens$nt.states,cp.red$dNs,cp.red$Ys,
```

```
    cp.red$sum.dNs,stateoccfn$all.ajs, stateoccfn$all.I_dA,stateoccfn$ps)

no.start.st <-  length(which(start.probs>0))


   if(cens.type=="ind" & no.start.st==1){
if(d.var==TRUE){
ee.vars <- BS.var(Data,tree,ns,et,cp.red$dNs,cens.type,B,LT,start.states)
var.Fs <- ee.vars$Fs
  var.Gs <- ee.vars$Gs
} else {
    var.Fs=NULL
    var.Gs=NULL
}
   res <- new("msSurv", tree=tree,ns=ns,et=et,pos.trans=pos.trans,
    nt.states=Cens$nt.states,dNs=cp.red$dNs,Ys=cp.red$Ys,
    ps=stateoccfn$ps,all.ajs=stateoccfn$all.ajs,Fs=ent.exit$Fs,
    Gs=ent.exit$Gs,out=variances$out,cov.p=variances$cov.p,
    sum.dNs=cp.red$sum.dNs, dNs.K=cp.red$dNs.K,Ys.K=cp.red$Ys.K,
sum.dNs.K=cp.red$sum.dNs.K,cov.dA=variances$varcov,
all.I_dA=stateoccfn$all.I_dA, Fs.var=var.Fs,Gs.var=var.Gs)
}

   if(cens.type=="ind" & no.start.st>1){
   bsvar <- BS.var(Data,tree,ns,et,cens.type,B,LT,start.states)
   res <- new("msSurv", tree=tree,ns=ns,et=et,pos.trans=pos.trans,
   nt.states=Cens$nt.states,dNs=cp.red$dNs,Ys=cp.red$Ys,
   ps=stateoccfn$ps,all.ajs=stateoccfn$all.ajs,Fs=ent.exit$Fs,
```

```
   Gs=ent.exit$Gs,out=bsvar$out,cov.p=bsvar$cov.p,
   sum.dNs=cp.red$sum.dNs, dNs.K=cp.red$dNs.K, Ys.K=cp.red$Ys.K,
 sum.dNs.K=cp.red$sum.dNs.K,cov.dA=variances$varcov,
 all.I_dA=stateoccfn$all.I_dA,Fs.var=bsvar$Fs.var,
 Gs.var=bsvar$Gs.var)}


  if(cens.type=="dep"){
   bsvar <- BS.var(Data,tree,ns,et,cens.type,B,LT,start.states)
   res <- new("msSurv", tree=tree,ns=ns,et=et,pos.trans=pos.trans,
   nt.states=Cens$nt.states,dNs=cp.red$dNs, Ys=cp.red$Ys,
   ps=stateoccfn$ps,all.ajs=stateoccfn$all.ajs,Fs=ent.exit$Fs,
    Gs=ent.exit$Gs,out=bsvar$out,cov.p=bsvar$cov.p,
    sum.dNs=cp.red$sum.dNs, dNs.K=cp.red$dNs.K,
Ys.K=cp.red$Ys.K,sum.dNs.K=cp.red$sum.dNs.K,
cov.dA=variances$varcov,all.I_dA=stateoccfn$all.I_dA,
Fs.var=bsvar$Fs.var,Gs.var=bsvar$Gs.var)
}
   return(res)
}
```

B Display functions in the msSurv package

Key functions in the R package **msSurv** for displaying nonparametric estimation of right censored and possibly left truncated data.

```
## Transition Probability P(s,t) ##
Pst <- function(object,s=0,t="last",deci=4,covar=FALSE){
    if (!(0 <= s & s < t))
        stop("'s' and 't' must be positive, and s < t")
    if (t <= object@et[1] | s >= object@et[length(object@et)])
        stop("Either 's' or 't' is an invalid time")

if(t=="last") t <- object@et[length(object@et)]

idx <- which(s<=object@et & object@et<=t) #location of those [s,t]
l.idx <- length(idx)

cum.prod <- diag(object@ns)
rownames(cum.prod) <- nodes(object@tree)
red.all.ajs <- array(dim=c(object@ns,object@ns,nrow(object@dNs)),
dimnames=list(rows=nodes(object@tree),cols=nodes(object@tree),
dim=rownames(object@dNs)))

for(i in idx){
cum.prod <- cum.prod %*% object@all.I_dA[,,i]
      red.all.ajs[,,i] <- cum.prod
}

    if(covar == TRUE){

bl.Id <- diag(1,(object@ns)^2) #Ident matrix for Kronecker product
  var.Pst <- array(0, dim=c(dim(object@all.I_dA[,,idx])[c(1, 2)]^2,
  nrow(object@dNs)))
```

```
  colnames(var.Pst) <- rownames(var.Pst) <- paste(rep(nodes(object@tree),
   object@ns), sort(rep(nodes(object@tree),object@ns)))
  Id <- diag(1,object@ns)

for(i in idx){
  if(i==idx[1]) var.Pst[, , i] <- bl.Id%*% object@cov.dA[,,i] %*% bl.Id
  else var.Pst[, , i] <- (t(object@all.I_dA[, , i]) %x% Id) %*%
   var.Pst[, , i-1] %*%((object@all.I_dA[, , i]) %x% Id) +
      (Id %x% red.all.ajs[, , i-1]) %*% object@cov.dA[,,i]  %*%
      (Id%x% t(red.all.ajs[, , i-1]))
  } #end of for idx
    } #end of if var

cat(paste("Estimate of P(",s,",",t,")\n", sep = ""))
        print(round(cum.prod,digits=deci))
        cat("\n")
       if (!is.null(object@out) & covar == TRUE) {
           cat(paste("Estimate of cov(P(",s,",",t,"))\n", sep = ""))
           print(round(var.Pst[,,max(idx)],digits=deci))
       }



}

## State Occ for Specific Time t ##
st.t<- function(object,t="last",deci=4,covar=FALSE){

if(t=="last") t <- object@et[length(object@et)]
```

```
t.loc<- length(object@et[object@et<= t])

cat(paste("The state occupation probabilities at time ",t," are:
\n", sep = ""))
for(i in nodes(object@tree)){
   cat(paste("State ",i,": ",round(object@ps[t.loc,as.numeric(i)],
   deci),"\n",sep = ""))
}
cat("\n")

       if (!is.null(object@out) & covar == TRUE) {
           cat(paste("Covariance Estimates for State
           Occupation Probability: \n", sep = ""))

   for(i in nodes(object@tree)){
      cat(paste("State ",i,": ",round(object@cov.p[t.loc,as.numeric(i)],deci),
      "\n",sep = ""))
   }
       }


}
## State Entry/Exit Time Distribution ##
EntryExit <- function(object,t="last",deci=4,covar=FALSE){
if(covar==TRUE & is.null(object@Fs.var)){
stop(paste("msSurv object does not have variance estimates
for entry/exit time distributions.
Please re-run the msSurv object with the argument
```

```
'd.var=FALSE' and then try again. \n", sep=""))
}
      entry.st <- which(!(sapply(inEdges(object@tree),
       function(x) length(x) == 0)))
initial <- which(!(nodes(object@tree)%in%entry.st))
exit.st <- which(sapply(edges(object@tree),
function(x) length(x) > 0))
terminal <- which(!(nodes(object@tree)%in%exit.st))

if(t=="last") t <- object@et[length(object@et)]
t.loc<- length(object@et[object@et<= t])

cat(paste("The state entry distributions at time ",
t," are:\n", sep = ""))
for(i in entry.st){
   cat(paste("State ",i,": ",round(object@Fs[t.loc,][[i]],deci),
   "\n",sep = ""))

}
   cat(paste("State entry distributions for state ",
   as.character(initial),"is omitted
since there are no transitions into that state."))
cat("\n","\n")

       if (covar==TRUE) {

           cat(paste("Variance Estimates for State Entry
           Distributions: \n", sep = ""))
```

```
   for(i in entry.st){
      cat(paste("State ",i,": ",round(object@Fs.var[t.loc,][[i]],deci),
      "\n",sep = ""))

   } #end of entry.st loop

   cat("Variance estimates of state entry distributions for state ",
   as.character(initial),"is omitted
since there are no transitions into that state.")
     cat("\n")

       } #end of if covar loop


       cat("\n","\n")

cat(paste("The state exit distributions at time ",t,
" are:\n", sep = ""))
for(i in exit.st){
   cat(paste("State ",i,": ",round(object@Gs[t.loc,][[i]],deci),
   "\n",sep = ""))

}
cat("State exit distributions for state(s) ",as.character(terminal),
"is (are) omitted
since there are no transitions into that (those) state(s).")
cat("\n","\n")
```

```
        if (covar==TRUE) {

            cat(paste("Variance Estimates for State Exit Distributions: \n",
             sep = ""))

   for(i in exit.st){
      cat(paste("State ",i,": ",round(object@Gs.var[t.loc,][[i]],
      deci),"\n",sep = ""))
   } #end of entry.st loop

   cat("Variance estimates of state exit distributions for state(s) ",
   as.character(terminal),"is (are) omitted
since there are no transitions into that (those) state(s).")

   cat("\n")

        } #end of if covar loop
} #end of loop


setMethod("print", signature(x="msSurv"),
   function(x, covar = FALSE, ee.distn=TRUE, ...) {
   transient <- as.character(which(sapply(edges(tree(x)),
      function(x) length(x) > 0)))
   absorb <- as.character(which(sapply(edges(tree(x)),
   function(x) length(x) == 0)))
```

```
      trans <- strsplit(colnames(x@dNs),"")
      idx1 <- sapply(trans, function(x) x[4])
      idx2 <- sapply(trans, function(x) x[6])
      trans <- paste(idx1,idx2)

        cat(paste("The specified multistate model has",
         length(transient),
         "transient state(s) and \n",length(absorb),
         "absorbing state(s)\n\n", sep = " "))

        cat("Possible States in this Model:\n")
        print(nodes(x@tree))
        cat("\n")

        cat("Possible Transitions for this Model:\n")
        print(trans)
        cat("\n")

## start of state occupation prob info

        cat("State Occupation Information at time ",
        max(as.numeric(rownames(x@dNs))),": \n",sep="")
        cat("\n")

        cat(paste("Estimates of State Occupation Probabilities",
        "\n", sep = ""))
        print(round(x@ps[nrow(x@ps),],digits=4))
        cat("\n")
```

```
        if (!is.null(x@cov.p) & covar == TRUE) {
            cat("Estimates of Covariance of State Occupation
             Probabilities","\n")
            print(round(x@cov.p[nrow(x@cov.p),],digits=4))
            cat("\n")
        }

        if (ee.distn) {
            cat("Estimates of State Entry Time Distribution","\n")
            print(round(x@Fs[nrow(x@Fs),],digits=4))
            cat("\n")

            cat("Estimates of State Exit Time Distribution","\n")
            print(round(x@Gs[nrow(x@Gs),],digits=4))
            cat("\n")
        }

        cat("\n")

## transition probability info
        cat("Transition Probability Information:", "\n", "\n")
        cat(paste("Estimate of P(",0,",",
        max(as.numeric(rownames(x@dNs))),")\n", sep = ""))

        ## set up currently to do P(0,max(x@et))
        print(round(x@all.ajs[,,dim(x@all.ajs)[3]],digits=4))
        cat("\n")
```

```
        if (!is.null(x@out) & covar == TRUE) {
            cat(paste("Estimate of cov(P(",0,",",
            max(as.numeric(rownames(x@dNs))),"))\n", sep = ""))
            print(round(x@out[, , dim(x@out)[3]],digits=4))
        }

        cat("\n")
        invisible()
})

##
setMethod("show", "msSurv",
   function(object) {

     ## nonterminal states
     transient <- as.character(which(sapply(edges(tree(object)),
      function(object) length(object) > 0)))
     ## absorbing states
     absorb <- as.character(which(sapply(edges(tree(object)),
       function(object) length(object) == 0)))
     trans <- strsplit(colnames(object@dNs),"")
     idx1 <- sapply(trans, function(x) x[4])
     idx2 <- sapply(trans, function(x) x[6])
     trans <- paste(idx1,idx2)

       cat(paste("The specified multistate model has", length(transient),
```

```
        "transient state(s) and ", length(absorb), "absorbing state(s)
        \n\n", sep = " "))

       cat("Possible States in this Model:\n")
       print(nodes(object@tree))
       cat("\n")

       cat("Possible Transitions for this Model:\n")
       print(trans)
       cat("\n")

## start of state occupation prob info
       cat("State Occupation Information:", "\n", "\n")

       cat(paste("Estimates of State Occupation Probabilities","\n",
         sep = ""))
       print(object@ps)
       cat("\n")

           cat("Estimates of State Entry Time Distribution","\n")
           print(object@Fs)
           cat("\n")

           cat("Estimates of State Exit Time Distribution","\n")
           print(object@Gs)
           cat("\n")
```

```
        cat("\n")

## transition probability info
        cat("Transition Probability Information:", "\n", "\n")
        cat(paste("Estimate of P(",0,",",
        max(as.numeric(rownames(object@dNs))),")\n", sep = ""))

        ## set up currently to do P(0,max(object@et))
        print(object@all.ajs[,,dim(object@all.ajs)[3]])
        cat("\n")

        cat("\n")
        invisible()
})

setMethod("summary","msSurv",
   function(object, digits=3,all = FALSE, ci.fun = "linear",
   ci.level = 0.95, stateocc=TRUE, trans.pr=TRUE) {

        if (ci.level <= 0 | ci.level > 1) {
            stop ("confidence level must be between 0 and 1")
        }

        tmp <- MSM.CIs(object,ci.level=0.95)
        times <-object@et

        if(!all){
            dt <- quantile(times, probs = c(0,0.25,0.5,0.75,1))
```

```
        ind <- findInterval(dt,times)
    }

## State Occupation Probability Section
    if(stateocc){
        cat("State Occupation Information:", "\n", "\n")

        if(all){

            for(i in seq(object@ns)){
                cat(paste("State ", i, "\n"))
                sop.sum <- data.frame(time=times,
                estimate=object@ps[,i],
                variance=object@cov.p[,i],lower.ci=tmp$CI.p[,2,i],
                upper.ci=tmp$CI.p[,3,i],Fs=object@Fs[,i],
                Gs=object@Gs[,i])
                print(sop.sum,row.names=FALSE,digits=digits)
                cat("\n")
            } #end of for statement

        } #end of if(all

        else{
            for(i in seq(object@ns)){

                cat(paste("State ", i, "\n"))
                sop.sum <- data.frame(time=times[ind],
                estimate=object@ps[ind,i],
```

```
                    variance=object@cov.p[ind,i],lower.ci=tmp$CI.p[ind,2,i],
                    upper.ci=tmp$CI.p[ind,3,i],entry.d=object@Fs[ind,i],
exit.d=object@Gs[ind,i])

                    print(sop.sum,row.names=FALSE,digits=digits)
                    cat("\n")
                } #end of for statement

            } #end of else
        } #end of if(stateocc)


## Transition Probability Matrix Section
        if(trans.pr){

            cat("Transition Probability Information:", "\n", "\n")

            lt <- length(object@pos.trans)
            tts <- strsplit(object@pos.trans, split = " ")

            if(all){

                i=seq_along(object@pos.trans)[2]
                for (i in seq_along(object@pos.trans)) {

                    cat(paste("Transition", tts[[i]][1], "->", tts[[i]][2], "\n",
                     sep = " "))
```

```
## code to add number at risk and number transitions
## print # events for transitions out of stage, else print # le
dns.name <- ifelse(tts[[i]][1] == tts[[i]][2],
paste("dN", tts[[i]][1], ".", sep=" "), paste("dN",
 object@pos.trans[i], sep=" "))

ifelse(dns.name %in% colnames(object@dNs),
 n.event <- object@dNs[, dns.name],
 n.event <- object@sum.dNs[, dns.name])
ifelse(dns.name %in% colnames(object@dNs.K),
 n.event.K <- object@dNs.K[, dns.name],
 n.event.K <- object@sum.dNs.K[, dns.name])

ys.name <- paste("y", tts[[i]][1], sep=" ")
n.risk <- object@Ys[, ys.name]
n.risk.K <- object@Ys.K[,ys.name]

if (dns.name %in% colnames(object@dNs)) {
    tp.sum <- data.frame(time=times,estimate=tmp$CI.trans[,1,i]
    variance=tmp$CI.trans[,4,i],lower.ci=tmp$CI.trans[,2,i],
    upper.ci=tmp$CI.trans[,3,i],n.risk = n.risk, n.event=n.even
    n.risk.K=n.risk.K,n.event.K=n.event.K)
} else {
    tp.sum <- data.frame(time=times,estimate=tmp$CI.trans[,1,i]
    variance=tmp$CI.trans[,4,i],lower.ci=tmp$CI.trans[,2,i],
    upper.ci=tmp$CI.trans[,3,i],n.risk = n.risk,
    n.remain=n.risk-n.event,n.risk.K=n.risk.K,
    n.remain.K=n.risk.K-n.event.K)
```

```
        }
        print(tp.sum, row.names = FALSE,digits=digits)
        cat("\n")
    } #end of for loop
} #end of if(all)
else{
    for (i in seq_along(object@pos.trans)) {
        cat(paste("Transition", tts[[i]][1], "->", tts[[i]][2], "\n", s
        dns.name <- ifelse(tts[[i]][1] == tts[[i]][2], paste("dN", tts[
         ".", sep=" "),paste("dN", object@pos.trans[i], sep=" "))

        ifelse(dns.name %in% colnames(object@dNs),
         n.event <- object@dNs[, dns.name],
         n.event <- object@sum.dNs[, dns.name])

        ifelse(dns.name %in% colnames(object@dNs.K),
         n.event.K <- object@dNs.K[, dns.name],
         n.event.K <- object@sum.dNs.K[, dns.name])

        ys.name <- paste("y", tts[[i]][1], sep=" ")
        n.risk <- object@Ys[, ys.name]
        n.risk.K <- object@Ys.K[,ys.name]

        if (dns.name %in% colnames(object@dNs)) {
            tp.sum <- data.frame(time=times[ind], estimate=tmp$CI.trans
            variance=tmp$CI.trans[ind,4,i],lower.ci=tmp$CI.trans[ind,2,
            upper.ci=tmp$CI.trans[ind,3,i],n.risk = n.risk[ind],
            n.event=n.event[ind],n.risk.K = n.risk.K[ind], n.event.K=n.
```

```
                    } else {
                        tp.sum <- data.frame(time=times[ind],estimate=tmp$CI.trans[
                      variance=tmp$CI.trans[ind,4,i],lower.ci=tmp$CI.trans[ind,2,i]
                      upper.ci=tmp$CI.trans[ind,3,i],n.risk = n.risk[ind],
                      n.remain=n.risk[ind]-n.event[ind],n.risk.K=n.risk.K[ind],
                      n.remain.K=n.risk.K[ind]-n.event.K[ind])
                    }
                    print(tp.sum, row.names = FALSE,digits=digits)
                    cat("\n")
                } #end of for statement
            } #end of else
        } #end of if trans.pr
} #end of summary function
) #ends setMethod
setMethod("plot", signature(x="msSurv", y="missing"),
          function (x, states="ALL", trans="ALL", plot.type="stateocc",
CI=TRUE,ci.level=0.95,ci.trans="linear",...) {
plot.type=match.arg(plot.type, c("stateocc", "transprob","entry.d","exit.d"))
if(plot.type=="stateocc"){
tmp <- MSM.CIs(x,ci.level=0.95) #Calling CIs
if(states[1]=="ALL") states<-nodes(x@tree)
f.st <- factor(states)
ls <- length(states)
sl <- which(nodes(x@tree)%in%as.numeric(states))

if(CI==TRUE & !is.null(x@cov.p)){
rd <- tmp$CI.p
  dimnames(rd)$dim=gsub("p", "State", dimnames(rd)$dim)
```

```
y <- as.vector(rd[,1,sl])
y2 <- as.vector(rd[,2,sl]) #lower limit
y3 <- as.vector(rd[,3,sl]) #upper limit
x <- rep(as.numeric(dimnames(rd)[[1]]), length(states))
f.st <- as.factor(rep(dimnames(rd)$dim[sl], each=dim(rd)[1]))
## NOTE: add '...' argument below
st.plot <- xyplot(y + y2 + y3 ~ x | f.st,
allow.multiple=TRUE,type="s",lty=c(1,2,2),col=c(1,2,2),...)
st.plot <- update(st.plot,main="Plot of State Occupation Probabilites",
 xlab="Event Times",ylab="State Occupation Probabilities",
 key = list(lines=list(col=c(1, 2, 2), lty=c(1, 2, 2)),
 text=list(c("Est", "Lower CI", "Upper CI")),
 columns=3))
print(st.plot)
 } #end of CIs TRUE
if(CI==FALSE){
rd <- tmp$CI.p
dimnames(rd)$dim=gsub("p", "State", dimnames(rd)$dim)
y <- as.vector(rd[,1,sl])
x <- rep(as.numeric(dimnames(rd)[[1]]), length(states))
f.st <- as.factor(rep(dimnames(rd)$dim[sl], each=dim(rd)[1]))
st.plot <- xyplot(y~x|f.st, type="s",col=1)
st.plot <- update(st.plot,main="Plot of State Occupation Probabilites",
 xlab="Event Times",ylab="State Occupation Probabilities",
 key = list(lines=list(col=c(1), lty=c(1)), text=list(c("Est"))))
print(st.plot)
} #end of no CIs
} #end of state occ plot
```

```
if(plot.type=="transprob"){
tmp <- MSM.CIs(x,ci.level,ci.trans) #Calling CIs
all.trans <- x@pos.trans
cc <- strsplit(all.trans," ")
cc2 <- sapply(cc,function(x) x[1])
cc3 <- sapply(cc,function(x) x[2])
all.trans <- paste(cc2,cc3,sep="")

if(trans[1] =="ALL") trans <- all.trans

rd <- tmp$CI.trans
names(rd) <- paste(trans,"transition")
tr <- which(all.trans%in%trans)

if(CI==TRUE & !is.null(x@out)){
y <- as.vector(rd[,1,tr])
y2 <- as.vector(rd[,2,tr]) #lower limit
y3 <- as.vector(rd[,3,tr]) #upper limit
x <- rep(as.numeric(dimnames(rd)[[1]]),
length(trans))
f.tp <- as.factor(rep(dimnames(rd)$dim[tr],
 each=dim(rd)[1]))
tr.plot <- xyplot(y + y2 + y3 ~ x | f.tp,
allow.multiple=TRUE,type="s",lty=c(1,2,2),col=c(1,2,2),...)
tr.plot <- update(tr.plot,main="Plot of Transition Probabilites",
 xlab="Event Times",ylab="Transition Probabilites",
 key = list(lines=list(col=c(1, 2, 2), lty=c(1, 2, 2)),
```

```
 text=list(c("Est", "Lower CI", "Upper CI")),
 columns=3))
print(tr.plot)
} #end of CIs TRUE

if(CI==FALSE){
rd <- tmp$CI.trans
y <- as.vector(rd[,1,tr])
x <- rep(as.numeric(dimnames(rd)[[1]]), length(trans))
f.tp <- as.factor(rep(dimnames(rd)$dim[tr],
each=dim(rd)[1]))
tr.plot <- xyplot(y~x|f.tp, allow.multiple=FALSE,
type="s",col=1,...)
tr.plot <- update(tr.plot,main="Plot of Transition Probabilites",
xlab="Event Times",ylab="Transition Probabilities",
 key = list(lines=list(col=1, lty=1), text=list("Est"),columns=1))
print(tr.plot)

} #end of no CIs

} #end of trans prob plot

if(plot.type=="entry.d"){
    enter <- as.character(which(!(sapply(inEdges(x@tree),
      function(x) length(x) == 0))))
if(states[1]=="ALL") states<-enter

f.st <- factor(states)
```

```
ls <- length(states)
sl <- which(nodes(x@tree)%in%as.numeric(states))

if(CI==TRUE){
   if(!is.null(x@Fs.var)){
tmp <- Dist.CIs(x,ci.level,ci.trans) #Calling CIs
rd <- tmp$CI.Fs
  dimnames(rd)$dim=gsub("F", "State", dimnames(rd)$dim)
y <- as.vector(rd[,1,sl])
y2 <- as.vector(rd[,2,sl]) #lower limit
y3 <- as.vector(rd[,3,sl]) #upper limit
x <- rep(as.numeric(dimnames(rd)[[1]]), length(states))
f.st <- as.factor(rep(dimnames(rd)$dim[sl], each=dim(rd)[1]))
ent.plot <- xyplot(y + y2 + y3 ~ x | f.st, allow.multiple=TRUE,
type="s",lty=c(1,2,2),col=c(1,2,2))
ent.plot <- update(ent.plot,main="Plot of State Entry Time Distributions",
 xlab="Event Times",ylab="State Entry Time Distributions",
 key = list(lines=list(col=c(1, 2, 2), lty=c(1, 2, 2)),
 text=list(c("Est", "Lower CI", "Upper CI")),
 columns=3))

print(ent.plot)
   }

   else {
rd <- x@Fs
dimnames(rd)[[2]]=gsub("F", "State", dimnames(rd)[[2]])
y <- as.vector(rd[,sl])
```

```
x <- rep(as.numeric(dimnames(rd)[[1]]), length(states))
f.st <- as.factor(rep(dimnames(rd)[[2]][sl], each=dim(rd)[1]))
ent.plot <- xyplot(y~x|f.st, type="s",col=1)
ent.plot <- update(ent.plot,main="Plot of State Entry Time Distributions",
 xlab="Event Times",ylab="State Entry Time Distributions",
 key = list(lines=list(col=c(1), lty=c(1)), text=list(c("Est"))))
print(ent.plot)

   }
} #end of CI False

if(CI==FALSE){
rd <- x@Fs
dimnames(rd)[[2]]=gsub("F", "State", dimnames(rd)[[2]])
y <- as.vector(rd[,sl])
x <- rep(as.numeric(dimnames(rd)[[1]]), length(states))
f.st <- as.factor(rep(dimnames(rd)[[2]][sl], each=dim(rd)[1]))
ent.plot <- xyplot(y~x|f.st, type="s",col=1)
ent.plot <- update(ent.plot,main="Plot of State Entry Time Distributions",
 xlab="Event Times",ylab="State Entry Time Distributions",
 key = list(lines=list(col=c(1), lty=c(1)), text=list(c("Est"))))
print(ent.plot)
} #end of CI False

} #end of entry distribution plot


if(plot.type=="exit.d"){
```

```
transient <- as.character(which(sapply(edges(x@tree),
function(x) length(x) > 0)))
if(states[1]=="ALL") states<-transient

f.st <- factor(states)
ls <- length(states)
sl <- which(nodes(x@tree)%in%as.numeric(states))

if(CI==TRUE){
   if(!is.null(x@Gs.var)){
tmp <- Dist.CIs(x,ci.level,ci.trans) #Calling CIs
rd <- tmp$CI.Gs
dimnames(rd)$dim=gsub("G", "State", dimnames(rd)$dim)
y <- as.vector(rd[,1,sl])
y2 <- as.vector(rd[,2,sl]) #lower limit
y3 <- as.vector(rd[,3,sl]) #upper limit
x <- rep(as.numeric(dimnames(rd)[[1]]), length(states))
f.st <- as.factor(rep(dimnames(rd)$dim[sl], each=dim(rd)[1]))
ent.plot <- xyplot(y + y2 + y3 ~ x | f.st, allow.multiple=TRUE,
type="s",lty=c(1,2,2),col=c(1,2,2),...)
ent.plot <- update(ent.plot,main="Plot of State Exit Time Distributions",
 xlab="Event Times",ylab="State Exit Time Distributions",
 key = list(lines=list(col=c(1, 2, 2), lty=c(1, 2, 2)),
 text=list(c("Est", "Lower CI", "Upper CI")),
 columns=3))
print(ent.plot)
   }
```

```
   else{

rd <- x@Gs
dimnames(rd)[[2]]=gsub("G", "State", dimnames(rd)[[2]])
y <- as.vector(rd[,sl])
x <- rep(as.numeric(dimnames(rd)[[1]]), length(states))
f.st <- as.factor(rep(dimnames(rd)[[2]][sl], each=dim(rd)[1]))
exit.plot <- xyplot(y~x|f.st, type="s",col=1)
exit.plot <- update(exit.plot,main="Plot of State Exit Time Distributions",
 xlab="Event Times",ylab="State Exit Time Distributions",
 key = list(lines=list(col=c(1), lty=c(1)), text=list(c("Est"))))
print(exit.plot)
   } #end of null variance loop
} #end of CI FALSE loop

if(CI==FALSE){
rd <- x@Gs
dimnames(rd)[[2]]=gsub("G", "State", dimnames(rd)[[2]])
y <- as.vector(rd[,sl])
x <- rep(as.numeric(dimnames(rd)[[1]]), length(states))
f.st <- as.factor(rep(dimnames(rd)[[2]][sl], each=dim(rd)[1]))
exit.plot <- xyplot(y~x|f.st, type="s",col=1)
exit.plot <- update(exit.plot,main="Plot of State Exit Time Distributions",
 xlab="Event Times",ylab="State Exit Time Distributions",
 key = list(lines=list(col=c(1), lty=c(1)), text=list(c("Est"))))
print(exit.plot)
} #end of CI FALSE loop
} #end of entry distribution plot
```

```
}#end of function
)
```

C Interval Censored Code

Key functions for nonparametric estimation of interval censored data for the three state tracking model.

```
### Function for reducing the interval censored dataset
redICds <- function(int.cens,o.dat,smooth.fit.Ns){

IC.keep <- which(!int.cens$times%in%names(which(smooth.fit.Ns[,1]==1)))
IC.keep.2 <- which(!int.cens$times%in%names(which(smooth.fit.Ns[,2]==1)))

dat.keep <- which(smooth.fit.Ns[,1]<1)
dat.keep.2 <- which(smooth.fit.Ns[,2]<1)

## The reduced data sets
red.IC.12 <- int.cens[IC.keep,]; red.IC.23 <- int.cens[IC.keep.2,]
red.dat.12 <- o.dat[dat.keep,]; red.dat.23 <- o.dat[dat.keep.2,]

## Number of individuals in each data set
nind.12 <- length(unique(red.dat.12$id))
nind.23 <- length(unique(red.dat.23$id))

list(data.keep=dat.keep,dat.keep.2=dat.keep.2,IC.keep=IC.keep,
IC.keep.2=IC.keep.2,nind.12=nind.12,nind.23=nind.23,
red.IC.12=red.IC.12,red.IC.23=red.IC.23,red.dat.12=red.dat.12,
red.dat.23=red.dat.23, dat.keep=dat.keep,dat.keep.2=dat.keep.2)
```

```
}

ICest <- function(Data,tree,full.data){

## data with sorted inspection times
data <- with(Data, Data[order(Data$times),])
nind <- length(unique(Data$id))

## allowable transitions
   nt.states <- which(sapply(edgeL(tree),
    function(x) length(x$edges)>0))
   lng <- sapply(edges(tree)[nodes(tree)%in%names(nt.states)],
     length)
   trans <- paste(rep(nodes(tree)[nodes(tree)%in%names(nt.states)],
     lng), unlist(edges(tree)[nodes(tree)%in%names(nt.states)]),sep="")

## Indicators I(Ujj'<=cik)
Is <- matrix(0, nrow=length(data$times), ncol=length(trans))
    colnames(Is)=paste("I",trans,sep="")
    rownames(Is)=data$times

for(i in nodes(tree)){## generalized code for any tree ...
if(length(inEdges(tree)[[i]])==0) next
ld <- inEdges(tree)[[i]] #nodes from
ex <- edges(tree)[[i]]  #nodes to
later.stages <- names(acc(tree, i)[[1]])
stages <- c(i, later.stages)
    b <- paste("I", inEdges(tree)[[i]], i,sep="")
```

```
row.idx <- which(data$stage%in%stages)
col.idx <- which(colnames(Is)%in%b)
Is[row.idx,col.idx]<-1
} #end of for loop through nodes

## Initial Fit/Smooth using gpava
bw <- bw.SJ(data$times)
# bw <- 0.4

#storage for initial estiamtes, row names are the inspection times
fit.ls.Ns <- smooth.fit.Ns <- matrix(0, nrow=length(data$times),
 ncol=length(trans))
    colnames(fit.ls.Ns)=colnames(smooth.fit.Ns)=paste("I",trans,sep="")
    rownames(fit.ls.Ns)=rownames(smooth.fit.Ns)=data$times

## Initial fit
fit.ls.Ns[,colnames(Is)] <- apply(Is, 2, function(a) gpava(z=data$times, y=a)$x)
smooth.fit.Ns[,colnames(Is)] <- apply(fit.ls.Ns, 2,
function(x) ksmooth(sort(data$times),
x, kernel="normal",bandwidth = bw, x.points = sort(data$times))$y)


## Counting process for initial fit
nw.Ns <- apply(smooth.fit.Ns,2,function(x) x*nind)

## reducing the data set
red <- redICds(Data,data,smooth.fit.Ns)
```

```
## Constraints to perform isotonic regression
Atot12 <- cbind(1:(length(red$red.dat.12$times)-1),
 2:(length(red$red.dat.12$times)))
Atot23 <- cbind(1:(length(red$red.dat.23$times)-1),
 2:(length(red$red.dat.23$times)))

## Variance computaions for the reduced data set
ids.ts <- red$red.IC.12$id[order(red$red.IC.12$times)] ##for 12
res.Ns <- tapply(smooth.fit.Ns[red$dat.keep,1], ids.ts, function(p) {
mat <- outer(p, p, FUN = function(x,y) x-x*y)
diag(mat) <- p*(1-p)
mat[upper.tri(mat,diag=FALSE) ]<-0
mat[lower.tri(mat,diag=FALSE) ]<-0
return(mat)})
bigmat <- matrix(0,nrow=nrow(red$red.dat.12), ncol=nrow(red$red.dat.12))
idx <- c(0, cumsum(table(red$red.dat.12$id)))
for (i in 1:red$nind.12) {
bigmat[(idx[i]+1):idx[i+1], (idx[i]+1):idx[i+1]] <- res.Ns[[i]]
}
ordered.bigmat <- bigmat[order(red$red.IC.12$times),
order(red$red.IC.12$times)]

ids.ts <- red$red.IC.23$id[order(red$red.IC.23$times)] ## for 23
res.Ns.23 <- tapply(smooth.fit.Ns[red$dat.keep.2,2], ids.ts, function(p) {
mat <- outer(p, p, FUN = function(x,y) x-x*y)
diag(mat) <- p*(1-p)
mat[upper.tri(mat,diag=FALSE) ]<-0
mat[lower.tri(mat,diag=FALSE) ]<-0
```

```
return(mat)})
bigmat <- matrix(0,nrow=nrow(red$red.dat.23), ncol=nrow(red$red.dat.23))
idx <- c(0, cumsum(table(red$red.dat.23$id)))
for (i in 1:red$nind.23) {
bigmat[(idx[i]+1):idx[i+1], (idx[i]+1):idx[i+1]] <- res.Ns.23[[i]]
}
ordered.bigmat.2 <- bigmat[order(red$red.IC.23$times),
 order(red$red.IC.23$times)]

## Weighted GLS fit

fit.gls.N12 <- activeSet(Atot12, "LS", y = Is[red$dat.keep,1],
weights=diag(ginv(ordered.bigmat)))$x
fit.gls.N23 <- activeSet(Atot23, "LS", y = Is[red$dat.keep.2,2],
weights=diag(ginv(ordered.bigmat.2)))$x

bw12 <- bw.SJ(red$red.dat.12$times)
bw23 <- bw.SJ(red$red.dat.23$times)

ifelse(max(red$red.dat.12$times)>max(red$red.dat.23$times)
maxt <- max(red$red.dat.12$times),maxt<-max(red$red.dat.23$times))

Ngrid <- nrow(data)*2
timegrid <- seq(from=0,to=maxt+2*bw,length.out=Ngrid-length(data$times))
timegrid <- sort(c(timegrid,data$times)) #adding inspection times
it.idx <- which(timegrid%in%red$red.dat.12$times)
it.idx2 <- which(timegrid%in%red$red.dat.23$times)
```

```
smooth.glsfit.Ns1 <- ksmooth(red$red.dat.12$times, fit.gls.N12,
 kernel="normal", bandwidth = bw12, x.points = timegrid)$y
smooth.glsfit.Ns2 <- ksmooth(red$red.dat.23$times, fit.gls.N23,
 kernel="normal", bandwidth = bw23, x.points = timegrid)$y

## Filling in Nans and putting in matrix
smooth.glsfit.Ns <- matrix(0, nrow=length(timegrid), ncol=length(trans))
smooth.glsfit.Ns[,1] <- na.locf(smooth.glsfit.Ns1)
smooth.glsfit.Ns[,2] <- na.locf(smooth.glsfit.Ns2)

## Counting Process and Risk Set
Ns <- matrix(0, nrow=length(timegrid), ncol=length(trans))
colnames(Ns) <- trans; rownames(Ns) = timegrid
Ns[,1] <- smooth.glsfit.Ns[,1]*red$nind.12
Ns[,2] <- smooth.glsfit.Ns[,2]*red$nind.23

##Risk set for initial state (ie: Y1)
Y1 <- as.vector(rep(nind,length=nrow(Ns)))
Y1 <- Y1-Ns[,1]

##Risk set for the transient state (ie: Y2)
Y2 <- c(0,Ns[,1]-Ns[,2])[-nrow(Ns)]
Y2[Y2<=0] <- 0

## ESTIMATION FOR IC DATA

dNs <- apply(Ns,2,function(x)diff(x))
```

```
N12.Y1 <- dNs[,1]/Y1[-length(Y1)]
N12.Y1[is.nan(N12.Y1)] <- 0

N23.Y2 <- dNs[,2]/Y2[-length(Y2)]
N23.Y2[is.nan(N23.Y2)] <- 0
N23.Y2[is.infinite(N23.Y2)] <- 0

## State Occupation Probabilities

P1 <- exp(-cumsum(N12.Y1))

cs <- vector(length=(length(timegrid)-1))
P2 <- vector(length=length(timegrid)-1)
term1 <- P1*N12.Y1

for(i in 1:(length(timegrid)-1)){
term2 <- numeric(i)
for (j in 1:i) {
term2[j] <- exp(-sum(N23.Y2[j:i]))
}
P2[i] <- sum(term2[1:i] * term1[1:i])
}

P3 <- cumsum(P2*N23.Y2)

P1 <- c(1,P1); P2 <- c(0,P2); P3 <- c(0,P3)

## State Entry/Exit Distributions
```

```
## Exit for state 1 & 2, Entry for state 2 & 3

G1<-(P2+P3)/(P2[length(P2)]+P3[length(P3)])
G1[which(G1>1)] <-1
G2<-P3/P3[length(P3)]

F2<-(P2+P3)/(P2[length(P2)]+P3[length(P3)])
F2[which(F2>1)]<-1
F3<-P3/P3[length(P3)]

list(nw.L1=nw.L1,w.L1=w.L1,Ns=Ns,fullfit=fullfit,ws.L1=ws.L1)

} #end of function
```

D Data generation function for interval censored data

This is a function used to generate interval censored data for the 3 state Markov model with Weibull waiting times and uniform censoring times.

```
wu.sim <- function(N,wshape=3,wscale=1){
id=1:N

V1<-round(rweibull(N,shape=wshape,scale=wscale),4)
V2<-round(qweibull(pweibull(V1,shape=wshape,scale=wscale)
+runif(N,0,1)*(1-pweibull(V1,shape=wshape,scale=wscale)),
shape=wshape,scale=wscale),4)

#Generating states, for now assuming every makes these transitions
```

```
s1 <-rbinom(N,1,1)+1; s2 <-rbinom(N,1,1)+2

##creating data frame
d1 <- data.frame(id=id,times=V1,stage=s1)
d2 <- data.frame(id=id,times=V2,stage=s2)
data <- rbind(d1,d2)
data <- with(data,data[order(data$id),])

## FULL DATA
full.data <- with(data,data[order(id,times),])
augment <- cbind(1:N, rep(0,N), rep(1, N))
colnames(augment) <- names(full.data)
full.data <- rbind(full.data, augment)
full.data <- with(full.data, full.data[order(id,times),])

## INTERVAL CENSORED DATA
ninspect <- sample(2:4, N, replace=TRUE)
indmax.times <- tapply(full.data$times, full.data$id,
function(x) max(x) + 2*mad(x))
inspection.times <- unlist(mapply(function(x,y)
runif(x,0,max(y)), ninspect, indmax.times))
id.inspect <- rep(1:N,ninspect)
inspection.states <- by(full.data,full.data$id,
function(x) sapply(inspection.times[id.inspect==x$id],
function(y) x$stage[max(which(y>x$times))]))
## warnings are ok ...

int.cens <- data.frame(id=id.inspect,times=inspection.times,
```

```
stage=unlist(inspection.states))
int.cens <- with(int.cens,int.cens[order(id,times),])

list(int.cens=int.cens,full.data=full.data)
}
```

CURRICULUM VITAE

NAME: Amanda Nicole Ferguson

ADDRESS: Department of Bioinformatics and Biostatistics
University of Louisville
Louisville, KY 40202

EDUCATION:

M.S., Statistics
University of Georgia
2004
B.S., Mathematics & Statistics
University of Georgia
2002

PROFESSIONAL SOCIETIES:

American Statistical Association (ASA)

NATIONAL MEETING PRESENTATIONS:

msSurv: an R package for nonparametric estimation of multistate models
at Joint Statistical Meetings (JSM) in Vancouver, BC, August 2011.

PUBLICATIONS:

1. Ferguson, A. N., Datta, S., Brock, G. N. msSurv, an R package for nonparametric estimation of multistate models. (Submitted).

2. Datta, S. and Ferguson, A. N. (2011) Nonparametric estimation of marginal temporal functionals in a multistate model. In *Recent Advances in System Reliability: Signatures, Multi-state Systems and Statistical Inference.* Edited by Ilia Frenkel and Anatoly Lisnianski, Springer, New York, to appear.

3. Brock, G. N., Mostajabi, F., Ferguson, N., Carrubba, C. J., Eng, M., Buell, J. F., and Marvin, M. R. (2011). Prophylaxis against de novo hepatitis B For liver transplantation utilizing hep B core (+) donors: does hepatitis B immunoglobulin provide a survival advantage? *Transplant International*, **24**(6):570-581.