Inference in Semi-Markov Models with Panel Data

Cason Wight

A selected project submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Statistics

Brigham Young University

April 2021

ABSTRACT

Inference in Semi-Markov Models with Panel Data

Cason Wight
Department of Statistics, BYU
Master of Science

Semi-Markov processes model waiting times and transition probabilities for multi-state scenarios. In many applications, data are collected at intermittent points in time where the state of a process is observed. These observation times do not show the true times or paths of transitions. Intermittently observed measurements such as these are known as *panel data*. Our purpose is to estimate the parameters of a semi-Markov model with panel data. The state-of-the-art technique uses a stochastic EM algorithm for inference. Two setbacks to this method are sampling inefficiency and slow convergence. We propose improvements to this method by leveraging properties of semi-Markov processes.

Keywords: Markov models, panel data, stochastic process, likelihood, FFT, EM algorithm

ACKNOWLEDGMENTS

I want to first thank my wife Megan for continually supporting me through this project and the program. I also want to acknowledge how helpful and kind both Dr. Warr and Dr. Hartman have been throughout my time at BYU. As teachers, mentors, and friends, they have made me feel excited and confident as I make school and life decisions. Lastly, I would like to thank Adam, Brandon, Daniel, and Devin for providing feedback and comradery over the past few years.

CONTENTS

CHAPTER 1

INTRODUCTION

Multistate statistical models are vital in modeling many processes. One area that gains particular benefit from these models is disease progression. The medical profession has categorized significant states for diseases such as cancer, Alzheimer's, Parkinson's, and chronic kidney disease to name just a few. These categorizations naturally map to states in a model. One challenge that arises from this type of model is that patients are rarely observed continuously. The state of a patient is typically known only at appointments or other intermittent observation times; data observed at intermittent intervals like this are known as *panel data*. With the incomplete and censored observations that are so natural in a medical setting, researchers should use as much information from the collected data as possible.

One common multistate model is a semi-Markov Process (SMP). A Markov process is a finite state process where the current state is the only relevant information affecting future states or sojourn times. This characteristic is known as the memoryless property. The memoryless property of Markov models require exponentially distributed sojourn times. An SMP generalizes the concept of a Markov process to include any continuous-time distribution for sojourn times. This generalization reduces the memoryless property to hold only at times of transition. Thus, in SMPs, both the current state and the time since the last transition affect future movement.

As a simple example of an SMP, consider the multistate process shown in Figure 1.1. In this SMP, there are four states. Forward movement along the states is possible. Backward transitions are only available from state 2 to state 1 and from state 3 to state 2, making state 4 an *absorbing* state. The arrows represent possible sojourn paths, each with their own distribution. When a state has more than one possible out-transition path (such as paths from state 2, in this example), there are probabilities associated with each possible path.

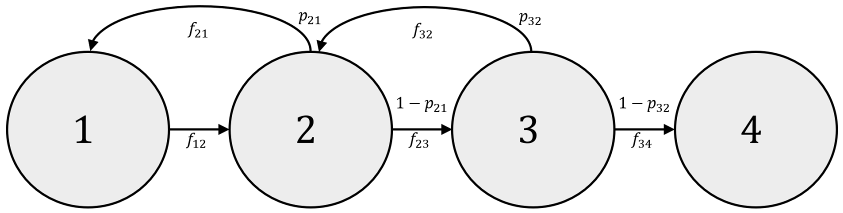

Figure 1.1: Simple 4-state semi-Markov Process of diabetic retinopathy progression, introduced in Marshall and Jones (1995). State 1 represents no retinopathy, state 2 represents microaneurysms, state 3 represents intermediate background retinopathy, state 3 represents preproliferative retinopathy, and state 4 represents preproliferative and proliferative retinopathy. State 4 is considered an *absorbing* state because transitions out of this state are not possible and/or modeled.

In this paper, we are particularly interested in SMPs as they relate to medical patients, but this work could be generalized to other settings as well. We seek to solve for the parameters of SMPs (both the transition probabilities and the parameters for the sojourn time distributions) using only panel data. Given parameter estimates, many useful quantities can be calculated. Some examples include mortality rates after some number of years, expected time until reaching some state, or probability of disease progression, among many others. The proper calculation of these quantities is important to clinicians and insurers, and thus the parameter estimation must be sound.

We focus on inference for a SMP involving diabetic retinopathy patients, introduced in Marshall and Jones (1995), whose SMP is shown by Figure 1.1. Diabetic retinopathy occurs when high blood pressure damages blood vessels to the eyes, causing vision impairment. The data we analyze come from 277 patients who had type I diabetes for at least 5 years. Each patient had at least two visits at the University of Colorado Health Sciences Center. At each visit, the patients' state was observed. No retinopathy is represented by state 1, microaneurysms are labeled state 2, intermediate background retinopathy is state 3, and preproliferative/proliferative retinopathy is state 4. Proliferative retinopathy is an advanced stage where new blood vessels are growing, which can cause permanent damage. In this work, we model the progression of diabetic retinopathy by estimating the transition probabilities, as well as the distributional parameters for all sojourn paths. Accurate estimates for the probability of progression, distribution of progression time, and other quantities can help give understanding for how diabetic patients' retinopathy evolves over time.

Many modeling techniques for SMPs with panel data are constrained by simplifications, additional assumptions, and/or computational challenges. Several of these drawbacks are detailed in the literature review. In numerous studies, the transition times from one state to another are treated as known and having occurred on the date/time they were documented, even though the transition truly occurred earlier. This assumes the inherently 'incomplete' data as 'complete' and introduces bias into a study. Other studies assume minimal movement between panel observations. Due to the complexity introduced by incomplete data, unrealistic assumptions such as these are often used to solve for the parameters of the SMP. Medical researchers are especially adverse to biased inference and want to reduce the number of model assumptions. Therefore we approach the problem with minimal assumptions which are: the process adheres to the semi-Markov property, the chosen parametric models are correct, and the patients are mutually independent.

The state-of-the-art method, proposed by Aralis and Brookmeyer (2019), uses a stochastic sampling procedure, coupled with the EM algorithm to estimate the parameters of the SMP. The EM algorithm is a method for imputing missing data to estimate desirable parameters through an iterative process. In this algorithm, each iteration starts by assuming values for the parameters of interest. In the *expectation* step, these values are used to calculate the expectations of the missing information (when state transitions occurred). In the *maximization* step, the imputed transition times, calculated in the expectation step, are used to obtain maximum likelihood estimates (MLEs) for the parameters of interest. These new MLEs then feed in as the assumed parameters of the next iteration of the algorithm. See Redner and Walker (1984) for an introduction to the EM algorithm. The Aralis and Brookmeyer (2019) method uses rejection sampling of multistate trajectories to obtain stochastic expectations of the missing data.

The stochastic EM method allows for flexible sojourn time distributions and produces unbiased estimates; however, a major downside to this approach is the computational inefficiency. Because the expectation step is fully stochastic, the EM algorithm converges slowly. Additionally, the sampling procedure produces abundant waste; this inefficiency can be exacerbated by inaccurate initial parameters estimates. In this work, we introduce improvements to this method by employing convenient properties of the SMP. These improvements eliminate the inefficiencies of rejection sampling of patient trajectories with directly sampled first-passage trajectories. The fully stochastic expectation is replaced by conditional expectations from the partial patient trajectories. By improving the current methodology, we allow for the same flexibility in sojourn time distributions and maintain unbiased estimation. The proposed method aims to reduce and bound computational time.

1.1 Literature Review

There have been many attempts to incorporate incomplete data in multistate models for survival data; most of these approaches approximate the likelihood function for the incomplete observations. Some of these methods are presented here. Kryscio and Abner (2013) summarize the typical problems that are encountered with real semi-Markov data (especially medical). Kalbfleisch and Lawless (1985) define an approach to approximating parameters of a Markov model with panel data using direct maximum likelihood estimation. This method includes implementation of covariates. An R package has been developed by Jackson (2011) that fits a Markov model based on this approach. Lebreton and Cefe (2002) consider multistate models for capture-recapture data where an individual can transit to more than one state between captures. With the assumption that transition times are known, Yau and Huzurbazar (2002) incorporates multiple state transitions between observation periods. Foucher et al. (2007) modeled incomplete data with some restrictions on the shape of the hazard function. Ferguson et al. (2012) created a package in R that estimates multi-state models using a non-parametric paradigm. The package works with either panel observations or exact transition times.

Many have also worked to solve the problem in an SMP model with panel data. Steps have been made towards a robust solution, but most solutions rely heavily on strict assumptions or inefficient sampling. The approach of Chen and Tien (2004) includes a pseudo likelihood approach when transition times are incomplete. Gentleman et al. (1994) consider panel data with some additional restrictions. Kang and Lagakos (2007) propose a direct maximum likelihood approach to estimating semi-Markov parameters, by looking at probabilities for the number of transitions between panel points. This method requires that one or more transitions be modeled using an exponential distribution; even with this simplification, the calculations are challenging and left to the modeler. A likelihood function approach that incorporates interval censoring has been established by Foucher et al. (2010), but requires the assumption of Weibull-distributed movement between states. Lange and Minin (2013) assume a latent Markov model, with multiple latent states mapping to each observed state. The latent structure involves phase-type distributions (see Asmussen et al. (1996) for more on phase-type distributions), which have a tractable likelihood function for panel data. They do not always yield identifiable parameters for the SMP. Titman (2014) also uses phase-type distributions to get a tractable likelihood to estimate the parameters of a semi-Markov model. A direct maximum likelihood approach has been established in Wei and Kryscio (2016), but assumes Weibull or Exponentially distributed transitions and requires quasi-Monte Carlo techniques for high-level integrals. Another method introduced by Mohammadi (2020) has a partial likelihood approach, but also assumes Weibull or Gamma-distributed sojourn times. Aralis and Brookmeyer (2019) propose a stochastic EM approach to finding the likelihood. We consider this the state-of-the-art method for incorporating panel data in SMPs and will discuss it in more detail in Section 2.1.

CHAPTER 2

METHODOLOGY

The solution introduced in this paper improves the SEM methodology proposed by Aralis and Brookmeyer (2019). This approach iterates between filling in the "incomplete" data implied by the panel data and producing converging estimates for the parameters of the SMP.

An SMP is a generalization of a Markov process, relaxing the requirement of exponentially distributed sojourn times. Here, we closely follow the definition of SMP given by Kulkarni (2016). Consider a multistate process, $\{Z(t), t \geq 0\}$, with S possible states. This process begins in state Z_0 at time $t = 0$. This process remains in Z_0 for some sojourn time, Y_1, at which it transitions into state Z_1. This continues generally for all Z_m, where the process arrives in state Z_{m+1} after sojourn time Y_{m+1}, $m \geq 0$. At any given point in time $t \geq 0$, the state of the process is $Z(t) \in \{1, \ldots, S\}$. Let $t'_m = \sum_{i=1}^{m} Y_i$ be the time of the mth transition for a process. This process is called a SMP if the following holds:

Definition 1 (Semi-Markov Process) *The stochastic process $\{Z(t), t \geq 0\}$ is SMP if it has a finite state space S and the following holds for all $\{Z_0, (Z_m.Y_m)\}, m \geq 1$:*

$$\begin{aligned}&\Pr\left(Z(t'_{m+1}) = j, Y_{m+1} \leq t \middle| \left(Z(t'_m) = i, Y_m\right), \left(Z(t'_{m-1}), Y_{m-1}\right), \ldots, (Z(0))\right)\\ &= \Pr\left(Z(t'_1) = j, Y_1 \leq t \middle| (Z(0) = i)\right), \quad \forall\, i, j \in \{1, 2, \ldots, S\}.\end{aligned} \tag{2.1}$$

This definition allows sojourn times between states to follow any continuous-time distribution. For inference on SMPs with panel data, we appeal to many useful properties of the SMP.

2.1 Improved Stochastic Expectation-Maximization Algorithm

The current method alters the expectation maximization (EM) algorithm, substituting the expectation calculation with a fully stochastic expectation step. Given only panel data, the true transition times between states are unknown. The transition path is also unknown. There is no current closed-form solution for the expectation of the transition path and times, given the panel data. The current method takes parameter estimates from the maximization step and randomly generates semi-Markov process data, using these parameters, until they have samples that would produce the same panel data as observed, given the panel observation times. The accepted samples are taken as true, fully observed transitions and replace expectation calculations. With *complete* data, MLEs are easily obtained for all parameters of the semi-Markov model. The parameter estimates of the SEM algorithm converge as the number of samples for each panel set increases.

The proposed method reduces the stochastic variation and limits the excessive sampling of the previous approach, at the cost of more calculation. Instead of generating fully random transition sets, this method appeals to the properties of semi-Markov models to generate alternative types of samples.

This paper will introduce the method in two parts: the expectation step and the maximization step. Before providing details, a few important notational and computational conventions need to be presented. These definitions are provided by Warr and Collins (2015) and shown in Table 2.1; the reader should refer to this work for details on how to compute the various quantities of interest.

In an SMP, both the transition probability matrix $\mathbf{p}$ and the sojourn time distribution parameters β are estimated. Even when β is unknown, distributional families $f_{i,j}(y;\beta)$ are assumed $\forall\ i,j \in \{1,2,\ldots,S\}$. In an SMP, the available sojourn paths are also assumed (that is, the set of pairs $(i,j) : p_{i,j} \neq 0$). With known $\mathbf{p}$ and β, all calculations in Table 2.1 are available.

Table 2.1: Notation for SMP Quantities. *Assumes that patient is in state i at time 0.

S	The number of states in the SMP
$Z(y)$	The state of a patient at time y
$p_{i,j}$	*The probability that the next state of a patient will be j
$f_{i,j}(y)$	*The PDF of the direct sojourn time into state j at time y
$P_{i,j}(t)$	$\Pr(Z(t) = j \mid Z(0) = i)$ *The probability the patient is in state j at time t
$G_{i,j}(y)$	$\Pr(N_j(y) > 0 \mid Z(0) = i)$ *The CDF of first-passage movement into state j at time y
$g_{i,j}(y)$	$\frac{d}{dy} \Pr(N_j(y) > 0 \mid Z(0) = i)$ *The PDF of first-passage movement into state j at time y
$g_{i,j}^{(k)}(y)$	$g_{i,j}^{(k)}(y) = g_{i,j}(y) \star [g_{j,j}(y)]^{\star(k-I[i=j])}, \quad k \in \{0, 1, 2, \dots\}$ *The PDF of k-passage movement into state j at time y
$q_{i,j}(y)$	$p_{i,j} f_{i,j}(y)$ *The *transmittance* of the direct passage into state j at time y
$H_{i,i}(y)$	$\int_0^y \sum_{j=1}^S q_{i,j}(\xi) d\xi$ *The probability that the patient has transitioned out of state i by time y

In the SEM algorithm, $\mathbf{p}$ and β are assumed known in each stochastic expectation step. In the first iteration, these parameters are initialized by some starting guess, which ideally would be close to the final parameter estimates. $\mathbf{p}$ and β are estimated in the maximization step at each iteration. This algorithm continues until the parameter estimates converge. In our proposed method, the stochastic expectation step may need to include incrementally more samples to achieve convergence, similar to the existing approach.

Expectation Step

With the SEM algorithm and necessary SMP calculations introduced, we now suggest an improved stochastic expectation sampling procedure. With panel data, each patient's observation times, $\mathbf{t}_n$, and observed states, $\mathbf{z}_n = \left[Z(t_{n,1}), Z(t_{n,1}), \dots, Z(t_{n,M_n)}) \right]'$, are used to obtain a set of exact transition times $\mathbf{t}'_n$, corresponding to the times of arrival into the states observed by $\mathbf{z}_n$. The transition times $\mathbf{t}'_n$ are calculated as an expectation. Because the transition path is unknown, we partially sample this path. Instead of full trajectories, we sample the number of arrivals of the patient to state $z_{n,m}$ between the two observation times $t_{n,m-1}$ and $t_{n,m}$.

Consider again the SMP of Figure 1.1. Take the simple example of a patient where $\mathbf{z}_1 = \begin{bmatrix}1, 2\end{bmatrix}'$ and $\mathbf{t}_1 = \begin{bmatrix}0, t\end{bmatrix}'$. The patient may have had an appointment at time 0, when a medical issue was diagnosed, and another appointment at time t, when the issue was observed to have progressed. It is known that there was at least one transition from state 1 to state 2 between time 0 and time t, but the path is unclear. Figure 2.1 displays some of the possible trajectories of the patient between the two observation times. Another way of looking at the trajectory is that there is a known direct transition from $z_{1,1}$ to $z_{1,2}$, but an *unknown* number, $k_{1,1}$, of succeeding loops from $z_{1,2}$ back into $z_{1,2}$ between time 0 and time t. We outline a sampling procedure for the number of first-passage loops, $k_{n,m}$, for each panel observation m of each patient n.

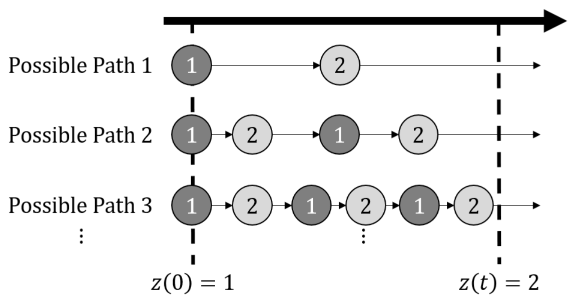

Figure 2.1: With the SMP given in Figure 1.1 and a single patient with panel data $\mathbf{z}_1 = \begin{bmatrix}1, 2\end{bmatrix}'$ and $\mathbf{t}_1 = \begin{bmatrix}0, t\end{bmatrix}'$, there are many possible paths that the patient could have taken. Each involve one first-passage (direct, in this case) transition from state 1 to state 2 and $k \in \{0, 1, 2, \dots\}$ first–passage transitions from state 2 back into state 2.

With the partial transition path known, the expected times of the final transitions $\mathbf{t}'$ before the observation times $\mathbf{t}$ are calculable. The convolutional sum of a first-passage distribution from a state i into state j, with k first-passage distribution loops from state j back into state j is known as a *k-passage* distribution from state i to state j. Thus, the result of the stochastic expectation step is a set of sampled k-passages and the expected times $\mathbf{t}'$ of these k-passages, from the k-passage distributions. An increasing number of samples Φ, for each patient are obtained as the SEM algorithm iterates. We recommend setting the number of samples, Φ, to the iteration number of the algorithm. Thus on the first iteration of the SEM algorithm, each patient receives 1 sampled expectation $(\mathbf{k}_{n,\phi}, \mathbf{t}'_{n,\phi})$, and on the second each receives 2, and so on. In many cases, the algorithm may converge even if Φ remains at 1.

Before outlining the sampling procedure, there is one adjustment to the SMP that must be made for each panel observation pair. For a patient n, the expected k-passage transition times $\mathbf{t}'_n$ represent the expected times of the *final* transitions before the observed panel times $\mathbf{t}_n$. Thus, when a final transition time t'_{m-1} is sampled, the implicit restriction is that there are no additional state transitions between time t'_{m-1} and t_m. Figure 2.2 shows, by the shaded region, when the patient is restricted from movement. It is necessary to incorporate this requirement when obtaining the stochastic expectation for panel observation m. Thus when sampling the number of loops and calculating the final time for the subsequent $k_{n,\phi,m+1}$-passage movement, the outward sojourn time distributions from $z_{n,m}$ should be truncated such that they are greater than $t_{n,m} - t'_{n,\phi,m}$.

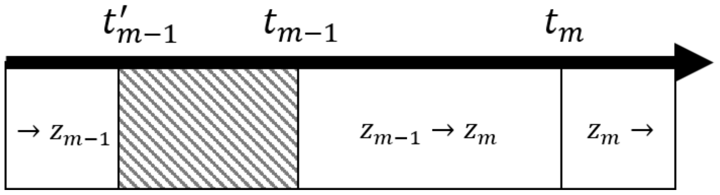

Figure 2.2: After obtaining the expected sojourn time t'_{m-1}, the patient's sampled trajectory should have no movement between this time until after the panel time t_{m-1}. Thus all outward transitions from z_{m-1} are truncated.

For the stochastic expectation procedure, first-passage distributions are needed. It is inappropriate to simply truncate the passage-distributions when calculating passage probabilities or expected transition times. We propose adding an additional state to the existing SMP that mirrors the exit state of interest, $z_{n,m-1}$. This new state has the same departing transition paths and probabilities as the exit state, but the sojourn times are truncated to be above $t_{n,m} - t'_{n,\phi,m}$; there are no in-transitions for the new state. The $S \times S$ transition probability matrix would now become a $S+1 \times S+1$ matrix. The departing transition sojourn time distributions $f_{i',j}(y)$ will be truncated as follows:

$$f_{i',j}(y) = \frac{f_{i,j}(y) I\left[y > t_{n,m} - t'_{n,\phi,m}\right]}{\int_{t_{n,m}-t'_{n,\phi,m}}^{\infty} f_{i,j}(y)dy}. \tag{2.2}$$

Consider the example where $z_{n,m-1} = 2$ and $z_{n,m} = 1$. Suppose that we have already performed the stochastic expectation step for the $m-1$ panel observation, so we already know $t'_{n,\phi,m-1}$. We implicitly assume then that there are no transitions between time $t'_{n,\phi,m-1}$ and $t_{n,\phi,m-1}$. Then in order to sample $k_{n,\phi,m}$ and calculate $t'_{n,\phi,m}$, we would need to incorporate this period of no movement. We will then append a new state to the existing SMP that represents a state for $z_{n,m-1}$ that includes a truncated sojourn time distribution. Figure 2.3 shows how the existing SMP would be adjusted, where dashed lines represent left-censored transition distributions and the dashed circle is the additional state. If there is movement back into and then out of state $z_{n,m-1}$ before $t_{n,m}$, the truncation would no longer be relevant. Thus this additional state will have no inward paths.

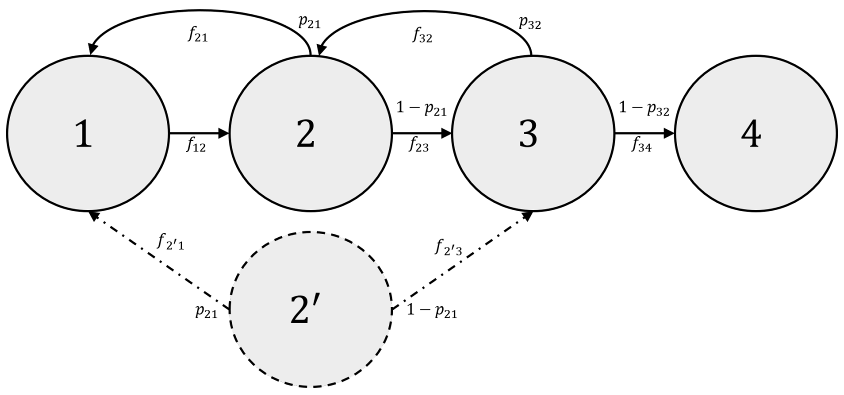

Figure 2.3: The additional state resolves the truncation requirement involved in the stochastic expectation procedure. The dashed circle represented the added state. There are no inward transition paths into this state, and the outward transition paths are identical to those of state $z_{n,m} = 2$, with truncation on the departure transition distributions.

We append the current notation so that i', the additional state, replaces state i, which mirrors state i with no inward transitions and truncated departure distributions.

For the sampling procedure, let $i = z_{n,\phi,m-1}$, $j = z_{n,\phi,m}$, and $\Delta = t_{n,m,\phi} - t'_{n,\phi,m-1}$. The convolutional sum of two density functions $f(t)$ and $g(t)$ is denoted $(f \star g)(t)$. The convolution of a density function $f(t)$, ℓ times, is denoted $[f(t)]^{\star\ell}$, thus for example, $f(t)\star f(t) = [f(t)]^{\star 2}$. The k-passage distributions described above can then be defined as in Table 2.1. The sample probabilities used to obtain $k_{n,m,\phi}$ is then calculated as follows:

$$\begin{aligned}
&\Pr\left(K = k_{n,\phi,m} | Z(t'_{n,\phi,m-1}) = i', Z(t_{m,\phi,m}) = j\right) \\
&= \Pr\left(K = k_{n,\phi,m} | Z(0) = i', Z(\Delta) = j\right) \\
&= \frac{\Pr\left(K = k_{n,\phi,m}, Z(\Delta) = j | Z(0) = i'\right)}{\Pr\left(Z(\Delta) = j | Z(0) = i'\right)} \\
&= \frac{\Pr\left(K = k_{n,\phi,m}, Z(\Delta) = j | Z(0) = i'\right)}{\sum_{\ell=0}^{\infty} \Pr\left(K = \ell, Z(\Delta) = j | Z(0) = i'\right)},
\end{aligned} \tag{2.3}$$

where $\Pr\left(K = k_{n,\phi,m}, Z(\Delta) = j | Z(0) = i'\right) = \int_0^\Delta g_{i',j}^{(k_{n,\phi,m})}(y)\left(1 - H_{j,j}(\Delta - y)\right) dy$.

As previously mentioned, all calculations in Table 2.1 are available when $\mathbf{p}$ and $\boldsymbol{\beta}$ are known. The above probabilities are calculated using only these values, so the probabilities defined in Equation 2.3 can be used for sampling $\mathbf{k}_{n,\phi}$. Given the trajectory from $z_{n,\phi,m-1}$, as well as the maximum sojourn time $t_{n,\phi} - t'_{n,\phi,m-1}$, the truncated k-passage can be used to calculate the expected sojourn time $y_{n,\phi,m}$ and then find the end time $t'_{n,\phi,m}$ of the movement before the panel observation time $t_{n,\phi,m}$:

$$\begin{aligned}
t'_{n,\phi,m} &= t'_{n,\phi,m-1} + E(Y_{n,\phi,m} | t'_{n,\phi,m} < t_{n,\phi,m}, i, j, k_{n,\phi,m}, t'_{n,\phi,m-1}, t_{n,\phi}) \\
&= t'_{n,\phi,m-1} + \frac{\int_0^\Delta y g_{i',j}^{(k_{n,\phi,m})}(y)\left(1 - H_{j,j}(\Delta - y)\right) dy}{\int_0^\Delta g_{i',j}^{(k_{n,\phi,m})}(y)\left(1 - H_{j,j}(\Delta - y)\right) dy}
\end{aligned} \tag{2.4}$$

In the case that $k_{n,\phi,m}$ is sampled as 0 and $i = j$, the expected time $t'_{n,\phi,m} = t'_{n,\phi,m-1}$ because there was no transition sampled between these two observation times.

The incoming relevant pieces of information for the stochastic expectation step are the assumed parameters of the SMP $\mathbf{p}$ and $\boldsymbol{\beta}$, the sets of observation times $\{\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_N\}$, and the sets of observed states $\{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N\}$. From these values, the number of first-passage loops for each panel observation are sampled $\{\mathbf{k}_1, \mathbf{k}_2, \ldots, \mathbf{k}_N\}$. The sets of conditional expectations for the sampled k-passage transition times $\{\mathbf{t}'_1, \mathbf{t}'_2, \ldots, \mathbf{t}'_N\}$ are then calculated. These two sets represent the stochastic expectations for the patients, and are used in the maximization step to find MLEs for $\mathbf{p}$ and $\boldsymbol{\beta}$.

Maximization Step

Given partial trajectories and expected sojourn times from the stochastic expectation step of the SEM algorithm, MLEs can be calculated for the parameters of the SMP. There are Φ sampled trajectory sets and expected time sets for each patient. Allowing once again for $i = z_{n,\phi,m-1}$, and $j = z_{n,\phi,m}$, as well as $\Delta' = t'_{n,m,\phi} - t'_{n,\phi,m-1}$, the likelihood function for this problem is then as follows:

$$\mathcal{L}(\mathbf{p}, \boldsymbol{\beta}) = \prod_{n=1}^{N} \prod_{\phi=1}^{\Phi} \prod_{m=1}^{M_n} \Pr\left(K = k_{n,\phi,m} | i', j\right) g_{i',j}^{(k_{n,\phi,m})}(\Delta') \left[1 - H_{j,j}\left(t_{n,m,\phi} - t'_{n,\phi,m}\right)\right]. \tag{2.5}$$

Maximizing the likelihood function provides parameters estimates for the next iteration of the SEM algorithm. The SEM iterations continue until the parameters $\boldsymbol{\theta}$ converge. We will use the existing approach's convergence criteria from Aralis and Brookmeyer (2019), which is that the following hold for three consecutive iterations:

$$\max_i \left(\frac{\left|\theta_i^{(r)} - \theta_i^{(r-1)}\right|}{\left|\theta_i^{(r-1)}\right| + \delta_1} \right) < \delta_2, \tag{2.6}$$

where δ_1 and δ_2 are convergence parameters and r is the iteration index.

In the original SEM procedure, sampling the trajectories is computationally expensive, while maximum likelihood estimation of parameters is simple. In the updated SEM method, the maximum likelihood estimation is more burdensome, because all parameters have to be estimated simultaneously, and the likelihood function evaluation can be computationally intense. The expectation step generates Φ sets of first-passage loops and times of transition ($\mathbf{k}_{n,\phi}$ and $\mathbf{t}'_{n,\phi}$) for each patient n. The maximization step then uses these values, and obtains parameters that maximize the likelihood, shown by Equation 2.5. Upon convergence, the latest maximum likelihood estimates of the algorithm give final parameter estimates for the SMP.

2.2 Computation via the FFT

The method described in this work incorporates both truncation and censoring in the sojourn time distributions. There is no general closed-form solution to the Laplace transform or moment generating function of most truncated/censored distributions. Thus, when we perform the convolutions necessary for many of the equations of Table 2.1, we numerically estimate the transforms.

One computationally efficient approach is to discretize the sojourn time distributions and then estimate their transforms using the Fast-Fourier Transform (FFT). The FFT is widely available as a fast implementation of the discrete Fourier transform on most hardware. Censoring and truncation in a discrete distribution are trivial. The FFTs of the transitions can replace the Laplace transforms of the definitions in Warr and Collins (2015). A benefit to this approach is the convenience of the inverse discrete Fourier transform in a computational setting, as opposed to the more expensive methods of inverting a Laplace transform. A downside to this approach is the discretization and truncation error of the FFT. If a researcher has a predetermined maximum error, it is possible to select a truncation point and discretization width such that the error will not exceed a predetermined bound. For details on how to calculate the error bounds involved in convolutions with the FFT, see Warr and Wight (2020). Because the FFT is fast, and the numerical error can be managed, it is a suitable approach in this application. The pieces of this method that are estimated via the FFT are the k-passage distributions $g_{i,j}^{(k)}(y)$ and the function $H_{j,j}(y)$, both described in Table 2.1.

CHAPTER 3

RESULTS

A simulation study is used to evaluate the accuracy and bias of the proposed improvements. The SMP involved in this simulation study resembles the SMP of the diabetic retinopathy application, mentioned in Chapter 1. The true parameters of this study are taken from a simple exploratory analysis of the diabetic retinopathy data. After the simulation study, the updated SEM method is applied to the original data.

3.1 Preliminary Simulation Study

The structure of the SMP involved in the simulation study is identical to that shown in Figure 1.1. The sojourn times are gamma distributed, with the exact parameters shown in Table 3.1. Transition data for 100 patients were simulated using this SMP. For each patient, a random draw of observation times is used to convert the true transition data into panel data. The observation times used in the simulation study are random draws from the panel observation times of the actual diabetic retinopathy patients. The parameter estimates from the updated SEM method are shown in Table 3.1 along with the true parameters used in the simulation study.

As shown in Table 3.1, these results are promising. Most parameters estimates are reasonable, and this is one simulation study on 100 patients. The estimate of the mean 1-to-2 sojourn is the least accurate of the means (estimated 11.82 from a true mean of 18.23). Both probability estimates are within 7% of the true parameters used in the simulation. Note that only a single replicate is obtained in this simulation study. Future work in this project will involve a comprehensive simulation study and analysis of bias. A more comprehensive simulation study could also experiment with the effect of panel observation times on the results.

Table 3.1: Results of a single simulation, where the true parameters are known. All sojourn-time distributions are assumed gamma distributed.

Parameter	True Value	SEM Estimate
p_{21}	.45	.52
p_{32}	.82	.81
μ_{12}	18.23	11.82
μ_{23}	14.76	14.51
μ_{34}	9.50	11.10
μ_{32}	7.22	9.77
μ_{21}	15.71	13.63
σ^2_{12}	58.87	21.29
σ^2_{23}	96.51	60.49
σ^2_{34}	84.50	97.12
σ^2_{32}	7.44	6.77
σ^2_{21}	38.37	25.46

For these results, the parameters of $\delta_1 = .001$ and $\delta_2 = .05$ are used (taken from previous work). Additionally, we set $\Phi \equiv 1$. Other applications may require Φ to increase with each iteration for convergence to be achieved. The previous approach, under the specified simulated conditions, has exaggerated sampling inefficiency that makes computation time unknown. One of the benefits of the proposed method is a resolution to this issue.

3.2 Diabetic Retinopathy Study

After a simple simulation study, the proposed method is applied to the diabetic retinopathy data. Many details on these data are included in Chapter 1. The data include information on patients such as age, gender, blood pressure at time of appointment, and a few other variables. A simple exploratory analysis of these data do not reveal any covariate to be particularly interesting for inclusion in a semi-Markov model. Figure 3.1 shows that there is no clear relationship between age, smoker status, or systolic blood pressure and the observed time between different states for the patients. These plots do not represent true transition times, but simply the naive panel observation time differences. In Marshall and Jones (1995), age and gender were found to be significant covariates for inclusion in a Markov model for these data, but these covariates are not included here.

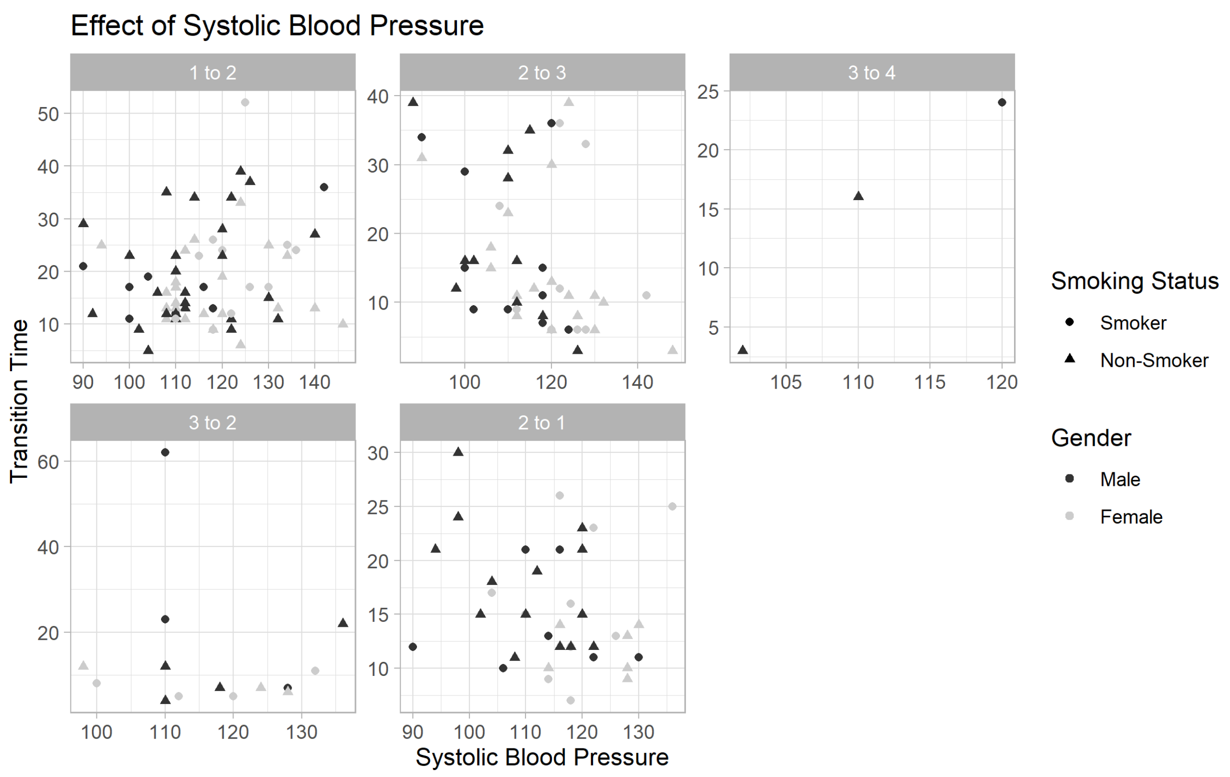

Figure 3.1: Times between diabetic retinopathy states observed at appointments. In many studies, these times would be used instead of utilizing all the information included in panel data. None of the three covariates included in these plots appear to be particularly interesting for estimating the model.

Using the methodology detailed in Section 2.1, the parameters of the SMP are estimated and shown in Table 3.2. The mean time for back-transitions (state 2 to state 1 or state 3 to state 2) was 8.43 years and 10.22 years. Retinopathy progression had mean times of 8.97 years for no retinopathy to microaneurysms, 11.48 years for microaneurysms to intermediate retinopathy, and 17.05 years for intermediate retinopathy to pre-proliferative or proliferative retinopathy.

With the SMP parameters estimated, many interesting research questions can be answered. This section will answer some of these questions; with these parameters, the interested reader can refer to Warr and Collins (2015) and solve specific quantities that may be of interest.

Table 3.2: Estimated parameters of the SMP for diabetic retinopathy progression shown in Figure 1.1. As in the simulation study, the sojourn times are assumed to be gamma distributed.

Parameter	SEM Estimate
p_{21}	.62
p_{32}	.85
μ_{12}	8.97
μ_{23}	11.48
μ_{34}	17.05
μ_{32}	8.43
μ_{21}	10.22
σ^2_{12}	16.84
σ^2_{23}	54.08
σ^2_{34}	83.17
σ^2_{32}	5.36
σ^2_{21}	19.90

The first result included is the CDF of the first-passage movement between states. This result shows how long it takes for a patient to move from no retinopathy to intermediate retinopathy, a microaneurysms patient to move to proliferative retinopathy, or any other possible indirect or direct route. This also includes the time of leaving the current state and arriving back in the same state (a microaneurysms patient transitioning back into microaneurysms, for example). These are shown in Figure 3.2. Notice that some are defective (such as the 3-to-1 first-passage, for example). In these cases, $G(\infty) < 1$. These paths are not guaranteed to occur, and $G(\infty)$ reveals the probability of the trajectory ever occurring. The mean times of these distributions are also shown in Figure 3.2, conditioned on that passage movement actually occurring.

Another possible research question may concern the time-dependent state probabilities. Time-dependent state probabilities represent the probability that a patient is in a particular state at time $t = t_1$, given some starting state at time $t = 0$. All of the time-dependent state probabilities for years 0–100 are shown in Figure 3.3. Note that the probabilities of being in state 4 are equal to the values shown in Figure 3.2; this is because

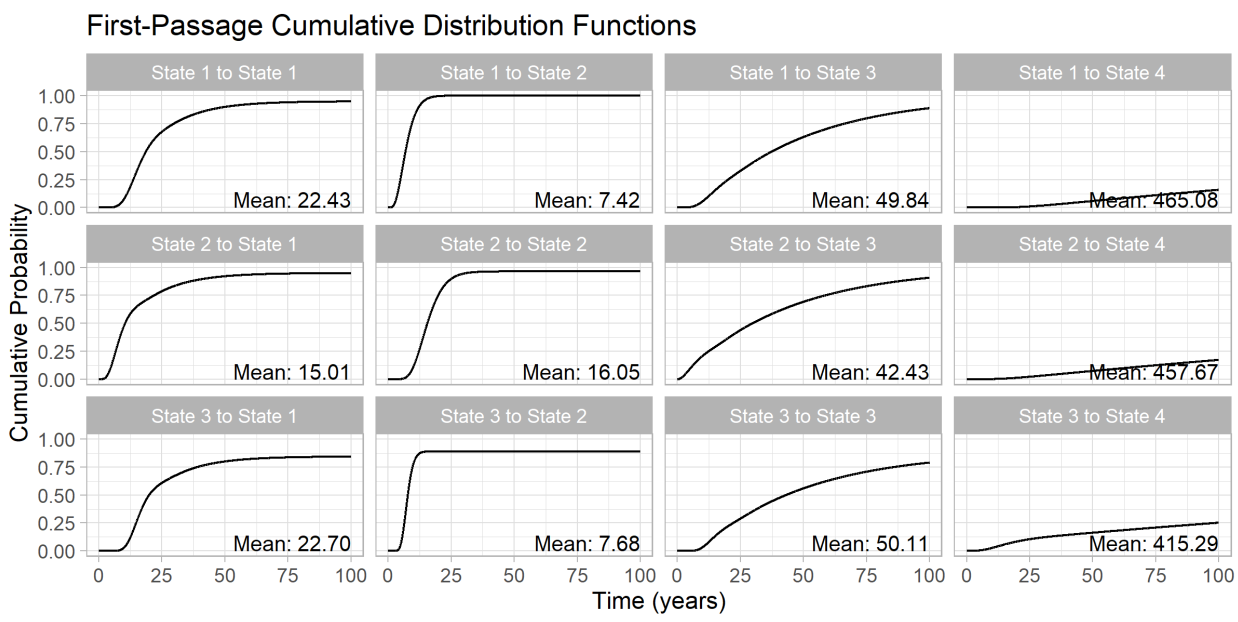

Figure 3.2: Estimated first-passage cumulative distribution functions between states. The mean times are conditional on the state being reached. It should be noted that this figure is extrapolating well beyond a reasonable range, because the data come from a relatively short study period. These estimates are assuming that the model results are infallible.

state 4 is absorbing. Time-dependent state probabilities can help researchers understand the risks of progression in diabetic retinopathy.

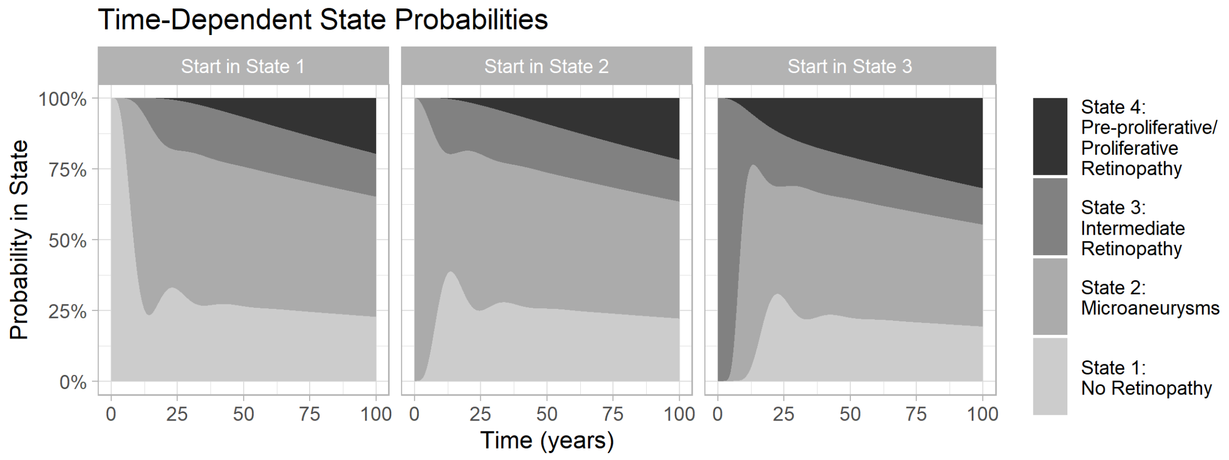

Figure 3.3: Estimated time-dependent state probabilities. These plots show the probability of being in the specific states $z(t) = j$, given that a patient just arrived at some state $z(0) = i$: $\Pr\left(z(t) = j | z(0) = i\right)$.

Other research questions can be easily answered, but a full analysis of the different possible features of diabetic retinopathy progression is beyond the scope of this project.

CHAPTER 4

DISCUSSION

Semi-Markov models are widely useful for processes involving state progression, especially in the medical field. In practice, panel data make the estimation of these models difficult. Many studies include limiting assumptions on sojourn time distributions, or assume that data is complete, when it is not. The state-of-the-art stochastic EM approach proposed in Aralis and Brookmeyer (2019) allows for parametric, unbiased estimation and fully flexible sojourn time distributions. The downside to this method is the difficulty in convergence due to the stochasticity and the infeasibility of the sampling procedure when the true parameters are unknown or patients have data from several appointments. When patients have data from many appointments, producing a valid sample trajectory is increasingly difficult. Additionally, if a patient has a strange set of panel observations (unexpected states or multiple consecutive appointments), the sampling procedure struggles to produce valid stochastic expectations. If the SEM algorithm begins with unrealistic parameter estimates, the time to complete the initial expectation varies greatly and does not have an upper bound. In our experience, this approach often leads to unending sampling attempts.

Our work proposes a method that relies on SMP properties to directly sample partial patient trajectories and calculate expected movement times. If a patient has many appointments or an unexpected trajectory, the computation load is the same for each observation, as opposed to the prior method. The proposed method is more robust to outlier patient trajectories, additional appointments, or varying starting parameter estimates.

Panel data complicates the estimation procedure for semi-Markov models. The existing solution to this problem is both innovative and flexible, but due to the inconsistency in computation time, we propose a more reliable approach. Even though the proposed approach has a more consistent computation time, the maximization step is still expensive. As it stands, each likelihood function evaluation is a costly calculation. Throughout the simulation study, the likelihood takes roughly 70 seconds to evaluate, on average, for 100 patients. Our SEM implementation includes parallel computing on 5 cores, each simultaneously evaluating the likelihood function when performing the maximization step. Under these conditions, the diabetic retinopathy model was estimated in 42 hours. Parallel computing could be expanded to many more cores. Even within a single likelihood evaluation, the calculation of the contributions of the patients could be put on separate jobs. The updated SEM algorithm still depends on choosing convergence parameters; changing these values could also affect the total computation time.

Using the proposed methodology, this project fits a semi-Markov model on Diabetic Retinopathy patient data. The parameters estimated by this approach enable the estimation of the first-passage distributions and time-dependent state probabilities. These interesting quantities are summarized in Figures 3.2 and 3.3. A variety of other quantities of interest are calculable from the parameter estimates. In Marshall and Jones (1995), the diabetic retinopathy data were originally modeled assuming exponentially distributed sojourn times. The proposed method allows for much more flexibility in the sojourn time distributions; this report analyzes the data assuming gamma-distributed sojourn times.

4.1 Future Work

There are many other steps that could be taken to improve this methodology. A next step will be a full simulation study that explores bias under many different conditions, as explained in Section 3.1. Other potential expansions to this simulation study might include more patients per data set, changes to the average length of the intervals between panel observations, or exploration of other sojourn time distributions.

If possible, a closed-form MLE for some of the parameters would help reduce the dimensionality of the likelihood function. Currently, the diabetic retinopathy SMP has 12 parameters. Numerical maximum likelihood estimation on such a high dimensional problem requires many evaluations of the likelihood function. This is the source of slow estimation for the proposed method.

Currently, a closed-form tractable likelihood function on panel data is not available. It may be possible to expand the likelihood function of Equation 2.5 to simultaneously include all possible trajectories. The proposed likelihood only has a single piece that is sampled, which is the number of end-state loops, k. If a new likelihood were obtained that included all possible k, the EM estimation technique may avoided entirely and parameters could be estimated through direct maximum likelihood.

BIBLIOGRAPHY

[1] Aralis, H., and Brookmeyer, R. (2019), "A stochastic estimation procedure for intermittently-observed semi-Markov multistate models with back transitions," *Statistical methods in medical research*, 28, 770–787.

[2] Asmussen, S., Nerman, O., and Olsson, M. (1996), "Fitting phase-type distributions via the EM algorithm," *Scandinavian Journal of Statistics*, 419–441.

[3] Chen, P.-L., and Tien, H.-C. (2004), "Semi-Markov models for multistate data analysis with periodic observations," *Communications in Statistics—Theory and Methods*.

[4] Ferguson, N., Datta, S., and Brock, G. (2012), "msSurv: An R package for nonparametric estimation of multistate models," *Journal of Statistical Software*, 50, 1–24.

[5] Foucher, Y., Giral, M., Soulillou, J., and Daures, J. (2010), "A flexible semi-Markov model for interval-censored data and goodness-of-fit testing," *Statistical Methods in Medical Research*, 19, 127–145.

[6] Foucher, Y., Giral, M., Soulillou, J.-P., and Daures, J.-P. (2007), "A semi-Markov model for multistate and interval-censored data with multiple terminal events. Application in renal transplantation," *Statistics in Medicine*, 26, 5381–5393.

[7] Gentleman, R., Lawless, J., Lindsey, J., and Yan, P. (1994), "Multi-state Markov models for analysing incomplete disease history data with illustrations for HIV disease," *Statistics in Medicine*, 13, 805–821.

[8] Jackson, C. H. (2011), "Multi-state models for panel data: the msm package for R," *Journal of Statistical Software*, 38, 1–29.

[9] Kalbfleisch, J., and Lawless, J. F. (1985), "The analysis of panel data under a Markov assumption," *Journal of the American Statistical Association*, 80, 863–871.

[10] Kang, M., and Lagakos, S. W. (2007), "Statistical methods for panel data from a semi-Markov process, with application to HPV," *Biostatistics*, 8, 252–264.

[11] Kryscio, R. J., and Abner, E. L. (2013), "Are Markov and semi-Markov models flexible enough for cognitive panel data?" *Journal of biometrics & biostatistics*, 4.

[12] Kulkarni, V. G. (2016), *Modeling and analysis of stochastic systems*, Crc Press/Taylor and Francis Group.

[13] Lange, J. M., and Minin, V. N. (2013), "Fitting and interpreting continuous-time latent Markov models for panel data," *Statistics in Medicine*, 32, 4581–4595.

[14] Lebreton, J.-D., and Cefe, R. P. (2002), "Multistate recapture models: modelling incomplete individual histories," *Journal of Applied Statistics*, 29, 353–369.

[15] Marshall, G., and Jones, R. H. (1995), "Multi-state models and diabetic retinopathy," *Statistics in medicine*, 14, 1975–1983.

[16] Mohammadi, K. A. (2020), "A Partial Likelihood Approach to Longitudinal Categorical Data Using a Continuous Time Semi-Markov Chain Model," *UT School of Public Health.*

[17] Redner, R. A., and Walker, H. F. (1984), "Mixture densities, maximum likelihood and the EM algorithm," *SIAM review*, 26, 195–239.

[18] Titman, A. C. (2014), "Estimating parametric semi-Markov models from panel data using phase-type approximations," *Statistics and Computing*, 24, 155–164.

[19] Warr, R. L., and Collins, D. H. (2015), "A comprehensive method for solving finite–state semi–Markov processes," *International Journal of Simulation and Process Modelling*, 10, 89–99.

[20] Warr, R. L., and Wight, C. J. (2020), "Error Bounds for Cumulative Distribution Functions of Convolutions via the Discrete Fourier Transform," *Methodology and Computing in Applied Probability*, 22, 881–904.

[21] Wei, S., and Kryscio, R. J. (2016), "Semi-Markov models for interval censored transient cognitive states with back transitions and a competing risk," *Statistical Methods in Medical Research*, 25, 2909–2924.

[22] Yau, C. L., and Huzurbazar, A. V. (2002), "Analysis of censored and incomplete survival data using flowgraph models," *Statistics in Medicine*, 21, 3727–3743.

APPENDICES

APPENDIX A

CODE

```
library(optimParallel)
library(tidyverse)

options(scipen = 8)

# Read in data (not publicly available)
diabetes <- read_tsv("Diabetic␣Ret␣Data.txt",
                     skip = 13,
                     col_types = cols(.default = "d"))

# Get ids of patients
patient_ids <- diabetes %>%
  pull(Subject) %>%
  unique()

# Convert data to list of panel data
panel_data <- lapply(patient_ids,
                     function(id) {
                       diabetes %>%
                         setNames(tolower(colnames(.))) %>%
                         filter(subject==id) %>%
                         select(state, time) %>%
                         as.matrix(ncol=2) %>%
                         t()})


# Set up support points
N <- 2^18
trunc <- 4000
supp <- seq(from = 0, to = trunc, length.out = N)

# Function for obtaining the truncated k-passage distributions
get_G <- function(i, j, time_no_move, k,
                  first_pass_ft = first_pass_ft,
                  f12=f12, f23=f23, f34=f34,
                  f32=f32,f21=f21,p21=p21,p32=p32,
                  pars = pars){

  # The paths to be truncated
  ghost_transitions <- paste0(i, which(pars$prob_mat[i,] != 0))

  # Distributions of the paths to be truncated
  dists_to_trunc <- lapply(ghost_transitions,
                           function(x) {
                             get(paste0("f", x))
                             })

  # Truncated distributions
  dists_trunc <- lapply(dists_to_trunc,
                        function(dist){
                          dist * (supp > time_no_move)
                          })
```

```
# If the total truncated probabilities, sum to <0, this is a fix
for(ghost_option in 1:length(dists_trunc)){
  if(sum(dists_trunc[[ghost_option]])<=0) {
    dists_trunc[[ghost_option]] <- rep(1,N)
  }
}

# Truncated distributions with normalization
fs_trunc <- lapply(dists_trunc,
                   function(dist_trunc) {
                     dist_trunc / sum(dist_trunc)
                    })

# FFT of truncated distributions
fts_trunc <- lapply(fs_trunc, function(f_trunc) fft(f_trunc))

# Probabilities of the truncated path(s)
prob_paths <- pars$prob_mat[i,which(pars$prob_mat[i,] != 0)]

# The path ends from the truncated distributions
after_ghosts <- as.numeric(sapply(ghost_transitions,
                                  function(path){substr(path,2,2)
                                    }))

ghost_paths_to_end <- list()

# We will incorporate the truncated transitions into the k-passage distribution
for(ghost_option in 1:length(fts_trunc)){

  # Transmittance for the truncated path
  ghost_transition <- prob_paths[ghost_option]*
    fts_trunc[[ghost_option]]

  # Ending state for the truncated path
  after_ghost <- after_ghosts[ghost_option]
  transition_to_end <- first_pass_ft[after_ghost,j,]


  if(i==j){
    if(k==0){
      # No movement
      ghost_path_to_end_ft <- fft(c(1,rep(0,N-1)))
    } else{
      # Movement, but back into starting state
      k <- k-1
      ghost_path_to_end_ft <- ghost_transition *
                                transition_to_end *
                                (first_pass_ft[j,j,]^k)
      k <- k+1
    }
  } else{
    if(after_ghost==j){
      # Truncated path ends with the same state as the ending state
      ghost_path_to_end_ft <- ghost_transition *
                                (first_pass_ft[j,j,]^k)
    } else{
      # Truncated path ends not with the same state as the
      #     ending or the starting state
      ghost_path_to_end_ft <- ghost_transition *
                                transition_to_end *
                                (first_pass_ft[j,j,]^k)
    }
  }
```

```
    # Converting back to time domain (from transform domain)
    ghost_path_to_end <- Re(fft(ghost_path_to_end_ft, inverse=TRUE))/N

    ghost_paths_to_end[[ghost_option]] <- ghost_path_to_end
  }

  # Weighting across the different truncation path options
  G <- ghost_paths_to_end %>%
    unlist() %>%
    matrix(ncol=length(ghost_transitions)) %>%
    rowSums()

  return(G)
}

# Function for obtaining the probability of leaving state j
get_H <- function(j, total_time,
                  first_pass_ft = first_pass_ft,
                  f12=f12, f23=f23, f34=f34,
                  f32=f32,f21=f21,p21=p21,p32=p32,
                  pars = pars){

  # The potential outward paths and associated probabilities from j
  out_paths <- which(pars$prob_mat[j,]!=0)
  prob_paths <- pars$prob_mat[j,out_paths]

  # The shapes and rates of the gamma-distributed paths
  shapes <- sapply(paste0("a",j,out_paths),
                   function(name){
                     pars[[name]]
                     })
  rates <- sapply(paste0("b",j,out_paths),
                  function(name) {
                    pars[[name]]
                    })

  # Probability of taking the different paths by time t
  all_PMFs <- sapply(1:length(shapes),
                     function(x) {
                       prob_paths[x]*
                        pgamma(total_time-supp[supp<total_time],
                               shapes[x],rates[x])
                     }) %>%
    matrix(ncol = length(out_paths))

  # Summing across all possible paths
  H <- rowSums(all_PMFs)
  return(H)
}

# Function for obtaining Pr(K=k)
get_probs <- function(i=1, j=2, time_no_move=10, total_time=50,
                      first_pass_ft = first_pass_ft,
                      f12=f12, f23=f23, f34=f34,
                      f32=f32,f21=f21,p21=p21,p32=p32, pars = pars){
  probs <- c()

  # Obtaining the H piece of the Pr(K=k) equation (not dependent on k)
  if(j!=4){
    H <- get_H(j, total_time, first_pass_ft = first_pass_ft,
               f12=f12, f23=f23, f34=f34,
               f32=f32,f21=f21,p21=p21,p32=p32,
               pars = pars)
  } else{
    prob_0 <- 1
    names(prob_0) <- 0
    return(prob_0)
  }

  k <- 0
  KEEP_GOING <- TRUE

  # We will continually calculate Pr(K=k, Z(t)'=j | Z(0)=i) until it
  # is sufficiently small
  while(KEEP_GOING){
    # The G portion of Pr(K=k) equation (dependent on k)
    G <- get_G(i, j, time_no_move, k, first_pass_ft = first_pass_ft,
               f12=f12, f23=f23, f34=f34,
               f32=f32,f21=f21,p21=p21,p32=p32,
               pars = pars)

    # The probs vector gets filled in until Pr(K=k, Z(t)'=j | Z(0)=i)
    # Is very small
    probs[k+1] <- sum(G[supp<total_time]*(1-H))
    KEEP_GOING <- probs[k+1]/max(probs) > 1/1000
    if(is.na(KEEP_GOING)){
      KEEP_GOING <- TRUE
    }
    k <- k+1
    # We will only get a maximum of 40 loops (very big for this case)
    if(k == 40) {
      KEEP_GOING <- FALSE
    }
  }

  probs <- pmax(0,probs)

  if(k>=40){
    probs <- rep(1,40)/40
  }

  # Rename and normalize
  names(probs) <- 0:(length(probs)-1)
  return(probs / sum(probs))
}


# Get the function for obtaining E(t|K=k), called get_t_prime()
get_t_prime <- function(i = 1, j = 2, num_loops=0, time_no_move = 10,
                        total_possible_time = 100,
                        first_pass_ft = first_pass_ft,
                        f12=f12, f23=f23, f34=f34,
                        f32=f32,f21=f21,p21=p21,p32=p32,
                        pars = pars){
  # H portion of E(t') equation
  if(j!=4){
    H <- get_H(j, total_possible_time,
               first_pass_ft = first_pass_ft,
               f12=f12, f23=f23, f34=f34,
               f32=f32,f21=f21,p21=p21,p32=p32,
               pars = pars)
  } else{
    H <- 0
  }

  # G portion of E(t') equation
  G <- get_G(i, j, time_no_move, num_loops,
             first_pass_ft = first_pass_ft,
             f12=f12, f23=f23, f34=f34,
             f32=f32,f21=f21,p21=p21,p32=p32,
             pars = pars)

  # Calculating the conditional mean
  cond_mean_numerator <- supp[supp<total_possible_time] *
                          G[supp<total_possible_time] *
                          (1-H)

  cond_mean_denomenerator <- sum(G[supp<total_possible_time] * (1-H))

  return(sum(cond_mean_numerator / cond_mean_denomenerator))
}


# The specific g(t') values for the likelihood function
get_g_val <- function(i = 1, j = 2, num_loops=0, time_no_move = 10,
                      total_time = 100, time = 50,
                      first_pass_ft = first_pass_ft,
                      f12=f12, f23=f23, f34=f34,
                      f32=f32,f21=f21,p21=p21,p32=p32,
                      pars = pars){

  index_val <- max(which(supp < time))
  G <- get_G(i, j, time_no_move, num_loops,
             first_pass_ft = first_pass_ft,
             f12=f12, f23=f23, f34=f34,
             f32=f32,f21=f21,p21=p21,p32=p32,
             pars = pars)[index_val]

  log(max(G, .Machine$double.eps))
}

# The specific H(t-t') values for the likelihood function
get_h_val <- function(i = 1, j = 2, num_loops=0, time = 50,
                      time_no_move = 10, total_time = 100,
                      first_pass_ft = first_pass_ft,
                      f12=f12, f23=f23, f34=f34,
                      f32=f32,f21=f21,p21=p21,p32=p32,
                      pars = pars){
  index_val <- max(which(supp < time))

  out_paths <- which(pars$prob_mat[j,]!=0)
  prob_paths <- pars$prob_mat[j,out_paths]

  shapes <- sapply(paste0("a",j,out_paths),
                   function(name){
                     pars[[name]]
                   })
  rates <- sapply(paste0("b",j,out_paths),
                  function(name) {
                    pars[[name]]
                  })

  if(j == 4){
    H <- 0
  } else{
    H <- sum(prob_paths * pgamma(total_time - time, shapes, rates))
  }

  log(max(1-H, .Machine$double.eps))
}
```

```
# Function that takes patient panel data and samples k and calculates the
    expectation of t'
get_single_expectation <- function(patient_data = panel_data[[8]],
                                   first_pass_ft = first_pass_ft,
                                   f12=f12, f23=f23, f34=f34,
                                   f32=f32,f21=f21,p21=p21,p32=p32,
                                   pars = pars){
  # We know z and t
  z <- patient_data[1,]
  t <- patient_data[2,]

  # We want to know t_prime and k
  t_prime <- rep(0,length(t))
  k <- t_prime

  # We will look at each panel observation pair
  for(m in seq_along(z)[-1]){
    # Initial setup for the given panel observation pair
    time_before_panel <- t[m-1] - t_prime[m-1]
    elapsed_time <- t[m] - t_prime[m-1]
    start_state <- z[m-1]
    end_state <- z[m]

    # If not in the absorbing state, find the probabilities for each k, and sample
        k
    if(end_state != 4){
      probs_of_k <- get_probs(i=start_state, j=end_state,
                              time_no_move=time_before_panel,
                              total_time=elapsed_time,
                              first_pass_ft = first_pass_ft,
                              f12=f12, f23=f23, f34=f34,
                              f32=f32,f21=f21,p21=p21,p32=p32,
                              pars = pars)
      k[m] <- sample(x=as.numeric(names(probs_of_k)),
                     size = 1, prob = probs_of_k)
    } else{
      k[m] <- 0
    }

    # Using k, get the expectation for t'
    if(k[m] == 0 & end_state==start_state){
      t_prime[m] <- t_prime[m-1]
    } else{
      t_prime[m] <- t_prime[m-1] +
                      get_t_prime(i=start_state, j=end_state,
                                  num_loops=k[m],
                                  time_no_move=time_before_panel,
                                  total_possible_time=elapsed_time,
                                  first_pass_ft = first_pass_ft,
                                  f12=f12, f23=f23, f34=f34,
                                  f32=f32,f21=f21,p21=p21,p32=p32,
                                  pars = pars)
    }
  }

  # Combine k and t'
  result <- rbind(state=z,panel_time=t,sampled_loops=k,
                  expected_time=t_prime)
  cols_to_remove <- which(diff(result[4,])==0) + 1

  if(ncol(result) %in% cols_to_remove) {
    # These have some movement if non-zero
    how_much_move <- abs(c(0,diff(result[1,]))) + result[3,]
    if(sum(how_much_move)>0){
      start_of_end <- ncol(result) - min(which(rev(how_much_move!=0))) + 1
    }else{
      start_of_end <- 1
    }

    end_end <- ncol(result)

    result <- cbind(result[,1:start_of_end], result[,end_end])

    cols_to_remove <- as.numeric()
  }

  # If there is no movement, we don't need it in the likelihood
  if(length(cols_to_remove)>0) {
    result <- result[,-cols_to_remove]
  }

  # Rename the expectation matrix
  result <- matrix(result, nrow = 4)
  rownames(result) <- c("state","panel_time","sampled_loops","expected_time")

  return(result)

}

# Negative log likelihood function: - sum( log(Pr(K=k)) + log(g) + log(1-H) )
neg_log_likelihood <- function(pars_vec, expectations){

  names(pars_vec) <- c('p21', 'p32',
                            'mean12', 'mean23',
                            'mean34', 'mean32',
                            'mean21', 'var12',
                            'var23', 'var34',
                            'var32','var21')

  # Convert incoming parameters into list form
  pars_used <- list(p21 = pars_vec['p21'],
                    p32 = pars_vec['p32'],
                    a12 = pars_vec['mean12']^2/pars_vec['var12'],
                    b12 = pars_vec['mean12']/pars_vec['var12'],
                    a23 = pars_vec['mean23']^2/pars_vec['var23'],
                    b23 = pars_vec['mean23']/pars_vec['var23'],
                    a34 = pars_vec['mean34']^2/pars_vec['var34'],
                    b34 = pars_vec['mean34']/pars_vec['var34'],
                    a32 = pars_vec['mean32']^2/pars_vec['var32'],
                    b32 = pars_vec['mean32']/pars_vec['var32'],
                    a21 = pars_vec['mean21']^2/pars_vec['var21'],
                    b21 = pars_vec['mean21']/pars_vec['var21'],
                    prob_mat = matrix(c(0,1,0,0,
                                        pars_vec[1],0,1-pars_vec[1],0,
                                        0,pars_vec[2],0,1-pars_vec[2],
                                        0,0,0,0), nrow = 4, byrow = TRUE))

  # PMFs of all possible transitions
  f12 <- c(0,diff(pgamma(supp, pars_used$a12, pars_used$b12)))
  f21 <- c(0,diff(pgamma(supp, pars_used$a21, pars_used$b21)))
  f23 <- c(0,diff(pgamma(supp, pars_used$a23, pars_used$b23)))
  f32 <- c(0,diff(pgamma(supp, pars_used$a32, pars_used$b32)))
  f34 <- c(0,diff(pgamma(supp, pars_used$a34, pars_used$b34)))

  # FFts of the PMFs
  ft12 <- fft(f12)
  ft21 <- fft(f21)
  ft23 <- fft(f23)
```

```
ft32 <- fft(f32)
ft34 <- fft(f34)

# First-passage distribution calculations (see Warr and Collins, 2018)
first_pass_ft_used <- array(0,c(4,4,N))
first_pass_ft_used[1,1,] <- (ft12*ft21*pars_used$p21) / (1 + ft23*ft32*pars_used
    $p21*pars_used$p32 - ft23*ft32*pars_used$p32)
first_pass_ft_used[1,2,] <- ft12
first_pass_ft_used[1,3,] <- (ft12*ft23*(1 - pars_used$p21))/(1 - ft12*ft21*pars_
    used$p21)
first_pass_ft_used[1,4,] <- (ft12*(ft23*ft34 - ft23*ft34*pars_used$p21 - ft23*
    ft34*pars_used$p32 + ft23*ft34*pars_used$p21*pars_used$p32))/(1 - ft12*ft21*
    pars_used$p21 - ft23*ft32*pars_used$p32 + ft23*ft32*pars_used$p21*pars_used$
    p32)
first_pass_ft_used[2,1,] <- (((ft21*pars_used$p21*(ft23*ft32*pars_used$p21*pars_
    used$p32 - ft23*ft32*pars_used$p32 + 1))/(ft12*ft21*pars_used$p21 + ft23*ft32
    *pars_used$p32 - ft23*ft32*pars_used$p21*pars_used$p32 - 1) - (ft21*ft23*ft32
    *pars_used$p21*pars_used$p32*(pars_used$p21 - 1))/(ft12*ft21*pars_used$p21 +
    ft23*ft32*pars_used$p32 - ft23*ft32*pars_used$p21*pars_used$p32 - 1))*(ft12*
    ft21*pars_used$p21 + ft23*ft32*pars_used$p32 - ft23*ft32*pars_used$p21*pars_
    used$p32 - 1))/(ft23*ft32*pars_used$p21*pars_used$p32 - ft23*ft32*pars_used$
    p32 + 1)
first_pass_ft_used[2,2,] <- ((ft12*ft21*pars_used$p21)/(ft12*ft21*pars_used$p21
    + ft23*ft32*pars_used$p32 - ft23*ft32*pars_used$p21*pars_used$p32 - 1) - (
    ft23*ft32*pars_used$p32*(pars_used$p21 - 1))/(ft12*ft21*pars_used$p21 + ft23*
    ft32*pars_used$p32 - ft23*ft32*pars_used$p21*pars_used$p32 - 1))*(ft12*ft21*
    pars_used$p21 + ft23*ft32*pars_used$p32 - ft23*ft32*pars_used$p21*pars_used$
    p32 - 1)
first_pass_ft_used[2,3,] <- (((ft21*pars_used$p21*(ft12*ft23 - ft12*ft23*pars_
    used$p21))/(ft12*ft21*pars_used$p21 + ft23*ft32*pars_used$p32 - ft23*ft32*
    pars_used$p21*pars_used$p32 - 1) + (ft23*(pars_used$p21 - 1)*(ft12*ft21*pars_
    used$p21 - 1))/(ft12*ft21*pars_used$p21 + ft23*ft32*pars_used$p32 - ft23*ft32
    *pars_used$p21*pars_used$p32 - 1))*(ft12*ft21*pars_used$p21 + ft23*ft32*pars_
    used$p32 - ft23*ft32*pars_used$p21*pars_used$p32 - 1))/(1 - ft12*ft21*pars_
    used$p21)
first_pass_ft_used[2,4,] <- (ft23*ft34*(pars_used$p21 - 1)*(pars_used$p32 - 1)*(
    ft12*ft21*pars_used$p21 - 1))/(ft12*ft21*pars_used$p21 + ft23*ft32*pars_used$
    p32 - ft23*ft32*pars_used$p21*pars_used$p32 - 1) - (ft21*pars_used$p21*(ft12*
    ft23*ft34 - ft12*ft23*ft34*pars_used$p21 - ft12*ft23*ft34*pars_used$p32 +
    ft12*ft23*ft34*pars_used$p21*pars_used$p32))/(ft12*ft21*pars_used$p21 + ft23*
    ft32*pars_used$p32 - ft23*ft32*pars_used$p21*pars_used$p32 - 1)
first_pass_ft_used[3,1,] <- (ft21*ft32*pars_used$p21*pars_used$p32)/(ft23*ft32*
    pars_used$p21*pars_used$p32 - ft23*ft32*pars_used$p32 + 1)
first_pass_ft_used[3,2,] <- ft32*pars_used$p32
first_pass_ft_used[3,3,] <- (ft23*ft32*pars_used$p32*(1 - pars_used$p21))/(1 -
    ft12*ft21*pars_used$p21)
first_pass_ft_used[3,4,] <- ft34*(pars_used$p32 - 1) - (ft32*pars_used$p32*(ft23
    *ft34 - ft23*ft34*pars_used$p21 - ft23*ft34*pars_used$p32 + ft23*ft34*pars_
    used$p21*pars_used$p32))/(1 - ft12*ft21*pars_used$p21 - ft23*ft32*pars_used$
    p32 + ft23*ft32*pars_used$p21*pars_used$p32)


# the v and g portions of the likelihood (after taking log)
k_like <- lapply(expectations, function(x) rep(NA,ncol(x)-1))
g_like <- lapply(expectations, function(x) rep(NA,ncol(x)-1))
h_like <- lapply(expectations, function(x) rep(NA,ncol(x)-1))

# For each expectation set
for(n_phi in 1:length(expectations)){
  # Extract useful info
  z <- expectations[[n_phi]][1,]
  t <- expectations[[n_phi]][2,]
  k <- expectations[[n_phi]][3,]
  t_prime <- expectations[[n_phi]][4,]
```

```
    # For each panel observation set (that has sampled movement)
    for(m in seq_along(z)[-1]){
      start_state <- z[m-1]
      end_state <- z[m]
      time <- t_prime[m]-t_prime[m-1]
      total_time <- t[m] - t_prime[m-1]
      time_no_move <- t[m-1] - t_prime[m-1]

      # probabilities for K
      k_probs <- get_probs(i=start_state, j=end_state, time_no_move=time_no_move,
          total_time=total_time, first_pass_ft = first_pass_ft_used,
                          f12=f12, f23=f23, f34=f34,
                          f32=f32,f21=f21,p21=p21,p32=p32, pars = pars_used)
      k_like[[n_phi]][m-1] <- log(max(k_probs[which(names(k_probs) == k[m])], .
          Machine$double.eps))

      if(start_state == end_state & k[m] == 0){
        g_like[[n_phi]][m-1] <- 0
        h_like[[n_phi]][m-1] <- 0
      } else{
        # G portions of the likelihood
        g_like[[n_phi]][m-1] <- get_g_val(i=start_state, j=end_state,
                                           num_loops=k[m], time_no_move = time_no_
                                               move,
                                           total_time = total_time, time = time,
                                           first_pass_ft = first_pass_ft_used,
                                           f12=f12, f23=f23, f34=f34,
                                           f32=f32,f21=f21,p21=p21,p32=p32, pars =
                                               pars_used)
        # H portions of the likelihood
        h_like[[n_phi]][m-1] <- get_h_val(i=start_state, j=end_state,
                                           num_loops=k[m], time_no_move = time_no_
                                               move,
                                           total_time = total_time, time = time,
                                           first_pass_ft = first_pass_ft_used,
                                           f12=f12, f23=f23, f34=f34,
                                           f32=f32,f21=f21,p21=p21,p32=p32, pars =
                                               pars_used)
      }


    }
  }

  # Combine the different elements of the likelihood
  out <- 0
  out <- out - sum(unlist(k_like))
  out <- out - sum(unlist(g_like))
  out <- out - sum(unlist(h_like))

  if(is.na(out) | abs(out) == Inf){
    out <- 10000000
  }
  return(out)
}

# Parallelization for the m step
cl <- makeCluster(6)     # set the number of processor cores
setDefaultCluster(cl=cl) # set 'cl' as default cluster

# Function that completes maximization step
get_new_parameters <- function(pars = EDA_pars,
                              expectations = lapply(panel_data[1:25],
                                                    get_single_expectation)){

  # Reparameterizing for means and variances
  means <- c(mean12 = pars$a12/pars$b12,
             mean23 = pars$a23/pars$b23,
             mean34 = pars$a34/pars$b34,
             mean32 = pars$a32/pars$b32,
             mean21 = pars$a21/pars$b21)
  variances <- c(var12 = pars$a12/pars$b12^2,
                 var23 = pars$a23/pars$b23^2,
                 var34 = pars$a34/pars$b34^2,
                 var32 = pars$a32/pars$b32^2,
                 var21 = pars$a21/pars$b21^2)

  vectorized_pars <- c(unlist(pars)[1:2], means, variances)

  names(vectorized_pars) <- c('p21', 'p32',
                              'mean12', 'mean23', 'mean34',
                              'mean32', 'mean21',
                              'var12', 'var23', 'var34',
                              'var32','var21')


  clusterExport(cl, c("neg_log_likelihood", "supp",
                      "N", "expectations", "get_probs",
                      "get_H", "get_G", "%>%", "get_g_val",
                      "num_evaluations", "get_h_val"))

  # Maximizing the likelihood to get the best parameters
  new_pars_vec <- optimParallel(vectorized_pars,
                                function(x) neg_log_likelihood(x, expectations),
                                lower = rep(.001,length(vectorized_pars)),
                                upper = c(rep(.999,2),rep(100,5),rep(200,5)),
                                control = list(factr=1e16, parscale=vectorized_
                                    pars),
                                method = "L-BFGS-B")$par

  # Reparameterizing for shapes and rates
  list(p21 = new_pars_vec['p21'],
       p32 = new_pars_vec['p32'],
       a12 = new_pars_vec['mean12']^2/new_pars_vec['var12'],
       b12 = new_pars_vec['mean12']/new_pars_vec['var12'],
       a23 = new_pars_vec['mean23']^2/new_pars_vec['var23'],
       b23 = new_pars_vec['mean23']/new_pars_vec['var23'],
       a34 = new_pars_vec['mean34']^2/new_pars_vec['var34'],
       b34 = new_pars_vec['mean34']/new_pars_vec['var34'],
       a32 = new_pars_vec['mean32']^2/new_pars_vec['var32'],
       b32 = new_pars_vec['mean32']/new_pars_vec['var32'],
       a21 = new_pars_vec['mean21']^2/new_pars_vec['var21'],
       b21 = new_pars_vec['mean21']/new_pars_vec['var21'],
       prob_mat = matrix(c(0,1,0,0,
                           new_pars_vec[1],0,1-new_pars_vec[1],0,
                           0,new_pars_vec[2],0,1-new_pars_vec[2],
                           0,0,0,0), nrow = 4, byrow = TRUE))
}

#start_pars <- EDA_pars
start_pars <- list(p21 = .5,
                   p32 = .8,
                   a12 = 5.5,
                   b12 = .5,
                   a21 = 6.5,
                   b21 = .5,
                   a23 = 2.5,
                   b23 = .2,
                   a32 = 7,
                   b32 = 1,
                   a34 = 1,
                   b34 = .1,
                   prob_mat = matrix(c(0,1,0,0,
                                       .5,0,.5,0,
                                       0,.5,0,.5,
                                       0,0,0,0), ncol = 4, byrow = TRUE))
pars <- start_pars
Phi <- 1
loop <- 1
delta_1 <- .001
delta_2 <- .05
overall_start <- proc.time()

all_par_ests <- c()

converged <- converged_1 <- converged_2 <- converged_3 <- FALSE


# SEM Algorithm
while(!converged){
  overall_loop_start <- proc.time()
  cat(paste(rep("-",10), collapse= ""))
  cat(sprintf("\nLoop␣%s\n",loop))
  cat(paste(rep("-",10), collapse= ""))
  cat("\nSetting␣up␣Parameters...")
  # PMFs of all possible transitions
  f12 <- c(0,diff(pgamma(supp, pars$a12, pars$b12)))
  f21 <- c(0,diff(pgamma(supp, pars$a21, pars$b21)))
  f23 <- c(0,diff(pgamma(supp, pars$a23, pars$b23)))
  f32 <- c(0,diff(pgamma(supp, pars$a32, pars$b32)))
  f34 <- c(0,diff(pgamma(supp, pars$a34, pars$b34)))

  # FFts of the PMFs
  ft12 <- fft(f12)
  ft21 <- fft(f21)
  ft23 <- fft(f23)
  ft32 <- fft(f32)
  ft34 <- fft(f34)

  # Put FFTs of PMFs in the matrix format
  dist_ft <- array(0,c(4,4,N))
  dist_ft[1,2,] <- ft12
  dist_ft[2,1,] <- ft21
  dist_ft[2,3,] <- ft23
  dist_ft[3,2,] <- ft32
  dist_ft[3,4,] <- ft34

  # First-passage distributions in matrix format
  first_pass_ft <- array(0,c(4,4,N))
  ID <- diag(4)

  for(i in 1:N){
    QMAT <- pars$prob_mat * dist_ft[,,i]
    first_pass_ft[,,i] <- QMAT %*% solve(ID-QMAT, solve(ID * solve(ID-QMAT)))
  }
  cat("␣Done.\n")

  # E Step
  expectations <- list()
  expectations_start <- proc.time()
```

```
cntr <- 1
cat("Performing␣E␣Step...")
#pb <- txtProgressBar(min = 0, max = length(panel_data)*Phi, style = 3)
for(i in 1:length(panel_data)){
  for(phi in 1:Phi){
    #setTxtProgressBar(pb, cntr)
    expectations[[cntr]] <- get_single_expectation(patient_data = panel_data[[i
        ]],
                                                   first_pass_ft = first_pass_ft
                                                       ,
                                                   f12=f12, f23=f23, f34=f34,
                                                   f32=f32,f21=f21,p21=p21,p32=
                                                       p32,
                                                   pars = pars)
    cntr <- cntr + 1
  }
}
cat(sprintf("␣Done.␣E␣Step␣took␣%3.4f␣minutes",
            (proc.time() - expectations_start)[3]/60))

# M step
maximization_step <- proc.time()
cat("\nPerforming␣M␣Step...")
prev_pars <- pars
num_evaluations <<- 0
pars <- get_new_parameters(prev_pars, expectations)
num_evaluations <- M_results$num_evals
cat(sprintf("␣Done.␣M␣Step␣took␣%3.4f␣minutes␣(with␣%s␣likelihood␣evaluations)\n
    ",
            (proc.time() - maximization_step)[3]/60, num_evaluations))
summary_pars <- c(p21 = pars$p21, p32 = pars$p32,
                  Mean_12 = pars$a12/pars$b12,
                  Mean_23 = pars$a23/pars$b23,
                  Mean_34 = pars$a34/pars$b34,
                  Mean_32 = pars$a32/pars$b32,
                  Mean_21 = pars$a21/pars$b21,
                  Var_12 = pars$a12/pars$b12^2,
                  Var_23 = pars$a23/pars$b23^2,
                  Var_34 = pars$a34/pars$b34^2,
                  Var_32 = pars$a32/pars$b32^2,
                  Var_21 = pars$a21/pars$b21^2)
names(summary_pars) <- c('p21', 'p32',
                         'mean12', 'mean23', 'mean34', 'mean32', 'mean21',
                         'var12', 'var23', 'var34', 'var32','var21')
cat(sprintf("Current␣NEGATIVE␣Log-Likelihood:␣%3.4f\n",
            neg_log_likelihood(summary_pars, expectations)))
#Phi <- Phi + 1
loop <- loop + 1
converged_3 <- converged_2
converged_2 <- converged_1
conv_level <- max(abs((unlist(pars)[1:12] - unlist(prev_pars)[1:12])/
                       (unlist(prev_pars)[1:12] + delta_1)))
converged_1 <- conv_level < delta_2
converged <- converged_1 * converged_2 * converged_3
cat(sprintf("Convergence␣level␣(<%3.4f␣for␣3␣iterations):␣%3.4f\n",
            delta_2, conv_level))
cat(sprintf("Current␣EM␣Loop␣time:␣%3.4f␣minutes\n",
            (proc.time() - overall_loop_start)[3]/60))
cat(sprintf("Total␣Time:␣%3.4f␣minutes\n",
            (proc.time() - overall_start)[3]/60))
cat("Parameter␣Update:\n")
print(summary_pars)
```

```
  all_par_ests <- rbind(all_par_ests, summary_pars)


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


# Displaying final output
cat(paste(rep("-",10), collapse= ""))
cat("\nInitial␣Parameter␣Estimates:␣\n")
summary_start_pars <- c(p21 = start_pars$p21, p32 = start_pars$p32,
                        Mean_12 = start_pars$a12/start_pars$b12,
                        Mean_23 = start_pars$a23/start_pars$b23,
                        Mean_34 = start_pars$a34/start_pars$b34,
                        Mean_32 = start_pars$a32/start_pars$b32,
                        Mean_21 = start_pars$a21/start_pars$b21,
                        Var_12 = start_pars$a12/start_pars$b12^2,
                        Var_23 = start_pars$a23/start_pars$b23^2,
                        Var_34 = start_pars$a34/start_pars$b34^2,
                        Var_32 = start_pars$a32/start_pars$b32^2,
                        Var_21 = start_pars$a21/start_pars$b21^2)
print(summary_start_pars)
cat("\n\nFinal␣Parameter␣Estimates:␣\n")
print(summary_pars)
cat("\n")
cat(paste(rep("-",10), collapse= ""))

all_par_ests <- rbind(summary_start_pars, all_par_ests)

save(all_par_ests, file = "results.Rdata")
```