SLCARE: An R Package for Semiparametric Latent Class Analysis of Recurrent Events

by Qi Yu and Limin Peng

Abstract Recurrent event data frequently arise in biomedical follow-up studies. The concept of latent classes enables researchers to characterize complex population heterogeneity in a plausible and parsimonious way. This article introduces the R package SLCARE, which implements a robust and flexible algorithm to carry out Zhao, Peng, and Hanfelt (2022)'s latent class analysis method for recurrent event data, where semiparametric multiplicative intensity modeling is adopted. SLCARE returns estimates for non-functional model parameters along with the associated variance estimates and p values. Visualization tools are provided to depict the estimated functional model parameters and related functional quantities of interest. SLCARE also delivers a model checking plot to help assess the adequacy of the fitted model.

1 Introduction

Recurrent event data frequently arise in biomedical follow-up studies where the event of interest, such as infection or hospitalization, occurs repeatedly over time. The event recurrence often demonstrates different patterns across individuals. To accommodate such heterogeneity, many regression methods have been developed with implementation available as R packages. To name a few, analyses based on the proportional intensity model (Andersen and Gill, 1982) can be carried out with function `coxph()` from R package **survival** (Therneau, 2023) or `cph()` from R package **rms** (Harrell Jr, 2023). Fitting Wang et al. (2001) 's method based on a multiplicative intensity model is implemented by function `reReg()` from R package **reReg** (Chiou et al., 2023). Modeling the gap time between recurrent events, Clement and Strawderman (2009) proposed conditional generalized estimating equation, and developed R package **condGEE** implementing this approach. Fine et al. (2004) 's temporal regression strategy can be applied to regress the mean function of recurrent events through using function `tpr()` from R package **tpr** (Yan and Fine, 2004).

More recently, viewing the observed data as a manifestation of latent classes or subgroups, Zhao et al. (2022) proposed a semiparametric latent class analysis (LCA) method to help reveal more realistic heterogeneity structure of recurrent event data that may not be adequately captured by a single regression model. Zhao et al. (2022) adopted latent variable mixture modeling (LVMM) while avoiding stringent parametric assumptions commonly imployed in LCA literature. Note that several R packages are available for fitting LVMM for data types other than recurrent event data. For example, R package **flexmix** (Grün and Leisch, 2008) delivers a general tool for tackling finite mixtures of regression models, such as the standard linear model and the generalized linear model, with the expectation-maximization (EM) algorithms. R package **lcmm** (Proust-Lima et al., 2017) allows for fitting joint mixture models with longitudinal and single time-to-event outcomes based on the Newton-Raphson algorithm. However, none of these packages offer options to conduct LCA oriented to recurrent event outcomes.

In this paper, we introduce the R package **SLCARE**, which delivers robust and flexible implementation of Zhao et al. (2022) 's LCA method for recurrent event data and is available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=SLCARE` and Github at `https://github.com/qyxxx/SLCARE`. The package **SLCARE** offers a user-friendly software for conducting LCA of recurrent event data in R with the core function `SLCARE()`. The function `SLCARE()` enables a variety of options for specifying and estimating the flexible semiparametric latent class model of Zhao et al. (2022) . For example, the number of latent classes can be pre-specified or chosen based on an entropy-based measure. Users can opt to built-in or customized initializer. Algorithm convergence criterion can be set through specifying the maximum number of iterations or the minimum change in parameter estimates. The function `SLCARE()` also provides visual tools to aid in result presentation and interpretation as well as model checking upon **ggplot2** plotting environment (Wickham, 2016).

The remainder of this paper is organized as follows. Next section describes the methodological background for the R package **SLCARE**, including model assumptions, estimation procedure and algorithm, inference, and model evaluation. In the third section, we elaborate the structure of the package. The utility of the package is illustrated via simulated data examples in Section 4. A real application is presented in Section 5. We conclude with a few remarks in the last section.

2 Methodological background

2.1 Recurrent event data and notation

For subject i, let $T_i^{(j)}$ denote time to the j^{th} recurrent event ($i = 1, \ldots, n$, $j = 1, 2, \ldots$). The underlying counting process of recurrent events is defined as $N_i^*(t) = \sum_{j=1}^{\infty} I(T_i^{(j)} \leq t)$,representing the number of events occurred before or at time t for subject i, where $I(\cdot)$ denotes the indicator function. Suppose the observation of subject i is ended at time C_i. Then the observed counting process is given by $N_i(t) = N_i^*(\min(t, C_i)) = \sum_{j=1}^{\infty} I(T_i^{(j)} \leq \min(t, C_i))$ ($i = 1, \ldots, n$). Let $\tilde{Z}_i$ be a p-dimensional vector of time-independent covariates for subject i. The observed data consist of $\{N_i(t), C_i, \tilde{Z}_i\}_{i=1}^n$.

2.2 Models and assumptions

Suppose the whole population consists of K latent classes, and assume that how covariates are related to the intensity function of recurrent events are the same within each latent class while being allowed to vary across different latent classes. Zhao et al. (2022) proposed a latent class multiplicative intensity model, which assumes that $N_i^*(t)$ is a nonstationary Poisson process with the intensity function,

$$\lambda_i(t) = \sum_{k=1}^{K} I(\xi_i = k) \times \lambda_0(t) \times W_i \times \eta_{0,k} \times \exp(\tilde{Z}_i^\top \tilde{\beta}_{0,k}) \tag{1}$$

where $I(\cdot)$ denotes the indicator function, ξ_i denotes the unobserved latent class membership, $\lambda_0(t)$ is an unspecified, continuous, nonnegative baseline intensity function shared by all latent classes, W_i is a positive subject-specific latent variable independent of $(\xi_i, \tilde{Z}_i, C_i)$. The latent variable W_i plays the same role as individual frailty reflecting more or less frequent occurrences of recurrent events. Here $\eta_{0,k}$ is a positive number that captures the class-k scale shift from the baseline intensity function, and $\tilde{\beta}_{0,k}$ is the regression coefficient that represents the class-k covariate effects on the intensity function. For the identifiability of $\lambda_0(t)$ and $\eta_{0,k}$, it is assumed that $E(W_i|\tilde{Z}_i, \xi_i = k) = 1$ for $k = 1, \ldots, K$ and $\int_0^{\nu^*} \lambda_0(u)du = 1$, where ν^* is a predetermined constant. As commented in Zhao et al. (2022), one may choose ν^* to be slightly smaller than the upper bound of C_i's support. A different choice of ν^* only results in a scale shift to $\lambda_0(t)$ and has no influence on the regression coefficients, $\tilde{\beta}_{0,k}$'s.

Zhao et al. (2022) modeled the distribution of the latent class membership ξ_i by a logistic regression model:

$$P(\xi_i = k|\tilde{Z}_i) = p_k(\alpha_0, \tilde{Z}_i) \doteq \frac{\exp(\tilde{Z}_i^\top \alpha_{0,k})}{\sum_{k=1}^{K} \exp(\tilde{Z}_i^\top \alpha_{0,k})}, \quad k = 1, \cdots, K \tag{2}$$

where $\alpha_0 = (\alpha_{0,1}^\top, \ldots, \alpha_{0,K}^\top)^\top$ with $\alpha_{0,1} = \mathbf{0}_{p\times 1}$.

2.3 Estimation and inference procedures

Estimation procedure

Let $Z_i = (1, \tilde{Z}_i^\top)^\top$ and $\beta_{0,k} = (\log \eta_{0,k}, \tilde{\beta}_{0,k}^\top)^\top$. By the model assumptions stated in Section 2, the following equalities hold:

$$E[I(\xi_i = k)Z_i\{\frac{N_i^*(C_i)}{\mu_0(C_i)} - \exp(Z_i^\top \beta_{0,k})\}] = 0, \;\; k = 1, \ldots, K, \tag{3}$$

$$E[\tilde{Z}_i\{I(\xi_i = k) - \frac{\exp(\tilde{Z}_i^\top \alpha_{0,k})}{\sum_{k=1}^{K} \exp(\tilde{Z}_i^\top \alpha_{0,k})}\}] = 0, \;\; k = 1, \ldots, K, \tag{4}$$

where $\mu_0(t) = \int_0^t \lambda_0(s)ds$, representing the cumulative baseline intensity function.

Equalities in (3) and (4) cannot be directly used to construct estimating equations for α_0 and $\beta_{0,k}$'s because ξ_i's are not observable and $\mu_0(\cdot)$ is unknown. Adapting the principle of conditional score (Stefanski and Carroll, 1987) for handling missing covariates, Zhao et al. (2022) proposed to address the unobserved $I(\xi_i = k)$ by substituting it with $\tau_{ik} \doteq E[I(\xi_i = k)|Z_i, C_i, D_i]$, where $D_i \doteq N_i(C_i)$, capturing the number of the observed recurrent events from subject i.

First, it follows from the definition that

$$\tau_{ik} = \frac{P(D_i = d_i|\xi_i = k, Z_i, C_i)P(\xi_i = k|Z_i, C_i)}{\sum_{l=1}^{K} P(D_i = d_i|\xi_i = l, Z_i, C_i)P(\xi_i = l|Z_i, C_i)}. \tag{5}$$

As implied by model (2) and the assumption that $C_i \perp (N_i^*(\cdot), \xi_i)|Z_i$,

$$P(\xi_i = k|Z_i, C_i) = p_k(\alpha_0, \tilde{Z}_i) = \frac{\exp(\tilde{Z}_i^\top \alpha_{0,k})}{\sum_{l=1}^{K} \exp(\tilde{Z}_i^\top \alpha_{0,l})} \tag{6}$$

and

$$\begin{aligned} &P(D_i = d_i|\xi_i = k, Z_i, C_i) \\ &= \int_0^\infty \frac{\{\exp(Z_i^\top \beta_{0,k})w \cdot \mu_0(C_i)\}^{d_i}}{d_i!} \exp\{-\exp(Z_i^\top \beta_{0,k})w \cdot \mu_0(C_i)\} \cdot f_W(w)dw, \end{aligned} \tag{7}$$

where $f_W(\cdot)$ denotes the density function of frailty W. Combining the results in (5)-(7), τ_{ik} can be explicitly expressed as a function of $\alpha_0 \doteq (\alpha_{0,1}^\top \ldots, \alpha_{0,K}^\top)^\top, \beta_0 \doteq (\beta_{0,1}^\top, \ldots, \beta_{0,k}^\top)^\top$ and $\mu_0(\cdot)$, which is denoted by $\tau_{ik}(\alpha_0, \beta_0, \mu_0)$.

In addition, Zhao et al. (2022) proposed a Nelson-Aalen type estimator of $\mu_0(\cdot)$, given by

$$\hat{\mu}(t) = \exp\{\hat{H}(t)\} \qquad \text{with} \quad \hat{H}(t) = -\int_t^{\nu^*} \frac{\sum_{i=1}^n dN_i(s)}{\sum_{i=1}^n I(C_i \geq s)N_i(s)}. \tag{8}$$

Replacing $I(\xi_i = k)$ and $\mu_0(\cdot)$ by $\tau_{ik}(\alpha, \beta, \hat{\mu})$ and $\hat{\mu}(\cdot)$ respectively in the empirical counterparts of (3)-(4) then leads to the following estimating equations:

$$n^{1/2} S_{1,n}(\alpha, \beta, \hat{\mu}) = 0, \tag{9}$$

$$n^{1/2} S_{2,n}(\alpha, \beta, \hat{\mu}) = 0, \tag{10}$$

where $S_{j,n}(\alpha, \beta, \hat{\mu}) = (S_{j,n,1}(\alpha, \beta, \hat{\mu})^\top, \cdots, S_{j,n,K}(\alpha, \beta, \hat{\mu})^\top)^\top$ $(j = 1, 2)$ with

$$S_{1,n,k}(\alpha, \beta, \mu) = \frac{1}{n}\sum_{i=1}^{n} \tau_{ik}(\alpha, \beta, \mu)Z_i\{\frac{N_i^*(C_i)}{\hat{\mu}(C_i)} - \exp(Z_i^\top \beta_k)\}, \quad k = 1, \ldots, K; \tag{11}$$

$$S_{2,n,k}(\alpha, \beta, \mu) = \frac{1}{n}\sum_{i=1}^{n} \tilde{Z}_i\{\tau_{ik}(\alpha, \beta, \mu) - \frac{\exp(\tilde{Z}_i^\top \alpha_k)}{\sum_{j=1}^{K} \exp(\tilde{Z}_i^\top \alpha_j)}\}, \quad k = 1, \ldots, K. \tag{12}$$

Solving equations (9) and (10) renders estimators of α_0 and β_0, denoted by $\hat{\alpha}$ and $\hat{\beta}$. Detailed derivations of estimating equations as well as the asymptotic properties of $\hat{\alpha}$ and $\hat{\beta}$ can be found in Zhao et al. (2022) .

Estimation algorithm

Based on the result in (8) and estimating equations in (9), and (10), we propose the following algorithm to obtain $\hat{\alpha}$ and $\hat{\beta}$, the parameter estimates for models (1) and (2).

Step 1: Compute $\hat{\mu}(\cdot)$ based on formula (8). Let $L_{iter} = 0$ and set initial values of α_0 and β_0 as $\hat{\alpha}^*$ and $\hat{\beta}^*$ respectively. Calculate $\hat{\tau}_{ik}^* \doteq \tau_{ik}(\hat{\alpha}^*, \hat{\beta}^*, \hat{\mu})$ based on equations (5), (6) and (7).

Step2: Repeat in the loop:

(i) Solve estimating equations (9) and (10) with $\tau_{ik}(\alpha, \beta, \hat{\mu})$ fixed as $\hat{\tau}_{ik}^*$. Denote the solutions by $\check{\alpha}$ and $\check{\beta}$.

(ii) $\hat{\alpha}^* := \check{\alpha}$; $\hat{\beta}^* := \check{\beta}$; $\hat{\tau}_{ik}^* := \tau_{ik}(\hat{\alpha}^*, \hat{\beta}^*, \hat{\mu})$. Increase L_{iter} by 1.

(iii) If $\max(\|\frac{\check{\beta}-\hat{\beta}^*}{\hat{\beta}^*}\|_\infty, \|\frac{\check{\alpha}-\hat{\alpha}^*}{\hat{\alpha}^*}\|_\infty)$ is smaller than a pre-specified threshold value or L_{iter} is greater than a pre-specified maximum number of iteration, then exit the loop. Here $\|\cdot\|_\infty$ denotes the L_∞ norm and the fraction between vectors α and β is component-wise fraction.

End repeat loop.

Return $\check{\alpha}$ and $\check{\beta}$ as the final estimates for α_0,and β_0 (i.e., $\hat{\alpha}$ and $\hat{\beta}$).

Algorithm implementation

In the following, we present some details related to the algorithm implementation, such as how to set initial values, $\hat{\alpha}^*$ and $\hat{\beta}^*$, and how to solve the estimating equations involved in the presented algorithm.

Setting initial values for the estimation algorithm The function `SLCARE()` in package **SLCARE** allows users to specify the initial values, $\hat{\alpha}^*$ and $\hat{\beta}^*$, by their own choice. Unless users manually specify the initial values, `SLCARE()` implements an automated initializer, which obtains $\hat{\alpha}^*$ and $\hat{\beta}^*$ through the following procedure:

> Step 1: Perform K-means clustering (Lloyd, 1982) of the observed covariates and number of recurrent events, i.e., $\{\tilde{Z}_i, N_i(C_i)\}_{i=1}^n$, with R function `kmeans()`, and divide all subjects into K group, where K stands for the number of latent classes. Denote the assigned group membership for subject i by $G_i (\in \{1, \ldots, K\})$.
> Step 2: Obtain $\hat{\alpha}^*$, the initial estimate for α_0, as the regression coefficients from fitting the multinomial regression of $\{G_i\}_{i=1}^n$ on covariates $\{\tilde{Z}_i\}_{i=1}^n$.
> Step 3: Obtain $\hat{\beta}^*$, the initial estimate of β_0, as $(\hat{\beta}_1^{*\top}, \ldots, \hat{\beta}_K^{*\top})$, where $\hat{\beta}_k^{*\top}$ is the regression coefficient estimate from fitting Wang et al. (2001) 's multiplicative intensity model to the subset with $G_i = k$ using `reReg()` from R package **reReg** (Chiou et al., 2023).

Specifying the distribution of frailty W The `SLCARE()` function allows users to specify the distribution of frailty W. While Zhao et al. (2022) 's method allows the frailty distribution to be a general parametric distribution, the implementation by `SLCARE()` confines to settings where $W = 1$ or W follows a distribution that is parameterized as $Gamma(k, k)$. These choices of frailty distributions cover a variety of density forms.

Solving estimating equations In Step 2(i) of the presented algorithm, an updated estimate for β_0, $\check{\beta}$, is obtained from solving equations, $\sum_{i=1}^n \hat{\tau}_{ik} \cdot Z_i \cdot \left\{ \frac{N_i^*(C_i)}{\hat{\mu}(C_i)} - \exp(Z_i^\top \beta_k) \right\} = 0, \quad k = 1, \ldots, K$. As suggested by Zhao et al. (2022), we solve these equations by fitting a 'pseudo' weighted Poisson regression model to an augmented dataset which includes responses, $\{\frac{N_i^*(C_i)}{\hat{\mu}(C_i)}\}_{i=1}^n$, and covariates, $\{Z_i\}_{i=1}^n$, along with weights $\hat{\tau}_{ik}$. **SLCARE** performs such Poisson regression with R function `glm.fit()` from R package **stats**.

Meanwhile, in Step 2(i) of the presented algorithm, an updated estimate for α_0, $\check{\alpha}$, is obtained by solving equations, $\frac{1}{\sqrt{n}} \sum_{i=1}^n \tilde{Z}_i \left\{ \hat{\tau}_{ik} - \frac{\exp(\tilde{Z}_i^\top \alpha_k)}{\sum_{j=1}^K \exp(\tilde{Z}_i^\top \alpha_j)} \right\} = 0, \quad k = 2, \cdots, K$. To solve these equations, we use an alternative and yet equivalent approach that fits weighted multinomial regression model on an augmented dataset, which consists of nK response-covariate pairs, $\{(1, \tilde{Z}_i), \ldots, (K, \tilde{Z}_i)\}_{i=1}^n$ with weight $\hat{\tau}_{ik}$ assigned to the pair $(k, \tilde{Z}_i)$ $(k = 1, \ldots, K)$. **SLCARE** implements the weighted multinomial regression with function `multinom()` from R package **nnet** (Venables and Ripley, 2002).

Determination of the number of latent classes Zhao et al. (2022) proposed to determine K, the number of latent classes, based on a relative entropy measure (Ramaswamy et al., 1993). That is, upon fitting models (1) and (2) that assume M latent classes, the corresponding relative entropy measure is defined as

$$Entropy(M) = 1 - \frac{\sum_{i=1}^n \sum_{k=1}^M -\hat{\tau}_{ik} \log(\hat{\tau}_{ik})}{n \log(M)} \tag{13}$$

where $\hat{\tau}_{ik} = \tau_{ik}(\hat{\alpha}, \hat{\beta}, \hat{\mu})$ $(k = 1, \ldots, M)$. By definition, $Entropy(M)$ is bounded between 0 and 1 with a larger value indicating a better fit to the data (Celeux and Soromenho, 1996). Following the rationale, one may choose K as the maximizer of $Entropy(\cdot)$ over a sequence of candidate values for K. Given a pre-specified K, the value of $Entropy(K)$ can be generated by S3 method `print(x, type = "Entropy")`.

Model checking Zhao et al. (2022) proposed a graphical method for checking the overall fit of models (1) and (2). The key idea is to compare the observed recurrent events $D_i \doteq N_i(C_i)$ versus the expected number of recurrent events under models (1)-(2). Specifically, models (1) and (2) imply $E\{N_i(C_i)|Z_i\} = E\{N_i^*(C_i)|Z_i\} = \sum_{k=1}^K \tau_{ik} \cdot \mu_0(C_i) \exp(Z_i^\top \beta_{0,k})$. Then the expected number of recurrent events under the assumed models can be approximated by

$$\hat{D}_i \doteq \sum_{i=1}^K \hat{\tau}_{ik} \cdot \hat{\mu}(C_i) \cdot \exp(Z_i^\top \hat{\beta}_k). \tag{14}$$

Therefore, in the scatter plot of $\hat{D}_i$ versus D_i, a major departure from the identity line $D_i = \hat{D}_i$ may suggest a lack-of-fit of the assumed models. Model checking plot can be generated by S3 method `plot(x, type = "ModelChecking")`.

Class-specific mean function of recurrent events To help illustrate heterogeneity in recurrent event occurrence across different latent classes, **SLCARE** computes crude estimates for the class-specific mean functions of recurrent events. Specifically, the crude estimate for the class-specific mean function of recurrent event is computed as

$$\frac{\sum_{i=1}^{n} I(\hat{\xi}_i = k) \sum_{j=1}^{K} \hat{\tau}_{ij} \cdot \hat{\mu}(t) \exp(Z_i^\top \hat{\beta}_j)}{\sum_{i=1}^{n} I(\hat{\xi}_i = k)}, \quad k = 1, \ldots, K, \tag{15}$$

where $\hat{\xi}_i = \arg\max_{1 \le k \le K} \hat{\tau}_{ik}$. A plot of the estimated mean functions stratified by latent groups can be generated by S3 method `plot(x, type = "EstMeans")`.

3 Package structure

The main function of R package **SLCARE** is `SLCARE()`. The dataset imported to function `SLCARE()` should take the long format, where each row corresponds to one time point of a subject at which a recurrent event is observed or censored. With a dataset coded in the wide format, where each row contains the timing information of all recurrent events observed or censored within one subject, users may convert such a dataset into the required long format by using function `pivot_longer()` from package **tidyr** (Wickham et al., 2023) or function `melt()` from package **reshape2** (Wickham, 2007). In addition to allowing users to specify initial parameter estimates manually, `SLCARE()` offers a default initializer that implements the informative selection of initial values described in Section **Estimation procedure**. The built-in initializer performs K-means clustering of the observed data using function `kmeans()` from package **stats** and fitting multiplicative intensity models using `reReg()` from package **reReg**. When solving the estimating equations involved in the iterative estimation algorithm, `SLCARE()` performs weighted Poisson regression using `glm.fit()` from package **stats** and weighted multinomial regression using `multinom()` from package **nnet**. After running `SLCARE()`, model fitting results can be easily extracted from the output with S3 methods `summary()`, `print()`, `predict()`, and `plot()`. Graphical results generated by `SLCARE()` are presented via **ggplot2** environment and are fully customizable.

Table 1 lists the main function `SLCARE()`, corresponding S3 methods and other functions called by this function, along with brief descriptions of their purposes.

Table 1: An overview of the main function and imported functions in SLCARE

FUNCTION	PURPOSE
SLCARE()	Conduct LCA with recurrent events data based on semiparametric multiplicative intensity modeling.
summary()	Generic function; used to summarize estimates for model parameters along with the associated variance estimates and p values.
print()	Generic function; used to initial estimates for the estimation algorithm convergence criterion, latent class membership probability and predicted number of recurrent events.
predict()	Generic function; used to predict the posterior number of recurrent events.
plot()	Generic function; used to generate cumulative baseline intensity function estimated mean function, model checking plot.
reReg()	Imported from package reReg; used to find an initial estimate for β_0.
multinom()	Imported from package nnet; used to find an initial estimate for α_0; used to solve the estimating equation to update the estimate for α_0.
kmeans()	Imported from package stats; used to find initial estimates for α_0 and β_0.
glm.fit()	Imported from package stats; used to solve the estimating equation to update the estimate for β_0.

4 Illustrations

The package **SLCARE** is designed to conduct LCA of recurrent events data based on semiparametric multiplicative intensity modeling (Zhao et al., 2022). The main function `SLCARE()` has function arguments that enable flexible model specification and method implementation. In the below, we illustrate the use of `SLCARE()` through simulated datasets.

4.1 Input dataset

An input dataset for `SLCARE()` is a data frame containing time to recurrent events or censoring along with time-independent covariates of interest. The input dataset should take the long format, where each row contains the information in one time interval in which a recurrent event is observed or censored. As such, multiple rows in the dataset may correspond to one subject. As mentioned in Section **Package structure**, a dataset in the wide format data can be readily converted into the long format required by `SLCARE()` with the use of function `pivot_longer()` from package **tidyr** or function `melt()` from package **reshape2**.

The package **SLCARE** has a build-in dataset, `SimData`, which is a simulated dataset perturbed from a real dataset. `SimData` contains 48 subjects. For each subject, `SimData` contains values for the following six variables: `id`, which is the unique identifier for each subject; `start`, which records the starting time in minutes of the counting process interval; `stop`, which records the ending time in minutes of the counting process interval. If the subject entered the study at time zero, this column represents the time in minutes from the baseline visit to the occurrence of event or censoring; `event`, which indicates whether a recurrent event is observed (`event = 1`) or not (`event = 0`) at the end of the counting process interval; `x1`, which is a binary covariate; `x2`, which is a continuous covariate. Below we display how the data from one example subject are recorded in `SimData`:

```
#> # A tibble: 3 x 6
#> # Groups:   id [1]
#>   id    start  stop event    x1    x2
#>   <chr> <dbl> <int> <int> <int> <dbl>
#> 1 G052      0  1950     1     1 0.444
#> 2 G052   1950  2580     1     1 0.444
#> 3 G052   2580  7085     0     1 0.444
```

As shown above, subject `G052` experienced two current events at time 1950 and time 2580 before the censoring time 7085. The time-independent covariates, `x1` and `x2`, remain unchanged across different time points within the same subject.

4.2 Command line and function arguments

The LCA of recurrent event data outlined in Section **Methodological background** can be carried out with a single command line `SLCARE()`. `SLCARE()` provides flexible function arguments, which are shown below:

```
#> function (formula = "x1 + x2", alpha = NULL, beta = NULL, data = data,
#>     id_col = "id", start_col = "start", stop_col = "stop", event_col = "event",
#>     K = NULL, gamma = 0, max_epochs = 500, conv_threshold = 0.01,
#>     boot = NULL)
#> NULL
```

At minimum, `SLCARE()` requires the following three arguments:

`data`: a dataframe with the format similar to `SimData`.

`K`: pre-determined number of latent classes.

`formula`: a string containing the covariates of interest in `data`.

The optional arguments of `SLCARE()` are:

`alpha`: initial estimate for α_0 in the estimation procedure. The default is NULL, which represents the initial estimate for α_0 resulted from the automated initializer described in Section **Estimation and inference procedures**.

`beta`: initial estimate for β_0 in the estimation procedure. The default is NULL, which represents the initial estimate for α_0 resulted from the automated initializer described in Section **Estimation and inference procedures**.

`id_col`: name of the column that includes subject identifiers.

`start_col`: name of the column that records the start time of each at-risk time interval.

`stop_col`: name of the column that records the start time of each at-risk time interval.

`event_col`: name of the column that indicates whether a recurrent event is observed or not (i.e, 1=observed; 0=otherwise).

`gamma`: parameter that indicates the distribution of frailty W. The default is 0 which indicates model (1) holds without the subject-specific frailty (i.e., $W = 1$); `gamma` $= k$ indicates that W follows the $Gamma(k, k)$ distribution.

`max_epochs`: maximum number of iterations for the estimation algorithm.

`conv_threshold`: convergence threshold for the estimation algorithm.

`boot`: number of bootstrap replicates used to obtain the standard error estimation. The default is NULL which indicates bootstrap is not conducted.

4.3 Output and illustration with a sample dataset

The following code is used to perform LCA of the recurrent event dataset `SimData` with two latent classes (i.e., $K = 2$), 20 bootstrap replicates, and covariates `x1` and `x2`.

```
set.seed(0)
model1 <- SLCARE(formula = "x1 + x2", data = SimData,
                 id_col = "id", start_col = "start", stop_col = "stop",
                 event_col = "event",  K=2, boot = 20)
```

The default output/print of `SLCARE()` includes parameter estimates and the associated variance estimates and p-values, along with the model's relative entropy measure. The same results can also be obtained using the generic method `summary(model1, digits = 3)`, which allows for manually setting the minimum number of significant digits.

```
model1
```

```
#> Call:
#> SLCARE(formula = "x1 + x2", data = SimData, id_col = "id", start_col = "start",
#>     stop_col = "stop", event_col = "event", K = 2, boot = 20)
#>
#> Coefficients for Beta:
#>        (intercept)      x1     x2
#> class1        3.21 -0.2350 -4.933
#> class2        2.41 -0.0413  0.309
#>
#> Std. Errors for Beta:
#>        (intercept)    x1    x2
#> class1       0.216 0.278 0.813
#> class2       0.142 0.105 0.352
#>
#> P. Values for Beta:
#>        (intercept)    x1       x2
#> class1    4.81e-50 0.398 1.28e-09
#> class2    1.85e-64 0.693 3.80e-01
#>
#> Coefficients for Alpha:
#>           x1   x2
#> class1  0.00 0.00
#> class2 -0.71 4.81
#>
#> Std. Errors for Alpha:
#>           x1   x2
#> class1 0.000 0.00
#> class2 0.928 1.88
#>
```

```
#> P. Values for Alpha:
#>           x1     x2
#> class1   NaN    NaN
#> class2 0.444 0.0106
#>
#> Relative Entropy: 0.539
```

Recall that in model (2), $\alpha_{0,1} = \mathbf{0}$, indicating that `class1` is used as the reference group for evaluating the odds ratio of being in another latent class. The numbers in the row corresponding to `class2` represent covariate effects on the log odds ratio of belonging to `class2` versus `class1`.

These results are also can be extracted separately with the generic method function `print(model1, type = "Est")`, `print(model1, type = "SE")`, `print(model1, type = "PValue")`, and `print(model1, type = "Entropy")`.

Initial estimates for the estimation algorithm

Unless manually specifying the arguments `alpha` and `beta`, `SLCARE()` implements the automated initializer described in Section **Estimation and inference procedures**. The resulting initial estimates for β_0 and α_0 can be found through the following generic method function `print()`:

```
print(model1, type = "Init")
```

```
#> $beta
#>        (intercept)          x1          x2
#> class1     2.01630  0.09073593 -0.45944133
#> class2     2.56397 -0.24415627  0.06202144
#>
#> $alpha
#>               x1       x2
#> class1 0.0000000 0.000000
#> class2 0.5386871 2.985595
```

Convergence criterion

`SLCARE()` provides arguments to control the termination of the iterative estimation procedure. By default, `max\_epochs=500` and `conv_threshold=0.01`. This means that the iterations would stop when the number of iterations reaches 500 or the magnitude of the change in parameter estimates is below 0.01. In addition, users are allowed to check the change in parameter estimates in the last iteration using the following code:

```
print(model1, type = "ConvergeLoss")
```

```
#> [1] 0.00909132
```

If the output is greater than the pre-determined convergence threshold, users may consider increasing the maximum number of iterations and/or setting a larger convergence threshold.

Latent class membership probability

`SLCARE()` generates output on the estimated probability that a subject belong to different latent classes. The following are the example code and output in the case where one is interested in the class membership probabilities of the 8^{th} to 10^{th} subjects in `SimData`.

```
print(model1, type = "ClassProb")[8:10,]
```

```
#> # A tibble: 3 x 3
#> # Groups:   ID [3]
#>   ID       class1  class2
#>   <chr>     <dbl>   <dbl>
#> 1 G052   9.94e- 1 0.00610
#> 2 UOM043 5.33e- 1 0.467
#> 3 G064   4.20e-22 1
```

Predicted number of recurrent events

`SLCARE()` computes the predicted number of recurrent events for each subject according to equation (14). The code to extract such a result for the 8^{th} to 10^{th} subjects in `SimData` is shown below, along with the corresponding output:

```
predict(model1)[8:10,]
```

```
#>        ID PosteriorPrediction
#> 8    G052            2.231237
#> 9  UOM043            2.892228
#> 10   G064           13.333784
```

Note that `SLCARE()` returns the estimated numbers of recurrent events as float rather than integer. Users may use the argument `integer = TRUE` in the genetic function `preduct()` to round those numbers to integers.

```
predict(model1, integer = TRUE)[8:10,]
```

```
#>        ID PosteriorPrediction
#> 8    G052                  2
#> 9  UOM043                  2
#> 10   G064                 13
```

Model checking plot

`SLCARE()` generates a model checking plot following the procedure described in Section **Methodological background** via **ggplot2** environment. The following shows the `S3` method for plotting a model checking plot:

```
plot(model1, type = "ModelChecking")
```

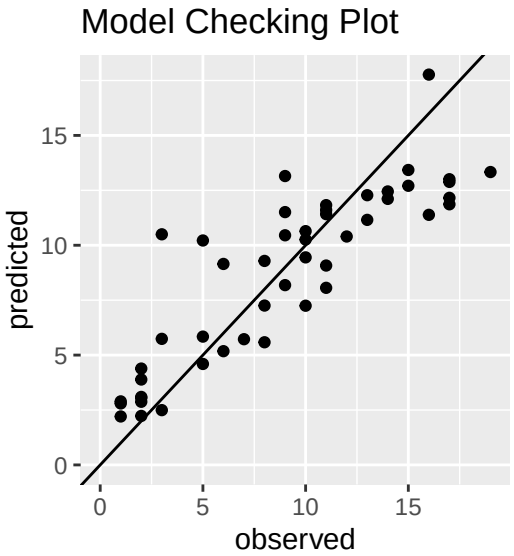

Figure 1: Predicted numbers of events versus the observe numbers of events.

Estimated cumulative baseline intensity function

By equation (8), `SLCARE()` calculates $\hat{\mu}(t)$, which is an estimates for $\mu_0(t)$, cumulative baseline intensity function. A plot of $\hat{\mu}(t)$ versus t can be generated with the following code:

```
plot(model1, type = "mu0")
```

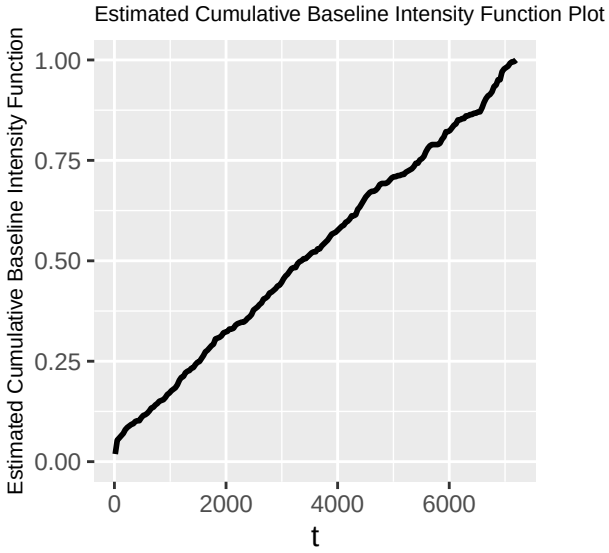

Figure 2: Estimated cumulative baseline intensity function.

Evaluation of $\hat{\mu}(t)$ at specific time points, for example, $t = 100, 1000$ and 5000, can be obtained by the following command:

```
model1$est_mu0(c(100, 1000, 5000))

#> [1] 0.06086907 0.17089670 0.70936436
```

Estimated mean function plot

Estimated mean function plots are useful for describing and comparing the expected number of recurrent events across different latent groups over time. `SLCARE()` computes the estimated mean function over time according to equation (15), which can be plotted with the following code:

```
plot(model1, type = "EstMeans")
```

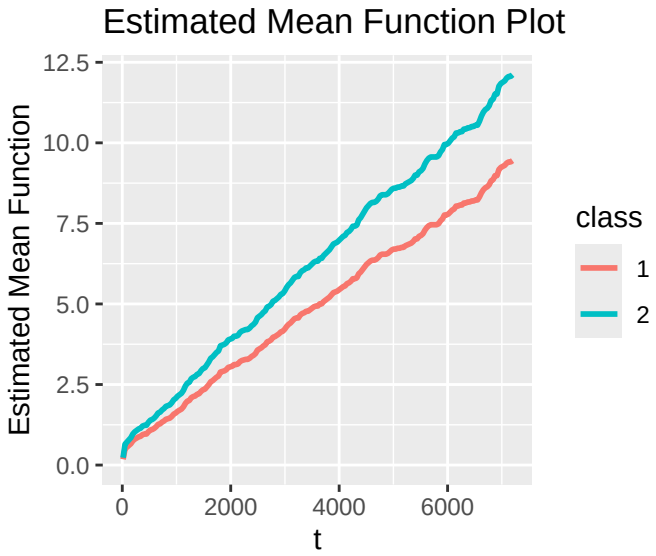

Figure 3: Estimated mean function for two latent classes.

5 A real application

We apply **SLCARE** to a dataset from a randomized phase III clinical trial FFCD 2000-05 conducted between 2002 and 2007 in patients diagnosed with advanced colorectal cancer (Ducreux et al., 2011).

The dataset is publicly available in **frailtypack** package (Rondeau et al., 2012) and is also included in the **SLCARE** package. One recurrent event of interest is the appearance of new colorectal lesions. The colorectal lesion is the region in a colon that has suffered damage through colorectal cancer. The appearance of new colorectal lesions indicates colorectal cancer progression. The main interest of our analysis is to explore the heterogeneity in the recurrence patterns of colorectal lesions.

To address this interest, we extract the dataset `colorectalSLCARE` from the `colorectal` dataset. The dataset 'colorectalSLCARE is arranged in the long format, consisting of 150 patients and 289 records.

Each record stores the value of the six variables described as follows. Variable `id` is the unique patient identifier. Variable `time0` is the start time (in years) of the recurrent event interval. Variable `time1` records the interval end time (in years) of the detection of new colorectal lesions or terminal event (death or right-censoring). Variable `new.lesions = 1` indicates that the appearance of a new colorectal lesion is observed and `new.lesions = 0` indicates that a terminal event is observed. The other two variables represent covariates of interest: (1) treatment, which is coded as `treatment = 1` if the patient received a combination chemotherapy and as `treatment = 0` if the patient received sequential chemotherapy; (2) previous resection indicator, which is coded as `prev.resection = 1` if the patient had a previous resection and as `prev.resection = 0` otherwise.

The construction and the structure of the `colorectalSLCARE` dataset are presented below.

```
data("colorectal", package = "SLCARE")
colorectalSLCARE <- colorectal |>
  dplyr::select(id, time0, time1, new.lesions, treatment, prev.resection) |>
  dplyr::mutate(id = paste0("patient", id), treatment = as.numeric(treatment) -1,
                prev.resection = as.numeric(prev.resection) -1)
str(colorectalSLCARE, vec.len = 3)

#> 'data.frame':    289 obs. of  6 variables:
#>  $ id            : chr  "patient1" "patient2" "patient3" ...
#>  $ time0         : num  0 0 0 0.525 ...
#>  $ time1         : num  0.71 1.282 0.525 0.921 ...
#>  $ new.lesions   : int  0 0 1 1 0 1 0 1 ...
#>  $ treatment     : num  0 1 0 0 0 1 1 0 ...
#>  $ prev.resection: num  0 0 0 0 0 1 1 0 ...
```

We fit models (1) and (2) to the dataset `colorectal_SLCARE` with $T_i^{(j)}$ corresponding to time to the j-th detection of new colorectal lesions. We consider three candidate distributions for W, the frailty term in model (1), which are $W = 1$ and $Gamma(r, r)$, $r = 1, 3$. For each candidate $f_W(\cdot)$, we calculate the relative entropy measure *Entropy* as described in Section **Estimation and inference procedures** using the command illustrated in Section **Illustrations**. The results are presented in Table 2.

Table 2: Real data example: Relative entropy calculated with different choices of K and $f_W(w)$.

	K = 2	K = 3
Gamma(1,1)	0.785	0.788
Gamma(3,3)	0.802	0.793
W = 1	0.462	0.459

As shown in Table 2, the maximum relative entropy is achieved with the combination of $K = 2$ and $Gamma(3, 3)$. Therefore, we set $K = 2$ and select $f_W(\cdot)$ as the density of $Gamma(3, 3)$ for the rest of the analysis.

```
set.seed(66)
finalmodel <- SLCARE(formula = "treatment + prev.resection", data = colorectalSLCARE,
      id_col = "id", start_col = "time0", stop_col = "time1", event_col = "new.lesions",
       K = 2, gamma = 3, boot = 200)
summary(finalmodel, digits = 3)

#> Call:
#> SLCARE(formula = "treatment + prev.resection", data = colorectalSLCARE,
#>     id_col = "id", start_col = "time0", stop_col = "time1", event_col = "new.lesions",
#>     K = 2, gamma = 3, boot = 200)
```

```
#>
#> Coefficients for Beta:
#>        (intercept) treatment prev.resection
#> class1       1.696    -0.415         -0.493
#> class2       0.811     0.691          0.494
#>
#> Std. Errors for Beta:
#>        (intercept) treatment prev.resection
#> class1        0.34     0.281          0.277
#> class2        0.48     0.887          0.703
#>
#> P. Values for Beta:
#>        (intercept) treatment prev.resection
#> class1    6.26e-07     0.140         0.0751
#> class2    9.11e-02     0.436         0.4823
#>
#> Coefficients for Alpha:
#>        treatment prev.resection
#> class1       0.0            0.0
#> class2     -12.3          -11.7
#>
#> Std. Errors for Alpha:
#>        treatment prev.resection
#> class1      0.00           0.00
#> class2      3.96           1.26
#>
#> P. Values for Alpha:
#>        treatment prev.resection
#> class1       NaN            NaN
#> class2   0.00196       1.51e-20
#>
#> Relative Entropy: 0.802
```

Based on the results from fitting models (1) and (2), we classify patients into two groups according to the modal class assignment rule, i.e., $\hat{\xi}_i = \arg\max_{1\le k\le K} \hat{\tau}_{ik}$. Table 3 (generated by R package **table1** (Rich, 2023)) summarizes the characteristics of the resulting two groups, Class 1 of size 127 and Class 2 of size 23. Comparing the total number of the observed colorectal lesions, captured by $N_i(C_i)$, we note that patients belong to Class 1 tend to experience more colorectal lesions as compared to those belong to Class 2. There is also notable separation in covariates `treatment` and `prev.resection`. That is, over 50% patients in Class 1 received combination chemotherapy in contrast to 0% patients in Class 2 on combination chemotherapy, and over 70% patients in Class 1 has previous resection, while all patients in Class 2 did not have resection in the past. These observations are consistent with the estimation results for α_0 (see model summary), which suggest that patients receiving combination chemotherapy or with previous resection are much less likely to be classified to Class 2. These observations entail a plausible characterization of the two latent classes. That is, Class 1 consists of patients with more aggressive progression that incurs more intensive past or ongoing treatment, while Class 2 is featured by relatively mild disease that requires less complicated treatment regimens.

We next examine the covariate effects within each latent class. The coefficient estimates presented in model summary indicate a reasonable direction of how `treatment` and `prev.resection` influence the intensity of lesion recurrence. That is, combination chemotherapy or previous resection may help reduce the risk of lesion recurrence. However, the estimated effects which conform to our intuition are associated with p values that do not reach statistical significance.

Table 3: Real data example: Characteristics of the two latent classes.

	Class 1	Class 2	P-value
	(N=127)	(N=23)	
observed.events			
Mean (SD)	1.02 (0.988)	0.391 (0.583)	<0.001
Median [Min, Max]	1.00 [0, 4.00]	0 [0, 2.00]	
treatment			
sequential chemotherapy	54 (42.5%)	23 (100%)	<0.001
combination chemotherapy	73 (57.5%)	0 (0%)	
prev.resection			
no	37 (29.1%)	23 (100%)	<0.001
yes	90 (70.9%)	0 (0%)	

In Figure 4, we plot the average estimated mean function of experiencing new colorectal lesions for the two latent classes. Figure 4 shows that Class 1 is associated with a lower frequency of experiencing new colorectal lesions than Class 2. Such a distinction may be contributed by the underlying difference in disease severity between the two latent classes, which is also manifested by the different distributions of receiving combination chemotherapy and having previous resection between these two classes demonstrated in Table 3.

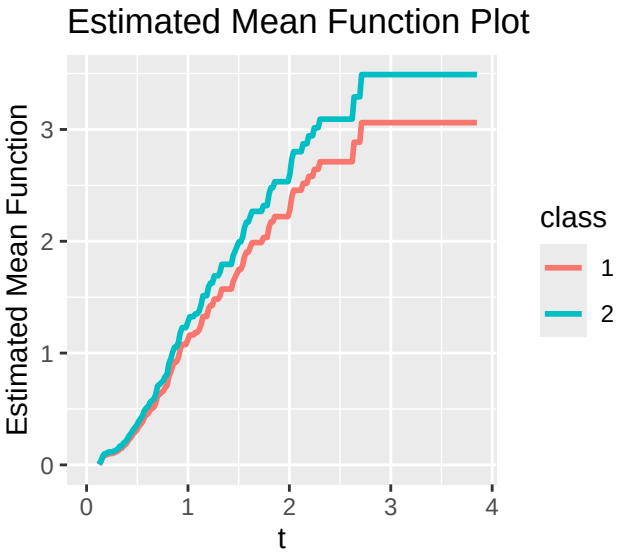

Figure 4: Real data example: Estimated mean function of experiencing new colorectal lesions for two classes.

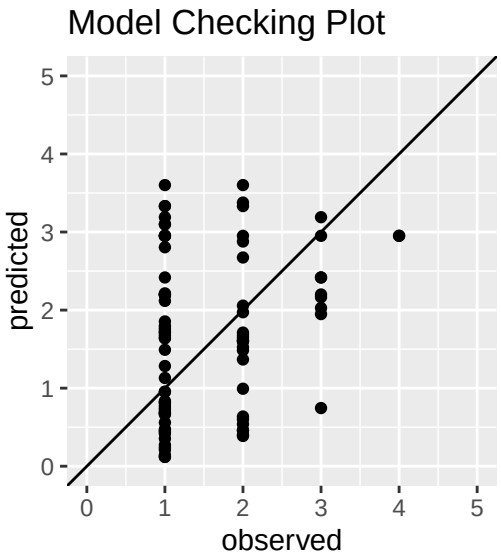

Figure 5: Real data example: Predicted numbers of new colorectal lesions versus the observed numbers of new colorectal lesions.

We further check the overall fit of the latent class models adopted by `SLCARE()` to the `colorectalSLCARE` dataset using the model checking plot described in Section **Methodological background**. As shown in Figure 5, the dots representing the predicted and the observed numbers of new colorectal lesions are generally balanced around the 45-degree line when there are two observed colorectal lesions. However, over-prediction and under-prediction are noted when the observed number of of colorectal lesions is one or three respectively. These observations may suggest a moderate lack-of-fit of the fitted model to this colorectical dataset.

6 Discussion

The R package **SLCARE** provides an easy-to-use software to conduct latent class analysis of recurrent events based on a flexible semiparametric multiplicative intensity modeling proposed by Zhao et al. (2022). A practical automated initializer is embedded in **SLCARE** to help users set initial estimates in an informative way. **SLCARE** provides a single command line function to implement full analysis with optional arguments for model customization. **SLCARE** also provides visualization tools for result summary and model evaluation with **ggplot2** plotting environment.

The R package **SLCARE** is maintained in both Comprehensive R Archive Network (CRAN) at https://CRAN.R-project.org/package=SLCARE and Github at https://github.com/qyxxx/SLCARE. There are several interesting directions to further improve this package. For example, computational speed may be improved with more efficient coding. A new option may be added to perform automated selection of K and frailty distribution. Such updates of package **SLCARE** will be made available at CRAN and Github.

References

P. K. Andersen and R. D. Gill. Cox's regression model for counting processes: a large sample study. *The annals of statistics*, 1982. URL https://www.jstor.org/stable/2240714. [p134]

G. Celeux and G. Soromenho. An entropy criterion for assessing the number of clusters in a mixture model. *Journal of Classification*, 1996. URL https://doi.org/10.1007/BF01246098. [p137]

S. H. Chiou, G. Xu, J. Yan, and C.-Y. Huang. Regression modeling for recurrent events possibly with an informative terminal event using R package reReg. *Journal of Statistical Software*, 2023. URL https://doi.org/10.18637/jss.v105.i05. [p134, 137]

D. Y. Clement and R. L. Strawderman. Conditional GEE for recurrent event gap times. *Biostatistics*, 2009. URL https://doi.org/10.1093/biostatistics/kxp004. [p134]

M. Ducreux, D. Malka, J. Mendiboure, P.-L. Etienne, P. Texereau, D. Auby, P. Rougier, M. Gasmi, M. Castaing, M. Abbas, et al. Sequential versus combination chemotherapy for the treatment of

advanced colorectal cancer (ffcd 2000–05): an open-label, randomised, phase 3 trial. *The lancet oncology*, 2011. URL https://doi.org/10.1016/S1470-2045(11)70199-1. [p143]

J. Fine, J. Yan, and M. Kosorok. Temporal process regression. *Biometrika*, 2004. URL https://doi.org/10.1093/biomet/91.3.683. [p134]

B. Grün and F. Leisch. FlexMix version 2: Finite mixtures with concomitant variables and varying and constant parameters. *Journal of Statistical Software*, 2008. URL https://doi.org/10.18637/jss.v028.i04. [p134]

F. E. Harrell Jr. *rms: Regression Modeling Strategies*, 2023. URL https://CRAN.R-project.org/package=rms. R package version 6.6-0. [p134]

S. Lloyd. Least squares quantization in PCM. *IEEE transactions on information theory*, 1982. URL https://doi.org/10.1109/TIT.1982.1056489. [p137]

C. Proust-Lima, V. Philipps, and B. Liquet. Estimation of extended mixed models using latent classes and latent processes: The R package lcmm. *Journal of Statistical Software*, 2017. URL https://doi.org/10.18637/jss.v078.i02. [p134]

V. Ramaswamy, W. S. DeSarbo, D. J. Reibstein, and W. T. Robinson. An empirical pooling approach for estimating marketing mix elasticities with PIMS data. *Marketing science*, 1993. URL https://doi.org/10.1287/mksc.12.1.103. [p137]

B. Rich. *table1: Tables of Descriptive Statistics in HTML*, 2023. URL https://CRAN.R-project.org/package=table1. R package version 1.4.3. [p145]

V. Rondeau, Y. Marzroui, and J. R. Gonzalez. frailtypack: an R package for the analysis of correlated survival data with frailty models using penalized likelihood estimation or parametrical estimation. *Journal of Statistical Software*, 2012. URL https://doi.org/10.18637/jss.v047.i04. [p144]

L. A. Stefanski and R. J. Carroll. Conditional scores and optimal scores for generalized linear measurement-error models. *Biometrika*, 1987. URL https://doi.org/10.1093/biomet/74.4.703. [p135]

T. M. Therneau. *A Package for Survival Analysis in R*, 2023. URL https://CRAN.R-project.org/package=survival. R package version 3.5-3. [p134]

W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. 2002. URL https://www.stats.ox.ac.uk/pub/MASS4/. [p137]

M.-C. Wang, J. Qin, and C.-T. Chiang. Analyzing recurrent event data with informative censoring. *Journal of the American Statistical Association*, 2001. URL https://doi.org/10.1198/016214501753209031. [p134, 137]

H. Wickham. Reshaping data with the reshape package. *Journal of Statistical Software*, 2007. URL http://www.jstatsoft.org/v21/i12/. [p138]

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. URL https://ggplot2.tidyverse.org. [p134]

H. Wickham, D. Vaughan, and M. Girlich. *tidyr: Tidy Messy Data*, 2023. URL https://CRAN.R-project.org/package=tidyr. R package version 1.3.0. [p138]

J. Yan and J. Fine. Estimating equations for association structures. *Statistics in medicine*, 2004. URL https://doi.org/10.1002/sim.1650. [p134]

W. Zhao, L. Peng, and J. Hanfelt. Semiparametric latent class analysis of recurrent event data. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 2022. URL https://doi.org/10.1111/rssb.12499. [p134, 135, 136, 137, 139, 147]

Qi Yu
Emory University
Department of Biostatistics and Bioinformatics
1518 Clifton Rd. NE, Atlanta, GA 30322, USA
ORCiD: 0009-0007-7585-1183
qi.yu2@emory.edu

Limin Peng
Emory University
Department of Biostatistics and Bioinformatics
1518 Clifton Rd. NE, Atlanta, GA 30322, USA
ORCiD: 0000-0002-8042-3112
lpeng@emory.edu