PyDTS: A Python Package for Discrete Time Survival Analysis with Competing Risks

Tomer Meir1, Rom Gutman1, and Malka Gorfine2

1 Technion - Israel Institute of Technology
2 Department of Statistics and Operations Research, Tel Aviv University

These authors contributed equally

Corresponding Authors: tomer1812@gmail.com, rom.gutman1@gmail.com

April 13, 2022

Abstract

Time-to-event analysis (survival analysis) is used when the outcome or the response of interest is the time until a pre-specified event occurs. Time-to-event data are sometimes discrete either because time itself is discrete or due to grouping of failure times into intervals or rounding off measurements. In addition, the failure of an individual could be one of several distinct failure types; known as competing risks (events) data. This work focuses on discrete-time regression with competing events. We emphasize the main difference between the continuous and discrete settings with competing events, develop a new estimation procedure, and present PyDTS, an open source Python package which implements our estimation procedure and other tools for discrete-time-survival analysis with competing risks.

Keywords Survival Analysis · Discrete time · Competing risks · Python package

1 Introduction

Discrete-data survival analysis refers to the case where data can only take values over a discrete grid. Sometimes, events can only occur at regular, discrete points in time. For example, in the United States a change in party controlling the presidency only occurs quadrennially in the month of January [1]. In other situations events may occur at any point in time, but available data record only the particular interval of time in which each event occurs. For example, death from cancer measured by months since time of diagnosis [2], or length of stay in hospital recorded on a daily basis. Most methods for regression survival data assume that time is measured as a continuous variable [3]. However, it is well-known that naively using standard continuous-time models (even after correcting for ties) with discrete-time data may result in biased estimators for the discrete time models.

Competing events arise when individuals are susceptible to several types of events but can experience at most one event. For example, competing risks for hospital length of stay are discharge and in-hospital death. Occurrence of one of these events precludes us from observing the other event on this patient. Another classical example of competing risks is cause-specific mortality, such as death from heart disease, death from cancer and death from other causes [4, 5].

Regression analysis of continuous-time data with competing risks can be done by standard tools of non-competing events, since the likelihood function for continuous-time data factors into separate likelihoods for each cause-specific hazard function [4]; but this is not the case in discrete-time data with competing risks (see [2] and references therein). Limited work has been done on discrete-time data with competing risks. For a recent overview the reader is referred to [6]. Most existing works are based on simultaneously estimating all the parameters via the full likelihood function, which are computationally time consuming. In contrast, Lee et al. [2] showed that if one naively treats competing events as censoring in the discrete-time likelihood, separate estimation of cause-specific hazard models for different
event types may be accomplished using a collapsed likelihood which is equivalent to fitting a generalized linear model to repeated binary outcomes. Moreover, the maximum likelihood estimators are consistent and asymptotically normal under the some regularity assumptions, which gives rise to Wald confidence intervals and likelihood ratio tests for the effects of covariates. To the best of our knowledge, currently, there is no publicly available Python implementation of any regression model of discrete-time survival data with competing events.

In this work we adopt the collapsing approach of [2] but provide a new estimation technique that saves substantial computation time with no substantial loss of accuracy or efficiency, as will be demonstrated by simulations. Additionally, we present PyDTS, a Python package which implements the estimation methods of Lee et al. [2] and ours.

2 Methods

2.1 Notation and Models

We let $T$ denote a discrete event time that can take on only the values $\{1, 2, ..., d\}$ and $J$ denote the type of event, $J \in \{1, \ldots, M\}$. Consider a $p \times 1$ vector of baseline covariates $Z$. A general discrete cause-specific hazard function is of the form

$$
\lambda_j(t|Z) = \Pr(T = t, J = j|T \geq t, Z) \quad t \in \{1, 2, ..., d\} \quad j = 1, \ldots, M.
$$

A popular semi-parametric model of the above hazard function based on a transformation regression model is of the form

$$
h\left(\lambda_j(t|Z)\right) = \alpha_{jt} + Z^T \beta_j \quad t \in \{1, 2, ..., d\} \quad j = 1, \ldots, M
$$

such that $h$ is a known function and reference therein. The total number of parameters in the model is $M(d + p)$. The logit function $h(a) = \log\{a/(1-a)\}$ yields

$$
\lambda_j(t|Z) = \frac{\exp(\alpha_{jt} + Z^T \beta_j)}{1 + \exp(\alpha_{jt} + Z^T \beta_j)}.
$$

It should be noted that leaving $\alpha_{jt}$ unspecified is analogous to having an unspecified baseline hazard function in the Cox proportional hazard model [7], and thus we consider the above as a semi-parametric model.

Let $S(t|Z) = \Pr(T > t|Z)$ be the overall survival given $Z$. Then, the probability of experiencing event of type $j$ at time $t$ equals

$$
\Pr(T = t, J = j|Z) = \lambda_j(t|Z) \prod_{k=1}^{t-1} \left\{1 - \sum_{j'=1}^{M} \lambda_{j'}(k|Z)\right\}
$$

and the cumulative incident function (CIF) of cause $j$ is given by

$$
F_j(t|Z) = \Pr(T \leq t, J = j|Z) = \sum_{m=1}^{t} \lambda_j(m|Z)S(m-1|Z)
$$

$$
= \sum_{m=1}^{t} \lambda_j(m|Z) \prod_{k=1}^{m-1} \left\{1 - \sum_{j'=1}^{M} \lambda_{j'}(k|Z)\right\}.
$$

Finally, the marginal probability of event type $j$ (marginally with respect to the time of event), given $Z$, equals

$$
\Pr(J = j|Z) = \sum_{m=1}^{d} \lambda_j(m|Z) \prod_{k=1}^{m-1} \left\{1 - \sum_{j'=1}^{M} \lambda_{j'}(k|Z)\right\}.
$$

In the next section we provide a fast estimation technique of the parameters $\{\alpha_{j1}, \ldots, \alpha_{jd}, \beta_j^T : j = 1, \ldots, M\}$.

2.2 The Collapsed Log-Likelihood Approach and the Proposed Estimators

For simplicity of presentation, we assume two competing events, i.e., $M = 2$ and our goal is estimating $\{\alpha_{11}, \ldots, \alpha_{1d}, \beta_{1}^T, \alpha_{21}, \ldots, \alpha_{2d}, \beta_{2}^T\}$ along with the standard error of the estimators. The data at hand consist of $n$ independent observations, each with $(X_i, \delta_i, J_i, Z_i)$ where $X_i = \min(C_i, T_i)$, $C_i$ is a right-censoring time, $\delta_i = I(X_i = T_i)$ is the event indicator and $J_i \in \{0, 1, 2\}$, where $J_i = 0$ if and only if $\delta_i = 0$. Assume that given
Alternatively, we propose the following simpler and faster estimation procedure, with a negligible efficiency loss, if any.

Our idea exploits the close relationship between conditional logistic regression analysis and stratified Cox regression, in general, for $M$ categories and one can fit the two models, separately.

Similarly, the collapsed log-likelihood for cause $J$ is given by

$$L = \prod_{i=1}^{n} \prod_{j=1}^{2} \prod_{m=1}^{X_i} \left\{ \frac{\lambda_j(m|Z_i)}{1 - \lambda_1(m|Z_i) - \lambda_2(m|Z_i)} \right\}^{\delta_{jim}} \prod_{k=1}^{X_i} \{1 - \lambda_1(k|Z_i) - \lambda_2(k|Z_i)\}$$

where $\delta_{jim}$ equals one if subject $i$ experienced event type $j$ at time $m$; and 0 otherwise. Clearly $L$ cannot be decomposed into separate likelihoods for each cause-specific hazard function $\lambda_j$. The log likelihood becomes

$$\log L = \sum_{i=1}^{n} \sum_{j=1}^{2} \sum_{m=1}^{X_i} \left[ \log \lambda_j(m|Z_i) - \delta_{jim} \{1 - \lambda_1(m|Z_i) - \lambda_2(m|Z_i)\} \right]$$

$$+ \sum_{k=1}^{X_i} \log \{1 - \lambda_1(k|Z_i) - \lambda_2(k|Z_i)\}$$

$$= \sum_{i=1}^{n} \sum_{m=1}^{X_i} \left[ \log \lambda_1(m|Z_i) + \log \lambda_2(m|Z_i) \right]$$

$$+ \{1 - \delta_{1im} - \delta_{2im}\} \log \{1 - \lambda_1(m|Z_i) - \lambda_2(m|Z_i)\} .$$

Instead of maximizing the $M(d + p)$ parameters simultaneously based on the above log-likelihood, the collapsed log-likelihood of Lee et al. [2] can be adopted. Specifically, the data are expanded such that for each observation $i$ the expanded dataset includes $X_i$ rows, one row for each time $t$, $t \leq X_i$. At each time point $t$ the expanded data are conditionally multinomial with one of three possible outcomes $\{\delta_{1it}, \delta_{2it}, 1 - \delta_{1it} - \delta_{2it}\}$. Then, for estimating $\{\alpha_{11}, \ldots, \alpha_{1d}, \beta_j^T\}$, we combine $\delta_{2it}$ and $1 - \delta_{1it} - \delta_{2it}$, and the collapsed log-likelihood for cause $J = 1$ based on a binary regression model with $\delta_{1it}$ as the outcome is given by

$$\log L_1 = \sum_{i=1}^{n} \sum_{m=1}^{X_i} \left[ \log \lambda_1(m|Z_i) + (1 - \delta_{1im}) \log \{1 - \lambda_1(m|Z_i)\} \right] .$$

Similarly, the collapsed log-likelihood for cause $J = 2$ based on a binary regression model with $\delta_{2it}$ as the outcome becomes

$$\log L_2 = \sum_{i=1}^{n} \sum_{m=1}^{X_i} \left[ \log \lambda_2(m|Z_i) + (1 - \delta_{2im}) \log \{1 - \lambda_2(m|Z_i)\} \right]$$

and one can fit the two models, separately.

In general, for $M$ competing events, the estimators of $\{\alpha_{j1}, \ldots, \alpha_{jd}, \beta_j^T\}$, $j = 1, \ldots, M$, are the respective values that maximize

$$\log L_j = \sum_{i=1}^{n} \sum_{m=1}^{X_i} \left[ \log \lambda_j(m|Z_i) + (1 - \delta_{jim}) \log \{1 - \lambda_j(m|Z_i)\} \right] .$$

Namely, each maximization $j$, $j = 1, \ldots, M$, consists of maximizing $d + p$ parameters simultaneously.

Alternatively, we propose the following simpler and faster estimation procedure, with a negligible efficiency loss, if any. Our idea exploits the close relationship between conditional logistic regression analysis and stratified Cox regression analysis [3]. We propose to estimate each $\beta_j$ separately, and given $\beta_j, \alpha_{jt}, t = 1, \ldots, d$, are separately estimated. In particular, the proposed estimating procedure consists of the following two speedy steps:

Step 1. Use the expanded dataset and estimate each vector $\beta_j$, $j \in \{1, \ldots, M\}$, by a simple conditional logistic regression, conditioning on the event time $X$, using a stratified Cox analysis.
Step 2. Given the estimators $\hat{\beta}_j$, $j \in \{1, \ldots, M\}$, of Step 1, use the original (un-expanded) data and estimate each $\alpha_{jt}$, $j \in \{1, \ldots, M\}$, $t = 1, \ldots, d$, separately, by

$$\hat{\alpha}_{jt} = \arg\min_a \left\{ \frac{1}{y_t} \sum_{i=1}^n I(X_i \geq t) \frac{\exp(a + Z_i^T \hat{\beta}_j)}{1 + \exp(a + Z_i^T \beta^j)} - \frac{n_{tj}}{y_t} \right\}^2 \quad (2)$$

where $y_t = \sum_{i=1}^n I(X_i \geq t)$ and $n_{tj} = \sum_{i=1}^n I(X_i = t, J_i = j)$.

Eq. (2) consists minimizing the squared distance between the observed proportion of failures of type $j$ at time $t$ ($n_{tj}/y_t$) and the expected proportion of failures given Model 1 and $\hat{\beta}_j$. The simulation results of Section 3 reveals that the above two-step procedure performs well in terms of bias, and provides similar standard error of that of [2]. However, the improvement in computational time, by using our procedure, could be improved by a factor of 1.5-3.5 depending on $d$, where the improvement factor increases as a function of $d$. Standard errors of $\hat{\beta}_j$, $j \in \{1, \ldots, M\}$, can be derived directly from the stratified Cox analysis.

3 PyDTS

PyDTS is an open source Python package which implements tools for discrete-time survival analysis with competing risks. The code is available at https://github.com/tomer1812/pydts [9] under MIT license. Documentation, with details about main functionalities, API, installation and usage examples, is available at https://tomer1812.github.io/pydts/ [10]. The package was developed following best practices of Python programming, automated testing was added for stability, Github Issues can be opened by any user for maintainability and open source community contributions are available for ongoing improvement.

The package is published using the Python Package Index (PyPI), can be installed (using the “pip” installer) and used with a few simple lines of code, as shown in the Quick Start section of the documentation. Yet, it is flexible enough to support more advanced pre-processing and fitting options, as demonstrated in the Hospitalization Length of Stay example of the documentation.

Based on available Python packages (Numpy [11], Pandas [12] [13], Scipy [14], Scikit-survival [15], Lifelines [16], and Statsmodels [17]), PyDTS package implements DataExpansionFitter, which is the estimation procedure of Lee et al. [2], and TwoStagesFitter, our proposed two-step estimation method.

We conducted a simulation study demonstrating the performances of the proposed approach and comparing it with that of Lee et al. [2]. For simplicity of presentation, we considered $M = 2$ competing events, though PyDTS can handle any number of competing events as long as there are enough observed failure time of each failure type, at each discrete time point. Here, $d = 30$ discrete time points, $n = 50,000$ observations, 5 covariates and right censoring. Failure times of observations were generated based on Model 1 with $\alpha_{1t} = -1 - 0.3 \log(t)$, $\alpha_{2t} = -1.75 - 0.15 \log(t)$, $t = 1, \ldots, d$, $\beta_1 = (-\log 0.8, \log 3, \log 3, \log 2.5, \log 2)$, and $\beta_2 = (-\log 1, \log 3, \log 4, \log 3, \log 2)$. Censoring time for each observation was sampled from a discrete uniform distribution, i.e. $C_i \sim \text{Uniform}\{1, \ldots, d+1\}$. The simulation results are based on 100 replications, and results are summarized in figures [1] and [3]. Evidently, both estimation methods perform very well in terms of bias and provide highly similar results in terms of point estimators and their standard errors (Fig. [1]). However, Fig. [3] shows that the computational running time of our proposed approach is 1.5-3.5 times shorter depending on $d$, where the improvement factor increases as a function of $d$. 4
Figure 1: Simulation results of both estimation methods and true values. **a** Results of $\alpha_{jt}$. True values of $\alpha_{1t}$ are in dashed blue line, Lee et al.’s estimates are in light blue circles, and our estimates are in blue stars. True values of $\alpha_{2t}$ are in dashed green line, Lee et al.’s estimates are in light green circles, and our estimates are in green stars. Number of events at each time $t$ is shown in red bars. **b** Results of $\beta_j$. True values of $\beta_1$ are in light blue bars, Lee et al.’s estimates are in light blue right arrowhead, and our estimates are in blue right triangle. True values of $\beta_2$ are in green bars, Lee et al.’s estimates are in light green left arrowhead, and our estimates are in green left triangle.

Figure 2: Simulation results of coefficients std of Lee et al. and our proposed procedure with 100 repetitions, $M = 2$, $n = 50,000$, $d = 30$ and $p = 5$. **a** $\alpha_1$ coefficients std for event $j=1$, **b** $\alpha_2$ coefficients std for event $j=2$, **c** $\beta_1$ coefficients std for event $j=1$, **d** $\beta_2$ coefficients std for event $j=2$. Equal std line is shown in **a-d** as a light green dashed line.
Figure 3: Simulation results of Computation time of Lee et al. and our proposed procedure with 100 repetitions for each value of $d$, $M = 2$, $n = 50,000$, and $p = 5$.

4 Summary

Discrete-time survival analysis with competing-risks is required in many practical cases. This work provides a new estimation procedure that could be highly useful in large datasets with a large number of covariates. We provided a simple to use Python package which implements the estimation procedure of Lee et al. [2] and our proposed method. We expect PyDTS to be adopted by Python users and to be further improved by comments and contributions from the open source Python community.

References

[1] Paul D. Allison. “Discrete-Time Methods for the Analysis of Event Histories”. In: Sociological Methodology 13 (1982), p. 61. ISSN: 00811750. DOI: 10.2307/270718.

[2] Minjung Lee, Eric J. Feuer, and Jason P. Fine. “On the analysis of discrete time competing risks data”. In: Biometrics 74.4 (Dec. 2018), pp. 1468–1481. ISSN: 0006-341X, 1541-0420. DOI: 10.1111/biom.12881.

[3] Rachel Heyard et al. “Dynamic clinical prediction models for discrete time-to-event data with competing risks—A case study on the OUTCOMEREA database”. In: Biometrical Journal 61.3 (2019), pp. 514–534.

[4] John D. Kalbfleisch and Ross L. Prentice. The Statistical Analysis of Failure Time Data. 2nd. Wiley, Jan. 2011. ISBN: 978-1-118-03123-0.

[5] John P. Klein and Melvin L. Moeschberger. Survival Analysis. Springer, 2003. ISBN: 978-0-387-95399-1.

[6] Matthias Schmid and Moritz Berger. “Competing risks analysis for discrete time-to-event data”. In: Wiley Interdisciplinary Reviews: Computational Statistics 13.5 (2021), e1529.

[7] D. R. Cox. “Regression Models and Life-Tables”. In: Journal of the Royal Statistical Society: Series B (Methodological) 34.2 (Jan. 1972), pp. 187–202. ISSN: 00359246. DOI: 10.1111/j.2517-6161.1972.tb00899.x.

[8] Ross L Prentice and Norman E Breslow. “Retrospective studies and failure time models”. In: Biometrika 65.1 (1978), pp. 153–158.

[9] Tomer Meir, Rom Gutman, and Malka Gorfine. PyDTS - Python Package for Discrete Time Survival-analysis - Github Repository. Apr. 2022. URL: https://github.com/tomer1812/pydts
Data availability statement

Simulated data is available at the package repository https://github.com/tomer1812/pydts

Code availability statement

The code is available under the MIT license at https://github.com/tomer1812/pydts

Competing Interests Statement

The authors declare no competing interests.

Authors Contributions

T.M. and R.G. conceived the project, designed and implemented the Python package PyDTS, conducted the analyses, interpreted the results and wrote the manuscript; M.G. developed the statistical theory, designed the analyses, interpreted the results, wrote the manuscript, supervised and conceived the project. M.G. work was supported by ISF 767/21 and Malag competitive grant in data science (DS).

Acknowledgements

The authors would like to thank Hagai Rossman and Ayya Keshet for their helpful discussions and suggestions.