Recurrent Event Analysis with reda and reReg

Recurrent event data arise in situations where the event of interest, such as hospital admissions, infections, or tumor recurrences, can recur in the same individual during follow-up. The standard "time-to-first" event analysis cannot capture the cumulative experience of the recurrent events and could lead to invalid inferences. The R packages **reda** (Wang et al., 2020) and **reReg** (HanChiou and Huang, 2020) provide a collection of visualization tools and statistical methods for exploring and analyzing recurrent event data.

Suppose in a study consists of a random sample of n subjects and $N_i(t)$ be the number of events the ith subject experienced over the interval $[0, t]$. Let D be the failure time of interest that could either be a terminal event (e.g., death) or a non-terminal event (e.g., treatment failure). Let C be the potential censoring time for reasons other than the failure event, the observed data are independent and identically distributed copies $\{N_i(t), Y_i, X_i; t \leq Y_i, i = 1, \ldots, n\}$, where $Y_i = \min(D, C)$, $\Delta_i = I(D \leq C)$, X_i is a covariate vector, $I(\cdot)$ is the indicator function, and the recurrent event process $N_i(\cdot)$ is observed up to the composite censoring time Y_i. Suppose we are interested in making inference about the recurrent event process and the failure event in the time interval $[0, \tau]$, where the constant τ is determined with the knowledge that recurrent and failure events could potentially be observed up to time τ. We will illustrate the rehospitalization data (González et al., 2005) from the **frailtypack** package (Rondeau et al., 2019).

In the packages **reda** and **reReg**, recurrent event data are represented using an object of type `Recur` created by the `Recur()` function. The `Recur` object is an `S4` class object that bundles together a set of recurrent times, failure time, and censoring status, allowing users an easy first glance of the recurrent event data. The `Recur` object is also used as the formula response for many key functions in **reda** and **reReg**. The following commands can be used to create a `Recur` object corresponding to the rehospitalization data:

```
library(reda); library(reReg)
data(readmission, package = "frailtypack")
with(readmission, Recur(t.stop, id, event, death))
```

```
Error: Subjects having multiple terminal events:
60, 109, 280.
```

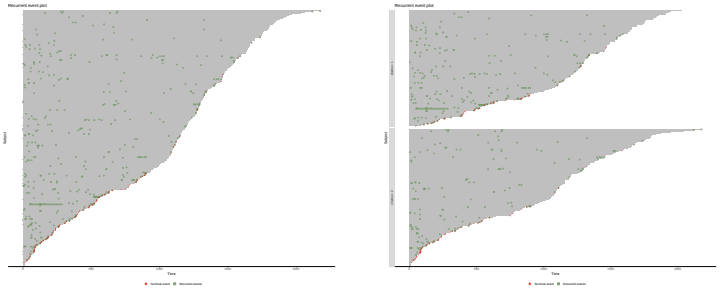
The `Recur()` internally checks whether the specified data fits into the recurrent event data framework and detected a possible issue on the data structure. The `show()` method for `Recur` objects presents recurrent events in intervals, where events happened at end of the recurrent episodes with censoring due to (or not) terminal indicated by a trailing `+` (or `*`). The following prints the `Recur` object for the first five subjects.

```
with(readmission[1:14,], Recur(t.stop, id, event, death))
```

```
[1] 1: (0, 24], (24, 457], (457, 1037+]
[2] 2: (0, 489], (489, 1182+]
[3] 3: (0, 15], (15, 783*]
[4] 4: (0, 163], (163, 288], ..., (686, 2048+]
[5] 5: (0, 1134], (1134, 1144+]
```

Another easy way to glance at recurrent event data is by event plots, which can be created by applying the generic function `plot()` to the `Recur` object when the **reReg** package is loaded. Additionally, the `plotEvents()` function from the **reReg** package allows users to stratify the event plots by discrete variables. The following codes produces event plots with and without stratifying by whether the patients receive chemotherapy.

```
df0 <- subset(readmission, !(id %in% c(60, 109, 280)))
obj <- with(df0, Recur(t.stop, id, event, death))
plot(obj, legend = "bottom") ## no stratification
fn <- Recur(t.stop, id, event, death) ~ chemo
plotEvents(fn, data = df0, legend = "bottom") ## by chemo
```



(a) No stratification (b) Stratified by `chemo`

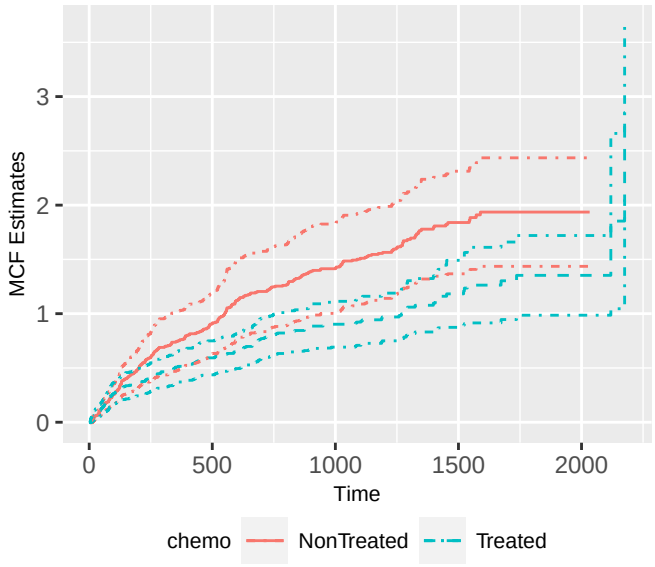
Figure 1: Event plots

The mean cumulative function (MCF) is often the focus in a nonparametric analysis of recurrent data. Let $M_i(t) = \mathbb{E}\{N_i(t)\}$ denote the MCF of $N_i(t)$. The Nelson-Aalen estimator (Nelson, 2003) are widely utilized in exploring the trend of recurrent event data.

$$\hat{M}(t) = \int_0^t \frac{dN(s)}{\delta(s)},$$

where $dN(s) = \sum_{i=1}^k dN_i(s)$, $\delta(s) = \sum_{i=1}^k \delta_i(s)$, $dN_i(s)$ and $\delta_i(s)$ is, respectively, the jump size and at-risk indicator of process i at time s. The MCF can be visualized by plotting the `Recur` object with argument `MCF = TRUE` when the **reReg** package is active. The **reReg** package also offers the `plotMCF()` function for plotting MCF estimates to be stratified by discrete variables. Alternatively, the `mcf()` function from the **reda** package provides a more sophisticated approach to plot MCFs. Some of the unique features the `mcf()` has include the different variance estimations, confidence interval constructions, and predictions. The following example uses the `mcf()` function to visualize MCF estimates stratified by whether the patients receive chemotherapy. The `plot()` method, as well as the `plotEvents()` function, return a `ggplot2` object (Wickham, 2016) so that users may further customize the plot easily.

```
re_mcf <- mcf(fn, data = df0)
plot(re_mcf, conf.int = TRUE, lty = 1:2) +
    ggplot2::theme(legend.position = "bottom")
```



Furthermore, the MCF difference between two groups can be tested with the two-sample pseudo-score tests (Cook et al., 1996) via `mcfDiff.test()` as follows:

```
mcfDiff.test(re_mcf)
```

```
Two-Sample Pseudo-Score Tests:
                Statistic Variance  Chisq DF Pr(>Chisq)
Constant Weight     47.49   416.71   5.41  1      0.020 *
Linear Weight       36.56   263.59   5.07  1      0.024 *
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Variance Estimator: robust
```

Both the constant weighted and the linear weighted tests indicate the MCF estimates are statistically different at a significance level of 0.05.

The `reReg()` function from the **reReg** package provides flexible approaches to fit semiparametric regression model to recurrent event data. At the default specification, a joint scale-change model for the recurrent event process and the failure time is fitted. Specifically, the joint model takes the form of

$$\lambda(t) = Z\lambda_0(te^{X^\top\alpha})e^{X^\top\beta}; h(t) = Zh_0(te^{X^\top\alpha})e^{X^\top\beta}, \quad (1)$$

where Z is a latent shared frailty variable to account for association between the two types of outcomes, and (α, β) is the regression coefficient.

Reference

Cook, R. J., Lawless, J. F., and Nadeau, C. (1996), "Robust Tests for Treatment Comparisons Based on Recurrent Event Responses," *Biometrics*, 52, 557–571.

González, J. R., Fernandez, E., Moreno, V., Ribes, J., Peris, M., Navarro, M., Cambray, M., and Borràs, J. M. (2005), "Sex differences in hospital readmission among colorectal cancer patients," *Journal of Epidemiology & Community Health*, 59, 506–511.

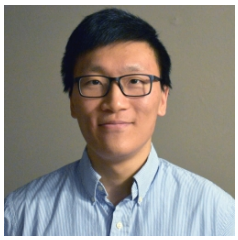
HanChiou, S. and Huang, C.-Y. (2020), **reReg**: *Recurrent Event Regression*, R package version 1.2.1.

Nelson, W. B. (2003), *Recurrent Events Data Analysis for Product Repairs, Disease Recurrences, and Other Applications*, Society for Industrial and Applied Mathematics.

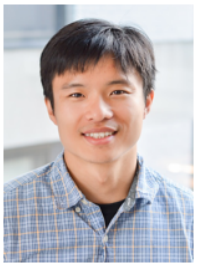
Rondeau, V., Gonzalez, J. R., Mazroui, Y., Mauguen, A., Diakite, A., Laurent, A., Lopez, M., Król, A., and Sofeu, C. L. (2019), *frailtypack: General Frailty Models: Shared, Joint and Nested Frailty Models with Prediction; Evaluation of Failure-Time Surrogate Endpoints*, R package version 3.0.3.

Wang, W., Fu, H., and Yan, J. (2020), **reda**: *Recurrent Event Data Analysis*, R package version 0.5.1.

Wickham, H. (2016), *ggplot2: Elegant Graphics for Data Analysis*, Springer-Verlag New York.



Wenjie Wang
Research Scientist
Machine Learning, Artificial Intelligence, and Connected Care
Advanced Analytics and Data Sciences
Eli Lilly and Company
Email: `wang_wenjie@lilly.com`



Sy Han Chiou
Assistant Professor
Department of Mathematical Sciences
University of Texas at Dallas
Email: `schiou@utdallas.edu`