Journal of Statistical Software

July 2014, Volume 59, Issue 2. *http://www.jstatsoft.org/*

The R Package survsim for the Simulation of Simple and Complex Survival Data

David Moriña
CREAL

Albert Navarro
Universitat Autònoma de Barcelona

Abstract

We present an R package for the simulation of simple and complex survival data. It covers different situations, including recurrent events and multiple events. The main simulation routine allows the user to introduce an arbitrary number of distributions, each corresponding to a new event or episode, with its parameters, choosing between the Weibull (and exponential as a particular case), log-logistic and log-normal distributions.

Keywords: simulation, recurrence, survival analysis, accelerated failure time.

1. Introduction

This paper describes the R (R Core Team 2014) package **survsim** (Moriña and Navarro 2014) available from the Comprehensive R Archive Network (CRAN) at `http://CRAN.R-project.org/package=survsim` for the simulation of simple and complex survival data, including multiple and recurrent survival data.

Compared to the simulation of other types of data, the process of simulating survival data requires certain specific considerations. First, to simulate the (randomly right) censored observations, we need to simulate a lifetime vector and, independently, a censoring time vector. On the other hand, there are various situations which make true survival data much more complex. The phenomenon of interest could occur more than once in a given individual (recurrent events), or perhaps we are interested in the instantaneous analysis of multiple events of different types. Moreover, we can work with dynamic cohorts in which an individual may be incorporated to follow-up after the study has begun, or an individual may be incorporated to follow-up after being at risk for some time. Discontinuous risk intervals or individual heterogeneity (propensity of an individual to suffer an event due to hidden variables) are other phenomena which involve situations that make simulation even more difficult.

Function	Purpose
`simple.surv.sim`	Simulation of standard survival times.
`mult.ev.sim`	Simulation of survival times in a context of multiple events.
`rec.ev.sim`	Simulation of survival times in a context of recurrent events.
`accum`	Aggregation of data generated by one of the above functions by subject.

Table 1: Main functions of **survsim** package.

Although there is an increasing interest, and need, to apply survival analysis to data sets with multiple and recurrent events per subject (Therneau and Grambsch 2000), very few published articles are available in the literature which make use of the simulation of complex survival data, one possible reason being a lack of easily used tools facilitating such simulations. Stata (StataCorp. 2013) has the **SURVSIM** module for the simulation of complex survival data, also in a context of non-proportional hazards, as described by Crowther (2011). Also, the **PermAlgo** package (Sylvestre, Evans, MacKenzie, and Abrahamowicz 2013) is available on CRAN, which allows simulating survival times based on an extension of the permutational algorithm introduced in Abrahamowicz, MacKenzie, and Esdaile (1996). The analysis of this type of data can be performed in R using the **eha** package (Broström 2014), which allows fitting models for recurrent and multiple events, as well as other contexts (Broström 2012).

2. Overview

The basic structure of the package **survsim** is detailed in Table 1. The package is based on three functions that generate data frames of simulated survival data in the contexts described in Table 1. Once a simulated data frame containing individual data is generated, the function `accum` allows the user to aggregate the data by subject.

In the next section we will describe the kind of data we are dealing with, in its simpler form, and give an example of the use of the package on the simulation of simple, single-event survival data. In Section 4, we will describe the simulation of multiple-event survival data and provide an example of the use of the package related with the appearance of adverse events in a clinical trial. The use of the package applied to recurrent events will be described in Section 5, based on an example about recurrent sick leave episodes. The structure of the data frame returned by the functions of the package is detailed in Section 6. Section 7 is focused on aggregated data instead of individual results, where for each subject only the total number of episodes and the accumulated follow-up are known. Finally, some applications using R packages **survival** (Therneau 2014; Therneau and Grambsch 2000) and **MASS** (Venables and Ripley 2002) for all the kinds of survival and aggregated data introduced throughout the paper are shown in Section 8.

3. Simple survival data

3.1. Description

Survival analysis is generally defined as a set of methods for analyzing data where the outcome

Distribution	Survival function	Density function	Parametrization
Weibull	$e^{(-\lambda_j t^p)}$	$\lambda_j p t^{p-1} e^{(-\lambda_j t^p)}$	$\lambda_j = e^{-p(\beta_0 + X_j \beta_j)}$
Log-normal	$1 - \Theta\left(\frac{\log(t)-\mu_j}{\sigma}\right)$	$\frac{1}{t\sigma\sqrt{2\pi}} e^{\left(-\frac{1}{2\sigma^2}(\log(t)-\mu_j)^2\right)}$	$\mu_j = \beta_0 + X_j \beta_j$
Log-logistic	$\frac{1}{1+(\lambda_j t)^{\frac{1}{\gamma}}}$	$\frac{\lambda_j^{\frac{1}{\gamma}} t^{\frac{1}{\gamma-1}}}{\gamma\left(1+(\lambda_j t)^{1/\gamma}\right)^2}$	$\lambda_j = e^{-(\beta_0 + X_j \beta_j)}$

Table 2: Functions involved in the simulations. Θ is the standard normal cumulative distribution.

variable is the time until the occurrence of an event. If the event of interest can occur only once in the same individual, we talk of standard or simple survival analysis.

In order to simulate censored survival data, two survival distributions are required, one for the uncensored survival times that would be observed if the follow-up had been sufficiently long to reach the event and another representing the censoring mechanism. The uncensored survival distribution could be generated to depend on a set of covariates with a specified relationship with survival, which represents the true prognostic importance of each covariate (Burton, Altman, Royston, and Holder 2006). **survsim** allows simulation of times using Weibull (and exponential as a particular case), log-normal and log-logistic distributions, as shown in Table 2.

In **survsim**, the first step to generate the uncensored survival distribution is to simulate a process T_i', $i = 1, \ldots, n$ subjects, based on the mentioned distributions. To induce individual heterogeneity or within-subject correlation we generate Z_i, a random effect covariate that follows a particular distribution (uniform, gamma, exponential, Weibull or inverse Gaussian).

$$t_i = t_i' \times z_i \tag{1}$$

When $z_i = 1$ for all subjects, this corresponds to the case of individual homogeneity and the survival times are completely specified by the covariates.

Random non-informative right censoring, C_i, can be generated in a similar manner to T_i', by assuming a particular distribution for the censoring times (Table 2), but without including any covariates or individual heterogeneity.

The observation times, Y_i', incorporating both events and censored observations are calculated for each case by combining the uncensored survival times, T_i, and the censoring times, C_i. If the uncensored survival time for an observation is less than or equal to the censoring time, then the event is considered to be observed and the observation time equals the uncensored survival time, otherwise the event is considered censored and the observation time equals the censoring time. In other words, once t_i and c_i have been simulated, we can define $Y_i' = \min(t_i, c_i)$ as the observation time with δ_i an indicator of non-censoring, i.e., $\delta_i = I(t_i \leq c_i)$.

While all y_i' start at 0, **survsim** allows to create dynamic cohorts. We can generate entry times higher than 0 (individuals joining the cohort after the beginning of follow-up) by adding a random value t_0 from an uniform distribution in $[0, t_{max}]$, where t_{max} is the maximum time of follow-up. We can also simulate subjects at risk prior to the start of follow-up ($y_i' = 0$), by including an uniform distribution for t_0 between $[-t_{old}, 0]$ for a fixed percentage of subjects, where t_{old} is the maximum time a subject can be at risk prior to the start of follow-up. Then:

$$y_i = y_i' + t_0, \tag{2}$$

Distribution	Time t	$\mathsf{E}[t]$	$\mathsf{VAR}[t]$
Weibull	$(-\frac{\log u}{\lambda_j})^{1/p}$	$\left(\frac{1}{\lambda_j}\right)^{1/p}\Gamma\left(1+\frac{1}{p}\right)$	$\left(\frac{1}{\lambda_j}\right)^{2/p}\left[\Gamma\left(1+\frac{2}{p}\right)-\Gamma\left(1+\frac{1}{p}\right)^2\right]$
Log-normal	$e^{\mu_j+\gamma(\log(u)-\log(1-u))}$	$e^{\mu_j+\frac{p^2}{2}}$	$\left(e^{p^2}-1\right)e^{2\mu_j+p^2}$
Log-logistic	$e^{\beta_0+X_j\beta_j+\sigma\Theta^{-1}(u)}$	$\frac{\gamma\pi/\beta_0+X_j\beta_j}{\sin(\pi/\beta_0+X_j\beta_j)}$	$\gamma^2\left(\frac{2\pi/\beta_0+\beta_j}{\sin(2\pi/\beta_0+X_j\beta_j)}-\frac{\pi^2/(\beta_0+X_j\beta_j)^2}{\sin(\pi/\beta_0+X_j\beta_j)}\right)$

Table 3: Properties of the distributions. $\mathsf{E}[t]$ is defined if $\beta_0 + X_j\beta_j > 1$. $\mathsf{VAR}[t]$ is defined if $\beta_0 + X_j\beta_j > 2$.

where $t_0 \sim U\,[0, t_{max}]$ if entry time is 0 or more and $t_0 \sim U\,[-t_{old}, 0]$ if entry time is less than 0.

Therefore, t_0 represents the initial point of the episode, y_i the endpoint and y_i' is the duration. Note that $y_i' + t_0$ can be higher than t_{max}, in which case y_i will be set at t_{max} and $\delta_i = 0$. Observations corresponding to subjects at risk prior to the start of follow-up have t_0 negative, so for these the initial point of the episode will be set at 0. y_i may also be negative, but in this case the episode will not be included in the simulated data since it will not be observed in practice.

In accordance with the parametrizations described in Table 2, the times T_i' and C_i may be generated as indicated in Table 3.

3.2. Simulating simple survival data

In a context of single-event data without left-censoring, the main function is `simple.surv.sim`, which calls the internal function `simple.ev.sim` in order to simulate the data for each individual. A call to this function might be

```
simple.surv.sim(n, foltime, dist.ev, anc.ev, beta0.ev,
  dist.cens = "weibull", anc.cens, beta0.cens, z = NA, beta = NA, x = NA)
```

The description of these arguments can be summarized as follows:

- `n`: Integer value indicating the desired size of the cohort to be simulated.
- `foltime`: Real number that indicates the maximum time of follow-up of the simulated cohort.
- `dist.ev`: Time to event distribution, with possible values `"weibull"`, for Weibull distribution, `"lnorm"` for log-normal distribution and `"llogistic"` for log-logistic distribution.
- `anc.ev`: Ancillary parameter for the time to event distribution.
- `beta0.ev`: β_0 parameter for the time to event distribution.
- `dist.cens`: String indicating the time to censoring distribution, with possible values `"weibull"` for Weibull distribution (this is the default value), `"lnorm"` for log-normal distribution and `"llogistic"` for log-logistic distribution. If no distribution is specified, the time to censoring is assumed to follow a Weibull distribution.

- `anc.cens`: Real number containing the ancillary parameter for the time to censoring distribution.
- `beta0.cens`: Real number containing the β_0 parameter for the time to censoring distribution.
- `z`: Vector with three elements that contains information relative to a random effect used in order to introduce individual heterogeneity. The first element indicates the distribution, where `"unif"` stands for a uniform distribution, `"gamma"` stands for a gamma distribution, `"exp"` stands for an exponential distribution, `"weibull"` stands for a Weibull distribution and `"invgauss"` stands for an inverse Gaussian distribution. The second and third elements indicate the minimum and maximum in the case of a uniform distribution (both must be positive) and the parameters in the case of the rest of distributions. Note that just one parameter is needed in the case of the exponential distribution. Its default value is `NA`, indicating that no individual heterogeneity is introduced.
- `beta`: List of vectors indicating the effect of the corresponding covariate. The number of vectors in `beta` must match the number of covariates, and the length of each vector must match the number of events considered. Its default value is `NA`, indicating that no covariates are included.
- `x`: List of vectors indicating the distribution and parameters of any covariate that the user needs to introduce in the simulated cohort. The possible distributions are normal distribution (`"normal"`), uniform distribution (`"unif"`) and Bernoulli distribution (`"bern"`). Its default value is `NA`, indicating that no covariates are included. The number of vectors in `x` must match the number of vectors in `beta`. Each vector in `x` must contain the name of the distribution and the parameter(s), which are: the probability of success in the case of a Bernoulli distribution, the mean and the variance in the case of a normal distribution; and the minimum and maximum in the case of a uniform distribution.

3.3. Example: First fall on residents admitted to a long-term care center

We are interested in simulating the time to first fall suffered by residents admitted to a long-term care center. Using the estimates obtained for this data by Ancizu and Navarro (2009), we generate a cohort of $n = 300$ residents, who are followed for 365 days. We consider two covariates representing the cognitive and physical impairment.

```
R> library("survsim")
R> dist.ev <- "weibull"
R> anc.ev <- 1
R> beta0.ev <- 5.268
R> dist.cens <- "weibull"
R> anc.cens <- 1
R> beta0.cens <- 5.368
R> x <- list(c("bern", 0.3), c("bern", 0.4))
R> beta <- list(-0.4, -0.25)
```

```
R> simple.dat <- simple.surv.sim(300, 365, dist.ev, anc.ev, beta0.ev,
+    dist.cens, anc.cens, beta0.cens, , beta, x)
```

As we will see in detail on Section 6, the object `simple.dat` has class '`simple.surv.sim`', an extension of class '`data.frame`', with its own `summary` method, giving relevant information about the simulated cohort:

```
R> summary(simple.dat)

Number of subjects at risk
---------------------------
  sub.risk
      300

Number of events
---------------------------
  num.events
       172

Proportion of subjects with event
---------------------------
  mean.ep.sub
   0.5733333

Total time of follow-up
---------------------------
   foltime
 28509.71

Time of follow-up (median)
---------------------------
  med.foltime
    64.68469

Density of incidence
---------------------------
   dens.incid
 0.006033032
```

4. Multiple event survival data

4.1. Description

Multiple event data occurs when each subject can have more than one event of entirely different natures (Kelly and Lim 2000). Examples of this type of events are the occurrence of tumors at different sites in the body or multiple sequalae after surgery.

id	start	stop	δ	x	k
1	0	19	0	0	1
1	0	19	0	0	2
1	0	19	0	0	3
2	0	5	1	1	1
2	0	30	0	1	2
2	0	23	1	1	3

Table 4: Occurrence of tumors.

For example, let $k = 3$ be possible adverse effects from taking some new drug in a clinical trial. We assume that the researchers define malaise ($k = 1$), vomiting ($k = 2$) and headache ($k = 3$). Table 4 represents the data structure for the events of two patients. This style of input is used to represent the counting process formulation (Aalen 1978), where each subject i is represented by k intervals: $start_{ik}$, $stop_{ik}$, δ_{ik} (non-censoring indicator), x_{ik} (covariate vector), with $(start_{ik}, stop_{ik}]$ being the interval of risk.

The first patient, with value 0 in the covariate, had no event and was censored on day 19. The second patient had malaise and headache, on days 5 and 23 respectively, and was at risk of vomiting until the end of follow-up (30 days).

Simulations can be conducted to generate multiple event data. In a similar way that we can simulate k simple independent survival distributions, we can obtain the observation time of the kth event in the ith subject, y_{ik}. The difference here is that we simulate k uncensored survival times and only one censoring time. Note that, in multiple-type events, T_{ik} and C_i are mutually independent and, furthermore, the failure in each event is independent of the others (within each subject all y_{ik} are independent for all k). In the case of multiple events, $start_{ik}$ is defined to be always equal to 0, since we assume that one is at risk of any of the events from the same instant in time and that no individuals were at risk before the beginning of follow-up. Also, $stop_{ik} = y_{ik}$.

4.2. Simulating multiple events

In a context of multiple event data, the main function is `mult.ev.sim`, which calls the internal function `mult.ev.ncens.sim` in order to simulate the data for each individual. A call to this function might be

```
mult.ev.sim(n, foltime, dist.ev, anc.ev, beta0.ev, dist.cens = "weibull",
  anc.cens, beta0.cens, z = NA, beta = NA, x = NA, priskb = 0, nsit)
```

The description of these arguments can be summarized as follows:

- `dist.ev`: Vector of arbitrary size indicating the time to event distributions, with possible values `"weibull"`, for Weibull distribution, `"lnorm"` for log-normal distribution and `"llogistic"` for log-logistic distribution.
- `anc.ev`: Vector of arbitrary size of real components containing the ancillary parameters for the time to event distributions.
- `beta0.ev`: Vector of arbitrary size of real components containing the β_0 parameters for the time to event distributions.

- `nsit`: Number of different events that a subject can suffer. It must match the number of distributions specified in `dist.ev`.

All remaining arguments are defined as specified for `simple.surv.sim`.

In order to get the function to work properly, the length of the vectors containing the parameters of the time to event and the number of distributions indicated in the argument `dist.ev` must be the same.

4.3. Example: Simulated clinical data

We continue with the clinical trial example introduced in Section 4.1. In this case it is convenient, in the simulated cohort, to restrict the number of different events an individual may suffer to 3, which we do by setting `nsit = 3`. Also, we consider that the time to right censoring follows an exponential distribution, and hence we are dealing with a special case of the Weibull distribution with a value for parameter `anc.cens` $= 1$. Imagine now that our researchers suspect that the appearance of adverse effects is related with the values of three covariates with distributions `x` $\sim N(26, 4.5)$, `x.1` $\sim$ *Unif*$(50, 75)$ and `x.2` $\sim$ *Bern*(0.25). For example, we could consider that `x` represents body mass index of the individuals, `x.1` represents age at entry to the cohort of each individual and `x.2` is a dichotomous variable indicating whether or not the subject has hypertension (we assume a prevalence of hypertension in the target population of around 25%). Notice that the parameter β corresponding to a covariate can vary depending on the event. The simulation of a cohort such as that described could be achieved by a call such as:

```
R> dist.ev <- c("weibull", "llogistic", "weibull")
R> anc.ev <- c(0.8, 0.9, 0.82)
R> beta0.ev <- c(3.56, 5.94, 5.78)
R> beta <- list(c(-0.04, -0.02, -0.01), c(-0.001, -0.0008, -0.0005),
+    c(-0.7, -0.2, -0.1))
R> x <- list(c("normal", 26, 4.5), c("unif", 50, 75), c("bern", 0.25))
R> clinical.data <- mult.ev.sim(n = 100, foltime = 30, dist.ev, anc.ev,
+    beta0.ev, dist.cens = "weibull", anc.cens = 1, beta0.cens = 5.2,
+    z = c("unif", 0.6, 1.4), beta, x, nsit = 3)
```

With this command we are generating a cohort of 100 individuals followed for a maximum of 30 days. For the first event the aim is to generate times following a Weibull distribution with decreasing hazard (`anc.ev` < 1) and an expected value of around 40 days (note that in accordance with Table 3, specifying parameter `anc.ev` and the expected value of the distribution results in determination of the other parameter, in this case we have `beta0.cens` $= 3.56$). The `beta` parameter contains the coefficients of each one of the covariates BMI, age and hypertension, respectively, for each of the events. The output data is a data frame containing the episodes for each of the 100 simulated subjects.

```
R> head(round(clinical.data, 2))
```

```
  nid ev.num  time status start  stop    z     x   x.1 x.2
1   1      1 13.39      1     0 13.39 0.86 28.37 71.88   0
2   1      2 30.00      0     0 30.00 0.86 28.37 71.88   0
3   1      3 30.00      0     0 30.00 0.86 28.37 71.88   0
4   2      1  0.76      1     0  0.76 0.61 27.19 52.77   0
5   2      2 30.00      0     0 30.00 0.61 27.19 52.77   0
6   2      3 30.00      0     0 30.00 0.61 27.19 52.77   0
```

A summary of the most relevant information about the simulated cohort may be obtained by calling the `summary` function with an object of class '`sim.mult.ev.data`' as argument, including the number of subjects at risk, the total number of events observed, the total time of follow-up, its median and the incidence density:

```
R> summary(clinical.data)

Number of subjects at risk
----------------------------
  ev.num sub.risk
      1      100
      2      100
      3      100


Number of events
----------------------------
  ev.num num.events
      1         86
      2         13
      3         16


Total time of follow-up
----------------------------
  ev.num   foltime
      1  876.7141
      2 2504.7251
      3 2377.7440


Time of follow-up (median)
----------------------------
  ev.num med.foltime
      1    5.248077
      2   30.000000
      3   30.000000
```

```
Density of incidence
----------------------------
  ev.num  dens.incid
       1 0.098093547
       2 0.005190190
       3 0.006729068
```

The result is a summary of information for each of the possible events. For example, 86 subjects had the first event, 13 had the second and 16 had the third. The median follow-up time to the first event was 5.25 and the total follow-up time to the first event was 876.71.

5. Recurrent event survival data

5.1. Description

Recurrent event data is a type of multiple event where the subject can experience repeated occurrences of the same type (Kelly and Lim 2000), for example repeated asthma attacks or sick leave episodes. In practice, the hazard of a recurrent event can vary depending on the number of previous occurrences, in terms of shape and intensity (Reis, Utzet, Rocca, Nedel, Martín, and Navarro 2011; Navarro, Moriña, Reis, Nedel, Martín, and Alvarado 2012). However, simulations based on a mixture of distributions with different baseline hazard rates are quite rare (Bender, Augustin, and Blettner 2005; Metcalfe and Thompson 2006). In a recurrent data context, each subject can present different number of episodes. We talk of episodes (or occurrences) rather than events since each occurrence is a new episode of the same event.

Consider Table 5, which shows an example of information about three workers in a company. We are interested in studying the possible association between a covariate x and the hazard of having a sick leave episode due to respiratory problems ($\delta = 1$). In this case the individuals may have different numbers of intervals since they may have different numbers of sick leave episodes. Imagine also that x can vary with time (for example x could be a dichotomous covariate indicating whether the worker is exposed to carrying heavy loads, which obviously may vary). Finally, the sick leave episodes may involve a period of time during which the individual ceases to be a risk, corresponding to the time they are off work due to illness. The time periods are expressed as semiclosed intervals (start2, stop2]. Thus, in Table 5 it may be seen that the first worker has a single sick leave episode, starting on day 1027. As a result this worker ceases to be at risk for five days. Subsequently, from day 1459 we find this worker is exposed to heavy loads (x switches from 0 to 1) and is followed until they are censored at the end of the follow-up period on day 1740. Note that in this case k represents the episode which individual i is at risk of suffering in each interval. For example, the second worker only begins to be at risk of their third sick leave episode on day 598 and they remain so until it occurs, on day 712. This worker changes to another company on day 827 and is therefore censored. If the study includes subjects at risk prior to the start of the follow-up period, we may encounter individuals whose first episodes start at times greater than zero. In the sick leave example this could happen if the origin of the time scale was the date the worker was hired. This is the case of the third worker, for whom follow-up only began after they had been in the job for six months (i.e., had accumulated six months of exposure by the time they

id	start2	stop2	δ	x	k
1	0	1027	1	0	1
1	1032	1458	0	0	2
1	1458	1740	0	1	3
2	0	328	1	1	1
2	341	591	1	1	2
2	597	712	1	1	3
2	758	827	0	1	4
3	180	715	1	0	1
3	723	1029	0	0	2

Table 5: Example of data entered for three individuals in a study of sick leave episodes.

were first observed). Note that in this case k might not correspond to the true number of episodes suffered by the individual, in fact this value is usually unknown.

To generate recurrent event survival data **survsim** assumes that there is a different and independent distribution Y_k depending on k, the number of episodes at risk. The simulating process for each Y_k is the same as that described in Section 4.1, but in this case, obviously, a subject cannot to be at risk for the kth episode if he/she had not had the $(k-1)$th.

Then,

- Y_1 is simulated specifying T_1 and C_1, for all subjects: $start2_{ik} = 0$ and $stop2_{ik} = y_{ik}$.
- For $k > 1$, Y_k is simulated specifying T_k and C_k, only for subjects with $\delta = 1$ in Y_{k-1}: $start2_{ik} = stop2_{ik-1} + long_{ik-1}$ ($long_{ik-1}$ is the duration of the previous episode and $stop2_{ik} = start2_{ik} + y_{ik}$. If $stop2_{ik} > t_{max}$, then $stop2_{ik}$ will be set to t_{max} and $\delta_i = 0$, where t_{max} is the maximum follow-up time).

5.2. Simulating recurrent events

The main function in the case of recurrent event data is `rec.ev.sim`, which calls the routines `rec.ev.ncens` and `rec.ev.cens` in order to produce the simulation for the individuals which were not at risk before the start of follow-up and for the individuals that were at risk before the start of follow-up, respectively, according to the parameters used as arguments in a call to the function, which are:

```
rec.ev.sim(n, foltime, dist.ev, anc.ev, beta0.ev,
  dist.cens = rep("weibull", length(beta0.cens)), anc.cens, beta0.cens,
  z = NA, beta = 0, x = NA, lambda = NA, max.ep = Inf, priskb = 0,
  max.old = 0)
```

The description of these arguments can be summarized as follows:

- `beta0.ev`: Vector of arbitrary size of real components containing the β_0 parameters for the time to event distributions. If a subject suffers more episodes than specified distributions, the last β_0 parameter specified here is used to generate times corresponding to later episodes.

- `dist.cens`: Vector of arbitrary size indicating the time to censure distributions, with possible values `"weibull"` for Weibull distribution (the default value), `"lnorm"` for log-normal distribution and `"llogistic"` for log-logistic distribution. If no distribution is specified, the time to censoring is assumed to follow as many different Weibull distributions as different parameters are introduced in `beta0.cens`.

- `anc.cens`: Vector of arbitrary size of real components containing the ancillary parameters for the time to censoring distributions. If a subject suffers more episodes than specified distributions, the last ancillary parameter specified here is used to generate times corresponding to later episodes.

- `beta0.cens`: Vector of arbitrary size of real components containing the β_0 parameters for the time to censoring distributions. If a subject suffers more episodes than specified distributions, the last β_0 parameter specified here is used to generate times corresponding to later episodes.

- `lambda`: Real number indicating the mean duration of each event or discontinuous risk time, assumed to follow a zero-truncated Poisson distribution. Its default value is `NA`, corresponding to the case where the duration of each event or discontinuous risk time is unnecessary information for the user.

- `max.ep`: Integer value that matches the maximum permitted number of episodes per subject. Its default value is `Inf`, i.e., the number of episodes per subject is not limited.

- `priskb`: Proportion of subjects at risk prior to the start of follow-up, defaults to 0.

- `max.old`: Maximum time at risk prior to the start of follow-up.

All remaining arguments are defined as specified for `simple.surv.sim` and for `mult.ev.sim`.

In order to get the function to work properly, the length of the vectors containing the parameters of the time to event and time to censoring distributions and the number of distributions indicated in the parameter `dist.ev` must be the same. Finally, `priskb` and `max.old` must be positive numbers, with `priskb` being between 0 and 1. Notice that large values of `max.old` can result in the routine taking a long time to simulate the cohort with the specified size.

The parameters for the simulation of cohorts in a context of recurrent events are very similar to those described for the case of multiple events. A remarkable difference is that in the latter situation, there is no interest in the duration of each event, and therefore there is no `lambda` parameter, and also the addition of the parameter `nsit`, that indicates the number of different situations that a subject can suffer.

5.3. Example: Recurrent sick leave episodes

Navarro *et al.* (2012) and Reis *et al.* (2011) analyze sick leave episodes occurring in a cohort of workers in the Hospital das Clínicas da Universidade Federal de Minas Gerais, Brasil, with a follow-up time of 10 years, in terms of a variety of diagnoses, including musculoskeletal diseases, diseases of the respiratory system, mental and behavioral disorders and all causes. Here we will focus on sick leave episodes due to respiratory diseases, as in the example in Section 5.1 The methods used to obtain the distributions which best fit the data are detailed

in Navarro *et al.* (2012), and the results for the case we are interested in may also be found there.

The authors assume that the distribution of new episodes after the fourth should be very similar to the fourth one, so four distributions and eight parameters are taken into account, and therefore a call to `rec.ev.sim` could take the form:

```
R> dist.ev <- c("lnorm", "llogistic", "weibull", "weibull")
R> anc.ev <- c(1.498, 0.924, 0.923, 1.051)
R> beta0.ev <- c(7.195, 6.583, 6.678, 6.430)
R> anc.cens <- c(1.272, 1.218, 1.341, 1.484)
R> beta0.cens <- c(7.315, 6.975, 6.712, 6.399)
R> beta <- list(c(-0.4, -0.5, -0.6, -0.7))
R> lambda <- c(2.18, 2.33, 2.40, 3.46)
R> sim.data <- rec.ev.sim(n = 500, foltime = 3600, dist.ev,
+    anc.ev, beta0.ev, , anc.cens, beta0.cens, z = c("unif", 0.8, 1.2),
+    beta, x = list(c("bern", 0.5)), lambda, priskb = 0.5, max.old = 730)
```

where we are simulating a cohort of 500 subjects with a follow-up time of 3600 days, a mean duration of $\lambda_1 = 2.18$, $\lambda_2 = 2.33$ and $\lambda_3 = 2.40$ days for each event from first to third, and a mean duration of $\lambda_k = 3.46, k \geq 4$ for successive episodes, a β_1 of -0.4 – representing a reduction of approximately 33% in the acceleration factor or time ratio (TR) –, decreasing on each recurrence and an heterogeneity random parameter following a uniform distribution in the interval $[0.8, 1.2]$. We are also assuming that in our simulated cohort 50% of the individuals have been at risk for 730 days or less prior to the start of follow-up, i.e., 50% of the subjects began working before the start of follow-up, a maximum of 730 days before the start of follow-up time. This is a common situation in this context, where some of the workers may had sick leaves before the start of follow-up time.

The output data is a data frame containing the episodes for each of the 500 simulated subjects.

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

```
  nid real.episode obs.episode       time status    start      stop
1   1            1           1  985.99220      1   0.0000  985.9922
2   1            2           2  357.15540      0 986.9922 1344.1476
3   2            1           1  149.87287      1   0.0000  149.8729
4   2            2           2   74.33743      0 150.8729  225.2103
5   3            1           1 1025.00137      0   0.0000 1025.0014
6   4            1           1 1211.06578      0   0.0000 1211.0658
       time2    start2     stop2 old risk.bef long         z x
1  985.99220 2524.1993 3510.1915  NA    FALSE    1 0.9211748 0
2   88.80854 3511.1915 3600.0000  NA    FALSE   NA 0.9211748 0
3  149.87287  846.0055  995.8783  NA    FALSE    1 1.1405794 1
4   74.33743  996.8783 1071.2158  NA    FALSE   NA 1.1405794 1
5  449.42472 3150.5753 3600.0000  NA    FALSE   NA 1.1656982 0
6 1211.06578 2118.0060 3329.0718  NA    FALSE   NA 1.0894661 1
```

The first part of the output data consists of those individuals who were not at risk prior to the start of follow-up, of whom there are $(1-$`priskb`$) \cdot$ `n`. In the example we are using,

individuals 1 to 250 who meet this condition, and the rest, individuals 251 to 500 were at risk prior to the start of follow-up. In order to see the first individual who was at risk prior to follow-up we can do the following:

```
R> sim.data[sim.data$nid == 251, ]
```

```
    nid real.episode obs.episode      time status     start      stop
466 251            1           1 426.08792      1    0.0000  426.0879
467 251            2           2 451.80922      1  428.0879  879.8971
468 251            3           3 301.17946      1  880.8971 1182.0766
469 251            4           4  58.61317      0 1184.0766 1242.6898
        time2   start2     stop2       old risk.bef long         z x
466 221.26152   0.0000  221.2615 -204.8264     TRUE    2 0.8023607 1
467 451.80922 223.2615  675.0707 -204.8264     TRUE    1 0.8023607 1
468 301.17946 676.0707  977.2502 -204.8264     TRUE    2 0.8023607 1
469  58.61317 979.2502 1037.8634 -204.8264     TRUE   NA 0.8023607 1
```

Again, a brief summary of the most relevant information about the simulated cohort may be obtained from:

```
R> summary(sim.data)
```

```
Number of subjects at risk
-------------------------------------
        ep.num sub.risk
            1      500
            2      208
            3      112
            4       66
          > 4       45
 All episodes      500


Number of events
-------------------------------------
        ep.num num.events
            1        208
            2        112
            3         66
            4         45
          > 4        108
 All episodes        539


Total time of follow-up
-------------------------------------
        ep.num   foltime
```

```
            1 339014.66
            2  83889.75
            3  37693.08
            4  17090.65
          > 4  31739.59
 All episodes 509427.73


Time of follow-up (median)
-------------------------------------
        ep.num med.foltime
            1    482.6887
            2    274.4953
            3    260.8960
            4    198.2772
          > 4    159.4306
 All episodes    301.1795


Density of incidence
-------------------------------------
        ep.num   dens.incid
            1 0.0006135428
            2 0.0013350856
            3 0.0017509845
            4 0.0026330189
          > 4 0.0034026903
 All episodes 0.0010580500
```

In this case we obtain a global summary (indicated as `All episodes`) and a summary stratified by episode number, those following the same distribution on the basis of a given occurrence number being grouped together (in the example four different distributions had been specified for the events, so that the results appear for the first four episodes and those for the fifth and subsequent ones are grouped together). Notice that if we are interested in generating a single-event cohort (see Example 3.3) but allowing for left-censoring, we can also use `rec.ev.sim` with `max.ep = 1`.

6. Simulated data structure

Output data is structured as an object of class of '`simple.surv.sim`', '`mult.ev.data.sim`' or '`rec.ev.data.sim`', depending on the nature of the simulated data. This class is just an extension of the '`data.frame`' class. Each row in the data frame is a new episode for the corresponding subject. The resulting data set after a call to `rec.ev.sim` has the following columns:

- `nid`: An integer number that identifies the subject.

- `real.episode`: Number of the episode corresponding to the real history of the individual.
- `obs.episode`: Number of the episode corresponding to the follow-up time.
- `time`: Time until the corresponding event happens (or time to subject drop-out), with respect to the beginning of the follow-up.
- `status`: Logical value indicating if the episode corresponds to an event (`status` = 1) or a drop-out (`status` = 0).
- `start`: Time at which an episode starts, taking the beginning of follow-up as the origin of the time scale.
- `stop`: Time at which an episode ends, taking the beginning of follow-up as the origin of the time scale.
- `time2`: Time until the corresponding event happens (or time to subject drop-out), in calendar time.
- `start2`: Time at which an episode starts, where the time scale is calendar time.
- `stop2`: Time at which an episode ends, where the time scale is calendar time.
- `old`: Real value indicating the time span that the individual was at risk before the beginning of follow-up.
- `risk.bef`: Factor that indicates if an individual was at risk before the beginning of follow-up or not.
- `long`: Time not at risk immediately after an episode.
- `z`: Individual heterogeneity, generated according to the specified distribution.
- `x`, `x.1`, ...: Value of each covariate randomly generated for each subject in the cohort.

This structure defines the class '`rec.ev.data.sim`'. The output of function `mult.ev.sim`, corresponding to multiple events context looks very similar, and is structured in the same way. The structure of '`simple.surv.sim`' objects, corresponding to single-event survival data, is structured in a very similar, simpler way. Although these classes are essentially data frames, they have a proper `summary` method, described in previous sections.

7. Aggregated data

Aggregated data is a classical representation of count data type, where for each subject we know the total number of episodes and the accumulated follow-up. This data is often analyzed by means of Poisson or negative Binomial regression.

If a user is interested in the aggregated instead of individual data, we provide the function `accum`, which works in an analogous manner in either of the two defined contexts. The output data after a call to `accum` contains the following columns with the corresponding aggregated values:

- `nid`: An integer number that identifies the subject.
- `old`: Real value indicating the time that the individual was at risk before the beginning of the follow-up.
- `risk.bef`: Logical value indicating if the subject was at risk before the beginning of the follow-up time or not.
- `z`: Individual heterogeneity, generated according to the specified distribution.
- `x`, `x.1`, ...: Value of each covariate randomly generated for each subject in the cohort.
- `obs.ep.accum`: Aggregated number of episodes suffered by an individual since the beginning of subject's follow-up time.
- `real.ep.accum`: Aggregated number of episodes suffered by an individual since the beginning of subject's history.
- `time.accum`: Global time of follow-up for each individual.
- `long.accum`: Global time not at risk within the follow-up time, corresponding to the sum of times between the end of an event and the beginning of the next.

Notice that aggregating data in a multiple events context may be confusing.

7.1. Example: Sick leave counts

If we want to work with aggregated instead of individual data, we can easily generate a data entry of the relevant aggregated data. This structure is indicated in the common case in which we were interested in the simulation of a study where only the total number of episodes and follow-up are known for each subject. In this case the data could be analyzed by counts modeling. For example, data corresponding to the sick leave introduced in Section 5.3 can be aggregated by typing

```
R> agg.data <- accum(sim.data)
```

The output data is a data frame with 500 elements containing one row per subject corresponding to the aggregated data.

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

```
  nid old risk.bef         z x obs.ep.accum real.ep.accum time.accum
1   1  NA    FALSE 0.9211748 0            1             1 1074.80075
2   2  NA    FALSE 1.1405794 1            1             1  224.21030
3   3  NA    FALSE 1.1656982 0            0             0  449.42472
4   4  NA    FALSE 1.0894661 1            0             0 1211.06578
5   5  NA    FALSE 1.0253948 0            0             0   36.58911
6   6  NA    FALSE 1.1424034 1            0             0  155.85532
  long.accum
1          1
```

```
2          1
3          0
4          0
5          0
6          0
```

8. Application

Cohorts simulated by **survsim** can be used in a direct way in the standard R survival functions. For example, a simple Cox regression on the data frame generated in Section 3 can be fitted by means of

```
R> library("survival")
R> single <- coxph(Surv(start, stop, status) ~ as.factor(x) + as.factor(x.1),
+    data = simple.dat)
```

A multiple event Cox regression can be fitted to the cohort generated in Section 4 by

```
R> multiple <- coxph(Surv(start, stop, status) ~ strata(ev.num) / (x + x.1 +
+    as.factor(x.2)) + cluster(nid), data = clinical.data)
```

An Andersen-Gill Cox regression model can be fitted to the cohort generated in Section 5, in a recurrent events context, by means of

```
R> AG.event <- coxph(Surv(start2, stop2, status) ~ as.factor(x) +
+    cluster(nid), data = sim.data)
```

For aggregated data, a negative Binomial regression model can be fitted using function `glm.nb` from package **MASS**. For exemple, we can use this over data frame `agg.data`, generated in Section 7.

```
R> library("MASS")
R> nb <- glm.nb(obs.ep.accum ~ as.factor(x) + offset(log(time.accum)),
+    data = agg.data)
```

9. Conclusions and further work

In this article we present a set of functions which allow the user to simulate a cohort with the objective of studying its behavior in a context of simple, recurrent or multiple events which can be generated by a variety of different processes, following different distributions. Among the improvements planned for the package **survsim**, we would like to mention the inclusion of new distributions for events and for censoring, the consideration of other contexts within the broader area of complex survival data such as competitive risks, and the inclusion of categorical covariates with more than two categories.

Acknowledgments

This study was partially funded by the Fondo de Investigación Sanitaria, Instituto de Salud Carlos III, Ministerio de Ciencia e Innovación, Gobierno de España (project PI080703).

References

Aalen OO (1978). "Nonparametric Inference for a Family of Counting Processes." *The Annals of Statistics*, **6**(4), 701–726.

Abrahamowicz M, MacKenzie T, Esdaile JM (1996). "Time-Dependent Hazard Ratio: Modelling and Hypothesis Testing with Application in Lupus Nephritis." *Journal of the American Statistical Association*, **91**(436), 1432–1439.

Ancizu I, Navarro A (2009). "Analyzing the Occurrence of Falls and its Risk Factors: Some Considerations." *Preventive Medicine*, **48**(3), 298–302.

Bender R, Augustin T, Blettner M (2005). "Generating Survival Times to Simulate Cox Proportional Hazards Models." *Statistics in Medicine*, **24**(11), 1713–1723.

Broström G (2012). *Event History Analysis with R*. The R Series. CRC Press.

Broström G (2014). ***eha**: Event History Analysis*. R package version 2.4-1, URL http://CRAN.R-project.org/package=eha.

Burton A, Altman DG, Royston P, Holder RL (2006). "The Design of Simulation Studies in Medical Statistics." *Statistics in Medicine*, **25**(24), 4279–4292.

Crowther MJ (2011). ***SURVSIM**: Stata Module to Simulate Complex Survival Data*. Statistical Software Components, URL http://ideas.RePEc.org/c/boc/bocode/s457317.html.

Kelly PJ, Lim LL (2000). "Survival Analysis for Recurrent Event Data: An Application to Childhood Infectious Diseases." *Statistics in Medicine*, **19**(1), 13–33.

Metcalfe C, Thompson SG (2006). "The Importance of Varying the Event Generation Process in Simulation Studies of Statistical Methods for Recurrent Events." *Statistics in Medicine*, **25**(1), 165–179.

Moriña D, Navarro A (2014). ***survsim**: Simulation of Simple and Complex Survival Data*. R package version 1.1.2, URL http://CRAN.R-project.org/package=survsim.

Navarro A, Moriña D, Reis R, Nedel FB, Martín M, Alvarado S (2012). "Hazard Functions to Describe Patterns of New and Recurrent Sick Leave Episodes for Different Diagnoses." *Scandinavian Journal of Work, Environment and Health*, **38**(5), 447–455.

R Core Team (2014). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Reis RJ, Utzet M, Rocca PFL, Nedel FB, Martín M, Navarro A (2011). "Previous Sick Leaves as Predictor of Subsequent Ones." *International Archives of Occupational and Environmental Health*, **84**(5), 491–499.

StataCorp (2013). *Stata Data Analysis Statistical Software: Release 12*. StataCorp LP, College Station, TX. URL http://www.stata.com/.

Sylvestre MP, Evans T, MacKenzie T, Abrahamowicz M (2013). ***PermAlgo**: Permutational Algorithm to Generate Event Times Conditional on a Covariate Matrix including Time-Dependent Covariates*. R package version 1.0, URL http://CRAN.R-project.org/package=PermAlgo.

Therneau TM (2014). ***survival**: A Package for Survival Analysis in S*. R package version 2.37-7, URL http://CRAN.R-project.org/package=survival.

Therneau TM, Grambsch PM (2000). *Modeling Survival Data: Extending the Cox Model*. Springer-Verlag.

Venables WN, Ripley BD (2002). *Modern Applied Statistics with S*. 4th edition. Springer-Verlag, New York.

Affiliation:

David Moriña
Centre for Research in Environmental Epidemiology (CREAL),
CIBER Epidemiología y Salud Pública (CIBERESP),
Barcelona Biomedical Research Park (PRBB),
Doctor Aiguader, 88, 08003 Barcelona, Spain
E-mail: dmorina@creal.cat

Albert Navarro
Grups de Recerca d'Àfrica i Amèrica Llatines (GRAAL),
Unitat de Bioestadística, Facultat de Medicina
Universitat Autònoma de Barcelona
08193 Cerdanyola del Vallès, Spain
E-mail: albert.navarro@uab.cat

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 59, Issue 2 *Submitted:* 2012-07-13
July 2014 *Accepted:* 2014-03-20