Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. doi: 10.18637/jss.v000.i00

casebase: An Alternative Framework For Survival Analysis

Sahir Bhatnagar *
McGill University

Maxime Turgeon *
University of Manitoba

Jesse Islam
McGill University

James Hanley
McGill University

Olli Saarela
University of Toronto

Abstract

The abstract of the article. * joint co-authors

Keywords: keywords, not capitalized, Java.

1. Introduction

The purpose of the **casebase** package is to provide practitioners with an easy-to-use software tool to predict the risk (or cumulative incidence (CI)) of an event, for a particular patient. The following points should be noted:

1. Time matching/risk set sampling (including Cox partial likelihood) eliminates the baseline hazard from the likelihood expression for the hazard ratios
2. If, however, the absolute risks are of interest, they have to be recovered using the semi-parametric Breslow estimator
3. Alternative approaches for fitting flexible hazard models for estimating absolute risks, not requiring this two-step approach? Yes! (Hanley and Miettinen 2009)

(Hanley and Miettinen 2009) propose a fully parametric hazard model that can be fit via logistic regression. From the fitted hazard function, cumulative incidence and, thus, risk functions of time, treatment and profile can be easily derived.

	Parameters (location in red)	Density R function	`dist`
Exponential	`rate`	`dexp`	`"exp"`
Weibull (accelerated failure time)	`shape, scale`	`dweibull`	`"weibull"`
Weibull (proportional hazards)	`shape, scale`	`dweibullPH`	`"weibullPH"`
Gamma	`shape, rate`	`dgamma`	`"gamma"`
Log-normal	`meanlog, sdlog`	`dlnorm`	`"lnorm"`
Gompertz	`shape, rate`	`dgompertz`	`"gompertz"`
Log-logistic	`shape, scale`	`dllogis`	`"llogis"`
Generalized gamma (Prentice 1975)	`mu, sigma, Q`	`dgengamma`	`"gengamma"`
Generalized gamma (Stacy 1962)	`shape, scale, k`	`dgengamma.orig`	`"gengamma.orig"`
Generalized F (stable)	`mu, sigma, Q, P`	`dgenf`	`"genf"`
Generalized F (original)	`mu, sigma, s1, s2`	`dgenf.orig`	`"genf.orig"`

Table 1: Built-in parametric survival distributions in **flexsurv**.

2. Theoretical details

As discussed in Hanley & Miettinen (2009), the key idea behind case-base sampling is to discretize the study base into an infinite amount of *person moments*. These person moments are indexed by both an individual in the study and a time point, and therefore each person moment has a covariate profile, an exposure status and an outcome status attached to it. We note that there is only a finite number of person moments associated with the event of interest (what Hanley & Miettinen call the *case series*). The case-base sampling refers to the sampling from the base of a representative finite sample called the *base series*.

As shown by Saarela & Arjas (2015) (and further expanded in Saarela (2016)), writing the likelihood arising from this data-generating mechanism using the framework of non-homogeneous Poisson processes, we eventually reach an expression where each person-moment's contribution is of the form

$$\frac{h(t)^{dN(t)}}{\rho(t) + h(t)},$$

where $N(t)$ is the counting process associated with the event of interest, $h(t)$ is the corresponding hazard function, and $\rho(t)$ is the hazard function for the Poisson process associated with case-base sampling. This parametric form suggests that we can readily estimate log-hazards of the form $\log(h(t)) = g(t; X)$ using logistic regression, where each observation corresponds to a person moment, the function $g(t; X)$ is linear in a finite number of parameters, and where we treat $-\log(\rho(t))$ as an offset.

In Hanley & Miettinen (2009), the authors suggest performing case-base sampling *uniformly*, i.e. to sample the base series uniformly from the study base. In terms of Poisson processes, this sampling strategy corresponds essentially to a time-homogeneous Poisson process with intensity equal to b/B, where b is the number of sampled observations in the base series, and B is the total population-time for the study base. More complex examples are also available;

see for example Saarela & Arjas (2015), where the probabilities of the sampling mechanism are proportional to the cardiovascular disease event rate given by the Framingham score.

The `casebase` package fits the family of hazard functions of the form

$$h(t; X) = \exp[g(t; X)]$$

where t denotes time and X, the individual's covariate profile. Different functions of t lead to different parametric hazard models. The simplest of these models is the one-parameter exponential distribution which is obtained by taking the hazard function to be constant over the range of t.

$$h(t; X) = \exp(\beta_0 + \beta_1 X)$$

The instantaneous failure rate is independent of t, so that the conditional chance of failure in a time interval of specified length is the same regardless of how long the individual has been in the study. This is also known as the *memoryless property* (Kalbfleisch and Prentice, 2002).

The Gompertz hazard model is given by including a linear term for time:

$$h(t; X) = \exp(\beta_0 + \beta_1 t + \beta_2 X)$$

Use of $\log(t)$ yields the Weibull hazard which allows for a power dependence of the hazard on time (Kalbfleisch and Prentice, 2002):

$$h(t; X) = \exp(\beta_0 + \beta_1 \log(t) + \beta_2 X)$$

For competing-risk analyses with J possible events, we can show that each person-moment's contribution of the likelihood is of the form

$$\frac{h_j(t)^{dN_j(t)}}{\rho(t) + \sum_{j=1}^{J} h_j(t)},$$

where $N_j(t)$ is the counting process associated with the event of type j and $h_j(t)$ is the corresponding hazard function. As may be expected, this functional form is similar to the terms appearing in the likelihood function for multinomial regression.[1]

3. Implementation details

The functions in the casebase package can be divided into two categories: 1) data visualization, in the form of population-time plots; and 2) parametric modeling. We explicitly aimed at

[1]Specifically, it corresponds to the following parametrization:

$$\log\left(\frac{P(Y = j \mid X)}{P(Y = J \mid X)}\right) = X^T \beta_j, \qquad j = 1, \ldots, J-1$$

being compatible with both `data.frame`s and `data.table`s. This is evident in some of the coding choices we made, and it is also reflected in our unit tests.

3.1. Population-time plots

3.2. Parametric modeling

The parametric modeling step was separated into three parts:

1. case-base sampling;
2. estimation of the smooth hazard function;
3. calculation of the risk function.

By separating the sampling and estimation functions, we allowed the possibility of users implementing more complex sampling scheme, as described in Saarela (2016).

The sampling scheme selected for `sampleCaseBase` was described in Hanley and Miettinen (2009): we first sample along the "person" axis, proportional to each individual's total follow-up time, and then we sample a moment uniformly over their follow-up time. This sampling scheme is equivalent to the following picture: imagine representing the total follow-up time of all individuals in the study along a single dimension, where the follow-up time of the next individual would start exactly when the follow-up time of the previous individual ends. Then the base series could be sampled uniformly from this one-dimensional representation of the overall follow-up time. In any case, the output is a dataset of the same class as the input, where each row corresponds to a person-moment. The covariate profile for each such person-moment is retained, and an offset term is added to the dataset. This output could then be used to fit a smooth hazard function, or for visualization of the base series.

The fitting function `fitSmoothHazard` starts by looking at the class of the dataset: if it was generated from `sampleCaseBase`, it automatically inherited the class `cbData`. If the dataset supplied to `fitSmoothHazard` does not inherit from `cbData`, then the fitting function starts by calling `sampleCaseBase` to generate the base series. In other words, the occasional user can bypass `sampleCaseBase` altogether and only worry about the fitting function `fitSmoothHazard`.

The fitting function retains the familiar formula interface of `glm`. The left-hand side of the formula should be the name of the column corresponding to the event type. The right-hand side can be any combination of the covariates, along with an explicit functional form for the time variable. Note that non-proportional hazard models can be achieved at this stage by adding an interaction term involving time. The offset term does not need to be specified by the user, as it is automatically added to the formula.

To fit the hazard function, we provide several approaches that are available via the `family` parameter. These approaches are:

- `glm`: This is the familiar logistic regression.
- `glmnet`: This option allows for variable selection using Lasso or elastic-net. This functionality is provided through the `glmnet` package (Friedman, Hastie, and Tibshirani 2010).

- `gam`: This option provides support for *Generalized Additive Models* via the `gam` package (Hastie and Tibshirani 1987).
- `gbm`: This option provides support for *Gradient Boosted Trees* via the `gbm` package. This feature is still experimental.

In the case of multiple events, the hazard is fitted via multinomial regression as performed by the **VGAM** package. This package was selected for its ability to fit multinomial regression models with an offset.

Once a model-fit object has been returned by `fitSmoothHazard`, all the familiar summary and diagnostic functions are available: `print`, `summary`, `predict`, `plot`, etc. Our package provides one more functionality: it computes risk functions from the model fit. For the case of a single event, it uses the familiar identity

$$S(t) = \exp\left(-\int_0^t h(u;X)du\right).$$

The integral is computed using either the `stats::integrate` function or Monte-Carlo integration. The risk function (or cumulative incidence function) is then defined as

$$CI(t) = 1 - S(t).$$

For the case of a competing-event analysis, the event-specific risk is computed using the following procedure: first, we compute the overall survival function (i.e. for all event types):

$$S(t) = \exp\left(-\int_0^t H(u;X)du\right), \qquad H(t;X) = \sum_{j=1}^{J} h_j(t;X).$$

From this, we can derive the event-specific subdensities:

$$f_j(t) = h_j(t)S(t).$$

Finally, by integrating these subdensities, we obtain the event-specific cumulative incidence functions:

$$CI_j(t) = \int_0^t f_j(u)du.$$

We created `absoluteRisk` as an `S3` generic, with methods for the different types of outputs of `fitSmoothHazard`. The method dispatch system of R then takes care of matching the correct output to the correct methodology for calculating the cumulative incidence function, without the user's intervention.

In the following sections, we illustrate these functionalities in the context of three case studies.

4. Case study 1–European Randomized Study of Prostate Cancer Screening

To introduce the different features available, we make use of the European Randomized Study of Prostate Cancer Screening data; this dataset is available through the **casebase** package:

```
R> data(ERSPC)
R> ERSPC$ScrArm <- factor(ERSPC$ScrArm,
R>                        levels = c(0,1),
R>                        labels = c("Control group", "Screening group"))
```

The results of this study were published by [schroder2009screening]. This data was obtained using the approach described in [liu2014recovering].

Population time plots can be extremely informative graphical displays of survival data. They should be the first step in an exploratory data analysis. We facilitate this task in the **casebase** package using the `popTime` function. We first create the necessary dataset for producing the population time plots, and we can produce the plot by using the corresponding `plot` method:

```
R> pt_object <- casebase::popTime(ERSPC, event = "DeadOfPrCa")
R> plot(pt_object)
```

We can also create exposure stratified plots by specifying the `exposure` argument in the `popTime` function:

```
R> pt_object_strat <- casebase::popTime(ERSPC,
R>                                      event = "DeadOfPrCa",
R>                                      exposure = "ScrArm")
R> plot(pt_object_strat)
```

We can also plot them side-by-side using the `ncol` argument:

```
R> plot(pt_object_strat, ncol = 2)
```

First, we fit a Cox model to the data, examine the hazard ratio for the screening group (relative to the control group), and plot the cumulative incidence function (CIF).

```
R> cox_model <- survival::coxph(Surv(Follow.Up.Time, DeadOfPrCa) ~ ScrArm,
R>                              data = ERSPC)
R> summary(cox_model)
```

We can plot the CIF for each group:

```
R> new_data <- data.frame(ScrArm = c("Control group", "Screening group"),
R>                        ignore = 99)
R>
R> plot(survfit(cox_model, newdata = new_data),
R>      xlab = "Years since Randomization",
R>      ylab = "Cumulative Incidence",
R>      fun = "event",
R>      xlim = c(0,15), conf.int = FALSE, col = c("red","blue"),
R>      main = sprintf("Estimated Cumulative Incidence (risk) of Death from Prostate
R>                     Cancer Screening group Hazard Ratio: %.2g (%.2g, %.2g)",
```

```
R>                     exp(coef(cox_model)),
R>                     exp(confint(cox_model))[1],
R>                     exp(confint(cox_model))[2]))
R> legend("topleft",
R>        legend = c("Control group", "Screening group"),
R>        col = c("red","blue"),
R>        lty = c(1, 1),
R>        bg = "gray90")
```

Next we fit several models using case-base sampling. The models we fit differ in how we choose to model time.

The `fitSmoothHazard` function provides an estimate of the hazard function $h(x, t)$ is the hazard function, where t denotes the numerical value of a point in prognostic/prospective time and x is the realization of the vector X of variates based on the patient's profile and intervention (if any).

```
R> casebase_exponential <- casebase::fitSmoothHazard(DeadOfPrCa ~ ScrArm,
R>                                                   data = ERSPC,
R>                                                   ratio = 100)
R>
R> summary(casebase_exponential)
R> exp(coef(casebase_exponential)[2])
R> exp(confint(casebase_exponential)[2,])
```

The `absoluteRisk` function provides an estimate of the cumulative incidence curves for a specific risk profile using the following equation:

$$CI(x,t) = 1 - \exp\left(-\int_0^t h(x,u)\mathrm{d}u\right)$$

In the plot below, we overlay the estimated CIF from the casebase exponential model on the Cox model CIF:

```
R> smooth_risk_exp <- casebase::absoluteRisk(object = casebase_exponential,
R>                                           time = seq(0,15,0.1),
R>                                           newdata = new_data)
R>
R> plot(survfit(cox_model, newdata = new_data),
R>      xlab = "Years since Randomization",
R>      ylab = "Cumulative Incidence",
R>      fun = "event",
R>      xlim = c(0,15), conf.int = FALSE, col = c("red","blue"),
R>      main = sprintf("Estimated Cumulative Incidence (risk) of Death from Prostate
R>                     Cancer Screening group Hazard Ratio: %.2g (%.2g, %.2g)",
R>                     exp(coef(cox_model)),
R>                     exp(confint(cox_model))[1],
R>                     exp(confint(cox_model))[2]))
```

```
R> lines(smooth_risk_exp[,1], smooth_risk_exp[,2], col = "red", lty = 2)
R> lines(smooth_risk_exp[,1], smooth_risk_exp[,3], col = "blue", lty = 2)
R>
R>
R> legend("topleft",
R>        legend = c("Control group (Cox)","Control group (Casebase)",
R>                   "Screening group (Cox)", "Screening group (Casebase)"),
R>        col = c("red","red", "blue","blue"),
R>        lty = c(1, 2, 1, 2),
R>        bg = "gray90")
```

As we can see, the exponential model is not a good fit. Based on what we observed in the population time plot, where more events are observed later on in time, this poor fit is expected. A constant hazard model would overestimate the cumulative incidence earlier on in time, and underestimate it later on; this is what we see on the cumulative incidence plot. This example demonstrates the benefits of population time plots as an exploratory analysis tool.

Next we enter time linearly into the model:

```
R> casebase_time <- fitSmoothHazard(DeadOfPrCa ~ Follow.Up.Time + ScrArm,
R>                                  data = ERSPC,
R>                                  ratio = 100)
R>
R> summary(casebase_time)
R> exp(coef(casebase_time))
R> exp(confint(casebase_time))

R> smooth_risk_time <- casebase::absoluteRisk(object = casebase_time,
R>                                            time = seq(0,15,0.1),
R>                                            newdata = new_data)
R>
R> plot(survfit(cox_model, newdata = new_data),
R>      xlab = "Years since Randomization",
R>      ylab = "Cumulative Incidence",
R>      fun = "event",
R>      xlim = c(0,15), conf.int = FALSE, col = c("red","blue"),
R>      main = sprintf("Estimated Cumulative Incidence (risk) of Death from Prostate
R>                     Cancer Screening group Hazard Ratio: %.2g (%.2g, %.2g)",
R>                     exp(coef(cox_model)),
R>                     exp(confint(cox_model))[1],
R>                     exp(confint(cox_model))[2]))
R> lines(smooth_risk_time[,1], smooth_risk_time[,2], col = "red", lty = 2)
R> lines(smooth_risk_time[,1], smooth_risk_time[,3], col = "blue", lty = 2)
R>
R> legend("topleft",
R>        legend = c("Control group (Cox)","Control group (Casebase)",
```

```
R>                   "Screening group (Cox)", "Screening group (Casebase)"),
R>        col = c("red","red", "blue","blue"),
R>        lty = c(1, 2, 1, 2),
R>        bg = "gray90")
```

We see that the Weibull model leads to a better fit.

Next we try to enter a smooth function of time into the model using the `splines` package:

```
R> casebase_splines <- fitSmoothHazard(DeadOfPrCa ~ bs(Follow.Up.Time) + ScrArm,
R>                                     data = ERSPC,
R>                                     ratio = 100)
R>
R> summary(casebase_splines)
R> exp(coef(casebase_splines))
R> exp(confint(casebase_splines))

R> smooth_risk_splines <- absoluteRisk(object = casebase_splines,
R>                                     time = seq(0,15,0.1),
R>                                     newdata = new_data)
R>
R> plot(survfit(cox_model, newdata = new_data),
R>      xlab = "Years since Randomization",
R>      ylab = "Cumulative Incidence",
R>      fun = "event",
R>      xlim = c(0,15), conf.int = FALSE, col = c("red","blue"),
R>      main = sprintf("Estimated Cumulative Incidence (risk) of Death from Prostate
R>                     Cancer Screening group Hazard Ratio: %.2g (%.2g, %.2g)",
R>                     exp(coef(cox_model)),
R>                     exp(confint(cox_model))[1],
R>                     exp(confint(cox_model))[2]))
R> lines(smooth_risk_splines[,1], smooth_risk_splines[,2], col = "red", lty = 2)
R> lines(smooth_risk_splines[,1], smooth_risk_splines[,3], col = "blue", lty = 2)
R>
R> legend("topleft",
R>        legend = c("Control group (Cox)","Control group (Casebase)",
R>                   "Screening group (Cox)", "Screening group (Casebase)"),
R>        col = c("red","red", "blue","blue"),
R>        lty = c(1, 2, 1, 2),
R>        bg = "gray90")
```

It looks like the best fit.

Since we are within the GLM framework, we can easily test for which model better fits the data using a Likelihood Ratio Test (LRT). The null hypothesis here is that the linear model is just as good as the larger (in terms of number of parameters) splines model.

```
R> anova(casebase_time, casebase_splines, test = "LRT")
```

As expected, we see that splines model provides a better fit.

5. Case study 2–Bone-marrow transplant

The next example shows how case-base sampling can also be used in the context of a competing risk analysis. For illustrative purposes, we will use the same data that was used in Scrucca *et al* (2010). The data was downloaded from the main author's website, and it is also available as part of the **casebase** package.

```
R> data(bmtcrr)
```

The data contains information on 177 patients who received a stem-cell transplant for acute leukemia. The event of interest is relapse, but other competing causes (e.g. transplant-related death) were also recorded Several covariates were also captured at baseline: sex, disease type (acute lymphoblastic or myeloblastic leukemia, abbreviated as ALL and AML, respectively), disease phase at transplant (Relapse, CR1, CR2, CR3), source of stem cells (bone marrow and peripheral blood, coded as BM+PB, or only peripheral blood, coded as PB), and age. A summary of these baseline characteristics appear in Table 2. We note that the statistical summaries were generated differently for different variable types: for continuous variables, we gave the range, followed by the mean and standard deviation; for categorical variables, we gave the counts for each category.

Variable	Description	Statistical summary
Sex	Sex	M=Male (100)
		F=Female (77)
D	Disease	ALL (73)
		AML (104)
Phase	Phase	CR1 (47)
		CR2 (45)
		CR3 (12)
		Relapse (73)
Source	Type of transplant	BM+PB (21)
		PB (156)
Age	Age of patient (years)	4–62
		30.47 (13.04)
Ftime	Failure time (months)	0.13–131.77
		20.28 (30.78)
Status	Status indicator	0=censored (46)
		1=relapse (56)
		2=competing event (75)

Table 2: Baseline characteristics of patients in the stem-cell transplant study.

In order to try and visualize the incidence density of relapse, we can look at the corresponding population-time plot. In Figure **??**, failure times associated with relapse are highlighted on the plot using red points, while Figure **??** provides a similar population-time plot for competing events.

Our main objective is to compute the absolute risk of relapse for a given set of covariates. First, we fit a smooth hazard to the data; for the sake of this example, we opted for a linear term for time:

```
R> model_cb <- fitSmoothHazard(
R>     Status ~ ftime + Sex + D + Phase + Source + Age,
R>     data = bmtcrr,
R>     ratio = 100,
R>     time = "ftime")
```

From the fit object, we can extract both the hazard ratios and their corresponding confidence intervals:

As we can see, the only significant hazard ratio is the one associated with the phase of the disease at transplant. More precisely, being in relapse at transplant is associated with a hazard ratio of 3.92 when compared to CR1.

Given our estimate of the hazard function, we can compute the absolute risk curve for a fixed covariate profile. We performed this computation for a 35 year old woman who received a stem-cell transplant from peripheral blood at relapse. We compared the absolute risk curve for such a woman with acute lymphoblastic leukemia with that for a similar woman with acute myeloblastic leukemia. Figure **??** shows these two curves as a function of time. This figure also shows the Kaplan-Meier estimate fitted to the two disease groups (ignoring the other covariates).

```
R> # Pick 100 equidistant points between 0 and 60 months
R> time_points <- seq(0, 60, length.out = 50)
R>
R> # Data.frame containing risk profile
R> newdata <- data.frame("Sex" = factor(c("F", "F"),
R>                                       levels = levels(bmtcrr[,"Sex"])),
R>                       "D" = c("ALL", "AML"),
R>                       "Phase" = factor(c("Relapse", "Relapse"),
R>                                        levels = levels(bmtcrr[,"Phase"])),
R>                       "Age" = c(35, 35),
R>                       "Source" = factor(c("PB", "PB"),
R>                                         levels = levels(bmtcrr[,"Source"])))
R>
R> # Estimate absolute risk curve
R> risk_cb <- absoluteRisk(object = model_cb, time = time_points,
R>                         method = "numerical", newdata = newdata)
```

6. Case study 3–The Cancer Genome Atlas

- Glmnet and TCGA data

7. Case study 4–Stanford Heart Transplant Data

Although the previous case studies provide a broad overview of the capabilities of **casebase**, they all have two properties in common:

- All covariates are fixed, i.e. they do not change over time;
- The estimated hazard functions satisfy the "proportional hazard" assumption.

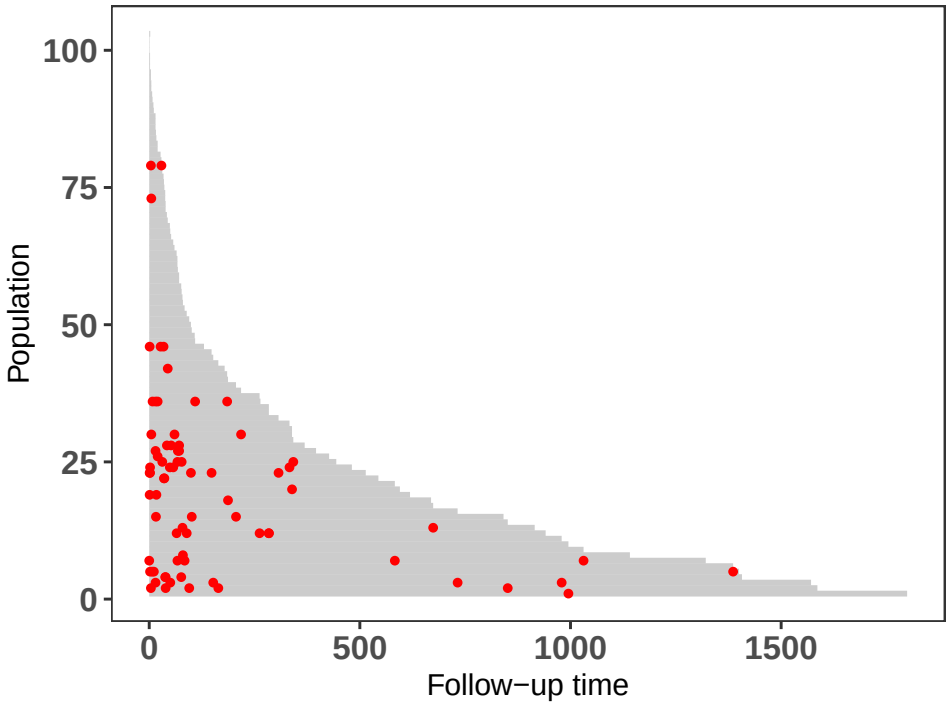
Case-base sampling has been used in the literature to study vaccination safety, where the exposure period was defined as the week following vaccination (Saarela and Hanley 2015). Hence, the main covariate of interest, i.e. exposure to the vaccine, was changing over time. In this context, case-base sampling offers an efficient alternative to nested case-control designs or self-matching.

The purpose of the next case study is to show how **casebase** can also be used for survival analysis problems that do not share the properties above. To this end, we will use the Stanford Heart Transplant Data (Clark, Stinson, Griepp, Schroeder, Shumway, and Harrison 1971, Crowley and Hu (1977)); we will use the version available in the **survival** package.

Recall the setting of this study: patients were admitted to the Stanford program after meeting with their physician and determining that they were unlikely to respond to other forms of treatment. After enrollment, the program searches for a suitable donor for the patient, which can take anywhere between a few days to almost a year. We are interested in the effect of a heart transplant on survival; therefore, the patient is considered exposed only after the transplant has occured.

As before, we can look at the population-time plot for a graphical summary of the event incidence. As we can see, most events occur early during the follow-up period, and therefore we do not expect the hazard to be constant.

```
R> library(survival)
R> library(casebase)
R>
R> stanford_popTime <- popTime(jasa, time = "futime",
R+                             event = "fustat")
R> plot(stanford_popTime)
```



Since the exposure is time-dependent, we need to manually define the exposure variable *after* case-base sampling and *before* fitting the hazard function. For this reason, we will use the `sampleCaseBase` function directly.

```
R> library(tidyverse)
R> library(lubridate)
R>
R> cb_data <- sampleCaseBase(jasa, time = "futime",
R+                           event = "fustat", ratio = 10)
```

Next, we will compute the number of days from acceptance into the program to transplant, and we use this variable to determine whether each population-moment is exposed or not.

```
R> # Define exposure variable
R> cb_data <- mutate(cb_data,
R+                   txtime = time_length(accept.dt %--% tx.date,
R+                                        unit = "days"),
R+                   exposure = case_when(
R+                     is.na(txtime) ~ 0L,
R+                     txtime > futime ~ 0L,
R+                     txtime <= futime ~ 1L
R+                   ))
```

Finally, we can fit the hazard using various linear predictors.

```
R> library(splines)
R> # Fit several models
```

```
R> fit1 <- fitSmoothHazard(fustat ~ exposure,
R+                         data = cb_data, time = "futime")
R> fit2 <- fitSmoothHazard(fustat ~ exposure + futime,
R+                         data = cb_data, time = "futime")
R> fit3 <- fitSmoothHazard(fustat ~ exposure + bs(futime),
R+                         data = cb_data, time = "futime")
R> fit4 <- fitSmoothHazard(fustat ~ exposure*bs(futime),
R+                         data = cb_data, time = "futime")
```

Note that the fourth model (i.e. `fit4`) includes an interaction term between exposure and follow-up time. In other words, this model no longer exhibit proportional hazards. The evidence of non-proportionality of hazards in the Stanford Heart Transplant data has been widely discussed (Arjas 1988).

We can then compare the goodness of fit of these four models using the Akaike Information Criterion (AIC).

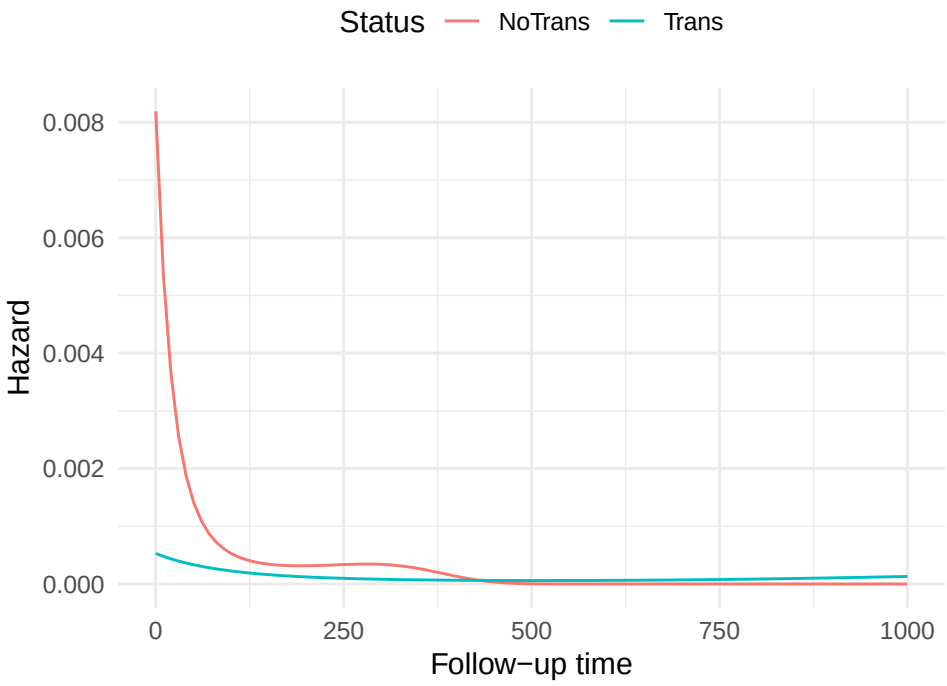
```
R> # Compute AIC
R> c("Model1" = AIC(fit1),
R+   "Model2" = AIC(fit2),
R+   "Model3" = AIC(fit3),
R+   "Model4" = AIC(fit4))

#> Model1 Model2 Model3 Model4
#>    796    760    730    728
```

As we can, the best fit is the fourth model. By visualizing the hazard functions for both exposed and unexposed individuals, we can more clearly see how the hazards are no longer proportional.

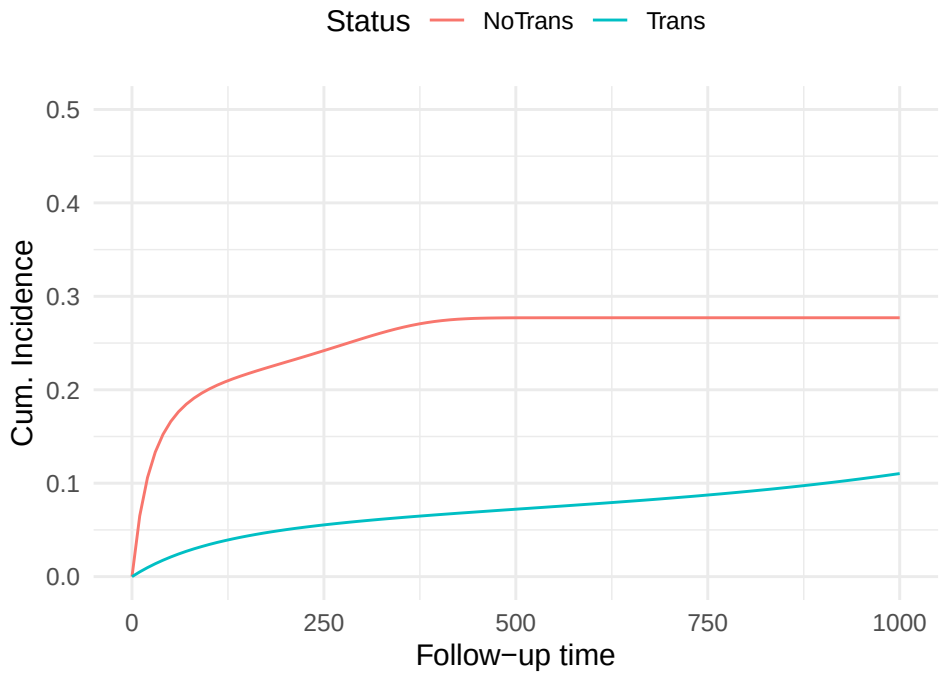
```
R> # Compute hazards---
R> # First, create a list of time points for both exposure status
R> hazard_data <- expand.grid(exposure = c(0, 1),
R+                            futime = seq(0, 1000,
R+                                         length.out = 100))
R> # Set the offset to zero
R> hazard_data$offset <- 0
R> # Use predict to get the fitted values, and exponentiate to
R> # transform to the right scale
R> hazard_data$hazard = exp(predict(fit4, newdata = hazard_data,
R+                                  type = "link"))
R> # Add labels for plots
R> hazard_data$Status = factor(hazard_data$exposure,
R+                            labels = c("NoTrans", "Trans"))
R>
R> ggplot(hazard_data, aes(futime, hazard, colour = Status)) +
R+     geom_line() +
```

```
R+     theme_minimal() +
R+     theme(legend.position = 'top') +
R+     ylab('Hazard') + xlab('Follow-up time')
```



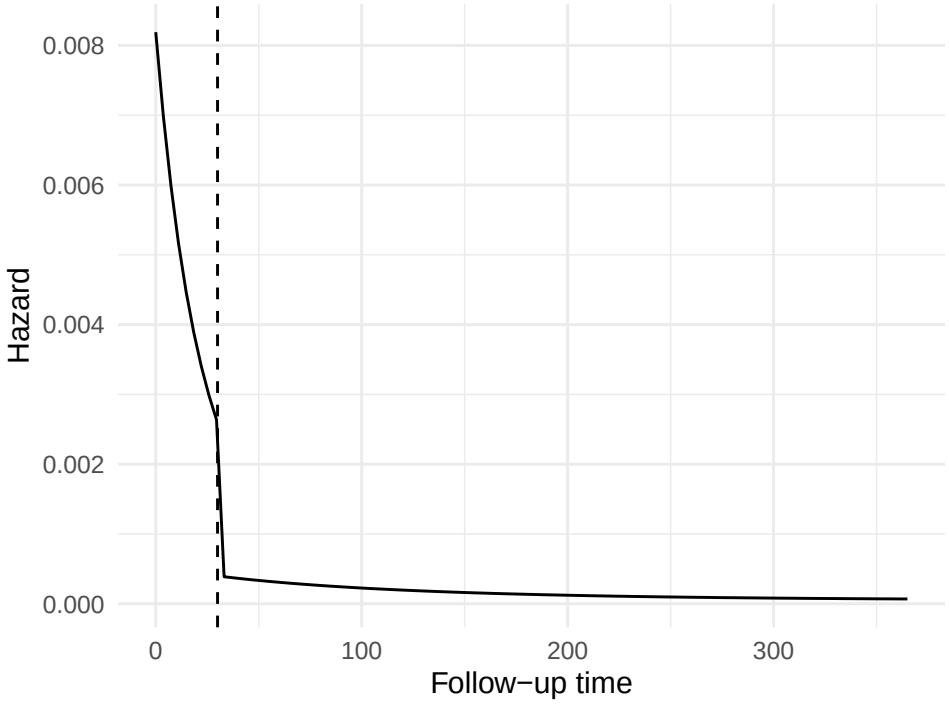
The non-proportionality seems to be more pronounced at the beginning of follow-up than the end. Finally, we can turn these estimate of the hazard function into estimates of the cumulative incidence functions.

```
R> # Compute absolute risk curves
R> newdata <- data.frame(exposure = c(0, 1))
R> absrisk <- absoluteRisk(fit4, newdata = newdata,
R+                         time = seq(0, 1000, length.out = 100))
R>
R> colnames(absrisk) <- c("Time", "NoTrans", "Trans")
R>
R> # Rearrange the data
R> absrisk <- gather(as.data.frame(absrisk),
R+                   "Status", "Risk", -Time)
R>
R> ggplot(absrisk, aes(Time, Risk, colour = Status)) +
R+   geom_line() +
R+   theme_minimal() +
R+   theme(legend.position = 'top') +
R+   expand_limits(y = 0.5) +
R+   xlab('Follow-up time') + ylab('Cum. Incidence')
```



Note that we can easily adapt the code above to the situation where a patient receives a heart transplant at a point in time of interest, for example after 30 days.

```
R> # Compute hazards---
R> # First, create a list of time points for both exposure status
R> one_yr_haz <- data.frame(futime = seq(0, 365,
R+                                       length.out = 100))
R> one_yr_haz <- mutate(one_yr_haz,
R+                      offset = 0,
R+                      exposure = if_else(futime < 30, 0, 1))
R> one_yr_haz$hazard = exp(predict(fit4, newdata = one_yr_haz,
R+                                 type = "link"))
R> ggplot(one_yr_haz, aes(futime, hazard)) +
R+   geom_line() +
R+   theme_minimal() +
R+   theme(legend.position = 'top') +
R+   ylab('Hazard') + xlab('Follow-up time') +
R+   geom_vline(xintercept = 30, linetype = 'dashed')
```



We can then compare the 1-year mortality risk without transplant and with transplant at 30 days.

```
R> absoluteRisk(fit4, newdata = data.frame(exposure = 0),
R+               time = 365)

#>
#> 365 0.27

R>
R> # Use the trapezoidal rule to estimate the cumulative hazard
R> min_hazard <- min(one_yr_haz$hazard)
R> max_hazard <- max(one_yr_haz$hazard)
R> increment <- 365/99
R> cumhaz_est <- 0.5*increment*(sum(2*one_yr_haz$hazard) - min_hazard - max_hazard)
R> 1 - exp(-cumhaz_est)

#> [1] 0.18
```

As we can see, the risk estimate at 1-year is about 30% lower if the patient receives a heart transplant at 30 days.

8. Discussion

In this article, we presented the R package **casebase** that provides functions to fit smooth parametric hazards and estimate cumulative incidence functions using case-base sampling.

We outlined the theoretical underpinnings of the approach, we provided details about our implementation, and we illustrated the merits of the approach and the package through three case studies.

In the following table we provide a comparison between the Cox model and case-base sampling:

9. Environment Details

This report was generated on 2019-12-08 11:49:58 using the following computational environment and dependencies:

```
R> # which R packages and versions?
R> devtools::session_info()

#> - Session info -------------------------------------------------------------
#>  setting  value
#>  version  R version 3.6.1 (2019-07-05)
#>  os       Ubuntu 18.04.3 LTS
#>  system   x86_64, linux-gnu
#>  ui       X11
#>  language (EN)
#>  collate  en_CA.UTF-8
#>  ctype    en_CA.UTF-8
#>  tz       America/Winnipeg
#>  date     2019-12-08
#>
#> - Packages -----------------------------------------------------------------
#>  package     * version    date       lib source
#>  assertthat    0.2.1      2019-03-21 [1] CRAN (R 3.6.1)
#>  backports     1.1.5      2019-10-02 [1] CRAN (R 3.6.1)
#>  broom         0.5.2      2019-04-07 [1] CRAN (R 3.6.1)
#>  callr         3.3.1      2019-07-18 [1] CRAN (R 3.6.1)
#>  casebase    * 0.2.1.9001 2019-07-29 [1] local
#>  cellranger    1.1.0      2016-07-27 [1] CRAN (R 3.5.0)
#>  cli           1.1.0      2019-03-19 [1] CRAN (R 3.6.1)
#>  colorspace    1.4-1      2019-03-18 [1] CRAN (R 3.6.1)
#>  crayon        1.3.4      2017-09-16 [1] CRAN (R 3.5.0)
#>  data.table    1.12.2     2019-04-07 [1] CRAN (R 3.6.1)
#>  desc          1.2.0      2018-05-01 [1] CRAN (R 3.5.0)
#>  devtools      2.1.0      2019-07-06 [1] CRAN (R 3.6.1)
#>  digest        0.6.22     2019-10-21 [1] CRAN (R 3.6.1)
#>  dplyr       * 0.8.3      2019-07-04 [1] CRAN (R 3.6.1)
#>  ellipsis      0.3.0      2019-09-20 [1] CRAN (R 3.6.1)
#>  evaluate      0.14       2019-05-28 [1] CRAN (R 3.6.1)
#>  forcats     * 0.4.0      2019-02-17 [1] CRAN (R 3.6.1)
#>  fs            1.3.1      2019-05-06 [1] CRAN (R 3.6.1)
#>  generics      0.0.2      2018-11-29 [1] CRAN (R 3.6.1)
```

```
#>  ggplot2     * 3.2.1      2019-08-10 [1] CRAN (R 3.6.1)
#>  glue          1.3.1      2019-03-12 [1] CRAN (R 3.6.1)
#>  gtable        0.3.0      2019-03-25 [1] CRAN (R 3.6.1)
#>  haven         2.1.1      2019-07-04 [1] CRAN (R 3.6.1)
#>  hms           0.5.2      2019-10-30 [1] CRAN (R 3.6.1)
#>  htmltools     0.3.6      2017-04-28 [1] CRAN (R 3.5.0)
#>  httr          1.4.1      2019-08-05 [1] CRAN (R 3.6.1)
#>  jsonlite      1.6        2018-12-07 [1] CRAN (R 3.6.1)
#>  knitr         1.23       2019-05-18 [1] CRAN (R 3.6.1)
#>  labeling      0.3        2014-08-23 [1] CRAN (R 3.5.0)
#>  lattice       0.20-38    2018-11-04 [4] CRAN (R 3.5.1)
#>  lazyeval      0.2.2      2019-03-15 [1] CRAN (R 3.6.1)
#>  lifecycle     0.1.0      2019-08-01 [1] CRAN (R 3.6.1)
#>  lubridate   * 1.7.4      2018-04-11 [1] CRAN (R 3.5.0)
#>  magrittr    * 1.5        2014-11-22 [1] CRAN (R 3.5.0)
#>  Matrix        1.2-18     2019-11-27 [4] CRAN (R 3.6.1)
#>  memoise       1.1.0      2017-04-21 [1] CRAN (R 3.5.0)
#>  mgcv          1.8-24     2018-06-18 [1] CRAN (R 3.5.1)
#>  modelr        0.1.5      2019-08-08 [1] CRAN (R 3.6.1)
#>  munsell       0.5.0      2018-06-12 [1] CRAN (R 3.6.1)
#>  nlme          3.1-142    2019-11-07 [4] CRAN (R 3.6.1)
#>  pillar        1.4.2      2019-06-29 [1] CRAN (R 3.6.1)
#>  pkgbuild      1.0.3      2019-03-20 [1] CRAN (R 3.6.1)
#>  pkgconfig     2.0.3      2019-09-22 [1] CRAN (R 3.6.1)
#>  pkgload       1.0.2      2018-10-29 [1] CRAN (R 3.6.1)
#>  prettyunits   1.0.2      2015-07-13 [1] CRAN (R 3.5.2)
#>  processx      3.4.1      2019-07-18 [1] CRAN (R 3.6.1)
#>  ps            1.3.0      2018-12-21 [1] CRAN (R 3.6.1)
#>  purrr       * 0.3.3      2019-10-18 [1] CRAN (R 3.6.1)
#>  R6            2.4.1      2019-11-12 [1] CRAN (R 3.6.1)
#>  Rcpp          1.0.3      2019-11-08 [1] CRAN (R 3.6.1)
#>  readr       * 1.3.1      2018-12-21 [1] CRAN (R 3.6.1)
#>  readxl        1.3.1      2019-03-13 [1] CRAN (R 3.6.1)
#>  remotes       2.1.0      2019-06-24 [1] CRAN (R 3.6.1)
#>  rlang         0.4.1      2019-10-24 [1] CRAN (R 3.6.1)
#>  rmarkdown     1.16       2019-10-01 [1] CRAN (R 3.6.1)
#>  rprojroot     1.3-2      2018-01-03 [1] CRAN (R 3.5.0)
#>  rstudioapi    0.10       2019-03-19 [1] CRAN (R 3.6.1)
#>  rticles       0.9.1      2019-07-22 [1] Github (rstudio/rticles@8d56fc6)
#>  rvest         0.3.5      2019-11-08 [1] CRAN (R 3.6.1)
#>  scales        1.0.0      2018-08-09 [1] CRAN (R 3.6.1)
#>  sessioninfo   1.1.1      2018-11-05 [1] CRAN (R 3.6.1)
#>  stringi       1.4.3      2019-03-12 [1] CRAN (R 3.6.1)
#>  stringr     * 1.4.0      2019-02-10 [1] CRAN (R 3.6.1)
#>  survival    * 2.44-1.1   2019-04-01 [4] CRAN (R 3.6.1)
#>  testthat      2.2.0      2019-07-22 [1] CRAN (R 3.6.1)
#>  tibble      * 2.1.3      2019-06-06 [1] CRAN (R 3.6.1)
```

```
#>  tidyr       * 1.0.0      2019-09-11 [1] CRAN (R 3.6.1)
#>  tidyselect    0.2.5      2018-10-11 [1] CRAN (R 3.5.2)
#>  tidyverse   * 1.2.1      2017-11-14 [1] CRAN (R 3.5.0)
#>  usethis       1.5.1      2019-07-04 [1] CRAN (R 3.6.1)
#>  vctrs         0.2.0      2019-07-05 [1] CRAN (R 3.6.1)
#>  VGAM          1.1-1      2019-02-18 [1] CRAN (R 3.6.1)
#>  withr         2.1.2      2018-03-15 [1] CRAN (R 3.5.0)
#>  xfun          0.8        2019-06-25 [1] CRAN (R 3.6.1)
#>  xml2          1.2.2      2019-08-09 [1] CRAN (R 3.6.1)
#>  yaml          2.2.0      2018-07-25 [1] CRAN (R 3.6.1)
#>  zeallot       0.1.0      2018-01-28 [1] CRAN (R 3.5.1)
#>
#> [1] /home/mturgeon/Rlibs
#> [2] /usr/local/lib/R/site-library
#> [3] /usr/lib/R/site-library
#> [4] /usr/lib/R/library
```

The current Git commit details are:

```
R> # what commit is this file at?
R> git2r::repository(here::here())
```

```
#> Local:    review-max /home/mturgeon/Documents/git_repositories/cbpaper
#> Head:     [f601653] 2019-12-08: Merge branch 'review-max' of https://github.com/sahirbh
```

References

Arjas E (1988). "A graphical method for assessing goodness of fit in Cox's proportional hazards model." *Journal of the American Statistical Association*, **83**(401), 204–212.

Clark DA, Stinson EB, Griepp RB, Schroeder JS, Shumway NE, Harrison D (1971). "Cardiac transplantation in man." *Annals of Internal Medicine*, **75**(1), 15–21.

Crowley J, Hu M (1977). "Covariance analysis of heart transplant survival data." *Journal of the American Statistical Association*, **72**(357), 27–36.

Friedman J, Hastie T, Tibshirani R (2010). "Regularization Paths for Generalized Linear Models via Coordinate Descent." *Journal of Statistical Software*, **33**(1). ISSN 1548-7660. doi:10.18637/jss.v033.i01.

Hanley JA, Miettinen OS (2009). "Fitting smooth-in-time prognostic risk functions via logistic regression." *The International Journal of Biostatistics*, **5**(1).

Hastie T, Tibshirani R (1987). "Generalized additive models: some applications." *Journal of the American Statistical Association*, **82**(398), 371–386.

Saarela O (2016). "A case-base sampling method for estimating recurrent event intensities." *Lifetime data analysis*, **22**(4), 589–605.

Saarela O, Arjas E (2015). "Non-parametric Bayesian Hazard Regression for Chronic Disease Risk Assessment." *Scandinavian Journal of Statistics*, **42**(2), 609–626.

Saarela O, Hanley JA (2015). "Case-base methods for studying vaccination safety." *Biometrics*, **71**(1), 42–52.

Scrucca L, Santucci A, Aversa F (2010). "Regression modeling of competing risk using R: an in depth guide for clinicians." *Bone marrow transplantation*, **45**(9), 1388.

Affiliation:

Sahir Bhatnagar *
McGill University
1020 Pine Avenue West Montreal, QC, Canada H3A 1A2
E-mail: sahir.bhatnagar@mail.mcgill.ca
URL: http://sahirbhatnagar.com/

Maxime Turgeon *
University of Manitoba
186 Dysart Road Winnipeg, MB, Canada R3T 2N2
E-mail: max.turgeon@umanitoba.ca
URL: https://maxturgeon.ca/

Jesse Islam
McGill University
1020 Pine Avenue West Montreal, QC, Canada H3A 1A2
E-mail: jesse.islam@mail.mcgill.ca

James Hanley
McGill University
1020 Pine Avenue West Montreal, QC, Canada H3A 1A2
E-mail: james.hanley@mcgill.ca
URL: http://www.medicine.mcgill.ca/epidemiology/hanley/

Olli Saarela
University of Toronto
Dalla Lana School of Public Health, 155 College Street, 6th floor, Toronto, Ontario M5T 3M7, Canada
E-mail: olli.saarela@utoronto.ca
URL: http://individual.utoronto.ca/osaarela/

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
MMMMMM YYYY, Volume VV, Issue II *Submitted:* yyyy-mm-dd
doi:10.18637/jss.v000.i00 *Accepted:* yyyy-mm-dd