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 Univeristy
Maxime Turgeon * University of Manitoba
Jesse Islam McGill University
James Hanley McGill Univeristy
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. 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.

2. Literature review

To conduct our review on current methods available in survival analysis, a scraper was coded to examine all packages in the survival task view on CRAN (2019). There was a total of 259 packages, many of which are not relevant to the problems casebase addresses. To filter these

	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**.

packages, a scraper was used to search for competing risks, cumulative incidence functions, non- proportional hazards and penalization. This reduced our set of 259 packages down to 60. From here, we manually examined each package to determine which ones are comparable to casebase, with either a different approach to the same problems or future features we would like to add to casebase.

These packages can belong to three main categories: non-proportional hazards, regularization and competing risks. Besides these three main categories, there are extra features of interest as well. Table 2 summarizes the findings.

Many packages exist that implement semi-parametric Cox models. Of those, **glmnet** (2011)???, **glmpath** (2018), **penalized** (2010), **RiskRegression** (2019) introduce penalization in their model fitting. **CoxRidge** (2015), **rstpm2** (2019) and **survival** (2015) include penalization, but they implement the extended cox model permitting intuitive non proportional hazard modeling while the previous four do not. In terms of penalization, the benefits of **casebase** comes from its flexibility. Linear models have a wide tool set, considerably larger than those made for Cox regression. **casebase** permits many regularization tools developed for linear regression to be repurposed within a survival context.

Parametric models permit non-proportional hazards and are implemented in a handful of packages. Of interest are **CFC** (2019), **flexsurv** (2016), **SmoothHazard** (2017), **rsptm2** (2019) and **survival** (2015). **SmoothHazard** is limited to Weibull distributions (2017), whereas **flexsurv** and **survival** contain implementations for users to supply any distribution they would like (2015), (2016). **flexsurv**, **smoothhazard**, **mets** and **rstpm2** can use splines, permitting a flexible fit over time (2016), (2017), (2014), (2019). Though **casebase** is limited to three distributions at the moment, it also uses splines allowing for distributions that may not be consistent over survival time.

Of the methods mentioned so far, only **CFC**, **flexsurv**, **mets** and **survival** contain implementations for competing risks (2019), (2016), (2014), (2015). The differentiating factor between

these packages and **casebase** is that **casebase** handles competing risks through multiple logistic regression. **SmoothHazard**, **survival** and **Rstpm2** can handle interval censoring, whereas **casebase** currently cannot. Interval censoring is a simple feature to implement and is a potential improvement to add to **casebase**.

Of all the packages listed in Table 2, **CFC**, **Flexsurv**, **mets**, **RiskRegression**, **rstpm2** and **survival** have implementations for cumulative incidence (2019), (2016), (2014), (2019), (2019), (2015). Since **casebase** is parametric, the baseline hazard is known and requires integration to retrieve the cumulative incidence. This is similar to the approaches of **CFC** and **flexsurv** (2019), (2016), while the other packages use semi-parametric approaches to retrieve the baseline after fitting (2014), (2019). **rstpm2** and **survival** use both for their respective parametric and semi-parametric model implementations (2019), (2015).

Package	Competing Risks	Non-proportional	Penalization	Splines	Parametric	Semi-parametric	Interval/left Censoring	Absolute Risk
casebase Hanley and Miettinen (2009)	x	x	x	x	x			x
CFC Sharabiani and Mahani (2019)	x	x			x			x
coxRidge Perperoglou (2015)		x	x			x		
crrp Fu	x		x					
fastcox Yi and Zou			x			x		
Flexrsurv Clerc-UrmÃ˙s, Grzebyk, HÃ©delin, and CENSUR working survival group (2017)		x		x	x			Cumulative hazard?
Flexsurv Jackson (2016)	x	x		x	x			Cumulative hazard?
glmnet Simon *et al.* (2011)			x			x		
glmpath Park and Hastie (2018)			x			x		
mets Scheike *et al.* (2014)	x			x		x		x
penalized Goeman, Meijer, Chaturvedi, and Lueder (2019)			x			x		
RiskRegression Gerds *et al.* (2019)			x			x		x
Rstpm2 Clements *et al.* (2019)		x	x	x	x	x	x	Cumm haz
SmoothHazard Touraine *et al.* (2017)		x		x	x		x	
Survival Therneau (2015)	x	x			x	x	x	x

Table 2: Different features of interest in various survival packages.

3. 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*.

Popular survival analysis methods, like Kaplan-Meier and Cox regression, rely on the notion of risk-set sampling. Case-base sampling can be seen as an alternative.

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-homogenous Poisson processes, we eventually reach an expression where each person-moment's contribution is of the form

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

where $N(t)$ is the counting process associated with the event of interest, $\lambda(t)$ 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 $\log(\lambda(t)) = g(t; X)$ 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 samping *uniformly*, i.e. to sample the base series uniformly from the study base. In terms of Poisson processes,

this sampling strategy corresponds 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

$$h(x,t) = exp[g(x,t)]$$

where t denotes the numerical value (number of units) 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). 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.

$$h(x,t) = exp(\beta_0 + \beta_1 x)$$

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 on study a.k.a the memoryless property (Kalbfleisch and Prentice, 2002).

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

$$h(x,t) = exp(\beta_0 + \beta_1 t + \beta_2 x)$$

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

$$h(x,t) = exp(\beta_0 + \beta_1 \log(t) + \beta_2 x)$$

4. 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) data analysis.

We explicitely aimed at 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 testing units.

4.1. Population-time plots

4.2. Data analysis

The data analysis step was separated into three parts: 1) case-base sampling; 2) estimation of the smooth hazard function; and 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.

Finally, the hazard function is fitted to the data using the function `glm`, unless there is more than one type event (i.e. in a competing-risk analysis), in which case it uses the multinomical regression capabilities of 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. For the case of a competing-event analysis, the event-specific risk is computed using a nested double integral; in this setting, Monte-Carlo integration is faster and more flexible than `stats::integrate`. This is due to the fact that the computation involves a double loop over the selected time points; Monte-Carlo integration performs this step by using the vectorized `rowSums` and `colSums` functions.

To decide between a single-event and a competing-event analysis, we created `absoluteRisk` as an `S3` generic, with methods for both `glm` and `CompRisk` objects (the latter inherits from `vglm` as well). The method dispatch system of `R` then takes care of matching the correct

input to the correct methodology, without the user's intervention.

5. 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 ~ splines::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 espected, we see that splines model provides a better fit.

6. 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 3. 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.

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:

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

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)
```

se study 3–Study to Understand Prognoses Preferences Outcomes and Risks of Trea

We examined the Study to Understand Prognoses Preferences Outcomes and Risks of Treatment (SUPPORT) dataset (REF from main site). the SUPPORT dataset tracks death for individuals who are considered seriously ill within a hospital. Imputation was conducted using the default methods in the mice package in R, available on CRAN. Before imputation, there was a total of 1000 individuals and 35 variables. After imputation, we have 474 observations where 67.5% of individuals in the study experience the event of interest. The SUPPORT dataset will be used to demonstrate regularization within casebase (REF), while comparing their absolute risk preditions for a new individual. A description of each variable can be found in Table˜4 and a breakdown of each categorical variable in Table˜5. The data was downloaded from the main author's website, and it is also available as part of the casebase package.

Hospital death as a covariate was removed, as this is directly informative of death. The glmnet package(REF) on CRAN was used to introduce elastic net, lasso and ridge regressions to casebase. To visualize the effect of imputation on incidence density, we can refer to the pre-imputation population-time plot in Figure **??** and post-imputation population time plot in Figure **??**. The incidence density is consistent between imputed and complete datasets.

```
R> #demonstrating incidence density before imputation
R> plot(casebase::popTime(as.data.frame(support),time = "d.time",event = "death"))

R> #demonstrating incidence density after imputation
R> plot(casebase::popTime(as.data.frame(workingCompleteData),time = "d.time",event = "deat
```

Our main objective is to compute the absolute risk of death for a given set of covariates, using regularization when fitting our model. First, we fit a smooth hazard to the data; for the sake of this example, we opted for a linear term for time. As casebase makes use of the glmnet package, we interact with the fitsmoothhazard.fit function using a matrix interface, where y contains the time and event variables and x contains all other variables we would like to include in the model. For the SUPPORT dataset, all factor variables were converted into dummy variables. Here, we used lasso penalization by setting alpha to 1.

```
R> #Using casebase's glmnet implementation, using splines and a case to base ratio of 100,
R> #to determine the hazard function
R> cb.Model=casebase::fitSmoothHazard.fit(x,y,family="glmnet",time="d.time",event="death",
R>                                 formula_time = ~d.time,alpha=1,ratio=100)
```

Then, we take our model and use the absoluteRisk function to integrate and retrieve our absolute risk function.

```
R> #Estimating the absolute risk curve using the newData parameter.
R> casebaseAbsolute=casebase::absoluteRisk(cb.Model,time = seq(1,max(completeData$d.time),
R>                                  s="lambda.1se",method=c("numerical"))
```

We will compare this curve to a regularized version of cox regression, and a kaplan-meier survival curve. The regularized cox regression absolute risk curve required two extra steps to retrive. Coxnet from the glmnet package has tools to fit a regularized cox model, but requires that a survival object skeleton of the selected variables, so that the survival package tools can be used to retrive the absolute risk. We handle this by fitting a regularized cox model with the glmnet package, and a cox model from the survival package with the selected variables from the regularized fit. This second model serves as a skeleton that permits the use of survival package functions. The coefficients from the regularized model will replace the coeficients in our skeleton. the resulting model will have the correct coefficients when integrating, but will have an incorrect standard error. For the purposes of this case study, we are only interested in the absolute risk curve itself and not the standard error.

```
R> #Create u and xCox, which will be used when fitting Cox with glmnet.
R> u=survival::Surv(time = as.numeric(y[,2]), event = as.factor(y[,1]))
R> xCox=as.matrix(sparse.model.matrix(death~ .-d.time,data=completeData))[-c(sam),]
R> #hazard for cox using glmnet
R> coxglmFit=glmnet::cv.glmnet(x=xCox,y=u, family="cox",alpha=1)
R> #convergence demonstrated in plot
R> plot(coxglmFit)
R> #taking the coefficient estimates for later use
R> nonzero_covariate_cox <- predict(coxglmFit, type = "nonzero", s = "lambda.1se")
R> nonzero_coef_cox <- coef(coxglmFit, s = "lambda.1se")
R> #creating a new dataset that only contains the covariates chosen through glmnet.
R> #cleanCoxData<- as.data.frame(cbind(as.numeric(workingData$d.time),as.factor(workingDat
R> cleanCoxData<-as.data.frame(cbind(as.numeric(y[,2]),as.numeric(y[,1]),xCox[,nonzero_cov
R> #newDataCox<-xCox[sam,nonzero_covariate_cox$X1]
R> #fitting a cox model using regular estimation, however we will not keep it.
R> #this is used more as an object place holder.
R> coxFit <- survival::coxph(Surv(time=V1,event=V2) ~ ., data = cleanCoxData)
R> #The coefficients of this object will be replaced with the estimates from coxglmFit.
R> #Doing so makes it so that everything is invalid aside from the coefficients.
R> #In this case, all we need to estimate the absolute risk is the coefficients.
R> #Std. error would be incorrect here, if we were to draw error bars.
R> coxFit$coefficients<-nonzero_coef_cox@x
R> #Fitting absolute risk curve for cox+glmnet
R> newDataCox=newData[nonzero_covariate_cox$X1-1]
R>
R> abCoxFit<-survival::survfit(coxFit,newdata=as.data.frame(t(newDataCox)),time = seq(0,ma
```

We used the survival package to calculate the kaplan-meier absolute risk function.

```
R> #creating a surv object to be used to fit an unadjusted absolute risk curve.
R> # (Kaplan-Meier risk curve)
R> km <- survival::Surv(time = completeData$d.time, event = completeData$death)
R> abKm<-survival::survfit(km~1,type='kaplan-meier',conf.type='log')
```

Now that our three absolute risk functions have been calculated, we compare them all in

Figure **??**. Both coxnet and casebase decrease the absolute risk by the end of the study, in comparison to kaplan-meier.

```
R> # A plot to compare all three Absolute risk (Commulative Incidence)
R> plot(casebaseAbsolute,type='l',col="black",lwd=2,main="Support- CaseBase vs. Cox+glmnet
R> lines(abCoxFit ,col="red",fun="event",lwd=3,conf.int = FALSE)
R> lines(abKm ,col="Blue",fun="event",lwd=3,conf.int = FALSE)
R> legend("bottom",
R>        legend = c( "Casebase (Lasso+linear)","semi-parametric (Cox)+glmnet","KM curve")
R>        col = c("black","red","Blue"),
R>        lty = c(1, 1, 1),
R>        bg = "gray90")
```

8. Discussion

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 12:21:42 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)
#>  codetools     0.2-16     2018-12-24 [4] CRAN (R 3.5.2)
#>  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)
#>  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)
#>  foreach     * 1.4.4      2017-12-12 [1] CRAN (R 3.5.0)
#>  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)
#>  glmnet      * 2.0-18     2019-05-20 [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)
#>  iterators     1.0.9      2017-12-12 [1] CRAN (R 3.5.0)
#>  jsonlite      1.6        2018-12-07 [1] CRAN (R 3.6.1)
#>  knitr         1.23       2019-05-18 [1] CRAN (R 3.6.1)
#>  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:    Jesse-Islam-master /home/mturgeon/Documents/git_repositories/cbpaper
#> Head:     [71a5a9c] 2019-12-06: Lit-Review Edits
```

References

Allignol A, Latouche A (2019). "CRAN Task View: Survival Analysis." URL https://cran.r-project.org/web/views/Survival.html.

Clements M, Liu XR, Lambert P, Jakobsen LH, Gasparini A, Smyth G, Alken P, Wood S, Ulerich R (2019). "Smooth Survival Models, Including Generalized Survival Models [R package rstpm2 version 1.5.1]." URL https://cran.r-project.org/web/packages/rstpm2/index.html.

Clerc-UrmÃ¨s I, Grzebyk M, HÃ©delin G, CENSUR working survival group (2017). *flexrsurv: An R package for relative survival analysis*. R package version 1.4.1, URL https://CRAN.R-project.org/package=flexrsurv.

Fu Z (????). "Package crrp." URL https://cran.r-project.org/web/packages/crrp/index.html.

Gerds TA, Blanche P, Mortersen R, Tollenaar N, Mogensen UB, Ozenne B (2019). "Risk Regression Models and Prediction Scores for Survival Analysis with Competing Risks [R package riskRegression version 2019.11.03]." URL https://CRAN.R-project.org/package=riskRegression.

Goeman J, Meijer R, Chaturvedi N, Lueder M (2019). "Package penalized." URL https://cran.r-project.org/web/packages/penalized/index.html.

Goeman JJ (2010). "L1 penalized estimation in the Cox proportional hazards model." *Biometrical Journal*, (52), –14.

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

Jackson C (2016). "flexsurv: A Platform for Parametric Survival Modeling in R." *Journal of Statistical Software*, **70**(8), 1–33. doi:10.18637/jss.v070.i08.

Park MY, Hastie T (2018). "Package glmpath." URL https://CRAN.R-project.org/package=glmpath.

Perperoglou A (2015). "Package CoxRidge." URL https://CRAN.R-project.org/package=CoxRidge.

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.

Scheike TH, Holst KK, Hjelmborg JB (2014). "Estimating twin concordance for bivariate competing risks twin data." *Statistics in medicine*, **33**(7), 1193–1204.

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.

Sharabiani MT, Mahani AS (2019). "Package CFC." URL https://cran.r-project.org/web/packages/CFC/index.html.

Simon N, Friedman J, Hastie T, Tibshirani R (2011). "Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent." *Journal of Statistical Software*, **39**(5), 1–13. URL http://www.jstatsoft.org/v39/i05/.

Therneau TM (2015). *A Package for Survival Analysis in S*. Version 2.38, URL https://CRAN.R-project.org/package=survival.

Touraine C, Gerds TA, Joly P (2017). "SmoothHazard: An R Package for Fitting Regression Models to Interval-Censored Observations of Illness-Death Models." *Journal of Statistical Software*, **79**(7), 1–22. doi:10.18637/jss.v079.i07.

Yi Y, Zou H (????). "Package fastcox." URL https://cran.r-project.org/web/packages/fastcox/index.html.

Affiliation:

Sahir Bhatnagar *
McGill Univeristy
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
318 Machray Hall Winnipeg, MB, Canada R3T 2N2
E-mail: max.turgeon@umanitoba.ca
URL: http://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 Univeristy
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

Name	Labels	Levels	Storage	NAs
age	Age		double	0
death	Death at any time up to NDI date:31DEC94		double	0
sex		2	integer	0
hospdead	Death in Hospital		double	0
slos	Days from Study Entry to Discharge		double	0
d.time	Days of Follow-Up		double	0
dzgroup		8	integer	0
dzclass		4	integer	0
num.co	number of comorbidities		double	0
edu	Years of Education		double	202
income		4	integer	349
scoma	SUPPORT Coma Score based on Glasgow D3		double	0
charges	Hospital Charges		double	25
totcst	Total RCC cost		double	105
totmcst	Total micro-cost		double	372
avtisst	Average TISS, Days 3-25		double	6
race		5	integer	5
meanbp	Mean Arterial Blood Pressure Day 3		double	0
wblc	White Blood Cell Count Day 3		double	24
hrt	Heart Rate Day 3		double	0
resp	Respiration Rate Day 3		double	0
temp	Temperature (celcius) Day 3		double	0
pafi	PaO2/(.01*FiO2) Day 3		double	253
alb	Serum Albumin Day 3		double	378
bili	Bilirubin Day 3		double	297
crea	Serum creatinine Day 3		double	3
sod	Serum sodium Day 3		double	0
ph	Serum pH (arterial) Day 3		double	250
glucose	Glucose Day 3		double	470
bun	BUN Day 3		double	455
urine	Urine Output Day 3		double	517
adlp	ADL Patient Day 3		double	634
adls	ADL Surrogate Day 3		double	310
sfdm2		5	integer	159
adlsc	Imputed ADL Calibrated to Surrogate		double	0

Table 4: A description of each variable in the SUPPORT dataset, taken from (REF).

Variable	Levels
sex	female
	male
dzgroup	ARF/MOSF w/Sepsis
	COPD
	CHF
	Cirrhosis
	Coma
	Colon Cancer
	Lung Cancer
	MOSF w/Malig
dzclass	ARF/MOSF
	COPD/CHF/Cirrhosis
	Coma
	Cancer
income	under $11k
	11−25k
	25−50k
	>$50k
race	white
	black
	asian
	other
	hispanic
sfdm2	no(M2 and SIP pres)
	adl>=4 (>=5 if sur)
	SIP>=30
	Coma or Intub
	<2 mo. follow-up

Table 5: A description of each level within each categorical variable in the dataset, taken from REF.