Package 'casebase'

April 29, 2017

Type Package

Title Fitting Flexible Smooth-in-Time Hazards and Risk Functions via Logistic and Multinomial Regression

Version 0.1.0

Date 2017-4-28

Description Implements the case-base sampling approach of Hanley and Miettinen (2009) <DOI:10.2202/1557-4679.1125>, Saarela and Arjas (2015) <DOI:10.1111/sjos.12125>, and Saarela (2015) <DOI:10.1007/s10985-015-9352-x>, for fitting flexible hazard regression models to survival data with single event type or multiple competing causes via logistic and multinomial regression. From the fitted hazard function, cumulative incidence, risk functions of time, treatment and profile can be derived. This approach accommodates any log-linear hazard function of prognostic time, treatment, and covariates, and readily allows for non-proportionality. We also provide a plot method for visualizing incidence density via population time plots.

Depends R (>= 3.3.1)

Imports data.table, ggplot2, methods, survival, VGAM

License MIT + file LICENSE

LazyData TRUE

Suggests eha, knitr, rmarkdown, splines, testthat

VignetteBuilder knitr

URL http://sahirbhatnagar.com/casebase/

BugReports https://github.com/sahirbhatnagar/casebase/issues

RoxygenNote 6.0.1

NeedsCompilation no

Author Sahir Bhatnagar [aut, cre] (http://sahirbhatnagar.com/), Maxime Turgeon [aut] (http://turgeonmaxime.github.io/), Olli Saarela [aut] (http://individual.utoronto.ca/osaarela/), James Hanley [aut] (http://www.medicine.mcgill.ca/epidemiology/hanley/)

Maintainer Sahir Bhatnagar <sahir.bhatnagar@gmail.com>

Repository CRAN

Date/Publication 2017-04-28 23:27:54 UTC

R topics documented:

`absoluteRisk` *Compute absolute risks using the fitted hazard function.*

Description

Using the output of the function `fitSmoothHazard`, we can compute absolute risks by integrating the fitted hazard function over a time period and then converting this to an estimated survival for each individual.

Usage

```
absoluteRisk(object, ...)

## Default S3 method:
absoluteRisk(object, ...)

## S3 method for class 'glm'
absoluteRisk(object, time, newdata, method = c("montecarlo",
  "numerical"), nsamp = 1000, ...)

## S3 method for class 'CompRisk'
absoluteRisk(object, time, newdata,
  method = c("montecarlo", "numerical"), nsamp = 1000, ...)
```

Arguments

`object`	Output of function `fitSmoothHazard`.
`...`	Extra parameters. Currently these are simply ignored.
`time`	A vector of time points at which we should compute the absolute risks.

newdata	Optionally, a data frame in which to look for variables with which to predict. If omitted, the mean absolute risk is returned.
method	Method used for integration. Defaults to "montecarlo", which implements Monte-Carlo integration. The only other option is "numerical", which simply calls the function integrate.
nsamp	Maximal number of subdivisions (if method = "numerical") or number of sampled points (if method = "montecarlo").

Details

If the user supplies the original dataset through the parameter newdata, the mean absolute risk can be computed as the average of the output vector.

In general, if time is a vector of length greater than one, the output will include a column corresponding to the provided time points. Some modifications of the time vector are done: time=0 is added, the time points are ordered, and duplicates are removed. All these modifications simplify the computations and give an output that can easily be used to plot risk curves.

On the other hand, if time corresponds to a single time point, the output does not include a column corresponding to time.

If there is no competing risk, the output is a matrix where each column corresponds to the several covariate profiles, and where each row corresponds to a time point. If there are competing risks, the output will be a 3-dimensional array, with the third dimension corresponding to the different events.

The numerical method should be good enough in most situation, but Monte Carlo integration can give more accurate results when the estimated hazard function is not smooth (e.g. when modeling with time-varying covariates). However, if there are competing risks, we strongly encourage the user to select Monte-Carlo integration, which is much faster than the numerical method. (This is due to the current implementation of the numerical method, and it may be improved in future versions.)

Value

Returns the estimated absolute risk for the user-supplied covariate profiles. This will be stored in a 2- or 3-dimensional array, depending on the input. See details.

Examples

```
# Simulate censored survival data for two outcome types from exponential distributions
library(data.table)
set.seed(12345)
nobs <- 1000
tlim <- 20

# simulation parameters
b1 <- 200
b2 <- 50

# event type 0-censored, 1-event of interest, 2-competing event
# t observed time/endpoint
# z is a binary covariate
```

```
DT <- data.table(z=rbinom(nobs, 1, 0.5))
DT[,`:=` ("t_event" = rweibull(nobs, 1, b1),
          "t_comp" = rweibull(nobs, 1, b2))]
DT[,`:=`("event" = 1 * (t_event < t_comp) + 2 * (t_event >= t_comp),
         "time" = pmin(t_event, t_comp))]
DT[time >= tlim, `:=`("event" = 0, "time" = tlim)]

out_linear <- fitSmoothHazard(event ~ time + z, DT)
out_log <- fitSmoothHazard(event ~ log(time) + z, DT)

linear_risk <- absoluteRisk(out_linear, time = 10, newdata = data.table("z"=c(0,1)))
log_risk <- absoluteRisk(out_log, time = 10, newdata = data.table("z"=c(0,1)))
```

`bmtcrr` *Data on transplant patients*

Description

Data on patients who underwent haematopoietic stem cell transplantation for acute leukaemia.

Usage

```
bmtcrr
```

Format

A dataframe with 177 observations and 7 variables:

Sex Gender of the individual

D Disease: lymphoblastic or myeloblastic leukemia, abbreviated as ALL and AML, respectively

Phase Phase at transplant (Relapse, CR1, CR2, CR3)

Age Age at the beginning of follow-up

Status Status indicator: 0=censored, 1=relapse, 2=competing event

Source Source of stem cells: bone marrow and peripheral blood, coded as BM+PB, or peripheral blood only, coded as PB

ftime Failure time in months

Source

Available at the following website: http://www.stat.unipg.it/luca/R/

References

Scrucca L, Santucci A, Aversa F. Competing risk analysis using R: an easy guide for clinicians. Bone Marrow Transplant. 2007 Aug;40(4):381-7. doi: 10.1038/sj.bmt.1705727.

checkArgsEventIndicator

Check that Event is in Correct Format

Description

Checks for event categories and gives a warning message indicating which level is assumed to be the reference level.

Usage

```
checkArgsEventIndicator(data, event, censored.indicator)
```

Arguments

data	a data.frame or data.table containing the source dataset.
event	a character string giving the name of the event variable contained in data. See Details. If event is a numeric variable, then 0 needs to represent a censored observation, 1 needs to be the event of interest. Integers 2, 3, ... and so on are treated as competing events. If event is a factor or character and censored.indicator is not specified, this function will assume the reference level is the censored indicator
censored.indicator	a character string of length 1 indicating which value in event is the censored. This function will use relevel to set censored.indicator as the reference level. This argument is ignored if the event variable is a numeric

Value

A list of length two. The first element is the factored event, and the second element is the numeric representation of the event

Examples

```
## Not run:
library(survival) # for veteran data
checkArgsEventIndicator(data = veteran, event = "celltype", censored.indicator = "smallcell")
checkArgsEventIndicator(data = veteran, event = "status")
checkArgsEventIndicator(data = veteran, event = "trt") # returns error

url <- "https://raw.githubusercontent.com/sahirbhatnagar/casebase/master/inst/extdata/bmtcrr.csv"
bmt <- read.csv(url)
checkArgsEventIndicator(data = bmt, event = "Sex", censored.indicator = "M")
checkArgsEventIndicator(data = bmt, event = "D", censored.indicator = "AML")
checkArgsEventIndicator(data = bmt, event = "D", censored.indicator = "AMLL") #returns error
checkArgsEventIndicator(data = bmt, event = "Source")
checkArgsEventIndicator(data = bmt, event = "Status")
```

```
checkArgsEventIndicator(data = bmt, event = "Status", censored.indicator = 3)

## End(Not run)
```

`CompRisk-class` *An S4 class to store the output of fitSmoothHazard*

Description

This class inherits from `vglm-class`.

Usage

```
summary(object, ...)

## S4 method for signature 'CompRisk'
summary(object)
```

Arguments

`object`	Object of class `CompRisk`
`...`	Extra parameters

Slots

`originalData` Data.frame containing the original data (i.e. before case-base sampling). This is used by the `absoluteRisk` function.

`typeEvents` Numeric factor which encodes the type of events being considered (including censoring).

`timeVar` Character string giving the name of the time variable, as appearing in `originalData`

`eventVar` Character string giving the name of the event variable, as appearing in `originalData`

`ERSPC` *Data on the men in the European Randomized Study of Prostate Cancer Screening*

Description

This data set lists the individual observations for 159,893 men in the core age group between the ages of 55 and 69 years at entry.

Usage

```
ERSPC
```

Format

A data frame with 159,893 observations on the following 3 variables:

ScrArm Whether in Screening Arm (1) or non-Screening arm (0) [numeric]

Follow.Up.Time The time, measured in years from randomization, at which follow-up was terminated

DeadOfPrCa Whether follow-up was terminated by Death from Prostate Cancer (1) or by death from other causes, or administratively (0)

Details

The men were recruited from seven European countries (centres). Each centre began recruitment at a different time, ranging from 1991 to 1998. The last entry was in December 2003. The uniform censoring date was December 31, 2006. The randomization ratio was 1:1 in six of the seven centres. In the seventh, Finland, the size of the screening group was fixed at 32,000 subjects. Because the whole birth cohort underwent randomization, this led to a ratio, for the screening group to the control group, of approximately 1 to 1.5, and to the non-screening arm being larger than the screening arm.

Source

The individual censored values were recovered by James Hanley from the Postcript code that the NEJM article (Schroder et al.) used to render Figure 2 (see Liu et al. for details). The uncensored values were more difficult to recover exactly, as the 'jumps' in the Nelson-Aalen plot are not as monotonic as first principles would imply. Thus, for each arm, the numbers of deaths in each 1-year time-bin were estimated from the differences in the cumulative incidence curves at years 1, 2, .. , applied to the numbers at risk within the time-interval. The death times were then distributed at random within each bin.

The interested reader can 'see' the large numbers of individual censored values by zooming in on the original pdf Figure, and watching the Figure being re-rendered, or by printing the graph and watching the printer 'pause' while it superimposes several thousand dots (censored values) onto the curve. Watching these is what prompted JH to look at what lay 'behind' the curve. The curve itself can be drawn using fewer than 1000 line segments, and unless on peers into the PostScript) the almost 160,000 dots generated by Stata are invisible.

References

Liu Z, Rich B, Hanley JA. Recovering the raw data behind a non-parametric survival curve. Systematic Reviews 2014 Dec 30;3:151. doi: 10.1186/2046-4053-3-151.

Schroder FH, et al., for the ERSPC Investigators.Screening and Prostate-Cancer Mortality in a Randomized European Study. N Engl J Med 2009;360:1320-8.

Examples

```
## Not run:

###  cumulative incidence plots
library(survival)
```

```
library(casebase)
data("ERSPC")
KM = survfit(Surv(Follow.Up.Time,DeadOfPrCa) ~ ScrArm, data = ERSPC)
str(KM)
par(mfrow=c(1,1),mar = c(5,5,0.1,0.1))
plot(KM$time[    1: 1501], 1-KM$surv[   1:1501], type="s", col="red" ,
     ylab = "Risk", xlab="Years since Randomization")
lines(KM$time[1502: 2923], 1-KM$surv[1502: 2923], type="s", col="green" )

###  PopulationTime plots
ds <- ERSPC
par(mfrow=c(1,1),mar = c(0.01,0.01,0.1,0.1))

plot(c(-0.5,15.75),c(-93000,80000), col="white" )
set.seed(7654321)

OFF = 2000


for(i in 0:1) {
    t=seq(0.01,14.9,0.01)
    S = function(x) sum(ds$Follow.Up.Time[ds$ScrArm==i] >= x)
    n = unlist(lapply(t,"S"))
    if(i==1) yy =  c(0,n,0) + OFF
    if(i==0) yy =  c(0,-n,0) - OFF
    polygon(c(0,t,14.9),yy,col="grey80",border=NA)

    t.d = ds$Follow.Up.Time[ds$ScrArm==i & ds$DeadOfPrCa==1]

    for( j in 1:length(t.d) ) {
        time.index =  ceiling(t.d[j]/0.01)
        nn   = n[ time.index ]
        if(i==1) h = runif(1,0.01*nn,0.99*nn)  + OFF
        if(i==0) h = runif(1,-0.99*nn,-0.01*nn) - OFF
        points(t.d[j],h, pch=19,cex=0.25,col="red")
    }
}

for (t in 1:15) text(t,0,toString(t), cex=0.75)
text(15.25,0,"Year", cex=0.75,adj=c(0,0.5))

for (n in seq(0,90000,10000)) {
    if(n> 0 & n < 80000) text(-0.1,n+OFF,format(n,big.mark=","), cex=0.75,adj=c(1,0.5))
    if(n> 0) text(-0.1,-n-OFF,format(n,big.mark=","), cex=0.75,adj=c(1,0.5))
    segments(-0.05,  n+OFF, 0, n+OFF , lwd=0.5)
    segments(-0.05, -n-OFF, 0, -n-OFF, lwd=0.5 )

}
text(4, 70000+OFF,"Screening Arm of ERSPC", cex=1,adj=c(0,0.5))
text(4,-85000-OFF,"No-Screening Arm", cex=1,adj=c(0,0.5))

text(-0.75,78000+OFF,"Number of
Men being Followed", cex=1,adj=c(0,0.5))
```

```
h = 50000+OFF
points(9.5,h, pch=19,cex=0.25,col="red")
text(9.6,h,"Death from Prostate Cancer", adj=c(0,0.5))

#The randomization of the Finnish cohorts were carried out on January 1 of
#each of the 4 years 1996 to 1999. This, coupled with the uniform December 31
#2006 censoring date, lead to large numbers of men with exactly 11, 10, 9 or 8
#years of follow-up.

#Tracked backwards in time (i.e. from right to left) , the PopulationTime
#plot shows the recruitment pattern from its beginning in 1991, and in
#particular the Jan 1 entries in successive years.

#Tracked forwards in time (i.e. from left to right), the plot for the first
#three years shows attrition due entirely to death (mainly from other causes).
#Since the Swedish and Belgian centres were the last to close their
#recruitment - in December 2003 - the minimum potential follow-up is three
#years. Tracked further forwards in time (i.e. after year 3) the attrition is
#a combination of deaths and staggered entries.

## End(Not run)
```

`fitSmoothHazard` *Fit smooth-in-time parametric hazard functions.*

Description

Miettinen and Hanley (2009) explained how case-base sampling can be used to estimate smooth-in-time parametric hazard functions. The idea is to sample person-moments, which may or may not correspond to an event, and then fit the hazard using logistic regression.

Usage

```
fitSmoothHazard(formula, data, time, censored.indicator, ...)
```

Arguments

`formula`	an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under Details.
`data`	a data frame, list or environment containing the variables in the model. If not found in data, the variables are taken from `environment(formula)`, typically the environment from which `fitSmoothHazard` is called.
`time`	a character string giving the name of the time variable. See Details.

censored.indicator
: a character string of length 1 indicating which value in event is the censored. This function will use relevel to set censored.indicator as the reference level. This argument is ignored if the event variable is a numeric

...
: Additional parameters passed to sampleCaseBase. If data inherits from the class cbData, then these parameters are ignored.

Details

The object data should either be the output of the function sampleCaseBase or the source dataset on which case-base sampling will be performed. In the latter case, it is assumed that data contains the two columns corresponding to the supplied time and event variables. If time is missing, the function looks for a column named "time" in the data. Note that the event variable is inferred from formula, since it is the left hand side.

Value

An object of glm and lm when there is only one event of interest, or of class CompRisk, which inherits from vglm, for a competing risk analysis. As such, functions like summary, deviance and coefficients give familiar results.

Examples

```
# Simulate censored survival data for two outcome types from exponential distributions
library(data.table)
set.seed(12345)
nobs <- 5000
tlim <- 20

# simulation parameters
b1 <- 200
b2 <- 50

# event type 0-censored, 1-event of interest, 2-competing event
# t observed time/endpoint
# z is a binary covariate
DT <- data.table(z=rbinom(nobs, 1, 0.5))
DT[,`:=` ("t_event" = rweibull(nobs, 1, b1),
          "t_comp" = rweibull(nobs, 1, b2))]
DT[,`:=`("event" = 1 * (t_event < t_comp) + 2 * (t_event >= t_comp),
         "time" = pmin(t_event, t_comp))]
DT[time >= tlim, `:=`("event" = 0, "time" = tlim)]

out_linear <- fitSmoothHazard(event ~ time + z, DT)
out_log <- fitSmoothHazard(event ~ log(time) + z, DT)
```

plot.popTime *Population Time Plot*

Description

plot method for objects of class popTime and popTimeExposure

Create a data frame for population time plots to give a visual representation of incidence density

Usage

```
## S3 method for class 'popTime'
plot(x, ..., xlab = "Follow-up time", ylab = "Population",
  line.width = 1, line.colour = "grey80", point.size = 1,
  point.colour = "red", legend = FALSE, legend.position = c("bottom",
  "top", "left", "right"))

## S3 method for class 'popTimeExposure'
plot(x, ..., ncol = 1, xlab = "Follow-up time",
  ylab = "Population", line.width = 1, line.colour = "grey80",
  point.size = 1, point.colour = "red", legend = FALSE,
  legend.position = c("bottom", "top", "left", "right"))

popTime(data, time, event, censored.indicator, exposure)

checkArgsTimeEvent(data, time, event)
```

Arguments

x	an object of class popTime or popTimeExposure.
...	Ignored.
xlab, ylab, line.width, line.colour, point.size, point.colour, legend, legend.position	See par.
ncol	Number of columns.
data	a data.frame or data.table containing the source dataset.
time	a character string giving the name of the time variable. See Details.
event	a character string giving the name of the event variable contained in data. See Details. If event is a numeric variable, then 0 needs to represent a censored observation, 1 needs to be the event of interest. Integers 2, 3, ... and so on are treated as competing events. If event is a factor or character and censored.indicator is not specified, this function will assume the reference level is the censored indicator
censored.indicator	a character string of length 1 indicating which value in event is the censored. This function will use relevel to set censored.indicator as the reference level. This argument is ignored if the event variable is a numeric

exposure — a character string of length 1 giving the name of the exposure variable which must be contained in data. Default is NULL. This is used to produced exposure stratified plots. If an exposure is specified, popTime returns an object of class popTimeExposure

Details

It is assumed that data contains the two columns corresponding to the supplied time and event variables. If either the time or event argument is missing, the function looks for columns that contain the words "time", "event", or "status" in them (case insensitive). The function first looks for the time variable, then it looks for the event variable. This order of operation is important if for example the time variable is named "event time" and the event variable is named "event indicator". This function will first (automatically) find the time variable and remove this as a possibility from subsequent searches of the event variable. The following regular expressions are used for the time and event variables:

time "[\s\W_]+time|^time\b"

event "[\s\W_]+event|^event\b|[\s\W_]+status|^status\b"

This allows for "time" to be preceded or followed by one or more white space characters, one or more non-word characters or one or more underscores. For example, the following column names would be recognized by the function as the "time" variable: "time of death", "death_time", "Time", "time", "diag
But the following will not be recognized: "diagtime","eventtime", "Timediag"

Value

The methods for plot return a population time plot, stratified by exposure status in the case of popTimeExposure.

The methods for plot return a population time plot, stratified by exposure status in the case of popTimeExposure.

An object of class popTime (or popTimeExposure if exposure is specified), data.table and data.frame in this order! The output of this function is to be used with the plot method for objects of class popTime or of class popTimeExposure, which will produce population time plots

See Also

plot.popTime, plot.popTimeExposure

Examples

```
## Not run:
data(bmtccr)
popTimeData <- popTime(data = bmtccr, time = "ftime", exposure = "D")
# p is an object of class gg and ggplot
p <- plot(popTimeData)
# you can further modify the object using all ggplot2 functions
# here we modify the number of y-tick labels
p + scale_y_continuous(breaks = seq(0, max(popTimeData$data$ycoord), 10))

## End(Not run)
```

`sampleCaseBase` *Create case-base dataset for use in fitting parametric hazard functions*

Description

This function implements the case-base sampling approach described in Hanley and Miettinen, Int J Biostatistics 2009. It can be used to fit smooth-in-time parametric functions easily, via logistic regression.

Usage

```
sampleCaseBase(data, time, event, ratio = 10, comprisk = FALSE,
  censored.indicator)
```

Arguments

`data`	a data.frame or data.table containing the source dataset.
`time`	a character string giving the name of the time variable. See Details.
`event`	a character string giving the name of the event variable. See Details.
`ratio`	Integer, giving the ratio of the size of the base series to that of the case series. Defaults to 10.
`comprisk`	Logical. Indicates whether we have multiple event types and that we want to consider some of them as competing risks.
`censored.indicator`	a character string of length 1 indicating which value in `event` is the censored. This function will use `relevel` to set `censored.indicator` as the reference level. This argument is ignored if the `event` variable is a numeric

Details

The base series is sampled using a multinomial scheme: individuals are sampled proportionally to their follow-up time.

It is assumed that `data` contains the two columns corresponding to the supplied time and event variables. If either the `time` or `event` argument is missing, the function looks for columns with appropriate-looking names (see `checkArgsTimeEvent`).

Value

The function returns a dataset, with the same format as the source dataset, and where each row corresponds to a person-moment sampled from the case or the base series. otherwise)

Examples

```
# Simulate censored survival data for two outcome types from exponential distributions
library(data.table)
set.seed(12345)
nobs <- 5000
tlim <- 10

# simulation parameters
b1 <- 200
b2 <- 50

# event type 0-censored, 1-event of interest, 2-competing event
# t observed time/endpoint
# z is a binary covariate
DT <- data.table(z=rbinom(nobs, 1, 0.5))
DT[,`:=` ("t_event" = rweibull(nobs, 1, b1),
          "t_comp" = rweibull(nobs, 1, b2))]
DT[,`:=`("event" = 1 * (t_event < t_comp) + 2 * (t_event >= t_comp),
         "time" = pmin(t_event, t_comp))]
DT[time >= tlim, `:=`("event" = 0, "time" = tlim)]

out <- sampleCaseBase(DT, time = "time", event = "event", comprisk = TRUE)
```

Index