Package 'casebase'

April 10, 2024

Type Package

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

Version 0.10.5

Date 2024-04-09

Description Fit flexible and fully parametric hazard regression models to survival data with single event type or multiple competing causes via logistic and multinomial regression. Our formulation allows for arbitrary functional forms of time and its interactions with other predictors for time-dependent hazards and hazard ratios. From the fitted hazard model, we provide functions to readily calculate and plot cumulative incidence and survival curves for a given covariate profile. 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. Based on 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>.

Depends R (>= 3.5.0)

Imports data.table, ggplot2 (>= 3.4.0), methods, mgcv, stats, survival, VGAM

License MIT + file LICENSE

LazyData TRUE

Suggests colorspace, covr, dplyr, eha, glmnet, knitr, lubridate, progress, rmarkdown, splines, testthat (>= 3.0.0), visreg

VignetteBuilder knitr

URL https://sahirbhatnagar.com/casebase/

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

RoxygenNote 7.3.0

Encoding UTF-8

NeedsCompilation no

Author Sahir Bhatnagar [aut, cre] (https://sahirbhatnagar.com/),
Maxime Turgeon [aut] (https://www.maxturgeon.ca/),
Jesse Islam [aut] (https://www.jesseislam.com/),
Olli Saarela [aut]
(https://www.dlsph.utoronto.ca/faculty-profile/saarela-olli/),
James Hanley [aut] (https://jhanley.biostat.mcgill.ca/)

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

Repository CRAN

Date/Publication 2024-04-10 12:00:06 UTC

R topics documented:

`absoluteRisk.CompRisk` *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.

Plot method for objects returned by the `absoluteRisk` function. Current plot types are cumulative incidence and survival functions.

Usage

```
absoluteRisk.CompRisk(
  object,
  time,
  newdata,
  method = c("numerical", "montecarlo"),
  nsamp = 100,
  onlyMain = TRUE,
  type = c("CI", "survival"),
  addZero = TRUE
)

absoluteRisk(
  object,
  time,
  newdata,
  method = c("numerical", "montecarlo"),
  nsamp = 100,
  s = c("lambda.1se", "lambda.min"),
  onlyMain = TRUE,
  type = c("CI", "survival"),
  addZero = TRUE,
  ntimes = 100,
  ...
)

## S3 method for class 'absRiskCB'
print(x, ...)

## S3 method for class 'absRiskCB'
plot(
  x,
  ...,
  xlab = "time",
 ylab = ifelse(attr(x, "type") == "CI", "cumulative incidence", "survival probability"),
  type = "l",
  gg = TRUE,
  id.names,
  legend.title
)
```

Arguments

object	Output of function fitSmoothHazard.
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. Alternatively, if newdata

	= "typical", the absolute risk will be computed at a "typical" covariate profile (see Details).
method	Method used for integration. Defaults to "numerical", which uses the trapezoidal rule to integrate over all time points together. The only other option is "montecarlo", which implements Monte-Carlo integration.
nsamp	Maximal number of subdivisions (if method = "numerical") or number of sampled points (if method = "montecarlo").
onlyMain	Logical. For competing risks, should we return absolute risks only for the main event of interest? Defaults to TRUE.
type	Line type. Only used if gg = FALSE. This argument gets passed to graphics::matplot(). Default: 'l'
addZero	Logical. Should we add time = 0 at the beginning of the output? Defaults to TRUE.
s	Value of the penalty parameter lambda at which predictions are required (for class cv.glmnet).
ntimes	Number of time points (only used if time is missing).
...	further arguments passed to matplot. Only used if gg=FALSE.
x	Fitted object of class absRiskCB. This is the result from the absoluteRisk() function.
xlab	xaxis label, Default: 'time'
ylab	yaxis label. By default, this will use the "type" attribute of the absRiskCB object
gg	Logical for whether the ggplot2 package should be used for plotting. Default: TRUE
id.names	Optional character vector used as legend key when gg=TRUE. If missing, defaults to V1, V2, ...
legend.title	Optional character vector of the legend title. Only used if gg = FALSE. Default is 'ID'

Details

If newdata = "typical", we create a typical covariate profile for the absolute risk computation. This means that we take the median for numerical and date variables, and we take the most common level for factor variables.

In general, 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.

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).

Value

If `time` was provided, returns the estimated absolute risk for the user-supplied covariate profiles. This will be stored in a matrix or a higher dimensional array, depending on the input (see details). If both `time` and `newdata` are missing, returns the original data with a new column containing the risk estimate at failure times.

A plot of the cumulative incidence or survival curve

See Also

`matplot`, `absoluteRisk`, `as.data.table`, `setattr`, `melt.data.table`

Examples

```
# Simulate censored survival data for two outcome types
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, ratio = 10)

linear_risk <- absoluteRisk(out_linear, time = 10,
                            newdata = data.table("z" = c(0,1)))
# Plot CI curves----
library(ggplot2)
data("brcancer")
mod_cb_tvc <- fitSmoothHazard(cens ~ estrec*log(time) +
                                horTh +
                                age +
                                menostat +
                                tsize +
                                tgrade +
                                pnodes +
                                progrec,
                              data = brcancer,
                              time = "time", ratio = 1)
smooth_risk_brcancer <- absoluteRisk(object = mod_cb_tvc,
```

```
                                    newdata = brcancer[c(1,50),])

class(smooth_risk_brcancer)
plot(smooth_risk_brcancer)
```

`bmtcrr` *Data on transplant patients*

Description

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

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

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.

brcancer *German Breast Cancer Study Group 2*

Description

A data frame containing the observations from the GBSG2 study. This is taken almost verbatim from the TH.data package.

Usage

```
brcancer
```

Format

This data frame contains the observations of 686 women:

horTh hormonal therapy, a factor at two levels no and yes.

hormon numeric version of horTh

age of the patients in years.

menostat menopausal status, a factor at two levels pre (premenopausal) and post (postmenopausal).

meno Numeric version of menostat

tsize tumor size (in mm).

tgrade tumor grade, a ordered factor at levels I < II < III.

pnodes number of positive nodes.

progrec progesterone receptor (in fmol).

estrec estrogen receptor (in fmol).

time recurrence free survival time (in days).

cens censoring indicator (0- censored, 1- event).

Source

Torsten Hothorn (2019). TH.data: TH's Data Archive. R package version 1.0-10. https://CRAN.R-project.org/package=TH.data

References

M. Schumacher, G. Basert, H. Bojar, K. Huebner, M. Olschewski, W. Sauerbrei, C. Schmoor, C. Beyerle, R.L.A. Neumann and H.F. Rauschecker for the German Breast Cancer Study Group (1994), Randomized 2×2 trial evaluating hormonal treatment and the duration of chemotherapy in node-positive breast cancer patients. *Journal of Clinical Oncology*, **12**, 2086–2093.

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

```
if (requireNamespace("survival", quietly = TRUE)) {
library(survival) # for veteran data
checkArgsEventIndicator(data = veteran, event = "celltype",
                        censored.indicator = "smallcell")
checkArgsEventIndicator(data = veteran, event = "status")
}
data("bmtcrr") # from casebase
checkArgsEventIndicator(data = bmtcrr, event = "Sex",
                        censored.indicator = "M")
checkArgsEventIndicator(data = bmtcrr, event = "D",
                        censored.indicator = "AML")
checkArgsEventIndicator(data = bmtcrr, event = "Status")
```

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`

confint.absRiskCB *Compute confidence intervals for risks*

Description

This function uses parametric bootstrap to compute confidence intervals for the risk estimates. Since it relies on MLE theory for the validity of these intervals, this function only works for `fit_obj` that was fitted using `family = "glm"` (i.e. the default).

Usage

```
## S3 method for class 'absRiskCB'
confint(object, parm, level = 0.95, nboot = 500, ...)
```

Arguments

object	Output of function absoluteRisk.
parm	Output of function fitSmoothHazard that was used to compute object.
level	The confidence level required.
nboot	The number of bootstrap samples to use.
...	Additional arguments for methods.

Details

If the package progress is available, the function also reports on progress of the sampling (which can take some time if there are many covariate profiles and/or time points).

Value

If there is only one covariate profile, the function returns a matrix with the time points, the risk estimates, and the confidence intervals. If there are more than one covariate profile, the function returns a list with three components.

---

eprchd *Estrogen plus Progestin and the Risk of Coronary Heart Disease (eprchd)*

Description

This data was reconstructed from the curves in figure 2 (Manson 2003).Compares placebo to hormone treatment.

Usage

```
eprchd
```

Format

A dataframe with 16608 observations and 3 variables:

time Years (continuous)

status 0=censored, 1=event

treatment placebo, estPro

References

Manson, J. E., Hsia, J., Johnson, K. C., Rossouw, J. E., Assaf, A. R., Lasser, N. L., ... & Strickland, O. L. (2003). Estrogen plus progestin and the risk of coronary heart disease. New England Journal of Medicine, 349(6), 523-534.

Examples

```
data("eprchd")
fit <- fitSmoothHazard(status ~ time + treatment, data = eprchd)
```

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 (centers). 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.

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 Population-Time 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 3 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.

Source

The individual censored values were recovered by James Hanley from the Postcript code that the NEJM article (Schroder et al., 2009) used to render Figure 2 (see Liu et al., 2014, 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; 3:151. doi:10.1186/204640533151.

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. doi:10.1056/NEJMoa0810084.

Examples

```
data("ERSPC")
set.seed(12345)
pt_object_strat <- casebase::popTime(ERSPC[sample(1:nrow(ERSPC), 10000),],
                                     event = "DeadOfPrCa",
                                     exposure = "ScrArm")

plot(pt_object_strat,
     facet.params = list(ncol = 2))
```

`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,
  family = c("glm", "gam", "glmnet"),
  censored.indicator,
  ratio = 100,
  ...
)

fitSmoothHazard.fit(
  x,
  y,
  formula_time,
  time,
  event,
  family = c("glm", "glmnet"),
  censored.indicator,
  ratio = 100,
  ...
)

prepareX(formula, data)
```

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.
family	a character string specifying the family of regression models used to fit the hazard.
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.
ratio	integer, giving the ratio of the size of the base series to that of the case series. Defaults to 100.
...	Additional parameters passed to fitting functions (e.g. `glm`, `glmnet`, `gam`).
x	Matrix containing covariates.
y	Matrix containing two columns: one corresponding to time, the other to the event type.

formula_time	A formula describing how the hazard depends on time. Defaults to linear.
event	a character string giving the name of the event variable.

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. The variable time is used for the sampling the base series, and therefore it should represent the time variable on its original (i.e. non transformed) scale. 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.

For single-event survival analysis, it is also possible to fit the hazard function using glmnet or gam. The choice of fitting family is controlled by the parameter family. The default value is glm, which corresponds to logistic regression. For competing risk analysis, only glm and glmnet are allowed.

We also provide a matrix interface through fitSmoothHazard.fit, which mimics glm.fit. This is mostly convenient for family = "glmnet", since a formula interface becomes quickly cumbersome as the number of variables increases. In this setting, the matrix y should have two columns and contain the time and event variables (e.g. like the output of survival::Surv). We need this linear function of time in order to perform case-base sampling. Therefore, nonlinear functions of time should be specified as a one-sided formula through the argument formula_time (the left-hand side is always ignored).

prepareX is a slightly modified version of the same function from the glmnet package. It can be used to convert a data.frame to a matrix with categorical variables converted to dummy variables using one-hot encoding

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)
nobs <- 500
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, ratio = 10)
out_log <- fitSmoothHazard(event ~ log(time) + z, DT, ratio = 10)

# Use GAMs
library(mgcv)
DT[event == 2, event := 1]
out_gam <- fitSmoothHazard(event ~ s(time) + z, DT,
                           ratio = 10, family = "gam")
```

---

hazardPlot *Plot Fitted Hazard Curve as a Function of Time*

Description

Visualize estimated hazard curves as a function of time with confidence intervals. This function takes as input, the result from the fitSmoothHazard() function. The user can also specify a sequence of times at which to estimate the hazard function. These plots are useful to visualize the non-proportional hazards, i.e., time dependent interactions with a covariate.

Usage

```
hazardPlot(
  object,
  newdata,
  type = c("hazard"),
  xlab = NULL,
  breaks = 100,
  ci.lvl = 0.95,
  ylab = NULL,
  line.col = 1,
  ci.col = "grey",
  lty = par("lty"),
  add = FALSE,
  ci = !add,
  rug = !add,
  s = c("lambda.1se", "lambda.min"),
  times = NULL,
  ...
)
```

Arguments

object	Fitted object of class glm, gam, cv.glmnet or gbm. This is the result from the fitSmoothHazard() function.
newdata	A data frame in which to look for variables with which to predict. This is required and must contain all the variables used in the model. Only one covariate profile can be used. If more than one row is provided, only the first row will be used.
type	Type of plot. Currently, only "hazard" has been implemented. Default: c("hazard")
xlab	x-axis label. Default: the name of the time variable from the fitted object.
breaks	Number of points at which to estimate the hazard. This argument is only used if argument times=NULL. This function will calculate a sequence of times between the minimum and maximum of observed event times. Default: 100.
ci.lvl	Confidence level. Must be in (0,1), Default: 0.95
ylab	y-axis label. Default: NULL which means the function will put sensible defaults.
line.col	Line color, Default: 1. See graphics::par() for details.
ci.col	Confidence band color. Only used if argument ci=TRUE, Default: 'grey'
lty	Line type. See graphics::par() for details, Default: par("lty")
add	Logical; if TRUE add to an already existing plot; Default: FALSE
ci	Logical; if TRUE confidence bands are calculated. Only available for family="glm" and family="gam", Default: !add
rug	Logical. Adds a rug representation (1-d plot) of the event times (only for status=1), Default: !add
s	Value of the penalty parameter lambda at which predictions are required (for class cv.glmnet only). Only the first entry will be used if more than one numeric value is provided, Default: c("lambda.1se", "lambda.min")
times	Vector of numeric values at which the hazard should be calculated. Default: NULL which means this function will use the minimum and maximum of observed event times with the breaks argument.
...	further arguments passed to graphics::matplot()

Details

This is an earlier version of a function to plot hazards. We recommend instead using the plot method for objects returned by fitSmoothHazard(). See plot.singleEventCB().

Value

a plot of the hazard function and a data.frame of original data used in the fitting along with the data used to create the plots including predictedhazard which is the predicted hazard for a given covariate pattern and time predictedloghazard is the predicted hazard on the log scale. lowerbound and upperbound are the lower and upper confidence interval bounds on the hazard scale (i.e. used to plot the confidence bands). standarderror is the standard error of the log hazard (only if family="glm" or family="gam")

See Also

fitSmoothHazard()

Examples

```
data("simdat")
mod_cb <- fitSmoothHazard(status ~ trt * eventtime,
                                    time = "eventtime",
                                    data = simdat[1:200,],
                                    ratio = 1,
                                    family = "glm")

results0 <- hazardPlot(object = mod_cb, newdata = data.frame(trt = 0),
           ci.lvl = 0.95, ci = FALSE, lty = 1, line.col = 1, lwd = 2)
head(results0)
hazardPlot(object = mod_cb, newdata = data.frame(trt = 1), ci = FALSE,
           ci.lvl = 0.95, add = TRUE, lty = 2, line.col = 2, lwd = 2)
legend("topleft", c("trt=0","trt=1"),lty=1:2,col=1:2,bty="y", lwd = 2)
```

---

plot.popTime *Population Time Plot*

Description

`plot` method for objects of class `popTime`

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",
  add.case.series = TRUE,
  add.base.series = FALSE,
  add.competing.event = FALSE,
  casebase.theme = TRUE,
  ribbon.params = list(),
  case.params = list(),
  base.params = list(),
  competing.params = list(),
  color.params = list(),
  fill.params = list(),
  theme.params = list(),
  facet.params = list(),
```

```
  ratio = 1,
  censored.indicator,
  comprisk = FALSE,
  legend = TRUE,
  ncol,
  legend.position,
  line.width,
  line.colour,
  point.size,
  point.colour
)

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

checkArgsTimeEvent(data, time, event)
```

Arguments

x	an object of class popTime or popTimeExposure.
...	Ignored.
xlab, ylab	The title of the respective axis. Default: 'Follow-up time' for xlab and 'Population' for ylab
add.case.series	Logical indicating if the case series should be added to the plot. Default: TRUE
add.base.series	Logical indicating if the base series should be added to the plot. Default: FALSE
add.competing.event	Logical indicating if the competing event should be added to the plot. Default: FALSE
casebase.theme	Logical indication if the casebase theme be used. The casebase theme uses theme_minimal. Default: TRUE.
ribbon.params	A list containing arguments that are passed to geom_ribbon which is used to plot the population-time area. These arguments will override the function defaults. For example, you can set ribbon.params = list(colour = 'green') if you want the area to be green.
case.params, base.params, competing.params	A list containing arguments that are passed to geom_point which is used to plot the case series, base series, competing events. These arguments will override the function defaults. For example, you can set case.params = list(size = 1.5) if you want to increase the point size for the case series points. Note: do not use this argument to change the color of the points. Doing so will result in unexpected results for the legend. See the color.params and fill.params arguments, if you want to change the color of the points.
color.params	A list containing arguments that are passed to scale_color_manual which is used to plot the legend. Only used if legend=TRUE. These arguments will override the function defaults. Use this argument if you want to change the color of the points. See examples for more details.

fill.params — A list containing arguments that are passed to scale_fill_manual which is used to plot the legend. Only used if legend=TRUE. These arguments will override the function defaults. Use this argument if you want to change the color of the points. See examples for more details.

theme.params — A list containing arguments that are passed to theme. For example theme.params = list(legend.position = 'none').

facet.params — A list containing arguments that are passed to facet_wrap which is used to create facet plots. Only used if plotting exposure stratified population time plots. These arguments will override the function defaults.

ratio — If add.base.series=TRUE, integer, giving the ratio of the size of the base series to that of the case series. This argument is passed to the sampleCaseBase function. Default: 10.

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

comprisk — If add.base.series=TRUE, logical indicating whether we have multiple event types and that we want to consider some of them as competing risks. This argument is passed to the sampleCaseBase function. Note: should be TRUE if your data has competing risks, even if you don't want to add competing risk points (add.competing.event=FALSE). Default: FALSE

legend — Logical indicating if a legend should be added to the plot. Note that if you want to change the colors of the points, through the color.params and fill.params arguments, then set legend=TRUE. If you want to change the color of the points but not have a legend, then set legend=TRUE and theme.params = list(legend.position = 'none'. Default: FALSE

ncol — Deprecated. Use facet.params instead.

legend.position — Deprecated. Specify the legend.position argument instead in the theme.params argument. e.g. theme.params = list(legend.position = 'bottom').

line.width — Deprecated.

line.colour — Deprecated. specify the fill argument instead in ribbon.params. e.g. ribbon.params = list(fill = 'red').

point.size — Deprecated. specify the size argument instead in the case.params or base.params or competing.params argument. e.g. case.params = list(size = 1.5).

point.colour — Deprecated. Specify the values argument instead in the color.params and fill.params argument. See examples for details.

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

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 exposure attribute which contains the name of the exposure variable in the dataset. The plot method for objects of class popTime will use this exposure attribute to create exposure stratified population time plots.

percentile_number — Default=0.5. Give a value between 0-1. if the percentile number of available subjects at any given point is less than 10, then sample regardless of case status. Depending on distribution of survival times and events event points may not be evenly distributed with default value.

Details

This function leverages the ggplot2 package to build population time plots. It builds the plot by adding layers, starting with a layer for the area representing the population time. It then sequentially adds points to the plots to show the casebase sampling mechanism. This function gives user the flexibility to add any combination of the case.series, base.series and competing events. The case series and competing events are sampled at random vertically on the plot in order to visualise the incidence density using the popTime function. That is, imagine we draw a vertical line at a specific event time. We then plot the point at a randomly sampled y-coordinate along this vertical line. This is done to avoid having all points along the upper edge of the plot (because the subjects with the least amount of observation time are plotted at the top of the y-axis). By randomly distributing them, we can get a better sense of the incidence density. The base series is sampled horizontally on the plot using the sampleCaseBase function.

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", "diagnosis_time", "time.diag", "diag__time". 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. Note that these are ggplot2 objects and can therefore be used in subsequent ggplot2 type plots. See examples and vignette for details.

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. This dataset augments the original data with the following columns:

original.time value of the time variable in the original dataset - the one specified by the time user argument to this function

original.event value of the event variable in the original dataset - the one specified by the event user argument to this function

time renames the user specified time column to time

event renames the user specified event argument to event

See Also

geom_point,geom_ribbon,theme, scale_colour_manual, scale_fill_manual, sampleCaseBase

plot.popTime

Examples

```
# change color of points
library(ggplot2)
data("bmtcrr")
popTimeData <- popTime(data = bmtcrr, time = "ftime", event = "Status")
fill_cols <- c("Case series" = "black", "Competing event" = "#009E73",
               "Base series" = "#0072B2")
color_cols <- c("Case series" = "black", "Competing event" = "black",
                "Base series" = "black")

plot(popTimeData,
  add.case.series = TRUE,
  add.base.series = TRUE,
  add.competing.event = FALSE,
  legend = TRUE,
  comprisk = TRUE,
  fill.params = list(
    name = element_blank(),
    breaks = c("Case series", "Competing event", "Base series"),
    values = fill_cols
  ),
  color.params = list(
    name = element_blank(),
    breaks = c("Case series", "Competing event", "Base series"),
    values = color_cols
  )
)
data("bmtcrr")
popTimeData <- popTime(data = bmtcrr, time = "ftime")
class(popTimeData)
popTimeData <- popTime(data = bmtcrr, time = "ftime", exposure = "D")
attr(popTimeData, "exposure")
```

plotHazardRatio *Plot Hazards and Hazard Ratios*

Description

Plot method for objects returned by the `fitSmoothHazard` function. Current plot types are hazard function and hazard ratio. The `visreg` package must be installed for `type="hazard"`. This function accounts for the possible time-varying exposure effects.

Usage

```
plotHazardRatio(x, newdata, newdata2, ci, ci.lvl, ci.col, rug, xvar, ...)

## S3 method for class 'singleEventCB'
plot(
  x,
  ...,
  type = c("hazard", "hr"),
  hazard.params = list(),
  newdata,
  exposed,
  increment = 1,
  var,
  xvar = NULL,
  ci = FALSE,
  ci.lvl = 0.95,
  rug = !ci,
  ci.col = "grey"
)

incrVar(var, increment = 1)
```

Arguments

`x`	Fitted object of class `glm`, `gam`, `cv.glmnet` or `gbm`. This is the result from the `fitSmoothHazard()` function.
`newdata`	Required for `type="hr"`. The `newdata` argument is the "unexposed" group, while the exposed group is defined by either: (i) a change (defined by the `increment` argument) in a variable in newdata defined by the `var` argument ; or (ii) an exposed function that takes a data-frame and returns the "exposed" group (e.g. `exposed = function(data) transform(data, treat=1)`). This is a generalization of the behavior of the rstpm2 plot function. It allows both numeric and factor variables to be incremented or decremented. See references for rstpm2 package. Only used for `type="hr"`
`newdata2`	`data.frame` for exposed group. calculated and passed internally to `plotHazardRatio` function

ci — Logical; if TRUE confidence bands are calculated. Only available for family="glm" and family="gam", and only used for type="hr", Default: !add. Confidence intervals for hazard ratios are calculated using the Delta Method.

ci.lvl — Confidence level. Must be in (0,1), Default: 0.95. Only used for type="hr".

ci.col — Confidence band color. Only used if argument ci=TRUE, Default: 'grey'. Only used for type="hr".

rug — Logical. Adds a rug representation (1-d plot) of the event times (only for status=1), Default: !ci. Only used for type="hr".

xvar — Variable to be used on x-axis for hazard ratio plots. If NULL, the function defaults to using the time variable used in the call to fitSmoothHazard. In general, this should be any continuous variable which has an interaction term with another variable. Only used for type="hr".

... — further arguments passed to plot. Only used if type="hr". Any of lwd,lty,col,pch,cex will be applied to the hazard ratio line, or point (if only one time point is supplied to newdata).

type — plot type. Choose one of either "hazard" for hazard function or "hr" for hazard ratio. Default: type = "hazard".

hazard.params — Named list of arguments which will override the defaults passed to visreg::visreg(), The default arguments are list(fit = x, trans = exp, plot = TRUE, rug = FALSE, alpha = 1, partial = FALSE, overlay = TRUE). For example, if you want a 95% confidence band, specify hazard.params = list(alpha = 0.05). Note that The cond argument must be provided as a named list. Each element of that list specifies the value for one of the terms in the model; any elements left unspecified are filled in with the median/most common category. Only used for type="hazard". All other argument are used for type="hr". Note that the visreg package must be installed for type="hazard".

exposed — function that takes newdata and returns the exposed dataset (e.g. function(data) transform(data, treat = 1)). This argument takes precedence over the var argument, i.e., if both var and exposed are correctly specified, only the exposed argument will be used. Only used for type="hr".

increment — Numeric value indicating how much to increment (if positive) or decrement (if negative) the var variable in newdata. See var argument for more details. Default is 1. Only used for type="hr".

var — specify the variable name for the exposed/unexposed (name is given as a character variable). If this argument is missing, then the exposed argument must be specified. This is the variable which will be incremented by the increment argument to give the exposed category. If var is coded as a factor variable, then increment=1 will return the next level of the variable in newdata. increment=2 will return two levels above, and so on. If the value supplied to increment is greater than the number of levels, this will simply return the max level. You can also decrement the categorical variable by specifying a negative value, e.g., increment=-1 will return one level lower than the value in newdata. If var is a numeric, than increment will increment (if positive) or decrement (if negative) by the supplied value. Only used for type="hr".

Details

This function has only been thoroughly tested for `family="glm"`. If the user wants more customized plot aesthetics, we recommend saving the results to a `data.frame` and using the graphical package of their choice.

Value

a plot of the hazard function or hazard ratio. For `type="hazard"`, a `data.frame` (returned invisibly) of the original data used in the fitting along with the data used to create the plots including `predictedhazard` which is the predicted hazard for a given covariate pattern and time. `predictedloghazard` is the predicted hazard on the log scale. `lowerbound` and `upperbound` are the lower and upper confidence interval bounds on the hazard scale (i.e. used to plot the confidence bands). `standarderror` is the standard error of the log hazard or log hazard ratio (only if `family="glm"` or `family="gam"`). For `type="hr"`, `log_hazard_ratio` and `hazard_ratio` is returned, and if `ci=TRUE`, `standarderror` (on the log scale) and `lowerbound` and `upperbound` of the `hazard_ratio` are returned.

References

Mark Clements and Xing-Rong Liu (2019). rstpm2: Smooth Survival Models, Including Generalized Survival Models. R package version 1.5.1. https://CRAN.R-project.org/package=rstpm2

Breheny P and Burchett W (2017). Visualization of Regression Models Using visreg. The R Journal, 9: 56-71.

See Also

`modifyList`, `fitSmoothHazard()`, `graphics::par()`, `visreg::visreg()`

Examples

```
if (requireNamespace("splines", quietly = TRUE)) {
data("simdat") # from casebase package
library(splines)
simdat <- transform(simdat[sample(1:nrow(simdat), size = 200),],
                    treat = factor(trt, levels = 0:1,
                    labels = c("control","treatment")))

fit_numeric_exposure <- fitSmoothHazard(status ~ trt*bs(eventtime),
                                        data = simdat,
                                        ratio = 1,
                                        time = "eventtime")

fit_factor_exposure <- fitSmoothHazard(status ~ treat*bs(eventtime),
                                       data = simdat,
                                       ratio = 1,
                                       time = "eventtime")

newtime <- quantile(fit_factor_exposure[["data"]][[fit_factor_exposure[["timeVar"]]]],
                    probs = seq(0.05, 0.95, 0.01))
```

```
par(mfrow = c(1,3))
plot(fit_numeric_exposure,
     type = "hr",
     newdata = data.frame(trt = 0, eventtime = newtime),
     exposed = function(data) transform(data, trt = 1),
     xvar = "eventtime",
     ci = TRUE)

#by default this will increment `var` by 1 for exposed category
plot(fit_factor_exposure,
     type = "hr",
     newdata = data.frame(treat = factor("control",
               levels = c("control","treatment")), eventtime = newtime),
     var = "treat",
     increment = 1,
     xvar = "eventtime",
     ci = TRUE,
     ci.col = "lightblue",
     xlab = "Time",
     main = "Hazard Ratio for Treatment",
     ylab = "Hazard Ratio",
     lty = 5,
     lwd = 7,
     rug = TRUE)


# we can also decrement `var` by 1 to give hazard ratio for control/treatment
result <- plot(fit_factor_exposure,
               type = "hr",
               newdata = data.frame(treat = factor("treatment",
                                    levels = c("control","treatment")),
                                    eventtime = newtime),
               var = "treat",
               increment = -1,
               xvar = "eventtime",
               ci = TRUE)

# see data used to create plot
head(result)
}
```

---

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 (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.

Warning

The offset is calculated using the total follow-up time for all individuals in the study. Therefore, we need time to be on the original scale, not a transformed scale (e.g. logarithmic). Otherwise, the offset and the estimation will be wrong.

Examples

```
# Simulate censored survival data for two outcome types from exponential
library(data.table)
```

```
set.seed(12345)
nobs <- 500
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)
```

`simdat` *Simulated data under Weibull model with Time-Dependent Treatment Effect*

Description

This simulated data is and description is taken verbatim from the `simsurv`.

Usage

```
simdat
```

Format

A dataframe with 1000 observations and 4 variables:

id patient id

eventtime time of event

status event indicator (1 = event, 0 = censored)

trt binary treatment indicator

Details

Simulated data under a standard Weibull survival model that incorporates a time-dependent treatment effect (i.e. non-proportional hazards). For the time-dependent effect we included a single binary covariate (e.g. a treatment indicator) with a protective effect (i.e. a negative log hazard ratio), but we will allow the effect of the covariate to diminish over time. The data generating model will be

$$h_i(t) = \gamma\lambda(t^{\gamma-1})exp(\beta_0 X_i + \beta_1 X_i x log(t))$$

where where Xi is the binary treatment indicator for individual i, λ and γ are the scale and shape parameters for the Weibull baseline hazard, β_0 is the log hazard ratio for treatment when t=1 (i.e. when log(t)=0), and β_1 quantifies the amount by which the log hazard ratio for treatment changes for each one unit increase in log(t). Here we are assuming the time-dependent effect is induced by interacting the log hazard ratio with log time. The true parameters are 1. β_0 = -0.5 2. β_1 = 0.15 3. λ = 0.1 4. γ = 1.5

Source

See `simsurv` vignette: https://cran.r-project.org/package=simsurv/vignettes/simsurv_usage.html

References

Sam Brilleman (2019). simsurv: Simulate Survival Data. R package version 0.2.3. https://CRAN.R-project.org/package=simsurv

Examples

```
if (requireNamespace("splines", quietly = TRUE)) {
library(splines)
data("simdat")
mod_cb <- casebase::fitSmoothHazard(status ~ trt + ns(log(eventtime),
                                                      df = 3) +
                                    trt:ns(log(eventtime),df=1),
                                    time = "eventtime",
                                    data = simdat,
                                    ratio = 1)
}
```

`support` *Study to Understand Prognoses Preferences Outcomes and Risks of Treatment (SUPPORT)*

Description

The SUPPORT dataset tracks four response variables: hospital death, severe functional disability, hospital costs, and time until death and death itself. The patients are followed for up to 5.56 years. Data included only tracks follow-up time and death.

Usage

```
support
```

Format

A dataframe with 9104 observations and 34 variables after imputation and the removal of response variables like hospital charges, patient ratio of costs to charges and micro-costs. Ordinal variables, namely functional disability and income, were also removed. Finally, Surrogate activities of daily living were removed due to sparsity. There were 6 other model scores in the data-set and they were removed; only aps and sps were kept.

Age Stores a double representing age.

death Death at any time up to NDI date: 31DEC94.

sex 0=female, 1=male.

slos Days from study entry to discharge.

d.time days of follow-up.

dzgroup Each level of dzgroup: ARF/MOSF w/Sepsis, COPD, CHF, Cirrhosis, Coma, Colon Cancer, Lung Cancer, MOSF with malignancy.

dzclass ARF/MOSF, COPD/CHF/Cirrhosis, Coma and cancer disease classes.

num.co the number of comorbidities.

edu years of education of patient.

scoma The SUPPORT coma score based on Glasgow D3.

avtisst Average TISS, days 3-25.

race Indicates race. White, Black, Asian, Hispanic or other.

hday Day in Hospital at Study Admit

diabetes Diabetes (Com 27-28, Dx 73)

dementia Dementia (Comorbidity 6)

ca Cancer State

meanbp Mean Arterial Blood Pressure Day 3.

wblc White blood cell count on day 3.

hrt Heart rate day 3.

resp Respiration Rate day 3.

temp Temperature, in Celsius, on day 3.

pafi PaO2/(0.01*FiO2) Day 3.

alb Serum albumin day 3.

bili Bilirubin Day 3.

crea Serum creatinine day 3.

sod Serum sodium day 3.

ph Serum pH (in arteries) day 3.

glucose Serum glucose day 3.

bun BUN day 3.

urine urine output day 3.

adlp ADL patient day 3.

adlsc Imputed ADL calibrated to surrogate, if a surrogate was used for a follow up.

sps SUPPORT physiology score

aps Apache III physiology score

Details

Some of the original data was missing. Before imputation, there were a total of 9105 individuals and 47 variables. Of those variables, a few were removed before imputation. We removed three response variables: hospital charges, patient ratio of costs to charge,s and patient micro-costs. Next, we removed hospital death as it was directly informative of our event of interest, namely death. We also removed functional disability and income as they are ordinal covariates. Finally, we removed 8 covariates related to the results of previous findings: we removed SUPPORT day 3 physiology score (`sps`), APACHE III day 3 physiology score (`aps`), SUPPORT model 2-month survival estimate, SUPPORT model 6-month survival estimate, Physician's 2-month survival estimate for pt., Physician's 6-month survival estimate for pt., Patient had Do Not Resuscitate (DNR) order, and Day of DNR order (<0 if before study). Of these, `sps` and `aps` were added on after imputation, as they were missing only 1 observation. First we imputed manually using the normal values for physiological measures recommended by Knaus et al. (1995). Next, we imputed a single dataset using **mice** with default settings. After imputation, we noted that the covariate for surrogate activities of daily living was not imputed. This is due to collinearity between the other two covariates for activities of daily living. Therefore, surrogate activities of daily living was removed.

Source

Available at the following website: https://biostat.app.vumc.org/wiki/Main/SupportDesc. note: must unzip and process this data before use.

References

Knaus WA, Harrell FE, Lynn J et al. (1995): The SUPPORT prognostic model: Objective estimates of survival for seriously ill hospitalized adults. Annals of Internal Medicine 122:191-203. doi:10.7326/0003481912231995020100007.

http://biostat.mc.vanderbilt.edu/wiki/Main/SupportDesc

http://biostat.mc.vanderbilt.edu/wiki/pub/Main/DataSets/Csupport.html

Examples

```
data("support")
# Using the matrix interface and log of time
x <- model.matrix(death ~ . - d.time - 1, data = support)
y <- with(support, cbind(death, d.time))

fit_cb <- casebase::fitSmoothHazard.fit(x, y, time = "d.time",
                                        event = "death",
                                        formula_time = ~ log(d.time),
                                        ratio = 1)
```

Index