Package 'ems'

March 18, 2020

Title Epimed Solutions Collection for Data Editing, Analysis, and Benchmark of Health Units

Version 1.3.2

Date 2020-03-18

Description Collection of functions for data analysis and editing of clinical and epidemiological data.
Most of them are related to benchmark with prediction models.

Depends R (>= 3.2.2)

Imports boot, survival, rms

License GPL (>= 2)

LazyData true

RoxygenNote 6.1.1

NeedsCompilation no

Author Lunna Borges [aut, cre]

Maintainer Lunna Borges <lunna.borges@epimedsolutions.com>

Repository CRAN

Date/Publication 2020-03-18 21:10:02 UTC

R topics documented:

`breastCancer` *German Breast Cancer Dataset*

Description

A dataset containing variables related with breast cancer from german population.

Usage

```
breastCancer
```

Format

A data frame with 686 rows and 14 variables

Details

- `age` Patient age (years).
- `meno` Menopausal status (0 = premeno, 1 = postmeno).
- `size` Tumour size (mm).
- `gradd1` 1 = tumour grade 2 or 3; 0 = grade 1.
- `gradd2` 1 = tumour grade 3; 0 = grade 1 or 2.
- `nodes` Number of positive lymph nodes.
- `enodes` emp(-0.12 x `nodes`).
- `pgr` Progesterone receptor status (fmol 1^-1).
- `er` Oestrogen reeceotor status (fmol 1^-1).
- `hormon` Tamoxifen treatment (0 = no; 1 = yes).
- `rectime` Time (days) to death or cancer recurrence.
- `censrec` Censoring (0 = censored; 1 = event).

Source

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

`calcurve` *Calibration Curve*

Description

`calcurve` function returns a data.frame containing the number of patients, the observed mortality rate and the predicted mortality rate for each category of the predicted mortality rate. If any other acute physiology score is given, the function will also return the mortality rate predicted by this score for each category.

Usage

```
calcurve(deaths, pred, score = NULL, name_score = "Saps3",
  other_score = NULL, name_other_score = NULL,
  categories_option = c("predicted", "score"), table = FALSE,
  plot = TRUE, title_label = "Calibration Curve",
  y1axis_label = "Patients (n)", y2axis_label = "Mortality Rate (%)",
  score_color = c("#cac7cc", "#ffc341", "#33cca3"),
  bar_color = "#1f77b4", points = c(19, 18, 17))

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

## S3 method for class 'calcurve'
plot(x, ..., main = x$title_label,
  text = x$y2axis_label, ylab = x$y1axis_label, col = c(x$bar_color,
  x$score_color), pch = x$points)
```

Arguments

`deaths`	a numerical vector that only contains 0 and 1, indicating whether the patient was alive or dead, respectively.
`pred`	a numerical vector that contains the mortality rate predicted by the main score, in percentage, for each patient.
`score`	a numerical vector that contains the main score punctuation for each patient, or NULL.
`name_score`	a character string which determines the name of the main score.
`other_score`	a list of numerical vectors, where each vector contains the mortality rate predicted by other score, in percentage, for each patient, or NULL (the default).
`name_other_score`	if other_score variable is different from NULL, this argument must be a vector with the name(s) of the score(s) given.
`categories_option`	a character string which determines if the categories will refer to the main score or to the predicted mortality rate. Accepted values are 'predicted' (the default) or 'score'.

table	logical; if TRUE prints the data.frame.
plot	logical; if TRUE (the default) plots the categories chosen versus the mortality rates in the secondary vertical axis. The main vertical axis refers to the number of patients in each category, represented by the bars.
title_label	main title for calcurve.
y1axis_label, y2axis_label	labels of the main vertical axis and the secondary axis, respectively, for calcurve.
score_color	a vector with the colors to be used in the score traces for calcurve.
bar_color	color of the bars for calcurve.
points	a vector with markers types of the scores for calcurve.
x	an object of class 'calcurve'.
...	further arguments passed to plot.
main	main title for plot.calcurve.
text	label of the secondary vertical axis for plot.calcurve.
ylab	label of the main vertical axis for plot.calcurve.
col	character vector with the colors of the bars and score traces, in this order, for plot.calcurve.
pch	a vector with markers types of the scores for plot.curve.

Details

- If categories_option = 'score', the categories will refer to the deciles of the main score punctuation. If categories_option = 'predicted', the categories will refer to fixed intervals of the predicted mortality rate.

Author(s)

Camila Cardoso

Examples

```
# Loading the dataset
data(icu)

# Calibration Curve Plot
a <- calcurve(deaths = icu$UnitDischargeName,
pred = icu$Saps3DeathProbabilityStandardEquation,
score = icu$Saps3Points, name_score = 'Saps3',
categories_option = 'score', table = FALSE, plot = TRUE)
```

dataquality *Collection of functions to check data quality in a dataset and remove not valid or extreme values.*

Description

These functions return the counts and fractions of expected values, unexpected values, missing values and not valid values. They are able to do it with factor variables, numeric variables and date variables. `t_factor`, `t_num`, and `t_date` do the job for a single variable and have simpler arguments, while `factor.table`, `num.table`, and `date.table` do the job for several variables at once. `rm.unwanted` cheks the factor and numeric variables and remove the not valid or extreme values. This approach is attractive before data imputation. They all return a `data.frame`.

`t_factor` and `factor.table` will try to get factor or character variables and check how much of its content match with the expectd. They will try to treat the levels or cells containing " " as `NAs`.

`t_num` will try to get a numeric variable (even if it is currently formated as character or factor) and check how much of its content is expected (match a desired range), unexpected, non-numeric values and missing vlaues. `num.table` does the same, but for two or more variables at once.

`t_date` will try to get a date variable (even if it is currently formated as character or factor) and check how much of its content is expected (match a desired range), unexpected, non-date values and missing vlaues. `date.table` does the same, but for two or more variables at once.

`rm.unwanted` will chek in data the variables specified in the limits object according to the limits specified for each variable. If there are levels considered not valid in a factor variable, these levels are deleted. For example, if Sex is expected to be "M" and "F", and there is also an "I" level in data, every "I" is replaced by `NA`. Similarly, misspelled levels will be understood as non-valid levels and coercerd to `NA`, with the exception of leading or trailing empty spaces and lower and upper cases diferences if `try.keep = TRUE`. If there is a continuous numeric variable and it is expected to have values ranging from 30 to 700, the values outside this range, i.e. higher then 700 or lower then 30, are replaced by `NA`. Non-numeric elements, i.e. non-valid elements that should be numeric, will also be coerced to `NA`. If a varible is specified in `num.limits`, then it will be returned as a numeric variable, even if it was formated as factor or character. If a variable is specified in limits, the returnig format will depend on the `stringAsFactors` argument, unless it is formated as logical. In this case it is skipped. The arguments `limits` and `num.limits` may be `NULL`, meaning that the factor-character variables or the numeric variables , respectively, will not be edited.

Usage

```
t_factor(data, variable, legal, var.labels = attr(data,
  "var.labels")[match(variable, names(data))], digits = 3)

factor.table(data, limits, var.labels = attr(data,
  "var.labels")[match(unlist(sapply(seq_along(limits), function(i)
  limits[[i]][1])), names(data))], digits = 3)

t_num(data, num.var, num.max = 100, num.min = 0,
  var.labels = attr(data, "var.labels")[match(num.var, names(data))],
  digits = 3)
```

```
num.table(data, num.limits, var.labels = attr(data,
  "var.labels")[match(num.limits$num.var, names(data))], digits = 3)

t_date(data, date.var, date.max = as.Date("2010-11-30"),
  date.min = as.Date("2010-01-31"), format.date = "auto", digits = 3,
  var.labels = attr(data, "var.labels")[match(date.var, names(data))])

date.table(data, date.limits, format.date = "auto", digits = 3,
  var.labels = attr(data, "var.labels")[match(date.limits$date.var,
  names(data))])

rm.unwanted(data, limits = NULL, num.limits = TRUE, try.keep = TRUE,
  stringAsFactors = TRUE)
```

Arguments

`data`	A data.frame where variables will be tested.
`variable`	A character vector of length one, indicating the name of the variable in the dataset to be tested.
`legal`	A character vector representeing the expected levels of the tested variable.
`var.labels`	Variables labels for a nice output. Must be informed in the same order as variable argument. By default, it captures the labels stored in attr(data, "var.labels"), if any. If not informed, the function returns the variables names.
`digits`	Number of decimal places for rounding.
`limits`	a list of two or more lists, each containing the arguments variable name and legal levels (in this order), to check on the factor variables. In the case of `rm.unwanted`, if left NULL, it means no numeric variable will be checked. See examples.
`num.var`	A character vector indicating the name of a variable that should be numeric (although it can yet be formated as character or factor).
`num.max, num.min`	The maximal and minimal limits of acceptable range of a numeric variable.
`num.limits`	A data.frame with the following variables: num.var, num.max and num.min, representing the numeric variables names, maximal and minimal expected valid values. In the case of `rm.unwanted`, if left NULL, it means no numeric variable will be checked. See example.
`date.var`	A character vector indicating the name of a variable in data that should be a date (althoug it can yet be formated as character or factor).
`date.max, date.min`	The maximal and minimal limits of acceptable range of a date variable.
`format.date`	Default is "auto". If so, `t_date` will use `f.date` to detect the date format and format it as date. If not set to "auto", it should be a date format to be passed to `as.Date` format argument. If `format.date` is misspecified, then `t_date` and `date.table` will identify all dates as non-dates. For `date.table`, if it is set to

'auto' , it will use f.date to detect the date format and format it as date. If different from 'auto', one should specify the desired date formats in the date.limits data.frame. See example.

date.limits A data.frame with the following variables: date.var, date.max, date.min, and (optionaly) format.date. These represent values of the arguments above. See example.

try.keep Default is TRUE. If TRUE, remove.unwanted will first trim all empty spaces and transform all levels to lower case characters before comparing the found levels and expected levels of a character/factor variable. Therefore, found levels such as "yes " will be considered identical to the expected level "Yes", and will not be coerced to NA.

stringAsFactors

In rm.unwanted, if set to TRUE, the default value, variables in the limits argument that are character and numeric variables in data will be returned as factors. Logical variables are skipped. However, a variable will be returned as logical if it is originally a factor but its final levels are TRUE and FALSE and stringAsFactors = FALSE.

Author(s)

Lunna Borges & Pedro Brasil

See Also

miscellaneous

Examples

```
# Simulating a dataset with 5 factor variables and assigning labels
y <- data.frame(Var1 = sample(c("Yes","No", "Ignored", "", "yes ", NA), 200, replace = TRUE),
                Var2 = sample(c("Death","Discharge", "", NA), 200, replace = TRUE),
                Var3 = sample(c(16:35, NA), 200, replace = TRUE),
                Var4 = sample(c(12:300, "Female", "", NA), 200, replace = TRUE),
                Var5 = sample(c(60:800), 200, replace = TRUE))
attr(y, "var.labels") <- c("Intervention use","Unit destination","BMI","Age","Cholesterol")
summary(y)

# Cheking the quality only the first variable
t_factor(y, "Var1", c("Yes","No","Ignored"))

# Checking two or more variables at once
factor.limits  = list(list("Var1",c("Yes","No")),
                      list("Var2",c("Death","Discharge")))
factor.table(y, limits = factor.limits)

# Checking only one variable that shohuld be numeric
t_num(y,"Var3", num.min = 17, num.max = 32)

# Making the limits data.frame
num.limits <- data.frame(num.var = c("Var3","Var4","Var5"),
```

```
              num.min = c(17,18,70), num.max = c(32,110,300))
num.limits

# Checking two or more numeric variables (or the ones that
#          should be as numeric) at once
num.table(y, num.limits)

# Removing the unwanted values (extremes or not valid).
y <- rm.unwanted(data = y, limits = factor.limits,
                           num.limits = num.limits)
summary(y)

rm(y, num.limits, factor.limits)
#'
# Loading a dataset and assigning labels
data(icu)
attr(icu, "var.labels")[match(c("UnitAdmissionDateTime","UnitDischargeDateTime",
   "HospitalAdmissionDate", "HospitalDischargeDate"), names(icu))] <-
   c("Unit admission","Unit discharge","Hospital admission","Hospital discharge")

# Checking only one variable that should be a date.
t_date(icu, "HospitalDischargeDate", date.max = as.Date("2013-10-30"),
                                     date.min = as.Date("2013-02-20"))

# Checking a date variable misspecifying the date format
# will cause the variable dates to be identified as non-date values.
t_date(data = icu, date.var = "HospitalDischargeDate",
                   date.max = as.Date("2013-10-30"),
                   date.min = as.Date("2013-02-20"),
                   format.date = "%d/%m/%Y")

# Making a limit data.frame assuming an 'auto' format.date
d.lim <- data.frame(date.var = c("UnitAdmissionDateTime","UnitDischargeDateTime",
                   "HospitalAdmissionDate","HospitalDischargeDate"),
                   date.min = rep(as.Date("2013-02-28"), 4),
                   date.max = rep(as.Date("2013-11-30"), 4))
d.lim

# Checking two or more date variables (or the ones that should be as date) at once
date.table(data = icu, date.limits = d.lim)

# Making a limit data.frame specifying format.date argument
# Here the the last 'format.date' is missspecified on purpose
# So, the last date will be identified as non-date values.
d.lim <- data.frame(date.var = c("UnitAdmissionDateTime","UnitDischargeDateTime",
         "HospitalAdmissionDate","HospitalDischargeDate"),
          date.min = rep(as.Date("2013-02-28"), 4),
          date.max = rep(as.Date("2013-11-30"), 4),
          format.date = c(rep("%Y/%m/%d",3), "%Y-%m-%d"))
d.lim

# Checking the quality of date variable with new limits.
# The 'format.date = ""' is required to force the function to look the format
```

```
# into the date.limits data.frame
date.table(data = icu, date.limits = d.lim, format.date = "")

rm(icu, d.lim)
```

---

funnel *Funnel plot for benchmarking health units*

Description

Produces a variety of funnel plots comparing health units or ICUs (intensive care units) making easy to identify those units which deviate from the group. There is a function that calculates all the values required and returns the values for all units and the funnel, and there is a function that calls graphical parameters from the former values. The options of funnels available are the funnel for rate, for ratio of rates, for proportions, for difference of proportions and for ratio of proportions.

The funnel for rates are usually plots of either SMR or SRU at vertical axis. If the direct method is chosen, the horizontal axis will display the number of admissions. If the indirect method is chosen instead, the expected number of deaths will be displayed for SMR or the expected length of stay for SRU. As consequence of this differentiation, the interpretation regarding the clssification of the points displayed will be the same in every case.

The funnel for ratio of rates are usually plots of ratios of SMRs (or SRUs) within the same units. These two SMRs are, for example, from the same units in different time periods. Therefore, it expresses how the SMR changed over time. If the number of expected deaths is different in both periods, the plot will return at the horizontal axis a parametrization of the geometric mean of the expected number of deaths in both periods for each unit. If the number of expected deaths is identical in both periods, the plot will return at the horizontal axis the arithmetic mean of the observed number of deaths in both periods for each unit.

The funnel for proportions plots on the vertical axis the percentage of observed deaths of the units and on the horizontal axis the number (volume) of admissions. The funnel for ratio of proportions and for difference of proportions are usually used to express the fraction of deaths of the same units in different time period. Therefore, they express how the fraction of deaths changed over time in each unit. If one picks the difference of proportions, the horizontal axis will display a parametrization of the arithmetic mean of the number of admissions in both periods. If one picks the ratio of proportions, the horizontal axis will display a parametrization of the geometric mean of the number of admissions in both periods.

Usage

```
funnel(unit, y, n, n1, n2, o, o1, o2, e, e1, e2,
  lambda1 = sum(o1)/sum(n1), lambda2 = sum(o2)/sum(n2),
  pi1 = sum(o1)/sum(n1), pi2 = sum(o2)/sum(n2), y.type = c("SMR",
  "SRU"), p = c(0.95, 0.998), theta, method = c("normal", "exact"),
  direct = FALSE, myunits = rep(0, length(unit)), option = c("rate",
  "ratioRates", "prop", "diffProp", "ratioProp"), printUnits = TRUE,
  plot = TRUE, digits = 5, overdispersion = FALSE, ...)
```

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

## S3 method for class 'funnel'
plot(x, ..., col = c("darkblue", "paleturquoise3",
  "gray26"), lwd = 2, lty = c(2, 6, 1), bty = "n", pch = 21,
  pt.col = "white", bg = "orange", pt.cex = 1.5,
  auto.legend = TRUE, text.cex = 0.7, text.pos = NULL,
  mypts.col = "darkblue", printUnits = x$printUnits, xlab = x$xlab,
  ylab = x$ylab, xlim = x$xlim, ylim = x$ylim)

rateFunnel(unit, y, n, o, e, y.type, p, theta = 1, method = c("exact",
  "normal"), direct, ..., printUnits, auto.xlab = TRUE,
  xlab = c("Volume of cases", "Expected values"), ylab = y.type[1],
  xlim = c(0, max(rho)), ylim = c(min(lowerCI[[which(p == max(p))]]),
  max(upperCI[[which(p == max(p))]])), myunits, digits, overdispersion)

changeRateFunnel(unit, n1, n2, o1, e1, o2, e2, lambda1, lambda2, y.type, p,
  ..., printUnits, auto.xlab = TRUE, xlab = c("Average observed count",
  "Expectation per period"), auto.ylab = TRUE,
  ylab = c(paste0(y.type[1], "'s Ratio"), paste0("Log(", y.type[1],
  "'s Ratio)")), ylim = c(max(lowerCI[[which(p == max(p))]]) - 1.5 *
  theta, min(upperCI[[which(p == max(p))]]) + 1.5 * theta), xlim = c(0,
  max(rho)), myunits, digits, overdispersion)

propFunnel(unit, o, n, theta, p, method = c("exact", "normal"), ...,
  printUnits, ylab = "%", xlab = "Volume", ylim = c(0,
  min(upperCI[[which(p == max(p))]]) + 2.5 * theta), xlim = c(0, max(n)),
  myunits, digits, overdispersion)

changePropFunnel(unit, o1, o2, n1, n2, p, pi1, pi2, method = c("diff",
  "ratio"), ..., printUnits, xlab = "Sample size per period",
  auto.ylab = TRUE, ylab = c("Proportions difference",
  "Proportions ratio log"), ylim = c(max(lowerCI[[which(p == max(p))]]) -
  6 * theta, min(upperCI[[which(p == max(p))]]) + 6 * theta), xlim = c(0,
  max(rho)), myunits, digits, overdispersion)
```

Arguments

`unit`	A factor vector representing the unit names.
`y`	A numeric vector representing the "Standardized rate" for each unit, usually the SMR (Standardized Mortality Ratio), or possibly the SRU (Standardized Resource Use), accordind to `y.type` . It's also called "indicator".
`n`	A numeric vector representing the case volume, or number of admissions, for each unit.
`n1, n2`	If one picks `option = "ratioRates"` or `option = "diffProp"` or `option = "ratioProp"`, then `n1` and `n2` are numeric vectors representing the total of admissions at 1st

and 2nd periods, respectively.

`o` A numeric vector representing the observed death. Acceptable values are 0 (absence) or 1 (presence).

`o1, o2` If one picks `option = "ratioRates"` or `option = "diffProp"` or `option = "ratioProp"`, then `o1` and `o2` are numeric vectors representing the observed deaths at 1st and 2nd periods, respectively.

`e` Used only when `option = "rate"` and `direct = FALSE`. This is a numeric vector representing the expected number of deaths.

`e1, e2` If one picks `option = "ratioRates"`, `e1` and `e2` are numeric vectors representing the expected number of deaths at 1st and 2nd periods, respectively.

`lambda1, lambda2`
Values correponding to the rate at which a death occurs in the institutions at the 1st and 2nd periods, respectively. It is assumed that the parameters `o1` and `o2` are distributed as `oi` ~ Poisson(`lambdai`) when `option = "ratioRates"`. The default value for `lambdai` is `lambdai = sum(oi)/sum(ni)`, where ni is the value of the parameter `n1` or `n2` when i equals 1 or 2. `lambdai` is the estimate for the mean of the poisson distribution.

`pi1, pi2` Values correponding to the probability for the occurrence of a death in the institutions at the 1st and 2nd periods, respectively. Its assumed that the paramenters `o1` and `o2` are distributed as `oi` ~ Bin(`pii,ni`) when `option = "diffProp"` or `option = "ratioProp"`. `ni` is the value of the parameter n1 or n2 when i equals 1 or 2. The default value for `pii` is `sum(oi)/sum(ni)`, the estimate for the mean of the poisson distribution.

`y.type` A character vector representing the indicator type. It is used to name the vertical axis if `option = "rate"` or `option = "ratioRate"` and ignored otherwise. It usually is 'SMR' or 'SRU'.

`p` A confidence level numeric vector. The function will return a confidence interval for each value in p. The default is 2 and 3 standard deviations (`p = c(.95,998)`).

`theta` The target value which specifies the desired expectation for institutions considered "in control". Used when `option = "prop"` or `option = "rate"`. Usually, this function internally estimates a theta to represent a central tendency of the group. However, one may want to set a pre-specified value for theta to indicate a "baseline" parameter for comparison (e.g 1 for `option = "rate"` or .20 for `option = "prop"`). If this is the case, the horizontal line representing theta may not be centralized in the funnel or may be even outside the funnel, making the plot look unusual.

`method` There are two kinds of approximations for the CI, as mentioned in `direct` parameter. The one from the exact distribuition (binomial or poisson) and the one from the normal distribution. So, `method` is a character vector representing the kind of approximation desired, being `"exact"` (default) or `"normal"` the two options. It is used when `option = "rate"` or `option = "prop"`. The original report makes no formal comparison of which method is best, however it is mentioned that the funnels from different methods should look identical or very similar if all units have 100 or more observations. If any unit has less, the funnel from the normal approximation may mislead the interpretation. See details.

direct	Logical (default = FALSE); Used when option = "rate". If TRUE, we assume the vector of rates "y" is being reported as a rate per (say) 1000 individuals, and that it has been transformed to a proportion between 0 and 1. The associated error (horizontal axis) will be measured accordingly to the size of the populations n. The CI - confidence interval - is calculated by a binomial distribution. If FALSE, the associated error will be measured accordingly to the expected number of deaths e. The CI is calculated by a poisson distribution instead. See details.
myunits	A numeric vector coded with 0 and 1 indicating which units one would like to benchmark among all units. These will be highlighted with dots of different collors in the plot.
option	A character specifying the type of funnel plot one wants to produce. It can assume "rate", "ratioRates", "prop", "diffProp" or "ratioProp". If option = "rate", funnel plots a standardized rate y versus the expected number of deaths or case volume (number of unit admissions) for all units. If option = "ratioRate", funnel can be used to compare units at two diferent periods. It plots a ratio of rates y versus a precision parameter rho. If option = "prop", funnel plots a proportion y versus its case volume (number of admissions). If option = "ratioProp" or option = "diffProp", funnel can be used to compare units at two diferent periods. It plots a ratio (or difference) of proportions y versus a precision parameter rho. See details.
printUnits	Logical (default = TRUE); If TRUE, the units are identified in the plot and printed in de console. The numbers plotted correspond to the row numbers printed in the console.
plot	Logical; If TRUE (default), the correspondent graphic is plotted with the standard options.
digits	Integer indicating the number of decimals to be used in the output.
overdispersion	Logical (default = FALSE); If TRUE, introduces an multiplicative over-dispersion factor phi that will inflate the CI null variance. See details.
...	Further arguments passed to plot.
x	An object of class 'funnel'.
col	A character vector representing the colors for the CI funnel lines. Must have same length of p + 1 with the target line color in the last position.
lwd	A positive number specifying the lines width. It's the same for all lines in the plot. See par.
lty	A numeric vector representing the CI lines types. See par.
bty	A character string which represents the type of box which is drawn around plots. See par.
pch	Either an integer or a single character specifying a symbol to be used as the default in plotting points. See points for possible values and their interpretation. Note that only integers and single-character strings can be set as a graphics parameter (and not NA nor NULL).
pt.col	A character specifying the points colors.
bg	A character specifying the color to be used for the points background when pch = 21 (default). See par.

pt.cex	A numerical value giving the amount by which plotting points should be magnified relative to the default. See par.
auto.legend	Logical; If TRUE (default), prints a legend with default arguments.
text.cex	A numerical value giving the amount by which plotting text should be magnified relative to the default. See par.
text.pos	A position specifier for numbers that correspond to the units in the plot. Values of 1, 2, 3 and 4, respectively indicate positions below, to the left of, above and to the right of the points.
mypts.col	A character representing the color used to benchmark the units specified in myunits.
xlab, ylab	A title for the x and y axis. See title
xlim, ylim	Limits of horizontal and vertical axis. These limits are defined in the funnel plot and passed to plot.funnel. The user may redefine the limits in plot.funnel. Ultimately, these arguments are passed to plot.default.
auto.xlab, auto.ylab	Logical. If TRUE, one is not able to change x and y axis labels, respectively.

Details

- For every possible value of option, if overdispersion = TRUE, the CI can be inflated by a overdispersion parameter phi. There is a test for overdispersion which inflates the funnel if it's necessary. An "Winsorized" over-dispersion parameter is estimated and is used to inflate the funnel limits if it is significantly greater than 1. The parameter phi is returned as an funnel object.
- If option = "rate", funnel plots a standardized rate y versus the expected number of deaths or volume value for several units.

  To choose the direct argument, one should pay attention if one wants to use a Direct or Indirect Standardized Rate. If direct, we assume the rate is reported as a rate per (say) 1000 individuals, then it is treated as a proportion. If indirect, we assume it is a cross-sectional data that leads to a standardized event ratio.

  In many circumstances we can assume an exact or approximate normal distribution for the data. Using the method argument, one could choose between "exact" or "normal". For direct standardized rates, the exact distribuition is binomial and for indirect standardized rates, the exact distribuition is poisson. Assume rho is the precision parameter (volume, for direct rates; expected value, for indirect rates). The original report claims that, for rho > 100, the normal and exact curves almost coincide. So, one could perfectly use normal approximation if ones data parameter precision is greater than 100, in general.

  The console warns if there are units with volume/expected value less than 100.

  phi = (1/total) * sum((y - theta) ^ 2 * rho)/g(theta)

  var(y|theta,rho) = (phi * g(theta))/rho
- If option = "ratioRate", funnel can be used to compare units at two diferent periods. It plots a ratio of rates y versus a precision parameter rho.

  Suppose we have two measures for each institution: O1; E1 in a baseline period and O2; E2 in a subsequent period, and we wish to assess the change in the underlying rate (SMR or SRU). We shall only consider the ratio of rates option. The exact method will automatically

be applied if E1 = E2, and the indirect method, of normal approximations, otherwise. On this second method, for low (especially zero) counts the `funnel` function adds 0.5 to all parameters O and E in order to stabilize the estimates.

Y = (O1/E1)/(O2/E2) and the target theta = lambda2/lambda1.

When E1 = E2, y is plotted versus the average observed count (rho).

When E1 is different of E2, i.e., it is used normal approximation. It is convenient to work on a logarithmic scale so that log(theta) is a target for log(Y). Y is plotted versus a different rho depending on the chosen rate.

- If `option = "prop"`, `funnel` plots a proportion y versus its volume. It is used for cross-sectional data. Suppose in each institution that O events are observed out of a sample size of N:

  The indicator is the observed proportion y = O/N

  Assume `N` is the precision parameter (volume). Similarly to when `option = "rate"`, for `N` > 100 the normal and exact curves almost coincide. So, one could perfectly use normal approximation on the parameter `method` if ones data parameter precision is greater than 100, in general.

  phi = (1/total) * sum((y - theta) ^ 2 * N)/g(theta)

  var(y|theta,N) = (phi * g(theta))/N

- If `option = "ratioProp"` or `option = "diffProp"`, `funnel` can be used to compare units at two diferent periods. It plots a ratio (or difference) of proportions y versus a precision parameter rho to assess the change in the underlying proportion from pi1 to pi2. Normal approximations are used throughout, and for low (especially zero) counts, the function adds 0.5 to all arguments r and 1 to all arguments n in order to stabilize the estimates.

  In the case `option = "diffProp"`, the indicator is Y = (O2/N2 - O1/N1) and theta = pi2 - pi1. If `option = "ratioProp"`, the indicator is Y = (O2/N2)/(O1/N1) and theta = pi2/pi1. It is convenient to work on a logarithmic scale, so that log(theta) is a target for log(Y) in this case as well.

  For these two parameter options, the precision parameter (plotted at horizontal axis) can be interpreted as approximately the sample size per period.

Value

A table with unit names, y, observed (Obs), expected (Exp) and admissions (N) for each unit, a binary column showing which units one would like to highlight in the plot (myunits) and final columns show which units are out of control.

References

Spiegelhalter, David J. "Funnel plots for comparing institutional performance." Statistics in medicine 24.8 (2005): 1185-1202.

See Also

`SMR`, `SRU`, `reclass`

Examples

```
# Loading data
data(icu)

# Some edition
icu$Saps3DeathProbabilityStandardEquation <- icu$Saps3DeathProbabilityStandardEquation / 100
icu <- icu[-which(icu$Unit == "F"),]
icu$myunits <- ifelse(icu$Unit == "A",1,0) #my units
icu <- droplevels(icu)

# Getting the cross-sectional arguments to use in funnel
x <- SMR.table(data = icu, group.var = "Unit",
         obs.var = "UnitDischargeName", pred.var = "Saps3DeathProbabilityStandardEquation")
myunit_names <- unique(icu$Unit[which(icu$myunits == 1)])
x$myunits <- ifelse(x$Levels %in% myunit_names, 1,0)

# Analysis of proportions
f1 <- funnel(unit = x$Levels[-1], o = x[-1,]$Observed, theta = x$Observed[1] / x$N[1],
        n = x[-1,]$N, method = "exact", myunits = x$myunits[-1], option = "prop", plot = FALSE)
f1
plot(f1, main = "Cross-sectional proportions")

# To analyze rates (SMR)
f2 <- funnel(unit = x$Levels[-1], y = x[-1,]$SMR, method = "exact", direct = TRUE,
            theta = x$SMR[1], e = x[-1,]$Expected, n = x[-1,]$N, o = x[-1,]$Observed,
            option = "rate", plot = FALSE)
f2
plot(f2, main = "Cross-sectional rate (SMR)")

# Creating a variable containing month information about each admission
icu$month <- as.numeric(format(as.Date(icu$UnitAdmissionDateTime),"%m"))

# First quarter
dt1 <- icu[which(icu$month %in% c(1,2,3)),]

# Second quarter
dt2 <- icu[which(icu$month %in% c(4,5,6)),]

# Getting the two period arguments to use in funnel
z <- SMR.table(data = dt1, group.var = "Unit", obs.var = "UnitDischargeName",
              pred.var = "Saps3DeathProbabilityStandardEquation")
w <- SMR.table(data = dt2, group.var = "Unit", obs.var = "UnitDischargeName",
              pred.var = "Saps3DeathProbabilityStandardEquation")

z$myunits <- ifelse(z$Levels %in% myunit_names, 1,0)
w$myunits <- ifelse(w$Levels %in% myunit_names, 1,0)
# To analyze periods using ratio rates with e1 = e1
f3 <- funnel(unit = z$Levels[-1], n1 = z$N[-1], o1 = z$Observed[-1],
            e1 = z$Expected[-1],
            n2 = w$N[-1], o2 = w$Observed[-1], e2 = z$Expected[-1],
            myunits = z$myunits[-1], option = "ratioRates", plot = FALSE)
f3
```

```
plot(f3, main = "Ratio of SMRs of periods with same expectation of death")


# To analyze periods using ratio rates with e1 =! e1
f4 <- funnel(unit <- z$Levels[-1], n1 = z$N[-1], o1 = z$Observed[-1],
            e1 = z$Expected[-1], n2 = w$N[-1], o2 = w$Observed[-1], e2 = w$Expected[-1],
            option = "ratioRates", plot = FALSE)
f4
plot(f4, main = "Ratio of SMRs of periods with different expectation of death",
    ylim = c(-1.5,1.5), xlim = c(0,200))

# To analyze periods by difference in proportions
f5 <- funnel(unit <- z$Levels[-1], n1 = z$N[-1], o1 = z$Observed[-1],
            n2 = w$N[-1], o2 = w$Observed[-1], option = "diffProp", plot = FALSE)
f5
plot(f5, main = "Difference in proportions of death for two periods")

# To analyze periods by ratio of proportions
f6 <- funnel(unit <- z$Levels[-1], n1 = z$N[-1], o1 = z$Observed[-1],
            n2 = w$N[-1], o2 = w$Observed[-1], option = "ratioProp", plot = FALSE)
f6
plot(f6, main = "Ratio of proportions of death for two periods")

rm(icu, x, z, w, dt1, dt2, unit, f1, f2, f3, f4, f5, f6)
```

`icu` *Data from ICU admissions.*

Description

A dataset containing selected data from some ICU (intensive care units) admissions and its outcomes at the year 2013 used in the ORCHESTRA study.

Usage

```
icu
```

Format

A data frame with 13709 rows and 24 variables

Details

- `Unit` The name of the ICU unit.
- `Age` Patient age.
- `Gender` Male = 1, Female = 0
- `UnitAdmissionDateTime` ICU unit admission date and time.

- `UnitDischargeDateTime` ICU unit discharge date and time.
- `UnitDischargeName` Unit admission outcome. Death = 1, or Discharge = 0.
- `UnitDestinationName` ICU unit destination after discharge.
- `HospitalAdmissionDate` Hospital admission date.
- `HospitalDischargeDate` Hospital discharge date.
- `HospitalDischargeName` Hospital admission outcome. Death = 1, Discharge = 0.
- `LengthHospitalStayPriorUnitAdmission` Hospital length of stay before unit admission.
- `AdmissionSourceName` The origin of the patient before ICU admission.
- `AdmissionTypeName_pri` Admission as Clinical treatment (1) Elective surgery (2), or Urgent surgery (3).
- `AdmissionReasonName_pri` Main diagnosis groups.
- `Vasopressors_D1` Vasopressors use at ICU first day admission? No = No, Yes = 1
- `IsMechanicalVentilation1h` Required mechanichal ventilation at 1st hour of admission. No = 0, Yes = 1.
- `CharlsonComorbidityIndex` Charlson comorbidity index.
- `Saps3Points` SAPS 3 score
- `Saps3DeathProbabilityStandardEquation` SAPS 3 estimated probability
- `SofaScore` SOFA score.

Source

Organizational characteristics, outcomes, and resource use in 78 Brazilian intensive care units: the ORCHESTRA study. Intensive Care Med. 2015 Dec;41(12):2149-60.

`miscellaneous` *Miscellaneous functions for data editing*

Description

Collection of functions for data editing, usually used as lower levels for other functions.

`f.num` is a wrapper to format numeric variables that are stored as character or factor, simultaneously it will try to detect comma spearated and replace it by dots before formating the variable as numeric. Any non-numeric encoding will be coerced to NA.

`f.date` is a wrapper either to `as.Date` or `strptime` to format character or factor variables into dates. In Epimed Solutions database there are a few pre-specified formats that `f.date` will try to detect and return a formated date. `f.date` will try to dected if more than half of the elements in a vector have a pre-specified format. If so, the remaining will be coerced to NA if they have different format from the detected. See example.

`remove.na` identifies all the empty spaces, i.e. the " " cells, of character or factor variables in a data.frame and returns the same data.frame with these empty cells replaced, by default, by NAs. It does not matter the length of the empty spaces. Also, `remove.na` trims the leading and trailing

empty spaces from all character and factor variables. It does not format the numeric variables. It may also return at the console a few information about the " " fields.

tab2tex removes the empty rows, and also tunrs the rownames of a table epiDisplay::tableStack into the first column, to make it easier to paste the table into a rtf or latex document without empty rows or rownames conflicts.

trunc_num truncates a numeric vector by replacing the values below the min value or above the max values by the min and max values respectively or optionall to NA. See example.

dummy.columns takes a data.frame with one column with concatatenated levels of a factor (or character) variable and return a data.frame with additional columns with zeros and ones (dummy values), which names are the factor levels of the original column. See example below. rm.dummy.columns is an internal function of dummy.columns that deletes the new dummy columns which have less then a specified minimum events.

funnelEstimate estimates funnel confidence intervals (CI) for binomial, poisson or normal distribution. Used inside funnel.

winsorising is an internal function that estimates a phi parameter after shirinking extreme z-scores. This parameter is used to inflate funnel CIs due overdispersion presence.

Usage

```
f.num(num.var)

f.date(date)

remove.na(data, replace = NA, console.output = TRUE)

tab2tex(x, nc = ncol(x))

trunc_num(x, min, max, toNA = FALSE)

dummy.columns(data, original.column, factors, scan.oc = FALSE,
  sep = ",", colnames.add = "Dummy.", min.events = NULL,
  rm.oc = FALSE, warn = FALSE, return.factor = TRUE)

rm.dummy.columns(data, colnames, event = "1", min.events = 50,
  warn = FALSE)

funnelEstimate(y, range, u, totalAdmissions, totalObserved, p = 0.95,
  theta = 1, overdispersion = TRUE, dist = c("binomial", "normal",
  "poisson"), rho, gdetheta)

winsorising(z_score, u)
```

Arguments

num.var	A character, or factor variable to be formated as numeric.
date	A character or factor variable to be formated as date.
data	A data.frame.

replace	By default, NA. But could be any vector of length 1.
console.output	Logical. Print at the console a few informations about the " " fields?
x, nc	For tab2tex x is a object from epiDisplay::tableStack. nc is the number of the last column to keep in the table. If the table has 5 columns and nc = 3, then columns 4 and 5 are removed. For trunc_num, x is a numeric vector.
min, max	For trunc_num, min and max are the minimal and maximal numeric values where the numeric vector will be truncated.
toNA	For trunc_num, if FALSE any min and max are the minimal and maximal numeric values where the numeric vector will be truncated.
original.column	A character vector representing the name of the column the be transformed in dummy variables.
factors	A character vector to make new dummy columns and to match values in original.column. This is interesting if the user desires to make dummy only from a few factors in the originlal column. Ignored if scan.oc = TRUE
scan.oc	Default = FALSE, if TRUE, dummy.columns scans the specified original.column and uses all factors to generate dummy variables. It overrides the factor argument.
sep	A character of legth one that systematically split the factors in the original columns. It wil be passed to the sep argument in the scan function.
colnames.add	The default is '= "Dummy_"'. This is a character vector of length one to stick in the colnames of the dummy variables. For example, if the orginal column has A;B;C factor levels, the new dummy variables colnames would be "Dummy_A", "Dummy_B", and "Dummy_C"
min.events	Either NULL (default), or a numeric scalar. If any of the new variables have less events then specified in min.events, they will be deleted before returning the output data.
rm.oc	Default is FALSE. If TRUE, dummy.columns will delete the original column before returnig the final data.frame.
warn	Default is FALSE. If TRUE, dummy.columns will print at the console the deleted columns names.
return.factor	Default is TRUE. If TRUE, dummy.columns return factor columns with "0" and "1" levels, or numeric otherwise.
colnames	For rm.dummy.columns this is the names of the columns to be tested and deleted inside dummy.columns.
event	A character string to be detected as a event. In rm.dummy.columns, if the columns are coded as '0' and '1', the event is '1', if it is coded as logical, the events is 'TRUE'.
y	A numeric vector representing the "Standardized rate" for each unit, usually the SMR or possibly the SRU , accordind to y.type.
range	A numeric range representing for which values the funnel will be estimated. Usually the same variable in x axis (the precision parameter).
u	A number indicating the total amount of ICUs in the data.

totalAdmissions	The quantity of admissions in all units.
totalObserved	The quantity of observed death in all units.
p	A number between 0 and 1 indicating the confidence interval level for the funnel.
theta	Target value which specifies the desired expectation for institutions considered "in control".
overdispersion	Logical (default = FALSE); If TRUE, introduces an multiplicative over-dispersion factor phi that will inflate the CI null variance. See funnel details.
dist	A character specifying the distribution about the funnel control limits will be estimated. It can be "binomial" (default), "normal" or "poisson".
rho	A numeric vector representing the funnel precision parameter. It is calculated inside funnel and used to calculer z_score.
gdetheta	A numeric auxiliary numeric vector used to calculate z_score to be used to calculate estimate funnel control limits.
z_score	A numeric vector indicating the standardized Pearson residual or the "naive" Z-Score for each unit.

Author(s)

Lunna Borges & Pedro Brasil

See Also

dataquality

Examples

```
# Formating character or factor variable that should be numeric variables
f.num(c("2,4000","10,0000","5.0400"))

# Simulating a dataset
y <- data.frame(v1 = sample(c(" F","M  ","   "), 10, replace = TRUE),
                v2 = sample(c(1:3,"     "), 10, replace = TRUE),
                v3 = sample(c("Alive","Dead",""), 10, replace = TRUE))
y

# Replacing the "" cells by NA
y <- remove.na(y)
y

rm(y)

# Formating dates
x <- f.date(c("28/02/2013","16/07/1998","31/03/2010"))
x
class(x)

# The first element (i.e., the different one) is coerced to NA
x <- f.date(c("2013-02-28 12:40","16/07/1998","31/03/2010"))
```

```
x
class(x)

# The last element (i.e. the different one) is coerced to NA
x <- f.date(c("2013-02-28 12:40","1998-07-16 18:50","31/03/2010"))
x
class(x)

# Truncating numeric vectors
trunc_num(1:12, min = 3, max = 10)

# Truncating numeric vectors but returning NAs instead
trunc_num(1:12, min = 3, max = 10, toNA = TRUE)

# Simulating a dataset for dummy.columns example

y <- data.frame(v1 = 1:20,
          v2 = sapply(1:20, function(i) toString(sample(c("Code1","Code2","Code3","Code4"),
                      size = sample(2:4, 1), replace = FALSE))))
y


# For a few of the codes in the original column
y <- dummy.columns(y, original.column = "v2", factor = c("Code2","Code3"))
y

# For all codes in the original column
y <- dummy.columns(y[, 1:2], original.column = "v2", scan.oc = TRUE)
y

# Funnel Estimate
data(icu)
icu

funnelEstimate(y = icu$Saps3DeathProbabilityStandardEquation,
               range = 1, u = length(unique(icu$Unit)),
               totalAdmissions = nrow(icu),
               totalObserved = sum(icu$UnitDischargeName),
               theta = mean(icu$Saps3DeathProbabilityStandardEquation),
               dist = 'normal', rho = 1, gdetheta = 1)

rm(y, icu)
```

`mortality_rate` *Mortality Rate*

Description

`mortality_rate` function returns a list with the mortality rate and the number of patients for each month or quarter of the year.

Usage

```
mortality_rate(deaths, period = NULL, isQuarter = FALSE,
  isYear = FALSE, option = c("both", "monthly", "quarterly", "annual"),
  periodName = NULL, default_tapply = NA)
```

Arguments

deaths	a numerical vector that only contains 0 and 1, indicating whether the patient was alive or dead, respectively.
period	a numerical vector that contains the order of months when the patients were admitted to the hospital unit. If period variable is NULL (the default), the function will return a single mortality rate.
isQuarter	logical indicating whether the period refers to quarter or not. The default is FALSE.
isYear	logical indicating whether the period refers to years or not. The default is FALSE.
option	a character string which determines what the function mortality_rate returns. If the option is chosen to be 'both' (the default), the function will return a list containing monthly mortality rate, quarterly mortality rate, annual mortality rate and the number of patients in each month, quarter and year. If the option is 'monthly', only the monthly mortality rate and the number of patients in each month are returned. If the option is 'quarterly', only the quarterly mortality rate and the number of patients in each quarter are returned. If the option is 'annual', only the annual mortality rate and the number of patients in each year are returned.
periodName	a character vector that contains the name of months when the patients were admitted to de hospital unit. Used only if period is not NULL.
default_tapply	argument to set the default in tapply function when evaluating the mortality rate for each period. Can be equal to 0 or NA (the default).

Author(s)

Camila Cardoso <camila.cardoso@epimedsolutions.com> Lunna Borges <lunna.borges@epimedsolutions.com>

Examples

```
# Loading the dataset
data(icu)

# Creating a vector of months
date <- as.Date(icu$UnitDischargeDateTime, tryFormats = '%d/%m/%Y')
months <- as.numeric(format(date, '%m'))

# Vector of deaths
deaths <- icu$UnitDischargeName

# Calculating monthly and quarterly mortality rate
```

```
mortality_rate(deaths = deaths, period = months, option = 'both')
```

---

reclass *Comparisson of the Standardized Resource Use (SRU)*

Description

Compares ICU's (intensive care units) SRU with diferent severity classes or compares ICU's SRU at two diferents times. This comparison checks if the ICUs remains in the same quadrant after a time period, and highlights their rank changes over time.

`plot.reclass` Plots a SMR vs. SRU scatter plot with the ICUs which had their quadrant/rank classification changed.

`print.reclass` Prints a table with information about which ICUs changed from a classification to another.

Usage

```
reclass(x, y, same = TRUE, plot = FALSE, digits = 2,
  compare = c("SRU", "SMR", "BOTH"), decreasing = FALSE,
  complete.rank = TRUE)

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

## S3 method for class 'reclass'
plot(x, ..., xlim_x = range(x$smr_x),
  ylim_x = range(x$sru_x), xlim_y = range(x$smr_y),
  ylim_y = range(x$sru_y), xlab = "SMR", ylab = "SRU",
  points.arg_x = list(pch = 21, col = "white", bg = "yellow", cex = 2),
  points.arg_y = list(pch = 21, col = "white", bg = "yellow", cex = 2),
  med.arg_x = list(col = "dodgerblue4", lwd = 2, lty = 1),
  med.arg_y = list(col = "dodgerblue4", lwd = 2, lty = 1),
  tert.arg_x = list(col = "darkorange2", lty = 2, lwd = 1),
  tert.arg_y = list(col = "darkorange2", lty = 2, lwd = 1),
  text.arg_x = list(labels = seq(1, nrow(x$tab)), cex = 0.6),
  text.arg_y = list(labels = seq(1, nrow(x$tab)), cex = 0.6),
  worse.arg_x = list(x = x$worse_x, pch = 21, col = "white", bg =
  "tomato", cex = 2), worse.arg_y = list(x = x$worse_y, pch = 21, col =
  "white", bg = "tomato", cex = 2), better.arg_x = list(x = x$better_x,
  pch = 21, col = "white", bg = "mediumseagreen", cex = 2),
  better.arg_y = list(x = x$better_y, pch = 21, col = "white", bg =
  "mediumseagreen", cex = 2), auto.legend = TRUE, leg.arg = list(x =
  "topleft", bty = "n", xpd = NA, inset = c(-1.8, -0.2), ncol = 1, horiz =
  F, pch = 19, cex = 0.8, pt.cex = 1.5), main.arg_x = list(main =
  "1st Stage"), main.arg_y = list(main = "2nd Stage"))
```

Arguments

x, y	Objects of class 'SRU'. x is the SRU analsys from the 1st period (e.g. first trimester) and y from the 2nd period (e.g. second trimester). For print.reclass or plot.reclass, x is an object of class 'reclass'.
same	Logical; If TRUE, compare the same units, with the same severity classes at two consecutive time periods (default). If same = TRUE and the ICUs do not match exactly in 'x' and 'y', there is a warning and non matching units are discarded from the analysis. If FALSE, it compares the same units, with different severity classes within the same period. In this case, if the ICUs do not match exactly in 'x' and'y', the function will return an error.
plot	Logical. If TRUE (default), plots a SMR vs. SRU scatter plot highlighting the ICUs which had their classification changed.
digits	Integer indicating the number of decimal places to be used in the output.
compare	The way one prefers to benchmark the ICUs: by "SRU" (default), "SMR" or "BOTH". If "BOTH", the ICUs will be ranked by their SRU.
decreasing	Logical. Should the sort order of ICU's rank be increasing or decreasing?
complete.rank	Logical. If TRUE (default), returns all ICUs ranked. If FALSE, returns only ICUs whose efficiency classification ranked changed.
...	Arguments to be passed to methods (see par).
xlim_x, ylim_x	Limits for x and y axis for 1st stage plot for plot.reclass.
xlim_y, ylim_y	Limits for x and y axis for 2nd stage plot for plot.reclass.
xlab, ylab	Labels of x and y axis for plot.reclass.
points.arg_x, points.arg_y	List of arguments passed to points for plotting points correponding to units' SMR and SRU in 1st and 2nd stage plots for plot.reclass.
med.arg_x, med.arg_y	List of arguments passed to abline for plotting lines corresponding to SRU and SMR medians in 1st and 2nd stage plots for plot.reclass.
tert.arg_x, tert.arg_y	List of arguments passed to abline for plotting lines corresponding to SRU and SMR tertiles in 1st and 2nd stage plots for plot.reclass.
text.arg_x, text.arg_y	List of arguments passed to text for plotting units labels in 1st and 2nd stage plots for plot.reclass.
worse.arg_x, worse.arg_y	List of arguments passed to points for plotting points correponding to units which got their rank worse in 1st and 2nd stage plots for plot.reclass.
better.arg_x, better.arg_y	List of arguments passed to points for plotting points correponding to units which got their rank better in 1st and 2nd stage plots for plot.reclass.
auto.legend	Logical. If TRUE, it prints a legend with leg.arg arguments for plot.reclass.
leg.arg	List of arguments passed to legend for plotting legends corresponding to SRU and SMR medians and tertiles in 1st and 2nd stage plots for plot.reclass.
main.arg_x, main.arg_y	List of arguments passed to plot for the titles for the 1st and 2nd stage plots for plot.reclass.

Value

`reclass` retunrs a data.frame with the following columns:

- `Unit` Names of the ICU.
- `Admission` Number of admissions in each ICU.
- `From` ICU's initial efficiency quadrant.
- `To` ICU's final efficiency quadrant.
- `SRU.1st` ICU's initial SRU estimate.
- `SRU.2nd` ICU's final SRU estimate.
- `SMR.1st` ICU's initial SMR estimate.
- `SMR.2nd` ICU's final SMR estimate.
- `Rank1` ICU's initial SRU (or SMR) rank.
- `Rank2` ICU's final SRU (or SMR) rank.

`plot.reclass` returns a scatter plot with grpahical comparison of the two periods/stages with their respective medians and tertiles.

Author(s)

Lunna Borges and Pedro Brasil

See Also

SRU, SMR, funnel

Examples

```
data(icu)
# A little editing
icu$Saps3DeathProbabilityStandardEquation <- icu$Saps3DeathProbabilityStandardEquation / 100
icu <- icu[-which(icu$los < 0 ),]

# Subseting the data for the 1st quarter
x <- droplevels(icu[which(format(as.Date(icu$UnitAdmissionDate),"%m") %in% c("01","02","03")),])

# Subseting the data for the 2nd quarter
y <- droplevels(icu[which(format(as.Date(icu$UnitAdmissionDate),"%m") %in% c("04","05","06")),])

# Running the SRU analysis for both quarters
FirstQ <- SRU(prob = x$Saps3DeathProbabilityStandardEquation, death = x$UnitDischargeName,
unit = x$Unit, los = x$los, score = x$Saps3Points, originals = TRUE, type = 1, plot = FALSE)
FirstQ

SecondQ <- SRU(prob = y$Saps3DeathProbabilityStandardEquation, death = y$UnitDischargeName,
unit = y$Unit, los = y$los, score = y$Saps3Points, originals = TRUE, type = 1, plot = FALSE)
SecondQ
```

```
z <- reclass(x = FirstQ, y = SecondQ, same = TRUE)
z
plot(z)

rm(icu, x, y, FirstQ, SecondQ, z)
```

SMR *Standardized Mortality Ratio (SMR)*

Description

Calculates the standardized mortality ratio and its confidence interval. SMR, for a group, is defined as the ratio of the observed deaths in this group and the sum of the predicted individual probabilities of death by any model (expected deaths).

`SMR.table` estimates at once the overall SMR and the SMR across several groups, e.g. ICU units or clinical characteristics. The `SMR.table` can be ordered by the SMR estimate or its confidence intervals, facilitating the comparison of the units ranks.

`forest.SMR` shows the `SMR.table` output as a forest plot. The plot opens two windows and plot at the left side the values from the `SMR.table` and at the right side the points and lines graphically representing each SMR and its confidence interval.

Usage

```
SMR(obs.var, pred.var, digits = 5, ci.method = c("Hosmer", "Byar"),
  ci.level = 0.95)

SMR.table(data, group.var, obs.var, pred.var, digits = 5,
  use.label = FALSE, var.labels = attr(data,
  "var.labels")[match(group.var, names(data))], ci.method = c("Hosmer",
  "Byar"), ci.level = 0.95, reorder = c("no", "SMR", "lower.CI",
  "upper.CI"), decreasing = FALSE)

forest.SMR(x, mar1 = c(5.1, 1, 4.1, 1), mar.SMR = c(5.1, 7, 4.1, 1),
  overall.arg = list(x = 0.01, font = 2, las = 1, labels = var.labels[1],
  xpd = NA, adj = 0), NOE.overall.args = list(x = c(N.values.arg$x,
  O.values.arg$x, E.values.arg$x), font = 2, las = 1, xpd = NA),
  var.labels.arg = list(x = 0.01, font = 2, las = 1, cex = 1, xpd = NA,
  adj = 0), cat.labels.arg = list(x = 0.1, font = 3, las = 1, cex = 0.95,
  col = gray(0.4), xpd = NA, adj = 0), N.values.arg = list(x = 0.5, col =
  gray(0.4), xpd = NA), O.values.arg = list(x = 0.675, col = gray(0.4),
  xpd = NA), E.values.arg = list(x = 0.85, col = gray(0.4), xpd = NA),
  NOE.head.arg = list(font = 2, labels = c("N", "O", "E"), xpd = NA),
  Overall.seg.arg = list(col = "navyblue", xpd = NA, lwd = 2),
  Overall.p.arg = list(pch = 23, cex = 2, col = "black", bg = gray(0.4),
  xpd = NA), Overall.est.arg = list(x = smr.xlim[1] - 0.06, las = 1, font
  = 2, xpd = NA, adj = 1), cat.seg.arg = list(col = "navyblue", xpd = NA,
```

```
lwd = 2), cat.p.arg = list(pch = 22, cex = 1, col = "black", bg =
gray(0.4), xpd = NA), cat.est.arg = list(x = smr.xlim[1] - 0.06, las =
1, col = gray(0.4), xpd = NA, adj = 1), SMR.head.arg = list(smr.xlim[1]
- 0.06, font = 2, labels = "SMR [95% CIs]", xpd = NA, adj = 1),
smr.xlab = "Standardized Mortality Ratio", smr.xlim = "auto",
grid = TRUE, digits = 3)
```

Arguments

obs.var	Observed death. Accepted values are 0 (absence) or 1 (presence) in a vector. For SMR.table it must be a character indicating the name of the variable in the data.
pred.var	Death individual predictions (ranging from 0 to 1) in a vector. For SMR.table it must be a character indicating the name of the variable in the data.
digits	Number of digits for rounding the output.
ci.method	Method to estimate the confidence interval. "Hosmer" (default) or "Byar" are acceptable values.
ci.level	Level of the confidence interval. Default is 0.95.
data	For SMR.table, a dataset where pred.var, obs.var and group.var are in.
group.var	For SMR.table, this is a character vector indicating the name(s) of the variable(s) in the data that will form the groups where SMR will be calculated. The variables must be factors.
use.label	Logical. Default is FALSE. For SMR.table this option will replace the variables names by its labels in var.labels argument.
var.labels	A character vector with variables labels. The default is to replace the variable name by the label stored at attr(data, "var.labels"). But one may specify labels directly.
reorder	Default is "no". Possible values are: "no", "SMR","lower.Cl", and "upper.Cl". It will make the SMR.table to be ordered within each varibale by its original order, or by SMR order, or by lower.Cl order, or by upper.Cl.
decreasing	Logical. When 'reorderd' is TRUE, should the order be decreasing or incresing? See order
x	For the forest.SMR this is the output of SMR.table.
mar1, mar.SMR	Values to set the margins (mar parameter) of left and right windows. See par
overall.arg	A list of arguments passed to text for plotting the overall label. Internally, 'y' coordinate is replaced.
NOE.overall.args	A list of arguments passed to text for plotting the overall N (number of observations), O (observed deaths) and E (expected deaths). Internally, 'labels' and 'y' arguments are replaced.
var.labels.arg	A list of arguments passed to text for plotting the variables labels. Internally, 'y' coordinate is replaced.
cat.labels.arg	A list of arguments passed to text for plotting the categories labels. Internally, 'y' coordinate is replaced.

N.values.arg	A list of arguments passed to text for plotting the values of N (number of observations) of each subgroup. Internally, the arguments 'label' and 'y' coordinate are replaced.
O.values.arg	A list of arguments passed to text for plotting the values of Observed deaths of each subgroup. Internally, the arguments 'label' and 'y' coordinate are replaced.
E.values.arg	A list of arguments passed to text for plotting the values of Expected deaths of each subgroup. Internally, the arguments 'label' and 'y' coordinate are replaced.
NOE.head.arg	A list of arguments passed to text for plotting the labels of the columns N, E and O on the top of the graph. Internally, the 'x' and 'y' coordinates are replaced. The x coordinates are taken from the x in N.values.arg, O.values.arg and E.values.arg.
Overall.seg.arg	A list of arguments passed to segments for plotting the lines corresponding to overall SMR confidence intervals. Internally, 'x' and 'y' coordinates are replaced.
Overall.p.arg	A list of arguments passed to points for plotting the points corresponding to overall SMR. Internally, 'x' and 'y' coordinates are replaced.
Overall.est.arg	A list of arguments passed to text for plotting the overall SMR beside the graph. Internally, 'y' coordinate and 'label' argument are replaced.
cat.seg.arg	A list of arguments passed to segments for plotting the lines corresponding to SMR confidence intervals for all groups. Internally, 'x' and 'y' coordinates are replaced.
cat.p.arg	A list of arguments passed to points for plotting the points corresponding to all categoreis SMR. Internally, 'x' and 'y' coordinates are replaced.
cat.est.arg	A list of arguments passed to text for plotting the categories SMR beside the graph. Internally, 'y' coordinate and 'label' arguments are replaced.
SMR.head.arg	A list of arguments passed to text for plotting the label of the SMR column on the top of the graph. Internally, the 'y' coordinate is replaced.
smr.xlab	Label of the x axis. Default is "Standardized Mortality Ratio".
smr.xlim	Limits of x axis of the forest.SMR plot. Default is "auto", which internally will pick the highest values of all upper.Cl and the lowest lower.Cl. Besides "auto", only a vector of 2 numbers is valid, and will be passed to plot.default.
grid	Logical. If TRUE (default), it will draw a grid with the grid default arguments.

Value

If SMR, then:

- N Number of subjects analyzed.
- O Observed number of deaths.
- E Expected number of deaths.
- SMR Standardized mortality ratio.
- lower.Cl lower confidence limit.

- upper.Cl upper confidence limit.

If SMR.table, then a data.frame with the same information as above, and the additional information is returned: "Variables" (variables names), "Levels" (variables levels).

If forest.SMR, then a plot is returned.

Author(s)

Lunna Borges and Pedro Brasil

References

David W. Hosmer and Stanley Lemeshow. Confidence intervals estimates of an index of quality performance basend on logistic regression models. Statistics in Medicine , vol. 14, 2161-2172 (1995)

See Also

SRU, reclass, funnel

Examples

```
# Loading a example data
data(icu)

# Setting variable labels to data
attr(icu, "var.labels")[match(c("Unit", "IsMechanicalVentilation1h",
          "AdmissionTypeName_pri","Vasopressors_D1"), names(icu))] <-
  c("ICU unit","Mechanichal ventilation","Admission type","Vasopressors at admission")

# Some editing
icu$Saps3DeathProbabilityStandardEquation <- icu$Saps3DeathProbabilityStandardEquation /100
icu$IsMechanicalVentilation1h <- as.factor(ifelse(icu$IsMechanicalVentilation1h == 1, "Yes", "No"))
icu$AdmissionTypeName_pri <- as.factor(icu$AdmissionTypeName_pri)
levels(icu$AdmissionTypeName_pri) <- c("Clinical","Elective surgery", "Urgent surgery")
icu$Vasopressors_D1 <- as.factor(ifelse(icu$Vasopressors_D1 == 1, "Yes", "No"))

# The overall SMR for the whole sample
SMR(icu$UnitDischargeName, icu$Saps3DeathProbabilityStandardEquation)

# The overall SMR and for some subgroups
x <- SMR.table(data = icu, obs.var = "UnitDischargeName",
               pred.var = "Saps3DeathProbabilityStandardEquation",
               group.var = c( "IsMechanicalVentilation1h",
               "AdmissionTypeName_pri","Vasopressors_D1"),
               reorder = "no",
               decreasing = TRUE,
               use.label = TRUE)
x

# A forest plot for all groups SMR (resize the window may be required)
forest.SMR(x, digits = 2)
```

```
# The same thing but reordering the categories
x <- SMR.table(data = icu, obs.var = "UnitDischargeName",
               pred.var = "Saps3DeathProbabilityStandardEquation",
               group.var = c( "IsMechanicalVentilation1h",
               "AdmissionTypeName_pri", "Vasopressors_D1"),
               reorder = "SMR",
               decreasing = TRUE,
               use.label = TRUE)
forest.SMR(x, digits = 2)

# The overall SMR and for all Units
x <- SMR.table(data = icu, obs.var = "UnitDischargeName",
               pred.var = "Saps3DeathProbabilityStandardEquation",
               group.var = "Unit",
               reorder = "no",
               decreasing = TRUE,
               use.label = TRUE)
x

# A forest plot for all Units
forest.SMR(x, digits = 2)

# The same thing but reordering the categories
x <- SMR.table(data = icu, obs.var = "UnitDischargeName",
               pred.var = "Saps3DeathProbabilityStandardEquation",
               group.var = "Unit",
               reorder = "SMR",
               decreasing = TRUE,
               use.label = TRUE)
forest.SMR(x, digits = 2)

rm(x, icu)
```

SRU *Standardized Resource Use (SRU)*

Description

SRU calculates the standardized resource use for ICUs (Intensive Care Units) from information regarding admissions of individual patients. Resource use is represented by the patient's length of stay (LOS). Therefore the SRU for each unit is defined as the observed LOS divided by it's expected LOS. To estimate the expected LOS for each ICU one must define a severity score, here defined by the SAPS 3 score. In theory, the 'score' could be any score/probability that estimates death for each ICU admission.

The `plot.SRU` function will return a SMR versus SRU scatter plot with its medians and tertiles. Thus, it classifies each unit in the quadrants formed by these two medians as: most efficient (ME) which is the lower left quadrant (both SRU and SMR below their medians); least efficient (LE) is the upper right quadrant (both SRU and SMR above their medians); and least achieving (LA) - the lower right

quadrant (SRU below and SMR above their medians); and over achieving (OA) - the upper left quadrant (SRU above and SMR below their medians).

`print.SRU` Prints a object of class 'SRU'.

`cut_in` is used to find limits to define severity classes which are used in `SRU` function. The severity classes are necessary to calculate the average of days to produce one survivor and consequently to estimate the expected LOS in each ICU. Its rationale is to find the limits for the severity classes that yeld a desired average of days to produce one survivor. At some point in time, we made a study to test if different arrangements of the severity classes would yeld different classifications in the efficiency quadrants. Despite the fact that this study did not show any difference from each approach, we left the function in the package. Therefore, any arbitrary severity classes should yeild the same results.

`SRUcalc` is a simpler function to estimte SRU and returns, for each unit, the SRU value, the observed and expected number of deaths, and the observed and expected LOS.

Usage

```
SRU(prob, death, unit, los, los.exp, class, score, plot = FALSE,
  type = 1, digits = 2, digits2 = 5, originals = FALSE,
  myunits = NULL)

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

## S3 method for class 'SRU'
plot(x, ..., xlim = range(x$rates[, 2]),
  ylim = range(x$rates[, 1]), xlab = "SMR", ylab = "SRU",
  points.arg = list(pch = 21, col = "white", bg = "cadetblue3", cex =
  1.5), med.arg = list(col = "dodgerblue4", lwd = 2, lty = 1),
  tert.arg = list(col = "darkorange2", lty = 2, lwd = 1),
  auto.legend = TRUE, leg.arg = list(x = "top", bty = "n", xpd = NA,
  inset = -0.2, ncol = 2), bty = "n", myunits = x$myunits,
  myunitspts.arg = list(pch = 21, col = "white", bg = "red", cex = 1.5),
  myunitstext.arg = list(pos = 1, font = 2, cex = 0.8))

cut_in(score, los, death, unit, days, min = 200, exc.ICU = TRUE,
  complete = FALSE, digits = 5)

SRUcalc(prob, death, unit, los, score, digits = 2)
```

Arguments

`prob`	Individual predicted probability of death (ranging from 0 to 1) in a vector.
`death`	Observed death. Accepted values are 0 (absence) or 1 (presence) in a vector.
`unit`	A character or factor variable indicating the ICU where the patient is admitted.
`los`	A numeric variable indicating the observed length of stay for each patient.
`los.exp`	Estimated length of stay (LOS). This argument is optional and will be required only if `type = 2`. If the user has an alternative model to estimate the individual

LOS, the predicted individual LOS should be passed to this argument. If this is the case, the predicted ICU LOS is estimated as the mean of the individual predictions of the LOS of these groups.

`class` A factor variable indicating the class of severity score (e.g. SAPS 3). In the case of SAPS 3, this is a cut in the SAPS 3 score, grouping patients into severity classes. This will be required if the argument `original = FALSE` and NAs are not allowed; if `original = TRUE`, class is ignored.

`score` A numeric vector with the Acute Physiology Score (SAPS) 3 score for each admission. The function will use this argument to know to which severity class each patient will be assigned to. It is used only when `originals = TRUE` and ignored otherwise. NAs are not allowed.

`plot` Logical; If `TRUE`, plots a SMR versus SRU scatter plot.

`type` A Way to calculate SRU. If `type = 1`, it does as the original article to estimate the ICU's expected LOS (default). First, it multiplies the overall average of days of each severity class by the number of survivors in the same severity class in that ICU. Than, it sums the expected LOS for each severity class in that ICU. If `type = 2`, the user must provide the `los.exp` (expected LOS) for each subject (i.e. from a prediction model), and the function will estimate the ICU's expected LOS as the mean of all individual LOS for patients in that ICU.

`digits, digits2`
Integer indicating the number of decimals to be used in the output.

`originals` Logical; If `TRUE`, it uses the severity classes and average days as the original article and will override the `class` argument, if any. It requires the `score` argument and it must be the SAPS 3 score. We recommend not to set it `TRUE` unless you really know what you are doing. Even if one wishes to have severity classes identical to the original paper, it is better to set the severity classes before running the analysis. This way, the function will estimate the average days from the data instead of using the fixed average days from the original paper.

`myunits` A character vector with the unit names which one would like to benchmark among all units. These units will be highlighted with dots of different collors in the plot. Default is `NULL`.

`x` For `print.SRU` or `plot.SRU`, an object of class 'SRU'.

`...` Arguments to be passed to `plot.default` or to `print`.

`xlim, ylim` Limits of x and y axis for `plot.SRU`.

`xlab, ylab` Labels of x and y axis for `plot.SRU`.

`points.arg` List of arguments passed to `points` for plotting points correponding to ICU's SMR and SRU.

`med.arg` List of arguments passed to `abline` for plotting lines corresponding to SRU's and SMR's medians.

`tert.arg` List of arguments passed to `abline` for plotting lines corresponding to SRU's and SMR's tertiles.

`auto.legend` Logical; If `TRUE`, prints a legend with parameters in `leg.arg` arguments.

`leg.arg` List of arguments passed to `legend` for plotting legends in `plot.SRU`.

bty	A character string which determines the type of box that is drawn about plots. See par
myunitspts.arg	List of arguments passed to points for plotting points correponding to myunits's SMR and SRU.
myunitstext.arg	List of arguments passed to text for labelling points correponding to myunits's position.
days	For cut_in, this is a vector of days to get an avarage. See example.
min	For cut_in, this is the minimum desired quantity of patients in each severity class (default = 200) to estimate the average days.
exc.ICU	Logical; For cut_in, if TRUE, ICUs without surviving patients are ignored.
complete	Logical; For cut_in, if TRUE, shows additional information about severity classes.

Value

Two tables: one with information about severity classes and the respective quantities required to estimate the expected LOS, and another with information about ICUs classified as Most Efficient (ME) or Least Efficient (LE).

- Sev Severity class.
- Total Total of patients.
- Surv Total of survivors.
- Total.LOS Total length of stay (days).
- AvDays Average days to produce a survivor.
- N.Unit Quantity of ICUs.
- N.Pat Quantity of patients.
- SMR Standardized Mortality Ratio Mean (standard deviation).
- SRU Standardized Resource Use Mean (standard deviation).

Most Efficient ICUs have SRU, SMR < median. Least Efficient ICUs have SRU, SMR > median.

cut_in returns a vector with the limits to cut the severity score.

SRUcalc returns a table with:

- Unit ICUs names.
- SMR or SRU Standardized Rate.
- N Number of subjects analyzed.
- Observed Observed number of deaths.
- Expected Expected number of deaths.
- LOS_esp Expected length of stay.

Author(s)

Lunna Borges and Pedro Brasil

References

Rothen HU, Stricker K, Einfalt J, Bauer P, Metnitz PGH, Moreno RP, Takala J (2007) Variability in outcome and resource use in intensive care units. Intensive Care Med 33:1329-1336

See Also

SMR, reclass, funnel

Examples

```
# Loading the dataset
data(icu)

# Removing data with inapropriate values and some editing
icu <- icu[-which(icu$los < 0 ),]
icu$Saps3DeathProbabilityStandardEquation <- icu$Saps3DeathProbabilityStandardEquation / 100

# Setting classes acording to limits of SAPS 3 score
days <- seq(1,100)
cut_lims <- cut_in(icu$Saps3Points, icu$los, icu$UnitDischargeName,
                   icu$Unit, days, exc.ICU = TRUE)
icu$class <- cut(icu$Saps3Points, breaks = cut_lims, include.lowest = TRUE)

# Estimating the SRU benchmarking myunit A and B
x <- SRU(prob = icu$Saps3DeathProbabilityStandardEquation,
death = icu$UnitDischargeName, unit = icu$Unit,
los = icu$los, score = icu$Saps3Points,
originals = TRUE, type = 1, plot = FALSE, myunits = c("A","B"))
x
plot(x)

# To see the units rankings and individual SMR and SRU, ordering by its SRU
x$rates[order(x$rates$sru),]

# SRU with diferent severity classes created by cut_in function
y <- SRU(prob = icu$Saps3DeathProbabilityStandardEquation,
death = icu$UnitDischargeName, unit = icu$Unit,
los = icu$los, score = icu$Saps3Points,
originals = FALSE, type = 1, plot = FALSE, class = icu$class)
y

# Using SRUcalc
SRUcalc(prob = icu$Saps3DeathProbabilityStandardEquation,
        death = icu$UnitDischargeName, unit = icu$Unit, los = icu$los,
        score = icu$Saps3Points)

rm(x, y, days, icu, cut_lims)
```

survPerformance *Survival models performance analysis*

Description

Colection of functions for survival models performance analysis.

R2sh estimates a distance-based estimator of survival predictive accuracy proposed by Schemper and Henderson. It was inspirated in survAUC::schemper function, but receives the predicted values directly. Besides that, R2sh does bootstrap resampling and returns its confidence interval estimate.

R2pm calculates a estimator of survival predictive accuracy proposed by Kent & O'Quigley and its bootstrap confidence interval.

cal.Slope returns the calibration slope of a survival model and its bootstrap confidence interval.

Usage

```
R2sh(time, status, lin.pred, data, R)

R2pm(lin.pred, R)

cal.Slope(time, status, lin.pred, R)
```

Arguments

time	A vector of event times.
status	A indicator vector of event occurrence.
lin.pred	A vector of linear predictors of a survival model for each observation. (prognostic index)
data	A data.frame where to find column vectors.
R	The number of bootstrap replicates. Usually this will be a single positive integer. For importance resampling, some resamples may use one set of weights and others use a different set of weights. In this case R would be a vector of integers where each component gives the number of resamples from each of the rows of weights. To be passed to boot.

Value

R2sh returns a list with the following components:

- D: The estimator of predictive accuracy obtained from the covariate-free null model.
- Dx: The estimator of predictive accuracy obtained from the Cox model.
- V: The estimator of relative gains in predictive accuracy.
- Mhat: The absolute distance estimator obtained from the Cox model (evaluated at the event times of the test data).

- `Mhat.0`: The absolute distance estimator obtained from the covariate-free null model (evaluated at the event times of the test data).
- `timep`: The event times of the test data.
- `lower`: V lower confidence limit.
- `upper`: V upper confidence limit.
- `boot`: An object of class "`boot`".
- `bootCI`: Boot confidence intervals resampling.

`R2pm` returns a list with the following components:

- `r2`: The estimator of predictive accuracy obtained from the Cox model.
- `lower`: r2 lower confidence limit.
- `upper`: r2 upper confidence limit.
- `boot`: An object of class "`boot`".
- `bootCI`: Boot confidence intervals resampling.

`cal.Slope` returns a list with the following components:

- `slope`: The calibration slope measure of a survival model.
- `lower`: slope lower confidence limit.
- `upper`: slope upper confidence limit.
- `boot`: An object of class "`boot`".
- `bootCI`: Boot confidence intervals resampling.

Author(s)

Lunna Borges <lunna.borges@epimedsolutions.com>

References

Schemper, M. and R. Henderson (2000). Predictive accuracy and explained variation in Cox regression. Biometrics 56, 249-255.

Davison, A.C. and Hinkley, D.V. (1997) Bootstrap Methods and Their Application, Chapter 5. Cambridge University Press.

DiCiccio, T.J. and Efron B. (1996) Bootstrap confidence intervals (with Discussion). Statistical Science, 11, 189-228.

Efron, B. (1987) Better bootstrap confidence intervals (with Discussion). Journal of the American Statistical Association, 82, 171-200.

Kent, John T., and J. O. H. N. O'QUIGLEY. "Measures of dependence for censored survival data." Biometrika 75.3 (1988): 525-534.

van Houwelingen, Hans C. "Validation, calibration, revision and combination of prognostic survival models." Statistics in medicine 19.24 (2000): 3401-3415.

Rahman, M. Shafiqur, et al. "Review and evaluation of performance measures for survival prediction models in external validation settings." BMC medical research methodology 17.1 (2017): 60.

Examples

```
#### Survival model ####

data(breastCancer)
class(breastCancer$gradd1) <- "character"
class(breastCancer$gradd2) <- "character"

traindata <- breastCancer[sample(nrow(breastCancer), nrow(breastCancer)*2/3),]
newdata <- breastCancer[-sample(nrow(breastCancer), nrow(breastCancer)*2/3),]
model <- rms::cph(survival::Surv(rectime, censrec) ~ rms::rcs(age,6) +
 rms::rcs(nodes,3) + rms::rcs(pgr,3) + gradd1 + gradd2 +
 hormon, data = traindata)

lp <- predict(model, newdata = newdata)

#### R2sh  example ####

R2sh(newdata$rectime, newdata$censrec, lp, data = newdata, R = 50)

#### R2pm example ####

R2pm(lp, R = 50)

#### cal.slope example ####

cal.Slope(newdata$rectime, newdata$censrec, lp, R = 50)
```

---

tableStack *Tabulation of variables in a stack form*

Description

There are two functionalities: Tabulation of variables with the same possible range of distribution and stack into a new table with or without other descriptive statistics or to breakdown distribution of more than one row variables against a column variable

Usage

```
tableStack(vars, dataFrame, minlevel = "auto", maxlevel = "auto",
  count = TRUE, na.rm = FALSE, means = TRUE, medians = FALSE,
  sds = TRUE, decimal = 2, total = TRUE, var.labels = TRUE,
  var.labels.trunc = 150, reverse = FALSE, vars.to.reverse = NULL,
  by = NULL, vars.to.factor = NULL, iqr = "auto",
  prevalence = FALSE, percent = c("column", "row", "none"),
  frequency = TRUE, test = TRUE, name.test = TRUE,
  total.column = FALSE, simulate.p.value = FALSE, sample.size = TRUE,
  assumption.p.value = 0.01, NAcol = FALSE, NArow = FALSE,
  drplvls = FALSE)
```

Arguments

vars	a vector of variables in the data frame. The imput may be given with or without quotes.
dataFrame	source data frame of the variables
minlevel	possible minimum value of items specified by user
maxlevel	possible maximum value of items specified by user
count	whether number of valid records for each item will be displayed
na.rm	whether missing value would be removed during calculation mean score of each person
means	whether means of all selected items will be displayed
medians	whether medians of all selected items will be displayed
sds	whether standard deviations of all selected items will be displayed
decimal	number of decimals displayed
total	display of means and standard deviations of total and average scores
var.labels	presence of descriptions of variables on the last column of output
var.labels.trunc	number of characters used for variable description
reverse	whether item(s) negatively correlated with other majority will be reversed
vars.to.reverse	variable(s) to reverse
by	a variable for column breakdown. If NONE is given, only the 'total column' will be displayed. More on Details.
vars.to.factor	variable(s) to be converted to factor for tabulaton
iqr	variable(s) to display median and inter-quartile range
prevalence	for logical or dichotomous variables, whether prevalence of the dichotomous row variable in each column subgroup will be displayed
percent	type of percentage displayed when the variable is categorical and for NArow when activated. Default is column
frequency	whether to display frequency in the cells when the variable is categorical and for NArow when activated
test	whether statistical test(s) will be computed
name.test	display name of the test and relevant degrees of freedom
total.column	whether to add 'total column' to the output or not
simulate.p.value	simulate P value for Fisher's exact test
sample.size	whether to display non-missing sample size of each column
assumption.p.value	level of Bartlett's test P value to judge whether the comparison and the test should be parametric
NAcol	whether to add 'NA column' to the output or not
NArow	whether to add 'NA rows' for each variable to the output or not
drplvls	whether to hide non used levels on factor and character variables or not

Details

This function is a clone of `tableStack` from the `epiDisplay` package. Comparing to the original, tt adds options to show the NA in the variables as categories, similar to the option `useNA` in the `table` function, and it also fix few bugs, such as showing the `total.column` without the need to `test = TRUE`, and to show or hide levels with zero counts without returning error.

This function simultaneously explores several variables with a fixed integer rating scale. For non-factor variables, the default values for tabulation are the minimum and the maximum of all variables but can be specified by the user.

When 'by' is omitted, all variables must be of the same class, and must be 'integer', 'factor' or 'logical. Some parameters are only used if by is omitted, others are only used if by is available. The by-omitted dependent variable are minlevel, maxlevel, count, na.rm, means, medians, sds, total, reverse, vars.to.reverse. The by-available dependent variables are iqr, prevalence, percent, frequency, test, name.test, total.column, simulate.p.value, sample.size, assumption.p.value, NArow, NAcol, drplvls. Unlike function 'alpha', the argument 'reverse' has a default value of FALSE. This argument is ignored if 'vars.to.reverse' is specified.

Options for 'reverse', 'vars.to.reverse' and statistics of 'means', 'medians', 'sds' and 'total' are available only if the items are not factor. To obtain statistics of factor items, users need to use 'unclassDataframe' to convert them into integer.

When the 'by' argument is given, 'reverse' and 'vars.to.reverse' do not apply, as mentioned before. Instead, columns of the 'by' variable will be formed. A table will be created against each selected variable. If the variable is a factor or coerced to factor with 'vars.to.factor', cross-tabulation will result with percents as specified, ie. "column", "row", or "none" (FALSE). For a dichotomous row variable, if set to 'TRUE', the prevalence of row variable in the form of a fraction is displayed in each subgroup column. For objects of class 'numeric' or 'integer', means with standard deviations will be displayed. For variables with residuals that are not normally distributed or where the variance of subgroups are significantly not normally distributed (using a significance level of 0.01), medians and inter-quartile ranges will be presented if the argument 'iqr' is set to "auto" (by default). Users may specify a subset of the selected variables (from the 'vars' argument) to be presented in such a form. Otherwise, the argument could be set as any other character string, except the variables names, to insist to present means and standard deviations.

When 'test = TRUE' (default), Pearson's chi-squared test (or a two-sided Fisher's exact test, if the sample size is small) will be carried out for a categorical variable or a factor. Parametric or non-parametric comparison and test will be carried out for a object of class 'numeric' or 'integer' (See 'iqr' and 'assumption.p.value' below). If the sample size of the numeric variable is too small in any group, the test is omitted and the problem reported.

For Fisher's exact test, the default method employs 'simulate.p.value = FALSE'. See further explanation in 'fisher.test' procedure. If the dataset is extraordinarily large, the option may be manually set to TRUE.

When 'by' is specified as a single character object (such as 'by="none"') or when 'by = NONE' there will be no column breakdown and all tests will be omitted. Only the total column is displayed. Only the 'total' column is shown.

If this 'total column' is to accompany the 'by' breakdown, the argument 'total.column=TRUE' should be specified. The 'sample.size' is TRUE by default. The total number of records for each group is displayed in the first row of the output. However, the variable in each row may have some missing records, the information on which is reported by NArow for each variable on 'vars' and by NAcol for the variable on 'by'.

By default, Epicalc sets 'var.labels=TRUE' in order to give nice output. However, 'var.labels=FALSE' can sometimes be more useful during data exploration. Variable numbers as well as variable names are displayed instead of variable labels. Names and numbers of abnormally distributed variables, especially factors with too many levels, can be easily identified for further relevelling or recoding.

The argument 'iqr' has a default value being "auto". Non-parametric comparison and test will be automatically chosen if Bartlett's test P value is below the 'assumption.p.value'.

The test can be forced to parametric by setting 'iqr=NULL' and to non-parametric by if iqr is set to the variable number of cont.var (See examples.).

Value

an object of class 'tableStack' and 'list' when by=NULL

`results`	an object of class 'noquote' which is used for print out
`items.reversed`	name(s) of variable(s) reversed
`total.score`	a vector from 'rowSums' of the columns of variables specified in 'vars'
`mean.score`	a vector from 'rowMeans' of the columns of variables specified in 'vars'
`mean.of.total.scores`	mean of total scores
`sd.of.total.scores`	standard deviation of total scores
`mean.of.average.scores`	mean of mean scores
`sd.of.average.scores`	standard deviation of mean scores

When 'by' is specified, an object of class 'tableStack' and 'table is returned.

Author(s)

Virasakdi Chongsuvivatwong <cvirasak@medicine.psu.ac.th>

Caio Ferreira <caio.ferreira@epimedsolutions.com>

Lunna Borges <caio.ferreira@epimedsolutions.com>

Pedro Brasil <pedro.brasil@epimedsolutions.com>

References

'table', 'tab1', 'summ', 'alpha', 'unclassDataframe'

Examples

```
set.seed(1)
data <- data.frame(sex = sample(c("M","F"), 50, rep = TRUE),
age = sample(c(NA,20:70), 50, rep = TRUE),
admissionType = sample(c(NA,"urgency", "clinical", "scheduled"), 50, rep = TRUE),
hospitalizationTime = sample(c(0:10), 50, rep = TRUE),
numberOfChildren = sample(c(NA,0:3), 50, rep = TRUE),
cancerInFamily = sample(c(NA,TRUE,FALSE), 50, rep = TRUE),
diabetesInFamily = sample(c(TRUE,FALSE), 50, rep = TRUE),
thrombosisInFamily = sample(c(TRUE,FALSE), 50, rep = TRUE),
mentaldiseasesInFamily = sample(c(TRUE,FALSE), 50, rep = TRUE),
cardiadicdiseaseInFamily = sample(c(NA,TRUE,FALSE), 50, rep = TRUE),
```

```
readmission = sample(c(NA,TRUE,FALSE), 50, rep = TRUE))

attach(data)
tableStack(cancerInFamily:cardiadicdiseaseInFamily, dataFrame = data)
detach(data)
tableStack(cancerInFamily:cardiadicdiseaseInFamily, data) # Default data frame is data
# "by" compares variables
tableStack(cancerInFamily:cardiadicdiseaseInFamily, data, by= readmission)
# "prevalence" returns the prevalence instead of the absolute values
tableStack(cancerInFamily:cardiadicdiseaseInFamily, data,
by= readmission, prevalence=TRUE)
# "percent" as FALSE hides the percentage in parenthesis
tableStack(cancerInFamily:cardiadicdiseaseInFamily, data,
by= readmission, percent=FALSE)
# "name.test" as FALSE hides the column that shows the tests names
tableStack(cancerInFamily:cardiadicdiseaseInFamily, data,
by= readmission, percent=FALSE, name.test=FALSE)
# "NAcol" displays a column of NA values on the variable on "by"
tableStack(cancerInFamily:cardiadicdiseaseInFamily, data,
by= readmission, NAcol = TRUE)
# "NArow" displays rows of NA values on the variables on "vars"
tableStack(cancerInFamily:cardiadicdiseaseInFamily, data,
by= readmission, NAcol = TRUE, NArow = TRUE)

# the specification of the vars may be done as the range
tableStack(vars=2:7, data, by=sex)
# "by" var may be specified as "none" and the selected vars will be crossed only against the total
tableStack(vars=2:7, data, by="none")
# by = NONE works just as by = "none"
tableStack(vars=2:7, data, by = NONE)
# total.column displays a column of totals in adition to the variable on by
tableStack(vars=2:7, data, by=sex, total.column=TRUE)

var.labels <- c("sex", "Type of admission for each patient",
"age", "Duration time in days of the patient's hospitalization",
"Number of children that the patient have",
"whether or not the patient has cancer in family",
"whether or not the patient has diabetes in family",
"whether or not the patient has thrombosis in family",
"whether or not the patient has mental diseases in family",
"whether or not the patient has cardiac diseases in family",
"whether or not the patient is on a relapse admission")
#setting the attribute var.labels
attr(data, "var.labels") <- var.labels
rm(var.labels)

# May need full screen of Rconsole
tableStack(vars=c(numberOfChildren,hospitalizationTime), data)
# Fits in with default R console screen
tableStack(vars=c(numberOfChildren,hospitalizationTime), data,
var.labels.trunc=35)
tableStack(vars=c(age,numberOfChildren,hospitalizationTime),
data, reverse=TRUE) -> a
```

```
a
## Components of 'a' have appropriate items reversed
a$mean.score -> mean.score
a$total.score -> total.score
data$mean.score <- mean.score
data$total.score <- total.score

# hiding the test column
tableStack(c(age, numberOfChildren,hospitalizationTime,
mean.score,total.score), data, by=sex, test=FALSE)
# variables specified on iqr will not display SD but IQR instead
tableStack(3:5, data, by=sex, iqr=hospitalizationTime)
## 'vars' can be mixture of variables of different classes
tableStack(3:5, data, by=admissionType,
iqr=c(hospitalizationTime, total.score))

data$highscore <- mean.score > 4
# a variable with some comparison may be created easily
tableStack(mean.score:highscore, data,
by=sex, iqr=total.score)

# the percentage information may be hidden
tableStack(vars=c(readmission,admissionType),
data, by=sex, percent="none")
# it may be shown the prevalende of the
# variable instead of the values themselves
tableStack(vars=c(readmission,admissionType), data,
by=sex, prevalence = TRUE)
# the name of the tests may be hidden
# while the test itself still shows
tableStack(vars=c(readmission,admissionType), data,
by=sex, name.test = FALSE)

## Variable in numeric or factor
# as continuous varaibles
tableStack(vars=3:5, data, by=sex)
# as factors
tableStack(vars=3:5, data, by=sex, vars.to.factor = 3:5)

## Using drplvls
# a dataframe will be created containing a factor with an unused level
bloodbank <- data.frame(AgeInDays =
    sample(0:15,200, replace = TRUE), Type =
    factor(sample(c("A","B","0"), 200, replace = TRUE),
      levels = c("A","B","AB","0")), Origin =
    sample(c("US","CA"), 200, replace = TRUE))

# by using drplvls the row of the unused fator is hidden
tableStack(vars = c(AgeInDays, Type),
bloodbank, by = Origin) #usual
tableStack(vars = c(AgeInDays, Type),
bloodbank, by = Origin,
drplvls = TRUE) # with drplvls
```

```
rm(total.score, mean.score, a, data, bloodbank)
```

Index