Package ‘CalibrationCurves’

July 21, 2025

Type Package

Title Calibration Performance

Version 2.0.7

Description Plots calibration curves and computes statistics for assessing calibration performance. See De Cock Campo (2023) <doi:10.48550/arXiv.2309.08559> and Van Calster et al. (2016) <doi:10.1016/j.jclinepi.2015.12.005>.

License GPL (>= 3)

LazyData TRUE

Depends R (>= 3.5.0), rms (>= 7.0-0), ggplot2

Imports grDevices, graphics, methods, stats, utils, survival, Hmisc, bookdown, rstudioapi, timeROC, riskRegression

Suggests knitr, rmarkdown, mgcv, MASS, magrittr, Matrix

RoxygenNote 7.3.1

Encoding UTF-8

VignetteBuilder knitr

URL `https://bavodc.github.io/websiteCalibrationCurves/`

NeedsCompilation no

Author De Cock Bavo [aut, cre],
Nieboer Daan [aut],
Van Calster Ben [aut],
Steyerberg Ewout [aut],
Vergouwe Yvonne [aut]

Maintainer De Cock Bavo <`bavo.decock@kuleuven.be`>

Repository CRAN

Date/Publication 2025-07-08 00:00:10 UTC

Contents

`.rcspline.plot` *Internal function*

Description

Adjusted version of the `rcspline.plot` function where only the output is returned and no plot is made

Usage

```
.rcspline.plot(
  x,
  y,
  model = c("logistic", "cox", "ols"),
  xrange,
  event,
  nk = 5,
  knots = NULL,
  show = c("xbeta", "prob"),
  adj = NULL,
  xlab,
  ylab,
  ylim,
  plim = c(0, 1),
  plotcl = TRUE,
  showknots = TRUE,
  add = FALSE,
  plot = TRUE,
  subset,
  lty = 1,
```

```
    noprint = FALSE,
    m,
    smooth = FALSE,
    bass = 1,
    main = "auto",
    statloc
  )
```

Arguments

`x`	a numeric predictor
`y`	a numeric response. For binary logistic regression, `y` should be either 0 or 1.
`model`	`"logistic"` or `"cox"`. For `"cox"`, uses the `coxph.fit` function with `method="efron"` argument set.
`xrange`	range for evaluating `x`, default is f and $1-f$ quantiles of `x`, where $f = \frac{10}{\max{(n,200)}}$ and n the number of observations
`event`	event/censoring indicator if `model="cox"`. If `event` is present, `model` is assumed to be `"cox"`
`nk`	number of knots
`knots`	knot locations, default based on quantiles of `x` (by `rcspline.eval`)
`show`	`"xbeta"` or `"prob"` - what is plotted on `y`-axis
`adj`	optional matrix of adjustment variables
`xlab`	`x`-axis label, default is the “label” attribute of `x`
`ylab`	`y`-axis label, default is the “label” attribute of `y`
`ylim`	`y`-axis limits for logit or log hazard
`plim`	`y`-axis limits for probability scale
`plotcl`	plot confidence limits
`showknots`	show knot locations with arrows
`add`	add this plot to an already existing plot
`plot`	logical to indicate whether a plot has to be made. `FALSE` suppresses the plot.
`subset`	subset of observations to process, e.g. `sex == "male"`
`lty`	line type for plotting estimated spline function
`noprint`	suppress printing regression coefficients and standard errors
`m`	for `model="logistic"`, plot grouped estimates with triangles. Each group contains `m` ordered observations on `x`.
`smooth`	plot nonparametric estimate if `model="logistic"` and `adj` is not specified
`bass`	smoothing parameter (see `supsmu`)
`main`	main title, default is `"Estimated Spline Transformation"`
`statloc`	location of summary statistics. Default positioning by clicking left mouse button where upper left corner of statistics should appear. Alternative is `"ll"` to place below the graph on the lower left, or the actual `x` and `y` coordinates. Use `"none"` to suppress statistics.

Value

list with components (‘`knots`’, ‘`x`’, ‘`xbeta`’, ‘`lower`’, ‘`upper`’) which are respectively the knot locations, design matrix, linear predictor, and lower and upper confidence limits

See Also

`lrm`, `cph`, `rcspline.eval`, `plot`, `supsmu`, `coxph.fit`, `lrm.fit`

`auc.nonpara.mw` *AUC Based on the Mann-Whitney Statistic*

Description

Obtain the point estimate and the confidence interval of the AUC by various methods based on the Mann-Whitney statistic.

Usage

```
auc.nonpara.mw(x, y, conf.level=0.95,
               method=c("newcombe", "pepe", "delong",
                        "jackknife", "bootstrapP", "bootstrapBCa"),
               nboot)
```

Arguments

`x`	a vector of observations from class P.
`y`	a vector of observations from class N.
`conf.level`	confidence level of the interval. The default is 0.95.
`method`	a method used to construct the CI. `newcombe` is the method recommended in Newcombe (2006); `pepe` is the method proposed in Pepe (2003); `delong` is the method proposed in Delong et al. (1988); `jackknife` uses the jackknife method; `bootstrapP` uses the bootstrap with percentile CI; `bootstrapBCa` uses bootstrap with bias-corrected and accelerated CI. The default is `newcombe`. It can be abbreviated.
`nboot`	number of bootstrap iterations.

Details

The function implements various methods based on the Mann-Whitney statistic.

Value

Point estimate and lower and upper bounds of the CI of the AUC.

Note

The observations from class P tend to have larger values than that from class N.

This help-file is a copy of the original help-file of the function `auc.nonpara.mw` from the auRoc-package. It is important to note that, when using `method="pepe"`, the confidence interval is computed as documented in Qin and Hotilovac (2008) and that this is different from the original function.

References

Elizabeth R Delong, David M Delong, and Daniel L Clarke-Pearson (1988) Comparing the areas under two or more correlated receiver operating characteristic curves: a nonparametric approach. *Biometrics* **44** 837-845

Dai Feng, Giuliana Cortese, and Richard Baumgartner (2015) A comparison of confidence/credible interval methods for the area under the ROC curve for continuous diagnostic tests with small sample size. *Statistical Methods in Medical Research* DOI: 10.1177/0962280215602040

Robert G Newcombe (2006) Confidence intervals for an effect size measure based on the Mann-Whitney statistic. Part 2: asymptotic methods and evaluation. *Statistics in medicine* **25(4)** 559-573

Margaret Sullivan Pepe (2003) The statistical evaluation of medical tests for classification and prediction. *Oxford University Press*

Qin, G., & Hotilovac, L. (2008). Comparison of non-parametric confidence intervals for the area under the ROC curve of a continuous-scale diagnostic test. *Statistical Methods in Medical Research*, **17(2)**, pp. 207-21

`CalibrationCurves` *General information on the package and its functions*

Description

Using this package, you can assess the calibration performance of your prediction model. That is, to which extent the predictions and correspond with what we observe empirically. To assess the calibration of model with a binary outcome, you can use the `val.prob.ci.2` or the `valProbggplot` function. If the outcome of your prediction model is not binary but follows a different distribution of the exponential family, you can employ the `genCalCurve` function.

If you are not familiar with the theory and/or application of calibration, you can consult the vignette of the package. This vignette provides a comprehensive overview of the theory and contains a tutorial with some practical examples. Further, we suggest the reader to consult the paper on generalized calibration curves on arXiv. In this paper, we provide the theoretical background on the generalized calibration framework and illustrate its applicability with some prototypical examples of both statistical and machine learning prediction models that are well-calibrated, overfit and underfit.

Originally, the package only contained functions to assess the calibration of prediction models with a binary outcome. The details section provides some background information on the history of the package's development.

Details

Some years ago, Yvonne Vergouwe and Ewout Steyerberg adapted the function `val.prob` from the rms-package (`https://cran.r-project.org/package=rms`) into `val.prob.ci` and added the following functions to `val.prob`:

- Scaled Brier score by relating to max for average calibrated Null model
- Risk distribution according to outcome
- 0 and 1 to indicate outcome label; set with `d1lab=".."`, `d0lab=".."`
- Labels: y axis: "Observed Frequency"; Triangle: "Grouped observations"
- Confidence intervals around triangles
- A cut-off can be plotted; set x coordinate

In December 2015, Bavo De Cock, Daan Nieboer, and Ben Van Calster adapted this to `val.prob.ci.2`:

- Flexible calibration curves can be obtained using loess (default) or restricted cubic splines, with pointwise 95% confidence intervals. Flexible calibration curves are now given by default and this decision is based on the findings reported in Van Calster et al. (2016).
- Loess: confidence intervals can be obtained in closed form or using bootstrapping (CL.BT=T will do bootstrapping with 2000 bootstrap samples, however this will take a while)
- RCS: 3 to 5 knots can be used
  - the knot locations will be estimated using default quantiles of x (by `rcspline.eval`, see `rcspline.plot` and `rcspline.eval`)
  - if estimation problems occur at the specified number of knots (nr.knots, default is 5), the analysis is repeated with nr.knots-1 until the problem has disappeared and the function stops if there is still an estimation problem with 3 knots
- You can now adjust the plot through use of normal plot commands (`cex.axis` etcetera), and the size of the legend now has to be specified in `cex.leg`
- Label y-axis: "Observed proportion"
- Stats: added the Estimated Calibration Index (ECI), a statistical measure to quantify lack of calibration (Van Hoorde et al., 2015)
- Stats to be shown in the plot: by default we show the `"abc"` of model performance (Steyerberg et al., 2011). That is, calibration intercept (calibration-in-the-large), calibration slope and c- statistic. Alternatively, the user can select the statistics of choice (e.g. `dostats=c("C (ROC)","R2")` or `dostats=c(2,3)`.
- Vectors p, y and logit no longer have to be sorted

In 2023, Bavo De Cock (Campo) published a paper that introduces the generalized calibration framework. This framework is an extension of the logistic calibration framework to prediction models where the outcome's distribution is a member of the exponential family. As such, we are able to assess the calibration of a wider range of prediction models. The methods in this paper are implemented in the CalibrationCurves package.

The most current version of this package can always be found on `https://github.com/BavoDC` and can easily be installed using the following code:

```
install.packages("devtools") # if not yet installed
require(devtools)
install_github("BavoDC/CalibrationCurves", dependencies = TRUE, build_vignettes = TRUE)
```

References

De Cock Campo, B. (2023). Towards reliable predictive analytics: a generalized calibration framework. arXiv:2309.08559, available at https://arxiv.org/abs/2309.08559.

Steyerberg, E.W.Van Calster, B., Pencina, M.J. (2011). Performance measures for prediction models and markers : evaluation of predictions and classifications. *Revista Espanola de Cardiologia*, **64(9)**, pp. 788-794

Van Calster, B., Nieboer, D., Vergouwe, Y., De Cock, B., Pencina M., Steyerberg E.W. (2016). A calibration hierarchy for risk models was defined: from utopia to empirical data. *Journal of Clinical Epidemiology*, **74**, pp. 167-176

Van Hoorde, K., Van Huffel, S., Timmerman, D., Bourne, T., Van Calster, B. (2015). A spline-based tool to assess and visualize the calibration of multiclass risk predictions. *Journal of Biomedical Informatics*, **54**, pp. 283-93

`genCalCurve` *Calibration performance using the generalized calibration framework*

Description

Function to assess the calibration performance of a prediction model where the outcome's distribution is a member of the exponential family (De Cock Campo, 2023). The function plots the generalized calibration curve and computes the generalized calibration slope and intercept.

Usage

```
genCalCurve(
  y,
  yHat,
  family,
  plot = TRUE,
  Smooth = FALSE,
  GLMCal = TRUE,
  lwdIdeal = 2,
  colIdeal = "gray",
  ltyIdeal = 1,
  lwdSmooth = 1,
  colSmooth = "blue",
  ltySmooth = 1,
  argzSmooth = alist(degree = 2),
  lwdGLMCal = 1,
  colGLMCal = "red",
  ltyGLMCal = 1,
  AddStats = T,
  Digits = 3,
  cexStats = 1,
  lwdLeg = 1.5,
```

```
  Legend = TRUE,
  legendPos = "bottomright",
  xLim = NULL,
  yLim = NULL,
  posStats = NULL,
  confLimitsSmooth = c("none", "bootstrap", "pointwise"),
  confLevel = 0.95,
  Title = "Calibration plot",
  xlab = "Predicted value",
  ylab = "Empirical average",
  EmpiricalDistribution = TRUE,
  length.seg = 1,
  ...
)
```

Arguments

`y`	a vector with the values for the response variable
`yHat`	a vector with the predicted values
`family`	a description of the type of distribution and link function in the model. This can be a character string naming a family function, a family function or the result of a call to a family function. (See family for details of family functions.)
`plot`	logical, indicating if a plot should be made or not.
`Smooth`	logical, indicating if the flexible calibration curve should be estimated.
`GLMCal`	logical, indicating if the GLM calibration curve has to be estimated.
`lwdIdeal`	the line width of the ideal line.
`colIdeal`	the color of the ideal line.
`ltyIdeal`	the line type of the ideal line.
`lwdSmooth`	the line width of the flexible calibration curve.
`colSmooth`	the color of the flexible calibration curve.
`ltySmooth`	the line type of the flexible calibration curve.
`argzSmooth`	arguments passed to `loess`.
`lwdGLMCal`	the line width of the GLM calibration curve.
`colGLMCal`	the color of the GLM calibration curve.
`ltyGLMCal`	the line type of the GLM calibration curve.
`AddStats`	logical, indicating whether to add the values of the generalized calibration slope and intercept to the plot.
`Digits`	the number of digits of the generalized calibration slope and intercept.
`cexStats`	the font size of the statistics shown on the plot.
`lwdLeg`	the line width in the legend.
`Legend`	logical, indicating whether the legend has to be added.
`legendPos`	the position of the legend on the plot.

`xLim, yLim`	numeric vectors of length 2, giving the x and y coordinates ranges (see `plot.window`)
`posStats`	numeric vector of length 2, specifying the x and y coordinates of the statistics (generalized calibration curve and intercept) printed on the plot. Default is `NULL` which places the statistics in the top left corner of the plot.
`confLimitsSmooth`	character vector to indicate if and how the confidence limits for the flexible calibration curve have to be computed. `"none"` omits the confidence limits, `"bootstrap"` uses 2000 bootstrap samples to calculate the 95% confidence limits and `"pointwise"` uses the pointwise confidence limits.
`confLevel`	the confidence level for the calculation of the pointwise confidence limits of the flexible calibration curve.
`Title`	the title of the plot
`xlab`	x-axis label, default is `"Predicted value"`.
`ylab`	y-axis label, default is `"Empirical average"`.
`EmpiricalDistribution`	logical, indicating if the empirical distribution of the predicted values has to be added to the bottom of the plot.
`length.seg`	controls the length of the histogram lines. Default is `1`.
`...`	arguments to be passed to `plot`, see `par`

Value

An object of type `GeneralizedCalibrationCurve` with the following slots:

`call`	the matched call.
`ggPlot`	the ggplot object.
`stats`	a vector containing performance measures of calibration.
`cl.level`	the confidence level used.
`Calibration`	contains the calibration intercept and slope, together with their confidence intervals.
`Cindex`	the value of the c-statistic, together with its confidence interval.
`warningMessages`	if any, the warning messages that were printed while running the function.
`CalibrationCurves`	The coordinates for plotting the calibration curves.

References

De Cock Campo, B. (2023). Towards reliable predictive analytics: a generalized calibration framework. arXiv:2309.08559, available at `https://arxiv.org/abs/2309.08559`.

Examples

```
library(CalibrationCurves)
library(mgcv)
data("poissontraindata")
data("poissontestdata")

glmFit = glm(Y ~ ., data = poissontraindata, family = poisson)

# Example of a well calibrated poisson prediction model
yOOS = poissontestdata$Y
yHat = predict(glmFit, newdata = poissontestdata, type = "response")
genCalCurve(yOOS, yHat, family = "poisson", plot = TRUE)

# Example of an overfit poisson prediction model
gamFit = gam(Y ~ x1 + x3 + x1:x3 + s(x5), data = poissontraindata, family = poisson)
yHat = as.vector(predict(gamFit, newdata = poissontestdata, type = "response"))
genCalCurve(yOOS, yHat, family = "poisson", plot = TRUE)

# Example of an underfit poisson prediction model
glmFit = glm(Y ~ x2, data = poissontraindata, family = poisson)
yOOS = poissontestdata$Y
yHat = predict(glmFit, newdata = poissontestdata, type = "response")
genCalCurve(yOOS, yHat, family = "poisson", plot = TRUE)
```

`LibraryM` *Function to load multiple packages at once*

Description

Function to load multiple packages at once

Usage

```
LibraryM(...)
```

Arguments

`...`	the packages that you want to load

Value

invisible `NULL`

Examples

```
LibraryM(CalibrationCurves)
```

---

print.CalibrationCurve

Print function for a CalibrationCurve object

Description

Prints the call, confidence level and values for the performance measures.

Usage

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

Arguments

`x`	an object of type CalibrationCurve, resulting from `val.prob.ci.2`.
`...`	arguments passed to `print`

Value

The original `CalibrationCurve` object is returned.

See Also

`val.prob.ci.2`

---

print.GeneralizedCalibrationCurve

Print function for a GeneralizedCalibrationCurve object

Description

Prints the call, confidence level and values for the performance measures.

Usage

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

Arguments

`x`	an object of type GeneralizedCalibrationCurve, resulting from `genCalCurve`.
`...`	arguments passed to `print`

Value

The original `GeneralizedCalibrationCurve` object is returned.

See Also

`genCalCurve`

`print.ggplotCalibrationCurve`

Print function for a ggplotCalibrationCurve object

Description

Prints the ggplot, call, confidence level and values for the performance measures.

Usage

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

Arguments

`x`	an object of type ggplotCalibrationCurve, resulting from `valProbggplot`.
`...`	arguments passed to `print`

Value

The original `ggplotCalibrationCurve` object is returned.

See Also

`valProbggplot`

`print.SurvivalCalibrationCurve`

Print function for a SurvivalCalibrationCurve object

Description

Print function for a SurvivalCalibrationCurve object

Usage

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

Arguments

`x`	an object of type SurvivalCalibrationCurve, resulting from `valProbSurvival`.
`...`	arguments passed to `print`

Value

The original `SurvivalCalibrationCurve` object is returned.

See Also

`valProbSurvival`

`simulateddata` *Simulated data sets to illustrate the package functionality*

Description

Both the `traindata` and `testdata` dataframe are synthetically generated data sets to illustrate the functionality of the package. The `traindata` has 1000 observations and the `testdata` has 500 observations. The same settings were used to generate both data sets.

Usage

```
data(traindata)
data(testdata)
```

Format

`y`	the binary outcome variable
`x1`	covariate 1
`x2`	covariate 2
`x3`	covariate 3
`x4`	covariate 4

Details

See the examples for how the data sets were generated.

Examples

```
# The data sets were generated as follows
set.seed(1782)

# Simulate training data
nTrain    = 1000
B         = c(0.1, 0.5, 1.2, -0.75, 0.8)
X         = replicate(4, rnorm(nTrain))
p0true    = binomial()$linkinv(cbind(1, X) %*% B)
y         = rbinom(nTrain, 1, p0true)
colnames(X) = paste0("x", seq_len(ncol(X)))
traindata = data.frame(y, X)

# Simulate validation data
nTest    = 500
X        = replicate(4, rnorm(nTest))
p0true   = binomial()$linkinv(cbind(1, X) %*% B)
y        = rbinom(nTest, 1, p0true)
colnames(X) = paste0("x", seq_len(ncol(X)))
testdata = data.frame(y, X)
```

`simulatedpoissondata` *Simulated data sets to illustrate the package functionality*

Description

Both the `traindata` and `testdata` dataframe are synthetically generated data sets to illustrate the functionality of the package. The `traindata` has 5000 observations and the `testdata` has 1000 observations. The same settings were used to generate both data sets.

Usage

```
data(poissontraindata)
data(poissontestdata)
```

Format

`y`	the poisson distributed outcome variable
`x1`	covariate 1
`x2`	covariate 2
`x3`	covariate 3
`x4`	covariate 4
`x5`	covariate 5

Details

See the examples for how the data sets were generated.

Examples

```
# The data sets were generated as follows
library(MASS)
library(magrittr)
ScaleRange <- function(x, xmin = -1, xmax = 1) {
xRange = range(x)
(x - xRange[1]) / diff(xRange) * (xmax - xmin) + xmin
}

set.seed(144)
p    = 5
N    = 1e6
n    = 5e3
nOOS = 1e3
S    = matrix(NA, 5, 5)
rho  = c(0.025, 0, 0, 0.05, 0.075, 0, 0, 0.025, 0, 0)
S[upper.tri(S)] = rho
S[lower.tri(S)] = t(S)[lower.tri(S)]
diag(S) = 1
Matrix::isSymmetric(S)


X  = mvrnorm(N, rep(0, p), Sigma = S, empirical = TRUE)
X  = apply(X, 2, ScaleRange)
B  = c(-2.3, 1.5, 2, -1, -2, -1.5)
mu = poisson()$linkinv(cbind(1, X) %*% B)
Y  = rpois(N, mu)

Df = data.frame(Y, X)
colnames(Df)[-1] %<>% tolower()

set.seed(2)
DfS   = Df[sample(1:nrow(Df), n, FALSE), ]
DfOOS = Df[sample(1:nrow(Df), nOOS, FALSE), ]

poissontraindata = DfS
poissontestdata  = DfOOS
```

`simulatedsurvivaldata` *Breast Cancer Survival Data from Rotterdam and Germany*

Description

The training dataset contains real-life survival data from patients who underwent primary surgery for breast cancer between 1978 and 1993 in Rotterdam. The patients were followed until 2007, resulting in a model development cohort of 2982 patients after exclusions. The primary outcome measured was recurrence-free survival, defined as the time from primary surgery to recurrence or death.

The validation dataset consists of 686 patients with primary node-positive breast cancer from the German Breast Cancer Study Group. In this cohort, 285 patients suffered a recurrence or died within 5 years of follow-up, while 280 were censored before 5 years. Five-year predictions were chosen as that was the lowest median survival from the two cohorts (Rotterdam cohort, 6.7 years; German cohort, 4.9 years).

Usage

```
data(trainDataSurvival)
data(testDataSurvival)
```

Format

A data frame with observations on the following 26 variables.

pid patient identifier

year year of surgery

age age at surgery

meno menopausal status (0 = premenopausal, 1 = postmenopausal)

size tumor size, a factor with levels <= 20, 20-50, >50

grade differentiation grade

nodes number of positive lymph nodes

pgr progesterone receptors (fmol/l)

er estrogen receptors (fmol/l)

hormon hormonal treatment (0 = no, 1 = yes)

chemo chemotherapy

rtime days to relapse or last follow-up

recur 0 = no relapse, 1 = relapse

dtime days to death or last follow-up

death 0 = alive, 1 = dead

ryear Follow-up time for RFS, in years (numeric)

rfs Recurrence-free survival status (0 = no event, 1 = event) (numeric)

pgr2 Winsorized progesterone receptor level (numeric)

nodes2 Winsorized node count (numeric)

csize Categorized tumor size, copied from `size` (factor)

cnode Categorized node involvement (factor: "0", "1-3", ">3")

grade3 Recoded grade factor (levels: "1-2", "3")

nodes3 Restricted cubic spline basis for `nodes2` (numeric)

pgr3 Restricted cubic spline basis for original `pgr` (numeric)

epoch Follow-up epoch indicator after splitting at 5 years (numeric)

Details

The data sets are based on the publicly available code and data used in the repository Prediction_performance_survival by Giardiello et al. (2023), which accompanies the Annals of Internal Medicine article "Assessing Performance and Clinical Usefulness in Prediction Models With Survival Outcomes: Practical Guidance for Cox Proportional Hazards Models".

All preprocessing steps, such as converting survival time to years, defining recurrence-free survival status via ‘rfs = pmax(recur, death)‘, correcting 43 discordant cases using death time, 99th-percentile winsorization of ‘pgr‘ and ‘nodes‘, spline transformations (‘nodes3‘, ‘pgr3‘), splitting follow-up at 5 years (‘epoch‘), and recoding categorical variables (‘csize‘, ‘cnode‘, ‘grade3‘)—were performed exactly as in the Giardiello code.

The training dataset, `trainDataSurvival`, consists of 2982 patients, with 1713 events occurring over a maximum follow-up time of 19.3 years. The estimated median potential follow-up time, calculated using the reverse Kaplan- method, was 9.3 years. Out of these patients, 1275 suffered a recurrence or death within the follow-up time of interest (5 years), and 126 were censored before 5 years.

The validation dataset, `testDataSurvival`, consists of 686 patients with primary node-positive breast cancer from the German Breast Cancer Study Group. In this cohort, 285 patients suffered a recurrence or died within 5 years of follow-up, while 280 were censored before 5 years. Five-year predictions were chosen as that was the lowest median survival from the two cohorts (Rotterdam cohort, 6.7 years; German cohort, 4.9 years).

References

David J. McLernon, Daniele Giardiello, Ben Van Calster, et al. (2023). Assessing Performance and Clinical Usefulness in Prediction Models With Survival Outcomes: Practical Guidance for Cox Proportional Hazards Models. *Annals of Internal Medicine*, 176(1), pp. 105-114, doi:10.7326/M22-0844

Examples

```
data(testDataSurvival)
## Explore the structure of the dataset
str(testDataSurvival)
```

`val.prob.ci.2` *Calibration performance*

Description

The function `val.prob.ci.2` is an adaptation of `val.prob` from Frank Harrell’s rms package, `https://cran.r-project.org/package=rms`. Hence, the description of some of the functions of `val.prob.ci.2` come from the the original `val.prob`.

The key feature of `val.prob.ci.2` is the generation of logistic and flexible calibration curves and related statistics. When using this code, please cite: Van Calster, B., Nieboer, D., Vergouwe, Y., De Cock, B., Pencina, M.J., Steyerberg, E.W. (2016). A calibration hierarchy for risk models was defined: from utopia to empirical data. *Journal of Clinical Epidemiology*, **74**, pp. 167-176

Usage

```
val.prob.ci.2(
  p,
  y,
  logit,
  group,
  weights = rep(1, length(y)),
  normwt = FALSE,
  pl = TRUE,
  smooth = c("loess", "rcs", "none"),
  CL.smooth = "fill",
  CL.BT = FALSE,
  lty.smooth = 1,
  col.smooth = "black",
  lwd.smooth = 1,
  nr.knots = 5,
  logistic.cal = FALSE,
  lty.log = 1,
  col.log = "black",
  lwd.log = 1,
  xlab = "Predicted probability",
  ylab = "Observed proportion",
  xlim = c(-0.02, 1),
  ylim = c(-0.15, 1),
  m,
  g,
  cuts,
  emax.lim = c(0, 1),
  legendloc = c(0.5, 0.27),
  statloc = c(0, 0.85),
  dostats = TRUE,
  cl.level = 0.95,
  method.ci = "pepe",
  roundstats = 2,
  riskdist = "predicted",
  cex = 0.75,
  cex.leg = 0.75,
  connect.group = FALSE,
  connect.smooth = TRUE,
  g.group = 4,
  evaluate = 100,
  nmin = 0,
  d0lab = "0",
  d1lab = "1",
  cex.d01 = 0.7,
  dist.label = 0.04,
  line.bins = -0.05,
  dist.label2 = 0.03,
```

```
  cutoff,
  las = 1,
  length.seg = 1,
  y.intersp = 1,
  lty.ideal = 1,
  col.ideal = "red",
  lwd.ideal = 1,
  allowPerfectPredictions = FALSE,
  argzLoess = alist(degree = 2),
  ...
)
```

Arguments

`p`	predicted probability
`y`	vector of binary outcomes
`logit`	predicted log odds of outcome. Specify either `p` or `logit`.
`group`	a grouping variable. If numeric this variable is grouped into `g.group` quantile groups (default is quartiles). Set `group=TRUE` to use the `group` algorithm but with a single stratum for `val.prob`.
`weights`	an optional numeric vector of per-observation weights (usually frequencies), used only if `group` is given.
`normwt`	set to `TRUE` to make `weights` sum to the number of non-missing observations.
`pl`	`TRUE` to plot the calibration curve(s). If `FALSE` no calibration curves will be plotted, but statistics will still be computed and outputted.
`smooth`	`"loess"` generates a flexible calibration curve based on `loess`, `"rcs"` generates a calibration curves based on restricted cubic splines (see `rcs` and `rcspline.plot`), `"none"` suppresses the flexible curve. We recommend to use loess unless N is large, for example N>5000. Default is `"loess"`.
`CL.smooth`	`"fill"` shows pointwise 95% confidence limits for the flexible calibration curve with a gray area between the lower and upper limits, `TRUE` shows pointwise 95% confidence limits for the flexible calibration curve with dashed lines, `FALSE` suppresses the confidence limits. Default is `"fill"`.
`CL.BT`	`TRUE` uses confidence limits based on 2000 bootstrap samples, `FALSE` uses closed form confidence limits. Default is `FALSE`.
`lty.smooth`	the linetype of the flexible calibration curve. Default is `1`.
`col.smooth`	the color of the flexible calibration curve. Default is `"black"`.
`lwd.smooth`	the line width of the flexible calibration curve. Default is `1`.
`nr.knots`	specifies the number of knots for rcs-based calibration curve. The default as well as the highest allowed value is 5. In case the specified number of knots leads to estimation problems, then the number of knots is automatically reduced to the closest value without estimation problems.
`logistic.cal`	`TRUE` plots the logistic calibration curve, `FALSE` suppresses this curve. Default is `FALSE`.

`lty.log`	if `logistic.cal=TRUE`, the linetype of the logistic calibration curve. Default is `1`.
`col.log`	if `logistic.cal=TRUE`, the color of the logistic calibration curve. Default is `"black"`.
`lwd.log`	if `logistic.cal=TRUE`, the line width of the logistic calibration curve. Default is `1`.
`xlab`	x-axis label, default is `"Predicted Probability"`.
`ylab`	y-axis label, default is `"Observed proportion"`.
`xlim, ylim`	numeric vectors of length 2, giving the x and y coordinates ranges (see `plot.window`)
`m`	If grouped proportions are desired, minimum no. observations per group
`g`	If grouped proportions are desired, number of quantile groups
`cuts`	If grouped proportions are desired, actual cut points for constructing intervals, e.g. `c(0,.1,.8,.9,1)` or `seq(0,1,by=.2)`
`emax.lim`	Vector containing lowest and highest predicted probability over which to compute `Emax`.
`legendloc`	if `pl=TRUE`, list with components `x,y` or vector `c(x,y)` for bottom right corner of legend for curves and points. Default is `c(.50, .27)` scaled to lim. Use `locator(1)` to use the mouse, `FALSE` to suppress legend.
`statloc`	the "abc" of model performance (Steyerberg et al., 2011)-calibration intercept, calibration slope, and c statistic-will be added to the plot, using statloc as the upper left corner of a box (default is c(0,.85). You can specify a list or a vector. Use locator(1) for the mouse, `FALSE` to suppress statistics. This is plotted after the curve legends.
`dostats`	specifies whether and which performance measures are shown in the figure. `TRUE` shows the `"abc"` of model performance (Steyerberg et al., 2011): calibration intercept, calibration slope, and c-statistic. `TRUE` is default. `FALSE` suppresses the presentation of statistics in the figure. A `c()` list of specific stats shows the specified stats. The key stats which are also mentioned in this paper are `"C (ROC)"` for the c statistic, `"Intercept"` for the calibration intercept, `"Slope"` for the calibration slope, and `"ECI"` for the estimated calibration index (Van Hoorde et al, 2015). The full list of possible statistics is taken from `val.prob` and augmented with the estimated calibration index: `"Dxy"`, `"C (ROC)"`, `"R2"`, `"D"`, `"D:Chi-sq"`, `"D:p"`, `"U"`, `"U:Chi-sq"`, `"U:p"`, `"Q"`, `"Brier"`, `"Intercept"`, `"Slope"`, `"Emax"`, `"Brier scaled"`, `"Eavg"`, `"ECI"`. These statistics are always returned by the function.
`cl.level`	if `dostats=TRUE`, the confidence level for the calculation of the confidence intervals of the calibration intercept, calibration slope and c-statistic. Default is `0.95`.
`method.ci`	method to calculate the confidence interval of the c-statistic. The argument is passed to `auc.nonpara.mw` from the auRoc-package and possible methods to compute the confidence interval are `"newcombe"`, `"pepe"`, `"delong"` or `"jackknife"`. Bootstrap-based methods are not available. The default method is `"pepe"` and here, the confidence interval is the logit-transformation-based confidence interval as documented in Qin and Hotilovac (2008). See `auc.nonpara.mw` for more information on the other methods.

`roundstats`	specifies the number of decimals to which the statistics are rounded when shown in the plot. Default is 2.
`riskdist`	Use `"calibrated"` to plot the relative frequency distribution of calibrated probabilities after dividing into 101 bins from `lim[1]` to `lim[2]`. Set to `"predicted"` (the default as of rms 4.5-1) to use raw assigned risk, `FALSE` to omit risk distribution. Values are scaled so that highest bar is `0.15*(lim[2]-lim[1])`.
`cex`, `cex.leg`	controls the font size of the statistics (`cex`) or plot legend (`cex.leg`). Default is 0.75
`connect.group`	Defaults to `FALSE` to only represent group fractions as triangles. Set to `TRUE` to also connect with a solid line.
`connect.smooth`	Defaults to `TRUE` to draw smoothed estimates using a line. Set to `FALSE` to instead use dots at individual estimates
`g.group`	number of quantile groups to use when `group` is given and variable is numeric.
`evaluate`	number of points at which to store the `lowess`-calibration curve. Default is 100. If there are more than `evaluate` unique predicted probabilities, `evaluate` equally-spaced quantiles of the unique predicted probabilities, with linearly interpolated calibrated values, are retained for plotting (and stored in the object returned by `val.prob`.
`nmin`	applies when `group` is given. When `nmin` > 0, `val.prob` will not store coordinates of smoothed calibration curves in the outer tails, where there are fewer than `nmin` raw observations represented in those tails. If for example `nmin`=50, the `plot` function will only plot the estimated calibration curve from a to b, where there are 50 subjects with predicted probabilities $< a$ and $> b$. `nmin` is ignored when computing accuracy statistics.
`d0lab`, `d1lab`	controls the labels for events and non-events (i.e. outcome y) for the histograms. Defaults are `d1lab="1"` for events and `d0lab="0"` for non-events.
`cex.d01`	controls the size of the labels for events and non-events. Default is 0.7.
`dist.label`	controls the horizontal position of the labels for events and non-events. Default is 0.04.
`line.bins`	controls the horizontal (y-axis) position of the histograms. Default is -0.05.
`dist.label2`	controls the vertical distance between the labels for events and non-events. Default is 0.03.
`cutoff`	puts an arrow at the specified risk cut-off(s). Default is none.
`las`	controls whether y-axis values are shown horizontally (1) or vertically (0).
`length.seg`	controls the length of the histogram lines. Default is `1`.
`y.intersp`	character interspacing for vertical line distances of the legend (`legend`)
`lty.ideal`	linetype of the ideal line. Default is `1`.
`col.ideal`	controls the color of the ideal line on the plot. Default is `"red"`.
`lwd.ideal`	controls the line width of the ideal line on the plot. Default is `1`.
`allowPerfectPredictions`	Logical, indicates whether perfect predictions (i.e. values of either 0 or 1) are allowed. Default is `FALSE`, since we transform the predictions using the logit

	transformation to calculate the calibration measures. In case of 0 and 1, this results in minus infinity and infinity, respectively. if `allowPerfectPredictions = TRUE`, 0 and 1 are replaced by 1e-8 and 1 - 1e-8, respectively.
`argzLoess`	a list with arguments passed to the `loess` function
`...`	arguments to be passed to `plot`, see `par`

Details

When using the predicted probabilities of an uninformative model (i.e. equal probabilities for all observations), the model has no predictive value. Consequently, where applicable, the value of the performance measure corresponds to the worst possible theoretical value. For the ECI, for example, this equals 1 (Edlinger et al., 2022).

Value

An object of type `CalibrationCurve` with the following slots:

`call`	the matched call.
`stats`	a vector containing performance measures of calibration.
`cl.level`	the confidence level used.
`Calibration`	contains the calibration intercept and slope, together with their confidence intervals.
`Cindex`	the value of the c-statistic, together with its confidence interval.
`warningMessages`	if any, the warning messages that were printed while running the function.
`CalibrationCurves`	The coordinates for plotting the calibration curves.

Note

In order to make use (of the functions) of the package auRoc, the user needs to install JAGS. However, since our package only uses the `auc.nonpara.mw` function which does not depend on the use of JAGS, we therefore copied the code and slightly adjusted it when `method="pepe"`.

References

Edlinger, M, van Smeden, M, Alber, HF, Wanitschek, M, Van Calster, B. (2022). Risk prediction models for discrete ordinal outcomes: Calibration and the impact of the proportional odds assumption. *Statistics in Medicine*, **41(8)**, pp. 1334– 1360

Qin, G., & Hotilovac, L. (2008). Comparison of non-parametric confidence intervals for the area under the ROC curve of a continuous-scale diagnostic test. *Statistical Methods in Medical Research*, **17(2)**, pp. 207-21

Steyerberg, E.W., Van Calster, B., Pencina, M.J. (2011). Performance measures for prediction models and markers : evaluation of predictions and classifications. *Revista Espanola de Cardiologia*, **64(9)**, pp. 788-794

Van Calster, B., Nieboer, D., Vergouwe, Y., De Cock, B., Pencina M., Steyerberg E.W. (2016). A calibration hierarchy for risk models was defined: from utopia to empirical data. *Journal of Clinical Epidemiology*, **74**, pp. 167-176

Van Hoorde, K., Van Huffel, S., Timmerman, D., Bourne, T., Van Calster, B. (2015). A spline-based tool to assess and visualize the calibration of multiclass risk predictions. *Journal of Biomedical Informatics*, **54**, pp. 283-93

Examples

```
# Load package
library(CalibrationCurves)
set.seed(1783)

# Simulate training data
X      = replicate(4, rnorm(5e2))
p0true = binomial()$linkinv(cbind(1, X) %*% c(0.1, 0.5, 1.2, -0.75, 0.8))
y      = rbinom(5e2, 1, p0true)
Df     = data.frame(y, X)

# Fit logistic model
FitLog = lrm(y ~ ., Df)

# Simulate validation data
Xval   = replicate(4, rnorm(5e2))
p0true = binomial()$linkinv(cbind(1, Xval) %*% c(0.1, 0.5, 1.2, -0.75, 0.8))
yval   = rbinom(5e2, 1, p0true)
Pred   = binomial()$linkinv(cbind(1, Xval) %*% coef(FitLog))

# Default calibration plot
val.prob.ci.2(Pred, yval)

# Adding logistic calibration curves and other additional features
val.prob.ci.2(Pred, yval, CL.smooth = TRUE, logistic.cal = TRUE, lty.log = 2,
 col.log = "red", lwd.log = 1.5)

val.prob.ci.2(Pred, yval, CL.smooth = TRUE, logistic.cal = TRUE, lty.log = 9,
col.log = "red", lwd.log = 1.5, col.ideal = colors()[10], lwd.ideal = 0.5)
```

`valProbggplot` *Calibration performance: ggplot version*

Description

The function `valProbggplot` is an adaptation of `val.prob` from Frank Harrell's rms package, `https://cran.r-project.org/package=rms`. Hence, the description of some of the functions of `valProbggplot` come from the the original `val.prob`.

The key feature of `valProbggplot` is the generation of logistic and flexible calibration curves and related statistics. When using this code, please cite: Van Calster, B., Nieboer, D., Vergouwe, Y.,

De Cock, B., Pencina, M.J., Steyerberg, E.W. (2016). A calibration hierarchy for risk models was defined: from utopia to empirical data. *Journal of Clinical Epidemiology*, **74**, pp. 167-176

Usage

```
valProbggplot(
  p,
  y,
  logit,
  group,
  weights = rep(1, length(y)),
  normwt = FALSE,
  pl = TRUE,
  smooth = c("loess", "rcs", "none"),
  CL.smooth = "fill",
  CL.BT = FALSE,
  lty.smooth = 1,
  col.smooth = "black",
  lwd.smooth = 1,
  nr.knots = 5,
  logistic.cal = FALSE,
  lty.log = 1,
  col.log = "black",
  lwd.log = 1,
  xlab = "Predicted probability",
  ylab = "Observed proportion",
  xlim = c(-0.02, 1),
  ylim = c(-0.15, 1),
  m,
  g,
  cuts,
  emax.lim = c(0, 1),
  legendloc = c(0.5, 0.27),
  statloc = c(0, 0.85),
  dostats = TRUE,
  cl.level = 0.95,
  method.ci = "pepe",
  roundstats = 2,
  riskdist = "predicted",
  size = 3,
  size.leg = 5,
  connect.group = FALSE,
  connect.smooth = TRUE,
  g.group = 4,
  evaluate = 100,
  nmin = 0,
  d0lab = "0",
  d1lab = "1",
  size.d01 = 5,
```

```
  dist.label = 0.01,
  line.bins = -0.05,
  dist.label2 = 0.04,
  cutoff,
  length.seg = 0.85,
  lty.ideal = 1,
  col.ideal = "red",
  lwd.ideal = 1,
  allowPerfectPredictions = FALSE,
  argzLoess = alist(degree = 2)
)
```

Arguments

`p`	predicted probability
`y`	vector of binary outcomes
`logit`	predicted log odds of outcome. Specify either `p` or `logit`.
`group`	a grouping variable. If numeric this variable is grouped into `g.group` quantile groups (default is quartiles). Set `group=TRUE` to use the `group` algorithm but with a single stratum for `val.prob`.
`weights`	an optional numeric vector of per-observation weights (usually frequencies), used only if `group` is given.
`normwt`	set to `TRUE` to make `weights` sum to the number of non-missing observations.
`pl`	`TRUE` to plot the calibration curve(s). If `FALSE` no calibration curves will be plotted, but statistics will still be computed and outputted.
`smooth`	`"loess"` generates a flexible calibration curve based on `loess`, `"rcs"` generates a calibration curves based on restricted cubic splines (see `rcs` and `rcspline.plot`), `"none"` suppresses the flexible curve. We recommend to use loess unless N is large, for example N>5000. Default is `"loess"`.
`CL.smooth`	`"fill"` shows pointwise 95% confidence limits for the flexible calibration curve with a gray area between the lower and upper limits, `TRUE` shows pointwise 95% confidence limits for the flexible calibration curve with dashed lines, `FALSE` suppresses the confidence limits. Default is `"fill"`.
`CL.BT`	`TRUE` uses confidence limits based on 2000 bootstrap samples, `FALSE` uses closed form confidence limits. Default is `FALSE`.
`lty.smooth`	the linetype of the flexible calibration curve. Default is `1`.
`col.smooth`	the color of the flexible calibration curve. Default is `"black"`.
`lwd.smooth`	the line width of the flexible calibration curve. Default is `1`.
`nr.knots`	specifies the number of knots for rcs-based calibration curve. The default as well as the highest allowed value is 5. In case the specified number of knots leads to estimation problems, then the number of knots is automatically reduced to the closest value without estimation problems.
`logistic.cal`	`TRUE` plots the logistic calibration curve, `FALSE` suppresses this curve. Default is `FALSE`.

`lty.log`	if `logistic.cal=TRUE`, the linetype of the logistic calibration curve. Default is `1`.
`col.log`	if `logistic.cal=TRUE`, the color of the logistic calibration curve. Default is `"black"`.
`lwd.log`	if `logistic.cal=TRUE`, the line width of the logistic calibration curve. Default is `1`.
`xlab`	x-axis label, default is `"Predicted Probability"`.
`ylab`	y-axis label, default is `"Observed proportion"`.
`xlim`, `ylim`	numeric vectors of length 2, giving the x and y coordinates ranges (see `xlim` and `ylim`).
`m`	If grouped proportions are desired, minimum no. observations per group
`g`	If grouped proportions are desired, number of quantile groups
`cuts`	If grouped proportions are desired, actual cut points for constructing intervals, e.g. `c(0,.1,.8,.9,1)` or `seq(0,1,by=.2)`
`emax.lim`	Vector containing lowest and highest predicted probability over which to compute `Emax`.
`legendloc`	if `pl=TRUE`, list with components `x,y` or vector `c(x,y)` for bottom right corner of legend for curves and points. Default is `c(.50, .27)` scaled to lim. Use `locator(1)` to use the mouse, `FALSE` to suppress legend.
`statloc`	the "abc" of model performance (Steyerberg et al., 2011)-calibration intercept, calibration slope, and c statistic-will be added to the plot, using statloc as the upper left corner of a box (default is c(0,.85). You can specify a list or a vector. Use locator(1) for the mouse, `FALSE` to suppress statistics. This is plotted after the curve legends.
`dostats`	specifies whether and which performance measures are shown in the figure. `TRUE` shows the `"abc"` of model performance (Steyerberg et al., 2011): calibration intercept, calibration slope, and c-statistic. `TRUE` is default. `FALSE` suppresses the presentation of statistics in the figure. A `c()` list of specific stats shows the specified stats. The key stats which are also mentioned in this paper are `"C (ROC)"` for the c statistic, `"Intercept"` for the calibration intercept, `"Slope"` for the calibration slope, and `"ECI"` for the estimated calibration index (Van Hoorde et al, 2015). The full list of possible statistics is taken from `val.prob` and augmented with the estimated calibration index: `"Dxy"`, `"C (ROC)"`, `"R2"`, `"D"`, `"D:Chi-sq"`, `"D:p"`, `"U"`, `"U:Chi-sq"`, `"U:p"`, `"Q"`, `"Brier"`, `"Intercept"`, `"Slope"`, `"Emax"`, `"Brier scaled"`, `"Eavg"`, `"ECI"`. These statistics are always returned by the function.
`cl.level`	if `dostats=TRUE`, the confidence level for the calculation of the confidence intervals of the calibration intercept, calibration slope and c-statistic. Default is `0.95`.
`method.ci`	method to calculate the confidence interval of the c-statistic. The argument is passed to `auc.nonpara.mw` from the auRoc-package and possible methods to compute the confidence interval are `"newcombe"`, `"pepe"`, `"delong"` or `"jackknife"`. Bootstrap-based methods are not available. The default method is `"pepe"` and here, the confidence interval is the logit-transformation-based confidence interval as documented in Qin and Hotilovac (2008). See `auc.nonpara.mw` for more information on the other methods.

`roundstats`	specifies the number of decimals to which the statistics are rounded when shown in the plot. Default is 2.
`riskdist`	Use `"calibrated"` to plot the relative frequency distribution of calibrated probabilities after dividing into 101 bins from `lim[1]` to `lim[2]`. Set to `"predicted"` (the default as of rms 4.5-1) to use raw assigned risk, `FALSE` to omit risk distribution. Values are scaled so that highest bar is `0.15*(lim[2]-lim[1])`.
`size`, `size.leg`	controls the font size of the statistics (`size`) or plot legend (`size.leg`). Default is 3 and 5, respectively.
`connect.group`	Defaults to `FALSE` to only represent group fractions as triangles. Set to `TRUE` to also connect with a solid line.
`connect.smooth`	Defaults to `TRUE` to draw smoothed estimates using a line. Set to `FALSE` to instead use dots at individual estimates
`g.group`	number of quantile groups to use when `group` is given and variable is numeric.
`evaluate`	number of points at which to store the `lowess`-calibration curve. Default is 100. If there are more than `evaluate` unique predicted probabilities, `evaluate` equally-spaced quantiles of the unique predicted probabilities, with linearly interpolated calibrated values, are retained for plotting (and stored in the object returned by `val.prob`.
`nmin`	applies when `group` is given. When `nmin` > 0, `val.prob` will not store coordinates of smoothed calibration curves in the outer tails, where there are fewer than `nmin` raw observations represented in those tails. If for example `nmin`=50, the `plot` function will only plot the estimated calibration curve from a to b, where there are 50 subjects with predicted probabilities $< a$ and $> b$. `nmin` is ignored when computing accuracy statistics.
`d0lab`, `d1lab`	controls the labels for events and non-events (i.e. outcome y) for the histograms. Defaults are `d1lab="1"` for events and `d0lab="0"` for non-events.
`size.d01`	controls the size of the labels for events and non-events. Default is 5.
`dist.label`	controls the horizontal position of the labels for events and non-events. Default is 0.01.
`line.bins`	controls the horizontal (y-axis) position of the histograms. Default is -0.05.
`dist.label2`	controls the vertical distance between the labels for events and non-events. Default is 0.03.
`cutoff`	puts an arrow at the specified risk cut-off(s). Default is none.
`length.seg`	controls the length of the histogram lines. Default is `0.85`.
`lty.ideal`	linetype of the ideal line. Default is `1`.
`col.ideal`	controls the color of the ideal line on the plot. Default is `"red"`.
`lwd.ideal`	controls the line width of the ideal line on the plot. Default is `1`.
`allowPerfectPredictions`	Logical, indicates whether perfect predictions (i.e. values of either 0 or 1) are allowed. Default is `FALSE`, since we transform the predictions using the logit transformation to calculate the calibration measures. In case of 0 and 1, this results in minus infinity and infinity, respectively. if `allowPerfectPredictions = TRUE`, 0 and 1 are replaced by 1e-8 and 1 - 1e-8, respectively.
`argzLoess`	a list with arguments passed to the `loess` function

Details

When using the predicted probabilities of an uninformative model (i.e. equal probabilities for all observations), the model has no predictive value. Consequently, where applicable, the value of the performance measure corresponds to the worst possible theoretical value. For the ECI, for example, this equals 1 (Edlinger et al., 2022).

Value

An object of type `ggplotCalibrationCurve` with the following slots:

`call`	the matched call.
`ggPlot`	the ggplot object.
`stats`	a vector containing performance measures of calibration.
`cl.level`	the confidence level used.
`Calibration`	contains the calibration intercept and slope, together with their confidence intervals.
`Cindex`	the value of the c-statistic, together with its confidence interval.
`warningMessages`	if any, the warning messages that were printed while running the function.
`CalibrationCurves`	The coordinates for plotting the calibration curves.

Note

In order to make use (of the functions) of the package auRoc, the user needs to install JAGS. However, since our package only uses the `auc.nonpara.mw` function which does not depend on the use of JAGS, we therefore copied the code and slightly adjusted it when `method="pepe"`.

References

Edlinger, M, van Smeden, M, Alber, HF, Wanitschek, M, Van Calster, B. (2022). Risk prediction models for discrete ordinal outcomes: Calibration and the impact of the proportional odds assumption. *Statistics in Medicine*, **41(8)**, pp. 1334– 1360

Qin, G., & Hotilovac, L. (2008). Comparison of non-parametric confidence intervals for the area under the ROC curve of a continuous-scale diagnostic test. *Statistical Methods in Medical Research*, **17(2)**, pp. 207-21

Steyerberg, E.W., Van Calster, B., Pencina, M.J. (2011). Performance measures for prediction models and markers : evaluation of predictions and classifications. *Revista Espanola de Cardiologia*, **64(9)**, pp. 788-794

Van Calster, B., Nieboer, D., Vergouwe, Y., De Cock, B., Pencina M., Steyerberg E.W. (2016). A calibration hierarchy for risk models was defined: from utopia to empirical data. *Journal of Clinical Epidemiology*, **74**, pp. 167-176

Van Hoorde, K., Van Huffel, S., Timmerman, D., Bourne, T., Van Calster, B. (2015). A spline-based tool to assess and visualize the calibration of multiclass risk predictions. *Journal of Biomedical Informatics*, **54**, pp. 283-93

Examples

```
# Load package
library(CalibrationCurves)
set.seed(1783)

# Simulate training data
X      = replicate(4, rnorm(5e2))
p0true = binomial()$linkinv(cbind(1, X) %*% c(0.1, 0.5, 1.2, -0.75, 0.8))
y      = rbinom(5e2, 1, p0true)
Df     = data.frame(y, X)

# Fit logistic model
FitLog = lrm(y ~ ., Df)

# Simulate validation data
Xval   = replicate(4, rnorm(5e2))
p0true = binomial()$linkinv(cbind(1, Xval) %*% c(0.1, 0.5, 1.2, -0.75, 0.8))
yval   = rbinom(5e2, 1, p0true)
Pred   = binomial()$linkinv(cbind(1, Xval) %*% coef(FitLog))

# Default calibration plot
valProbggplot(Pred, yval)

# Adding logistic calibration curves and other additional features
valProbggplot(Pred, yval, CL.smooth = TRUE, logistic.cal = TRUE, lty.log = 2,
 col.log = "red", lwd.log = 1.5)

valProbggplot(Pred, yval, CL.smooth = TRUE, logistic.cal = TRUE, lty.log = 9,
col.log = "red", lwd.log = 1.5, col.ideal = colors()[10], lwd.ideal = 0.5)
```

`valProbSurvival` *Plot a calibration curve for a Cox Proportional Hazards model*

Description

Plot a calibration curve for a Cox Proportional Hazards model

Usage

```
valProbSurvival(
  fit,
  valdata,
  alpha = 0.05,
  timeHorizon = 5,
  nk = 3,
  plotCal = c("none", "base", "ggplot"),
  addCox = FALSE,
  addRCS = TRUE,
  CL.cox = c("fill", "line"),
```

```
  CL.rcs = c("fill", "line"),
  xlab = "Predicted probability",
  ylab = "Observed proportion",
  xlim = c(-0.02, 1),
  ylim = c(-0.15, 1),
  lty.ideal = 1,
  col.ideal = "red",
  lwd.ideal = 1,
  lty.cox = 1,
  col.cox = "grey",
  lwd.cox = 1,
  fill.cox = "lightgrey",
  lty.rcs = 1,
  col.rcs = "black",
  lwd.rcs = 1,
  fill.rcs = rgb(177, 177, 177, 177, maxColorValue = 255),
  riskdist = "predicted",
  d0lab = "0",
  d1lab = "1",
  size.d01 = 5,
  dist.label = 0.01,
  line.bins = -0.05,
  dist.label2 = 0.04,
  length.seg = 0.85,
  legendloc = c(0.5, 0.27)
)
```

Arguments

`fit`	the model fit, has to be of type `coxph`
`valdata`	the validation data set
`alpha`	the significance level
`timeHorizon`	the time point at which the predictions have to be evaluated
`nk`	the number of knots, for the restricted cubic splines fit
`plotCal`	indicates if and how the calibration curve has to be plotted. `plotCal = "none"` plots no calibration curve, `plotCal = "base"` plots the calibration curve using base R (see `plot`) and `plotCal = "ggplot"` creates a plot using `ggplot`
`addCox`	logical, indicates if the Cox's estimated calibration curve has to be added to the plot
`addRCS`	logical, indicates if the restricted cubic splines' (RCS) estimated calibration curve has to be added to the plot
`CL.cox`	`"fill"` shows pointwise 95% confidence limits for the Cox calibration curve with a gray area between the lower and upper limits and `"line"` shows the confidence limits with a dotted line
`CL.rcs`	`"fill"` shows pointwise 95% confidence limits for the RCS calibration curve with a gray area between the lower and upper limits and `"line"` shows the confidence limits with a dotted line

`xlab`	x-axis label, default is `"Predicted Probability"`.
`ylab`	y-axis label, default is `"Observed proportion"`.
`xlim, ylim`	numeric vectors of length 2, giving the x and y coordinates ranges (see `plot.window`)
`lty.ideal`	linetype of the ideal line. Default is `1`.
`col.ideal`	controls the color of the ideal line on the plot. Default is `"red"`.
`lwd.ideal`	controls the line width of the ideal line on the plot. Default is `1`.
`lty.cox`	if `addCox = TRUE`, the linetype of the Cox calibration curve
`col.cox`	if `addCox = TRUE`, the color of the Cox calibration curve
`lwd.cox`	if `addCox = TRUE`, the linewidth of the Cox calibration curve
`fill.cox`	if `addCox = TRUE` and `CL.cox = "fill"`, the fill of the Cox calibration curve
`lty.rcs`	if `addRCS = TRUE`, the linetype of the RCS calibration curve
`col.rcs`	if `addRCS = TRUE`, the color of the RCS calibration curve
`lwd.rcs`	if `addRCS = TRUE`, the linewidth of the RCS calibration curve
`fill.rcs`	if `addRCS = TRUE` and `CL.rcs = "fill"`, the fill of the RCS calibration curve
`riskdist`	Use `"calibrated"` to plot the relative frequency distribution of calibrated probabilities after dividing into 101 bins from `lim[1]` to `lim[2]`. Set to `"predicted"` (the default as of rms 4.5-1) to use raw assigned risk, `FALSE` to omit risk distribution. Values are scaled so that highest bar is `0.15*(lim[2]-lim[1])`.
`d0lab, d1lab`	controls the labels for events and non-events (i.e. outcome y) for the histograms. Defaults are `d1lab="1"` for events and `d0lab="0"` for non-events.
`size.d01`	controls the size of the labels for events and non-events. Default is 5 and this value is multiplied by 0.25 when `plotCal = "base"`.
`dist.label`	controls the horizontal position of the labels for events and non-events. Default is 0.04.
`line.bins`	controls the horizontal (y-axis) position of the histograms. Default is -0.05.
`dist.label2`	controls the vertical distance between the labels for events and non-events. Default is 0.03.
`length.seg`	controls the length of the histogram lines. Default is `1`.
`legendloc`	if `pl=TRUE`, list with components `x,y` or vector `c(x,y)` for bottom right corner of legend for curves and points. Default is `c(.50, .27)` scaled to lim. Use `locator(1)` to use the mouse, `FALSE` to suppress legend.

Value

An object of type `SurvivalCalibrationCurves` with the following slots:

`call`	the matched call.
`stats`	a list containing performance measures of calibration.
`alpha`	the significance level used.
`Calibration`	contains the estimated calibration slope, together with their confidence intervals.
`CalibrationCurves`	The coordinates for plotting the calibration curves.

References

van Geloven N, Giardiello D, Bonneville E F, Teece L, Ramspek C L, van Smeden M et al. (2022). Validation of prediction models in the presence of competing risks: a guide through modern methods. *BMJ*, **377:e069249**, doi:10.1136/bmj-2021-069249

Examples

```
## Not run:
library(CalibrationCurves)
data(trainDataSurvival)
data(testDataSurvival)
sFit = coxph(Surv(ryear, rfs) ~ csize + cnode + grade3, data = trainDataSurvival,
 x = TRUE, y = TRUE)
calPerf = valProbSurvival(sFit, gbsg5, plotCal = "base", nk = 5)

## End(Not run)
```

`%<=%` *Infix operator to run background jobs*

Description

This infix operator can be used to create a background job in RStudio/Posit and, once completed, the value of rhs is assigned to lhs.

Usage

```
lhs %<=% rhs
```

Arguments

`lhs`	the object that the rhs value is assigned to
`rhs`	the value you want to assign to lhs

Value

prints the ID of the background job in the console and, once completed, the value of lhs is assigned to rhs

Examples

```
# Can only be executed in Rstudio
## Not run: x %<=% rnorm(1e7)
```

`%{}%` *Infix operator to run background jobs*

Description

This infix operator can be used to create a background job for a block of code in RStudio/Posit and, once completed, all objects created in the block of code are imported into the global environment.

Usage

```
lhs %{}% rhs
```

Arguments

`lhs`	not used, see details and examples
`rhs`	the block of code that you want to run

Details

You can use this infix operator in two different ways. Either you set the left-hand side to `NULL` or you use the syntax `` `%{}%` ({BlockOfCode}) ``

Value

prints the ID of the background job in the console and, once completed, the objects created in the block of code are imported into the global environment

Examples

```
# Can only be executed in Rstudio
## Not run:
NULL %{}% {
 x = rnorm(1e7)
 y = rnorm(1e7)
}
`%{}%` ({
 x = rnorm(1e7)
 y = rnorm(1e7)
})

## End(Not run)
```

Index