Package: CalibrationCurves (via r-universe)

November 2, 2026

Type Package

Title Calibration Performance

Version 3.1.0

Date 2026-03-25

Description Plots calibration curves and computes statistics for assessing calibration performance. See Lasai et al. (2025) <doi:10.48550/arXiv.2503.08389>, 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, pec, riskRegression, meta, metafor, zoo, lme4, merTools, dplyr, magrittr

Suggests knitr, rmarkdown, bookdown, rstudioapi, mgcv, MASS, Matrix, testthat (>= 3.0.0)

Config/testthat/edition 3

RoxygenNote 7.3.3

Encoding UTF-8

VignetteBuilder knitr

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

NeedsCompilation no

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

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

Config/pak/sysreqs cmake make libicu-dev libuv1-dev libxml2-dev libx11-dev

Repository https://bavodc.r-universe.dev

Date/Publication 2026-03-27 07:38:02 UTC

RemoteUrl https://github.com/cran/CalibrationCurves

RemoteRef HEAD

RemoteSha a602b8f6f48d1e3b654b8e8c5b6ebdd9fe586556

Contents

.rcspline.plot . . . 2
%<=% . . . 4
%{}% . . . 5
auc.nonpara.mw . . . 6
CalibrationCurves . . . 7
CGC . . . 9
genCalCurve . . . 11
LibraryM . . . 14
MAC2 . . . 14
MIXC . . . 16
print.CalibrationCurve . . . 18
print.ClusteredCalibrationCurve . . . 18
print.GeneralizedCalibrationCurve . . . 19
print.ggplotCalibrationCurve . . . 19
print.SurvivalCalibrationCurve . . . 20
simulatedclustereddata . . . 21
simulateddata . . . 22
simulatedpoissondata . . . 23
simulatedsurvivaldata . . . 24
val.prob.ci.2 . . . 26
valProbCluster . . . 32
valProbggplot . . . 35
valProbSurvival . . . 41

Index **44**

`.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

%<=% *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)
```

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

The **CalibrationCurves** package provides tools to assess and visualize the calibration performance of prediction models. Calibration refers to the agreement between predicted probabilities or values and what is actually observed.

The package covers a broad range of outcome types and modelling settings:

- **Binary outcomes** — `val.prob.ci.2` (base R graphics) and `valProbggplot` (ggplot2) compute flexible calibration curves (loess or restricted cubic splines) with pointwise 95% confidence intervals, logistic calibration slope and intercept, c-statistic, Brier score, and other statistics.
- **Clustered binary outcomes** — `valProbCluster` assesses calibration while accounting for clustering via three approaches: Clustered Grouped Calibration (`CGC`), Meta-Analytical Calibration Curve (`MAC2`), and Mixed-Effects Model Calibration (`MIXC`). See Barreñada et al. (2025).
- **Generalized outcomes (exponential family)** — `genCalCurve` extends the calibration framework to outcomes whose distribution belongs to the exponential family (e.g., Poisson, Gamma). It estimates the generalized calibration slope and intercept and plots the generalized calibration curve. See De Cock Campo (2023).
- **Survival outcomes** — `valProbSurvival` evaluates calibration for a fitted Cox proportional hazards model at a given time horizon, producing calibration curves and summary statistics for time-to-event predictions.

A vignette is available that provides a comprehensive overview of the theory and illustrates the functions with worked examples. Further background is available in the linked papers below.

Details

History

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 features:

- 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 using loess (default) or restricted cubic splines, with pointwise 95% confidence intervals.
- Loess: confidence intervals can be obtained in closed form or using bootstrapping (`CL.BT=TRUE` uses 2000 bootstrap samples).
- RCS: 3 to 5 knots; knot locations estimated via default quantiles of the predictor (by `rcspline.eval`).
- Plot customization through standard plot arguments (`cex.axis`, etc.); legend size controlled via `cex.leg`.
- Label y-axis: "Observed proportion".
- Added the Estimated Calibration Index (ECI) to quantify lack of calibration (Van Hoorde et al., 2015).
- By default shows the "abc" of model performance: calibration intercept, calibration slope, and c-statistic (Steyerberg et al., 2011).
- Vectors p, y and logit no longer have to be sorted.

A ggplot2-based equivalent, `valProbggplot`, was subsequently added, offering the same functionality with ggplot2 graphics.

In 2023, Bavo De Cock (Campo) introduced the generalized calibration framework (De Cock Campo, 2023), extending logistic calibration to prediction models with outcomes from any distribution in the exponential family, implemented in `genCalCurve`.

Support for **survival models** was added via `valProbSurvival`, enabling calibration assessment of Cox proportional hazards model predictions at a specified time horizon.

In 2025, methods for **clustered data** were introduced (Barreñada et al., 2025), accessible through `valProbCluster`, which supports CGC, MAC2, and MIXC approaches.

The most current version of this package can be found on https://github.com/BavoDC/CalibrationCurves.

References

Barreñada, L., De Cock Campo, B., Wynants, L., Van Calster, B. (2025). Clustered Flexible Calibration Plots for Binary Outcomes Using Random Effects Modeling. arXiv:2503.08389, available at https://arxiv.org/abs/2503.08389.

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

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

`CGC` *Internal function for the Clustered Grouped Calibration Curve (CGC)*

Description

Estimates the calibration curves using the CGC approach. The function supports two grouping methods: equal-sized groups (`"grouped"`) or interval-based groups (`"interval"`). Optionally, a calibration plot can be produced with cluster-specific curves.

Usage

```
CGC(
  data = NULL,
  p,
  y,
  cluster,
  cl.level = 0.95,
  ntiles = 10,
  cluster_curves = FALSE,
  plot = TRUE,
  size = 1,
  linewidth = 0.4,
  univariate = FALSE,
  method = c("grouped", "interval")
)
```

Arguments

data	optional data frame containing the variables p, y, and cluster. If supplied, variable names should be given without quotation marks.
p	predicted probabilities (numeric vector) or name of the column in data.
y	binary outcome variable or the name of the column in data.
cluster	cluster identifier (factor, character, or integer) or name of the column in data.
cl.level	the confidence level for the calculation of the confidence interval. Default is 0.95.
ntiles	integer, number of groups (tiles) for calibration. Default is 10.
cluster_curves	logical, whether to include cluster-specific calibration curves in the plot. Default is FALSE.
plot	logical, whether to return a calibration plot. Default is TRUE.
size	numeric, point size for plotted curves. Default is 1.
linewidth	numeric, line width for plotted curves. Default is 0.4.
univariate	logical, whether to use univariate meta-analysis. Default is FALSE.
method	character, grouping method: "grouped" (equal-sized groups) or "interval" (interval-based). Default is "grouped".

Details

When method = "grouped", the predictions are divided into equal-sized bins using quantiles. Conversely, if method ="interval", the predictions are divided into fixed-width bins across [0, 1].

The function performs a meta-analysis within each group. This can be either a univariate or bivariate analysis, which is specified in the univariate argument. The univariate analysis is performed using the metaprop function and the bivariate analysis employs the rma.mv function. Hereafter, the results are aggregated and plotted as calibration curves.

Value

A list containing:

plot_data Data frame of meta-analysis calibration estimates.

trad_grouped Data frame with traditional grouped calibration results.

observed_data Data frame with per-observation calibration data.

cluster_data Data frame with cluster-specific calibration summaries.

plot A ggplot2 object if plot = TRUE, otherwise NULL.

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 = TRUE,
  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)
```

`MAC2` *Internal function for the Meta-Analytical Calibration Curve (MAC2)*

Description

Computes meta-analytical calibration curves using multiple methods (logistic regression, loess or splines) and performs meta-analysis across clusters to generate aggregated calibration curves with confidence and prediction intervals.

Usage

```
MAC2(
  data = NULL,
  p,
  y,
  cluster,
  grid,
  cl.level = 0.95,
  alpha.lr = 0.05/3,
  plot = TRUE,
  cluster_curves = FALSE,
  knots = 3,
  transf = "logit",
  method_choice = c("splines", "log", "loess"),
  method.tau = "REML",
  prediction = TRUE,
  random = TRUE,
  sm = "PLOGIT",
  hakn = FALSE,
  linewidth = 1,
  method.predict = "HTS",
  verbose = FALSE
)
```

Arguments

`data`	optional data frame containing the variables `p`, `y`, and `cluster`. If supplied, variable names should be given without quotation marks.
`p`	predicted probabilities (numeric vector) or name of the column in `data`.
`y`	binary outcome variable or the name of the column in `data`.
`cluster`	Cluster identifier (factor, character, or integer) or name of the column in `data`.
`grid`	the grid for the calibration curve evaluation
`cl.level`	the confidence level for the calculation of the confidence interval. Default is `0.95`.
`alpha.lr`	the alpha-level used for the likelihood ratio test, selecting the number of knots for the restricted cubic splines
`plot`	logical, indicates whether to plot the calibration curves. Default is `TRUE`.
`cluster_curves`	logical, whether to include cluster-specific curves in the plot. Default is `FALSE`.
`knots`	integer, number of knots for splines. Default is `3`.
`transf`	character, transformation for predictions: `"logit"` or `"identity"`. Default is `"logit"`.
`method_choice`	character, which method to use for meta-analysis. Options are: `"log"`, `"loess"` or `"splines"`. Default is `"splines"`.
`method.tau`	character, method for between-study heterogeneity estimation. Default is `"REML"`. This argument is passed to the `metagen` function.

prediction	logical, whether to compute prediction intervals. Default is TRUE. This argument is passed to the prediction argument of the metagen function.
random	logical, whether to use random-effects model. Default is TRUE. This argument is passed to the random argument of the metagen function.
sm	character, summary measure for meta-analysis. Default is "PLOGIT". This argument is passed to the sm argument of the metagen function.
hakn	logical, whether to use Hartung-Knapp adjustment. Default is FALSE. This argument is passed to the method.random.ci argument of the metagen function.
linewidth	numeric, line width for the meta-curve. Default is 1.
method.predict	character, method for prediction intervals. Default is "HTS". This argument is passed to the method.predict argument of the metagen function.
verbose	logical, indicates whether progress has to be printed in the console.

Details

This function estimates the center-specific calibration curves using logistic regression, loess or splines. Hereafter, it aggregates the calibration curves using meta-analytical techniques. The meta-analysis is performed using the function metagen from the meta package. The method_choice argument determines which method is for the meta-analytical aggregation.

Value

A list containing:

cluster_data Data frame with linear predictors and standard errors for each method per cluster

plot_data Data frame with meta-analysis results including predictions and intervals

plot A ggplot2 object if plot = TRUE, otherwise NULL

MIXC *Internal function for the Mixed-Effects Model Calibration Curve (MIXC)*

Description

Estimates the calibration curve using a logistic generalized linear mixed model.

Usage

```
MIXC(
  data = NULL,
  p,
  y,
  cluster,
  grid,
  method = c("slope", "intercept"),
```

```
  plot = TRUE,
  cluster_curves = FALSE,
  nsims_pi = 10000,
  CI = TRUE,
  CI_method = c("naive", "delta"),
  cl.level = 0.95
)
```

Arguments

data	optional data frame containing the variables p, y, and cluster. If supplied, variable names should be given without quotation marks.
p	predicted probabilities (numeric vector) or name of the column in data.
y	binary outcome variable or the name of the column in data.
cluster	Cluster identifier (factor, character, or integer) or name of the column in data.
grid	the grid for the calibration curve evaluation
method	character, type of mixed-effects model: "intercept" (random intercept) or "slope" (random slope). Default is "slope".
plot	logical, indicating whether to generate a calibration plot. Default is TRUE.
cluster_curves	logical, whether to include cluster-specific curves in the plot. Default is FALSE.
nsims_pi	integer, number of simulations for prediction intervals. Default is 10000.
CI	logical, whether to calculate confidence intervals. Default is TRUE.
CI_method	character, method for computing the confidence intervals of the observed proportions. If "delta", the delta method is applied. Conversely, when CI_method == "naive", no correction is applied. Default is "naive".
cl.level	the confidence level for the calculation of the confidence interval. Default is 0.95.

Details

This function estimates the calibration curves using a logistic generalized linear mixed model.

Value

A list containing:

model The fitted mixed-effects model object

cluster_data Data frame with calibration data for each cluster

plot_data Data frame with calibration data for the average cluster

observed_data Data frame with calibration data for individual observations

plot A ggplot2 object if plot = TRUE, otherwise NULL

`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.ClusteredCalibrationCurve` *Print function for a ClusteredCalibrationCurve object*

Description

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

Usage

```
## S3 method for class 'ClusteredCalibrationCurve'
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.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

simulatedclustereddata *Simulated data sets to illustrate the package functionality*

Description

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

Usage

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

Format

y the binary outcome variable

cluster the cluster

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
lapply(c("magrittr", "dplyr"), library, character.only = TRUE)
set.seed(1234)

# Simulate training data
nClusters = 10
p         = 5
Uj        = scale(rnorm(nClusters))
nPop      = 1e6
nSample   = 1e3
nTest     = 1e3
X         = replicate(p, rnorm(nPop))
Beta      = rnorm(p)
cluster   = sample(seq_len(nClusters), nPop, TRUE)
```

```
table(cluster)
eta       = X %*% Beta + Uj[match(cluster, seq_len(nClusters))]
y         = rbinom(nPop, 1, binomial()$linkinv(eta))
Dt        = data.frame(y, X, cluster)
colnames(Dt) %<>% tolower

clustertraindata = Dt %>%
  filter(cluster %in% 1:5) %>%
  group_by(cluster) %>%
  sample_n(size = nSample) %>%
  as.data.frame
clustertestdata = Dt %>%
  filter(cluster %in% 6:10) %>%
  group_by(cluster) %>%
  sample_n(size = nTest) %>%
  as.data.frame
```

`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 = glm(y ~ ., Df, family = binomial)

# 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)
```

`valProbCluster` *Calibration performance with cluster adjustment (ggplot version)*

Description

This function evaluates the calibration performance of a model's predicted probabilities whilst accounting for clustering. The function supports multiple approaches (‘"default"‘, ‘"CGC"‘, ‘"MAC2"‘, ‘"MIXC"‘) and returns the results as well as a ‘ggplot‘ object.

Usage

```
valProbCluster(
  data = NULL,
  p,
  y,
  cluster,
  plot = TRUE,
  approach = c("default", "MIXC", "CGC", "MAC2"),
  cl.level = 0.95,
  xlab = "Predicted probability",
  ylab = "Observed proportion",
  grid_l = 100,
  rangeGrid = range(p),
  ...
)
```

Arguments

`data`	optional, a data frame containing the variables `p`, `y`, and `cluster`. If supplied, variable names should be given without quotation marks.
`p`	predicted probabilities (numeric vector) or name of the column in `data`
`y`	binary outcome variable or the name of the column in `data`
`cluster`	cluster identifier (factor, character, or integer) or name of the column in `data`
`plot`	logical, indicates whether a plot needs to be produced. If `TRUE`, a plot will be constructed by the chosen subfunction.
`approach`	character string specifying which calibration method to use. Must be one of the following: • `"default"`: Combines MAC2 (splines) for the overall calibration curve, CI and PI bands, with MIXC for the cluster-specific curves. This is the recommended approach; • `"CGC"`: Clustered Grouped Calibration; • `"MAC2"`: Meta-Analytical Calibration Curve; • `"MIXC"`: Mixed-Effects Model Calibration. Defaults to `"default"`.
`cl.level`	the confidence level for the calculation of the confidence intervals. Default is `0.95`.
`xlab`	label for the x-axis of the plot (default is `"Predicted probability"`).
`ylab`	label for the y-axis of the plot (default is `"Observed proportion"`).
`grid_l`	integer. Number of points in the probability grid for plotting (default is `100`).
`rangeGrid`	the range of the grid. Default is `range(p)`.
`...`	additional arguments to be passed to the selected subfunction (`CGC`, `MAC2` and `MIXC`).

Details

The function internally calls one of the following subfunctions:

- `CGC(p, y, cluster, plot, ...)`
- `MAC2(p, y, cluster, plot, grid, ...)`
- `MIXC(p, y, cluster, plot, CI, grid, ...)`

Extra arguments supplied via the ellipsis argument `...` are passed directly to the chosen subfunction. Please check the additional documentation of `CGC`, `MAC2` and `MIXC` for detailed information on the arguments.

Value

An object of class `"ClusteredCalibrationCurve"` containing:

- `call`: the matched call.
- `approach`: the chosen approach.
- `cl.level`: the confidence level used.
- `grid`: probability grid used for plotting.
- `ggPlot`: a `ggplot` object if `plot = TRUE`, otherwise `NULL`.
- `results`: results from the chosen subfunction. For `approach = "default"`, this is a list with two elements: `overall` (MAC2 results with the overall curve data) and `clusters` (MIXC results with cluster-specific data).

References

Barreñada, L., De Cock Campo, B., Wynants, L., Van Calster, B. (2025). Clustered Flexible Calibration Plots for Binary Outcomes Using Random Effects Modeling. Research Synthesis Methods. Published online 2025:1-22. doi:10.1017/rsm.2025.10046

See Also

`CGC`, `MAC2` and `MIXC`

Examples

```
library(lme4)
data("clustertraindata")
data("clustertestdata")
mFit = glmer(y ~ x1 + x2 + x3 + x5 + (1 | cluster),
             data = clustertraindata, family = "binomial")
preds          = predict(mFit, clustertestdata, type = "response", re.form = NA)
y              = clustertestdata$y
cluster        = clustertestdata$cluster
valClusterData = data.frame(y = y, preds = preds, center = cluster)

# Assess calibration performance
Results  = valProbCluster(
p = valClusterData$preds, y = valClusterData$y, cluster = valClusterData$center,
```

```
plot = TRUE,
approach = "MIXC", method = "slope", grid_l = 100
)
Results
```

---

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 = glm(y ~ ., Df, family = binomial)

# 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,
  weights = NULL,
  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
weights	vector of case weights
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)
library(survival)
data(trainDataSurvival)
data(testDataSurvival)
sFit = coxph(Surv(ryear, rfs) ~ csize + cnode + grade3, data = trainDataSurvival,
 x = TRUE, y = TRUE)
calPerf = valProbSurvival(sFit, testDataSurvival, plotCal = "base", nk = 5)

## End(Not run)
```

Index

∗ **datasets**
  simulatedclustereddata, 21
  simulateddata, 22
  simulatedpoissondata, 23
  simulatedsurvivaldata, 24
∗ **htest**
  auc.nonpara.mw, 6
.rcspline.plot, 2
%<=%, 4
%{}%, 5

auc.nonpara.mw, 6, *29*, *38*

CalibrationCurves, 7
CalibrationCurves-package *(*CalibrationCurves*)*, 7
CGC, *7*, 9, *33, 34*
clustertestdata *(*simulatedclustereddata*)*, 21
clustertraindata *(*simulatedclustereddata*)*, 21
coxph, *42*
coxph.fit, *4*
cph, *4*

genCalCurve, *7, 8*, 11, *19*
ggplot, *42*

legend, *30*
LibraryM, 14
loess, *12*, *28*, *31*, *36*, *39*
lrm, *4*
lrm.fit, *4*

MAC2, *7*, 14, *33, 34*
meta, *16*
metagen, *15, 16*
metaprop, *10*
MIXC, *7*, 16, *33, 34*

par, *13*, *31*
plot, *4*, *13*, *31*, *42*
plot.window, *12*, *29*, *42*
poissontestdata *(*simulatedpoissondata*)*, 23
poissontraindata *(*simulatedpoissondata*)*, 23
print, *18–20*
print.CalibrationCurve, 18
print.ClusteredCalibrationCurve, 18
print.GeneralizedCalibrationCurve, 19
print.ggplotCalibrationCurve, 19
print.SurvivalCalibrationCurve, 20

rcs, *28*, *36*
rcspline.eval, *3, 4*, *8*
rcspline.plot, *2*, *28*, *36*
rma.mv, *10*

simulatedclustereddata, 21
simulateddata, 22
simulatedpoissondata, 23
simulatedsurvivaldata, 24
supsmu, *4*

testdata *(*simulateddata*)*, 22
testDataSurvival *(*simulatedsurvivaldata*)*, 24
traindata *(*simulateddata*)*, 22
trainDataSurvival *(*simulatedsurvivaldata*)*, 24

val.prob, *8*, *26*, *29*, *35*, *38*
val.prob.ci.2, *7, 8*, *18*, 26
valProbCluster, *7, 8*, 32
valProbggplot, *7, 8*, *18–20*, 35
valProbSurvival, *7, 8*, *20*, 41

xlim, *37*

ylim, *37*