Package 'pminternal'

July 23, 2025

Title Internal Validation of Clinical Prediction Models

Version 0.1.0

Maintainer Stephen Rhodes <steverho89@gmail.com>

Description Conduct internal validation of a clinical prediction model for a binary outcome. Produce bias corrected performance metrics (c-statistic, Brier score, calibration intercept/slope) via bootstrap (simple bootstrap, bootstrap optimism, .632 optimism) and cross-validation (CV optimism, CV average). Also includes functions to assess model stability via bootstrap resampling. See Steyerberg et al. (2001) <doi:10.1016/s0895-4356(01)00341-9>; Harrell (2015) <doi:10.1007/978-3-319-19425-7>; Riley and Collins (2023) <doi:10.1002/bimj.202200302>.

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.2

URL https://github.com/stephenrho/pminternal, https://stephenrho.github.io/pminternal/

BugReports https://github.com/stephenrho/pminternal/issues

Imports dcurves, graphics, grDevices, insight, marginaleffects, methods, pmcalibration, pROC, stats, parallel, pbapply, purrr

Suggests ggplot2, glmnet, Hmisc, knitr, rmarkdown, ranger, gbm, rms, mgcv, mice

VignetteBuilder knitr

Config/Needs/website rmarkdown

NeedsCompilation no

Author Stephen Rhodes [aut, cre, cph]

Repository CRAN

Date/Publication 2025-03-18 08:50:02 UTC

Contents

`boot_optimism` *Calculate optimism and bias-corrected scores via bootstrap resampling*

Description

Estimate bias-corrected scores via calculation of bootstrap optimism (standard or .632). Can also produce estimates for assessing the stability of prediction model predictions. This function is called by `validate`.

Usage

```
boot_optimism(
  data,
  outcome,
  model_fun,
  pred_fun,
  score_fun,
  method = c("boot", ".632"),
  B = 200,
  ...
)
```

Arguments

data	the data used in developing the model. Should contain all variables considered (i.e., even those excluded by variable selection in the development sample)
outcome	character denoting the column name of the outcome in data.
model_fun	a function that takes at least one argument, data. This function should implement the entire model development procedure (i.e., hyperparameter tuning, variable selection, imputation). Additional arguments can be provided via .... This function should return an object that works with pred_fun.
pred_fun	function that takes at least two arguments, model and data. This function should return a numeric vector of predicted probabilities of the outcome with the same length as the number of rows in data so it is important to take into account how missing data is treated (e.g., predict.glm omits predictions for rows with missing values). see vignette("missing-data", package = "pminternal").
score_fun	a function to calculate the metrics of interest. If this is not specified score_binary is used.
method	"boot" or ".632". The former estimates bootstrap optimism for each score and subtracts from apparent scores (simple bootstrap estimates are also produced as a by-product). The latter estimates ".632" optimism as described in Harrell (2015). See validate details.
B	number of bootstrap resamples to run
...	additional arguments for model_fun, pred_fun, and/or score_fun.

Value

a list of class internal_boot containing:

- apparent - scores calculated on the original data using the original model.
- optimism - estimates of optimism for each score (average difference in score for bootstrap models evaluated on bootstrap vs original sample) which can be subtracted from 'apparent' performance calculated using the original model on the original data.
- corrected - 'bias corrected' scores (apparent - optimism)
- simple - if method = "boot", estimates of scores derived from the 'simple bootstrap'. This is the average of each score calculated from the bootstrap models evaluated on the original outcome data. NULL if method = ".632"
- stability - if method = "boot", a N,(B+1) matrix where N is the number of observations in data and B is the number of bootstrap samples. The first column contains the original predictions and each of subsequent B columns contain the predicted probabilities of the outcome from each bootstrap model evaluated on the original data. There may be fewer than B+1 columns if errors occur during resamples (when model_fun throws an error all scores are NA). NULL if method = ".632"

References

Steyerberg, E. W., Harrell Jr, F. E., Borsboom, G. J., Eijkemans, M. J. C., Vergouwe, Y., & Habbema, J. D. F. (2001). Internal validation of predictive models: efficiency of some procedures for logistic regression analysis. Journal of clinical epidemiology, 54(8), 774-781.

Harrell Jr F. E. (2015). Regression Modeling Strategies: with applications to linear models, logistic and ordinal regression, and survival analysis. New York: Springer Science, LLC.

Examples

```
library(pminternal)
set.seed(456)
# simulate data with two predictors that interact
dat <- pmcalibration::sim_dat(N = 1000, a1 = -2, a3 = -.3)
mean(dat$y)
dat$LP <- NULL # remove linear predictor

# fit a (misspecified) logistic regression model
model_fun <- function(data, ...){
  glm(y ~ x1 + x2, data=data, family="binomial")
}

pred_fun <- function(model, data, ...){
  predict(model, newdata=data, type="response")
}

boot_optimism(data=dat, outcome="y", model_fun=model_fun, pred_fun=pred_fun,
              method="boot", B=20) # B set to 20 for example but should be >= 200
```

---

calibration_stability *Plot calibration stability across bootstrap replicates*

Description

A calibration (in)stability plot shows calibration curves for bootstrap models evaluated on original outcome. A stable model should produce boot calibration curves that differ minimally from the 'apparent' curve.

Usage

```
calibration_stability(
  x,
  calib_args,
  xlim,
  ylim,
  xlab,
  ylab,
  col,
  subset,
  plot = TRUE
)
```

Arguments

x	an object produced by validate with method = "boot_*" (or boot_optimism with method="boot")
calib_args	settings for calibration curve (see pmcalibration::pmcalibration). If unspecified settings are given by cal_defaults with 'eval' set to 100 (evaluate each curve at 100 points between min and max prediction).
xlim	x limits (default = c(0,1))
ylim	y limits (default = c(0,1))
xlab	a title for the x axis
ylab	a title for the y axis
col	color of lines for bootstrap models (default = grDevices::grey(.5, .3))
subset	vector of observations to include (row indices). If dataset is large fitting B curves is demanding. This can be used to select a random subset of observations.
plot	if FALSE just returns curves (see value)

Value

plots calibration (in)stability. Invisibly returns a list containing data for each curve (p=x-axis, pc=y-axis). The first element of this list is the apparent curve (original model on original outcome).

References

Riley, R. D., & Collins, G. S. (2023). Stability of clinical prediction models developed using statistical or machine learning methods. Biometrical Journal, 65(8), 2200302. doi:10.1002/bimj.202200302

Examples

```
set.seed(456)
# simulate data with two predictors that interact
dat <- pmcalibration::sim_dat(N = 2000, a1 = -2, a3 = -.3)
mean(dat$y)
dat$LP <- NULL # remove linear predictor

# fit a (misspecified) logistic regression model
m1 <- glm(y ~ ., data=dat, family="binomial")

# internal validation of m1 via bootstrap optimism with 10 resamples
# B = 10 for example but should be >= 200 in practice
m1_iv <- validate(m1, method="boot_optimism", B=10)

calibration_stability(m1_iv)
```

`cal_plot` *Plot apparent and bias-corrected calibration curves*

Description

Plot apparent and bias-corrected calibration curves

Usage

```
cal_plot(
  x,
  xlim,
  ylim,
  xlab,
  ylab,
  app_col,
  bc_col,
  app_lty,
  bc_lty,
  plotci = c("if", "yes", "no")
)
```

Arguments

`x`	an object returned from `validate`. Original call should have specified 'eval' argument. See `score_binary`.
`xlim`	x limits (default = c(0, max of either curve))
`ylim`	y limits (default = c(0, max of either curve))
`xlab`	a title for the x axis
`ylab`	a title for the y axis
`app_col`	color of the apparent calibration curve (default = 'black')
`bc_col`	color of the bias-corrected calibration curve (default = 'black')
`app_lty`	line type of the apparent calibration curve (default = 1)
`bc_lty`	line type of the bias-corrected calibration curve (default = 2)
`plotci`	plot confidence intervals ('yes') or not ('no'). If 'yes' x should have confidence intervals added by `confint.internal_validate`. 'if' (default) plots CIs if they are available.

Value

plots apparent and bias-corrected curves. Silently returns a data.frame that can be used to produce a more 'publication ready' plot. Columns are as follows: predicted = values for the x-axis, apparent = value of apparent curve, bias_corrected = value of bias-corrected curve. Confidence intervals are included if available.

Examples

```
library(pminternal)
set.seed(456)
# simulate data with two predictors that interact
dat <- pmcalibration::sim_dat(N = 2000, a1 = -2, a3 = -.3)
mean(dat$y)
dat$LP <- NULL # remove linear predictor

# fit a (misspecified) logistic regression model
m1 <- glm(y ~ x1 + x2, data=dat, family="binomial")

# to get a plot of bias-corrected calibration we need
# to specify 'eval' argument via 'calib_args'
# this argument specifies at what points to evalulate the
# calibration curve for plotting. The example below uses
# 100 equally spaced points between the min and max
# original prediction.

p <- predict(m1, type="response")
p100 <- seq(min(p), max(p), length.out=100)

m1_iv <- validate(m1, method="cv_optimism", B=10,
                  calib_args = list(eval=p100))
# calib_ags can be used to set other calibration curve
# settings: see pmcalibration::pmcalibration

cal_plot(m1_iv)
```

---

classification_stability

Classification instability plot

Description

Classification instability plot shows the relationship between original model estimated risk and the classification instability index (CII). The CII is the proportion of bootstrap replicates where the predicted class (0 if p <= threshold; 1 if p > threshold) is different to that obtained from the original model. Those with risk predictions around the threshold will exhibit elevated CII but an unstable model will exhibit high CII across a range of risk predictions. See Riley and Collins (2023).

Usage

```
classification_stability(
  x,
  threshold,
  xlim,
  ylim,
  xlab,
```

```
  ylab,
  pch,
  cex,
  col,
  subset,
  plot = TRUE
)
```

Arguments

x	an object produced by validate with method = "boot_*" (or boot_optimism with method="boot")
threshold	estimated risks above the threshold get a predicted 'class' of 1, otherwise 0.
xlim	x limits (default = range of estimated risks)
ylim	y limits (default = c(0, maximum CII))
xlab	a title for the x axis
ylab	a title for the y axis
pch	plotting character (default = 16)
cex	controls point size (default = 1)
col	color of points (default = grDevices::grey(.5, .5))
subset	vector of observations to include (row indices). This can be used to select a random subset of observations.
plot	if FALSE just returns CII values (see value)

Value

plots classification (in)stability. Invisibly returns estimates of CII for each observation.

References

Riley, R. D., & Collins, G. S. (2023). Stability of clinical prediction models developed using statistical or machine learning methods. Biometrical Journal, 65(8), 2200302. doi:10.1002/bimj.202200302

Examples

```
set.seed(456)
# simulate data with two predictors that interact
dat <- pmcalibration::sim_dat(N = 2000, a1 = -2, a3 = -.3)
mean(dat$y)
dat$LP <- NULL # remove linear predictor

# fit a (misspecified) logistic regression model
m1 <- glm(y ~ ., data=dat, family="binomial")

# internal validation of m1 via bootstrap optimism with 10 resamples
# B = 10 for example but should be >= 200 in practice
m1_iv <- validate(m1, method="boot_optimism", B=10)
```

```
classification_stability(m1_iv, threshold=.2)
```

---

confint.internal_validate *Confidence intervals for bias-corrected performance measures*

Description

Implements the methods discussed in Noma et al. (2021), plus some others that have not been tested. Specifically, Noma et al. discuss bootstrap optimism correction ("boot_optimism" and ".632") and the percentile bootstrap (`ci_type = "perc"`). Their paper contains some simulation results on coverage properties of these CIs. If you used `validate` to do something other than bootstrap optimism correction or if you request normal approximation CIs please note that these approaches have (to my knowledge) not been thoroughly tested. `ci_type = "norm"` is included as it might be able to reduce the number of runs needed for "twostage" CIs. See details for the difference between "shifted" and "twostage". "norm" CIs are likely to perform poorly for some performance measures, such as calibration Intercept and Slope, which for regular glms are always 0 and 1, respectively, on assessment of apparent performance. As "shifted" CIs are based on apparent performance they will be meaningless for these measures. Use the untested methods with caution!

Usage

```
## S3 method for class 'internal_validate'
confint(
  object,
  parm,
  level = 0.95,
  method = c("shifted", "twostage"),
  ci_type = c("perc", "norm"),
  R = 1000,
  add = TRUE,
  ...
)
```

Arguments

`object`	created by call to `validate`
`parm`	a specification of which performance measures are to be given confidence intervals, either a vector of numbers or a vector of names. If missing, all scores are considered.
`level`	the confidence level required
`method`	"shifted" or "twostage" (see details)
`ci_type`	percentile ("perc") or normal approximation ("norm") bootstrap CIs
`R`	number of replicates

add	return the object with an additional slot containing CIs (default) or just return the CIs
...	additional arguments (currently ignored)

Details

The two methods are as follows (see Noma et al. (2021) for more details):

shifted (default) This approach is based on shifting bootstrap CIs for apparent performance by optimism. This makes it the faster option as only the calculation of apparent performance is needed for each replicate. If the CI for apparent performance is [lower, upper], the resulting CI for bias-corrected performance is [lower - optimism, upper - optimism]. Note this method is only available when using an optimism based approach (and "cv_optimism" was untested in Noma et al).

twostage This approach creates a bootstrap resample of the data and runs the entire validation procedure on the resample (with the same number of 'inner' replicates, determined by B in the original validate call). The CI is then constructed using the corrected estimates from the R 'outer' replicates. As this involves R*B replicates, this could take a long time. Note `validate` takes a `cores` argument that can allow the inner samples to run in parallel.

Value

A list with two elements, each a matrix with columns giving lower and upper confidence limits for each measure. One for apparent and one for bias-corrected measures. Columns will be labelled as (1-level)/2 and 1 - (1-level)/2 in % (by default 2.5% and 97.5%).

References

Noma, H., Shinozaki, T., Iba, K., Teramukai, S., & Furukawa, T. A. (2021). Confidence intervals of prediction accuracy measures for multivariable prediction models based on the bootstrap-based optimism correction methods. Statistics in Medicine, 40(26), 5691-5701.

`crossval` *Calculate bias-corrected scores via cross-validation*

Description

Estimate bias-corrected scores via cross-validation. CV is used to calculate optimism which is then subtracted from apparent scores and to calculate average performance in the out of sample (held out) data. This function is called by `validate`.

Usage

```
crossval(data, outcome, model_fun, pred_fun, score_fun, k = 10, ...)
```

Arguments

data	the data used in developing the model. Should contain all variables considered (i.e., even those excluded by variable selection in the development sample)
outcome	character denoting the column name of the outcome in data.
model_fun	a function that takes at least one argument, data. This function should implement the entire model development procedure (i.e., hyperparameter tuning, variable selection, imputation). Additional arguments can be provided via .... This function should return an object that works with pred_fun.
pred_fun	function that takes at least two arguments, model and data. This function should return a numeric vector of predicted probabilities of the outcome with the same length as the number of rows in data so it is important to take into account how missing data is treated (e.g., predict.glm omits predictions for rows with missing values).
score_fun	a function to calculate the metrics of interest. If this is not specified score_binary is used.
k	number of folds. Typically scores need greater than 2 observations to be calculated so folds should be chosen with this in mind.
...	additional arguments for model_fun, pred_fun, and/or score_fun.

Value

a list of class internal_cv containing:

- apparent - scores calculated on the original data using the original model.
- optimism - estimates of optimism for each score (average difference in score for training data vs test data on each fold) which can be subtracted from 'apparent' performance calculated using the original model on the original data.
- cv_optimism_corrected - 'bias corrected' scores (apparent - optimism). This is what is produced by rms::validate, rms::predab.resample.
- cv_average - average of scores calculated on the test (held out) data. This is the metric described in Steyerberg et al. (2001).
- indices - indices used to define test set on each fold.

References

Steyerberg, E. W., Harrell Jr, F. E., Borsboom, G. J., Eijkemans, M. J. C., Vergouwe, Y., & Habbema, J. D. F. (2001). Internal validation of predictive models: efficiency of some procedures for logistic regression analysis. Journal of clinical epidemiology, 54(8), 774-781.

Examples

```
library(pminternal)
set.seed(456)
# simulate data with two predictors that interact
dat <- pmcalibration::sim_dat(N = 1000, a1 = -2, a3 = -.3)
mean(dat$y)
```

```
dat$LP <- NULL # remove linear predictor

# fit a (misspecified) logistic regression model
#m1 <- glm(y ~ x1 + x2, data=dat, family="binomial")

model_fun <- function(data, ...){
  glm(y ~ x1 + x2, data=data, family="binomial")
}

pred_fun <- function(model, data, ...){
  predict(model, newdata=data, type="response")
}

# CV Corrected = Apparent - CV Optimism
# CV Average = average score in held out fold
crossval(data=dat, outcome="y", model_fun=model_fun, pred_fun=pred_fun, k=10)
```

`dcurve_stability` *Plot decision curve stability across bootstrap replicates*

Description

A decision curve (in)stability plot shows decision curves for bootstrap models evaluated on original outcome. A stable model should produce curves that differ minimally from the 'apparent' curve. See Riley and Collins (2023).

Usage

```
dcurve_stability(
  x,
  thresholds = seq(0, 0.99, by = 0.01),
  xlim,
  ylim,
  xlab,
  ylab,
  col,
  subset,
  plot = TRUE
)
```

Arguments

x	an object produced by validate with method = "boot_*" (or boot_optimism with method="boot")
thresholds	points at which to evaluate the decision curves (see dcurves::dca)
xlim	x limits (default = range of thresholds)
ylim	y limits (default = range of net benefit)

xlab	a title for the x axis
ylab	a title for the y axis
col	color of points (default = grDevices::grey(.5, .5))
subset	vector of observations to include (row indices). This can be used to select a random subset of observations.
plot	if FALSE just returns curves (see value)

Value

plots decision curve (in)stability. Invisibly returns a list containing data for each curve. These are returned from dcurves::dca. The first element of this list is the apparent curve (original model on original outcome).

References

Riley, R. D., & Collins, G. S. (2023). Stability of clinical prediction models developed using statistical or machine learning methods. Biometrical Journal, 65(8), 2200302. doi:10.1002/bimj.202200302

Examples

```
set.seed(456)
# simulate data with two predictors that interact
dat <- pmcalibration::sim_dat(N = 2000, a1 = -2, a3 = -.3)
mean(dat$y)
dat$LP <- NULL # remove linear predictor

# fit a (misspecified) logistic regression model
m1 <- glm(y ~ ., data=dat, family="binomial")

# internal validation of m1 via bootstrap optimism with 10 resamples
# B = 10 for example but should be >= 200 in practice
m1_iv <- validate(m1, method="boot_optimism", B=10)

dcurve_stability(m1_iv)
```

---

mape_stability *Mean absolute predictor error (MAPE) stability plot*

Description

A MAPE (in)stability plot shows mean absolute predictor error (average absolute difference between original estimated risk and risk from B bootstrap models) as a function of apparent estimated risk (prediction from original/development model). See Riley and Collins (2023).

Usage

```
mape_stability(x, xlim, ylim, xlab, ylab, pch, cex, col, subset, plot = TRUE)
```

Arguments

x	an object produced by validate with method = "boot_*" (or boot_optimism with method="boot")
xlim	x limits (default = range of estimated risks)
ylim	y limits (default = c(0, maximum mape))
xlab	a title for the x axis
ylab	a title for the y axis
pch	plotting character (default = 16)
cex	controls point size (default = 1)
col	color of points (default = grDevices::grey(.5, .5))
subset	vector of observations to include (row indices). This can be used to select a random subset of observations.
plot	if FALSE just returns MAPE values (see value)

Value

plots calibration (in)stability. Invisibly returns a list containing individual and average MAPE.

References

Riley, R. D., & Collins, G. S. (2023). Stability of clinical prediction models developed using statistical or machine learning methods. Biometrical Journal, 65(8), 2200302. doi:10.1002/bimj.202200302

Examples

```
set.seed(456)
# simulate data with two predictors that interact
dat <- pmcalibration::sim_dat(N = 2000, a1 = -2, a3 = -.3)
mean(dat$y)
dat$LP <- NULL # remove linear predictor

# fit a (misspecified) logistic regression model
m1 <- glm(y ~ ., data=dat, family="binomial")

# internal validation of m1 via bootstrap optimism with 10 resamples
# B = 10 for example but should be >= 200 in practice
m1_iv <- validate(m1, method="boot_optimism", B=10)

mape_stability(m1_iv)
```

`prediction_stability` *Plot prediction stability across bootstrap replicates*

Description

A prediction (in)stability plot shows estimated risk probabilities from models developed on resampled data evaluated on the original development data as a function of the 'apparent' prediction (prediction from original/development model evaluated on original data). A stable model should produce points that exhibit minimal dispersion. See Riley and Collins (2023).

Usage

```
prediction_stability(
  x,
  bounds = 0.95,
  smooth_bounds = FALSE,
  xlab,
  ylab,
  pch,
  cex,
  col,
  lty,
  span,
  subset,
  plot = TRUE
)
```

Arguments

`x`	an object produced by `validate` with method = "boot_*" (or `boot_optimism` with method="boot")
`bounds`	width of the 'stability interval' (percentiles of the bootstrap model predictions). NULL = do not add bounds to plot.
`smooth_bounds`	if TRUE, use `loess` to smooth the bounds (default = FALSE)
`xlab`	a title for the x axis
`ylab`	a title for the y axis
`pch`	plotting character (default = 16)
`cex`	controls point size (default = 0.4)
`col`	color of points (default = grDevices::grey(.5, .5))
`lty`	line type for bounds (default = 2)
`span`	controls the degree of smoothing (see `loess`; default = 0.75)
`subset`	vector of observations to include (row indices). If dataset is large plotting N points for B bootstrap resamples is demanding. This can be used to select a random subset of observations.
`plot`	if FALSE just returns stability matrix

Value

plots prediction (in)stability. The stability bounds are not smoothed. Invisibly returns stability matrix (where column 1 are original predictions) that can be used for creating plots with other packages/software.

References

Riley, R. D., & Collins, G. S. (2023). Stability of clinical prediction models developed using statistical or machine learning methods. Biometrical Journal, 65(8), 2200302. doi:10.1002/bimj.202200302

Examples

```
set.seed(456)
# simulate data with two predictors that interact
dat <- pmcalibration::sim_dat(N = 2000, a1 = -2, a3 = -.3)
mean(dat$y)
dat$LP <- NULL # remove linear predictor

# fit a (misspecified) logistic regression model
m1 <- glm(y ~ ., data=dat, family="binomial")

# internal validation of m1 via bootstrap optimism with 10 resamples
# B = 10 for example but should be >= 200 in practice
m1_iv <- validate(m1, method="boot_optimism", B=10)

prediction_stability(m1_iv)
```

---

print.internal_boot *Print a internal_boot object*

Description

Print a internal_boot object

Usage

```
## S3 method for class 'internal_boot'
print(x, digits = 2, ...)
```

Arguments

x	an object created with boot_optimism
digits	number of digits to print (default = 2)
...	additional arguments to print

Value

invisibly returns x and prints estimates to console

print.internal_cv *Print a internal_cv object*

Description

Print a internal_cv object

Usage

```
## S3 method for class 'internal_cv'
print(x, digits = 2, ...)
```

Arguments

x	an object created with crossval
digits	number of digits to print (default = 2)
...	additional arguments to print

Value

invisibly returns x and prints estimates to console

print.internal_validate *print a internal_validate object*

Description

print a internal_validate object

Usage

```
## S3 method for class 'internal_validate'
print(x, digits = 2, ...)
```

Arguments

x	a internal_validate object
digits	number of digits to print
...	optional arguments passed to print

Value

prints a summary

print.internal_validatesummary *Print summary of internal_validate object*

Description

Print summary of internal_validate object

Usage

```
## S3 method for class 'internal_validatesummary'
print(x, digits = 2, ...)
```

Arguments

x	a internal_validatesummary object
digits	number of digits to print
...	ignored

Value

invisible(x) - prints a summary

score_binary *Score predictions for binary events*

Description

Calculate scores summarizing discrimination/calibration of predictions against observed binary events. If score_fun is not defined when calling validate this function is used.

Usage

```
score_binary(y, p, ...)
```

Arguments

y	vector containing a binary outcome
p	vector of predictions
...	additional arguments. This function only supports calib_args as an optional argument. calib_args should contain arguments for pmcalibration::pmcalibration. If a calibration plot (apparent vs bias corrected calibration curves via cal_plot) is desired the argument 'eval' should be provided. This should be the points at which to evaluate the calibration curve on each boot resample or crossvalidation fold. A good option would be calib_args = list(eval = seq(min(p), max(p),

length.out=100)); where p are predictions from the original model evaluated on the original data. Dots can be used to supply additional arguments to user-defined functions.

Details

The following measures are returned in a named vector.

C the c-statistic (aka area under the ROC curve). Probability that randomly selected observation with y = 1 with have higher p compared to randomly selected y = 0.

Brier mean squared error - mean((y - p)^2)

Intercept Intercept from a logistic calibration model: glm(y ~ 1 + offset(qlogis(p)), family="binomial")

Slope Slope from a logistic calibration model: glm(y ~ 1 + qlogis(p), family="binomial")

Eavg average absolute difference between p and calibration curve (aka integrated calibration index or ICI).

E50 median absolute difference between p and calibration curve

E90 90th percentile absolute difference between p and calibration curve

Emax maximum absolute difference between p and calibration curve

ECI average squared difference between p and calibration curve. Estimated calibration index (Van Hoorde et al. 2015)

cal_plot if eval is specified (via calib_args), values for plotting apparent and bias-corrected calibration curves are returned (see `cal_plot`). By default these are omitted from the summary printed (see `summary.internal_validate`).

Logistic calibration and other calibration metrics from non-linear calibration curves assessing 'moderate-calibration' (Eavg, E50, E90, Emax, ECI; see references) are calculated via the `pmcalibration` package. The default settings can be modified by passing calib_args to `validate` call. calib_args should be a named list corresponding to arguments to `pmcalibration::pmcalibration`.

Value

a named vector of scores (see Details)

References

Austin PC, Steyerberg EW. (2019) The Integrated Calibration Index (ICI) and related metrics for quantifying the calibration of logistic regression models. *Statistics in Medicine*. 38, pp. 1–15. https://doi.org/10.1002/sim.8281

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 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

Examples

```
p <- runif(100)
y <- rbinom(length(p), 1, p)
score_binary(y = y, p = p)
```

`summary.internal_validate` *Summarize a internal_validate object*

Description

Summarize a internal_validate object

Usage

```
## S3 method for class 'internal_validate'
summary(object, ignore_scores = "^cal_plot", ...)
```

Arguments

`object`	created by call to `validate`
`ignore_scores`	a string used to identify scores to omit from summary. `score_binary` produces scores with prefix 'cal_plot' when a calibration plot is desired (see `cal_plot`) and these are ignored by default.
`...`	ignored

Value

a data.frame with 4 columns (apparent score, optimism, bias-corrected score, number of successful resamples/folds) and one row per score. Not all methods produce an optimism estimate so this row may be all NA. If confidence intervals have been added for all measures via `confint.internal_validate`, two additional columns containing lower and upper bounds for bias-corrected performance.

Examples

```
library(pminternal)
set.seed(456)
# simulate data with two predictors that interact
dat <- pmcalibration::sim_dat(N = 2000, a1 = -2, a3 = -.3)
mean(dat$y)
dat$LP <- NULL # remove linear predictor

# fit a (misspecified) logistic regression model
m1 <- glm(y ~ ., data=dat, family="binomial")

# internal validation of m1 via bootstrap optimism with 10 resamples
# B = 10 for example but should be >= 200 in practice
m1_iv <- validate(m1, method="boot_optimism", B=10)
```

```
summary(m1_iv)
```

---

validate *Get bias-corrected performance measures via bootstrapping or cross-validation*

Description

Performs internal validation of a prediction model development procedure via bootstrapping or cross-validation. Many model types are supported via the `insight` and `marginaleffects` packages or users can supply user-defined functions that implement the model development procedure and retrieve predictions. Bias-corrected scores and estimates of optimism (where applicable) are provided. See `confint.internal_validate` for calculation of confidence intervals.

Usage

```
validate(
  fit,
 method = c("boot_optimism", "boot_simple", ".632", "cv_optimism", "cv_average", "none"),
  data,
  outcome,
  model_fun,
  pred_fun,
  score_fun,
  B,
  ...
)
```

Arguments

`fit`	a model object. If fit is given the `insight` package is used to extract data, outcome, and original model call. Therefore, it is important that fit be supported by `insight` and implements the entire model development process (see Harrell 2015). A fit given after selection of variables by some method will not give accurate bias-correction. Model predictions are obtained via `marginaleffects::get_predict` with type = "response" so fit should be compatible with this function. If fit is provided the arguments data, outcome, model_fun, and pred_fun are all ignored.
`method`	bias-correction method. Valid options are "boot_optimism", "boot_simple", ".632", "cv_optimism", "cv_average", or "none" (return apparent performance). See details.
`data`	a data.frame containing data used to fit development model
`outcome`	character denoting the column name of the outcome in data
`model_fun`	for models that cannot be supplied via fit this should be a function that takes one named argument: 'data' (function should include ... among arguments). This function should implement the entire model development procedure (hyperparameter tuning, variable selection, imputation etc) and return an object that can be used by pred_fun. Additional arguments can be supplied by ...

`pred_fun` for models that cannot be supplied via fit this should be a function that takes two named arguments: 'model' and 'data' (function should include ... among arguments). 'model' is an object returned by model_fun. The function should return a vector of predicted risk probabilities of the same length as the number of rows in data. Additional arguments can be supplied by ...

`score_fun` function used to produce performance measures from predicted risks and observed binary outcome. Should take two named arguments: 'y' and 'p' (function should include ... among arguments). This function should return a named vector of scores. If unspecified `score_binary` is used and this should be good for most purposes.

`B` number of bootstrap replicates or crossvalidation folds. If unspecified B is set to 200 for method = "boot_*"/".632", or is set to 10 for method = "cv_*".

`...` additional arguments for user-defined functions. Arguments for producing calibration curves can be set via 'calib_args' which should be a named list (see `cal_plot` and `score_binary`). For method = "boot_optimism", "boot_simple", or ".632" users can specify a `cores` argument (e.g., `cores = 4`) to run bootstrap samples in parallel.

Details

Internal validation can provide bias-corrected estimates of performance (e.g., C-statistic/AUC, calibration intercept/slope) for a model development procedure (i.e., expected performance if the same procedure were applied to another sample of the same size from the same population; see references). There are several approaches to producing bias-corrected estimates (see below). It is important that the `fit` or `model_fun` provided implement the entire model development procedure, including any hyperparameter tuning and/or variable selection.

Note that `validate` does very little to check for missing values in predictors/features. If `fit` is supplied `insight::get_data` will extract the data used to fit the model and usually this will result in complete cases being used. User-defined model and predict functions can be specified to handle missing values among predictor variables. Currently any user supplied data will have rows with missing outcome values removed.

method Different options for the method argument are described below:

boot_optimism (default) estimates optimism for each score and subtracts from apparent score (score calculated with the original/development model evaluated on the original sample). A new model is fit using the same procedure using each bootstrap resample. Scores are calculated when applying the boot model to the boot sample (S_{boot}) and the original sample (S_{orig}) and the difference gives an estimate of optimism for a given resample ($S_{boot} - S_{orig}$). The average optimism across the B resamples is subtracted from the apparent score to produce the bias corrected score.

boot_simple implements the simple bootstrap. B bootstrap models are fit and evaluated on the original data. The average score across the B replicates is the bias-corrected score.

.632 implements Harrell's adaption of Efron's .632 estimator for binary outcomes (see rms::predab.resample and rms::validate). In this case the estimate of optimism is $0.632 \times (S_{app} - mean(S_{omit} \times w))$ where S_{app} is the apparent performance score and S_{omit} is the score estimated using the bootstrap model evaluated on the out-of-sample observations and w weights for the proportion of observations omitted (see Harrell 2015, p. 115).

cv_optimism estimate optimism via B-fold crossvalidation. Optimism is the average of the difference in performance measure between predictions made on the training vs test (held out fold) data. This is the approach implemented in `rms::validate` with method="crossvalidation".

cv_average bias corrected scores are the average of scores calculated by assessing the model developed on each fold evaluated on the test/held out data. This approach is described and compared to "boot_optimism" and ".632" in Steyerberg et al. (2001).

Calibration curves To make calibration curves and calculate the associated estimates (ICI, ECI, etc - see `score_binary`) `validate` uses the default arguments in `cal_defaults`. These arguments are passed to the `pmcalibration` package (see `?pmcalibration::pmcalibration` for options).

If a calibration plot (apparent vs bias corrected calibration curves via `cal_plot`) is desired, the argument 'eval' should be provided. This should be the points at which to evaluate the calibration curve on each boot resample or crossvalidation fold. A good option would be `calib_args = list(eval = seq(min(p), max(p), length.out=100))`; where p are predictions from the original model evaluated on the original data.

Number of resamples/folds is less than requested If the `model_fun` produces an error or if `score_binary` is supplied with constant predictions or outcomes (e.g. all(y == 0)) the returned scores will all be NA. These will be omitted from the calculation of optimism or other bias-corrected estimates (cv_average, boot_simple) and the number of successful resamples/folds will be < B. `validate` collects error messages and will produce a warning summarizing them. The number of successful samples is given in the 'n' column in the printed summary of an 'internal_validate' object.

It is important to understand what is causing the loss of resamples/folds. Some potential sources (which will need to be added to) are that for rare events the resamples/folds may be resulting in samples that have zero outcomes. For 'cv_*' this will especially be the case if B (n folds) is set high. There may be problems with factor/binary predictor variables with rare levels, which could be dealt with by specifying a `model_fun` that omits variables for the model formula if only one level is present. The issue may be related to the construction of calibration curves and may be addressed by more carefully selecting settings (see section above).

Value

an object of class internal_validate containing apparent and bias-corrected estimates of performance scores. If method = "boot_*" it also contains results pertaining to stability of predictions across bootstrapped models (see Riley and Collins, 2023).

References

Steyerberg, E. W., Harrell Jr, F. E., Borsboom, G. J., Eijkemans, M. J. C., Vergouwe, Y., & Habbema, J. D. F. (2001). Internal validation of predictive models: efficiency of some procedures for logistic regression analysis. Journal of clinical epidemiology, 54(8), 774-781.

Harrell Jr F. E. (2015). Regression Modeling Strategies: with applications to linear models, logistic and ordinal regression, and survival analysis. New York: Springer Science, LLC.

Efron (1983). "Estimating the error rate of a prediction rule: improvement on cross-validation". Journal of the American Statistical Association, 78(382):316-331

Van Calster, B., Steyerberg, E. W., Wynants, L., and van Smeden, M. (2023). There is no such thing as a validated prediction model. BMC medicine, 21(1), 70.

Riley, R. D., & Collins, G. S. (2023). Stability of clinical prediction models developed using statistical or machine learning methods. Biometrical Journal, 65(8), 2200302. doi:10.1002/bimj.202200302

Examples

```
library(pminternal)
set.seed(456)
# simulate data with two predictors that interact
dat <- pmcalibration::sim_dat(N = 2000, a1 = -2, a3 = -.3)
mean(dat$y)
dat$LP <- NULL # remove linear predictor

# fit a (misspecified) logistic regression model
m1 <- glm(y ~ ., data=dat, family="binomial")

# internal validation of m1 via bootstrap optimism with 10 resamples
# B = 10 for example but should be >= 200 in practice
m1_iv <- validate(m1, method="boot_optimism", B=10)
m1_iv
```

Index