Package ‘pmcalibration’

July 23, 2025

Type Package

Title Calibration Curves for Clinical Prediction Models

Version 0.2.0

Maintainer Stephen Rhodes <steverho89@gmail.com>

Description Fit calibrations curves for clinical prediction models and calculate several associated metrics (Eavg, E50, E90, Emax). Ideally predicted probabilities from a prediction model should align with observed probabilities. Calibration curves relate predicted probabilities (or a transformation thereof) to observed outcomes via a flexible non-linear smoothing function. 'pmcalibration' allows users to choose between several smoothers (regression splines, generalized additive models/GAMs, lowess, loess). Both binary and time-to-event outcomes are supported. See Van Calster et al. (2016) <doi:10.1016/j.jclinepi.2015.12.005>; Austin and Steyerberg (2019) <doi:10.1002/sim.8281>; Austin et al. (2020) <doi:10.1002/sim.8570>.

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.2

URL https://github.com/stephenrho/pmcalibration

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

Imports Hmisc, MASS, mgcv, splines, graphics, stats, methods, survival, pbapply, parallel, grDevices

Suggests rmarkdown, data.table, ggplot2, rms, simsurv

NeedsCompilation no

Author Stephen Rhodes [aut, cre, cph]

Repository CRAN

Date/Publication 2025-02-21 22:20:02 UTC

Contents

`cal_metrics` *Calculate calibration metrics from calibration curve*

Description

Calculates metrics used for summarizing calibration curves. See Austin and Steyerberg (2019)

Usage

```
cal_metrics(p, p_c)
```

Arguments

`p`	predicted probabilities
`p_c`	probabilities from the calibration curve

Value

a named vector of metrics based on absolute difference between predicted and calibration curve implied probabilities `d = abs(p - p_c)`

- Eavg - average absolute difference (aka integrated calibration index or ICI)
- E50 - median absolute difference
- E90 - 90th percentile absolute difference
- Emax - maximum absolute difference
- ECI - average squared difference. Estimated calibration index (Van Hoorde et al. 2015)

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

```
library(pmcalibration)

LP <- rnorm(100) # linear predictor
p_c <- invlogit(LP) # actual probabilities
p <- invlogit(LP*1.3) # predicted probabilities that are miscalibrated

cal_metrics(p = p, p_c = p_c)
```

---

get_curve — *Extract plot data from* `pmcalibration` *object*

Description

Extract plot data from `pmcalibration` object

Usage

```
get_curve(x, conf_level = 0.95)
```

Arguments

`x`	`pmcalibration` object
`conf_level`	width of the confidence interval (0.95 gives 95% CI). Ignored if call to `pmcalibration` didn't request confidence intervals

Value

data frame for plotting with 4 columns

- `p` - values for the x-axis (predicted probabilities - note these are *not* from your data and are only used for plotting)
- `p_c` - probability implied by the calibration curve given `p`
- `lower` and `upper` - bounds of the confidence interval

Examples

```
library(pmcalibration)
# simulate some data with a binary outcome
n <- 500
dat <- sim_dat(N = n, a1 = .5, a3 = .2)
head(dat)
# predictions
p <- with(dat, invlogit(.5 + x1 + x2 + x1*x2*.1))

# fit calibration curve
cal <- pmcalibration(y = dat$y, p = p, smooth = "gam", k = 20, ci = "pw")

cplot <- get_curve(cal, conf_level = .95)
head(cplot)

if (requireNamespace("ggplot2", quietly = TRUE)){
library(ggplot2)
ggplot(cplot, aes(x = p, y = p_c, ymin=lower, ymax=upper)) +
  geom_abline(intercept = 0, slope = 1, lty=2) +
  geom_line() +
  geom_ribbon(alpha = 1/4) +
  lims(x=c(0,1), y=c(0,1))
}
```

---

logistic_cal *Run logistic calibration*

Description

Fit the models required to assess calibration in the large (calibration intercept), calibration slope, and overall 'weak' calibration (see, e.g., Van Calster et al. 2019). Fits the models required to do the three likelihood ratio tests described by Miller et al. (1993) (see `summary.logistic_cal`).

Usage

```
logistic_cal(y, p)
```

Arguments

y	binary outcome
p	predicted probabilities (these will be logit transformed)

Value

an object of class `logistic_cal` containing `glm` results for calculating calibration intercept, calibration slope, and LRTs.

References

Van Calster, B., McLernon, D. J., Van Smeden, M., Wynants, L., & Steyerberg, E. W. (2019). Calibration: the Achilles heel of predictive analytics. BMC medicine, 17(1), 1-7.

Miller, M. E., Langefeld, C. D., Tierney, W. M., Hui, S. L., & McDonald, C. J. (1993). Validation of probabilistic predictions. Medical Decision Making, 13(1), 49-57.

Examples

```
library(pmcalibration)
# simulate some data
n <- 500
dat <- sim_dat(N = n, a1 = .5, a3 = .2)

# predictions
p <- with(dat, invlogit(.5 + x1 + x2 + x1*x2*.1))

logistic_cal(y = dat$y, p = p)
```

`plot.pmcalibration` *Plot a calibration curve*

Description

Plot a `pmcalibration` object. For binary outcomes, also plot the distribution of predicted risks by outcome. Alternatively you can use `get_curve()` to get the data required to plot the calibration curve.

Usage

```
## S3 method for class 'pmcalibration'
plot(
  x,
  conf_level = 0.95,
  riskdist = TRUE,
  linecol = "black",
  fillcol = "grey",
  ideallty = 2,
  idealcol = "red",
  ...
)
```

Arguments

`x`	a `pmcalibration` calibration curve
`conf_level`	width of the confidence interval (0.95 gives 95% CI). Ignored if call to `pmcalibration` didn't request confidence intervals
`riskdist`	add risk distribution plot under calibration curve (TRUE) or not (FALSE)

linecol	color of the calibration curve line
fillcol	color of the confidence interval
ideallty	line type of the ideal unit slope line
idealcol	color of the ideal unit slope line
...	other args for plot() (currently only lims and labs can be specified)

Value

No return value, called for side effects

Examples

```
library(pmcalibration)
# simulate some data with a binary outcome
n <- 500
dat <- sim_dat(N = n, a1 = .5, a3 = .2)
head(dat)
# predictions
p <- with(dat, invlogit(.5 + x1 + x2 + x1*x2*.1))

# fit calibration curve
cal <- pmcalibration(y = dat$y, p = p, smooth = "gam", k = 20, ci = "pw", plot = FALSE)

plot(cal, xlab = "Predicted Risk of Outcome") # customize plot
```

---

pmcalibration *Create a calibration curve*

Description

Assess calibration of clinical prediction models (agreement between predicted and observed probabilities) via different smooths. Binary and time-to-event outcomes are supported.

Usage

```
pmcalibration(
  y,
  p,
  smooth = c("gam", "none", "ns", "bs", "rcs", "lowess", "loess"),
  time = NULL,
  ci = c("sim", "boot", "pw", "none"),
  n = 1000,
  transf = NULL,
  eval = 100,
  plot = TRUE,
  ...
)
```

Arguments

`y` a binary or a right-censored time-to-event outcome. Latter must be an object created via `survival::Surv`.

`p` predicted probabilities from a clinical prediction model. For a time-to-event object `time` must be specified and `p` are predicted probabilities of the outcome happening by `time` units of time follow-up.

`smooth` what smooth to use. Available options:

- 'gam' (default) = generalized additive model via `mgcv::gam` and `mgcv::s`. Optional arguments are `bs`, `k`, `fx`, `method` (see `?mgcv::gam` and `?mgcv::s`)
- 'rcs' = restricted cubic spline using `rms::rcs`. Optional arguments for this smooth are `nk` (number of knots; defaults to 5) and `knots` (knot positions; set by `Hmisc::rcs.eval` if not specified)
- 'ns' = natural spline using `splines::ns`. Optional arguments are `df` (default = 6), `knots`, `Boundary.knots` (see `?splines::ns`)
- 'bs' = B-spline using `splines::bs`. Optional arguments are `df` (default = 6), `knots`, `Boundary.knots` (see `?splines::bs`)
- 'lowess' = uses `lowess(x, y, iter = 0)` based on `rms::calibrate`. Only for binary outcomes.
- 'loess' = uses `loess` with all defaults. Only for binary outcomes.
- 'none' = logistic or Cox regression with single predictor variable (for binary outcome performs logistic calibration when `transf = "logit"`). See `logistic_cal`

'rcs', 'ns', 'bs', and 'none' are fit via `glm` or `survival::coxph` and 'gam' is fit via `mgcv::gam` with `family = Binomial(link="logit")` for a binary outcome or `mgcv::cox.ph` when `y` is time-to-event.

`time` what follow up time do the predicted probabilities correspond to? Only used if `y` is a `Surv` object

`ci` what kind of confidence intervals to compute?

- 'sim' = simulation based inference. Note this is currently only available for binary outcomes. `n` samples are taken from a multivariate normal distribution with mean vector = coef(mod) and variance covariance = vcov(model).
- 'boot' = bootstrap resampling with `n` replicates. `y` and `p` are sampled with replacement and the calibration curve is reestimated. If `knots` are specified the same knots are used for each resample (otherwise they are calculated using resampled `p` or transformation thereof)
- 'pw' = pointwise confidence intervals calculated via the standard errors produced by relevant `predict` methods. Only for plotting curves; if selected, CIs are not produced for metrics (not available for smooth = 'lowess')

Calibration metrics are calculated using each simulation or boot sample. For both options percentile confidence intervals are returned.

`n` number of simulations or bootstrap resamples

`transf` transformation to be applied to `p` prior to fitting calibration curve. Valid options are 'logit', 'cloglog', 'none', or a function (must retain order of `p`). If unspecified defaults to 'logit' for binary outcomes and 'cloglog' (complementary log-log) for time-to-event outcomes.

eval	number of points (equally spaced between `min(p)` and `max(p)`) to evaluate for plotting (0 or NULL = no plotting). Can be a vector of probabilities.
plot	should a plot be produced? Default = TRUE. Plot is created with default settings. See `plot.pmcalibration`.
...	additional arguments for particular smooths. For ci = 'boot' the user is able to run samples in parallel (using the `parallel` package) by specifying a `cores` argument

Value

a `pmcalibration` object containing calibration metrics and values for plotting

References

Austin P. C., Steyerberg E. W. (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 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. https://doi.org/10.1016/j.jclinepi.2015.12.005

Austin, P. C., Harrell Jr, F. E., & van Klaveren, D. (2020). Graphical calibration curves and the integrated calibration index (ICI) for survival models. *Statistics in Medicine*, 39(21), 2714-2742. https://doi.org/10.1002/sim.8570

Examples

```
# binary outcome -------------------------------------
library(pmcalibration)
# simulate some data
n <- 500
dat <- sim_dat(N = n, a1 = .5, a3 = .2)
head(dat)
# predictions
p <- with(dat, invlogit(.5 + x1 + x2 + x1*x2*.1))

# fit calibration curve
cal <- pmcalibration(y = dat$y, p = p, smooth = "gam", k = 20, ci = "pw")

summary(cal)

plot(cal)

# time to event outcome -------------------------------------
library(pmcalibration)
if (requireNamespace("survival", quietly = TRUE)){
library(survival)

data('transplant', package="survival")
transplant <- na.omit(transplant)
transplant = subset(transplant, futime > 0)
```

```
transplant$ltx <- as.numeric(transplant$event == "ltx")

# get predictions from coxph model at time = 100
# note that as we are fitting and evaluating the model on the same data
cph <- coxph(Surv(futime, ltx) ~ age + sex + abo + year, data = transplant)

time <- 100
newd <- transplant; newd$futime <- time; newd$ltx <- 1
p <- 1 - exp(-predict(cph, type = "expected", newdata=newd))
y <- with(transplant, Surv(futime, ltx))

cal <- pmcalibration(y = y, p = p, smooth = "rcs", nk=5, ci = "pw", time = time)

summary(cal)

plot(cal)

}
```

`print.logistic_cal` *Print a* `logistic_cal` *object*

Description

Print a `logistic_cal` object

Usage

```
## S3 method for class 'logistic_cal'
print(x, digits = 2, conf_level = 0.95, ...)
```

Arguments

`x`	a `logistic_cal` object
`digits`	number of digits to print
`conf_level`	width of the confidence interval (0.95 gives 95% CI)
`...`	optional arguments passed to print

Value

prints a summary

print.logistic_calsummary

Print a logistic_cal summary

Description

Print a logistic_cal summary

Usage

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

Arguments

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

Value

prints a summary

print.pmcalibration

print a pmcalibration object

Description

print a pmcalibration object

Usage

```
## S3 method for class 'pmcalibration'
print(x, digits = 2, conf_level = 0.95, ...)
```

Arguments

x	a pmcalibration object
digits	number of digits to print
conf_level	width of the confidence interval (0.95 gives 95% CI)
...	optional arguments passed to print

Value

prints a summary

print.pmcalibrationsummary — *Print summary of pmcalibration object*

Description

Print summary of pmcalibration object

Usage

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

Arguments

`x`	a `pmcalibrationsummary` object
`digits`	number of digits to print
`...`	ignored

Value

invisible(x) - prints a summary

sim_dat — *Simulate a binary outcome with either a quadratic relationship or interaction*

Description

Function for simulating data either with a single 'predictor' variable with a quadratic relationship with logit(p) or two predictors that interact (see references for examples).

Usage

```
sim_dat(N, a1, a2 = NULL, a3 = NULL)
```

Arguments

`N`	number of observations to simulate
`a1`	value of the intercept term (in logits). This must be provided along with either `a2` or `a3`.
`a2`	value of the quadratic coefficient. If specified the linear predictor is simulated as follows: `LP <- a1 + x1 + a2*x1^2` where `x1` is sampled from a standard normal distribution.
`a3`	value of the interaction coefficient. If specified the linear predictor is simulated as follows: `LP <- a1 + x1 + x2 + x1*x2*a3` where `x1` and `x2` are sampled from independent standard normal distributions.

Value

a simulated data set with `N` rows. Can be split into 'development' and 'validation' sets.

References

Austin, P. C., & Steyerberg, E. W. (2019). The Integrated Calibration Index (ICI) and related metrics for quantifying the calibration of logistic regression models. Statistics in medicine, 38(21), 4051-4065.

Rhodes, S. (2022, November 4). Using restricted cubic splines to assess the calibration of clinical prediction models: Logit transform predicted probabilities first. https://doi.org/10.31219/osf.io/4n86q

Examples

```
library(pmcalibration)
# simulate some data with a binary outcome
n <- 500
dat <- sim_dat(N = n, a1 = .5, a3 = .2)

head(dat) # LP = linear predictor
```

`summary.logistic_cal` *Summarize a logistic_cal object*

Description

Summarize a logistic_cal object

Usage

```
## S3 method for class 'logistic_cal'
summary(object, conf_level = 0.95, ...)
```

Arguments

`object`	a `logistic_cal` object
`conf_level`	width of the confidence interval (0.95 gives 95% CI)
`...`	ignored

Details

The likelihood ratio tests proposed by Miller et al. test the following: The first assesses weak calibration overall by testing the null hypothesis that the intercept (a) and slope (b) are equal to 0 and 1, respectively. The second assesses calibration in the large and tests the intercept against 0 with the slope fixed to 1. The third test assesses the calibration slope after correcting for calibration in the large (by estimating a new intercept term). Note the p-values from the calibration intercept and calibration slope estimates will typically agree with the p-values from the second and third likelihood ratio tests but will not always match perfectly as the former are based on z-statistics and the latter are based on log likelihood differences (chi-squared statistics).

Value

estimates and conf_level*100 confidence intervals for calibration intercept and calibration slope. The former is estimated from a `glm` (family = binomial("logit")) where the linear predictor (logit(p)) is included as an offset. Results of the three likelihood ratio tests described by Miller et al. (2013) (see details).

References

Miller, M. E., Langefeld, C. D., Tierney, W. M., Hui, S. L., & McDonald, C. J. (1993). Validation of probabilistic predictions. Medical Decision Making, 13(1), 49-57.

`summary.pmcalibration` *Summarize a pmcalibration object*

Description

Summarize a pmcalibration object

Usage

```
## S3 method for class 'pmcalibration'
summary(object, conf_level = 0.95, ...)
```

Arguments

`object`	object created with `pmcalibration`
`conf_level`	width of the confidence interval (0.95 gives 95% CI). Ignored if call to `pmcalibration` didn't request confidence intervals
`...`	ignored

Value

prints a summary of calibration metrics. Returns a list of two tables: `metrics` and `plot`

Examples

```
library(pmcalibration)
# simulate some data with a binary outcome
n <- 500
dat <- sim_dat(N = n, a1 = .5, a3 = .2)
head(dat)
# predictions
p <- with(dat, invlogit(.5 + x1 + x2 + x1*x2*.1))

# fit calibration curve
cal <- pmcalibration(y = dat$y, p = p, smooth = "gam", k = 20, ci = "pw")

summary(cal)
```

Index