Package 'ipeval'

September 13, 2026

Type Package

Title Interventional Prediction Evaluation

Version 0.1.2

Description Provides methods to evaluate predictive performance of models that estimate risks under hypothetical intervention scenarios (interventional/causal/counterfactual predictions) with observational data subject to treatment-outcome confounding. Inverse probability of treatment weighting (IPTW) is used to construct a pseudopopulation in which all individuals receive a specified intervention, enabling assessment of agreement between predicted risks under the intervention and observed outcomes in the pseudo-population corresponding to that intervention. Supports interventions with binary or categorical treatment levels, applied at a single time point. Performance measures supported are AUC (Area Under the receiving operating characteristic Curve), Brier score, observed-expected ratio, and calibration plots. Methods implemented in this package are based on work by Keogh and Van Geloven (2024) <DOI:10.1097/EDE.0000000000001713>.

License GPL (>= 3)

Encoding UTF-8

Imports stats, survival, prodlim, nnet

Depends R (>= 3.5)

URL https://github.com/survival-lumc/ipeval, https://survival-lumc.github.io/ipeval/

BugReports https://github.com/survival-lumc/ipeval/issues

Suggests knitr, rmarkdown, testthat (>= 3.0.0), ipw, riskRegression, dplyr, tidyr, vdiffr

Config/testthat/edition 3

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

RoxygenNote 7.3.3

NeedsCompilation no

Author Jasper van Egeraat [aut, cre],
Nan van Geloven [aut, cph],
Ruth Keogh [aut, cph],
Leiden University Medical Center [fnd]

Maintainer Jasper van Egeraat <j.w.a.van_egeraat@lumc.nl>

Repository CRAN

Date/Publication 2026-09-13 10:10:01 UTC

Contents

`ip_score` *Interventional prediction score*

Description

Estimates the performance of predictions of binary or time-to-event outcomes under baseline interventions, by reweighting the data to form a pseudo-population in which every subject was assigned the treatment level of interest.

Usage

```
ip_score(
  object,
  data,
  outcome,
  treatment_formula,
  treatment_of_interest,
  metrics = c("auc", "brier", "scaled_brier", "oeratio", "calplot"),
  time_horizon,
  cens_model = "KM",
  cens_formula = ~1,
  null_model = TRUE,
  bootstrap = 0,
  bootstrap_progress = TRUE,
  iptw,
  ipcw,
  quiet = FALSE,
  strip_ipt_models = TRUE
)
```

Arguments

object	One of the following three options can be used to input the predictions to be evaluated: • a numeric vector, corresponding to the risk estimates under evaluation • a glm or coxph model, from which the predictions under evaluation can be derived. See details. • a (named) list, with one or more of the previous 2 options, for evaluating and comparing multiple prediction vectors/models at once.
data	A data.frame containing the observed outcome, assigned treatment, and necessary adjustment variables (confounders) for the evaluation of object.
outcome	The outcome of interest within data. This could either be the name of a single numeric/logical column in data, or a Surv object for time-to-event data, e.g. Surv(time, status), if time and status are columns in data.
treatment_formula	A formula which indicates the treatment/intervention variable (left hand side) and the adjustment variables (right hand side) in the data. E.g. A ~ L. The left hand side can be either a binary treatment (coded as 0/1 numeric, logical or factor) or a treatment with more than two categories (coded as a factor). The right hand side variables are used to estimate the inverse probability of treatment weights (IPTW) with logistic/multinomial regression. The IPTW can also be specified directly as a vector using the iptw argument, in which case the right hand side of treatment_formula is ignored (the left hand side must still indicate the treatment, i.e. A ~ 1).
treatment_of_interest	A treatment level under which the predictions should be evaluated.
metrics	A character vector specifying which performance metrics to compute. Options are c("auc", "brier", "scaled_brier", "oeratio", "calplot"). See details.
time_horizon	For time to event data, the prediction horizon of interest.
cens_model	Model for estimating inverse probability of censored weights (IPCW). Methods currently implemented are Kaplan-Meier ("KM") or Cox ("cox"), with censoring times derived from the Surv object specified under outcome, reversing the event indicator, see details. KM is only supported when the right hand side of cens_formula is 1.
cens_formula	Model formula from which the right hand side is used in estimating the censoring probabilities. The left hand side must be left blank. E.g. ~ x1 + x2.
null_model	If TRUE fits a model without covariates that estimates the same probability for all subjects in data. The model is fitted using the reweighted data in which all subjects 'counterfactually' received the treatment level of interest (using the IPTW, as estimated using the treatment_formula or as given by the iptw argument). For time-to-event outcomes, the null model is also fitted using the IPCW, as estimated using the cens_formula, or as given by the ipcw argument. The null_model can be used as reference (baseline) model. If bootstrapping, a new null model is fit during each bootstrap iteration.
bootstrap	If this is an integer greater than 0, this indicates the number of bootstrap iterations, used to compute 95% confidence intervals around the performance metrics based on percentiles of the bootstrap results.

`bootstrap_progress`
: if set to TRUE, print a progress bar indicating the progress of the bootstrap procedure.

`iptw`
: A numeric vector, containing the inverse probability of treatment weights. If iptw is not specified, these weights are computed using the treatment_formula, but they can be specified directly via this argument. A user-defined function can also be specified, which takes as input 'data' and returns a numeric vector of IPTW weights. See details.

`ipcw`
: A numeric vector, containing the inverse probability of censoring weights at the time_horizon, or at a subject's event time, whichever happens first. For subjects who are censored before the time_horizon, the ipcw can be left at NA. If ipcw is not specified, these weights are computed using the cens_formula, but they can be specified directly via this argument. A user-defined function can also be specified, which takes as input 'data' and returns a numeric vector of IPCW weights. See details.

`quiet`
: If set to TRUE, don't print assumptions.

`strip_ipt_models`
: If set to TRUE (default), unnecessary components from the IPT- and IPC-model objects are not stored to save memory. Set to FALSE if you want to store the full IPT/IPC model objects.

Details

When supplying a glm or coxph model as object, the function will try to estimate risks from the model under the treatment level of interest for all subjects in data. If the model does not have the treatment as covariate, it is assumed it already estimates the risk under the treatment level of interest (e.g. because the model was fitted in a population all receiving the treatment level of interest). Alternatively, if the model includes the treatment as covariate, the function estimates the risk under the treatment level of interest for all subjects in data, even if they were assigned an alternative treatment level.

All performance metrics are computed on the weighted population mimicking the hypothetical situation where every subject's treatment level was set to the treatment level of interest (and where nobody was censored before time_horizon). "auc" is area under the (ROC) curve. "brier" is Brier score, ranging from 0 to 1. Scaled brier score is also available (metrics = "scaled_brier"), which expresses the Brier score relative to the null model. For the O/E ratio, the numerator (observed) is the (weighted) fraction of 'observed' events in the pseudo-population, and the denominator (expected) is the (unweighted) mean of risk estimates for all subjects in data. The calplot option generates a calibration plot, with default 8 subgroups. More/less subgroups can be specified by appending "calplot" with a number indicating the number of subgroups, e.g. metrics = "calplot10" for 10 subgroups.

The null model estimates are obtained as the weighted mean outcome in the subset observed with the treatment level of interest (and not censored before time_horizon). For time-to-event data, this null prediction could also be computed using a weighted Kaplan-Meier estimator, which would be more efficient, but computationally slower.

The censoring distribution is estimated with a Kaplan-Meier estimator implemented using 'prodlim::prodlim(..., reverse = TRUE)'. This correctly estimates the censoring distribution when there are ties between event and censoring times. When using a Cox model to estimate the censoring distribution, the

event indicator is reversed. This does not preserve the usual tie-handling convention: in standard survival analysis, censoring is assumed to occur after events at the same time point, but after reversing the indicator the opposite ordering is assumed. A possible workaround is to add a small positive offset (‘epsilon‘) to all censoring times before fitting the censoring model.

When supplying time-to-event data in which some subjects have follow-up beyond the specified time horizon, the full follow-up time is used when estimating the censoring distribution, rather than truncating follow-up at the prediction horizon. This does not affect the IPCW when the censoring distribution is estimated using the Kaplan-Meier estimator, but it can affect the IPCW when using a Cox model. If the censoring model should be estimated using information only up to the prediction horizon, the user must administratively censor subjects at the prediction horizon before supplying the data to ‘ip_score()‘.

Bootstrapping is not possible when manually specifying the IPTW/IPCW as numeric vectors. If specifying a user-defined function that computes the IPTW/IPCW given data, it is possible. The given function will be called on each bootstrapped dataset and resulting metrics are used to compute the 95% CIs with the percentile method. More advanced techniques, such as thresholding extreme IP weights, can be implemented through user-defined weight function. The censoring weight returned by this function should be the 1 / probability of remaining uncensored till the time_horizon, or till a subject's event time, whichever happens first. For subjects who are censored before the time_horizon, the ipcw can be left at NA.

Value

An object of class ‘ip_score‘, for which the ‘print()‘ and ‘plot()‘ methods are implemented. The object is a nested list containing:

- ‘$score‘, contains the estimated predictive performance metrics.
- ‘$bootstrap‘, if requested, the 95% confidence intervals of the performance metrics, and the performance metrics for each individual bootstrap iteration.
- ‘$outcome‘, the observed outcomes in data.
- ‘$treatment‘, the observed treatment levels in data.
- ‘$predictions‘, the predictions to be evaluated, i.e. the estimated probability of event under the intervention of interest for each subject.
- ‘$ipt‘, method, model and inverse probability of treatment weights (IPTW). The IPTW are NA for subjects who did not receive the treatment level of interest.
- ‘$ipc‘, method, model and inverse probability of censoring weights (IPCW). The IPCW are NA for subjects who were censored before the time_horizon.
- ‘$pseudopop‘, binary vector indicating which subjects in data were re-weighted to create the pseudo-population. These are the subjects who were observed to receive the treatment level of interest and were not censored before the time_horizon, if applicable.

The print method summarizes the results and (if quiet = FALSE), prints the assumptions required for valid inference.

References

Keogh RH, Van Geloven N. Prediction Under Interventions: Evaluation of Counterfactual Performance Using Longitudinal Observational Data. Epidemiology. 2024;35(3):329-339.

Boyer CB, Dahabreh IJ, Steingrimsson JA. Estimating and Evaluating Counterfactual Prediction Models. Statistics in Medicine. 2025;44(23-24):e70287.

Pajouheshnia R, Peelen LM, Moons KGM, Reitsma JB, Groenwold RHH. Accounting for treatment use when validating a prognostic model: a simulation study. BMC Medical Research Methodology. 2017;17(1):103.

Examples

```
n <- 1000

data <- data.frame(L = rnorm(n), P = rnorm(n))
data$A <- rbinom(n, 1, plogis(data$L))
data$Y <- rbinom(n, 1, plogis(0.1 + 0.5*data$L + 0.7*data$P - 2*data$A))

random <- runif(n, 0, 1)
model <- glm(Y ~ A + P, data = data, family = "binomial")

score <- ip_score(
  object = list(random, model),
  data = data,
  outcome = Y,
  treatment_formula = A ~ L,
  treatment_of_interest = 0,
)
print(score)
plot(score)
```

`observed_score` *Performance in observed dataset*

Description

This function computes the performance of the predictions in the given data, which may contain a mix of treated and untreated subjects. It exists only to demonstrate the difference between 'normal' performance and counterfactual performance. It is not user friendly and should not be relied on. Consider using riskRegression::Score() as an alternative.

Usage

```
observed_score(
  object,
  data,
  outcome,
  metrics = c("auc", "brier", "scaled_brier", "oeratio", "calplot"),
  time_horizon,
  cens_model = "KM",
  cens_formula = ~1,
  null_model = TRUE,
  ipcw
)
```

Arguments

object	One of the following three options to be validated: • a numeric vector, corresponding to risk predictions • a glm model • a (named) list, with one or more of the previous 2 options, for validating and comparing multiple models at once.
data	A data.frame containing the observed outcome.
outcome	The outcome, to be evaluated within data. This could either be the name of a numeric/logical column in data, or a Surv object for time-to-event data, e.g. Surv(time, status), if time and status are columns in data.
metrics	A character vector specifying which performance metrics to be computed. Options are c("auc", "brier", "oeratio", "calplot").
time_horizon	For time to event data, the prediction horizon of interest.
cens_model	Model for estimating inverse probability of censored weights (IPCW). Methods currently implemented are Kaplan-Meier ("KM") or Cox ("cox"), both applied to the censored times. KM is only supported when the right hand side of cens_formula is 1.
cens_formula	Formula for which the r.h.s. determines the censoring probabilities. I.e. ~ x1 + x2.
null_model	If TRUE fit a risk prediction model which ignores the covariates and predicts the same value for all subjects. For time-to-event outcomes, the subjects are 'counterfactually' uncensored (using the IPCW, as estimated using the cens_formula, or as given by the ipcw argument).
ipcw	A numeric vector, containing the inverse probability of censor weights. These are normally computed using the cens_formula, but they can be specified directly via this argument.

Value

Performance metrics in the observed dataset.

Examples

```
n <- 1000

data <- data.frame(L = rnorm(n), P = rnorm(n))
data$A <- rbinom(n, 1, plogis(data$L))
data$Y <- rbinom(n, 1, plogis(0.1 + 0.5*data$L + 0.7*data$P - 2*data$A))

random <- runif(n, 0, 1)
model <- glm(Y ~ A + P, data = data, family = "binomial")

observed_score(
  object = list("ran" = random, "mod" = model),
  data = data,
  outcome = Y,
  metrics = c("auc", "brier", "oeratio")
)
```

`plot.ip_score` *Plot calibration curve for an ip_score object*

Description

Produces a calibration plot comparing predicted and observed outcomes under the intervention of interest.

Usage

```
## S3 method for class 'ip_score'
plot(
  x,
  xlim = c(0, 1),
  ylim = c(0, 1),
  pty = "s",
  asp = NA,
  main,
  xlab = "Predicted",
  ylab = "Observed",
  cex.main = 0.8,
  legend = "topleft",
  ...
)
```

Arguments

`x`	The ‘ip_score‘ object returned by `ip_score`
`xlim`	The x limits of the plot, c(x1, x2)
`ylim`	The y limits of the plot, c(y1, y2)
`pty`	A character specifying the type of plot region to be used; "s" generates a square plotting region and "m" generates the maximal plotting region.
`asp`	The y/x aspect ratio
`main`	Character string giving the main title of the plot.
`xlab`	Character string specifying the x-axis label.
`ylab`	Character string specifying the y-axis label.
`cex.main`	Numeric value controlling the size of the main title.
`legend`	Keyword denoting the positioning of the legend. Can be "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right" and "center", or alternatively, "none" to hide the legend.
`...`	Currently ignored.

Details

Subjects are grouped into subgroups according to percentiles of the estimated risks. For each subgroup, the x-coordinate is the mean estimated risk of the subgroup. The y-coordinate is the inverse-probability-weighted proportion of 'observed' events.

The observed and predicted calibration subgroup coordinates are computed by the ip_score function and are stored in 'x$score$calplot', where 'x' is the 'ip_score' object. These raw values can be used to create custom calibration plots when additional control is needed.

If ip_score was run with bootstrap resampling ('bootstrap > 0'), additional panels are produced for every evaluated model showing the calibration curves from all bootstrap replicate in grey.

This method is available only when "calplot" was included in the 'metrics' argument of ip_score.

Value

Invisibly returns 'x'.

Examples

```
n <- 1000

data <- data.frame(L = rnorm(n), P = rnorm(n))
data$A <- rbinom(n, 1, plogis(data$L))
data$Y <- rbinom(n, 1, plogis(0.1 + 0.5*data$L + 0.7*data$P - 2*data$A))

random <- runif(n, 0, 1)
model <- glm(Y ~ A + P, data = data, family = "binomial")

score <- ip_score(
  object = list(random, model),
  data = data,
  outcome = Y,
  treatment_formula = A ~ L,
  treatment_of_interest = 0,
  bootstrap = 20,
  bootstrap_progress = FALSE,
  metrics = "calplot"
)

plot(score)
```

Index