Package 'ipeval'

May 6, 2026

Type Package

Title Evaluation of Interventional Predictions

Version 0.1.0

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. Package supports binary and time-to-event outcomes under binary interventions made 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

RoxygenNote 7.3.3

Imports stats, survival, prodlim

Depends R (>= 3.5)

URL https://jvelumc.github.io/ipeval/, https://github.com/jvelumc/ipeval

BugReports https://github.com/jvelumc/ipeval/issues

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

Config/testthat/edition 3

VignetteBuilder knitr

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 2026-05-06 20:30:11 UTC

Contents

`ip_score` *Counterfactual validation score*

Description

Estimates the predictive performance of predictions under interventions, by forming a weighted pseudopopulation in which every subject was assigned the treatment of interest.

Usage

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

Arguments

`object`	One of the following three options to be validated: • a numeric vector, corresponding to risk predictions under intervention of interest. • a glm or coxph model, capable of estimating risks under intervention of interest. See details. • 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, assigned treatment, and necessary confounders for the validation of object.
`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.
`treatment_formula`	A formula which identifies the treatment (left hand side) and the confounders (right hand side) in the data. E.g. A ~ L. The confounders are used to estimate the inverse probability of treatment weights (IPTW) model. The IPTW can also be specified themselves using the iptw argument, in which case the right hand side of this formula is ignored.
`treatment_of_interest`	A treatment level for which the counterfactual perormance measures should be evaluated.
`metrics`	A character vector specifying which performance metrics to be computed. Options are c("auc", "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"), 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. The model is fitted using the data in which all subjects are counterfactually assigned the treatment of interest (using the IPTW, as estimated using the treatment_formula or as given by the iptw argument). For time-to-event outcomes, the subjects are also counterfactually uncensored (using the IPCW, as estimated using the cens_formula, or as given by the ipcw argument).
`stable_iptw`	if TRUE, estimate stabilized IPT-weights. See details.
`bootstrap`	If this is an integer greater than 0, this indicates the number of bootstrap iterations, to compute 95% confidence intervals around the performance metrics.
`bootstrap_progress`	if set to TRUE, print a progress bar indicating the progress of bootstrap procedure.

iptw	A numeric vector, containing the inverse probability of treatment weights. These are normally computed using the treatment_formula, but they can be specified directly via this argument. If specified via this argument, bootstrap is not possible.
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. If specified via this argument, bootstrap is not possible.
quiet	If set to TRUE, don't print assumptions.

Details

When supplying a glm or coxph model, the model should be able to estimate risks under the intervention of interest. This could be done in two ways: the model does not have a treatment covariate, and always estimates the risk under intervention of interest. Alternatively, the model has a covariate for treatment. This function then automatically estimates the risk under the treatment of interest for all subjects, even if they were assigned alternative treatment.

All performance metrics are computed on the weighted population in which every subject was counterfactually assigned treatment of interest. `auc` is area under the (ROC) curve. Brier score is defined as `1 / sum(iptw) sum(predictions_i - outcome_i)^2`. Scaled brier score is also available (`metrics = "scaled_brier"`). For the O/E ratio, the numerator (observed) is the weighted fraction of events in the pseudopopulation, and the denominator (expected) is the unweighted mean of risk estimates of the original unweighted population. The `calplot` option generates a calibration plot, with default 8 knots. More/less knots can be specified by appending calplot with a number indicating the number of knots, e.g. `metrics = calplot10` for 10 knots.

Stabilized IPT-weigths are computed by estimating a null model for treatment. E.g. weights are `P(A = a) / P(A = a | L = l)`, if the given treatment_formula is `A ~ L`.

Value

A list with performance metrics, of class 'ip_score", for which the print and plot methods are implemented.

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")
naive_perfect <- data$Y

score <- ip_score(
  object = list("ran" = random, "mod" = model, "per" = naive_perfect),
  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. It does not support time-to-event data.

Usage

```
observed_score(
  object,
  data,
  outcome,
  metrics = c("auc", "brier", "oeratio", "calplot")
)
```

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 should be the name of a numeric column in data.
`metrics`	A character vector specifying which performance metrics to be computed. Options are c("auc", "brier", "oeratio", "calplot").

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")
)
```

Index