Package: ipeval (via r-universe)

June 8, 2026

Type Package

Title Interventional Prediction Evaluation

Version 0.1.0.9000

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

LazyData true

Imports stats, survival, prodlim, nnet

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, dplyr, tidyr, vdiffr

Config/testthat/edition 3

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

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

Date/Publication 2026-06-08 16:53:53 UTC

RemoteUrl https://github.com/jvelumc/ipeval

RemoteRef HEAD

RemoteSha 6691c65e053f00cce9a2b48fdfc2155c51a1f65b

Contents

add_lag_terms 2
ip_score 3
ip_score_long 7
observed_score 12
wide_to_long 14

Index **16**

`add_lag_terms` *Add lagged versions of a longitudinal variable*

Description

Creates one or more lagged versions of a variable in long-format data. Lagged values are computed separately within each subject and added as new columns. The data must be sorted by id and visit time before calling this function.

Usage

```
add_lag_terms(df, var, lag = 1, fill = 0)
```

Arguments

`df`	A long-format data frame containing an ‘id‘ column identifying subjects.
`var`	Character string giving the name of the variable to lag.
`lag`	Integer vector specifying the lag(s) to create.
`fill`	Value used when a lagged observation is unavailable (e.g. at the first visit). Defaults to 0.

Value

The input data frame with additional columns named ‘(var)_lag_(lag)‘ containing the requested lagged values.

Examples

```
df <- data.frame(
  id = c(1, 1, 1, 2, 2, 2),
  visit_time = c(0, 1, 2, 0, 1, 2),
  A = c(0, 1, 1, 1, 0, 0)
)

# Add a 1-visit lag of A
add_lag_terms(df, "A")

# Add both 1- and 2-visit lags
add_lag_terms(df, "A", lag = c(1, 2))
```

---

ip_score *Interventional prediction score*

Description

Estimates the performance of predictions of binary 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", "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,
  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 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 (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). The IPTW can also be specified directly using the iptw argument, in which case the right hand side of this 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 for which the right hand side is used in estimating the censoring probabilities. E.g. ~ x1 + x2.

`null_model` If TRUE fits a model without covariates (intercept only) 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 subjects are also 'counterfactually' uncensored (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.

`stable_iptw` if TRUE, estimate stabilized IPT-weights. Does not influence the metrics. See details.

`bootstrap` If this is an integer greater than 0, this indicates the number of bootstrap iterations, used to compute 95 around the performance metrics.

`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 their event time, whichever happens first. 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 always estimates the risk under 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). "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"). For the O/E ratio, the numerator (observed) is the (weighted) fraction of 'observed' events in the pseudopopulation, and the denominator (expected) is the (unweighted) mean of risk estimates in the original population. 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 KM censoring distribution is estimated 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 flipped. 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.

The null model is computed by the weighted mean outcome in the pseudopopulation. For survival data, this null prediction could also be computed using a weighted Kaplan-Meier estimator, which would be more efficient, but computationally slower.

Stabilized IPT-weigths can be computed by P(A = a) / P(A = a | L = l), if the given treatment_formula is A ~ L. In the setting that we consider here, the numerator of this expression is constant. The

resulting performance metrics are therefore not impacted by multiplication of all weights with the same constant.

Bootstrapping is not possible when manually specifiying the IPTW/IPCW as numeric vectors. If specifying a user-defined function that computes the ITPW/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. More advanced techniques, such as thresholding extreme IP weights, can be implemented through a user-defined weight function. The censoring weight returned by this function should be the 1 / probability of remaining uncensored at the time horizon, or at their event time, whichever happens first.

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 performance metrics, and the performance metrics for each individual bootstrap iteration.
- '$outcome', the observed outcomes in data.
- '$treatment', the observed treatment levels in data.
- '$predictions', the predictions to be evaluated, i.e. the probability of event under the intervention of interest for each subject.
- '$ipt', method, model and inverse probability of treatment weights (IPTW). These are NA for subjects who are not directly used in the pseudo-population.
- '$ipc', method, model and inverse probability of censoring weights (IPCW). These are NA for subjects who were censored.
- '$pseudopop', binary vector indicating which subjects of the original population were used to create the pseudo-population, by receiving the treatment level of interest and remaining uncensored, 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)
```

`ip_score_long` *Interventional prediction score for longitudinal treatment strategies*

Description

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

Usage

```
ip_score_long(
  probabilities,
  data_outcome,
  data_long,
  treatment_formula,
  treatment_of_interest,
  metrics = c("auc", "brier", "oeratio", "calplot"),
  visit_times,
  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

probabilities	A numeric vector corresponding to the risk estimates under evaluation, or a (named) list of multiple numeric vectors for evaluating and comparing multiple vectors.
data_outcome	A dataframe containing the observed outcomes. It must consist of 1 row per subject, with columns 'id', 'time', and 'status'. Time can be continuous, status represents the binary outcome status at 'time'.
data_long	A dataframe in 'long' format, containing the time-dependent treatment variable, the (potentially time-dependent) adjustment variables (confounders) and other time dependent covariates, possibly required for modeling the censoring mechanism. Baseline covariates can also be included, in which case they would be repeated for every visit. The data should be in 'long' format, i.e. each subject has one row for each of their visits. Must include the columns 'id' and 'visit_time'. There should not be any visits after a subject's (survival) time. All subjects should have data at each visit.
treatment_formula	A formula which indicates the treatment/intervention (left hand side) and the adjustment variables (right hand side) in data_long. E.g. A ~ L + A_lag_1. The left hand side can be either a binary treatment (codes as 0/1 numeric, logical or factor) or a treatment with more than two categories (coded as factor). The adjustment variables (right hand side) are used to estimate the inverse probability of treatment weights (IPTW). All variables on the right hand side must be present in data_long. The IPTW can also be specified directly using the iptw argument, in which case the right hand side of this formula is ignored (the left hand side must still indicate the treatment, i.e. A ~ 1).
treatment_of_interest	A treatment strategy under which the predictions should be evaluated. Should be in the form of a vector, with one element per visit. See details.
metrics	A character vector specifying which performance metrics to be computed. Options are c("auc", "brier", "scaled_brier", "oeratio", "calplot").
visit_times	A numeric vector, indicating the times of the visits. The first visit must always be at time 0. Should have the same length as 'treatment_of_interest'.
time_horizon	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 time,status variables in data_outcome, reversing the status indicator, see details. KM is only supported when the right hand side of cens_formula is 1.
cens_formula	Model formula used for estimating the censoring probabilities, e.g. ~ x1 + x2. Could consist of only baseline variables but also of time-dependent variables. All variables specified must be present in data_long.

`null_model` If TRUE a model without covariates (intercept only) is fitted to data that estimates the same probability for all subjects in data. The model is fitted using the reweighted data in which all subjects 'counterfactually' assigned the treatment strategy 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). The null model can be used as a reference (baseline) model.

`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 with same length as data_long, 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 arguments 'data_outcome' and 'data_long' and returns a numeric vector of the IPTW weight. The first argument 'data_outcome' should probably not be used in this function. See details.

`ipcw` A numeric vector, containing the inverse probability of censoring weights at the time horizon, or at their event time, whichever happens first. 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 arguments 'data_outcome' and 'data_long' and returns a numeric vector of the IPCW weight. 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

To form a pseudo-population that represents a setting in which everybody received treatment level 1 during five visits, set 'treatment_of_interest' to 'c(1,1,1,1,1)'. Any pattern can be chosen, e.g. 'c(1,0,1,0,1)' is also valid, as long as the number of visits equals 'n_visits'. Alternatively, one could also set this argument to "always" or "never" as a shortcut for 'rep(1, n_visits)' or 'rep(0, n_visits)'. Treatment strategies where treatment is only set at certainvisits are also possible via 'NA', e.g. use 'c(1,1,NA, NA, NA)' when you want to form a pseudo-population where everybody's treatment levels are set to 1 at the first 2 visits, and their remaining three can be whatever they would have normally been after those first two under the natural course. If treatment is categorical, this should be a character vector denoting the treatment levels of interest, e.g 'c("active","control").

The KM censoring distribution is estimated 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 flipped. 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.

Bootstrapping is not possible when manually specifiying the IPTW/IPCW as numeric vectors. If specifying a model user-defined function that computes the ITPW/IPCW given data, it is possible. The given function will be called on each bootstrapped dataset and resulting metrics are used to compute the (empirical) 95 weights, can be implemented through a user-defined weight function. The censoring weight returned by this function should be the 1 / probability of remaining uncensored at the time horizon, or at their event time, whichever happens first.

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 performance metrics, and the performance metrics for each individual bootstrap iteration.
- ‘$outcome‘, the observed outcomes from data_outcome.
- ‘$treatment‘, the observed treatment levels from data_long.
- ‘$predictions‘, the predictions to be evaluated, i.e. the probability of event under the intervention strategy of interest for each subject.
- ‘$ipt‘, method, model and inverse probability of treatment weights (IPTW). These are NA for subjects who are not directly used in the pseudo-population.
- ‘$ipc‘, method, model and inverse probability of censoring weights (IPCW). These are NA for subjects who were censored.
- ‘$pseudopop‘, binary vector indicating which subjects of the original population were used to create the pseudo-population, by receiving the treatment strategy of interest and remaining uncensored, 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.

See Also

`ip_score`

Open the corresponding vignette for more extensive examples with `vignette("longitudinal", package = "ipeval")`

Examples

```
set.seed(5)
n <- 1000
data <- data.frame(id = 1:n)

data <- within(data, {
  # 2 visits at t = 0 and t = 2. Time dependent confounding between A and L
  L0 <- rnorm(n)
  A0 <- rbinom(n, 1, plogis(0.7*L0))
  L1 <- rnorm(n, 0.8*L0 - 0.2*A0)
  A1 <- rbinom(n, 1, plogis(0.7*L1 + 0.6*A0))
  # Quickly generate random survival times as an example
  time <- runif(n, 0, 6)
  status <- rbinom(n, 1, 0.5)
})
# A0 is at time t = 0, A1 is at time t = 2

# If you have data in this 'wide' form, then usually you can use the convenience
# functions to make it suitable for ip_score_long
head(data)
# note that due to our simulation there may be measurements after a subject was
# censored or had event. This will be removed later.

data_outcome <- data[, c("id", "time", "status")]
data_long <- wide_to_long(
  df = data,
  baseline_variables = "id",
  wide_variables = list("A" = c("A0", "A1"),
                        "L" = c("L0", "L1")),
  visit_times = c(0,2),
  outcome_times = data$time
)
data_long <- add_lag_terms(data_long, "A")

head(data_long)
# note that the measurements that were generated after survival time have been
# removed by wide_to_long().

# To get some predictions to evaluate, here is a model that
# randomly predicts a number between 0.2 and 0.8
random_model <- runif(n, min = 0.2, max = 0.8)

# Our treatment at visit i depends on confounder value L at same visit and,
# if visit = 2, on treatment level at previous visit. We will illustrate a correctly
# specified IPTW model with A ~ L + A_lag_1.

# Estimate performance of random predictions in a pseudopopulation in
# which everybody was assigned treatment strategy {0, 0}
ip_score_long(
  probabilities = random_model,
  data_outcome = data_outcome,
  data_long = data_long,
```

```
  treatment_formula = A ~ L + A_lag_1,
  treatment_of_interest = c(0,0),
  visit_times = c(0,2),
  time_horizon = 4
)

# Performance of random predictions in a pseudopopulation in
# which everybody was assigned treatment level 1 at visit 1, and whatever
# they would normally have after that (natural course) at visit 2, and modelling
# the censoring distribution with Cox model
ip_score_long(
  probabilities = random_model,
  data_outcome = data_outcome,
  data_long = data_long,
  treatment_formula = A ~ L + A_lag_1,
  treatment_of_interest = c(1,NA),
  visit_times = c(0,2),
  time_horizon = 4,
  cens_model = "cox",
  cens_formula = ~ A + A_lag_1 + L
)
```

`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", "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")
)
```

`wide_to_long` *Convert wide longitudinal data to long format*

Description

Reshapes repeated measurements stored in wide format into a long-format dataset with one row per subject per visit. Rows corresponding to visits occurring after the subject's outcome time are removed.

Usage

```
wide_to_long(
  df,
  baseline_variables = c("id"),
  wide_variables,
  visit_times,
  outcome_times
)
```

Arguments

`df`	A data frame containing one row per subject.
`baseline_variables`	Character vector of baseline variables to retain and repeat across visits. Must include '"id"'.
`wide_variables`	Named list mapping long-format variable names to the corresponding wide-format column names for each visit.
`visit_times`	Numeric vector giving the visit times corresponding to the repeated measurements in 'wide_variables'.
`outcome_times`	Numeric vector of outcome or follow-up times, used to remove visits occurring after a subject's observed follow-up.

Value

A data frame in long format containing one row per subject per observed visit, with columns 'id', 'visit_time', the specified baseline variables, and the reshaped longitudinal variables.

Examples

```
data <- data.frame(
  id = 1:3,
  A0 = c(0, 1, 0),
  A1 = c(1, NA, 0),
  L0 = c(0.2, -1.1, 0.5),
  L1 = c(0.8, NA, 0.1),
  time = c(3, 1, 4)
)
```

```
wide_to_long(
  df = data,
  baseline_variables = "id",
  wide_variables = list(
    A = c("A0", "A1"),
    L = c("L0", "L1")
  ),
  visit_times = c(0, 2),
  outcome_times = data$time
)
```

Index

add_lag_terms, 2

ip_score, 3, *10*
ip_score_long, 7

observed_score, 12

wide_to_long, 14