Package ‘ipd’

January 7, 2025

Title Inference on Predicted Data

Version 0.1.4

Maintainer Stephen Salerno <ssalerno@fredhutch.org>

Description
Performs valid statistical inference on predicted data (IPD) using recent methods, where for a subset of the data, the outcomes have been predicted by an algorithm. Provides a wrapper function with specified defaults for the type of model and method to be used for estimation and inference. Further provides methods for tidying and summarizing results. Salerno et al., (2024) <doi:10.48550/arXiv.2410.09665>.

License MIT + file LICENSE

URL https://github.com/ipd-tools/ipd, https://ipd-tools.github.io/ipd/

BugReports https://github.com/ipd-tools/ipd/issues

Depends R (>= 3.5.0)

Imports caret, gam, generics, ranger, splines, stats, MASS, randomForest

Suggests knitr, patchwork, rmarkdown, spelling, testthat (>= 3.0.0), tidyverse

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.2

Language en-US

NeedsCompilation no

Author Stephen Salerno [aut, cre, cph] (<https://orcid.org/0000-0003-2763-0494>),
Jiacheng Miao [aut],
Awan Afiaz [aut],
Kentaro Hoffman [aut],
Anna Neufeld [aut],
Qiongshi Lu [aut],
Tyler H McCormick [aut],
Jeffrey T Leek [aut]

Repository CRAN

Date/Publication 2025-01-07 16:10:02 UTC

Contents

A *Calculation of the matrix A based on single dataset*

Description

A function for the calculation of the matrix A based on single dataset

Usage

```
A(X, Y, quant = NA, theta, method)
```

Arguments

X	Array or data.frame containing covariates
Y	Array or data.frame of outcomes
quant	quantile for quantile estimation
theta	parameter theta
method	indicates the method to be used for M-estimation. Options include "mean", "quantile", "ols", "logistic", and "poisson".

Value

matrix A based on single dataset

augment.ipd *Augment Data from an IPD Fit*

Description

Augments the data used for an IPD method/model fit with additional information about each observation.

Usage

```
## S3 method for class 'ipd'
augment(x, data = x$data_u, ...)
```

Arguments

x	An object of class ipd.
data	The data.frame used to fit the model. Default is x$data.
...	Additional arguments to be passed to the augment function.

Value

A data.frame containing the original data along with fitted values and residuals.

Examples

```
#-- Generate Example Data

set.seed(2023)

dat <- simdat(n = c(300, 300, 300), effect = 1, sigma_Y = 1)

head(dat)

formula <- Y - f ~ X1

#-- Fit IPD

fit <- ipd(formula, method = "postpi_analytic", model = "ols",

  data = dat, label = "set_label")

#-- Augment Data

augmented_df <- augment(fit)

head(augmented_df)
```

calc_lhat_glm *Estimate PPI++ Power Tuning Parameter*

Description

Calculates the optimal value of lhat for the prediction-powered confidence interval for GLMs.

Usage

```
calc_lhat_glm(
  grads,
  grads_hat,
  grads_hat_unlabeled,
  inv_hessian,
  coord = NULL,
  clip = FALSE
)
```

Arguments

- `grads` (matrix): n x p matrix gradient of the loss function with respect to the parameter evaluated at the labeled data.
- `grads_hat` (matrix): n x p matrix gradient of the loss function with respect to the model parameter evaluated using predictions on the labeled data.
- `grads_hat_unlabeled` (matrix): N x p matrix gradient of the loss function with respect to the parameter evaluated using predictions on the unlabeled data.
- `inv_hessian` (matrix): p x p matrix inverse of the Hessian of the loss function with respect to the parameter.
- `coord` (int, optional): Coordinate for which to optimize `lhat`. If `None`, it optimizes the total variance over all coordinates. Must be in (1, ..., d) where d is the shape of the estimand.
- `clip` (boolean, optional): Whether to clip the value of lhat to be non-negative. Defaults to `False`.

Value

(float): Optimal value of `lhat` in [0,1].

Examples

```
dat <- simdat(model = "ols")

form <- Y - f ~ X1

X_l <- model.matrix(form, data = dat[dat$set_label == "labeled",])
```

```
Y_l <- dat[dat$set_label == "labeled", all.vars(form)[1]] |> matrix(ncol = 1)

f_l <- dat[dat$set_label == "labeled", all.vars(form)[2]] |> matrix(ncol = 1)

X_u <- model.matrix(form, data = dat[dat$set_label == "unlabeled",])

f_u <- dat[dat$set_label == "unlabeled", all.vars(form)[2]] |> matrix(ncol = 1)

est <- ppi_plusplus_ols_est(X_l, Y_l, f_l, X_u, f_u)

stats <- ols_get_stats(est, X_l, Y_l, f_l, X_u, f_u)

calc_lhat_glm(stats$grads, stats$grads_hat, stats$grads_hat_unlabeled,

  stats$inv_hessian, coord = NULL, clip = FALSE)
```

`compute_cdf` *Empirical CDF of the Data*

Description

Computes the empirical CDF of the data.

Usage

```
compute_cdf(Y, grid, w = NULL)
```

Arguments

`Y`	(matrix): n x 1 matrix of observed data.
`grid`	(matrix): Grid of values to compute the CDF at.
`w`	(vector, optional): n-vector of sample weights.

Value

(list): Empirical CDF and its standard deviation at the specified grid points.

Examples

```
Y <- c(1, 2, 3, 4, 5)

grid <- seq(0, 6, by = 0.5)

compute_cdf(Y, grid)
```

compute_cdf_diff *Empirical CDF Difference*

Description

Computes the difference between the empirical CDFs of the data and the predictions.

Usage

```
compute_cdf_diff(Y, f, grid, w = NULL)
```

Arguments

`Y`	(matrix): n x 1 matrix of observed data.
`f`	(matrix): n x 1 matrix of predictions.
`grid`	(matrix): Grid of values to compute the CDF at.
`w`	(vector, optional): n-vector of sample weights.

Value

(list): Difference between the empirical CDFs of the data and the predictions and its standard deviation at the specified grid points.

Examples

```
Y <- c(1, 2, 3, 4, 5)

f <- c(1.1, 2.2, 3.3, 4.4, 5.5)

grid <- seq(0, 6, by = 0.5)

compute_cdf_diff(Y, f, grid)
```

est_ini *Initial estimation*

Description

`est_ini` function for initial estimation

Usage

```
est_ini(X, Y, quant = NA, method)
```

Arguments

X	Array or data.frame containing covariates
Y	Array or data.frame of outcomes
quant	quantile for quantile estimation
method	indicates the method to be used for M-estimation. Options include "mean", "quantile", "ols", "logistic", and "poisson".

Value

initial estimatior

---

glance.ipd *Glance at an IPD Fit*

Description

Glances at the IPD method/model fit, returning a one-row summary.

Usage

```
## S3 method for class 'ipd'
glance(x, ...)
```

Arguments

x	An object of class ipd.
...	Additional arguments to be passed to the glance function.

Value

A one-row data frame summarizing the IPD method/model fit.

Examples

```
#-- Generate Example Data

set.seed(2023)

dat <- simdat(n = c(300, 300, 300), effect = 1, sigma_Y = 1)

head(dat)

formula <- Y - f ~ X1

#-- Fit IPD

fit <- ipd(formula, method = "postpi_analytic", model = "ols",
```

```
  data = dat, label = "set_label")

#-- Glance Output

glance(fit)
```

ipd *Inference on Predicted Data (ipd)*

Description

The main wrapper function to conduct ipd using various methods and models, and returns a list of fitted model components.

Usage

```
ipd(
  formula,
  method,
  model,
  data,
  label = NULL,
  unlabeled_data = NULL,
  seed = NULL,
  intercept = TRUE,
  alpha = 0.05,
  alternative = "two-sided",
  n_t = Inf,
  na_action = "na.fail",
  ...
)
```

Arguments

formula	An object of class `formula`: a symbolic description of the model to be fitted. Must be of the form `Y - f ~ X`, where `Y` is the name of the column corresponding to the observed outcome in the labeled data, `f` is the name of the column corresponding to the predicted outcome in both labeled and unlabeled data, and `X` corresponds to the features of interest (i.e., `X = X1 + ... + Xp`). See **1. Formula** in the **Details** below for more information.
method	The IPD method to be used for fitting the model. Must be one of `"postpi_analytic"`, `"postpi_boot"`, `"ppi"`, `"ppi_plusplus"`, or `"pspa"`. See **3. Method** in the **Details** below for more information.
model	The type of downstream inferential model to be fitted, or the parameter being estimated. Must be one of `"mean"`, `"quantile"`, `"ols"`, `"logistic"`, or `"poisson"`. See **4. Model** in the **Details** below for more information.

data — A `data.frame` containing the variables in the model, either a stacked data frame with a specific column identifying the labeled versus unlabeled observations (`label`), or only the labeled data set. Must contain columns for the observed outcomes (`Y`), the predicted outcomes (`f`), and the features (`X`) needed to specify the `formula`. See **2. Data** in the **Details** below for more information.

label — A `string`, `int`, or `logical` specifying the column in the data that distinguishes between the labeled and unlabeled observations. See the `Details` section for more information. If NULL, `unlabeled_data` must be specified. See **2. Data** in the **Details** below for more information.

unlabeled_data — (optional) A `data.frame` of unlabeled data. If NULL, `label` must be specified. Specifying both the `label` and `unlabeled_data` arguments will result in an error message. If specified, must contain columns for the predicted outcomes (`f`), and the features (`X`) needed to specify the `formula`. See **2. Data** in the **Details** below for more information.

seed — (optional) An `integer` seed for random number generation.

intercept — `Logical`. Should an intercept be included in the model? Default is `TRUE`.

alpha — The significance level for confidence intervals. Default is `0.05`.

alternative — A string specifying the alternative hypothesis. Must be one of `"two-sided"`, `"less"`, or `"greater"`.

n_t — (integer, optional) Size of the dataset used to train the prediction function (necessary for the `"postpi_analytic"` and `"postpi_boot"` methods if `n_t < nrow(X_l)`. Defaults to `Inf`.

na_action — (string, optional) How missing covariate data should be handled. Currently `"na.fail"` and `"na.omit"` are accommodated. Defaults to `"na.fail"`.

... — Additional arguments to be passed to the fitting function. See the `Details` section for more information. See **5. Auxiliary Arguments** and **6. Other Arguments** in the **Details** below for more information.

Details

1. Formula:

The `ipd` function uses one formula argument that specifies both the calibrating model (e.g., PostPI "relationship model", PPI "rectifier" model) and the inferential model. These separate models will be created internally based on the specific `method` called.

2. Data:

The data can be specified in two ways:

1. Single data argument (`data`) containing a stacked `data.frame` and a label identifier (`label`).
2. Two data arguments, one for the labeled data (`data`) and one for the unlabeled data (`unlabeled_data`).

For option (1), provide one data argument (`data`) which contains a stacked `data.frame` with both the unlabeled and labeled data and a `label` argument that specifies the column identifying the labeled versus the unlabeled observations in the stacked `data.frame` (e.g., `label = "set_label"` if the column "set_label" in the stacked data denotes which set an observation belongs to).

NOTE: Labeled data identifiers can be:

String "l", "lab", "label", "labeled", "labelled", "tst", "test", "true"

Logical TRUE

Factor Non-reference category (i.e., binary 1)

Unlabeled data identifiers can be:

String "u", "unlab", "unlabeled", "unlabelled", "val", "validation", "false"

Logical FALSE

Factor Non-reference category (i.e., binary 0)

For option (2), provide separate data arguments for the labeled data set (`data`) and the unlabeled data set (`unlabeled_data`). If the second argument is provided, the function ignores the `label` identifier and assumes the data provided are not stacked.

NOTE: Not all columns in `data` or `unlabeled_data` may be used unless explicitly referenced in the `formula` argument or in the `label` argument (if the data are passed as one stacked data frame).

3. Method:

Use the `method` argument to specify the fitting method:

"postpi_analytic" Wang et al. (2020) Post-Prediction Inference (PostPI) Analytic Correction

"postpi_boot" Wang et al. (2020) Post-Prediction Inference (PostPI) Bootstrap Correction

"ppi" Angelopoulos et al. (2023) Prediction-Powered Inference (PPI)

"ppi_plusplus" Angelopoulos et al. (2023) PPI++

"pspa" Miao et al. (2023) Assumption-Lean and Data-Adaptive Post-Prediction Inference (PSPA)

4. Model:

Use the `model` argument to specify the type of downstream inferential model or parameter to be estimated:

"mean" Mean value of a continuous outcome

"quantile" `q`th quantile of a continuous outcome

"ols" Linear regression coefficients for a continuous outcome

"logistic" Logistic regression coefficients for a binary outcome

"poisson" Poisson regression coefficients for a count outcome

The `ipd` wrapper function will concatenate the `method` and `model` arguments to identify the required helper function, following the naming convention "method_model".

5. Auxiliary Arguments:

The wrapper function will take method-specific auxiliary arguments (e.g., `q` for the quantile estimation models) and pass them to the helper function through the "..." with specified defaults for simplicity.

6. Other Arguments:

All other arguments that relate to all methods (e.g., alpha, ci.type), or other method-specific arguments, will have defaults.

Value

a summary of model output.

A list containing the fitted model components:

coefficients Estimated coefficients of the model

se Standard errors of the estimated coefficients

ci Confidence intervals for the estimated coefficients

formula The formula used to fit the ipd model.

data The data frame used for model fitting.

method The method used for model fitting.

model The type of model fitted.

intercept Logical. Indicates if an intercept was included in the model.

fit Fitted model object containing estimated coefficients, standard errors, confidence intervals, and additional method-specific output.

... Additional output specific to the method used.

Examples

```
#-- Generate Example Data

set.seed(12345)

dat <- simdat(n = c(300, 300, 300), effect = 1, sigma_Y = 1)

head(dat)

formula <- Y - f ~ X1

#-- PostPI Analytic Correction (Wang et al., 2020)

ipd(formula, method = "postpi_analytic", model = "ols",

    data = dat, label = "set_label")

#-- PostPI Bootstrap Correction (Wang et al., 2020)

nboot <- 200

ipd(formula, method = "postpi_boot", model = "ols",

    data = dat, label = "set_label", nboot = nboot)

#-- PPI (Angelopoulos et al., 2023)

ipd(formula, method = "ppi", model = "ols",

    data = dat, label = "set_label")
```

```
#-- PPI++ (Angelopoulos et al., 2023)

ipd(formula, method = "ppi_plusplus", model = "ols",

    data = dat, label = "set_label")

#-- PSPA (Miao et al., 2023)

ipd(formula, method = "pspa", model = "ols",

    data = dat, label = "set_label")
```

`link_grad` *Gradient of the link function*

Description

`link_grad` function for gradient of the link function

Usage

```
link_grad(t, method)
```

Arguments

`t`	t
`method`	indicates the method to be used for M-estimation. Options include "mean", "quantile", "ols", "logistic", and "poisson".

Value

gradient of the link function

`link_Hessian` *Hessians of the link function*

Description

`link_Hessian` function for Hessians of the link function

Usage

```
link_Hessian(t, method)
```

Arguments

`t`	t
`method`	indicates the method to be used for M-estimation. Options include "mean", "quantile", "ols", "logistic", and "poisson".

Value

Hessians of the link function

`log1pexp` *Log1p Exponential*

Description

Computes the natural logarithm of 1 plus the exponential of the input, to handle large inputs.

Usage

```
log1pexp(x)
```

Arguments

`x`	(vector): A numeric vector of inputs.

Value

(vector): A numeric vector where each element is the result of log(1 + exp(x)).

Examples

```
x <- c(-1, 0, 1, 10, 100)

log1pexp(x)
```

logistic_get_stats *Logistic Regression Gradient and Hessian*

Description

Computes the statistics needed for the logstic regression-based prediction-powered inference.

Usage

```
logistic_get_stats(
  est,
  X_l,
  Y_l,
  f_l,
  X_u,
  f_u,
  w_l = NULL,
  w_u = NULL,
  use_u = TRUE
)
```

Arguments

`est`	(vector): Point estimates of the coefficients.
`X_l`	(matrix): Covariates for the labeled data set.
`Y_l`	(vector): Labels for the labeled data set.
`f_l`	(vector): Predictions for the labeled data set.
`X_u`	(matrix): Covariates for the unlabeled data set.
`f_u`	(vector): Predictions for the unlabeled data set.
`w_l`	(vector, optional): Sample weights for the labeled data set.
`w_u`	(vector, optional): Sample weights for the unlabeled data set.
`use_u`	(bool, optional): Whether to use the unlabeled data set.

Value

(list): A list containing the following:

grads (matrix): n x p matrix gradient of the loss function with respect to the coefficients.

grads_hat (matrix): n x p matrix gradient of the loss function with respect to the coefficients, evaluated using the labeled predictions.

grads_hat_unlabeled (matrix): N x p matrix gradient of the loss function with respect to the coefficients, evaluated using the unlabeled predictions.

inv_hessian (matrix): p x p matrix inverse Hessian of the loss function with respect to the coefficients.

Examples

```
dat <- simdat(model = "logistic")

form <- Y - f ~ X1

X_l <- model.matrix(form, data = dat[dat$set_label == "labeled",])

Y_l <- dat[dat$set_label == "labeled", all.vars(form)[1]] |> matrix(ncol = 1)

f_l <- dat[dat$set_label == "labeled", all.vars(form)[2]] |> matrix(ncol = 1)

X_u <- model.matrix(form, data = dat[dat$set_label == "unlabeled",])

f_u <- dat[dat$set_label == "unlabeled", all.vars(form)[2]] |> matrix(ncol = 1)

est <- ppi_plusplus_logistic_est(X_l, Y_l, f_l, X_u, f_u)

stats <- logistic_get_stats(est, X_l, Y_l, f_l, X_u, f_u)
```

---

mean_psi *Sample expectation of psi*

Description

`mean_psi` function for sample expectation of psi

Usage

```
mean_psi(X, Y, theta, quant = NA, method)
```

Arguments

`X`	Array or data.frame containing covariates
`Y`	Array or data.frame of outcomes
`theta`	parameter theta
`quant`	quantile for quantile estimation
`method`	indicates the method to be used for M-estimation. Options include "mean", "quantile", "ols", "logistic", and "poisson".

Value

sample expectation of psi

mean_psi_pop *Sample expectation of PSPA psi*

Description

mean_psi_pop function for sample expectation of PSPA psi

Usage

```
mean_psi_pop(
  X_lab,
  X_unlab,
  Y_lab,
  Yhat_lab,
  Yhat_unlab,
  w,
  theta,
  quant = NA,
  method
)
```

Arguments

X_lab	Array or data.frame containing observed covariates in labeled data.
X_unlab	Array or data.frame containing observed or predicted covariates in unlabeled data.
Y_lab	Array or data.frame of observed outcomes in labeled data.
Yhat_lab	Array or data.frame of predicted outcomes in labeled data.
Yhat_unlab	Array or data.frame of predicted outcomes in unlabeled data.
w	weights vector PSPA linear regression (d-dimensional, where d equals the number of covariates).
theta	parameter theta
quant	quantile for quantile estimation
method	indicates the method to be used for M-estimation. Options include "mean", "quantile", "ols", "logistic", and "poisson".

Value

sample expectation of PSPA psi

ols — *Ordinary Least Squares*

Description

Computes the ordinary least squares coefficients.

Usage

```
ols(X, Y, return_se = FALSE)
```

Arguments

`X`	(matrix): n x p matrix of covariates.
`Y`	(vector): p-vector of outcome values.
`return_se`	(bool, optional): Whether to return the standard errors of the coefficients.

Value

(list): A list containing the following:

theta (vector): p-vector of ordinary least squares estimates of the coefficients.

se (vector): If return_se == TRUE, return the p-vector of standard errors of the coefficients.

Examples

```
n <- 1000

X <- rnorm(n, 1, 1)

Y <- X + rnorm(n, 0, 1)

ols(X, Y, return_se = TRUE)
```

ols_get_stats — *OLS Gradient and Hessian*

Description

Computes the statistics needed for the OLS-based prediction-powered inference.

Usage

```
ols_get_stats(
  est,
  X_l,
  Y_l,
  f_l,
  X_u,
  f_u,
  w_l = NULL,
  w_u = NULL,
  use_u = TRUE
)
```

Arguments

`est`	(vector): Point estimates of the coefficients.
`X_l`	(matrix): Covariates for the labeled data set.
`Y_l`	(vector): Labels for the labeled data set.
`f_l`	(vector): Predictions for the labeled data set.
`X_u`	(matrix): Covariates for the unlabeled data set.
`f_u`	(vector): Predictions for the unlabeled data set.
`w_l`	(vector, optional): Sample weights for the labeled data set.
`w_u`	(vector, optional): Sample weights for the unlabeled data set.
`use_u`	(boolean, optional): Whether to use the unlabeled data set.

Value

(list): A list containing the following:

grads (matrix): n x p matrix gradient of the loss function with respect to the coefficients.

grads_hat (matrix): n x p matrix gradient of the loss function with respect to the coefficients, evaluated using the labeled predictions.

grads_hat_unlabeled (matrix): N x p matrix gradient of the loss function with respect to the coefficients, evaluated using the unlabeled predictions.

inv_hessian (matrix): p x p matrix inverse Hessian of the loss function with respect to the coefficients.

Examples

```
dat <- simdat(model = "ols")

form <- Y - f ~ X1

X_l <- model.matrix(form, data = dat[dat$set_label == "labeled",])

Y_l <- dat[dat$set_label == "labeled", all.vars(form)[1]] |> matrix(ncol = 1)
```

```
f_l <- dat[dat$set_label == "labeled", all.vars(form)[2]] |> matrix(ncol = 1)

X_u <- model.matrix(form, data = dat[dat$set_label == "unlabeled",])

f_u <- dat[dat$set_label == "unlabeled", all.vars(form)[2]] |> matrix(ncol = 1)

est <- ppi_plusplus_ols_est(X_l, Y_l, f_l, X_u, f_u)

stats <- ols_get_stats(est, X_l, Y_l, f_l, X_u, f_u, use_u = TRUE)
```

`optim_est` *One-step update for obtaining estimator*

Description

`optim_est` function for One-step update for obtaining estimator

Usage

```
optim_est(
  X_lab,
  X_unlab,
  Y_lab,
  Yhat_lab,
  Yhat_unlab,
  w,
  theta,
  quant = NA,
  method
)
```

Arguments

`X_lab`	Array or data.frame containing observed covariates in labeled data.
`X_unlab`	Array or data.frame containing observed or predicted covariates in unlabeled data.
`Y_lab`	Array or data.frame of observed outcomes in labeled data.
`Yhat_lab`	Array or data.frame of predicted outcomes in labeled data.
`Yhat_unlab`	Array or data.frame of predicted outcomes in unlabeled data.
`w`	weights vector PSPA linear regression (d-dimensional, where d equals the number of covariates).
`theta`	parameter theta
`quant`	quantile for quantile estimation
`method`	indicates the method to be used for M-estimation. Options include "mean", "quantile", "ols", "logistic", and "poisson".

Value

estimator

`optim_weights` *One-step update for obtaining the weight vector*

Description

`optim_weights` function for One-step update for obtaining estimator

Usage

```
optim_weights(
  X_lab,
  X_unlab,
  Y_lab,
  Yhat_lab,
  Yhat_unlab,
  w,
  theta,
  quant = NA,
  method
)
```

Arguments

`X_lab`	Array or data.frame containing observed covariates in labeled data.
`X_unlab`	Array or data.frame containing observed or predicted covariates in unlabeled data.
`Y_lab`	Array or data.frame of observed outcomes in labeled data.
`Yhat_lab`	Array or data.frame of predicted outcomes in labeled data.
`Yhat_unlab`	Array or data.frame of predicted outcomes in unlabeled data.
`w`	weights vector PSPA linear regression (d-dimensional, where d equals the number of covariates).
`theta`	parameter theta
`quant`	quantile for quantile estimation
`method`	indicates the method to be used for M-estimation. Options include "mean", "quantile", "ols", "logistic", and "poisson".

Value

weights

postpi_analytic_ols *PostPI OLS (Analytic Correction)*

Description

Helper function for PostPI OLS estimation (analytic correction)

Usage

```
postpi_analytic_ols(X_l, Y_l, f_l, X_u, f_u, scale_se = TRUE, n_t = Inf)
```

Arguments

X_l	(matrix): n x p matrix of covariates in the labeled data.
Y_l	(vector): n-vector of labeled outcomes.
f_l	(vector): n-vector of predictions in the labeled data.
X_u	(matrix): N x p matrix of covariates in the unlabeled data.
f_u	(vector): N-vector of predictions in the unlabeled data.
scale_se	(boolean): Logical argument to scale relationship model error variance. Defaults to TRUE; FALSE option is retained for posterity.
n_t	(integer, optional) Size of the dataset used to train the prediction function (necessary if `n_t` < `nrow(X_l)`. Defaults to `Inf`.

Details

Methods for correcting inference based on outcomes predicted by machine learning (Wang et al., 2020) https://www.pnas.org/doi/abs/10.1073/pnas.2001238117

Value

A list of outputs: estimate of the inference model parameters and corresponding standard error estimate.

Examples

```
dat <- simdat(model = "ols")

form <- Y - f ~ X1

X_l <- model.matrix(form, data = dat[dat$set_label == "labeled",])

Y_l <- dat[dat$set_label == "labeled", all.vars(form)[1]] |> matrix(ncol = 1)

f_l <- dat[dat$set_label == "labeled", all.vars(form)[2]] |> matrix(ncol = 1)

X_u <- model.matrix(form, data = dat[dat$set_label == "unlabeled",])
```

```
f_u <- dat[dat$set_label == "unlabeled", all.vars(form)[2]] |> matrix(ncol = 1)

postpi_analytic_ols(X_l, Y_l, f_l, X_u, f_u)
```

---

postpi_boot_logistic *PostPI Logistic Regression (Bootstrap Correction)*

Description

Helper function for PostPI logistic regression (bootstrap correction)

Usage

```
postpi_boot_logistic(
  X_l,
  Y_l,
  f_l,
  X_u,
  f_u,
  nboot = 100,
  se_type = "par",
  seed = NULL
)
```

Arguments

X_l	(matrix): n x p matrix of covariates in the labeled data.
Y_l	(vector): n-vector of labeled outcomes.
f_l	(vector): n-vector of predictions in the labeled data.
X_u	(matrix): N x p matrix of covariates in the unlabeled data.
f_u	(vector): N-vector of predictions in the unlabeled data.
nboot	(integer): Number of bootstrap samples. Defaults to 100.
se_type	(string): Which method to calculate the standard errors. Options include "par" (parametric) or "npar" (nonparametric). Defaults to "par".
seed	(optional) An `integer` seed for random number generation.

Details

Methods for correcting inference based on outcomes predicted by machine learning (Wang et al., 2020) https://www.pnas.org/doi/abs/10.1073/pnas.2001238117

Value

A list of outputs: estimate of inference model parameters and corresponding standard error based on both parametric and non-parametric bootstrap methods.

Examples

```
dat <- simdat(model = "logistic")

form <- Y - f ~ X1

X_l <- model.matrix(form, data = dat[dat$set_label == "labeled",])

Y_l <- dat[dat$set_label == "labeled", all.vars(form)[1]] |> matrix(ncol = 1)

f_l <- dat[dat$set_label == "labeled", all.vars(form)[2]] |> matrix(ncol = 1)

X_u <- model.matrix(form, data = dat[dat$set_label == "unlabeled",])

f_u <- dat[dat$set_label == "unlabeled", all.vars(form)[2]] |> matrix(ncol = 1)

postpi_boot_logistic(X_l, Y_l, f_l, X_u, f_u, nboot = 200)
```

---

postpi_boot_ols *PostPI OLS (Bootstrap Correction)*

Description

Helper function for PostPI OLS estimation (bootstrap correction)

Usage

```
postpi_boot_ols(
  X_l,
  Y_l,
  f_l,
  X_u,
  f_u,
  nboot = 100,
  se_type = "par",
  rel_func = "lm",
  scale_se = TRUE,
  n_t = Inf,
  seed = NULL
)
```

Arguments

X_l	(matrix): n x p matrix of covariates in the labeled data.
Y_l	(vector): n-vector of labeled outcomes.
f_l	(vector): n-vector of predictions in the labeled data.
X_u	(matrix): N x p matrix of covariates in the unlabeled data.

`f_u`	(vector): N-vector of predictions in the unlabeled data.
`nboot`	(integer): Number of bootstrap samples. Defaults to 100.
`se_type`	(string): Which method to calculate the standard errors. Options include "par" (parametric) or "npar" (nonparametric). Defaults to "par".
`rel_func`	(string): Method for fitting the relationship model. Options include "lm" (linear model), "rf" (random forest), and "gam" (generalized additive model). Defaults to "lm".
`scale_se`	(boolean): Logical argument to scale relationship model error variance. Defaults to TRUE; FALSE option is retained for posterity.
`n_t`	(integer, optional) Size of the dataset used to train the prediction function (necessary if `n_t` < `nrow(X_l)`. Defaults to `Inf`.
`seed`	(optional) An `integer` seed for random number generation.

Details

Methods for correcting inference based on outcomes predicted by machine learning (Wang et al., 2020) https://www.pnas.org/doi/abs/10.1073/pnas.2001238117

Value

A list of outputs: estimate of inference model parameters and corresponding standard error based on both parametric and non-parametric bootstrap methods.

Examples

```
dat <- simdat(model = "ols")

form <- Y - f ~ X1

X_l <- model.matrix(form, data = dat[dat$set_label == "labeled",])

Y_l <- dat[dat$set_label == "labeled", all.vars(form)[1]] |> matrix(ncol = 1)

f_l <- dat[dat$set_label == "labeled", all.vars(form)[2]] |> matrix(ncol = 1)

X_u <- model.matrix(form, data = dat[dat$set_label == "unlabeled",])

f_u <- dat[dat$set_label == "unlabeled", all.vars(form)[2]] |> matrix(ncol = 1)

postpi_boot_ols(X_l, Y_l, f_l, X_u, f_u, nboot = 200)
```

`ppi_logistic` *PPI Logistic Regression*

Description

Helper function for PPI logistic regression

Usage

```
ppi_logistic(X_l, Y_l, f_l, X_u, f_u, opts = NULL)
```

Arguments

`X_l`	(matrix): n x p matrix of covariates in the labeled data.
`Y_l`	(vector): n-vector of labeled outcomes.
`f_l`	(vector): n-vector of predictions in the labeled data.
`X_u`	(matrix): N x p matrix of covariates in the unlabeled data.
`f_u`	(vector): N-vector of predictions in the unlabeled data.
`opts`	(list, optional): Options to pass to the optimizer. See ?optim for details.

Details

Prediction Powered Inference (Angelopoulos et al., 2023) https://www.science.org/doi/10.1126/science.adi6000

Value

(list): A list containing the following:

est (vector): vector of PPI logistic regression coefficient estimates.

se (vector): vector of standard errors of the coefficients.

rectifier_est (vector): vector of the rectifier logistic regression coefficient estimates.

var_u (matrix): covariance matrix for the gradients in the unlabeled data.

var_l (matrix): covariance matrix for the gradients in the labeled data.

grads (matrix): matrix of gradients for the labeled data.

grads_hat_unlabeled (matrix): matrix of predicted gradients for the unlabeled data.

grads_hat (matrix): matrix of predicted gradients for the labeled data.

inv_hessian (matrix): inverse Hessian matrix.

Examples

```
dat <- simdat(model = "logistic")

form <- Y - f ~ X1

X_l <- model.matrix(form, data = dat[dat$set_label == "labeled",])

Y_l <- dat[dat$set_label == "labeled", all.vars(form)[1]] |> matrix(ncol = 1)

f_l <- dat[dat$set_label == "labeled", all.vars(form)[2]] |> matrix(ncol = 1)

X_u <- model.matrix(form, data = dat[dat$set_label == "unlabeled",])

f_u <- dat[dat$set_label == "unlabeled", all.vars(form)[2]] |> matrix(ncol = 1)

ppi_logistic(X_l, Y_l, f_l, X_u, f_u)
```

---

ppi_mean *PPI Mean Estimation*

Description

Helper function for PPI mean estimation

Usage

```
ppi_mean(Y_l, f_l, f_u, alpha = 0.05, alternative = "two-sided")
```

Arguments

`Y_l`	(vector): n-vector of labeled outcomes.
`f_l`	(vector): n-vector of predictions in the labeled data.
`f_u`	(vector): N-vector of predictions in the unlabeled data.
`alpha`	(scalar): type I error rate for hypothesis testing - values in (0, 1); defaults to 0.05.
`alternative`	(string): Alternative hypothesis. Must be one of `"two-sided"`, `"less"`, or `"greater"`.

Details

Prediction Powered Inference (Angelopoulos et al., 2023) https://www.science.org/doi/10.1126/science.adi6000

Value

tuple: Lower and upper bounds of the prediction-powered confidence interval for the mean.

Examples

```
dat <- simdat(model = "mean")

form <- Y - f ~ 1

Y_l <- dat[dat$set_label == "labeled",   all.vars(form)[1]] |> matrix(ncol = 1)

f_l <- dat[dat$set_label == "labeled",   all.vars(form)[2]] |> matrix(ncol = 1)

f_u <- dat[dat$set_label == "unlabeled", all.vars(form)[2]] |> matrix(ncol = 1)

ppi_mean(Y_l, f_l, f_u)
```

---

ppi_ols *PPI OLS*

Description

Helper function for prediction-powered inference for OLS estimation

Usage

```
ppi_ols(X_l, Y_l, f_l, X_u, f_u, w_l = NULL, w_u = NULL)
```

Arguments

X_l	(matrix): n x p matrix of covariates in the labeled data.
Y_l	(vector): n-vector of labeled outcomes.
f_l	(vector): n-vector of predictions in the labeled data.
X_u	(matrix): N x p matrix of covariates in the unlabeled data.
f_u	(vector): N-vector of predictions in the unlabeled data.
w_l	(ndarray, optional): Sample weights for the labeled data set. Defaults to a vector of ones.
w_u	(ndarray, optional): Sample weights for the unlabeled data set. Defaults to a vector of ones.

Details

Prediction Powered Inference (Angelopoulos et al., 2023) https://www.science.org/doi/10.1126/science.adi6000

Value

(list): A list containing the following:

est (vector): vector of PPI OLS regression coefficient estimates.

se (vector): vector of standard errors of the coefficients.

rectifier_est (vector): vector of the rectifier OLS regression coefficient estimates.

Examples

```
dat <- simdat()

form <- Y - f ~ X1

X_l <- model.matrix(form, data = dat[dat$set_label == "labeled",])

Y_l <- dat[dat$set_label == "labeled", all.vars(form)[1]] |> matrix(ncol = 1)

f_l <- dat[dat$set_label == "labeled", all.vars(form)[2]] |> matrix(ncol = 1)

X_u <- model.matrix(form, data = dat[dat$set_label == "unlabeled",])

f_u <- dat[dat$set_label == "unlabeled", all.vars(form)[2]] |> matrix(ncol = 1)

ppi_ols(X_l, Y_l, f_l, X_u, f_u)
```

---

ppi_plusplus_logistic *PPI++ Logistic Regression*

Description

Helper function for PPI++ logistic regression

Usage

```
ppi_plusplus_logistic(
  X_l,
  Y_l,
  f_l,
  X_u,
  f_u,
  lhat = NULL,
  coord = NULL,
  opts = NULL,
  w_l = NULL,
  w_u = NULL
)
```

Arguments

`X_l`	(matrix): n x p matrix of covariates in the labeled data.
`Y_l`	(vector): n-vector of labeled outcomes.
`f_l`	(vector): n-vector of predictions in the labeled data.
`X_u`	(matrix): N x p matrix of covariates in the unlabeled data.
`f_u`	(vector): N-vector of predictions in the unlabeled data.
`lhat`	(float, optional): Power-tuning parameter (see https://arxiv.org/abs/2311.01453). The default value, `NULL`, will estimate the optimal value from the data. Setting `lhat = 1` recovers PPI with no power tuning, and setting `lhat = 0` recovers the classical point estimate.
`coord`	(int, optional): Coordinate for which to optimize `lhat = 1`. If `NULL`, it optimizes the total variance over all coordinates. Must be in (1, ..., d) where d is the dimension of the estimand.
`opts`	(list, optional): Options to pass to the optimizer. See ?optim for details.
`w_l`	(ndarray, optional): Sample weights for the labeled data set. Defaults to a vector of ones.
`w_u`	(ndarray, optional): Sample weights for the unlabeled data set. Defaults to a vector of ones.

Details

PPI++: Efficient Prediction Powered Inference (Angelopoulos et al., 2023) https://arxiv.org/abs/2311.01453

Value

(list): A list containing the following:

est (vector): vector of PPI++ logistic regression coefficient estimates.

se (vector): vector of standard errors of the coefficients.

lambda (float): estimated power-tuning parameter.

rectifier_est (vector): vector of the rectifier logistic regression coefficient estimates.

var_u (matrix): covariance matrix for the gradients in the unlabeled data.

var_l (matrix): covariance matrix for the gradients in the labeled data.

grads (matrix): matrix of gradients for the labeled data.

grads_hat_unlabeled (matrix): matrix of predicted gradients for the unlabeled data.

grads_hat (matrix): matrix of predicted gradients for the labeled data.

inv_hessian (matrix): inverse Hessian matrix.

Examples

```
dat <- simdat(model = "logistic")

form <- Y - f ~ X1

X_l <- model.matrix(form, data = dat[dat$set_label == "labeled",])

Y_l <- dat[dat$set_label == "labeled", all.vars(form)[1]] |> matrix(ncol = 1)

f_l <- dat[dat$set_label == "labeled", all.vars(form)[2]] |> matrix(ncol = 1)

X_u <- model.matrix(form, data = dat[dat$set_label == "unlabeled",])

f_u <- dat[dat$set_label == "unlabeled", all.vars(form)[2]] |> matrix(ncol = 1)

ppi_plusplus_logistic(X_l, Y_l, f_l, X_u, f_u)
```

`ppi_plusplus_logistic_est` *PPI++ Logistic Regression (Point Estimate)*

Description

Helper function for PPI++ logistic regression (point estimate)

Usage

```
ppi_plusplus_logistic_est(
  X_l,
  Y_l,
  f_l,
  X_u,
  f_u,
  lhat = NULL,
  coord = NULL,
  opts = NULL,
  w_l = NULL,
  w_u = NULL
)
```

Arguments

X_l	(matrix): n x p matrix of covariates in the labeled data.
Y_l	(vector): n-vector of labeled outcomes.
f_l	(vector): n-vector of predictions in the labeled data.
X_u	(matrix): N x p matrix of covariates in the unlabeled data.

f_u	(vector): N-vector of predictions in the unlabeled data.
lhat	(float, optional): Power-tuning parameter (see https://arxiv.org/abs/2311.01453). The default value, NULL, will estimate the optimal value from the data. Setting lhat = 1 recovers PPI with no power tuning, and setting lhat = 0 recovers the classical point estimate.
coord	(int, optional): Coordinate for which to optimize lhat = 1. If NULL, it optimizes the total variance over all coordinates. Must be in (1, ..., d) where d is the dimension of the estimand.
opts	(list, optional): Options to pass to the optimizer. See ?optim for details.
w_l	(ndarray, optional): Sample weights for the labeled data set. Defaults to a vector of ones.
w_u	(ndarray, optional): Sample weights for the unlabeled data set. Defaults to a vector of ones.

Details

PPI++: Efficient Prediction Powered Inference (Angelopoulos et al., 2023) https://arxiv.org/abs/2311.01453`

Value

(vector): vector of prediction-powered point estimates of the logistic regression coefficients.

Examples

```
dat <- simdat(model = "logistic")

form <- Y - f ~ X1

X_l <- model.matrix(form, data = dat[dat$set_label == "labeled",])

Y_l <- dat[dat$set_label == "labeled", all.vars(form)[1]] |> matrix(ncol = 1)

f_l <- dat[dat$set_label == "labeled", all.vars(form)[2]] |> matrix(ncol = 1)

X_u <- model.matrix(form, data = dat[dat$set_label == "unlabeled",])

f_u <- dat[dat$set_label == "unlabeled", all.vars(form)[2]] |> matrix(ncol = 1)

ppi_plusplus_logistic_est(X_l, Y_l, f_l, X_u, f_u)
```

ppi_plusplus_mean *PPI++ Mean Estimation*

Description

Helper function for PPI++ mean estimation

Usage

```
ppi_plusplus_mean(
  Y_l,
  f_l,
  f_u,
  alpha = 0.05,
  alternative = "two-sided",
  lhat = NULL,
  coord = NULL,
  w_l = NULL,
  w_u = NULL
)
```

Arguments

`Y_l`	(vector): n-vector of labeled outcomes.
`f_l`	(vector): n-vector of predictions in the labeled data.
`f_u`	(vector): N-vector of predictions in the unlabeled data.
`alpha`	(scalar): type I error rate for hypothesis testing - values in (0, 1); defaults to 0.05.
`alternative`	(string): Alternative hypothesis. Must be one of `"two-sided"`, `"less"`, or `"greater"`.
`lhat`	(float, optional): Power-tuning parameter (see https://arxiv.org/abs/2311.01453). The default value, `NULL`, will estimate the optimal value from the data. Setting `lhat = 1` recovers PPI with no power tuning, and setting `lhat = 0` recovers the classical point estimate.
`coord`	(int, optional): Coordinate for which to optimize `lhat = 1`. If `NULL`, it optimizes the total variance over all coordinates. Must be in (1, ..., d) where d is the dimension of the estimand.
`w_l`	(ndarray, optional): Sample weights for the labeled data set. Defaults to a vector of ones.
`w_u`	(ndarray, optional): Sample weights for the unlabeled data set. Defaults to a vector of ones.

Details

PPI++: Efficient Prediction Powered Inference (Angelopoulos et al., 2023) https://arxiv.org/abs/2311.01453`

Value

tuple: Lower and upper bounds of the prediction-powered confidence interval for the mean.

Examples

```
dat <- simdat(model = "mean")

form <- Y - f ~ 1

Y_l <- dat[dat$set_label == "labeled",   all.vars(form)[1]] |> matrix(ncol = 1)

f_l <- dat[dat$set_label == "labeled",   all.vars(form)[2]] |> matrix(ncol = 1)

f_u <- dat[dat$set_label == "unlabeled", all.vars(form)[2]] |> matrix(ncol = 1)

ppi_plusplus_mean(Y_l, f_l, f_u)
```

`ppi_plusplus_mean_est` *PPI++ Mean Estimation (Point Estimate)*

Description

Helper function for PPI++ mean estimation (point estimate)

Usage

```
ppi_plusplus_mean_est(
  Y_l,
  f_l,
  f_u,
  lhat = NULL,
  coord = NULL,
  w_l = NULL,
  w_u = NULL
)
```

Arguments

`Y_l`	(vector): n-vector of labeled outcomes.
`f_l`	(vector): n-vector of predictions in the labeled data.
`f_u`	(vector): N-vector of predictions in the unlabeled data.
`lhat`	(float, optional): Power-tuning parameter (see https://arxiv.org/abs/2311.01453). The default value, `NULL`, will estimate the optimal value from the data. Setting `lhat = 1` recovers PPI with no power tuning, and setting `lhat = 0` recovers the classical point estimate.

coord	(int, optional): Coordinate for which to optimize lhat = 1. If NULL, it optimizes the total variance over all coordinates. Must be in (1, ..., d) where d is the dimension of the estimand.
w_l	(ndarray, optional): Sample weights for the labeled data set. Defaults to a vector of ones.
w_u	(ndarray, optional): Sample weights for the unlabeled data set. Defaults to a vector of ones.

Details

PPI++: Efficient Prediction Powered Inference (Angelopoulos et al., 2023) https://arxiv.org/abs/2311.01453

Value

float or ndarray: Prediction-powered point estimate of the mean.

Examples

```
dat <- simdat(model = "mean")

form <- Y - f ~ 1

Y_l <- dat[dat$set_label == "labeled",   all.vars(form)[1]] |> matrix(ncol = 1)

f_l <- dat[dat$set_label == "labeled",   all.vars(form)[2]] |> matrix(ncol = 1)

f_u <- dat[dat$set_label == "unlabeled", all.vars(form)[2]] |> matrix(ncol = 1)

ppi_plusplus_mean_est(Y_l, f_l, f_u)
```

---

ppi_plusplus_ols   *PPI++ OLS*

Description

Helper function for PPI++ OLS estimation

Usage

```
ppi_plusplus_ols(
  X_l,
  Y_l,
  f_l,
  X_u,
  f_u,
  lhat = NULL,
```

```
  coord = NULL,
  w_l = NULL,
  w_u = NULL
)
```

Arguments

- `X_l` (matrix): n x p matrix of covariates in the labeled data.
- `Y_l` (vector): n-vector of labeled outcomes.
- `f_l` (vector): n-vector of predictions in the labeled data.
- `X_u` (matrix): N x p matrix of covariates in the unlabeled data.
- `f_u` (vector): N-vector of predictions in the unlabeled data.
- `lhat` (float, optional): Power-tuning parameter (see https://arxiv.org/abs/2311.01453). The default value, `NULL`, will estimate the optimal value from the data. Setting `lhat = 1` recovers PPI with no power tuning, and setting `lhat = 0` recovers the classical point estimate.
- `coord` (int, optional): Coordinate for which to optimize `lhat = 1`. If `NULL`, it optimizes the total variance over all coordinates. Must be in (1, ..., d) where d is the dimension of the estimand.
- `w_l` (ndarray, optional): Sample weights for the labeled data set. Defaults to a vector of ones.
- `w_u` (ndarray, optional): Sample weights for the unlabeled data set. Defaults to a vector of ones.

Details

PPI++: Efficient Prediction Powered Inference (Angelopoulos et al., 2023) https://arxiv.org/abs/2311.01453`

Value

(list): A list containing the following:

est (vector): vector of PPI++ OLS regression coefficient estimates.

se (vector): vector of standard errors of the coefficients.

lambda (float): estimated power-tuning parameter.

rectifier_est (vector): vector of the rectifier OLS regression coefficient estimates.

var_u (matrix): covariance matrix for the gradients in the unlabeled data.

var_l (matrix): covariance matrix for the gradients in the labeled data.

grads (matrix): matrix of gradients for the labeled data.

grads_hat_unlabeled (matrix): matrix of predicted gradients for the unlabeled data.

grads_hat (matrix): matrix of predicted gradients for the labeled data.

inv_hessian (matrix): inverse Hessian matrix.

Examples

```
dat <- simdat(model = "ols")

form <- Y - f ~ X1

X_l <- model.matrix(form, data = dat[dat$set_label == "labeled",])

Y_l <- dat[dat$set_label == "labeled", all.vars(form)[1]] |> matrix(ncol = 1)

f_l <- dat[dat$set_label == "labeled", all.vars(form)[2]] |> matrix(ncol = 1)

X_u <- model.matrix(form, data = dat[dat$set_label == "unlabeled",])

f_u <- dat[dat$set_label == "unlabeled", all.vars(form)[2]] |> matrix(ncol = 1)

ppi_plusplus_ols(X_l, Y_l, f_l, X_u, f_u)
```

---

ppi_plusplus_ols_est *PPI++ OLS (Point Estimate)*

Description

Helper function for PPI++ OLS estimation (point estimate)

Usage

```
ppi_plusplus_ols_est(
  X_l,
  Y_l,
  f_l,
  X_u,
  f_u,
  lhat = NULL,
  coord = NULL,
  w_l = NULL,
  w_u = NULL
)
```

Arguments

X_l	(matrix): n x p matrix of covariates in the labeled data.
Y_l	(vector): n-vector of labeled outcomes.
f_l	(vector): n-vector of predictions in the labeled data.
X_u	(matrix): N x p matrix of covariates in the unlabeled data.
f_u	(vector): N-vector of predictions in the unlabeled data.

lhat (float, optional): Power-tuning parameter (see https://arxiv.org/abs/2311.01453). The default value, NULL, will estimate the optimal value from the data. Setting lhat = 1 recovers PPI with no power tuning, and setting lhat = 0 recovers the classical point estimate.

coord (int, optional): Coordinate for which to optimize lhat = 1. If NULL, it optimizes the total variance over all coordinates. Must be in (1, ..., d) where d is the dimension of the estimand.

w_l (ndarray, optional): Sample weights for the labeled data set. Defaults to a vector of ones.

w_u (ndarray, optional): Sample weights for the unlabeled data set. Defaults to a vector of ones.

Details

PPI++: Efficient Prediction Powered Inference (Angelopoulos et al., 2023) https://arxiv.org/abs/2311.01453

Value

(vector): vector of prediction-powered point estimates of the OLS coefficients.

Examples

```
dat <- simdat(model = "ols")

form <- Y - f ~ X1

X_l <- model.matrix(form, data = dat[dat$set_label == "labeled",])

Y_l <- dat[dat$set_label == "labeled", all.vars(form)[1]] |> matrix(ncol = 1)

f_l <- dat[dat$set_label == "labeled", all.vars(form)[2]] |> matrix(ncol = 1)

X_u <- model.matrix(form, data = dat[dat$set_label == "unlabeled",])

f_u <- dat[dat$set_label == "unlabeled", all.vars(form)[2]] |> matrix(ncol = 1)

ppi_plusplus_ols_est(X_l, Y_l, f_l, X_u, f_u)
```

---

ppi_plusplus_quantile *PPI++ Quantile Estimation*

Description

Helper function for PPI++ quantile estimation

Usage

```
ppi_plusplus_quantile(
  Y_l,
  f_l,
  f_u,
  q,
  alpha = 0.05,
  exact_grid = FALSE,
  w_l = NULL,
  w_u = NULL
)
```

Arguments

Y_l	(vector): n-vector of labeled outcomes.
f_l	(vector): n-vector of predictions in the labeled data.
f_u	(vector): N-vector of predictions in the unlabeled data.
q	(float): Quantile to estimate. Must be in the range (0, 1).
alpha	(scalar): type I error rate for hypothesis testing - values in (0, 1); defaults to 0.05.
exact_grid	(bool, optional): Whether to compute the exact solution (TRUE) or an approximate solution based on a linearly spaced grid of 5000 values (FALSE).
w_l	(ndarray, optional): Sample weights for the labeled data set. Defaults to a vector of ones.
w_u	(ndarray, optional): Sample weights for the unlabeled data set. Defaults to a vector of ones.

Details

PPI++: Efficient Prediction Powered Inference (Angelopoulos et al., 2023) https://arxiv.org/abs/2311.01453

Value

tuple: Lower and upper bounds of the prediction-powered confidence interval for the quantile.

Examples

```
dat <- simdat(model = "quantile")

form <- Y - f ~ X1

Y_l <- dat[dat$set_label == "labeled", all.vars(form)[1]] |> matrix(ncol = 1)

f_l <- dat[dat$set_label == "labeled", all.vars(form)[2]] |> matrix(ncol = 1)

f_u <- dat[dat$set_label == "unlabeled", all.vars(form)[2]] |> matrix(ncol = 1)
```

```
ppi_plusplus_quantile(Y_l, f_l, f_u, q = 0.5)
```

---

ppi_plusplus_quantile_est

PPI++ Quantile Estimation (Point Estimate)

Description

Helper function for PPI++ quantile estimation (point estimate)

Usage

```
ppi_plusplus_quantile_est(
  Y_l,
  f_l,
  f_u,
  q,
  exact_grid = FALSE,
  w_l = NULL,
  w_u = NULL
)
```

Arguments

Y_l	(vector): n-vector of labeled outcomes.
f_l	(vector): n-vector of predictions in the labeled data.
f_u	(vector): N-vector of predictions in the unlabeled data.
q	(float): Quantile to estimate. Must be in the range (0, 1).
exact_grid	(bool, optional): Whether to compute the exact solution (TRUE) or an approximate solution based on a linearly spaced grid of 5000 values (FALSE).
w_l	(ndarray, optional): Sample weights for the labeled data set. Defaults to a vector of ones.
w_u	(ndarray, optional): Sample weights for the unlabeled data set. Defaults to a vector of ones.

Details

PPI++: Efficient Prediction Powered Inference (Angelopoulos et al., 2023) https://arxiv.org/abs/2311.01453`

Value

(float): Prediction-powered point estimate of the quantile.

Examples

```
dat <- simdat(model = "quantile")

form <- Y - f ~ 1

Y_l <- dat[dat$set_label == "labeled", all.vars(form)[1]] |> matrix(ncol = 1)

f_l <- dat[dat$set_label == "labeled", all.vars(form)[2]] |> matrix(ncol = 1)

f_u <- dat[dat$set_label == "unlabeled", all.vars(form)[2]] |> matrix(ncol = 1)

ppi_plusplus_quantile_est(Y_l, f_l, f_u, q = 0.5)
```

---

ppi_quantile *PPI Quantile Estimation*

Description

Helper function for PPI quantile estimation

Usage

```
ppi_quantile(Y_l, f_l, f_u, q, alpha = 0.05, exact_grid = FALSE)
```

Arguments

Y_l	(vector): n-vector of labeled outcomes.
f_l	(vector): n-vector of predictions in the labeled data.
f_u	(vector): N-vector of predictions in the unlabeled data.
q	(float): Quantile to estimate. Must be in the range (0, 1).
alpha	(scalar): type I error rate for hypothesis testing - values in (0, 1); defaults to 0.05.
exact_grid	(bool, optional): Whether to compute the exact solution (TRUE) or an approximate solution based on a linearly spaced grid of 5000 values (FALSE).

Details

Prediction Powered Inference (Angelopoulos et al., 2023) https://www.science.org/doi/10.1126/science.adi6000

Value

tuple: Lower and upper bounds of the prediction-powered confidence interval for the quantile.

Examples

```
dat <- simdat(model = "quantile")

form <- Y - f ~ X1

Y_l <- dat[dat$set_label == "labeled", all.vars(form)[1]] |> matrix(ncol = 1)

f_l <- dat[dat$set_label == "labeled", all.vars(form)[2]] |> matrix(ncol = 1)

f_u <- dat[dat$set_label == "unlabeled", all.vars(form)[2]] |> matrix(ncol = 1)

ppi_quantile(Y_l, f_l, f_u, q = 0.5)
```

---

print.ipd *Print IPD Fit*

Description

Prints a brief summary of the IPD method/model combination.

Usage

```
## S3 method for class 'ipd'
print(x, ...)
```

Arguments

x	An object of class ipd.
...	Additional arguments to be passed to the print function.

Value

The input x, invisibly.

Examples

```
#-- Generate Example Data

set.seed(2023)

dat <- simdat(n = c(300, 300, 300), effect = 1, sigma_Y = 1)

head(dat)

formula <- Y - f ~ X1

#-- Fit IPD
```

```
fit <- ipd(formula, method = "postpi_analytic", model = "ols",

  data = dat, label = "set_label")

#-- Print Output

print(fit)
```

---

print.summary.ipd *Print Summary of IPD Fit*

Description

Prints a detailed summary of the IPD method/model combination.

Usage

```
## S3 method for class 'summary.ipd'
print(x, ...)
```

Arguments

x	An object of class summary.ipd.
...	Additional arguments to be passed to the print function.

Value

The input x, invisibly.

Examples

```
#-- Generate Example Data

set.seed(2023)

dat <- simdat(n = c(300, 300, 300), effect = 1, sigma_Y = 1)

head(dat)

formula <- Y - f ~ X1

#-- Fit IPD

fit <- ipd(formula, method = "postpi_analytic", model = "ols",

  data = dat, label = "set_label")

#-- Summarize Output
```

```
summ_fit <- summary(fit)

print(summ_fit)
```

`psi` *Estimating equation*

Description

`psi` function for estimating equation

Usage

```
psi(X, Y, theta, quant = NA, method)
```

Arguments

`X`	Array or data.frame containing covariates
`Y`	Array or data.frame of outcomes
`theta`	parameter theta
`quant`	quantile for quantile estimation
`method`	indicates the method to be used for M-estimation. Options include "mean", "quantile", "ols", "logistic", and "poisson".

Value

esimating equation

`pspa_logistic` *PSPA Logistic Regression*

Description

Helper function for PSPA logistic regression

Usage

```
pspa_logistic(X_l, Y_l, f_l, X_u, f_u, weights = NA, alpha = 0.05)
```

Arguments

`X_l`	(matrix): n x p matrix of covariates in the labeled data.
`Y_l`	(vector): n-vector of binary labeled outcomes.
`f_l`	(vector): n-vector of binary predictions in the labeled data.
`X_u`	(matrix): N x p matrix of covariates in the unlabeled data.
`f_u`	(vector): N-vector of binary predictions in the unlabeled data.
`weights`	(array): p-dimensional array of weights vector for variance reduction. PSPA will estimate the weights if not specified.
`alpha`	(scalar): type I error rate for hypothesis testing - values in (0, 1); defaults to 0.05

Details

Post-prediction adaptive inference (Miao et al. 2023) https://arxiv.org/abs/2311.14220

Value

A list of outputs: estimate of inference model parameters and corresponding standard error.

Examples

```
dat <- simdat(model = "logistic")

form <- Y - f ~ X1

X_l <- model.matrix(form, data = dat[dat$set_label == "labeled",])

Y_l <- dat[dat$set_label == "labeled", all.vars(form)[1]] |> matrix(ncol = 1)

f_l <- dat[dat$set_label == "labeled", all.vars(form)[2]] |> matrix(ncol = 1)

X_u <- model.matrix(form, data = dat[dat$set_label == "unlabeled",])

f_u <- dat[dat$set_label == "unlabeled", all.vars(form)[2]] |> matrix(ncol = 1)

pspa_logistic(X_l, Y_l, f_l, X_u, f_u)
```

`pspa_mean` *PSPA Mean Estimation*

Description

Helper function for PSPA mean estimation

Usage

```
pspa_mean(Y_l, f_l, f_u, weights = NA, alpha = 0.05)
```

Arguments

- `Y_l` (vector): n-vector of labeled outcomes.
- `f_l` (vector): n-vector of predictions in the labeled data.
- `f_u` (vector): N-vector of predictions in the unlabeled data.
- `weights` (array): 1-dimensional array of weights vector for variance reduction. PSPA will estimate the weights if not specified.
- `alpha` (scalar): type I error rate for hypothesis testing - values in (0, 1); defaults to 0.05.

Details

Post-prediction adaptive inference (Miao et al., 2023) https://arxiv.org/abs/2311.14220

Value

A list of outputs: estimate of inference model parameters and corresponding standard error.

Examples

```
dat <- simdat(model = "mean")

form <- Y - f ~ 1

Y_l <- dat[dat$set_label == "labeled",   all.vars(form)[1]] |> matrix(ncol = 1)

f_l <- dat[dat$set_label == "labeled",   all.vars(form)[2]] |> matrix(ncol = 1)

f_u <- dat[dat$set_label == "unlabeled", all.vars(form)[2]] |> matrix(ncol = 1)

pspa_mean(Y_l, f_l, f_u)
```

`pspa_ols` *PSPA OLS Estimation*

Description

Helper function for PSPA OLS for linear regression

Usage

```
pspa_ols(X_l, Y_l, f_l, X_u, f_u, weights = NA, alpha = 0.05)
```

Arguments

`X_l`	(matrix): n x p matrix of covariates in the labeled data.
`Y_l`	(vector): n-vector of labeled outcomes.
`f_l`	(vector): n-vector of predictions in the labeled data.
`X_u`	(matrix): N x p matrix of covariates in the unlabeled data.
`f_u`	(vector): N-vector of predictions in the unlabeled data.
`weights`	(array): p-dimensional array of weights vector for variance reduction. PSPA will estimate the weights if not specified.
`alpha`	(scalar): type I error rate for hypothesis testing - values in (0, 1); defaults to 0.05.

Details

Post-prediction adaptive inference (Miao et al. 2023) https://arxiv.org/abs/2311.14220

Value

A list of outputs: estimate of inference model parameters and corresponding standard error.

Examples

```
dat <- simdat(model = "ols")

form <- Y - f ~ X1

X_l <- model.matrix(form, data = dat[dat$set_label == "labeled",])

Y_l <- dat[dat$set_label == "labeled", all.vars(form)[1]] |> matrix(ncol = 1)

f_l <- dat[dat$set_label == "labeled", all.vars(form)[2]] |> matrix(ncol = 1)

X_u <- model.matrix(form, data = dat[dat$set_label == "unlabeled",])

f_u <- dat[dat$set_label == "unlabeled", all.vars(form)[2]] |> matrix(ncol = 1)

pspa_ols(X_l, Y_l, f_l, X_u, f_u)
```

`pspa_poisson` *PSPA Poisson Regression*

Description

Helper function for PSPA Poisson regression

Usage

```
pspa_poisson(X_l, Y_l, f_l, X_u, f_u, weights = NA, alpha = 0.05)
```

Arguments

X_l	(matrix): n x p matrix of covariates in the labeled data.
Y_l	(vector): n-vector of count labeled outcomes.
f_l	(vector): n-vector of binary predictions in the labeled data.
X_u	(matrix): N x p matrix of covariates in the unlabeled data.
f_u	(vector): N-vector of binary predictions in the unlabeled data.
weights	(array): p-dimensional array of weights vector for variance reduction. PSPA will estimate the weights if not specified.
alpha	(scalar): type I error rate for hypothesis testing - values in (0, 1); defaults to 0.05

Details

Post-prediction adaptive inference (Miao et al. 2023) https://arxiv.org/abs/2311.14220

Value

A list of outputs: estimate of inference model parameters and corresponding standard error.

Examples

```
dat <- simdat(model = "poisson")

form <- Y - f ~ X1

X_l <- model.matrix(form, data = dat[dat$set_label == "labeled",])

Y_l <- dat[dat$set_label == "labeled", all.vars(form)[1]] |> matrix(ncol = 1)

f_l <- dat[dat$set_label == "labeled", all.vars(form)[2]] |> matrix(ncol = 1)

X_u <- model.matrix(form, data = dat[dat$set_label == "unlabeled",])

f_u <- dat[dat$set_label == "unlabeled", all.vars(form)[2]] |> matrix(ncol = 1)

pspa_poisson(X_l, Y_l, f_l, X_u, f_u)
```

`pspa_quantile` *PSPA Quantile Estimation*

Description

Helper function for PSPA quantile estimation

Usage

```
pspa_quantile(Y_l, f_l, f_u, q, weights = NA, alpha = 0.05)
```

Arguments

`Y_l`	(vector): n-vector of labeled outcomes.
`f_l`	(vector): n-vector of predictions in the labeled data.
`f_u`	(vector): N-vector of predictions in the unlabeled data.
`q`	(float): Quantile to estimate. Must be in the range (0, 1).
`weights`	(array): 1-dimensional array of weights vector for variance reduction. PSPA will estimate the weights if not specified.
`alpha`	(scalar): type I error rate for hypothesis testing - values in (0, 1); defaults to 0.05.

Details

Post-prediction adaptive inference (Miao et al. 2023) https://arxiv.org/abs/2311.14220

Value

A list of outputs: estimate of inference model parameters and corresponding standard error.

Examples

```
dat <- simdat(model = "quantile")

form <- Y - f ~ 1

Y_l <- dat[dat$set_label == "labeled",   all.vars(form)[1]] |> matrix(ncol = 1)

f_l <- dat[dat$set_label == "labeled",   all.vars(form)[2]] |> matrix(ncol = 1)

f_u <- dat[dat$set_label == "unlabeled", all.vars(form)[2]] |> matrix(ncol = 1)

pspa_quantile(Y_l, f_l, f_u, q = 0.5)
```

pspa_y *PSPA M-Estimation for ML-predicted labels*

Description

pspa_y function conducts post-prediction M-Estimation.

Usage

```
pspa_y(
  X_lab = NA,
  X_unlab = NA,
  Y_lab,
  Yhat_lab,
  Yhat_unlab,
  alpha = 0.05,
  weights = NA,
  quant = NA,
  intercept = FALSE,
  method
)
```

Arguments

X_lab	Array or data.frame containing observed covariates in labeled data.
X_unlab	Array or data.frame containing observed or predicted covariates in unlabeled data.
Y_lab	Array or data.frame of observed outcomes in labeled data.
Yhat_lab	Array or data.frame of predicted outcomes in labeled data.
Yhat_unlab	Array or data.frame of predicted outcomes in unlabeled data.
alpha	Specifies the confidence level as 1 - alpha for confidence intervals.
weights	weights vector PSPA linear regression (d-dimensional, where d equals the number of covariates).
quant	quantile for quantile estimation
intercept	Boolean indicating if the input covariates' data contains the intercept (TRUE if the input data contains)
method	indicates the method to be used for M-estimation. Options include "mean", "quantile", "ols", "logistic", and "poisson".

Value

A summary table presenting point estimates, standard error, confidence intervals (1 - alpha), P-values, and weights.

Examples

```
data <- sim_data_y()

X_lab <- data$X_lab

X_unlab <- data$X_unlab

Y_lab <- data$Y_lab

Yhat_lab <- data$Yhat_lab

Yhat_unlab <- data$Yhat_unlab

pspa_y(X_lab = X_lab, X_unlab = X_unlab,

 Y_lab = Y_lab, Yhat_lab = Yhat_lab, Yhat_unlab = Yhat_unlab,

 alpha = 0.05, method = "ols")
```

`rectified_cdf` *Rectified CDF*

Description

Computes the rectified CDF of the data.

Usage

```
rectified_cdf(Y_l, f_l, f_u, grid, w_l = NULL, w_u = NULL)
```

Arguments

`Y_l`	(vector): Gold-standard labels.
`f_l`	(vector): Predictions corresponding to the gold-standard labels.
`f_u`	(vector): Predictions corresponding to the unlabeled data.
`grid`	(vector): Grid of values to compute the CDF at.
`w_l`	(vector, optional): Sample weights for the labeled data set.
`w_u`	(vector, optional): Sample weights for the unlabeled data set.

Value

(vector): Rectified CDF of the data at the specified grid points.

Examples

```
Y_l <- c(1, 2, 3, 4, 5)

f_l <- c(1.1, 2.2, 3.3, 4.4, 5.5)

f_u <- c(1.2, 2.3, 3.4)

grid <- seq(0, 6, by = 0.5)

rectified_cdf(Y_l, f_l, f_u, grid)
```

---

rectified_p_value *Rectified P-Value*

Description

Computes a rectified p-value.

Usage

```
rectified_p_value(
  rectifier,
  rectifier_std,
  imputed_mean,
  imputed_std,
  null = 0,
  alternative = "two-sided"
)
```

Arguments

`rectifier`	(float or vector): Rectifier value.
`rectifier_std`	(float or vector): Rectifier standard deviation.
`imputed_mean`	(float or vector): Imputed mean.
`imputed_std`	(float or vector): Imputed standard deviation.
`null`	(float, optional): Value of the null hypothesis to be tested. Defaults to `0`.
`alternative`	(str, optional): Alternative hypothesis, either 'two-sided', 'larger' or 'smaller'.

Value

(float or vector): The rectified p-value.

Examples

```
rectifier <- 0.7

rectifier_std <- 0.5

imputed_mean <- 1.5

imputed_std <- 0.3

rectified_p_value(rectifier, rectifier_std, imputed_mean, imputed_std)
```

Sigma_cal *Variance-covariance matrix of the estimation equation*

Description

`Sigma_cal` function for variance-covariance matrix of the estimation equation

Usage

```
Sigma_cal(
  X_lab,
  X_unlab,
  Y_lab,
  Yhat_lab,
  Yhat_unlab,
  w,
  theta,
  quant = NA,
  method
)
```

Arguments

`X_lab`	Array or data.frame containing observed covariates in labeled data.
`X_unlab`	Array or data.frame containing observed or predicted covariates in unlabeled data.
`Y_lab`	Array or data.frame of observed outcomes in labeled data.
`Yhat_lab`	Array or data.frame of predicted outcomes in labeled data.
`Yhat_unlab`	Array or data.frame of predicted outcomes in unlabeled data.
`w`	weights vector PSPA linear regression (d-dimensional, where d equals the number of covariates).
`theta`	parameter theta
`quant`	quantile for quantile estimation
`method`	indicates the method to be used for M-estimation. Options include "mean", "quantile", "ols", "logistic", and "poisson".

Value

variance-covariance matrix of the estimation equation

---

simdat — *Data generation function for various underlying models*

Description

Data generation function for various underlying models

Usage

```
simdat(
  n = c(300, 300, 300),
  effect = 1,
  sigma_Y = 1,
  model = "ols",
  shift = 0,
  scale = 1
)
```

Arguments

`n`	Integer vector of size 3 indicating the sample sizes in the training, labeled, and unlabeled data sets, respectively
`effect`	Regression coefficient for the first variable of interest for inference. Defaults is 1.
`sigma_Y`	Residual variance for the generated outcome. Defaults is 1.
`model`	The type of model to be generated. Must be one of `"mean"`, `"quantile"`, `"ols"`, `"logistic"`, or `"poisson"`. Default is `"ols"`.
`shift`	Scalar shift of the predictions for continuous outcomes (i.e., "mean", "quantile", and "ols"). Defaults to 0.
`scale`	Scaling factor for the predictions for continuous outcomes (i.e., "mean", "quantile", and "ols"). Defaults to 1.

Details

The `simdat` function generates three datasets consisting of independent realizations of Y (for `model = "mean"` or `"quantile"`), or $\{Y, \boldsymbol{X}\}$ (for `model = "ols"`, `"logistic"`, or `"poisson"`): a *training* dataset of size n_t, a *labeled* dataset of size n_l, and an *unlabeled* dataset of size n_u. These sizes are specified by the argument `n`.

NOTE: In the *unlabeled* data subset, outcome data are still generated to facilitate a benchmark for comparison with an "oracle" model that uses the true $Y^{\mathcal{U}}$ values for estimation and inference.

Generating Data

For "mean" and "quantile", we simulate a continuous outcome, $Y \in \mathbb{R}$, with mean given by the effect argument and error variance given by the sigma_y argument.

For "ols", "logistic", or "poisson" models, predictor data, $\boldsymbol{X} \in \mathbb{R}^4$ are simulated such that the ith observation follows a standard multivariate normal distribution with a zero mean vector and identity covariance matrix:

$$\boldsymbol{X_i} = (X_{i1}, X_{i2}, X_{i3}, X_{i4}) \sim \mathcal{N}_4(\boldsymbol{0}, \boldsymbol{I}).$$

For "ols", a continuous outcome $Y \in \mathbb{R}$ is simulated to depend on X_1 through a linear term with the effect size specified by the effect argument, while the other predictors, $\boldsymbol{X} \setminus X_1$, have nonlinear effects:

$$Y_i = effect \times Z_{i1} + \frac{1}{2}Z_{i2}^2 + \frac{1}{3}Z_{i3}^3 + \frac{1}{4}Z_{i4}^2 + \varepsilon_y,$$

and $\varepsilon_y \sim \mathcal{N}(0, sigma_y)$, where the sigma_y argument specifies the error variance.

For "logistic", we simulate:

$$\Pr(Y_i = 1 \mid \boldsymbol{X}) = logit^{-1}(effect \times Z_{i1} + \frac{1}{2}Z_{i2}^2 + \frac{1}{3}Z_{i3}^3 + \frac{1}{4}Z_{i4}^2 + \varepsilon_y)$$

and generate:

$$Y_i \sim Bern[1, \Pr(Y_i = 1 \mid \boldsymbol{X})]$$

where $\varepsilon_y \sim \mathcal{N}(0, sigma\_y)$.

For "poisson", we simulate:

$$\lambda_Y = exp(effect \times Z_{i1} + \frac{1}{2}Z_{i2}^2 + \frac{1}{3}Z_{i3}^3 + \frac{1}{4}Z_{i4}^2 + \varepsilon_y)$$

and generate:

$$Y_i \sim Poisson(\lambda_Y)$$

Generating Predictions

To generate predicted outcomes for "mean" and "quantile", we simulate a continuous variable with mean given by the empirical mean of the training data and error variance given by the sigma_y argument.

For "ols", we fit a generalized additive model (GAM) on the simulated *training* dataset and calculate predictions for the *labeled* and *unlabeled* datasets as deterministic functions of $\boldsymbol{X}$. Specifically, we fit the following GAM:

$$Y^{\mathcal{T}} = s_0 + s_1(X_1^{\mathcal{T}}) + s_2(X_2^{\mathcal{T}}) + s_3(X_3^{\mathcal{T}}) + s_4(X_4^{\mathcal{T}}) + \varepsilon_p,$$

where $\mathcal{T}$ denotes the *training* dataset, s_0 is an intercept term, and $s_1(\cdot)$, $s_2(\cdot)$, $s_3(\cdot)$, and $s_4(\cdot)$ are smoothing spline functions for X_1, X_2, X_3, and X_4, respectively, with three target equivalent degrees of freedom. Residual error is modeled as ε_p.

Predictions for *labeled* and *unlabeled* datasets are calculated as:

$$f(\boldsymbol{X}^{\mathcal{L}\cup\mathcal{U}}) = \hat{s}_0 + \hat{s}_1(X_1^{\mathcal{L}\cup\mathcal{U}}) + \hat{s}_2(X_2^{\mathcal{L}\cup\mathcal{U}}) + \hat{s}_3(X_3^{\mathcal{L}\cup\mathcal{U}}) + \hat{s}_4(X_4^{\mathcal{L}\cup\mathcal{U}}),$$

where $\hat{s}_0, \hat{s}_1, \hat{s}_2, \hat{s}_3$, and $\hat{s}_4$ are estimates of s_0, s_1, s_2, s_3, and s_4, respectively.

NOTE: For continuous outcomes, we provide optional arguments `shift` and `scale` to further apply a location shift and scaling factor, respectively, to the predicted outcomes. These default to `shift = 0` and `scale = 1`, i.e., no location shift or scaling.

For `"logistic"`, we train k-nearest neighbors (k-NN) classifiers on the simulated *training* dataset for values of k ranging from 1 to 10. The optimal k is chosen via cross-validation, minimizing the misclassification error on the validation folds. Predictions for the *labeled* and *unlabeled* datasets are obtained by applying the k-NN classifier with the optimal k to $\boldsymbol{X}$.

Specifically, for each observation in the *labeled* and *unlabeled* datasets:

$$\hat{Y} = \text{argmax}_c \sum_{i \in \mathcal{N}_k} I(Y_i = c),$$

where $\mathcal{N}_k$ represents the set of k nearest neighbors in the training dataset, c indexes the possible classes (0 or 1), and $I(\cdot)$ is an indicator function.

For `"poisson"`, we fit a generalized linear model (GLM) with a log link function to the simulated *training* dataset. The model is of the form:

$$\log(\mu^{\mathcal{T}}) = \gamma_0 + \gamma_1 X_1^{\mathcal{T}} + \gamma_2 X_2^{\mathcal{T}} + \gamma_3 X_3^{\mathcal{T}} + \gamma_4 X_4^{\mathcal{T}},$$

where $\mu^{\mathcal{T}}$ is the expected count for the response variable in the *training* dataset, γ_0 is the intercept, and γ_1, γ_2, γ_3, and γ_4 are the regression coefficients for the predictors X_1, X_2, X_3, and X_4, respectively.

Predictions for the *labeled* and *unlabeled* datasets are calculated as:

$$\hat{\mu}^{\mathcal{L}\cup\mathcal{U}} = \exp(\hat{\gamma}_0 + \hat{\gamma}_1 X_1^{\mathcal{L}\cup\mathcal{U}} + \hat{\gamma}_2 X_2^{\mathcal{L}\cup\mathcal{U}} + \hat{\gamma}_3 X_3^{\mathcal{L}\cup\mathcal{U}} + \hat{\gamma}_4 X_4^{\mathcal{L}\cup\mathcal{U}}),$$

where $\hat{\gamma}_0$, $\hat{\gamma}_1$, $\hat{\gamma}_2$, $\hat{\gamma}_3$, and $\hat{\gamma}_4$ are the estimated coefficients.

Value

A data.frame containing n rows and columns corresponding to the labeled outcome (Y), the predicted outcome (f), a character variable (set_label) indicating which data set the observation belongs to (training, labeled, or unlabeled), and four independent, normally distributed predictors (X1, X2, X3, and X4), where applicable.

Examples

```
#-- Mean

dat_mean <- simdat(c(100, 100, 100), effect = 1, sigma_Y = 1,

  model = "mean")
```

```
head(dat_mean)

#-- Linear Regression

dat_ols <- simdat(c(100, 100, 100), effect = 1, sigma_Y = 1,

  model = "ols")

head(dat_ols)
```

---

sim_data_y *Simulate the data for testing the functions*

Description

`sim_data_y` for simulation with ML-predicted Y

Usage

```
sim_data_y(r = 0.9, binary = FALSE)
```

Arguments

`r`	imputation correlation
`binary`	simulate binary outcome or not

Value

simulated data

---

summary.ipd *Summarize IPD Fit*

Description

Produces a summary of the IPD method/model combination.

Usage

```
## S3 method for class 'ipd'
summary(object, ...)
```

Arguments

`object`	An object of class `ipd`.
`...`	Additional arguments to be passed to the summary function.

Value

A list containing:

coefficients Model coefficients and related statistics.

performance Performance metrics of the model fit.

... Additional summary information.

Examples

```
#-- Generate Example Data

set.seed(2023)

dat <- simdat(n = c(300, 300, 300), effect = 1, sigma_Y = 1)

head(dat)

formula <- Y - f ~ X1

#-- Fit IPD

fit <- ipd(formula, method = "postpi_analytic", model = "ols",

  data = dat, label = "set_label")

#-- Summarize Output

summ_fit <- summary(fit)

summ_fit
```

`tidy.ipd` *Tidy an IPD Fit*

Description

Tidies the IPD method/model fit into a data frame.

Usage

```
## S3 method for class 'ipd'
tidy(x, ...)
```

Arguments

`x`	An object of class `ipd`.
`...`	Additional arguments to be passed to the tidy function.

Value

A tidy data frame of the model's coefficients.

Examples

```
#-- Generate Example Data

set.seed(2023)

dat <- simdat(n = c(300, 300, 300), effect = 1, sigma_Y = 1)

head(dat)

formula <- Y - f ~ X1

#-- Fit IPD

fit <- ipd(formula, method = "postpi_analytic", model = "ols",

  data = dat, label = "set_label")

#-- Tidy Output

tidy(fit)
```

`wls` *Weighted Least Squares*

Description

Computes the weighted least squares estimate of the coefficients.

Usage

```
wls(X, Y, w = NULL, return_se = FALSE)
```

Arguments

X	(matrix): n x p matrix of covariates.
Y	(vector): p-vector of outcome values.
w	(vector, optional): n-vector of sample weights.
return_se	(bool, optional): Whether to return the standard errors of the coefficients.

Value

(list): A list containing the following:

theta (vector): p-vector of weighted least squares estimates of the coefficients.
se (vector): If return_se == TRUE, return the p-vector of standard errors of the coefficients.

Examples

```
n <- 1000

X <- rnorm(n, 1, 1)

w <- rep(1, n)

Y <- X + rnorm(n, 0, 1)

wls(X, Y, w = w, return_se = TRUE)
```

---

zconfint_generic *Normal Confidence Intervals*

Description

Calculates normal confidence intervals for a given alternative at a given significance level.

Usage

```
zconfint_generic(mean, std_mean, alpha, alternative)
```

Arguments

mean	(float): Estimated normal mean.
std_mean	(float): Estimated standard error of the mean.
alpha	(float): Significance level in [0,1]
alternative	(string): Alternative hypothesis, either 'two-sided', 'larger' or 'smaller'.

Value

(vector): Lower and upper (1 - alpha) * 100% confidence limits.

Examples

```
n <- 1000

Y <- rnorm(n, 1, 1)

se_Y <-  sd(Y) / sqrt(n)

zconfint_generic(Y, se_Y, alpha = 0.05, alternative = "two-sided")
```

zstat_generic — *Compute Z-Statistic and P-Value*

Description

Computes the z-statistic and the corresponding p-value for a given test.

Usage

```
zstat_generic(value1, value2, std_diff, alternative, diff = 0)
```

Arguments

`value1`	(numeric): The first value or sample mean.
`value2`	(numeric): The second value or sample mean.
`std_diff`	(numeric): The standard error of the difference between the two values.
`alternative`	(character): The alternative hypothesis. Can be one of "two-sided" (or "2-sided", "2s"), "larger" (or "l"), or "smaller" (or "s").
`diff`	(numeric, optional): The hypothesized difference between the two values. Default is 0.

Value

(list): A list containing the following:

zstat (numeric): The computed z-statistic.

pvalue (numeric): The corresponding p-value for the test.

Examples

```
value1 <- 1.5

value2 <- 1.0

std_diff <- 0.2

alternative <- "two-sided"

result <- zstat_generic(value1, value2, std_diff, alternative)
```

Index