Package ‘PSLB’

April 13, 2022

Version 0.1

Date 2022-03-31

Title Propensity Score Analysis with Local Balance

Maintainer Yan Li <fiona19832008@gmail.com>

Depends Matching,
CBPS,
mgcv,
Matrix,
osqp,
quadprog,
kernlab,
RSpectra,
MASS

Description Implements the propensity score analysis with local balance proposed by Yan Li and Liang Li. This two-step method nonparametrically estimate propensity score weights that achieve local covariate balance.

License `mit_license()`

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.2.9000

R topics documented:

`Fitted.strata.diff` *Calculate Local Covariate Balance in Pre-specified Local Neighborhoods*

Description

This function evaluates the global and local balance by standardized difference or non-standardized difference of covariates after inverse probability weighting (IPW) weighting. The IPW is used for ATE estimation. The global balance is the difference measures calculated in all the subjects. The local balance is the difference measures calculated in each pre-defined local neighborhood.

Usage

```
Fitted.strata.diff(X, Z, ps, weight = NULL, ej, std = "std.diff")
```

Arguments

`X`	The covariate matrix with the rows corresponding to the subjects/participants and the columns corresponding to the covariates.
`Z`	The binary treatment indicator. A vector with 2 unique numeric values in 0 = untreated and 1 = treated. Note each subject/participant must has a value of the treatment indicator, thus its length should be the same as the number of all subjects/participants.
`ps`	The propensity score used to determine the propensity score (PS) stratified sub-population in each local neighborhoods. A vector with numeric values greater than 0 and less than 1 with the legnth of the number of all subjects/participants.
`weight`	The IPW weight used for estimating average treatement effect (ATE). It is calculated as Z/ps + (1-Z)/(1-ps), where Z is the treatment indicator and ps is the propensity score. A vector with positive values and the legnth of the number of all subjects/participants. If weight = NULL, the IPW for ATE will be calculated using the input propensity score. The weight will be normalized to sum to 1 within treated/untreated group. weight = NULL by default.
`ej`	The matrix of the local neighborhoods, which contains two columns of positive values greater or equal to 0 and less or equal to 1. The rows of ej represent the neighborhoods. The first column is the start point of the local neighborhoods. The second column is the end point of the neighborhoods.
`std`	The difference measure that evaluating the covariate balance. The default measure "std.diff" is the standardized difference of covariates, which is the difference in weighted means of the two treatment groups, divided by the pooled weighted standard deviation expressed as a percentage. The measure "std.norm" is the difference in weighted means of the two treatment groups. "std" can only take values of "std.diff" or "std.norm".

Value

The matrix whose each column representing the difference measure for each covariates. The first row represents the global balance. The rest of the row represents the local balance corresponding to the local neighborhoods. The global or local balance are standardized difference of each covariate in the whole or propensity score stratified (PS-stratified) sub-populations, and expressed as a percentage.

References

Li, L. and Greene, T. (2013) A weighting analogue to pair matching in propensity score analysis. The International Journal of Biostatistics, 9, 215-234.

Examples

```
KS = Kang_Schafer_Simulation(n = 2000, seeds = 5050)
# Define local neighborhoods
ej = cbind(seq(0,0.7,0.1),seq(0.3,1,0.1))
print(ej)
# Misspecified propensity score model
require(CBPS)
Z = KS$Data[,2]
X = KS$Data[,7:10]
cbps.fit = CBPS(Z~X, ATT = 0, method = "exact")
# Global and local balance of CBPS estimated propensity score
Fitted.strata.diff(X = X, Z = Z, ps = cbps.fit$fitted.values,
                   weight = cbps.fit$weights, ej = ej, std = "std.diff")
# Global and local balance of the true propensity score
true.ps = KS$Data[,11]
Fitted.strata.diff(X = X, Z = Z, ps = true.ps, ej = ej, std = "std.diff")
```

`Gaussian_Kernel_feature` *Kernel PCA using Gaussian kernel*

Description

This function transforms the imput covariate space to the feature space using kernel PCA. The kernel matrix K is constructed by Gaussian kernel. The input is the original covariate matrix and the number of candidate sigma from the kernel function used. The function returns the list of the feature space and other parameters from the kernel PCA.

Usage

```
Gaussian_Kernel_feature(X, n_sigma = 20, k = 500)
```

Arguments

`X`	The covariate matrix with the rows corresponding to the subjects/participants and the columns corresponding to the covariates.
`n_sigma`	The number of bandwidth value of the Gaussian kernel function used to generate the feature space. See details below. Default to 20.
`k`	The number of top K eigen value calculated. Note that k should be smaller than the sample size. The minimum of k and the sample size will be used. Defaults to 500.

Details

Kernel PCA is a simple way of implemention of RKHS to extend certain parametric methods with flexible modeling. Kernel PCA can extend the input covariates to a feature space of non-linear transformations defined by a kernel function. The kernel function (Gaussian kernel is implemented here) defines an inner product in the feature space between each pair of subjects. The kernel matrix K is constructed by calculating a measure of similarity between any two subjects with the Gaussian kernel function. And different sigma, the bandwitdh of the Gaussian kernel, determines the set of functions in the transformed space. Then eigen decomposition is conducted on the kernel matrix. The eigen vector matrix is the transformed feature matrix. For computational efficiency, only the top k eigen values are calculated. The range of the bandwidth sigma is the 0.1 and 0.9 quantiles of the L_2 distances between samples. The set of sigma includes n_sigma values from the function input within this range that are equally spaced on the log-scale. For each sigma value, the number of the eigen vectors corresponding to 90%, 95%, and 99% variance explained by the eigen values are calculated.

Value

A list containing the following components:

- "features": A list of transformed feature space corresponding to each bandwidth sigma. Each elements of this list is a list whose elements are the feature space matrixes containing the number of the selected columns correspond to 90%, 95%, and 99% variance explained by the eigen values.
- "r": A matrix containing the number of the eigen vectors corresponding to 90%, 95%, and 99% variance explained by the eigen values for each bandwith sigma.
- "d": A list of eigen value corresponding to each bandwidth sigma.
- "sigma_seq": The value of the bandwith sigma from the Gaussian kernel function.

References

Chen, W. and Zhang, H. (2007) The condition of kernelizing an algorithm and an equivalence between kernel methods. In Iberian Conference on Pattern Recognition and Image Analysis, 228-245. Springer.

Examples

```
KS = Kang_Schafer_Simulation(n = 1000, seeds = 5050)
# Misspecified propensity score model
X = KS$Data[,7:10]
# Transform X with Gaussian kernel using 10 sigma
x.k = Gaussian_Kernel_feature(X = X, n_sigma = 10)
```

`Kang_Schafer_Simulation` *Kang and Schafer simulation*

Description

This function generate the simulation scenarios presented in Kang and Schafer (2007)

Usage

```
Kang_Schafer_Simulation(
  n,
  beta = c(-1, 0.5, -0.25, -0.1),
  alpha = c(210, 27.4, 13.7, 13.7, 13.7),
  mu = rep(0, 4),
  sd = diag(4),
  seeds
)
```

Arguments

`n`	The number of sample size.
`beta`	The coefficient of the true propensity score model. Default to c(-1,0.5,-0.25,-0.1).
`alpha`	The coefficient of the true outcome model. Defaults to c(210,27.4,13.7,13.7,13.7).
`mu`	The mean of the covariates Z in the true propensity score model. Default to rep(0,4)
`sd`	The variance of the covariates Z in the true propensity score model. Default to diag(4)
`seeds`	The seed number

Value

A list containing the following components:

- "Data": The simulated data matrix includes the outcome Y (1st column), the treatment assignment Tr (2nd column), the covariates Z in the true propensit score mode (3rd to 6th column), the observed covariates X (7th to 10th column), and the true propensity score PS (11th column).
- "Treat.effect": The mean of the outcome Y

References

Kang, J. D. and Shafer, J. L. (2007) Demystifying double robustness: A comparison of alternative strategies for estimating a population mean from incomplete data. Statistical Science, 22, 523-539.

`PS.ConvertFrom.Weight` *Convert the Inverse Probability Weight (IPW) for ATE to Propensity Score*

Description

This function converts the IPW weights from PSLB 2 to the corresponding propensity scores. When the subject i is in the untreated group, the converted propensity score is 1-(1/weight_i). When the subject i is in the treated group, converted propensity score is 1/weight_i.

Usage

```
PS.ConvertFrom.Weight(w, Z)
```

Arguments

`w`	The input IPW weights.
`Z`	The binary treatment indicator. A vector with 2 unique numeric values in 0 = untreated and 1 = treated.

Value

The vector contanining the converted propensity score.

`PSLB` *Propensity Score Analysis with Local Balance (PSLB) Estimation*

Description

PSLB nonparametrically estimates the propensity score weights that achieve covariate balance in both whole study population (global balance) and the propensity score stratified (PS-stratified) subpopulations (local balance). Therefore, the PSLB estimated propensity score are appropriate since they approximates the balancing score, when the propensity score model is subject to model misspecification. PSLB include two steps (PSLB 1 and PSLB 2). In the 1st step, PSLB 1 implements a flexible form of covariate balancing propensity score (CBPS), and finds the model with the best local balance among the model pool. In the 2nd step, PSLB 2 refines the score from PSLB 1 so that the estimated score approximately balances the covariates to a pre-specified level in the local neighborhoods (each neighborhood, a small inverval between 0 and 1, represents a sub-population whose estimated score falls in this inverval). Note that PSLB 2 can be used to improve the local balance of propensity score estimated by any method, such as CBPS (see the examples).

Usage

```
PSLB(
  X,
  Z,
  n_sigma = 20,
  ej,
  selectInX = "cov",
  k = 500,
  method = "exact",
  standardize = TRUE,
  criteria = "mean",
  p.hat = NULL,
  epsilon = list(input = NULL, e.min = -10, bylength = 0.2, a = 3, epsilon.div = NULL)
)
```

Arguments

`X`	The covariate matrix with the rows corresponding to the subjects/participants and the columns corresponding to the covariates. The covariate matrix X can not contain missing value. The function will stop if NA value is detected in X.
`Z`	The binary treatment indicator. A vector with 2 unique numeric values in 0 = untreated and 1 = treated. Its length should be the same as the number of all subjects/participants without missing value.

`n_sigma`	The number of bandwidth value of the Gaussian kernel function used to generate the feature space in PSLB 1. See details in the description of function `Gaussian_Kernel_feature` and `PSLB1`. Default to 20.
`ej`	The local neighborhoods for PSLB 1 and PSLB 2. It can be one matrix or a list of two matrixes containing values greater or equal to 0 and less or equal to 1. If ej is one matrix, PSLB 1 and PSLB 2 both use this matrix as their local neighborhoods. If ej is a list of matrixes with length of two, PSLB 1 uses the first element of the list and PSLB 2 uses the second element of the list as their local neighborhoods. The rows of the matrixes represent the local neighborhoods. There are two columns, where the first columns corresponding the start and the second column corresponding the end point of the local neighborhoods.
`selectInX`	The matrix used to evaluate the local balance when tune the model in PSLB 1. selectInX = "cov" by default, which evaluates the local covariate balance on the input covariate matrix X. If selectInX = "feature", the local covariate balance is evaluated on the feature space transformed by kernel PCA.
`k`	The number of top K eigen value calculated in PSLB 1. Note that k should be smaller than the sample size. The minimum of k and the sample size will be used. Defaults to 500. See details in the description of function `Gaussian_Kernel_feature` and `PSLB1`.
`method`	Choose "exact" to fit the justidentified CBPS model in PSLB 1; choose "over" to fit the overidentified CBPS model in PSLB 1. See details in the description of function `PSLB1`. Default is "exact".
`standardize`	Default is TRUE, which normalizes weights of PSLB 1 to sum to 1 within treated/untreated group. Set to FALSE to return IPW weights for ATE.
`criteria`	The statistics used in PSLB 1 to select the model with the optimized local balance. Choose "mean" to use the mean of the absolute S/D among all covariates and neighborhoods for the tuning parameters by T4LB; choose "max" to use the max of the absolute S/D among all covariates and neighborhoods for the tuning parameters by T4LB. Default is "mean".
`p.hat`	User supplied propensity score. When p.hat is provided, PSLB 1 will not be fitted. PSLB 2 uses the provided p.hat to determine the PS-stratified sub-populations. Then IPW weight is estimated such that the propensity score within each strata are adjusted to minimizing the local imbalance in that interval while holding the PS-stratified sub-populations unchanged from p.hat. When p.hat = NULL, the function fits the PSLB 1 and PSLB 2 subsequently as discussed in the description. The default is NULL.
`epsilon`	A list of factors for controlling the local covariate imbalance parameters • "input": The matrix of candidate local imbalance parameters of choice. If input = NULL, the imbalance parameters can be set by our default rules. However, the users can supply their own values. The "input" must be a matrix with rows corresponding to each set of candidate imbalance parameters and columns corresponding to the local neighborhoods. Therefor, its number of columns must equal to the number of local neighborhoods. • "e.min": The minimum candidate value of the imbalance parameters in log2 scale. Default is -10 • "bylength": The step size of the candidate imbalance parameters in log2 scale. Default is 0.2 • "a": The factor determines the maximum candidate value of the imbalance parameters. The largest candidate value is set to be "a" times of the estimated standard deviation of the weighted mean difference of covariates between the two treatment groups in each sub-population. The default value

of a is 3. When PSLB 2 is unsolvable, increasing "a" may result in a solution. However, this solution gives larger local balance compared to when smaller "a" is used.

- "epsilon.div": The vector groups the neighboring stratas who share the same value of epsilon. When use non-overlapping strata (local neighborhood), "epsilon.div" is unnecessary. When use overlapping strata (local neighborhood), "epsilon.div" must be provided. Each element of "epsilon.div" corresponds the number of neighboring strata. For example, epsilon.div = c(2,3,2) means that the same imbalance parameter (epsilon) is used in strata 1 and 2; strata 3, 4, and 5; or strata 6 and 7, respectively. If the user would like to have different imbalance parameter in each neighborhood, "epsilon.div" is a vector of 1s, whose length equals to the number of stratas. For example, if there is 5 local neighborhoods, epsilon.div = rep(1,5) will allow different epsilon value in each strata.

Value

A list containing the following components:

- "PSLB2": A list containing the following components:
  - "weight": The optimal weights estimated by PSLB 2. If the fitting of PSLB 2 is unsuccess, weight returns NULL value.
  - "local.sample.size": The matrix contains the sample size in total (n), in treated group (n1), and in untreated group (n0) for each local neighborhoods. Each row of this matrix represents each neighborhood. The columns corresponds to n, n1/n, n0 and n1.
  - "p.hat": The propensity score used to determine the PS-stratified sub-population in PSLB 2. If PSLB 1 is fitted, the output "p.hat" is the estimated propensity score from PSLB 1. If PSLB 1 is not fitted, the output "p.hat" is just the input "p.hat".
  - "balance": The matrix containing the global and local balance of the estimated weights from PSLB 2. The first row contains the absolute S/D of each covariates in the whole study population, which represents the global covariate balance. The rest rows contain the absolute S/D of each covariates in the PS-stratified sub-population corresponding to the loal neighborhoods of PSLB 2, which represent the local covariate balance. If the fitting of PSLB 2 is unsuccess, balance returns NULL value.
  - "epsilon": The selected local imbalance parameter for each local neighborhood.
  - "epsilon.all": All the candidate values of the local imbalance parameter epsilon. If use non-overlapping strata, "epsilon.all" is a list whose each element corresponds to each strata. If use overlapping strata, "epsilon.all" is a vector since all local neighborhoods share one set of candidate values of epsilon.
- "PSLB1": The output list of function `PSLB1`
- "status": An integer code. 0 indicates successful completion of PSLB 2 estimation. Possible error codes are 1 and 2. 1 indicates that PSLB 2 can not find a solution under the current covariate imbalance condition. 2 indicates that there is an error occured when fits the PSLB2.
- "message": A character string giving the fitting status of PSLB 2.
- "X": The input covariate matrix
- "treat": The treatment assignment vector used

References

Li, Y. and Li, L. Propensity score analysis with local balance. manuscript under preperation.

See Also

PSLB1() for the detailed description of PSLB 1.

Examples

```
KS = Kang_Schafer_Simulation(n = 1000, seeds = 371834)
# Misspecified propensity score model
X = KS$Data[,7:10]
Z = KS$Data[,2]
# Specify the local neighborhoods
ej1 = cbind(seq(0,0.8,0.2),seq(0.2,1,0.2)) # non-overlapping strata
ej2 = cbind(seq(0,0.6,0.1),seq(0.4,1,0.1)) # overlapping strata
ej = list(ej1, ej2)
print(ej1)
print(ej2)
# Fit the PSLB model with overlapping strata
PSLB.fit = PSLB(X = X, Z = Z, n_sigma = 10, ej = ej, p.hat = NULL, epsilon = list(epsilon.div = c(2,3,2)))
str(PSLB.fit)
# Global and local balance of PSLB 2 in ej2
round(PSLB.fit$PSLB2$balance,3)
# Global and local balance of PSLB 1 in ej2
round(abs(Fitted.strata.diff(X = X, Z = Z, ps = PSLB.fit$PSLB1$propensity.score, ej = ej2)),3)
# Fit the PSLB model with non-overlapping strata
PSLB.fit.ej1 = PSLB(X = X, Z = Z, n_sigma = 10, ej = ej1, p.hat = PSLB.fit$PSLB1$propensity.score, epsilon = l
# Global and local balance of PSLB 2 in ej1
round(PSLB.fit.ej1$PSLB2$balance, 3)
# Fit the CBPS model
cbps.fit = CBPS(Z~X, ATT = 0, method = "exact")
cbps.ps = cbps.fit$fitted.values
# Improve the local balance of the estimated propensity score from CBPS
cbps.pslb.fit = PSLB(X = X, Z = Z, ej = ej, p.hat = cbps.ps, epsilon = list(epsilon.div = c(2,3,2)))
round(cbps.pslb.fit$PSLB2$balance,3)
# Global and local imbalance of CBPS
cbps.local = abs(Fitted.strata.diff(X = X, Z = Z, ps = cbps.ps, ej = ej2))
round(cbps.local, 3)
```

PSLB1 *Propensity Score Analysis with Local Balance 1 (PSLB 1) Estimation*

Description

PSLB1 estimates propensity scores by implementing a flexible form of the covariate balancing propensity score (CBPS) using kernel PCA, and tunes parameters (the bandwidth of the Gaussian kernel and the number of PCs) by the tuning for local balance (T4LB) algorithm, which finds the model with the best local balance (minimum absolute standardized difference (S/D) in the PS-stratified sub-populations) among the model pool. The method searches for the propensity score model with the best local balance while controling the global balance. The estimation is considered as "fail" if the minimum absolute S/D of input covariates in the whole population is more than 10% (uncontrolled global balance), and the function output NA value with an error message. The local balance is evaluated by a statistic (mean or max) of the absolute S/D in the PS-stratified sub-populations. The method only takes binary treatment.

Usage

```
PSLB1(
  X,
  Z,
  n_sigma = 20,
  ej,
  selectInX = "cov",
  k = 500,
  method = "exact",
  standardize = TRUE,
  criteria = "mean"
)
```

Arguments

`X`	The covariate matrix with the rows corresponding to the subjects/participants and the columns corresponding to the covariates. The covariate matrix X can not contain missing value. The function will stop if NA value is detected in X.
`Z`	The binary treatment indicator. A vector with 2 unique numeric values in 0 = untreated and 1 = treated. Its length should be the same as the number of all subjects/participants without missing value. The function will stop if NA value is detected in Z.
`n_sigma`	The number of bandwidth value of the Gaussian kernel function used to generate the feature space. See details in the description of function `Gaussian_Kernel_feature`. Default to 20.
`ej`	The matrix of the local neighborhoods with its rows representing the neighborhoods. It contains two columns of values greater or equal to 0 and less or equal to 1. The first columns are the start and the second column are the end point of the local neighborhoods.
`selectInX`	The matrix used to evaluate the local balance. selectInX = "cov" by default, which evaluates the local covariate balance on the input covariate matrix X. If selectInX = "feature", the local covariate balance is evaluated on the feature space transformed by kernel PCA.
`k`	The number of top K eigen value calculated. Note that k should be smaller than the sample size. The minimum of k and the sample size will be used. Defaults to 500. See details in the description of function `Gaussian_Kernel_feature`.
`method`	Choose "exact" to fit the justidentified CBPS model; choose "over" to fit the overidentified CBPS model. See details in Imai and Ratkovic (2014). Default is "exact".
`standardize`	Default is TRUE, which normalizes weights to sum to 1 within treated/untreated group. Set to FALSE to return IPW weights for ATE.
`criteria`	Choose "mean" to use the mean of the absolute S/D among all covariates and neighborhoods for the tuning parameters by T4LB; choose "max" to use the max of the absolute S/D among all covariates and neighborhoods for the tuning parameters by T4LB. Default is "mean".

Value

A list containing the following components:

- "weight": The IPW weights for binary ATE calculated by the fitted propensity score given by Tr/ps + (1-Tr)/(1-ps), where Tr is the treatment assignment and ps is the fitted propensity score. This expression for weight is before standardization (i.e. with standardize = FALSE). Standardization will make weights sum to 1 within untreated/treated group.
- "propensity.score": The fitted propensity score
- "balance": The matrix containing the global and local balance of the estimated propensity score. The first row contains the absolute S/D of each covariates in the whole study population, which represents the global covariate balance. The rest rows contain the absolute S/D of each covariates in the PS-stratified sub-population corresponding to the loal neighborhoods of "ej", which represent the local covariate balance.
- "coefficients": The coefficient of the fitted propensity score model
- "feature": The selected features by kernel PCA with the best loal covariate balance.
- "best.para": The parameters (bandwidth for Gaussian kernel and number of PCs) chosen with the best local covariate balance.
- "treat": The treatment assignment vector used

References

Imai, K. and Ratkovic, M. (2014) Covariate balancing propensity score. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 76, 243-263.

Examples

```
KS = Kang_Schafer_Simulation(n = 1500, seeds = 5050)
# Misspecified propensity score model
X = KS$Data[,7:10]
Z = KS$Data[,2]
# Specify the local neighborhoods
ej = cbind(seq(0,0.8,0.2),seq(0.2,1,0.2))
print(ej)
# PSLB 1 fitting
fit = PSLB1(X = X, Z = Z, n_sigma = 10, ej = ej)
print(fit$balance)
```

`sample.size.calculate` *Sample Size Calculation in Local Neighborhoods*

Description

This function calculate the sample sizes in pre-specified local neighborhoods (the number of subjects in each propensity score (PS) stratified sub-population).

Usage

```
sample.size.calculate(Z, p.hat, ej)
```

Arguments

Z	The binary treatment indicator. A vector with 2 unique numeric values in 0 = untreated and 1 = treated.
p.hat	The propensity score used to determine the sub-population in each local neighborhood.
ej	The matrix of the local neighborhoods, which contains two columns of positive values greater or equal to 0 and less or equal to 1. The rows of ej represent the neighborhoods. The first column is the start point of the local neighborhoods. The second column is the end point of the local neighborhoods.

Value

The matrix contains sample sizes in the local neighborhoods with each row corresponding to each local neighborhood (strata). The first gives the total number (n) of subjects in each strata. The third or fourth column give the sample sizes of untreated (n0) or treated (n1) subjects in each strata. The second column gives the ratio between n1 and n.

Examples

```
# Simulate data
KS = Kang_Schafer_Simulation(n = 1000, seeds = 5050)
# The treatment indicator and the true propensity score
Z = KS$Data[,2]
true.ps = KS$Data[,11]
# Local neighborhoods
ej = cbind(seq(0,0.7,0.1),seq(0.3,1,0.1))
# Calculate sample size in true PS-stratified sub-populations
local.sample.size = sample.size.calculate(Z = Z, p.hat = true.ps, ej)
```

T4LB *Tuning for Local Balance (T4LB) Algorithm*

Description

The T4LB algorithm optimze the balancing property of the estimated propensity score by minimizing a local balance criteria, while still controlling for the global balance. In addition to the the absolute standardized difference (S/D) of the covariates in the whole population, the T4LB selects the model with the minimum "mean" or "max" of the S/D in the pre-specified local neighborhoods. Hence, the selected model would have the best balancing property among the candidate models, and the estimated scores retain the interpretation as an appropriate propensity score, and the weights have the intepretation as an inverse probability weights (IPW). T4LB can be used in propensity score estimation approaches that requires parameter tuning, such as using machine learning approaches to estimate propensity scores.

Usage

```
T4LB(ps, weight, X, Z, ej, para, criteria = "mean")
```

Arguments

`ps`	A matrix contains the estimated propensity scores from candidate models. Each column of the "ps" matrix is the propensity scores estimated from each candidate model corresponding to each set of parameters.
`weight`	A matrix contains the weight calculated from the corresponding input propensity score. Each column of the "weight" matrix should correspond to each column of the "ps" matrix. If weight = NULL, the IPW of ATE will be calculated using the input propensity scores.
`X`	The covariate matrixes used for balance evaluation. It can be one or a list of covariate matrixes. If one covariate matrix is provided, this matrix will be used for balance evaluation for all candidate models. If a list of covariate matrixes are provided, each element of the list will be used for balance evaluation for each candidate models. The length of the "X" list must equal to the number of column of the "ps"/"weight" matrix. The rows of "X" matrix correspond to the subjects/participants, and the columns of the "X" matrix correspond to the covariates.
`Z`	The binary treatment indicator vector. A vector with 2 unique numeric values in 0 = untreated and 1 = treated.
`ej`	The matrix of the local neighborhoods, which contains two columns of positive values greater or equal to 0 and less or equal to 1. The rows of ej represent the neighborhoods. The first column is the start point of the local neighborhoods. The second column is the end point of the local neighborhoods.
`para`	The matrix whose each row contains the set of parameters for each candidate model.
`criteria`	Choose "mean" to use the mean of the absolute S/D among all covariates and neighborhoods as the criteria of local balance; choose "max" to use the max of the absolute S/D among all covariates and neighborhoods as the criteria of local balance. Default is "mean".

Value

A list containing the following components:

- "weight": The optimal weight from the selected model with the best local balance.
- "propensity.score": The optimal propensity score from the selected model with the best local balance.
- "parameter": The set of parameters of the selected model with the best local balance.
- "balance": The matrix containing the global and local balance of the selected propensity score model. The first row contains the absolute S/D of each covariates in the whole study population, which represents the global covariate balance. The rest rows contain the absolute S/D of the covariates in the PS-stratified sub-population corresponding to the loal neighborhoods of "ej", which represent the local covariate balance.

References

Lee, B. K., Lessler, J. and Stuart, E. A. (2009) Improving propensity score weighting using machine learning. Statistics in Medicine, 29, 337-346.

Examples

```
KS = Kang_Schafer_Simulation(n = 1500, seeds = 371834)
# Misspecified propensity score model
X = KS$Data[,7:10]
Z = KS$Data[,2]
# Local neighborhoods
ej = cbind(seq(0,0.8,0.2), seq(0.2,1,0.2))
print(ej)
# Transform X with Gaussian kernel using 10 sigma
x.k = Gaussian_Kernel_feature(X = X, n_sigma = 10)
sigma = rep(x.k$sigma_seq, each = 3)
r = as.vector(t(x.k$r))
para = cbind("sigma" = sigma, "l" = r)
# Fitting the kernelized logistic regression model
m = matrix(seq(1,30), nrow = 10, ncol = 3, byrow = T)
k.ps = matrix(nrow = nrow(X), ncol = nrow(para))
k.w = k.ps
for (i in 1:10) {
  for (j in 1:3) {
    log.fit = glm(Z~x.k$features[[i]][[j]], family = "binomial")
    A = weight.calculate.ps(Z, log.fit$fitted.values, standardize = T)
    k.ps[,m[i,j]] = A$ps
    k.w[,m[i,j]] = A$weight
  }
}
rm(A,log.fit,i,j,m)
best.glm = T4LB(ps = k.ps, weight = k.w, X = X, Z = Z, ej = ej, para = para)
str(t4lb.find)
best.glm$balance
```

`weight.calculate.ps` *The Inverse Probability Weight (IPW) for Average Treatment Effect (ATE)*

Description

This function calculates the IPW for ATE as Tr/ps + (1-Tr)/(1-ps), where Tr is the treatment indicator and ps is the propensity score.

Usage

```
weight.calculate.ps(Z, ps, standardize = TRUE)
```

Arguments

Z	The binary treatment indicator. A vector with 2 unique numeric values in 0 = untreated and 1 = treated.
ps	The propensity score used to calculate the IPW weight. A vector with numeric values greater than 0 and less than 1. The length of "ps" should equal to the length of "Z".

`standardize`	If standardize the calculated IPW or not. When standardize = FALSE, the weights are calculated by the IPW definition shown in the description. standardize = TRUE by default, which makes the the sum of weights be 1 in the untreated/treated group.

Value

A list containing the following components:

- "weight": The IPW weights calculated by the input propensity score
- "ps": The input propensity score

Examples

```
KS = Kang_Schafer_Simulation(n = 1000, seeds = 5050)
tr = KS$Data[,2]
true.ps = KS$Data[,11]
true.w = weight.calculate.ps(Z = tr, ps = true.ps, standardize = TRUE)
summary(true.w$weight)
c(sum(true.w$weight[which(tr==0)]), sum(true.w$weight[which(tr==1)]))
```

Index