Package: semidist (via r-universe)

October 17, 2024

Type Package

Title Measure Dependence Between Categorical and Continuous Variables

Version 0.1.0

Date 2023-11-19

Description Semi-distance and mean-variance (MV) index are proposed to measure the dependence between a categorical random variable and a continuous variable. Test of independence and feature screening for classification problems can be implemented via the two dependence measures. For the details of the methods, see Zhong et al. (2023) <doi:10.1080/01621459.2023.2284988>; Cui and Zhong (2019) <doi:10.1016/j.csda.2019.05.004>; Cui, Li and Zhong (2015) <doi:10.1080/01621459.2014.920256>.

License MIT + file LICENSE

Encoding UTF-8

Imports energy, FNN, furrr, purrr, Rcpp, stats

RoxygenNote 7.2.3

LinkingTo Rcpp, RcppArmadillo

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

URL `https://github.com/wzhong41/semidist`

BugReports `https://github.com/wzhong41/semidist/issues`

NeedsCompilation yes

Author Wei Zhong [aut], Zhuoxi Li [aut, cre, cph], Wenwen Guo [aut], Hengjian Cui [aut], Runze Li [aut]

Maintainer Zhuoxi Li `<chainchei@gmail.com>`

Repository CRAN

Date/Publication 2023-11-21 06:50:02 UTC

Contents

`MINTsemiperm` *Mutual information independence test (categorical-continuous case)*

Description

Implement the mutual information independence test (MINT) (Berrett and Samworth, 2019), but with some modification in estimating the mutual informaion (MI) between a categorical random variable and a continuous variable. The modification is based on the idea of Ross (2014).

`MINTsemiperm()` implements the permutation independence test via mutual information, but the parameter `k` should be pre-specified.

`MINTsemiauto()` automatically selects an appropriate `k` based on a data-driven procedure, and conducts `MINTsemiperm()` with the `k` chosen.

Usage

```
MINTsemiperm(X, y, k, B = 1000)

MINTsemiauto(X, y, kmax, B1 = 1000, B2 = 1000)
```

Arguments

`X`	Data of multivariate continuous variables, which should be an n-by-p matrix, or, a vector of length n (for univariate variable).
`y`	Data of categorical variables, which should be a factor of length n.
`k`	Number of nearest neighbor. See References for details.
`B`, `B1`, `B2`	Number of permutations to use. Defaults to 1000.
`kmax`	Maximum `k` in the automatic search for optimal `k`.

Value

A list with class `"indtest"` containing the following components

- `method`: name of the test;
- `name_data`: names of the `X` and `y`;
- `n`: sample size of the data;
- `num_perm`: number of replications in permutation test;
- `stat`: test statistic;
- `pvalue`: computed p-value.

For `MINTsemiauto()`, the list also contains

- `kmax`: maximum `k` in the automatic search for optimal `k`;
- `kopt`: optimal `k` chosen.

References

1. Berrett, Thomas B., and Richard J. Samworth. "Nonparametric independence testing via mutual information." *Biometrika* 106, no. 3 (2019): 547-566.
2. Ross, Brian C. "Mutual information between discrete and continuous data sets." *PloS one* 9, no. 2 (2014): e87357.

Examples

```
X <- mtcars[, c("mpg", "disp", "drat", "wt")]
y <- factor(mtcars[, "am"])

MINTsemiperm(X, y, 5)
MINTsemiauto(X, y, kmax = 32)
```

`mv` *Mean Variance (MV) statistics*

Description

Compute the statistics of mean variance (MV) index, which can measure the dependence between a univariate continuous variable and a categorical variable. See Cui, Li and Zhong (2015); Cui and Zhong (2019) for details.

Usage

```
mv(x, y, return_mat = FALSE)
```

Arguments

- x — Data of univariate continuous variables, which should be a vector of length n.
- y — Data of categorical variables, which should be a factor of length n.
- return_mat — A boolean. If FALSE (the default), only the calculated statistic is returned. If TRUE, also return the matrix of the indicator for x <= x_i, which is useful for the permutation test.

Value

The value of the corresponding sample statistic.

If the argument return_mat of mv() is set as TRUE, a list with elements

- mv: the MV index statistic;
- mat_x: the matrices of the distances of the indicator for x <= x_i;

will be returned.

See Also

- mv_test() for implementing independence test via MV index;
- mv_sis() for implementing feature screening via MV index.

Examples

```
x <- mtcars[, "mpg"]
y <- factor(mtcars[, "am"])
print(mv(x, y))

# Man-made independent data -------------------------------------------------
n <- 30; R <- 5; prob <- rep(1/R, R)
x <- rnorm(n)
y <- factor(sample(1:R, size = n, replace = TRUE, prob = prob), levels = 1:R)
print(mv(x, y))

# Man-made functionally dependent data --------------------------------------
n <- 30; R <- 3
x <- rep(0, n)
x[1:10] <- 0.3; x[11:20] <- 0.2; x[21:30] <- -0.1
y <- factor(rep(1:3, each = 10))
print(mv(x, y))
```

mv_sis *Feature screening via MV Index*

Description

Implement the feature screening for the classification problem via MV index.

Usage

```
mv_sis(X, y, d = NULL, parallel = FALSE)
```

Arguments

X	Data of multivariate covariates, which should be an n-by-p matrix.
y	Data of categorical response, which should be a factor of length n.
d	An integer specifying how many features should be kept after screening. Defaults to NULL. If NULL, then it will be set as $[n/log(n)]$, where $[x]$ denotes the integer part of x.
parallel	A boolean indicating whether to calculate parallelly via furrr::future_map. Defaults to FALSE.

Value

A list of the objects about the implemented feature screening:

- measurement: sample MV index calculated for each single covariate;
- selected: indicies or names (if avaiable as colnames of X) of covariates that are selected after feature screening;
- ordering: order of the calculated measurements of each single covariate. The first one is the largest, and the last is the smallest.

Examples

```
X <- mtcars[, c("mpg", "disp", "hp", "drat", "wt", "qsec")]
y <- factor(mtcars[, "am"])

mv_sis(X, y, d = 4)
```

mv_test *MV independence test*

Description

Implement the MV independence test via permutation test, or via the asymptotic approximation

Usage

```
mv_test(x, y, test_type = "perm", num_perm = 10000)
```

Arguments

x	Data of univariate continuous variables, which should be a vector of length n.
y	Data of categorical variables, which should be a factor of length n.
test_type	Type of the test: • "perm" (the default): Implement the test via permutation test; • "asym": Implement the test via the asymptotic approximation. See the Reference for details.
num_perm	The number of replications in permutation test.

Value

A list with class "indtest" containing the following components

- method: name of the test;
- name_data: names of the x and y;
- n: sample size of the data;
- num_perm: number of replications in permutation test;
- stat: test statistic;
- pvalue: computed p-value. (Notice: asymptotic test cannot return a p-value, but only the critical values crit_vals for 90%, 95% and 99% confidence levels.)

Examples

```
x <- mtcars[, "mpg"]
y <- factor(mtcars[, "am"])
test <- mv_test(x, y)
print(test)
test_asym <- mv_test(x, y, test_type = "asym")
print(test_asym)

# Man-made independent data --------------------------------------------------
n <- 30; R <- 5; prob <- rep(1/R, R)
x <- rnorm(n)
```

```
y <- factor(sample(1:R, size = n, replace = TRUE, prob = prob), levels = 1:R)
test <- mv_test(x, y)
print(test)
test_asym <- mv_test(x, y, test_type = "asym")
print(test_asym)

# Man-made functionally dependent data --------------------------------------
n <- 30; R <- 3
x <- rep(0, n)
x[1:10] <- 0.3; x[11:20] <- 0.2; x[21:30] <- -0.1
y <- factor(rep(1:3, each = 10))
test <- mv_test(x, y)
print(test)
test_asym <- mv_test(x, y, test_type = "asym")
print(test_asym)
```

---

print.indtest — *Print Method for Independence Tests Between Categorical and Continuous Variables*

Description

Printing object of class "indtest", by simple print method.

Usage

```
## S3 method for class 'indtest'
print(x, digits = getOption("digits"), ...)
```

Arguments

x	"indtest" class object.
digits	minimal number of *significant* digits.
...	further arguments passed to or from other methods.

Value

None

Examples

```
# Man-made functionally dependent data --------------------------------------
n <- 30; R <- 3
x <- rep(0, n)
x[1:10] <- 0.3; x[11:20] <- 0.2; x[21:30] <- -0.1
y <- factor(rep(1:3, each = 10))
test <- mv_test(x, y)
print(test)
```

```
test_asym <- mv_test(x, y, test_type = "asym")
print(test_asym)
```

---

sdcov *Semi-distance covariance and correlation statistics*

Description

Compute the statistics (or sample estimates) of semi-distance covariance and correlation. The semi-distance correlation is a standardized version of semi-distance covariance, and it can measure the dependence between a *multivariate* continuous variable and a categorical variable. See Details for the definition of semi-distance covariance and semi-distance correlation.

Usage

```
sdcov(X, y, type = "V", return_mat = FALSE)

sdcor(X, y)
```

Arguments

X	Data of multivariate continuous variables, which should be an n-by-p matrix, or, a vector of length n (for univariate variable).
y	Data of categorical variables, which should be a factor of length n.
type	Type of statistic: "V" (the default) or "U". See Details.
return_mat	A boolean. If FALSE (the default), only the calculated statistic is returned. If TRUE, also return the matrix of the distances of X and the divergences of y, which is useful for the permutation test.

Details

For $\boldsymbol{X} \in \mathbb{R}^p$ and $Y \in \{1, 2, \cdots, R\}$, the (population-level) semi-distance covariance is defined as

$$\mathrm{SDcov}(\boldsymbol{X}, Y) = \mathrm{E}\left[\|\boldsymbol{X} - \widetilde{\boldsymbol{X}}\| \left(1 - \sum_{r=1}^{R} I(Y = r, \widetilde{Y} = r)/p_r\right)\right],$$

where $p_r = P(Y = r)$ and $(\widetilde{\boldsymbol{X}}, \widetilde{Y})$ is an iid copy of $(\boldsymbol{X}, Y)$. The (population-level) semi-distance correlation is defined as

$$\mathrm{SDcor}(\boldsymbol{X}, Y) = \frac{\mathrm{SDcov}(\boldsymbol{X}, Y)}{\mathrm{dvar}(\boldsymbol{X})\sqrt{R-1}},$$

where $\mathrm{dvar}(\boldsymbol{X})$ is the distance variance (Szekely, Rizzo, and Bakirov 2007) of $\boldsymbol{X}$.

With n observations $\{(\boldsymbol{X}_i, Y_i)\}_{i=1}^n$, `sdcov()` and `sdcor()` can compute the sample estimates for the semi-distance covariance and correlation.

If `type = "V"`, the semi-distance covariance statistic is computed as a V-statistic, which takes a very similar form as the energy-based statistic with double centering, and is always non-negative. Specifically,

$$\mathrm{SDcov}_n(\boldsymbol{X}, y) = \frac{1}{n^2}\sum_{k=1}^{n}\sum_{l=1}^{n} A_{kl}B_{kl},$$

where

$$A_{kl} = a_{kl} - \bar{a}_{k.} - \bar{a}_{.l} + \bar{a}_{..}$$

is the double centering (Szekely, Rizzo, and Bakirov 2007) of $a_{kl} = \|\boldsymbol{X}_k - \boldsymbol{X}_l\|$, and

$$B_{kl} = 1 - \sum_{r=1}^{R} I(Y_k = r)I(Y_l = r)/\hat{p}_r$$

with $\hat{p}_r = n_r/n = n^{-1}\sum_{i=1}^{n} I(Y_i = r)$. The semi-distance correlation statistic is

$$\mathrm{SDcor}_n(\boldsymbol{X}, y) = \frac{\mathrm{SDcov}_n(\boldsymbol{X}, y)}{\mathrm{dvar}_n(\boldsymbol{X})\sqrt{R-1}},$$

where $\mathrm{dvar}_n(\boldsymbol{X})$ is the V-statistic of distance variance of $\boldsymbol{X}$.

If `type = "U"`, then the semi-distance covariance statistic is computed as an "estimated U-statistic", which is utilized in the independence test statistic and is not necessarily non-negative. Specifically,

$$\widetilde{\mathrm{SDcov}}_n(\boldsymbol{X}, y) = \frac{1}{n(n-1)}\sum_{i\neq j}\|\boldsymbol{X}_i - \boldsymbol{X}_j\|\left(1 - \sum_{r=1}^{R} I(Y_i = r)I(Y_j = r)/\tilde{p}_r\right),$$

where $\tilde{p}_r = (n_r - 1)/(n-1) = (n-1)^{-1}(\sum_{i=1}^{n} I(Y_i = r) - 1)$. Note that the test statistic of the semi-distance independence test is

$$T_n = n \cdot \widetilde{\mathrm{SDcov}}_n(\boldsymbol{X}, y).$$

Value

The value of the corresponding sample statistic.

If the argument `return_mat` of `sdcov()` is set as `TRUE`, a list with elements

- `sdcov`: the semi-distance covariance statistic;
- `mat_x, mat_y`: the matrices of the distances of X and the divergences of y, respectively;

will be returned.

See Also

- `sd_test()` for implementing independence test via semi-distance covariance;
- `sd_sis()` for implementing groupwise feature screening via semi-distance correlation.

Examples

```
X <- mtcars[, c("mpg", "disp", "drat", "wt")]
y <- factor(mtcars[, "am"])
print(sdcov(X, y))
print(sdcor(X, y))

# Man-made independent data ------------------------------------------------
n <- 30; R <- 5; p <- 3; prob <- rep(1/R, R)
X <- matrix(rnorm(n*p), n, p)
y <- factor(sample(1:R, size = n, replace = TRUE, prob = prob), levels = 1:R)
print(sdcov(X, y))
print(sdcor(X, y))

# Man-made functionally dependent data -------------------------------------
n <- 30; R <- 3; p <- 3
X <- matrix(0, n, p)
X[1:10, 1] <- 1; X[11:20, 2] <- 1; X[21:30, 3] <- 1
y <- factor(rep(1:3, each = 10))
print(sdcov(X, y))
print(sdcor(X, y))
```

`sd_sis` *Feature screening via semi-distance correlation*

Description

Implement the (grouped) feature screening for the classification problem via semi-distance correlation.

Usage

```
sd_sis(X, y, group_info = NULL, d = NULL, parallel = FALSE)
```

Arguments

X	Data of multivariate covariates, which should be an n-by-p matrix.
y	Data of categorical response, which should be a factor of length n.
group_info	A list specifying the group information, with elements being sets of indicies of covariates in a same group. For example, `list(c(1, 2, 3), c(4, 5))` specifies that covariates 1, 2, 3 are in a group and covariates 4, 5 are in another group. Defaults to `NULL`. If `NULL`, then it will be set as `list(1, 2, ..., p)`, that is, treat each single covariate as a group. If `X` has colnames, then the colnames can be used to specified the `group_info`. For example, `list(c("a", "b"), c("c", "d"))`. The names of the list can help recoginize the group. For example, `list(grp_ab = c("a", "b"), grp_cd = c("c", "d"))`. If names of the list are not specified, `c("Grp 1", "Grp 2", ..., "Grp J")` will be applied.

d — An integer specifying *at least* how many (single) features should be kept after screening. For example, if `group_info = list(c(1, 2), c(3, 4))` and `d = 3`, then all features 1, 2, 3, 4 must be selected since it should guarantee at least 3 features are kept.

Defaults to `NULL`. If `NULL`, then it will be set as $[n/log(n)]$, where $[x]$ denotes the integer part of x.

parallel — A boolean indicating whether to calculate parallelly via `furrr::future_map`. Defaults to `FALSE`.

Value

A list of the objects about the implemented feature screening:

- `group_info`: group information;
- `measurement`: sample semi-distance correlations calculated for the groups specified in `group_info`;
- `selected`: indicies/names of (single) covariates that are selected after feature screening;
- `ordering`: order of the calculated measurements of the groups specified in `group_info`. The first one is the largest, and the last is the smallest.

See Also

`sdcor()` for calculating the sample semi-distance correlation.

Examples

```
X <- mtcars[, c("mpg", "disp", "hp", "drat", "wt", "qsec")]
y <- factor(mtcars[, "am"])

sd_sis(X, y, d = 4)

# Suppose we have prior information for the group structure as
# ("mpg", "drat"), ("disp", "hp") and ("wt", "qsec")
group_info <- list(
  mpg_drat = c("mpg", "drat"),
  disp_hp = c("disp", "hp"),
  wt_qsec = c("wt", "qsec")
)
sd_sis(X, y, group_info, d = 4)
```

---

sd_test — *Semi-distance independence test*

Description

Implement the semi-distance independence test via permutation test, or via the asymptotic approximation when the dimensionality of continuous variables p is high.

Usage

```
sd_test(X, y, test_type = "perm", num_perm = 10000)
```

Arguments

`X`	Data of multivariate continuous variables, which should be an n-by-p matrix, or, a vector of length n (for univariate variable).
`y`	Data of categorical variables, which should be a factor of length n.
`test_type`	Type of the test: • `"perm"` (the default): Implement the test via permutation test; • `"asym"`: Implement the test via the asymptotic approximation when the dimension of continuous variables p is high. See the Reference for details.
`num_perm`	The number of replications in permutation test. Defaults to 10000. See Details and Reference.

Details

The semi-distance independence test statistic is

$$T_n = n \cdot \widetilde{\text{SDcov}}_n(X, y),$$

where the $\widetilde{\text{SDcov}}_n(X, y)$ can be computed by `sdcov(X, y, type = "U")`.

For the permutation test (`test_type = "perm"`), totally K replications of permutation will be conducted, and the argument `num_perm` specifies the K here. The p-value of permutation test is computed by

$$\text{p-value} = (\sum_{k=1}^{K} I(T_n^{*(k)} \geq T_n) + 1)/(K + 1),$$

where T_n is the semi-distance test statistic and $T_n^{*(k)}$ is the test statistic with k-th permutation sample.

When the dimension of the continuous variables is high, the asymptotic approximation approach can be applied (`test_type = "asym"`), which is computationally faster since no permutation is needed.

Value

A list with class `"indtest"` containing the following components

- `method`: name of the test;
- `name_data`: names of the `X` and `y`;
- `n`: sample size of the data;
- `test_type`: type of the test;
- `num_perm`: number of replications in permutation test, if `test_type = "perm"`;
- `stat`: test statistic;
- `pvalue`: computed p-value.

See Also

sdcov() for computing the statistic of semi-distance covariance.

Examples

```
X <- mtcars[, c("mpg", "disp", "drat", "wt")]
y <- factor(mtcars[, "am"])
test <- sd_test(X, y)
print(test)

# Man-made independent data -------------------------------------------------
n <- 30; R <- 5; p <- 3; prob <- rep(1/R, R)
X <- matrix(rnorm(n*p), n, p)
y <- factor(sample(1:R, size = n, replace = TRUE, prob = prob), levels = 1:R)
test <- sd_test(X, y)
print(test)

# Man-made functionally dependent data --------------------------------------
n <- 30; R <- 3; p <- 3
X <- matrix(0, n, p)
X[1:10, 1] <- 1; X[11:20, 2] <- 1; X[21:30, 3] <- 1
y <- factor(rep(1:3, each = 10))
test <- sd_test(X, y)
print(test)

#' Man-made high-dimensionally independent data -----------------------------
n <- 30; R <- 3; p <- 100
X <- matrix(rnorm(n*p), n, p)
y <- factor(rep(1:3, each = 10))
test <- sd_test(X, y)
print(test)

test <- sd_test(X, y, test_type = "asym")
print(test)

# Man-made high-dimensionally dependent data --------------------------------
n <- 30; R <- 3; p <- 100
X <- matrix(0, n, p)
X[1:10, 1] <- 1; X[11:20, 2] <- 1; X[21:30, 3] <- 1
y <- factor(rep(1:3, each = 10))
test <- sd_test(X, y)
print(test)

test <- sd_test(X, y, test_type = "asym")
print(test)
```

---

switch_cat_repr *Switch the representation of a categorical object*

Description

Categorical data with n observations and R levels can typically be represented as two forms in R: a factor with length n, or an n by K indicator matrix with elements being 0 or 1. This function is to switch the form of a categorical object from one to the another.

Usage

```
switch_cat_repr(obj)
```

Arguments

`obj`	an object representing categorical data, either a factor or an indicator matrix with each row representing an observation.

Value

categorical object in the another form.

`tr_estimate` *Estimate the trace of the covariance matrix and its square*

Description

For a design matrix $\mathbf{X}$, estimate the trace of its covariance matrix $\Sigma = \mathrm{cov}(\mathbf{X})$, and the square of covariance matrix Σ^2.

Usage

```
tr_estimate(X)
```

Arguments

`X`	The design matrix.

Value

A list with elements:

- `tr_S`: estimate for trace of Σ;
- `tr_S2`: estimate for trace of Σ^2.

Index