

Package: BCodifSIS (via r-universe)

October 6, 2026

Type Package

Title Ball-Codifference Sure Independence Screening

Version 0.3.2

Description Computes Ball-codifference scores and performs sure independence screening for high-dimensional predictors. The score combines random-ball empirical probabilities with a bounded local codifference weight based on trigonometric characteristic-function contrasts. The functions are intended for marginal screening when moment-based dependence summaries may be unstable or undefined, such as in heavy-tailed data. The implemented methodology is described in Rezapour and Maroufy (2026) ``Ball-Codifference Screening for Heavy-Tailed Predictors" <[doi:10.48550/arXiv.2607.20821](https://doi.org/10.48550/arXiv.2607.20821)>.

License MIT + file LICENSE

Encoding UTF-8

Imports stats

NeedsCompilation yes

SystemRequirements C99

URL <https://arxiv.org/abs/2607.20821>

Collate 'bcodif.R' 'quick_check.R'

Author Mohsen Rezapour [aut, cre]

Maintainer Mohsen Rezapour <mohsenrzp@gmail.com>

Repository <https://mohsenrzp.r-universe.dev>

Date/Publication 2026-08-06 14:42:34 UTC

RemoteUrl <https://github.com/cran/BCodifSIS>

RemoteRef HEAD

RemoteSha 0a00d15585d8d882899dd6a3104b23796b8fbcc1

Contents

BCodifSIS-package	2
bcodif_score_pair	2
bcodif_score_slow	3
bcodif_scores	4
bcodif_sis	5
quick_check	6
Index	7

BCodifSIS-package	<i>Ball-Codifference Sure Independence Screening</i>
-------------------	--

Description

Computes Ball-codifference scores and performs sure independence screening for high-dimensional predictors. The package provides compiled C routines and R wrappers for ranking predictors by a local codifference-weighted random-ball score.

Details

The main screening function is `bcodif_sis()`. Lower-level functions are available for computing scores for one predictor pair or for all columns of a predictor matrix.

References

Rezapour, M. and Maroufy, V. (2026). "Ball-Codifference Screening for Heavy-Tailed Predictors". [doi:10.48550/arXiv.2607.20821](https://doi.org/10.48550/arXiv.2607.20821).

See Also

`bcodif_sis()`, `bcodif_scores()`, `bcodif_score_pair()`

bcodif_score_pair	<i>Compute One Ball-Codifference Score</i>
-------------------	--

Description

Computes the Ball-codifference score for one predictor-response pair. The function `bcodif_score()` is an alias for `bcodif_score_pair()`.

Usage

```
bcodif_score_pair(x, y, normalize = FALSE, absolute_kappa = FALSE,
  ridge = 1e-12)

bcodif_score(x, y, normalize = FALSE, absolute_kappa = FALSE,
  ridge = 1e-12)
```

Arguments

x	Numeric predictor vector.
y	Numeric response vector.
normalize	Logical. If TRUE, use the normalized score.
absolute_kappa	Logical. If TRUE, use $ \kappa_{ij} $.
ridge	Nonnegative ridge used only when normalize = TRUE.

Value

One numeric score.

See Also

bcodif_scores(), bcodif_score_slow()

Examples

```
grid <- seq_len(40)
x <- sin(grid / 4) + cos(grid / 7)
y <- x + 0.5 * sin(grid / 3)
bcodif_score_pair(x, y)
```

bcodif_score_slow	<i>Slow Pure-R Ball-Codifference Score</i>
-------------------	--

Description

Computes the same unnormalized empirical Ball-codifference score as the compiled routine, but using simple R loops. This function is mainly intended for testing and small examples.

Usage

```
bcodif_score_slow(x, y, absolute_kappa = FALSE)
```

Arguments

x	Numeric predictor vector.
y	Numeric response vector.
absolute_kappa	Logical. If TRUE, use $ \kappa_{ij} $.

Value

One numeric score.

See Also

bcodif_score_pair(), quick_check()

Examples

```

grid <- seq_len(15)
x <- sin(grid / 4) + cos(grid / 7)
y <- x + sin(grid / 3)
fast <- bcodif_score(x, y)
slow <- bcodif_score_slow(x, y)
all.equal(fast, slow, tolerance = 1e-10)

```

bcodif_scores

Compute Ball-Codifference Scores for Predictor Columns

Description

Computes the empirical Ball-codifference score between each column of X and the response y . The compiled implementation uses a sorted sweep and Fenwick-tree updates to avoid a direct cubic loop over all observations for each ball center.

Usage

```
bcodif_scores(X, y, normalize = FALSE, absolute_kappa = FALSE, ridge = 1e-12)
```

Arguments

X	Numeric predictor matrix. Rows are observations and columns are predictors.
y	Numeric response vector with length <code>nrow(X)</code> .
<code>normalize</code>	Logical. If TRUE, use the normalized Ball-codifference score.
<code>absolute_kappa</code>	Logical. If TRUE, use $ \kappa_{ij} $ instead of signed κ_{ij} .
<code>ridge</code>	Nonnegative ridge used only when <code>normalize = TRUE</code> .

Value

A numeric vector of scores, one per predictor column. Column names are preserved when available.

See Also

```
bcodif_sis(), bcodif_score_pair()
```

Examples

```

grid <- seq_len(50)
X <- vapply(seq_len(5), function(j) {
  sin(grid / (j + 1)) + cos(grid * j / 9)
}, numeric(50))
y <- X[, 1] + sin(grid / 3)
bcodif_scores(X, y)

```

bcodif_sis

*Ball-Codifference Sure Independence Screening***Description**

Ranks predictors by their marginal Ball-codifference score and returns the top d variables. The alias `bcodifsis()` is provided for users who prefer a compact function name.

Usage

```
bcodif_sis(X, y, d = NULL, normalize = FALSE, absolute_kappa = FALSE,
           ridge = 1e-12)
```

```
bcodifsis(x, y, d = NULL, normalize = FALSE, absolute_kappa = FALSE,
          ridge = 1e-12)
```

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

Arguments

<code>X</code>	Numeric predictor matrix. Rows are observations and columns are predictors.
<code>x</code>	For <code>bcodifsis()</code> , a numeric predictor matrix. For the print method, an object returned by <code>bcodif_sis()</code> .
<code>y</code>	Numeric response vector with length <code>nrow(X)</code> .
<code>d</code>	Number of variables to retain. The default is <code>floor(nrow(X) / log(nrow(X)))</code> .
<code>normalize</code>	Logical. If TRUE, rank by the normalized score.
<code>absolute_kappa</code>	Logical. If TRUE, use $ \kappa_{ij} $.
<code>ridge</code>	Nonnegative ridge used only when <code>normalize = TRUE</code> .
<code>...</code>	Additional arguments for the print method, currently unused.

Value

An object of class `bcodif_sis`, which is a list with components:

<code>ix</code>	Integer indices of the selected variables.
<code>scores</code>	Numeric score for every predictor.
<code>ranking</code>	Integer ranking from largest score to smallest score.
<code>complete.info</code>	Data frame with variable index, name, statistic, rank, and selection flag.
<code>d</code>	Number of retained variables.
<code>method</code>	Name of the screening method used.
<code>normalize</code>	Whether normalized scores were used.
<code>absolute_kappa</code>	Whether the absolute local codifference weight was used.
<code>call</code>	Matched function call.

See Also

`bcodif_scores()`, `quick_check()`

Examples

```
grid <- seq_len(80)
X <- vapply(seq_len(10), function(j) {
  sin(grid / (j + 1)) + cos(grid * j / 9)
}, numeric(80))
y <- X[, 2] + 0.5 * sin(grid / 3)
fit <- bcodif_sis(X, y, d = 3)
fit$ix
head(fit$complete.info)
```

quick_check

Quick Installation and Functionality Check

Description

Runs a small self-test. The test compares the compiled score to the slow pure-R score and then fits a small screening example.

Usage

`quick_check()`

Value

A list with the compiled score, slow score, their absolute difference, and a fitted `bcodif_sis` object.

See Also

`bcodif_sis()`, `bcodif_score_slow()`

Examples

`quick_check()`

Index

* **package**

BCodifSIS-package, [2](#)

bcodif_score (bcodif_score_pair), [2](#)

bcodif_score_pair, [2](#)

bcodif_score_slow, [3](#)

bcodif_scores, [4](#)

bcodif_sis, [5](#)

BCodifSIS (BCodifSIS-package), [2](#)

bcodifsis (bcodif_sis), [5](#)

BCodifSIS-package, [2](#)

print.bcodif_sis (bcodif_sis), [5](#)

quick_check, [6](#)