Package 'aricode'

May 13, 2026

Type Package

Title Efficient Computations of Standard Clustering Comparison Measures

Version 1.1.0

Maintainer Julien Chiquet <julien.chiquet@inrae.fr>

Description Implements an efficient O(n) algorithm based on bucket-sorting for fast computation of standard clustering comparison measures. Available measures include adjusted Rand index (ARI), normalized information distance (NID), normalized mutual information (NMI), normalized variation information (NVI) and entropy, as described in Vinh et al (2009) <doi:10.1145/1553374.1553511>. Include AMI (Adjusted Mutual Information) since version 0.1.2, a modified version of ARI (MARI), as described in Sundqvist et al. <doi:10.1007/s00180-022-01230-7> and simple Chi-square distance since version 1.0.0.

License GPL (>= 3)

URL https://github.com/jchiquet/aricode

BugReports https://github.com/jchiquet/aricode/issues

Encoding UTF-8

Imports Matrix, Rcpp, lifecycle

Suggests testthat, spelling, mclust, ggplot2, pkgdown

LinkingTo Rcpp

Language en-US

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Author Julien Chiquet [aut, cre] (ORCID: <https://orcid.org/0000-0002-3629-3429>), Guillem Rigaill [aut], Martina Sundqvist [aut], Valentin Dervieux [ctb], Florent Bersani [ctb]

Repository CRAN

Date/Publication 2026-05-13 15:20:02 UTC

Contents

`AMI` *Adjusted Mutual Information*

Description

A function to compute the adjusted mutual information between two classifications

Usage

```
AMI(c1, c2, sorted_pairs = NULL)
```

Arguments

`c1`	A vector of length n with values between 0 and $N_1 < n$ representing the first classification. Supported types: integer, numeric, or factor. Avoid character vectors for better performance. Must not be a list.
`c2`	A vector of length n with values between 0 and $N_2 < n$ representing the second classification. Supported types: integer, numeric, or factor. Avoid character vectors for better performance. Must not be a list.
`sorted_pairs`	optional output of function sort_pairs (if already computed). If 'NULL' (the default), will be called internally

Value

a scalar with the adjusted rand index.

See Also

ARI, RI, NID, NVI, NMI, clustComp

Examples

```
data(iris)
cl <- cutree(hclust(dist(iris[, -5])), 4)
AMI(cl, iris$Species)
```

ARI *Adjusted Rand Index*

Description

A function to compute the adjusted rand index between two classifications

Usage

```
ARI(c1, c2, sorted_pairs = NULL)
```

Arguments

c1	A vector of length n with values between 0 and $N_1 < n$ representing the first classification. Supported types: integer, numeric, or factor. Avoid character vectors for better performance. Must not be a list.
c2	A vector of length n with values between 0 and $N_2 < n$ representing the second classification. Supported types: integer, numeric, or factor. Avoid character vectors for better performance. Must not be a list.
sorted_pairs	optional output of function sort_pairs (if already computed). If 'NULL' (the default), will be called internally

Value

a scalar with the adjusted Rand index.

See Also

RI, NID, NVI, NMI, clustComp

Examples

```
data(iris)
cl <- cutree(hclust(dist(iris[, -5])), 4)
ARI(cl, iris$Species)
```

Chi2 *Chi-square statistics*

Description

A function to compute the Chi-2 statistic

Usage

```
Chi2(c1, c2, sorted_pairs = NULL)
```

Arguments

c1	A vector of length n with values between 0 and $N_1 < n$ representing the first classification. Supported types: integer, numeric, or factor. Avoid character vectors for better performance. Must not be a list.
c2	A vector of length n with values between 0 and $N_2 < n$ representing the second classification. Supported types: integer, numeric, or factor. Avoid character vectors for better performance. Must not be a list.
sorted_pairs	optional output of function sort_pairs (if already computed). If 'NULL' (the default), will be called internally

Value

a scalar with the Chi-square statistic.

See Also

ARI, NID, NVI, NMI, clustComp

Examples

```
data(iris)
cl <- cutree(hclust(dist(iris[, -5])), 4)
Chi2(cl, iris$Species)
```

compare_clustering *Measures of similarity between two classification*

Description

A function for computing all the measures of similarity implemented in this package at once. Include (A)RI, (N)MI, (N)VI, (N)ID, Chi2, MARI, Frobenius

Usage

```
compare_clustering(c1, c2, sorted_pairs = NULL, AMI = FALSE)
```

Arguments

c1	A vector of length n with values between 0 and $N_1 < n$ representing the first classification. Supported types: integer, numeric, or factor. Avoid character vectors for better performance. Must not be a list.
c2	A vector of length n with values between 0 and $N_2 < n$ representing the second classification. Supported types: integer, numeric, or factor. Avoid character vectors for better performance. Must not be a list.
sorted_pairs	optional output of function sort_pairs (if already computed). If 'NULL' (the default), will be called internally
AMI	Boolean: should the AMI be computed (more costly than all other measures)? Default is 'FALSE'.

Value

a list with all the measures available

See Also

RI, NID, NVI, NMI, ARI

Examples

```
data(iris)
cl <- cutree(hclust(dist(iris[, -5])), 4)
compare_clustering(cl, iris$Species)
```

---

entropy *Entropy*

Description

A function to compute the empirical entropy for two vectors of classification and the joint entropy

Usage

```
entropy(c1, c2, sorted_pairs = NULL)
```

Arguments

c1	A vector of length n with values between 0 and $N_1 < n$ representing the first classification. Supported types: integer, numeric, or factor. Avoid character vectors for better performance. Must not be a list.
c2	A vector of length n with values between 0 and $N_2 < n$ representing the second classification. Supported types: integer, numeric, or factor. Avoid character vectors for better performance. Must not be a list.
sorted_pairs	optional output of function sort_pairs (if already computed). If 'NULL' (the default), will be called internally

Value

a list with the two conditional entropies, the joint entropy and output of sort_pairs.

Examples

```
data(iris)
cl <- cutree(hclust(dist(iris[, -5])), 4)
entropy(cl, iris$Species)
```

Frobenius *Frobenius norm*

Description

A function to compute the Frobenius norm between two classifications as defined in Lajugie et al. 2014 and Arlot et al 2019

Usage

```
Frobenius(c1, c2, sorted_pairs = NULL)
```

Arguments

c1	A vector of length n with values between 0 and $N_1 < n$ representing the first classification. Supported types: integer, numeric, or factor. Avoid character vectors for better performance. Must not be a list.
c2	A vector of length n with values between 0 and $N_2 < n$ representing the second classification. Supported types: integer, numeric, or factor. Avoid character vectors for better performance. Must not be a list.
sorted_pairs	optional output of function sort_pairs (if already computed). If 'NULL' (the default), will be called internally

Value

a scalar with the Frobenius norm.

References

- Rémi Lajugie, Francis Bach, and Sylvain Arlot. "Large-margin metric learning for constrained partitioning problems." International Conference on Machine Learning. PMLR, 2014. - Sylvain Arlot , Alain Celisse, and Zaid Harchaoui. "A kernel multiple change-point algorithm via model selection." Journal of machine learning research 20.162 (2019): 1-56.

See Also

ARI, NID, NVI, NMI, clustComp

Examples

```
data(iris)
cl <- cutree(hclust(dist(iris[, -5])), 4)
Frobenius(cl, iris$Species)
```

MARI *Modified Adjusted Rand Index*

Description

A function to compute a modified adjusted rand index between two classifications as proposed by Sundqvist et al. (2023), based on a multinomial model.

Usage

```
MARI(c1, c2, sorted_pairs = NULL, raw = FALSE)
```

Arguments

c1	A vector of length n with values between 0 and $N_1 < n$ representing the first classification. Supported types: integer, numeric, or factor. Avoid character vectors for better performance. Must not be a list.
c2	A vector of length n with values between 0 and $N_2 < n$ representing the second classification. Supported types: integer, numeric, or factor. Avoid character vectors for better performance. Must not be a list.
sorted_pairs	optional output of function sort_pairs (if already computed). If 'NULL' (the default), will be called internally
raw	Boolean: should the raw version of the MARI be computed? Default to 'FALSE'.

Value

a scalar with the modified ARI.

References

Sundqvist, Martina, Julien Chiquet, and Guillem Rigaill. "Adjusting the adjusted Rand Index: A multinomial story." Computational Statistics 38.1 (2023): 327-347.

See Also

ARI, NID, NVI, NMI, clustComp

Examples

```
data(iris)
cl <- cutree(hclust(dist(iris[, -5])), 4)
MARI(cl, iris$Species)
```

NID *Normalized information distance (NID)*

Description

A function to compute the NID between two classifications

Usage

```
NID(c1, c2, sorted_pairs = NULL)
```

Arguments

c1	A vector of length n with values between 0 and $N_1 < n$ representing the first classification. Supported types: integer, numeric, or factor. Avoid character vectors for better performance. Must not be a list.
c2	A vector of length n with values between 0 and $N_2 < n$ representing the second classification. Supported types: integer, numeric, or factor. Avoid character vectors for better performance. Must not be a list.
sorted_pairs	optional output of function sort_pairs (if already computed). If 'NULL' (the default), will be called internally

Value

a scalar with the normalized information distance .

See Also

RI, NMI, NVI, ARI, clustComp

Examples

```
data(iris)
cl <- cutree(hclust(dist(iris[, -5])), 4)
NID(cl, iris$Species)
```

NMI *Normalized mutual information (NMI)*

Description

A function to compute the NMI between two classifications

Usage

```
NMI(
  c1,
  c2,
  variant = c("max", "min", "sqrt", "sum", "joint"),
  sorted_pairs = NULL
)
```

Arguments

c1	A vector of length n with values between 0 and $N_1 < n$ representing the first classification. Supported types: integer, numeric, or factor. Avoid character vectors for better performance. Must not be a list.
c2	A vector of length n with values between 0 and $N_2 < n$ representing the second classification. Supported types: integer, numeric, or factor. Avoid character vectors for better performance. Must not be a list.
variant	a string in ("max", "min", "sqrt", "sum", "joint"): different variants of NMI. Default use "max".
sorted_pairs	optional output of function sort_pairs (if already computed). If 'NULL' (the default), will be called internally

Value

a scalar with the normalized mutual information .

See Also

RI, NID, NVI, ARI, clustComp

Examples

```
data(iris)
cl <- cutree(hclust(dist(iris[, -5])), 4)
NMI(cl, iris$Species)
```

NVI *Normalized variation of information (NVI)*

Description

A function to compute the NVI between two classifications

Usage

```
NVI(c1, c2, sorted_pairs = NULL)
```

Arguments

c1	A vector of length n with values between 0 and $N_1 < n$ representing the first classification. Supported types: integer, numeric, or factor. Avoid character vectors for better performance. Must not be a list.
c2	A vector of length n with values between 0 and $N_2 < n$ representing the second classification. Supported types: integer, numeric, or factor. Avoid character vectors for better performance. Must not be a list.
sorted_pairs	optional output of function sort_pairs (if already computed). If 'NULL' (the default), will be called internally

Value

a scalar with the normalized variation of information.

See Also

RI, NID, NMI, ARI, clustComp

Examples

```
data(iris)
cl <- cutree(hclust(dist(iris[, -5])), 4)
NVI(cl, iris$Species)
```

RI *Rand Index*

Description

A function to compute the Rand index between two classifications

Usage

```
RI(c1, c2, sorted_pairs = NULL)
```

Arguments

c1	A vector of length n with values between 0 and $N_1 < n$ representing the first classification. Supported types: integer, numeric, or factor. Avoid character vectors for better performance. Must not be a list.
c2	A vector of length n with values between 0 and $N_2 < n$ representing the second classification. Supported types: integer, numeric, or factor. Avoid character vectors for better performance. Must not be a list.
sorted_pairs	optional output of function sort_pairs (if already computed). If 'NULL' (the default), will be called internally

Value

a scalar with the Rand index.

See Also

ARI, NID, NVI, NMI, clustComp

Examples

```
data(iris)
cl <- cutree(hclust(dist(iris[, -5])), 4)
RI(cl, iris$Species)
```

---

sort_pairs *Sort Pairs*

Description

A function to sort pairs of integers or factors and identify the pairs between two classifications

Usage

```
sort_pairs(c1, c2, spMat = FALSE)
```

Arguments

c1	A vector of length n with values between 0 and $N_1 < n$ representing the first classification. Supported types: integer, numeric, or factor. Avoid character vectors for better performance. Must not be a list.
c2	A vector of length n with values between 0 and $N_2 < n$ representing the second classification. Supported types: integer, numeric, or factor. Avoid character vectors for better performance. Must not be a list.
spMat	Logical. If TRUE, returns the contingency table as a sparse matrix. Note: sparse encoding may be more computationally expensive than the algorithm itself. Default is FALSE.

Details

Pair sorting, which is at the heart of computing all clustering comparison measures, has been carefully optimized. Hence, even basic R operations (checking for the presence of NAs, type conversion, or constructing a sparse contingency matrix as an output) have non-negligible cost compared to the pair sorting itself. For optimal performance, please provide the vectors as integers or factors without any NAs.

Value

A list containing the following elements:

- **spMat**: A sparsely encoded contingency matrix (only if `spMat = TRUE`).
- **levels**: A list containing the retained levels for each classification.
- **nij**: A vector of positive pair counts.
- **ni., n.j**: Vectors of class counts for `c1` and `c2`, respectively.
- **pair_c1, pair_c2**: Integer vectors specifying the classes in `c1` and `c2` corresponding to the counts in `nij`. These provide the row and column indices for the contingency matrix.

Examples

```
data(iris)
cl <- cutree(hclust(dist(iris[, -5])), 4)
out <- sort_pairs(cl, iris$Species)
```

Index