Package 'c2c'

October 12, 2022

Type Package

Title Compare Two Classifications or Clustering Solutions of Varying Structure

Version 0.1.0

Maintainer Mitchell Lyons <mitchell.lyons@gmail.com>

Description Compare two classifications or clustering solutions that may or may not have the same number of classes, and that might have hard or soft (fuzzy, probabilistic) membership. Calculate various metrics to assess how the clusters compare to each other. The calculations are simple, but provide a handy tool for users unfamiliar with matrix multiplication. This package is not geared towards traditional accuracy assessment for classification/ mapping applications - the motivating use case is for comparing a probabilistic clustering solution to a set of reference or existing class labels that could have any number of classes (that is, without having to degrade the probabilistic clustering to hard classes).

Depends R (>= 3.1.0)

URL https://github.com/mitchest/c2c/

BugReports https://github.com/mitchest/c2c/issues

License GPL-3

Encoding UTF-8

LazyData true

Suggests testthat, knitr, rmarkdown, e1071

RoxygenNote 6.0.1

VignetteBuilder knitr

NeedsCompilation no

Author Mitchell Lyons [aut, cre]

Repository CRAN

Date/Publication 2017-07-23 17:50:40 UTC

R topics documented:

`calculate_clustering_metrics` *Calculate clustering metrics for a confusion matrix*

Description

Calculate a range of clustering metrics on a confusion confusion matrix, usually from `get_conf_mat`.

Usage

```
calculate_clustering_metrics(conf_mat)
```

Arguments

`conf_mat`	a confusion matrix, as produced by `get_conf_mat`, or otherwise a confusion matrix of the same form.

Details

Entropy calculated via `overall_entropy` and `class_entropy`, purity calculated via `overall_purity` and `class_purity`, percentage agreement calculated via `percentage_agreement` (only for confusion matrices of equal dimensions and matching class order)

Value

A list containing the metrics that can be calculated, see details.

Author(s)

Mitchell Lyons

References

Lyons, Foster and Keith (2017). Simultaneous vegetation classification and mapping at large spatial scales. *Journal of Biogeography*.

See Also

get_conf_mat, labels_to_matrix, get_hard

Examples

```
# meaningless data, but you get the idea

# compare two soft classifications
my_soft_mat1 <- matrix(runif(50,0,1), nrow = 10, ncol = 5)
my_soft_mat2 <- matrix(runif(30,0,1), nrow = 10, ncol = 3)
# make the confusion matrix and calculate stats
conf_mat <- get_conf_mat(my_soft_mat1, my_soft_mat2)
conf_mat; calculate_clustering_metrics(conf_mat)

# compare a soft classificaiton to a vector of hard labels
my_labels <- rep(c("a","b","c"), length.out = 10)
# utilising labels_to_matrix(my_labels)
conf_mat <- get_conf_mat(my_soft_mat1, my_labels)
conf_mat; calculate_clustering_metrics(conf_mat)

# make one of the soft matrices hard
# utilising get_hard(my_soft_mat2)
conf_mat <- get_conf_mat(my_soft_mat1, my_soft_mat2, make.B.hard = TRUE)
conf_mat; calculate_clustering_metrics(conf_mat)

# two classifications with same number of classes, enables percentage agreement
conf_mat <- get_conf_mat(my_soft_mat1, my_soft_mat1)
conf_mat; calculate_clustering_metrics(conf_mat)
```

---

class_entropy *Calculate cluster entropy per class*

Description

Used to calculate cluster entropy from a confusion matrix, for each class (i.e. each row and column of the confusion matrix).

Usage

```
class_entropy(conf_mat)
```

Arguments

conf_mat	A confusion matrix from get_conf_mat or otherwise (ideally a matrix, although data frames will probably work)

Details

Metrics per class are useful when you are comparing two classifications with different numbers of classes, when an overall measure might not be useful or sensible. Entropy as defined in Manning (2008).

Value

A data frame with two columns, the first corresponding to the confusion matrix rows, the second corresponding to the confusion matrix columns.

References

Manning, C. D., Raghavan, P., & Schütze, H. (2008) Introduction to information retrieval (Vol. 1, No. 1). Cambridge: Cambridge university press.

`class_purity` *Calculate cluster purity per class*

Description

Used to calculate cluster purity from a confusion matrix, for each class (i.e. each row and column of the confusion matrix).

Usage

```
class_purity(conf_mat)
```

Arguments

`conf_mat`	A confusion matrix from `get_conf_mat` or otherwise (ideally a matrix, although data frames will probably work)

Details

Metrics per class are useful when you are comparing two classifications with different numbers of classes, when an overall measure might not be useful or sensible. Purity as defined in Manning (2008).

Value

A data frame with two columns, the first corresponding to the confusion matrix rows, the second corresponding to the confusion matrix columns.

References

Manning, C. D., Raghavan, P., & Schütze, H. (2008) Introduction to information retrieval (Vol. 1, No. 1). Cambridge: Cambridge university press.

`get_conf_mat` *Generate a confusion matrix from two classifications/clustering solutions.*

Description

`get_conf_mat` takes two classifications or clustering solutions and creates a confusion matrix representing the number of shared sites between them.

Usage

```
get_conf_mat(A, B, make.A.hard = F, make.B.hard = F)
```

Arguments

`A`	A matrix or data.frame (or something that can be coerced to a matrix) of class membership or a vector of class labels (character or factor).
`B`	A matrix or data.frame (or something that can be coerced to a matrix) or class membership or a vector of class labels (character or factor).
`make.A.hard`	logical (defaults to FALSE). If TRUE, and if A= is a matrix of soft membership, it will be degraded to a hard binary matrix, taking the highest value, breaking ties at random
`make.B.hard`	logical (defaults to FALSE). If TRUE, and if B= is a matrix of soft membership, it will be degraded to a hard binary matrix, taking the highest value, breaking ties at random

Details

Takes inputs A and B (converting labels to matrices if required) and combines them via ($A^T B$). Soft classifications will necessarily be matrices. Hard classifications can be given as a binary matrix of membership or a vector of labels. For matrix inputs, rows should represent individual sites, observations, cases etc., and columns should represent classes. For class label inputs, the vector should be ordered similarly by site, observation, case etc; they will be converted to a binary matrix (see `labels_to_matrix`). Classes from matrix A are represented by rows of the output, and classes from matrix B are represented by the columns. Class names inherited from `names()` or `colnames()` - if at least one of the inputs has names, interpretation will be much easier. Ties in membership probability are broken at random - if you don't want this to happen, suggest you break the tie manually before proceeding.

Value

A confusion matrix

Author(s)

Mitchell Lyons

References

Lyons, Foster and Keith (2017). Simultaneous vegetation classification and mapping at large spatial scales. *Journal of Biogeography*.

See Also

calculate_clustering_metrics, labels_to_matrix, get_hard

Examples

```
# meaningless data, but you get the idea

# compare two soft classifications
my_soft_mat1 <- matrix(runif(50,0,1), nrow = 10, ncol = 5)
my_soft_mat2 <- matrix(runif(30,0,1), nrow = 10, ncol = 3)
# make the confusion matrix and calculate stats
conf_mat <- get_conf_mat(my_soft_mat1, my_soft_mat2)
conf_mat; calculate_clustering_metrics(conf_mat)

# compare a soft classification to a vector of hard labels
my_labels <- rep(c("a","b","c"), length.out = 10)
# utilising labels_to_matrix(my_labels)
conf_mat <- get_conf_mat(my_soft_mat1, my_labels)
conf_mat; calculate_clustering_metrics(conf_mat)

# make one of the soft matrices hard
# utilising get_hard(my_soft_mat2)
conf_mat <- get_conf_mat(my_soft_mat1, my_soft_mat2, make.B.hard = TRUE)
conf_mat; calculate_clustering_metrics(conf_mat)

# two classifications with same number of classes, enables percentage agreement
conf_mat <- get_conf_mat(my_soft_mat1, my_soft_mat1)
conf_mat; calculate_clustering_metrics(conf_mat)
```

---

get_hard *Decompose soft (fuzzy, probabilistic) membership to hard binary matrix*

Description

Used in get_conf_mat but might be useful separately

Usage

```
get_hard(x)
```

Arguments

x	A matrix or data frame (or something coercible to a matrix) containing memberships - rows are sites (observations, cases etc.) columns are classes

Value

Binary matrix of class membership. Class names inherited from `names()` or `colnames()`.

Examples

```
my_mat <- matrix(runif(20,0,1), nrow = 4)
get_hard(my_mat)
```

---

labels_to_matrix *Make a vector of class labels into a hard binary matrix*

Description

Used in `get_conf_mat` but might be useful separately

Usage

```
labels_to_matrix(x)
```

Arguments

x	Character or factor vector of class labels

Value

Binary matrix of class membership.

Examples

```
my_labels <- rep(c("a","b","c","d"), 5)
labels_to_matrix(my_labels)
```

overall_entropy *Calculate overall cluster entropy*

Description

Used to calculate overall cluster entropy from a confusion matrix.

Usage

```
overall_entropy(conf_mat)
```

Arguments

conf_mat	A confusion matrix from get_conf_mat or otherwise (ideally a matrix, although data frames will probably work)

Value

A scaler, cluster entropy as defined in Manning (2008)

References

Manning, C. D., Raghavan, P., & Schütze, H. (2008) Introduction to information retrieval (Vol. 1, No. 1). Cambridge: Cambridge university press.

overall_purity *Calculate overall cluster purity*

Description

Used to calculate overall cluster purity from a confusion matrix.

Usage

```
overall_purity(conf_mat)
```

Arguments

conf_mat	A confusion matrix from get_conf_mat or otherwise (ideally a matrix, although data frames will probably work)

Value

A scaler, cluster purity as defined in Manning (2008)

References

Manning, C. D., Raghavan, P., & Schütze, H. (2008) Introduction to information retrieval (Vol. 1, No. 1). Cambridge: Cambridge university press.

`percentage_agreement` *Calculate overall percentage agreement*

Description

Used to calculate overall percentage agreement for a confusion matrix - the confusion matrix must have equal dimensions and the diagonal must represent 'matching' class pairs (percentage agreement does not make sense otherwise)

Usage

```
percentage_agreement(conf_mat)
```

Arguments

`conf_mat`	A confusion matrix from `get_conf_mat` or otherwise (ideally a matrix, although data frames will probably work)

Value

A scaler, percentage agreement (sometime referred to as overall accuracy)

Index