Package ‘catsim’

July 22, 2025

Type Package

Title Binary and Categorical Image Similarity Index

Version 0.2.4

Description Computes a structural similarity metric (after the style of MS-SSIM for images) for binary and categorical 2D and 3D images. Can be based on accuracy (simple matching), Cohen's kappa, Rand index, adjusted Rand index, Jaccard index, Dice index, normalized mutual information, or adjusted mutual information. In addition, has fast computation of Cohen's kappa, the Rand indices, and the two mutual informations. Implements the methods of Thompson and Maitra (2020) <doi:10.48550/arXiv.2004.09073>.

License GPL-3

Encoding UTF-8

LazyData true

Imports Rcpp

RoxygenNote 7.3.2

URL `https://gzt.github.io/catsim/`

BugReports `https://github.com/gzt/catsim/issues`

LinkingTo Rcpp, testthat

Suggests testthat, covr, knitr, rmarkdown

VignetteBuilder knitr

Depends R (>= 3.6)

NeedsCompilation yes

Author Geoffrey Thompson [aut, cre] (ORCID: <`https://orcid.org/0000-0003-2436-8822`>)

Maintainer Geoffrey Thompson <`gzthompson@gmail.com`>

Repository CRAN

Date/Publication 2024-10-01 05:00:02 UTC

Contents

besag 2
binssim 3
catmssim_2d 4
catmssim_3d_cube 5
catmssim_3d_slice 6
catsim 8
gini 9
hoffmanphantom 10
rand 11

Index **14**

`besag` *A hand-constructed image from Besag (1986)*

Description

An 100×88 matrix representing a two-color hand-drawn scene designed specifically to contain some awkward features for an image reconstruction method evaluated in the paper.

Usage

```
data(besag)
```

Format

an 100×88 matrix with entries `1` and `2` denoting the color of the corresponding pixels. The example code will produce the image as it is in the original paper. To use as a `0-1` binary dataset, either use `besag - 1` or `besag %% 2`.

References

J. Besag, "On the statistical analysis of dirty pictures," Journal of the Royal Statistical Society: Series B (Methodological), vol. 48, no. 3, pp. 259–279, 1986. doi:10.1111/j.25176161.1986.tb01412.x

Examples

```
image(besag[, 88:1])
```

`binssim` *Categorical Structural Similarity Index Metric (whole image)*

Description

This computes the categorical or binary structural similarity index metric on a whole-image scale. The difference between this and the default 2-D method is that this considers the whole image at once and one scale rather than computing the index over a sliding window and downsampling to consider it at other scales.

Usage

```
binssim(
  x,
  y,
  alpha = 1,
  beta = 1,
  gamma = 1,
  c1 = 0.01,
  c2 = 0.01,
  method = "Cohen",
  ...
)
```

Arguments

`x, y`	binary or categorical image
`alpha`	normalizing parameter, by default 1
`beta`	normalizing parameter, by default 1
`gamma`	normalizing parameter, by default 1
`c1`	small normalization constant for the `c` function, by default 0.01
`c2`	small normalization constant for the `s` function, by default 0.01
`method`	whether to use Cohen's kappa (`Cohen`), Jaccard Index (`Jaccard`), Dice index (`Dice`), accuracy (`accuracy`), Rand index (`Rand`), Adjusted Rand Index (`AdjRand` or `ARI`), or normalized mutual information (`NMI` or `MI`) as the similarity index. Note Jaccard and Dice should only be used on binary data.
`...`	Constants can be passed to the components of the index.

Value

Structural similarity index.

Examples

```
set.seed(20181207)
x <- matrix(sample(1:4, 10000, replace = TRUE), nrow = 100)
y <- x
for (i in 1:100) y[i, i] <- 1
for (i in 1:99) y[i, i + 1] <- 1
binssim(x, y)
```

---

catmssim_2d *Multiscale Categorical Structural Similarity Index Measure (2D)*

Description

The categorical structural similarity index measure for 2D categorical or binary images for multiple scales. The default is to compute over 5 scales.

Usage

```
catmssim_2d(
  x,
  y,
  levels = NULL,
  weights = NULL,
  window = 11,
  method = "Cohen",
  ...,
  random = "random"
)
```

Arguments

`x, y`	a binary or categorical image
`levels`	how many levels of downsampling to use. By default, 5. If `weights` is specified and this is left blank, the argument will be inferred from the number of weights specified.
`weights`	a vector of weights for the different scales. By default, equal to `rep(1,levels)/levels`. If specified, there must at least as many weights as there are levels and the first `levels` weights will be used.
`window`	by default 11 for 2D and 5 for 3D images, but can be specified as a vector if the window sizes differ by dimension. The vector must have the same number of dimensions as the inputted `x` and `y`.
`method`	whether to use Cohen's kappa (`Cohen`), Jaccard Index (`Jaccard`), Dice index (`Dice`), accuracy (`accuracy`), Rand index (`Rand`), Adjusted Rand Index (`AdjRand` or `ARI`), normalized mutual information (`NMI` or `MI`) or the adjusted mutual information, `AMI` and `ami`, as the similarity index. Note Jaccard and Dice should only be used on binary data.

`...`	additional constants can be passed to internal functions.
`random`	whether to have deterministic PRNG (`pseudo`) or to use `sample()` (`random`). If `NULL`, will choose the first mode. For complete reproducibility, use `pseudo` or `NULL`.

Value

a value less than 1 indicating the similarity between the images.

Examples

```
set.seed(20181207)
x <- matrix(sample(0:3, 128^2, replace = TRUE), nrow = 128)
y <- x
for (i in 1:128) y[i, i] <- 0
for (i in 1:127) y[i, i + 1] <- 0
catmssim_2d(x, y, method = "Cohen", levels = 2) # the default
# now using a different similarity score (Jaccard Index)
catmssim_2d(x, y, method = "NMI")
```

`catmssim_3d_cube` *Multiscale Categorical Structural Similarity Index Measure for a Cube (3D)*

Description

The categorical structural similarity index measure for 3D categorical or binary images for multiple scales. The default is to compute over 5 scales. This computes a 3D measure based on $5 \times 5 \times 5$ windows by default with 5 levels of downsampling.

Usage

```
catmssim_3d_cube(
  x,
  y,
  levels = NULL,
  weights = NULL,
  window = 5,
  method = "Cohen",
  ...,
  random = "random"
)
```

Arguments

`x, y`	a binary or categorical image
`levels`	how many levels of downsampling to use. By default, 5. If `weights` is specified and this is left blank, the argument will be inferred from the number of weights specified.
`weights`	a vector of weights for the different scales. By default, equal to `rep(1,levels)/levels`. If specified, there must at least as many weights as there are levels and the first `levels` weights will be used.
`window`	by default 11 for 2D and 5 for 3D images, but can be specified as a vector if the window sizes differ by dimension. The vector must have the same number of dimensions as the inputted `x` and `y`.
`method`	whether to use Cohen's kappa (`Cohen`), Jaccard Index (`Jaccard`), Dice index (`Dice`), accuracy (`accuracy`), Rand index (`Rand`), Adjusted Rand Index (`AdjRand` or `ARI`), normalized mutual information (`NMI` or `MI`) or the adjusted mutual information, `AMI` and `ami`, as the similarity index. Note Jaccard and Dice should only be used on binary data.
`...`	additional constants can be passed to internal functions.
`random`	whether to have deterministic PRNG (`pseudo`) or to use `sample()` (`random`). If `NULL`, will choose the first mode. For complete reproducibility, use `pseudo` or `NULL`.

Value

a value less than 1 indicating the similarity between the images.

Examples

```
set.seed(20181207)
dim <- 16
x <- array(sample(0:4, dim^3, replace = TRUE), dim = c(dim, dim, dim))
y <- x
for (j in 1:dim) {
  for (i in 1:dim) y[i, i, j] <- 0
  for (i in 1:(dim - 1)) y[i, i + 1, j] <- 0
}
catmssim_3d_cube(x, y, weights = c(.75, .25))
# Now using a different similarity score
catmssim_3d_cube(x, y, weights = c(.75, .25), method = "Accuracy")
```

`catmssim_3d_slice` *Multiscale Categorical Structural Similarity Index Measure by Slice (3D)*

Description

The categorical structural similarity index measure for 3D categorical or binary images for multiple scales. The default is to compute over 5 scales. This computes a 2D measure for each x-y slice of the z-axis and then averages over the z-axis.

Usage

```
catmssim_3d_slice(
  x,
  y,
  levels = NULL,
  weights = NULL,
  window = 11,
  method = "Cohen",
  ...,
  random = "random"
)
```

Arguments

`x, y`	a binary or categorical image
`levels`	how many levels of downsampling to use. By default, 5. If `weights` is specified and this is left blank, the argument will be inferred from the number of weights specified.
`weights`	a vector of weights for the different scales. By default, equal to `rep(1,levels)/levels`. If specified, there must at least as many weights as there are levels and the first `levels` weights will be used.
`window`	by default 11 for 2D and 5 for 3D images, but can be specified as a vector if the window sizes differ by dimension. The vector must have the same number of dimensions as the inputted `x` and `y`.
`method`	whether to use Cohen's kappa (`Cohen`), Jaccard Index (`Jaccard`), Dice index (`Dice`), accuracy (`accuracy`), Rand index (`Rand`), Adjusted Rand Index (`AdjRand` or `ARI`), normalized mutual information (`NMI` or `MI`) or the adjusted mutual information, `AMI` and `ami`, as the similarity index. Note Jaccard and Dice should only be used on binary data.
`...`	additional constants can be passed to internal functions.
`random`	whether to have deterministic PRNG (`pseudo`) or to use `sample()` (`random`). If `NULL`, will choose the first mode. For complete reproducibility, use `pseudo` or `NULL`.

Value

a value less than 1 indicating the similarity between the images.

Examples

```
set.seed(20181207)
dim <- 8
x <- array(sample(0:4, dim^5, replace = TRUE), dim = c(dim^2, dim^2, dim))
y <- x
for (j in 1:(dim)) {
  for (i in 1:(dim^2)) y[i, i, j] <- 0
  for (i in 1:(dim^2 - 1)) y[i, i + 1, j] <- 0
}
catmssim_3d_slice(x, y, weights = c(.75, .25)) # by default method = "Cohen"
# compare to some simple metric:
mean(x == y)
```

---

catsim *Multiscale Categorical Structural Similarity Index Measure*

Description

The categorical structural similarity index measure for 2D or 3D categorical or binary images for multiple scales. The default is to compute over 5 scales. This determines whether this is a 2D or 3D image and applies the appropriate windowing, weighting, and scaling. Additional arguments can be passed. This is a wrapper function for the 2D and 3D functions whose functionality can be accessed through the ... arguments. This function is a wrapper for the `catmssim_2d()`, `catmssim_3d_slice()`, and `catmssim_3d_cube()` functions.

Usage

```
catsim(
  x,
  y,
  ...,
  cube = TRUE,
  levels = NULL,
  weights = NULL,
  method = "Cohen",
  window = NULL
)
```

Arguments

x, y	a binary or categorical image
...	additional arguments, such as window, can be passed as well as arguments for internal functions.
cube	for the 3D method, whether to use the true 3D method (cube or `catmssim_3d_cube()`) or compute the metric using 2D slices which are then averaged (`catmssim_3d_slice()`). By default, `TRUE`, which evaluates as a cube. `FALSE` will treat it as 2D slices.

`levels`	how many levels of downsampling to use. By default, 5. If `weights` is specified and this is left blank, the argument will be inferred from the number of weights specified.
`weights`	a vector of weights for the different scales. By default, equal to `rep(1,levels)/levels`. If specified, there must at least as many weights as there are levels and the first `levels` weights will be used.
`method`	whether to use Cohen's kappa (`Cohen`), Jaccard Index (`Jaccard`), Dice index (`Dice`), accuracy (`accuracy`), Rand index (`Rand`), Adjusted Rand Index (`AdjRand` or `ARI`), normalized mutual information (`NMI` or `MI`), or adjusted mutual information (`AMI`) as the similarity index. Note Jaccard and Dice should only be used on binary data.
`window`	by default 11 for 2D and 5 for 3D images, but can be specified as a vector if the window sizes differ by dimension. The vector must have the same number of dimensions as the inputted `x` and `y`.

Value

a value less than 1 indicating the similarity between the images.

Examples

```
set.seed(20181207)
dim <- 16
x <- array(sample(0:4, dim^3, replace = TRUE), dim = c(dim, dim, dim))
y <- x
for (j in 1:dim) {
  for (i in 1:dim) y[i, i, j] <- 0
  for (i in 1:(dim - 1)) y[i, i + 1, j] <- 0
}
catsim(x, y, weights = c(.75, .25))
# Now using a different similarity score
catsim(x, y, levels = 2, method = "accuracy")
# with the slice method:
catsim(x, y, weights = c(.75, .25), cube = FALSE, window = 8)
```

---

gini *Diversity Indices*

Description

`gini()` is a measure of diversity that goes by a number of different names, such as the probability of interspecific encounter or the Gibbs-Martin index. It is $1 - sum(p_i^2)$, where p_i is the probability of observing class i.

The corrected Gini-Simpson index, `ginicorr` takes the index and corrects it so that the maximum possible is 1. If there are `k` categories, the maximum possible of the uncorrected index is $1 - 1/k$. It corrects the index by dividing by the maximum. `k` must be specified.

The modified Gini-Simpson index is similar to the unmodified, except it uses the square root of the summed squared probabilities, that is, $1 - \sqrt{sum(p_i^2)}$, where p_i is the probability of observing class i.

The modified corrected Gini index then corrects the modified index for the number of categories, k.

Usage

```
gini(x)

ginicorr(x, k)

sqrtgini(x)

sqrtginicorr(x, k)
```

Arguments

x	binary or categorical image or vector
k	number of categories

Value

The index (between 0 and 1), with 0 indicating no variation and 1 being maximal. The Gini index is bounded above by $1 - 1/k$ for a group with k categories. The modified index is bounded above by $1 - 1/\sqrt{k}$. The corrected indices fix this by dividing by the maximum.

Examples

```
x <- rep(c(1:4), 5)
gini(x)

x <- rep(c(1:4), 5)
ginicorr(x, 4)

x <- rep(c(1:4), 5)
sqrtgini(x)

x <- rep(c(1:4), 5)
sqrtginicorr(x, 4)
```

---

hoffmanphantom *An example fMRI phantom*

Description

A 128×128 activation map for a slice of an fMRI phantom and an anatomical reference.

Usage

```
data(hoffmanphantom)
```

Format

a $128 \times 128 \times 2$ array with the first slice an activation map for an MRI phantom and the second an anatomical overlay. `NA` values are outside the surface. The activation map (`hoffmanphantom[,,1]`) is 1 if activated, 0 otherwise. The second layer (`hoffmanphantom[,,2]`) indicates the anatomical structure. Approximately 3.8 percent of the pixels are activated in this slice.

References

E. Hoffman, P. Cutler, W. Digby, and J. Mazziotta, “3-D phantom to simulate cerebral blood flow and metabolic images for PET,” Nuclear Science, IEEE Transactions on, vol. 37, pp. 616 – 620, 05 1990.

I. A. Almodóvar-Rivera and R. Maitra, “FAST adaptive smoothed thresholding for improved activation detection in low-signal fMRI,” IEEE Transactions on Medical Imaging, vol. 38, no. 12, pp. 2821–2828, 2019.

Examples

```
image(hoffmanphantom[, , 2], col = rev(gray(0:15 / 16))[1:4], axes = FALSE)
image(hoffmanphantom[, , 1],
  add = TRUE, zlim = c(0.01, 1),
  col = c("yellow", "maroon")
)
```

`rand` *Similarity Indices*

Description

The Rand index, rand_index, computes the agreement between two different clusterings or partitions of the same set of objects. The inputs to the function should be binary or categorical and of the same length.

The adjusted Rand index, `adj_rand`, computes a corrected version of the Rand index, adjusting for the probability of chance agreement of clusterings. A small constant is added to the numerator and denominator of the adjusted Rand index to ensure stability when there is a small or 0 denominator, as it is possible to have a zero denominator.

Cohen’s kappa, `cohen_kappa`, is an inter-rater agreement metric for two raters which corrects for the probability of chance agreement. Note there is a difference here between this measure and the Rand indices and mutual information: those consider the similarities of the groupings of points, while this considers how often the raters agreed on individual points.

Like the Rand index, the mutual information computes the agreement between two different clusterings or partitions of the same set of objects. If $H(X)$ is the entropy of some probability distribution X, then the mutual information of two distributions is $I(X;Y) = -H(X,Y)+H(X)+H(Y)$. The normalized mutual information, `normalized_mi`, is defined here as: $2I(X;Y)/(H(X)+H(Y))$, but is set to be 0 if both H(X) and H(Y) are 0.

The adjusted mutual information, `adjusted_mi`, is a correction of the mutual information to account for the probability of chance agreement in a manner similar to the adjusted Rand index or Cohen's kappa.

Usage

```
rand_index(x, y, na.rm = FALSE)

adj_rand(x, y, na.rm = FALSE)

cohen_kappa(x, y, na.rm = FALSE)

normalized_mi(x, y, na.rm = FALSE)

adjusted_mi(x, y, na.rm = FALSE)
```

Arguments

`x, y`	a numeric or factor vector or array
`na.rm`	whether to remove `NA` values. By default, `FALSE`. If `TRUE`, will perform pair-wise deletion.

Value

the similarity index, which is between 0 and 1 for most of the options. The adjusted Rand and Cohen's kappa can be negative, but are bounded above by 1.

References

W. M. Rand (1971). "Objective criteria for the evaluation of clustering methods". Journal of the American Statistical Association. American Statistical Association. 66 (336): 846–850. doi:10.2307/2284239

Lawrence Hubert and Phipps Arabie (1985). "Comparing partitions". Journal of Classification. 2 (1): 193–218. doi:10.1007/BF01908075

Cohen, Jacob (1960). "A coefficient of agreement for nominal scales". Educational and Psychological Measurement. 20 (1): 37–46. doi:10.1177/001316446002000104

Jaccard, Paul (1912). "The distribution of the flora in the alpine zone," New Phytologist, vol. 11, no. 2, pp. 37–50. doi:10.1111/j.14698137.1912.tb05611.x

Nguyen Xuan Vinh, Julien Epps, and James Bailey (2010). Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance. J. Mach. Learn. Res. 11 (December 2010), 2837–2854. `https://jmlr.org/papers/v11/vinh10a.html`

Examples

```
x <- rep(0:5, 5)
y <- c(rep(0:5, 4), rep(0, 6))
# Simple Matching, or Accuracy
mean(x == y)
# Hamming distance
sum(x != y)
rand_index(x, y)
adj_rand(x, y)
cohen_kappa(x, y)
normalized_mi(x, y)
adjusted_mi(x, y)
```

Index

∗ **datasets**
    besag, 2
    hoffmanphantom, 10

adj_rand *(rand)*, 11
Adjusted Mutual Information *(rand)*, 11
Adjusted Rand Index *(rand)*, 11
adjusted_mi *(rand)*, 11

besag, 2
binssim, 3

catmssim_2d, 4
catmssim_2d(), *8*
catmssim_3d_cube, 5
catmssim_3d_cube(), *8*
catmssim_3d_slice, 6
catmssim_3d_slice(), *8*
catsim, 8
Cohen's kappa *(rand)*, 11
cohen_kappa *(rand)*, 11
Corrected Gini-Simpson index *(gini)*, 9

gini, 9
ginicorr *(gini)*, 9

hoffmanphantom, 10

Modified Corrected Gini index *(gini)*, 9
Modified Gini-Simpson index *(gini)*, 9

Normalized Mutual Information *(rand)*, 11
normalized_mi *(rand)*, 11

rand, 11
rand_index, *11*
rand_index *(rand)*, 11

sample(), *5–7*
sqrtgini *(gini)*, 9
sqrtginicorr *(gini)*, 9