Package: gtexture (via r-universe)

September 6, 2024

Title Generalized Application of Co-Occurrence Matrices and Haralick Texture

Version 0.1.0

Description Generalizes application of gray-level co-occurrence matrix (GLCM) metrics to objects outside of images. The current focus is to apply GLCM metrics to the study of biological networks and fitness landscapes that are used in studying evolutionary medicine and biology, particularly the evolution of cancer resistance. The package was used in our publication, Barker-Clarke et al. (2023) <doi:10.1088/1361-6560/ace305>. A general reference to learn more about mathematical oncology can be found at Rockne et al. (2019) <doi:10.1088/1478-3975/ab1a09>.

License MIT + file LICENSE

URL https://rbarkerclarke.github.io/gtexture/

BugReports https://github.com/rbarkerclarke/gtexture/issues

Imports dlookr, dplyr (>= 1.0), fitscape (>= 0.1), igraph, magrittr (>= 2.0), rlang, tidyr

Suggests stats, testthat

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Repository https://rbarkerclarke.r-universe.dev

RemoteUrl https://github.com/rbarkerclarke/gtexture

RemoteRef HEAD

RemoteSha a70421439f3fb2d2d147862a3df1e3e47617f1e1

Contents

`autocorrelation` *Autocorrelation Metric for a GLCM*

Description

Calculate the autocorrelation feature or metric for a gray-level co-occurrence matrix. For definition and application, see Lofstedt et al. (2019) doi:10.1371/journal.pone.0212110.

Usage

```
autocorrelation(x, ...)

## Default S3 method:
autocorrelation(x, ...)

## S3 method for class 'matrix'
autocorrelation(x, ...)

## S3 method for class 'FitLandDF'
autocorrelation(x, nlevels, ...)
```

Arguments

`x`	gray-level co-occurrence matrix
`...`	additional parameters
`nlevels`	desired number of discrete gray levels

Value

autocorrelation metric of x

Examples

```
## calculate autocorrelation of arbitrary GLCM
# define arbitrary GLCM
x <- matrix(1:16, nrow = 4)

# normalize
n_x <- normalize_glcm(x)

# calculate autocorrelation
autocorrelation(n_x)

## calculate autocorrelation of arbitrary fitness landscape
# create fitness landscape using FitLandDF object
vals <- runif(64)
vals <- array(vals, dim = rep(4, 3))
my_landscape <- fitscape::FitLandDF(vals)

# calculate autocorrelation of fitness landscape, assuming 2 discrete gray levels
autocorrelation(my_landscape, nlevels = 2)

## confirm value of autocorrelation for fitness landscape
# extract normalized GLCM from fitness landscape
my_glcm <- get_comatrix(my_landscape, discrete = equal_discrete(2))

# calculate autocorrelation of extracted GLCM
autocorrelation(my_glcm)  # should match value of above autocorrelation function call
```

---

cluster_prom *Cluster Prominence Metric for a GLCM*

Description

Calculate the cluster prominence feature or metric for a gray-level co-occurrence matrix. For definition and application, see Lofstedt et al. (2019) doi:10.1371/journal.pone.0212110.

Usage

```
cluster_prom(x, ...)

## Default S3 method:
cluster_prom(x, ...)

## S3 method for class 'matrix'
cluster_prom(x, ...)
```

```
## S3 method for class 'FitLandDF'
cluster_prom(x, nlevels, ...)
```

Arguments

x	gray-level co-occurrence matrix
...	additional parameters
nlevels	desired number of discrete gray levels

Value

cluster prominence metric of x

Examples

```
## calculate cluster prominence of arbitrary GLCM
# define arbitrary GLCM
x <- matrix(1:16, nrow = 4)

# normalize
n_x <- normalize_glcm(x)

# calculate cluster prominence
cluster_prom(n_x)

## calculate cluster prominence of arbitrary fitness landscape
# create fitness landscape using FitLandDF object
vals <- runif(64)
vals <- array(vals, dim = rep(4, 3))
my_landscape <- fitscape::FitLandDF(vals)

# calculate cluster prominence of fitness landscape, assuming 2 discrete gray levels
cluster_prom(my_landscape, nlevels = 2)

## confirm value of cluster prominence for fitness landscape
# extract normalized GLCM from fitness landscape
my_glcm <- get_comatrix(my_landscape, discrete = equal_discrete(2))

# calculate cluster prominence of extracted GLCM
cluster_prom(my_glcm)  # should match value of above cluster_prom function call
```

---

cluster_shade *Cluster Shade Metric for a GLCM*

Description

Calculate the cluster shade feature or metric for a gray-level co-occurrence matrix. For definition and application, see Lofstedt et al. (2019) doi:10.1371/journal.pone.0212110.

Usage

```
cluster_shade(x, ...)

## Default S3 method:
cluster_shade(x, ...)

## S3 method for class 'matrix'
cluster_shade(x, ...)

## S3 method for class 'FitLandDF'
cluster_shade(x, nlevels, ...)
```

Arguments

x	gray-level co-occurrence matrix
...	additional parameters
nlevels	desired number of discrete gray levels

Value

cluster shade metric of x

Examples

```
## calculate cluster shade of arbitrary GLCM
# define arbitrary GLCM
x <- matrix(1:16, nrow = 4)

# normalize
n_x <- normalize_glcm(x)

# calculate cluster shade
cluster_shade(n_x)

## calculate cluster shade of arbitrary fitness landscape
# create fitness landscape using FitLandDF object
vals <- runif(64)
vals <- array(vals, dim = rep(4, 3))
my_landscape <- fitscape::FitLandDF(vals)

# calculate cluster shade of fitness landscape, assuming 2 discrete gray levels
cluster_shade(my_landscape, nlevels = 2)

## confirm value of cluster shade for fitness landscape
# extract normalized GLCM from fitness landscape
my_glcm <- get_comatrix(my_landscape, discrete = equal_discrete(2))

# calculate cluster shade of extracted GLCM
cluster_shade(my_glcm)  # should match value of above cluster_shade function call
```

comat *Calculate Co-Occurrence Matrix*

Description

Calculate generalized co-occurrence matrix from a variety of objects, currently including fitness landscapes stored as a `FitLandDF` instance from the `fitscape` package.

Usage

```
get_comatrix(x, ...)

## Default S3 method:
get_comatrix(x, ...)

## S3 method for class 'FitLandDF'
get_comatrix(
  x,
  discrete = equal_discrete(2),
  neighbor = manhattan(1),
  normalize = normalize_glcm,
  ...
)

## S3 method for class 'igraph'
get_comatrix(
  x,
  values,
  nlevels = length(unique(values)),
  normalize = normalize_glcm,
  verbose = TRUE,
  ...
)
```

Arguments

`x`	object upon which co-occurrence matrix will be calculated
`...`	additional arguments
`discrete`	function that discretizes object
`neighbor`	function that returns TRUE if two numeric vectors are within acceptable distance of one another or a single-element `character` vector that describes how to identify acceptable neighbors/offsets
`normalize`	function that normalizes the co-occurrence matrix
`values`	named numeric with values corresponding to the nodes in `x`.
`nlevels`	int number of levels to discretize into
`verbose`	bool

Value

co-occurrence matrix

Examples

```
# create fitness landscape as instance of FitLandDF object
a <- round(runif(64))
a <- array(a, dim = rep(4, 3))
my_landscape <- fitscape::FitLandDF(a)

# calculate co-occurrence matrix using:
#   Manhattan distance of 1
#   discretization into 2 equal-sized buckets
#   normalization: multiply all elements so that sum of matrix equals unity
comat <- get_comatrix(my_landscape,
                      discrete = equal_discrete(2),
                      neighbor = manhattan(1))

# print co-occurrence matrix
print(comat)
```

---

compute_all_metrics *Convenience function to compute all haralick texture features for a given comat*

Description

Convenience function to compute all haralick texture features for a given comat

Usage

```
compute_all_metrics(x)
```

Arguments

x	matrix computed glcm

Value

data.frame

contrast *Contrast Metric for a GLCM*

Description

Calculate the contrast feature or metric for a gray-level co-occurrence matrix. For definition and application, see Lofstedt et al. (2019) doi:10.1371/journal.pone.0212110.

Usage

```
contrast(x, ...)

## Default S3 method:
contrast(x, ...)

## S3 method for class 'matrix'
contrast(x, ...)

## S3 method for class 'FitLandDF'
contrast(x, nlevels, ...)
```

Arguments

x	gray-level co-occurrence matrix
...	additional parameters
nlevels	desired number of discrete gray levels

Value

contrast metric of x

Examples

```
## calculate contrast of arbitrary GLCM
# define arbitrary GLCM
x <- matrix(1:16, nrow = 4)

# normalize
n_x <- normalize_glcm(x)

# calculate contrast
contrast(n_x)

# calculate contrast of fitness landscape, assuming 2 discrete gray levels
vals <- runif(64)
vals <- array(vals, dim = rep(4, 3))
my_landscape <- fitscape::FitLandDF(vals)
```

```
my_glcm <- get_comatrix(my_landscape, discrete = equal_discrete(2))
contrast(my_landscape, nlevels = 2)

## confirm value of contrast for fitness landscape
# extract normalized GLCM from fitness landscape

contrast(my_glcm)  # should match value of above contrast function call
```

`correlation` *Correlation Metric for a GLCM*

Description

Calculate the correlation feature or metric for a gray-level co-occurrence matrix. For definition and application, see Lofstedt et al. (2019) doi:10.1371/journal.pone.0212110.

Usage

```
correlation(x, ...)

## Default S3 method:
correlation(x, ...)

## S3 method for class 'matrix'
correlation(x, ...)

## S3 method for class 'FitLandDF'
correlation(x, nlevels, ...)
```

Arguments

`x`	gray-level co-occurrence matrix
`...`	additional parameters
`nlevels`	desired number of discrete gray levels

Value

correlation metric of `x`

Examples

```
## calculate correlation of arbitrary GLCM
# define arbitrary GLCM
x <- matrix(1:16, nrow = 4)

# normalize
n_x <- normalize_glcm(x)
```

```
# calculate correlation
correlation(n_x)

## calculate autocorrelation of arbitrary fitness landscape
# create fitness landscape using FitLandDF object
vals <- runif(64)
vals <- array(vals, dim = rep(4, 3))
my_landscape <- fitscape::FitLandDF(vals)

# calculate correlation of fitness landscape, assuming 2 discrete gray levels
correlation(my_landscape, nlevels = 2)

## confirm value of correlation for fitness landscape
# extract normalized GLCM from fitness landscape
my_glcm <- get_comatrix(my_landscape, discrete = equal_discrete(2))

# calculate correlation of extracted GLCM
correlation(my_glcm)  # should match value of above correlation function call
```

`discretize` *Discretize Numeric Variable Into Categories*

Description

Takes a numeric variable (could be of class `numeric` or `integer`) and returns a discretized version, in which each element has been replaced by a single integer between `1` and `nlevels`, inclusive.

Usage

```
discretize(x, ...)

## S3 method for class 'numeric'
discretize(x, nlevels, method = "equal", ...)

## S3 method for class 'list'
discretize(x, nlevels, ...)

## S3 method for class 'integer'
discretize(x, nlevels, ...)

## S3 method for class 'FitLandDF'
discretize(x, nlevels, ...)
```

Arguments

`x`	either a vector (`numeric` or `integer`) or `FitLandDF` object
`...`	potential additional arguments, currently unnecessary
`nlevels`	positive integer indicating number of discrete categories

method	method by which to discretize; split into equal sections by default ("equal" value for parameter)

Value

discretized form of x

Examples

```
## discretize a numeric vector
vec <- 1:10
discretize(vec, nlevels = 5) # discretize into 5 categories
discretize(vec, 2)           # discretize into 2 categories

## discretize a fitness landscape
# create a 3x3x3 fitness landscape with values 1 through 27
fl_data <- array(1:27, dim = rep(3, 3))
my_fl <- fitscape::FitLandDF(fl_data)
discretize(my_fl, nlevels = 2) # discretize landscape into 2 categories
discretize(my_fl, 5)           # discretize landscape into 5 categories
```

---

energy *Energy Metric for a GLCM*

Description

Calculate the energy feature or metric for a gray-level co-occurrence matrix. For definition and application, see Lofstedt et al. (2019) doi:10.1371/journal.pone.0212110.

Usage

```
energy(x, ...)

## Default S3 method:
energy(x, ...)

## S3 method for class 'matrix'
energy(x, ...)

## S3 method for class 'FitLandDF'
energy(x, nlevels, ...)
```

Arguments

x	gray-level co-occurrence matrix
...	additional parameters
nlevels	desired number of discrete gray levels

Value

energy metric of x

Examples

```
## calculate energy of arbitrary GLCM
# define arbitrary GLCM
x <- matrix(1:16, nrow = 4)

# normalize
n_x <- normalize_glcm(x)

# calculate energy
energy(n_x)

## calculate energy of arbitrary fitness landscape
# create fitness landscape using FitLandDF object
vals <- runif(64)
vals <- array(vals, dim = rep(4, 3))
my_landscape <- fitscape::FitLandDF(vals)

# calculate energy of fitness landscape, assuming 2 discrete gray levels
energy(my_landscape, nlevels = 2)

## confirm value of energy for fitness landscape
# extract normalized GLCM from fitness landscape
my_glcm <- get_comatrix(my_landscape, discrete = equal_discrete(2))

# calculate energy of extracted GLCM
energy(my_glcm)  # should match value of above energy function call
```

---

entropy *Entropy Metric for a GLCM*

Description

Calculate the entropy feature or metric for a gray-level co-occurrence matrix. For definition and application, see Lofstedt et al. (2019) doi:10.1371/journal.pone.0212110.

Usage

```
entropy(x, ...)

## Default S3 method:
entropy(x, ...)

## S3 method for class 'matrix'
entropy(x, ...)
```

```
## S3 method for class 'FitLandDF'
entropy(x, nlevels, ...)
```

Arguments

x	gray-level co-occurrence matrix
...	additional parameters
nlevels	desired number of discrete gray levels

Value

entropy metric of x

Examples

```
## calculate entropy of arbitrary GLCM
# define arbitrary GLCM
x <- matrix(1:16, nrow = 4)

# normalize
n_x <- normalize_glcm(x)

# calculate entropy
entropy(n_x)

## calculate entropy of arbitrary fitness landscape
# create fitness landscape using FitLandDF object
vals <- runif(64)
vals <- array(vals, dim = rep(4, 3))
my_landscape <- fitscape::FitLandDF(vals)

# calculate entropy of fitness landscape, assuming 2 discrete gray levels
entropy(my_landscape, nlevels = 2)

## confirm value of entropy for fitness landscape
# extract normalized GLCM from fitness landscape
my_glcm <- get_comatrix(my_landscape, discrete = equal_discrete(2))

# calculate entropy of extracted GLCM
entropy(my_glcm)  # should match value of above entropy function call
```

---

equal_discrete *Function Factory for Even Discretization Functions*

Description

Returns a function that converts a continuous numeric vector into an integer vector with discrete levels.

Usage

```
equal_discrete(nlevels)
```

Arguments

nlevels	number of levels to split continuous vector into

Value

function that makes a numeric vector discrete

Examples

```
# test data
x <- 1:10

# create and apply function to split x into 2 discrete levels
split_2 <- equal_discrete(2)
split_2(x)

# create and apply function to split x into 5 discrete levels
split_5 <- equal_discrete(5)
split_5(x)
```

---

euclidean *Euclidean Distance Function Factory*

Description

Returns a function that checks whether the Euclidean distance between two numeric vectors is less than or equal to a given threshold.

Usage

```
euclidean(dist = 1)
```

Arguments

dist	threshold above which the function will return FALSE

Value

function that checks if Euclidean distance between two vectors exceeds dist

Examples

```
# test data: Euclidean distance equals sqrt(2) ~ 1.414
x <- rep(0, 5)
y <- c(0, 1, 0, 0, 1)

# should return TRUE when checking Manhattan distance <= 2
dist_2 <- euclidean(2)
dist_2(x, y)

# should return FALSE when checking Manhattan distance <= 1
dist_1 <- euclidean(1)
dist_1(x, y)
```

`glcm_metrics` *GLCM Metrics*

Description

GLCM Metrics

Usage

```
xplusy_k(glcm, k)

glcm_variance(glcm)
```

Arguments

`glcm`	gray level co-occurrence matrix
`k`	summation equal to k

Functions

- `glcm_variance()`: Variance

`homogeneity` *Homogeneity Metric for a GLCM*

Description

Calculate the homogeneity feature or metric for a gray-level co-occurrence matrix. For definition and application, see Lofstedt et al. (2019) doi:10.1371/journal.pone.0212110.

Usage

```
homogeneity(x, ...)

## Default S3 method:
homogeneity(x, ...)

## S3 method for class 'matrix'
homogeneity(x, ...)

## S3 method for class 'FitLandDF'
homogeneity(x, nlevels, ...)
```

Arguments

x	gray-level co-occurrence matrix
...	additional parameters
nlevels	desired number of discrete gray levels

Value

homogeneity metric of x

Examples

```
## calculate homogeneity of arbitrary GLCM
# define arbitrary GLCM
x <- matrix(1:16, nrow = 4)

# normalize
n_x <- normalize_glcm(x)

# calculate homogeneity
homogeneity(n_x)

## calculate homogeneity of arbitrary fitness landscape
# create fitness landscape using FitLandDF object
vals <- runif(64)
vals <- array(vals, dim = rep(4, 3))
my_landscape <- fitscape::FitLandDF(vals)

# calculate homogeneity of fitness landscape, assuming 2 discrete gray levels
homogeneity(my_landscape, nlevels = 2)

## confirm value of homogeneity for fitness landscape
# extract normalized GLCM from fitness landscape
my_glcm <- get_comatrix(my_landscape, discrete = equal_discrete(2))

# calculate homogeneity of extracted GLCM
homogeneity(my_glcm)  # should match value of above homogeneity function call
```

inv_diff *Inverse Difference Metric for a GLCM*

Description

Calculate the inverse difference feature or metric for a gray-level co-occurrence matrix. For definition and application, see Lofstedt et al. (2019) doi:10.1371/journal.pone.0212110.

Usage

```
inv_diff(x, ...)

## Default S3 method:
inv_diff(x, ...)

## S3 method for class 'matrix'
inv_diff(x, ...)

## S3 method for class 'FitLandDF'
inv_diff(x, nlevels, ...)
```

Arguments

x	gray-level co-occurrence matrix
...	additional parameters
nlevels	desired number of discrete gray levels

Value

inverse difference metric of x

Examples

```
## calculate inverse difference of arbitrary GLCM
# define arbitrary GLCM
x <- matrix(1:16, nrow = 4)

# normalize
n_x <- normalize_glcm(x)

# calculate inverse difference
inv_diff(n_x)

## calculate inverse difference of arbitrary fitness landscape
# create fitness landscape using FitLandDF object
vals <- runif(64)
vals <- array(vals, dim = rep(4, 3))
my_landscape <- fitscape::FitLandDF(vals)
```

```
# calculate inverse difference of fitness landscape, assuming 2 discrete gray levels
inv_diff(my_landscape, nlevels = 2)

## confirm value of inverse difference for fitness landscape
# extract normalized GLCM from fitness landscape
my_glcm <- get_comatrix(my_landscape, discrete = equal_discrete(2))

# calculate inverse difference of extracted GLCM
inv_diff(my_glcm)  # should match value of above inv_diff function call
```

`kmeans_discrete` *Kmeans clustering discretization Splitting of a vector of continuous values into k groups function to discretize using kmeans*

Description

Kmeans clustering discretization Splitting of a vector of continuous values into k groups function to discretize using kmeans

Usage

```
kmeans_discrete(nlevels)
```

Arguments

`nlevels`	number of levels to split continuous vector into

Value

function that makes a numeric vector discrete

`manhattan` *Manhattan Distance Function Factory*

Description

Returns a function that checks whether the Manhattan distance between two numeric vectors is less than or equal to a given threshold.

Usage

```
manhattan(dist = 1)
```

Arguments

`dist`	threshold above which the function will return `FALSE`

Value

function that checks if Manhattan distance between two vectors exceeds `dist`

Examples

```
# test data: Manhattan distance equals 2
x <- rep(0, 5)
y <- c(0, 1, 0, 0, 1)

# should return TRUE when checking Manhattan distance <= 3
dist_3 <- manhattan(3)
dist_3(x, y)

# should return FALSE when checking Manhattan distance <= 1
dist_1 <- manhattan(1)
dist_1(x, y)
```

---

max_prob *Maximum Probability Metric for a GLCM*

Description

Calculate the maximum probability feature or metric for a gray-level co-occurrence matrix. For definition and application, see Lofstedt et al. (2019) doi:10.1371/journal.pone.0212110.

Usage

```
max_prob(x, ...)

## Default S3 method:
max_prob(x, ...)

## S3 method for class 'matrix'
max_prob(x, ...)

## S3 method for class 'FitLandDF'
max_prob(x, nlevels, ...)
```

Arguments

`x`	gray-level co-occurrence matrix
`...`	additional parameters
`nlevels`	desired number of discrete gray levels

Value

maximum probability metric of `x`

Examples

```
## calculate maximum probability of arbitrary GLCM
# define arbitrary GLCM
x <- matrix(1:16, nrow = 4)

# normalize
n_x <- normalize_glcm(x)

# calculate maximum probability
max_prob(n_x)

## calculate maximum probability of arbitrary fitness landscape
# create fitness landscape using FitLandDF object
vals <- runif(64)
vals <- array(vals, dim = rep(4, 3))
my_landscape <- fitscape::FitLandDF(vals)

# calculate maximum probability of fitness landscape, assuming 2 discrete gray levels
max_prob(my_landscape, nlevels = 2)

## confirm value of maximum probability for fitness landscape
# extract normalized GLCM from fitness landscape
my_glcm <- get_comatrix(my_landscape, discrete = equal_discrete(2))

# calculate maximum probability of extracted GLCM
max_prob(my_glcm)  # should match value of above max_prob function call
```

`normalize_glcm` *Normalize a GLCM*

Description

Function that normalizes a gray-level co-occurrence matrix (GLCM) so that the sum of all the elements equals unity. This has the added benefit of converting the GLCM to a probability distribution.

Usage

```
normalize_glcm(mat)
```

Arguments

`mat`	gray-level co-occurrence matrix

Value

normalized GLCM as a numeric matrix

Examples

```
# normalize an arbitrary matrix
a <- matrix(1:9, nrow = 3)
n_a <- normalize_glcm(a)

print(a)
print(n_a)
```

`quantile_discrete` *Function to discretize based on quantiles*

Description

Function to discretize based on quantiles

Usage

```
quantile_discrete(nlevels)
```

Arguments

`nlevels`	number of levels to split continuous vector into

Value

function that makes a numeric vector discrete

Examples

```
# test data
```

`sum_squares` *Sum of Squares Metric for a GLCM*

Description

Calculate the sum of squares feature or metric for a gray-level co-occurrence matrix. For definition and application, see Lofstedt et al. (2019) doi:10.1371/journal.pone.0212110.

Usage

```
sum_squares(x, ...)

## Default S3 method:
sum_squares(x, ...)

## S3 method for class 'matrix'
sum_squares(x, ...)

## S3 method for class 'FitLandDF'
sum_squares(x, nlevels, ...)
```

Arguments

x	gray-level co-occurrence matrix
...	additional parameters
nlevels	desired number of discrete gray levels

Value

sum of squares metric of x

Examples

```
## calculate sum of squares of arbitrary GLCM
# define arbitrary GLCM
x <- matrix(1:16, nrow = 4)

# normalize
n_x <- normalize_glcm(x)

# calculate sum of squares
sum_squares(n_x)

## calculate sum of squares of arbitrary fitness landscape
# create fitness landscape using FitLandDF object
vals <- runif(64)
vals <- array(vals, dim = rep(4, 3))
my_landscape <- fitscape::FitLandDF(vals)

# calculate sum of squares of fitness landscape, assuming 2 discrete gray levels
sum_squares(my_landscape, nlevels = 2)

## confirm value of sum of squares for fitness landscape
# extract normalized GLCM from fitness landscape
my_glcm <- get_comatrix(my_landscape, discrete = equal_discrete(2))

# calculate sum of squares of extracted GLCM
sum_squares(my_glcm)  # should match value of above sum_squares function call
```

Index