Package ‘pscsne’

September 8, 2022

Type Package

Title Polyspherical Cauchy SNE

Version 0.0.1.900005

Date 2022-09-07

Description
Implementation the polyspherical cauchy SNE, a nonlinear dimensionality reduction technique.

License GPL-3

LazyData true

Encoding UTF-8

Depends R (>= 3.5.0)

Imports DirStats,
optimParallel,
parallel,
rlang,
sphunif,
sdetorus,
utils

Suggests knitr,
markdown,
rmarkdown,
scatterplot3d,
viridisLite,
Directional,
mvtnorm,
numDeriv,
rotasym,
testthat,
abind,
rgl

URL https://github.com/luisrodrigar/psc-sne

BugReports https://github.com/luisrodrigar/psc-sne

RoxygenNote 7.2.0

Roxygen list(old_usage = TRUE)

VignetteBuilder knitr

Config/testthat/edition 3

R topics documented:

pscsne-package — pscsne: *Polyspherical Cauchy SNE*

Description

Implementation of the Polyspherical Cauchy SNE

Author(s)

Eduardo García-Portugués and Luis Ángel Rodríguez García.

bin_search — *Binary search tree algorithm*

Description

Calculate the rho value based on a perplexity difference and current values of rho, rho min and rho max.

Usage

```
bin_search(perp_diff, rho, rho_min, rho_max)
```

Arguments

`perp_diff`	difference between the value obtained and the fixed perplexity
`rho`	the current value optimized
`rho_min`	min value for the current rho
`rho_max`	max value for the current rho

Value

Rho concentration parameter found in this step, currently min and max rho `clusterFactory` *Cluster factory*

Description

Cluster factory for concurrent tasks.

Usage

```
clusterFactory(num_cores, outfile = "")
```

Arguments

`num_cores`	number of cores to be available in the cluster.
`outfile`	text file where print the output, empty means no output.

Value

Cluster created with the number of cores passing as parameter.

Examples

```
clusterFactory(2)
clusterFactory(2, "log.txt")
file.remove("log.txt")
```

`cosine_polysph` *Cosine similarity for the poly-sphere*

Description

Calculates the cosine similarity for each sphere of the 3 dimensional array.

Usage

```
cosine_polysph(x)
```

Arguments

`x`	an array of size `c(n,d + 1,r)` with the polyspherical data, where `n` is the number of observations, `d` is the dimension of each sphere, and `r` is the number of spheres.

Value

An array of size `c(n,n,r)` with the cosine similarities of each sphere $\mathcal{S}^d$ from `x`.

Examples

```
x <- sphunif::r_unif_sph(100, 3, 3)
cosine_polysph(x)
```

`cos_sim_i` *Cosine similarities vector for the i-th observation*

Description

Obtain the index of an element within a vector for the equivalent in the upper triangular matrix without diagonal

Usage

```
cos_sim_i(cos_sim_psh, i, r, n)
```

Arguments

`cos_sim_psh`	a vector of size `n/2` with the cosine similarities in high-dimension for the poly-sphere $(\mathcal{S}^p)^r$. The way that the cosine similarities matrix is treated makes the calculus faster since it is flat in a vector object. Optional parameter, defaults to `NULL`.
`i`	corresponds to the i-th observation index.
`r`	the number of spheres in the polysphere
`n`	the sample size.

Examples

```
n <- 6
r <- 6
x <- sphunif::r_unif_sph(n, 3, r)
cos_sim_psh <- sphunif::Psi_mat(x, scalar_prod = TRUE)
cos_sim_i(cos_sim_psh, 4, r, n)
```

`diag_3d` *Set each object's diagonal of a 3d-array*

Description

Set a specific value to the diagonal of a each matrix of a 3 dimensional array.

Usage

```
diag_3d(x, k, val)
```

Arguments

`x`	an array of size `c(n,d + 1,r)` with the polyspherical data, where `n` is the number of observations, `d` is the dimension of each sphere, and `r` is the number of spheres.
`k`	the `k`-th sphere from $(\mathcal{S}^p)^r$.
`val`	value to set to each array's diagonal of the poly-sphere `x`.

Value

The `k`-th matrix of `x` with the diagonal set to `val`.

Examples

```
x <- sphunif::r_unif_sph(100, 3, 3)
diag_3d(x, 1, 0)
diag_3d(x, 3, 0)
```

---

d_ij_psph_cauchy — *Polyspherical Cauchy density for the i-th and j-th observations*

Description

Calculate the high-dimension polyspherical Cauchy density for the i-th and j-th observations.

Usage

```
d_ij_psph_cauchy(x, i, j, rho)
```

Arguments

`x`	an array of size `c(n,d + 1,r)` with the polyspherical data, where `n` is the number of observations, `d` is the dimension of each sphere, and `r` is the number of spheres.
`i`	corresponds to the i-th observation index.
`j`	corresponds to the j-th observation index.
`rho`	concentration parameter must be in [0, 1).

Value

Polyspherical Cauchy density value given the parameters.

---

d_i_psph_cauchy — *Marginal polyspherical Cauchy density for the i-th observation*

Description

Calculate the high-dimension polyspherical Cauchy density for the i-th observation.

Usage

```
d_i_psph_cauchy(x, i, rho_list)
```

Arguments

`x`	an array of size `c(n,d + 1,r)` with the polyspherical data, where `n` is the number of observations, `d` is the dimension of each sphere, and `r` is the number of spheres.
`i`	corresponds to the i-th observation index.
`rho_list`	rho list of size `n` for each `i`-th observation that stands for the concentration parameter.

Value

Polyspherical Cauchy density value given the parameters.

Examples

```
x <- sphunif::r_unif_sph(20, 3, 4)
d_i_psph_cauchy(x, 1, rep(0.5, 20))
d_i_psph_cauchy(x, 4, rep(0.9999, 20))
```

`d_i_sph_cauchy` *Marginal spherical Cauchy density function*

Description

Calculate the marginal low-dimension spherical Cauchy probability.

Usage

```
d_i_sph_cauchy(y, i, rho)
```

Arguments

`y`	matrix that stands for a sphere, $\mathcal{S}^d$.
`i`	corresponds to the i-th observation index.
`rho`	concentration parameter must be in [0, 1).

Value

Spherical marginal Cauchy density probability given the parameters.

d_sph_cauchy *High-dimension polyspherical Cauchy density*

Description

Calculate the high-dimension spherical Cauchy density function.

Usage

```
d_sph_cauchy(x, i, j, rho, k, p)
```

Arguments

x	an array of size c(n,d + 1,r) with the polyspherical data, where n is the number of observations, d is the dimension of each sphere, and r is the number of spheres.
i	corresponds to the i-th observation index.
j	corresponds to the j-th observation index.
rho	concentration parameter must be in [0, 1).
k	corresponds to the k-th sphere index.
p	corresponds to S^p, associated to R^(p+1).

Value

Spherical Cauchy density value given the parameters.

d_total_psph_cauchy *Polyspherical marginal density function values for all the observations*

Description

Calculate the marginal high-dimension polyspherical Cauchy probabilities for the i-th observation.

Usage

```
d_total_psph_cauchy(x, rho_list)
```

Arguments

x	an array of size c(n,d + 1,r) with the polyspherical data, where n is the number of observations, d is the dimension of each sphere, and r is the number of spheres.
rho_list	rho list of size n for each i-th observation that stands for the concentration parameter.

Value

Marginal polyspherical Cauchy probabilities vector given the ρ parameters.

Examples

```
x <- sphunif::r_unif_sph(20, 3, 4)
d_total_psph_cauchy(x, rep(0.5, 20))
d_total_psph_cauchy(x, rep(0.9999, 20))
```

`fibonacci_lattice` *Fibonacci lattice algorithm to generate evenly separated points*

Description

Generated optimal evenly separated points onto the sphere $\mathcal{S}^2$.

Usage

```
fibonacci_lattice(n)
```

Arguments

`n`	positive integer that defines the size of the sample to generate.

Value

Evenly optimal separated points onto the low-dimension sphere $\mathcal{S}^2$.

Examples

```
fibonacci_lattice(100)
fibonacci_lattice(250)
```

`gen_opt_sphere` *Generate optimum evenly separated points*

Description

Generated optimal evenly separated points onto the sphere $\mathcal{S}^d$.

Usage

```
gen_opt_sphere(n, d)
```

Arguments

`n`	positive integer that defines the size of the sample to generate.
`d`	size of the low-dimension which defines the sphere $\mathcal{S}^d$.

Value

Evenly optimal separated points onto the low-dimension sphere $\mathcal{S}^d$.

Examples

```
gen_opt_sphere(100, 1)
gen_opt_sphere(250, 2)
```

high_dimension — *Polyspherical Cauchy conditional probability matrix (matrix version)*

Description

Calculates the high-dimension conditional probabilities of a polyspherical Cauchy distribution. Matrix version algorithm.

Usage

```
high_dimension(x, rho_list, cos_sim_psh = NULL)
```

Arguments

x	an array of size `c(n,d + 1,r)` with the polyspherical data, where `n` is the number of observations, `d` is the dimension of each sphere, and `r` is the number of spheres.
rho_list	rho list of size `n` for each `i`-th observation that stands for the concentration parameter.
cos_sim_psh	a vector of size `n/2` with the cosine similarities in high-dimension for the polysphere $(\mathcal{S}^p)^r$. The way that the cosine similarities matrix is treated makes the calculus faster since it is flat in a vector object. Optional parameter, defaults to `NULL`.

Value

An array of size `c(n,n)` with the high-dimension conditional probabilities of `x`.

Examples

```
x <- sphunif::r_unif_sph(100, 3, 3)
high_dimension(x, rep(0.5, 100))
high_dimension(x, rep(0.5, 100), sphunif::Psi_mat(x, scalar_prod = TRUE))
```

high_dimension_i — *Polyspherical i-th Cauchy conditional probability vector (matrix version)*

Description

Calculates the high-dimension conditional probabilities of a polyspherical Cauchy distribution for the i-th observation. Matrix version algorithm.

Usage

```
high_dimension_i(x, i, rho, cos_sim_psh = NULL)
```

Arguments

`x`	an array of size `c(n,d + 1,r)` with the polyspherical data, where `n` is the number of observations, `d` is the dimension of each sphere, and `r` is the number of spheres.
`i`	corresponds to the i-th observation index.
`rho`	concentration parameter must be in [0, 1).
`cos_sim_psh`	a vector of size `n/2` with the cosine similarities in high-dimension for the polysphere $(\mathcal{S}^p)^r$. The way that the cosine similarities matrix is treated makes the calculus faster since it is flat in a vector object. Optional parameter, defaults to `NULL`.

Value

An vector of size `n` with the high-dimension conditional probabilities of the i-th observation in `x`.

Examples

```
x <- sphunif::r_unif_sph(100, 3, 3)
high_dimension_i(x, 1, 0.5)
high_dimension_i(x, 100, 0.5, sphunif::Psi_mat(x, scalar_prod = TRUE))
```

`high_dimension_p` *Polyspherical Cauchy conditional probability matrix (scalar version)*

Description

Calculates the high-dimension conditional probabilities of a polyspherical Cauchy distribution. Scalar version algorithm.

Usage

```
high_dimension_p(x, rho_list)
```

Arguments

`x`	an array of size `c(n,d + 1,r)` with the polyspherical data, where `n` is the number of observations, `d` is the dimension of each sphere, and `r` is the number of spheres.
`rho_list`	rho list of size `n` for each `i`-th observation that stands for the concentration parameter.

Value

An array of size `c(n,n)` with the high-dimension conditional probabilities of `x`.

Examples

```
x <- sphunif::r_unif_sph(100, 3, 3)
high_dimension_p(x, rep(0.5, 100))
```

`index_upper_trian` *Vector index of an upper triangular matrix withou diagonal*

Description

Obtain the index of an element within a vector for the equivalent in the upper triangular matrix without diagonal

Usage

```
index_upper_trian(i, j, n)
```

Arguments

i	corresponds to the i-th observation index.
j	corresponds to the j-th observation index.
n	the sample size of the symmetric matrix (n x n)

Examples

```
mat <- rbind(c(0, 1, 2), c(0, 0, 1), c(0, 0, 0))
vec <- c(1, 2, 1)
index <- index_upper_trian(1, 2, 3)
all.equal(vec[index], mat[1, 2])
```

`jcondi_psph` *Conditional polyspherical Cauchy probability for the i-th and j-th observations*

Description

Calculate the conditional high-dimension probability of the j-th observation given the i-th.

Usage

```
jcondi_psph(x, i, j, rho_list, d_total_i_psph_cauchy = NULL)
```

Arguments

x	an array of size `c(n,d + 1,r)` with the polyspherical data, where `n` is the number of observations, `d` is the dimension of each sphere, and `r` is the number of spheres.
i	corresponds to the i-th observation index.
j	corresponds to the j-th observation index.
rho_list	rho list of size `n` for each `i`-th observation that stands for the concentration parameter.
d_total_i_psph_cauchy	marginal probability of the i-th observation. Optional, defaults to `NULL`.

Value

Conditional polyspherical Cauchy probability of the j-th given the i-th observation.

Examples

```
x <- sphunif::r_unif_sph(20, 3, 4)
jcondi_psph(x, 1, 2, rep(0.5, 20), d_total_psph_cauchy(x, rep(0.5, 20))[1])
jcondi_psph(x, 4, 6, rep(0.9999, 20),
            d_total_psph_cauchy(x, rep(0.9999, 20))[4])
```

---

jcondi_sph — *Conditional spherical Cauchy probability*

Description

Calculate the probability of choosing a pair of elements where the i-th and the j-th observations are selected.

Usage

```
jcondi_sph(y, i, j, rho, d_i_sph_cauchy = NULL)
```

Arguments

y	matrix that stands for a sphere, $\mathcal{S}^d$.
i	corresponds to the i-th observation index.
j	corresponds to the j-th observation index.
rho	concentration parameter must be in [0, 1).
d_i_sph_cauchy	the marginal probability of the i-th observation. Optional, defaults to NULL.

Value

Spherical marginal Cauchy density probability given the parameters.

---

jdf — *Juan de Fuca currents*

Description

Sea currents defined by their angle and speed produced in some geographic coordinates in Juan de Fuca strait area from 2020-06-01 to 2022-07-01, last day excluded.

Usage

```
jdf
```

Format

A data frame with 638,400 observations (rows) and 5 variables (columns):

lat latitude coordinate in decimal format.

lon longitude coordinate in decimal format.

time time in the following format yyyy-MM-dd hh:mm:ss.

theta angle in radians of the sea current vector.

speed module of the sea current vector.

Details

The data object is created with the code in https://github.com/luisrodrigar/psc-sne/blob/main/data-raw/strait-juan-fuca/data-acquisition-juan-fuca.R

Source

https://github.com/luisrodrigar/psc-sne/blob/main/data/jdf.rda

References

TODO

Examples

```
# Load data
data("jdf")

# Plot vector field
# TODO
```

---

kl_divergence_grad *Kullback-Leibler divergence gradient*

Description

Calculates analytically the gradient of the Kullback-Leibler divergence between low- and high-dimensional pairwise probabilities.

Usage

```
kl_divergence_grad(Y, i, rho, d, P, cos_sim = NULL, Q = NULL)
```

Arguments

Y	matrix of size c(n,d), where n is the number of observation, with the points onto the sphere $\mathcal{S}^d$.
i	corresponds to the i-th observation index.
rho	concentration parameter must be in [0, 1).
d	target dimension to reduce the data.

P	matrix of size c(n,n) with the high-dimensional polyspherical Cauchy probabilities.
cos_sim	a vector of size n/2 with the cosine similarities in high-dimension for the polysphere $(\mathcal{S}^p)^r$. The way that the cosine similarities matrix is treated makes the calculus faster since it is flat in a vector object. Optional parameter, defaults to NULL.
Q	matrix of size c(n,n) with the low-dimension spherical Cauchy probabilities. Optional parameter, defaults to NULL.

Value

Resulting reduced data for the i-th observation onto the sphere $\mathcal{S}^d$.

Examples

```
Y <- sphunif::r_unif_sph(40, 3, 1)[ , , 1]
X <- sphunif::r_unif_sph(40, 3, 3)
P <- high_dimension(X, rep(0.5, 40))
kl_divergence_grad(Y, 3, 0.5, 2, P)
cos_sim <- vec2matrix(
    vec = drop(sphunif::Psi_mat(array(X, dim = c(nrow(X), ncol(X), 1)),
                               scalar_prod = TRUE)),
    n = nrow(X),
    diag_value = 1)
Q <- low_dimension_Q(Y, 0.5)
kl_divergence_grad(Y, 3, 0.5, 2, P, cos_sim, Q)
```

---

kms_dir *Kernel mean shift clustering for directional data*

Description

Performs kernel mean shift clustering on S^d using an adapted Euler algorithm and kernel density estimator.

Usage

```
kms_dir(data, x = data, h = NULL, N = 500, eps = 0.001, tol = 0.1,
  keep_paths = FALSE, show_prog = TRUE)
```

Arguments

data	a matrix of size c(n,d + 1) with the sample.
x	a matrix of size c(nx,d + 1) with the initial points for the Euler algorithm. Defaults to data.
h	bandwidth. Chosen automatically if NULL (default).
N	maximum number of iterations. Defaults to 500.
eps	convergence tolerance. Defaults to 1e-3.
tol	tolerance for equality of modes. Defaults to 1e-1.
keep_paths	keep the ascending paths? Defaults to FALSE.
show_prog	display progress? Defaults to TRUE.

Value

A list with the following entries:

- `end_points`: end points of the Euler algorithm. A matrix of the same size as `x`.
- `cluster`: vector giving the cluster labels.
- `modes`: estimated modes for each cluster (sorted).
- `paths`: ascension paths, if `keep_paths = TRUE`. A list of length `nx` with matrices of size `c(np,d + 1)`, where `np` is at most `N + 1`.
- `tree`: internal hierarchical clustering tree used to merge modes.
- `h`: used bandwidth.

Examples

```
# Detection of three clusters in S^2
samp <- rbind(
  rotasym::r_vMF(n = 50, mu = c(0, 0, 1), kappa = 5),
  rotasym::r_vMF(n = 50, mu = c(0, 0, -1), kappa = 5),
  rotasym::r_vMF(n = 50, mu = c(1, 0, 0), kappa = 5)
)
kms <- kms_dir(data = samp, keep_paths = TRUE)
sd3 <- scatterplot3d::scatterplot3d(samp, xlim = c(-1, 1),
                                    ylim = c(-1, 1), zlim = c(-1, 1),
                                    color = kms$cluster + 1, pch = 16,
                                    cex.symbol = 0.5)
for (i in seq_len(nrow(samp))) sd3$points3d(kms$paths[[i]], type = "l",
                                            lty = 3)
sd3$points3d(kms$end_points, col = kms$cluster + 1, pch = "*", cex = 2)

# Detection of three clusters in S^1
samp <- rbind(
  rotasym::r_vMF(n = 50, mu = c(0, 1), kappa = 5),
  rotasym::r_vMF(n = 50, mu = c(-sqrt(2), -sqrt(2)) / 2, kappa = 5),
  rotasym::r_vMF(n = 50, mu = c(sqrt(2), -sqrt(2)) / 2, kappa = 5)
)
kms <- kms_dir(data = samp, keep_paths = TRUE)
plot(samp, col = kms$cluster + 1, pch = 16, xlim = c(-1.5, 1.5),
     ylim = c(-1.5, 1.5))
for (i in seq_len(nrow(samp))) {
 l <- seq(0, 1, length.out = nrow(rbind(kms$paths[[i]])))
 lines(sqrt(1 + l) * kms$paths[[i]], lty = 3)
}
points(sqrt(2) * kms$end_points, col = kms$cluster + 1, pch = "*", cex = 2)
kms
```

`low_dimension_Q` *Low-dimension probabilities*

Description

Calculate the low-dimension probabilities of a reduced matrix Y.

Usage

```
low_dimension_Q(Y, rho)
```

Arguments

Y	matrix of size `c(n,d)`, where `n` is the number of observation, with the points onto the sphere $\mathcal{S}^d$.
rho	concentration parameter must be in [0, 1).

Value

A matrix with the values of x projected onto the sphere of radius 1.

Examples

```
Y <- rotasym::r_unif_sphere(100, 2)
low_dimension_Q(Y, 0)
low_dimension_Q(Y, 0.5)
low_dimension_Q(Y, 0.9999)
```

`prob_cond_i_psph` *Conditional polyspherical Cauchy probability for the i-th observation*

Description

Calculate the conditional high-dimension probability for all the j-th observation given the i-th.

Usage

```
prob_cond_i_psph(x, i, rho_list, d_total_i_psph_cauchy = NULL)
```

Arguments

x	an array of size `c(n,d + 1,r)` with the polyspherical data, where `n` is the number of observations, `d` is the dimension of each sphere, and `r` is the number of spheres.
i	corresponds to the i-th observation index.
rho_list	rho list of size `n` for each `i`-th observation that stands for the concentration parameter.
d_total_i_psph_cauchy	marginal probability of the i-th observation. Optional, defaults to `NULL`.

Value

Conditional polyspherical Cauchy probability for all the j-th given the i-th observation.

Examples

```
x <- sphunif::r_unif_sph(20, 3, 4)
rho_list_1 <- rep(0.5, 20)
rho_list_2 <- rep(0.9999, 20)
d_total_1_psph <- d_total_psph_cauchy(x, rho_list_1)[1]
d_total_4_psph <- d_total_psph_cauchy(x, rho_list_2)[4]
prob_cond_i_psph(x, 1, rho_list_1, d_total_1_psph)
prob_cond_i_psph(x, 4, rho_list_2, d_total_4_psph)
```

`prob_cond_i_sph` *Conditional spherical Cauchy probabilities given the i-th observation*

Description

Calculate n, where it stands for the sample size, probabilities where each element is the probability of choosing the i-th and the j-th observations, the i-th element is a fixed element in the chosen pairs.

Usage

```
prob_cond_i_sph(y, i, rho)
```

Arguments

y	matrix that stands for a sphere, $\mathcal{S}^d$.
i	corresponds to the i-th observation index.
rho	concentration parameter must be in [0, 1).

Value

Spherical marginal Cauchy density probability given the parameters.

Examples

```
y <- rotasym::r_unif_sphere(100, 2)
prob_cond_i_sph(y, 1, 0.5)
prob_cond_i_sph(y, 4, 0.9999)
```

`psc_sne` *Polyspherical Cauchy SNE*

Description

Calculates the polyspherical-Cauchy SNE given a data onto the polysphere and the reduced dimension.

Usage

```
psc_sne(X, d, rho_psc_list = NULL, rho = 0.5, perplexity = 30,
  maxit = 1000, initial_momentum = 0.5, final_momentum = 0.8,
  eta = 200, early_exaggeration = 4, colors = NULL, show_prog = 100,
  show_plots = TRUE, tol = 1e-06,
  parallel_cores = parallel::detectCores() - 1, init = c("equispaced",
  "random", "most_promising")[1], N = 10)
```

Arguments

- `X` an array of size `c(n,d + 1,r)` with the polyspherical data, where `n` is the number of observations, `d` is the dimension of each sphere, and `r` is the number of spheres.
- `d` the target dimension to use for the reduced data of `X`.
- `rho_psc_list` rho parameters of the high-dimensional polyspherical Cauchy probabilities. Multiple types of parameters are allowed, distinguishing three scenarios. The first one when the type of the parameter is a list, then it contains the vector `rho_values` and the matrix `P`, the second scenario when the type is a vector, then this object contains the rho values, within this function the `high_dimension` function is called to get the matrix `P`. The last scenario that is when this object is set to `NULL`, i.e., the `rho_optim_bst` function is called to get the rho values (given a fixed perplexity) and the probabilities matrix. Optional parameter, defaults to `NULL`).
- `rho` parameter of the low-dimensional spherical Cauchy probabilities. Optional, defaults to `0.5`).
- `perplexity` parameter that measures the number of neighbors to use when mapping between high- and low-dimension. Defaults to `30`).
- `maxit` maximum number of iterations. Defaults to `1e3`).
- `initial_momentum` first value of the momentum of the first `250` iterations. Defaults to `0.5`.
- `final_momentum` momentum to take into account after the `250` iteration. Defaults to `0.8`.
- `eta` is the learning rate of the optimization algorithm. Optional param, defaults to `200`.
- `early_exaggeration` the first `100` iterations results are exaggerated by this factor. Optional parameter, defaults to `4.0`).
- `colors` list with as many elements as observations are, only valid when visualization is true. Optional parameter, defaults to `NULL`).
- `show_prog` defines the number of iterations skipped when reporting the progress. Defaults to `100`, i.e., only multiples of `100` are reported. If `FALSE`, no progress is shown at all.
- `show_plots` show convergence plots? If `TRUE` (default), a plot is shown: after `2 * show_prog` iterations and at the end of the search.
- `tol` is the tolerance, when is below this value it is considered that a good solution has been obtained. Defaults to `1e-6`).
- `parallel_cores` number of cores to use concurrently for the calculation of the gradient. Defaults to `parallel::detectCores() -1`, that means that uses the total number of cores of the computer except one of them.

`init`	how to initialize the scores: `"equispaced"` (evenly spaced points on the circumference/sphere), `"random"` (random points generated uniformly), `"most_promising"` (best configuration obtained in `N` differently-initialized searches run with `maxit / 100` iterations), or a matrix. Defaults to `"equispaced"`.
`N`	number of differently-initialized searches (see above). Defaults to `10`.

Details

When `init = "most_promising"`, `N -1` initializations are random and one is an equispaced grid.

Value

A list with the following entries:

- `best_Y`: best configuration of scores found.
- `last_Y`: last configuration of scores found.
- `rho_psc_list`: vector or rho's.
- `diagnostics`: data frame with the objective function values, absolute/relative errors, gradient norms, and moment norms.
- `convergence`: convergence flag.

Examples

```
X <- sphunif::r_unif_sph(n = 100, p = 3, M = 3)
X[1:50, , 1] <- rotasym::r_vMF(n = 50, mu = c(0, 0, 1), kappa = 10)
X[51:100, , 1] <- rotasym::r_vMF(n = 50, mu = c(0, 0, -1), kappa = 10)
psc <- psc_sne(X = X, d = 1, parallel_cores = 2, eta = 50,
               init = "most_promising")
psc2 <- psc_sne(X = X, d = 1, parallel_cores = 2, eta = 50,
                init = psc$last_Y)
```

`radial_projection` *Radial projection onto the sphere*

Description

Projection of the points onto the sphere of radius 1.

Usage

```
radial_projection(y)
```

Arguments

`y`	matrix with the points in the sphere.

Value

A matrix with the values of x projected onto the sphere of radius 1.

Examples

```
y <- rotasym::r_unif_sphere(100, 2)
radial_projection(y)
```

`rho_optimize_1`	*Concurrent optimization of the ρ concentration parameters (matrix version)*

Description

Calculate the rho list values based on a fixed perplexity and a given data in $(\mathcal{S}^p)^r$. Optimize the value using the method L-BFGS-B. The boundaries are set from 0 to 0.9999. It prints the time consumption.

Usage

```
rho_optimize_1(x, perplexity, num_cores = parallel::detectCores() - 1,
  cos_sim_psh = NULL)
```

Arguments

`x`	an array of size `c(n,d + 1,r)` with the polyspherical data, where `n` is the number of observations, `d` is the dimension of each sphere, and `r` is the number of spheres.
`perplexity`	a fixed value (between 5 and 100) to optimize the rho parameters.
`num_cores`	number of cores to execute the code concurrently. This value must be below the total number of the CPU has available.
`cos_sim_psh`	a vector of size `n/2` with the cosine similarities in high-dimension for the polysphere $(\mathcal{S}^p)^r$. The way that the cosine similarities matrix is treated makes the calculus faster since it is flat in a vector object. Optional parameter, defaults to `NULL`.

Value

Rho list ($\boldsymbol{\rho}$) with the values optimized for the given perplexity.

`rho_optimParallel`	*Concurrent optim to calculate ρ concentration parameters (scalar version)*

Description

Calculate the rho list values based on a fixed perplexity and a given data in $(\mathcal{S}^p)^r$. Optimize the value using the concurrently method L-BFGS-B (optimParallel). The boundaries are set from 0 to 0.9999. It prints the time consumption.

Usage

```
rho_optimParallel(x, perplexity, num_cores = parallel::detectCores() - 1)
```

Arguments

x	an array of size c(n,d + 1,r) with the polyspherical data, where n is the number of observations, d is the dimension of each sphere, and r is the number of spheres.
perplexity	a fixed value (between 5 and 100) to optimize the rho parameters.
num_cores	number of cores to execute the code concurrently. This value must be below the total number of the CPU has available.

Value

Rho list (ρ) with the values optimized for the given perplexity.

---

rho_optim_bst *Binary search rho optimization for each observation*

Description

Calculate the rho list values based on a fixed perplexity and a given data in $(\mathcal{S}^p)^r$. The boundaries are set from 0 to 0.9999 for each value. It prints the time consumption.

Usage

```
rho_optim_bst(x, perp_fixed, num_cores = parallel::detectCores() - 1)
```

Arguments

x	an array of size c(n,d + 1,r) with the polyspherical data, where n is the number of observations, d is the dimension of each sphere, and r is the number of spheres.
perp_fixed	a fixed value used to optimized the rho values.
num_cores	number of cores to execute the code concurrently. This value must be below the total number of the CPU has available.

Value

Rho values and conditional probability matrix obtained as a result of the optimization.

Examples

```
x <- sphunif::r_unif_sph(20, 3, 4)
rho_optim_bst(x, perp_fixed = 15, num_cores = 2)
rho_optim_bst(x, perp_fixed = 26, num_cores = 2)
```

`rho_optim_ineff` *Serial optimization of the ρ concentration parameters (matrix version)*

Description

Calculate the rho list values based on a fixed perplexity and a given data in $(\mathcal{S}^p)^r$. Optimize the value using the method L-BFGS-B. The boundaries are set from 0 to 0.9999. It prints the time consumption.

Usage

```
rho_optim_ineff(x, perplexity)
```

Arguments

`x`	an array of size `c(n,d + 1,r)` with the polyspherical data, where `n` is the number of observations, `d` is the dimension of each sphere, and `r` is the number of spheres.
`perplexity`	a fixed value (between 5 and 100) to optimize the rho parameters.

Value

Rho list ($\boldsymbol{\rho}$) with the values optimized for the given perplexity.

`rho_optim_i_bst` *Binary search rho optimization for the i-th observation*

Description

Calculate the rho based on a fixed perplexity and a given data in $(\mathcal{S}^p)^r$. The boundaries are set from 0 to 0.9999.

Usage

```
rho_optim_i_bst(x, i, perp_fixed, tolerance = 0.001, rho = 0.5, max_tries = 20)
```

Arguments

`x`	an array of size `c(n,d + 1,r)` with the polyspherical data, where `n` is the number of observations, `d` is the dimension of each sphere, and `r` is the number of spheres.
`i`	the i-th observation.
`perp_fixed`	a fixed value used to optimized the rho values.
`tolerance`	whether the difference between previous and current results is below this value (optional, default value `1e-3`).
`rho`	parameter which determines the concentration of the spherical Cauchy distribution (optional, default value `0.5`).
`max_tries`	number of maximum tries for each value of the rho list.

Value

Rho concentration parameter and conditional probabilities calculated for the i-th observation.

`rho_optim_par` *Concurrent optimization of the ρ concentration parameters (scalar version)*

Description

Calculate the rho list values based on a fixed perplexity and a given data in $(\mathcal{S}^p)^r$. Optimize the value using the method L-BFGS-B, setting the boundaries within [0, 1). The limit is considered as `1 -1e-4`. Each value is calculated concurrently. At the end, the time consumption is shown.

Usage

```
rho_optim_par(x, perplexity, num_cores = parallel::detectCores() - 1)
```

Arguments

`x`	an array of size `c(n,d + 1,r)` with the polyspherical data, where `n` is the number of observations, `d` is the dimension of each sphere, and `r` is the number of spheres.
`perplexity`	a fixed value (between 5 and 100) to optimize the rho parameters.
`num_cores`	number of cores to execute the code concurrently. This value must be below the total number of the CPU has available.

Value

Rho list (ρ) with the values optimized for the given perplexity.

Examples

```
x <- sphunif::r_unif_sph(20, 3, 4)
rho_optim_par(x, 22, 2)
rho_optim_par(x, 30, 2)
```

`rho_optim_serial` *Serial optimization ρ concentration parameters (scalar version)*

Description

Calculate the rho list values based on a fixed perplexity and a given data in $(\mathcal{S}^p)^r$. Optimize the value using a method L-BFGS-B, setting the boundaries from 0 to 0.9999. Each value is calculated serially. It prints the time consumption.

Usage

```
rho_optim_serial(x, perplexity)
```

Arguments

x	an array of size c(n,d + 1,r) with the polyspherical data, where n is the number of observations, d is the dimension of each sphere, and r is the number of spheres.
perplexity	a fixed value (between 5 and 100) to optimize the rho parameters.

Value

Rho list (ρ) with the values optimized for the given perplexity.

---

r_block — *Sample correlation matrices with block structure*

Description

Sample a zero-mean multivariate normal $N_{gp}(0, \Sigma)$ with a diagonal block matrix Σ partitioned into g blocks with p variables. Each block is constructed as a toeplitz correlation matrix.

Usage

```
r_block(n, g = 5, p = 20, rho = rep(0.9, times = g))
```

Arguments

n	sample size.
g	number of groups of variables that are uncorrelated between them.
p	number of variables on each group.
rho	a vector of size g with the correlations determining the toeplitz correlation matrices of each group.

Value

A matrix of size c(n,g * p) with the sample.

Examples

```
# Visualize some features
n <- 200
g <- 10
p <- 10
x <- r_block(n = n, g = g, p = p)
pairs(x[, c(1:2, p + 1:2)],
labels = c("Var 1", "Var 2", paste("Var", p + 1), paste("Var", p + 2)))

# Standardize variables -- now the vectors of observations for each variable
# (the columns) live on \sqrt{n - 1} * S^{n - 1}!
x_sca <- scale(x)
colSums(x_sca^2)

# Make the features live on S^{n - 1}
x_sca <- scale(x) / sqrt(n - 1)
```

```
# Transpose matrix (features become observations)
feat <- t(x_sca)

# Run psc_sne() with colors being the groups of variables
# dim(feat) <- c(dim(feat), 1)
# cols <- rep(1:g, each = p)
# psc_sne(X = feat, d = 1, colors = cols)
```

---

samplers_one — *Samplers of one-dimensional modes of variation for polyspherical data*

Description

Functions for sampling data on $(S^d)^r$ for $d = 1, 2$ using one-dimensional modes of variation.

Usage

```
r_path_s1r(n, r, alpha = runif(r, -pi, pi), k = sample(-2:2, size = r,
  replace = TRUE), sigma = 0.25, angles = FALSE)

r_path_s2r(n, r, t = 0, c = 1, Theta = t(rotasym::r_unif_sphere(n = r, p
  = 3)), sigma = 0.25, spiral = FALSE)
```

Arguments

n	sample size.
r	number of spheres in the polysphere $(S^d)^r$.
alpha	a vector of size `r` valued in $[-\pi, \pi)$ with the initial angles for the linear trend. Chosen at random by default.
k	a vector of size `r` with the **integer** slopes defining the angular linear trend. Chosen at random by default.
sigma	standard deviation of the noise about the one-dimensional mode of variation. Defaults to `0.25`.
angles	return angles in $[-\pi, \pi)$? Defaults to `FALSE`.
t	latitude, with respect to `Theta`, of the small circle. Defaults to `0` (equator).
c	Clélie curve parameter, changing the spiral wrappings. Defaults to `1`.
Theta	a matrix of size `c(3,r)` giving the north poles for S^2. Useful for rotating the sample. Chosen at random by default.
spiral	consider a spiral (or, more precisely, a Clélie curve) instead of a small circle? Defaults to `FALSE`.

Value

An array of size `c(n,d,r)` with samples on $(S^d)^r$. If `angles = TRUE` for `r_path_s1r`, then a matrix of size `c(n ,r)` with angles is returned.

Examples

```
# Straight trends on (S^1)^2
n <- 200
samp_1 <- r_path_s1r(n = n, r = 2, k = c(1, 2), angles = TRUE)
plot(samp_1, xlim = c(-pi, pi), ylim = c(-pi, pi), col = rainbow(n),
     axes = FALSE, xlab = "", ylab = "", pch = 16)
sdetorus::torusAxis()

# Straight trends on (S^1)^3
n <- 200
samp_2 <- r_path_s1r(n = n, r = 3, angles = TRUE)
pairs(samp_2, xlim = c(-pi, pi), ylim = c(-pi, pi), col = rainbow(n),
      pch = 16)
sdetorus::torusAxis()
scatterplot3d::scatterplot3d(
  samp_2, xlim = c(-pi, pi), ylim = c(-pi, pi), zlim = c(-pi, pi),
  xlab = "", ylab = "", zlab = "", color = rainbow(n), pch = 16
)

# Small-circle trends on (S^2)^2
n <- 200
samp_3 <- r_path_s2r(n = n, r = 2, sigma = 0.1)
old_par <- par(mfrow = c(1, 2))
scatterplot3d::scatterplot3d(
  samp_3[, , 1], xlim = c(-1, 1), ylim = c(-1, 1), zlim = c(-1, 1),
  xlab = "", ylab = "", zlab = "", color = rainbow(n), pch = 16
)
scatterplot3d::scatterplot3d(
  samp_3[, , 2], xlim = c(-1, 1), ylim = c(-1, 1), zlim = c(-1, 1),
  xlab = "", ylab = "", zlab = "", color = rainbow(n), pch = 16
)
par(old_par)

# Spiral trends on (S^2)^2
n <- 200
samp_4 <- r_path_s2r(n = n, r = 2, c = 3, spiral = TRUE, sigma = 0.01)
old_par <- par(mfrow = c(1, 2))
scatterplot3d::scatterplot3d(
  samp_4[, , 1], xlim = c(-1, 1), ylim = c(-1, 1), zlim = c(-1, 1),
  xlab = "", ylab = "", zlab = "", color = rainbow(n), pch = 16
)
scatterplot3d::scatterplot3d(
  samp_4[, , 2], xlim = c(-1, 1), ylim = c(-1, 1), zlim = c(-1, 1),
  xlab = "", ylab = "", zlab = "", color = rainbow(n), pch = 16
)
par(old_par)
```

`show_iter_sol` *Visualize the reduction status*

Description

Visualize the iteration solution in a plot for the i-th iteration.

Usage

```
show_iter_sol(Y, i, d, colors = NULL)
```

Arguments

Y	matrix of size `c(n,d)`, where `n` is the number of observation, with the points onto the sphere $\mathcal{S}^d$.
i	the i-th iteration.
d	the dimension to reduce the original data.
colors	defines the group colors in the plot. Optional parameter, default value set to `NULL`.

`smallrna` *Small RNA dataset*

Description

"Small RNA dataset". Among others, used in Section 5.2 in Zoubouloglou et al. (2022) and references therein.

Usage

```
smallrna
```

Format

A data frame with 190 rows and 10 variables:

angles matrix of 7 dihedral angles.

clusters vector of cluster labels.

torsion matrix of 2 torsion angles.

Details

Clusters are defined from the torsion angles, see Section 5.2 in Zoubouloglou et al. (2022) and references therein.

Source

Dataset put together by Duarte and Pyle (1998) and updated by Wadley et al. (2007). Previously used by Eltzner et al. (2018).

References

Duarte, C. M. and Pyle, A. M. (1998). Stepping through an RNA structure: A novel approach to conformational analysis. *Journal of Molecular Biology*, 284(5):1465–1478. doi: 10.1006/jmbi.1998.2233.

Eltzner, B., Huckemann, S., and Mardia, K. V. (2018). Torus principal component analysis with applications to RNA structure. *The Annals of Applied Statistics*, 12(2):1332–1359. doi: 10.1214/17AOAS1115.

Wadley, L. M., Keating, K. S., Duarte, C. M., and Pyle, A. M. (2007). Evaluating and learning from RNA pseudotorsional space: Quantitative validation of a reduced representation for RNA structure. *Journal of Molecular Biology*, 372(4):942–957. doi: 10.1016/j.jmb.2007.06.058.

Zoubouloglou, P., García-Portugués, E., and Marron, J. S. (2022). Scaled torus principal component analysis. *Journal of Computational and Graphical Statistics*. doi: 10.1080/10618600.2022.2119985.

Examples

```
# Load data
data("smallrna")

# Clusters
pairs(smallrna$angles, col = smallrna$clusters, pch = 16)
```

`symmetric_probs` *Symmetric probabilities*

Description

Calculate the symmetric probabilities of a given conditional polyspherical Cauchy probability matrix.

Usage

```
symmetric_probs(P)
```

Arguments

`P`	matrix of probabilities $(P_{i\|j})_{ij}$ with size `c(n,n)` where `n` is the number of observations of the original array `x`.

Value

The sum of `P` and `t(P)` divided by twice the sample size.

Examples

```
symmetric_probs(matrix(runif(3 * 3), nrow = 3, ncol = 3))
symmetric_probs(diag(3))
```

to_perp — *Perplexity matrix (matrix version)*

Description

Calculate the perplexity of each observations for a given ρ parameters list. Matrix version algorithm.

Usage

```
to_perp(x, rho_list, cos_sim_psh = NULL)
```

Arguments

`x`	an array of size `c(n,d + 1,r)` with the polyspherical data, where `n` is the number of observations, `d` is the dimension of each sphere, and `r` is the number of spheres.
`rho_list`	rho list of size `n` for each `i`-th observation that stands for the concentration parameter.
`cos_sim_psh`	a vector of size `n/2` with the cosine similarities in high-dimension for the polysphere $(\mathcal{S}^p)^r$. The way that the cosine similarities matrix is treated makes the calculus faster since it is flat in a vector object. Optional parameter, defaults to `NULL`.

Value

Perplexity of each i-th observation for all the remainder observations.

Examples

```
x <- sphunif::r_unif_sph(25, 3, 3)
to_perp(x, rep(0.5, 25))
to_perp(x, rep(1 - 1e-4, 25), sphunif::Psi_mat(x, scalar_prod = TRUE))
```

to_perplexity_P — *Perplexity of the i-th observation (scalar version)*

Description

Calculate the perplexity of the i-th observation for a given a rho parameter.

Usage

```
to_perplexity_P(x, i, rho)
```

Arguments

`x`	an array of size `c(n,d + 1,r)` with the polyspherical data, where `n` is the number of observations, `d` is the dimension of each sphere, and `r` is the number of spheres.
`i`	corresponds to the i-th observation for the perplexity is calculated.
`rho`	concentration parameter must be in [0, 1).

Value

Perplexity and probabilities of the i-th observation for all the remainder observations.

Examples

```
x <- sphunif::r_unif_sph(25, 3, 3)
to_perplexity_P(x, 1, 0.5)
to_perplexity_P(x, 4, 1 - 1e-4)
```

`to_perp_i` *Perplexity for the i-th observation (vector version)*

Description

Calculate the perplexity of the i-th observation for a given ρ parameters list. Vector version algorithm.

Usage

```
to_perp_i(x, i, rho, cos_sim_psh = NULL)
```

Arguments

`x`	an array of size `c(n,d + 1,r)` with the polyspherical data, where `n` is the number of observations, `d` is the dimension of each sphere, and `r` is the number of spheres.
`i`	corresponds to the i-th observation index.
`rho`	concentration parameter must be in [0, 1).
`cos_sim_psh`	a vector of size `n/2` with the cosine similarities in high-dimension for the polysphere $(\mathcal{S}^p)^r$. The way that the cosine similarities matrix is treated makes the calculus faster since it is flat in a vector object. Optional parameter, defaults to `NULL`.

Value

Perplexity of the i-th observation for all the remainder observations.

Examples

```
x <- sphunif::r_unif_sph(25, 3, 3)
i <- 1
to_perp_i(x, i, 0.25)
```

`vec2matrix` *Reconstruct the symmetric matrix without diagonal*

Description

Convert the vector into a symmetric matrix with diagonal equal to a specific value.

Usage

```
vec2matrix(vec, n, diag_value)
```

Arguments

`vec`	vector containing all non-repeated values of the cosine similarity matrix (upper triangular matrix).
`n`	sample size of the original data.
`diag_value`	value for the diagonal of the resultant matrix.

Value

Matrix formed by the `vec`.

Examples

```
n <- 6
x <- sphunif::r_unif_sph(n, 3)
cos_sim_vec <- drop(sphunif::Psi_mat(x, scalar_prod = TRUE))
vec2matrix(cos_sim_vec, n, diag_value = 1)
```

Index