

# Package ‘pscsne’

September 11, 2022

**Type** Package

**Title** Polyspherical Cauchy SNE

**Version** 0.0.1.900005

**Date** 2022-09-07

**Description** Implementation of polyspherical Cauchy Stochastic Neighbor Embedding, psc-SNE, a nonlinear dimensionality reduction technique for polyspherical data (e.g., circular, spherical or toroidal data). The main function is `psc_sne()`.

**License** GPL-3

**LazyData** true

**Encoding** UTF-8

**Depends** R (>= 3.5.0)

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

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

**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:

psscne-package	2
d_ij_pspsh_cauchy	3
d_i_pspsh_cauchy	3
d_i_sph_cauchy	4
d_sph_cauchy	4
d_total_pspsh_cauchy	5
grid_sphere	5
high_dimension	6
jcondi_pspsh	6
jcondi_sph	7
jdf	8
kl_divergence_grad	9
kms_dir	10
low_dimension_Q	11
prob_cond_i_pspsh	12
prob_cond_i_sph	13
pssc_sne	13
radial_projection	15
rho_optimize_l	16
rho_optim_bst	16
r_block	17
samplers_one	18
show_iter_sol	20
smallrna	20
to_perp	21
to_perplexity_P	22
<b>Index</b>	<b>23</b>

---

psscne-package

psscne: *Polyspherical Cauchy SNE*

---

## Description

Implementation of polyspherical Cauchy Stochastic Neighbor Embedding, pssc-SNE, a nonlinear dimensionality reduction technique for polyspherical data (e.g., circular, spherical or toroidal data). The main function is `pssc_sne()`.

## Author(s)

Eduardo García-Portugués and Luis Ángel Rodríguez García.

---

d_ij_psph_cauchy	<i>Polyspherical Cauchy density for the <math>i</math>-th and <math>j</math>-th observations</i>
------------------	--

---

**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	<i>Marginal polyspherical Cauchy density for the <math>i</math>-th observation</i>
-----------------	--

---

**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	<i>Marginal spherical Cauchy density function</i>
----------------	---

---

**Description**

Calculate the marginal low-dimension spherical Cauchy probability.

**Usage**

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

**Arguments**

y	a matrix of size $c(n, d+1)$ with observations on $S^d$ .
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	<i>High-dimension polyspherical Cauchy density</i>
--------------	--

---

**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	dimension of the sphere, $S^p$ .

**Value**

Spherical Cauchy density value given the parameters.

---

d_total_psph_cauchy	<i>Polyspherical marginal density function values for all the observations</i>
---------------------	--

---

**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  $\rho$  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))
```

---

grid_sphere	<i>Generate optimum evenly separated points</i>
-------------	---

---

**Description**

Generated optimal evenly separated points on  $\mathcal{S}^1$  or  $\mathcal{S}^2$ .

**Usage**

```
grid_sphere(n, d)
```

**Arguments**

n	positive integer with the size of the grid to generate.
d	size of the low-dimension which defines the sphere $\mathcal{S}^d$ , must be 1 or 2.

**Value**

For  $\mathcal{S}^1$ , evenly optimal separated points. For  $\mathcal{S}^2$ , Fibonacci lattice is applied to generate points.  $\mathcal{S}^d$ .

**Examples**

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

---

high_dimension	<i>Polyspherical Cauchy conditional probability matrix</i>
----------------	--

---

**Description**

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

**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 $(S^p)^r$ . 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))
```

---

jcondi_psph	<i>Conditional polyspherical Cauchy probability for the <math>i</math>-th and <math>j</math>-th observations</i>
-------------	--

---

**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**

<code>x</code>	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.
<code>i</code>	corresponds to the $i$ -th observation index.
<code>j</code>	corresponds to the $j$ -th observation index.
<code>rho_list</code>	rho list of size $n$ for each $i$ -th observation that stands for the concentration parameter.
<code>d_total_i_psph_cauchy</code>	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**

<code>y</code>	a matrix of size $c(n, d+1)$ with observations on $S^d$ .
<code>i</code>	corresponds to the $i$ -th observation index.
<code>j</code>	corresponds to the $j$ -th observation index.
<code>rho</code>	concentration parameter must be in $[0, 1)$ .
<code>d_i_sph_cauchy</code>	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>

**Examples**

```
# Load data
data(jdf)
# Plot vector field
time_instant <- as.POSIXct("2021-06-30 03:00:00", tz = "UTC")
jdf_lat_lon <- jdf %>%
  dplyr::filter(time == time_instant) %>%
  unite(col = "coordinates", lat, lon, sep = ",", remove = FALSE)
jdf_by_time_latlon <- array(NA, dim = c(length(unique(jdf_lat_lon$lon)),
                                         length(unique(jdf_lat_lon$lat))))
lat_values <- rev(sort(unique(jdf_lat_lon$lat)))
lon_values <- sort(jdf_lat_lon$lon)

n <- nrow(jdf_lat_lon)
for (i in seq_len(n)) {
  index_lat <- which(round(lat_values, digits = 5) ==
                    round(jdf_lat_lon[i, ]$lat, digits = 5))
  index_lon <- which(round(lon_values, digits = 4) ==
                    round(jdf_lat_lon[i, ]$lon, digits = 4))
  jdf_by_time_latlon[index_lon, index_lat] <- val$theta
}
```



```

}
# Plot vector field
loc <- "Juan de Fuca"
speed <- jdf_lat_lon$speed
tim_i <- format(time_instant, "%Y-%m-%d %H:%M")
OceanView::quiver2D(cos(jdf_by_time_latlon), sin(jdf_by_time_latlon),
  x = lon_values,
  y = lat_values,
  colvar = speed, clim = c(0, 1),
  xlim = c(min(lon_values - 0.015),
    max(lon_values + 0.015)),
  ylim = c(min(lat_values - 0.015),
    max(lat_values + 0.015)),
  main = paste(loc, tim_i), colkey = FALSE,
  xlab = "Latitude", ylab = "Longitude")
points(expand.grid(lon_values, lat_values), pch = 16, cex = 0.5)

```

---

kl_divergence_grad	<i>Kullback–Leibler divergence gradient</i>
--------------------	---

---

## 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 $S^d$ .
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 $(S^p)^r$ . 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  $S^d$ .

**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 $c(n_x, d + 1)$ 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  $n_x$  with matrices of size  $c(n_p, d + 1)$ , where  $n_p$  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 $S^d$ .
$\rho$	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	<i>Conditional polyspherical Cauchy probability for the <math>i</math>-th observation</i>
------------------	---

---

**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**

<code>x</code>	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.
<code>i</code>	corresponds to the $i$ -th observation index.
<code>rho_list</code>	rho list of size $n$ for each $i$ -th observation that stands for the concentration parameter.
<code>d_total_i_psph_cauchy</code>	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	<i>Conditional spherical Cauchy probabilities given the <math>i</math>-th observation</i>
-----------------	---

---

### 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

<code>y</code>	a matrix of size $c(n, d+1)$ with observations on $\mathcal{S}^d$ .
<code>i</code>	corresponds to the $i$ -th observation index.
<code>rho</code>	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	<i>Polyspherical Cauchy SNE</i>
---------	---------------------------------

---

### Description

Computes polyspherical Cauchy SNE given a data onto the polysphere and a dimension to reduce to.

### 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

<code>X</code>	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.
<code>d</code>	the target dimension to use for the reduced data of $X$ .
<code>rho_psc_list</code>	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 <code>rho_values</code> and the matrix <code>P</code> , the second scenario when the type is a vector, then this object contains the rho values, within this function the <code>high_dimension</code> function is called to get the matrix <code>P</code> . The last scenario that is when this object is set to <code>NULL</code> , i.e., the <code>rho_optim_bst</code> function is called to get the rho values (given a fixed perplexity) and the probabilities matrix. Optional parameter, defaults to <code>NULL</code> .
<code>rho</code>	parameter of the low-dimensional spherical Cauchy probabilities. Optional, defaults to <code>0.5</code> .
<code>perplexity</code>	parameter that measures the number of neighbors to use when mapping between high- and low-dimension. Defaults to <code>30</code> .
<code>maxit</code>	maximum number of iterations. Defaults to <code>1e3</code> .
<code>initial_momentum</code>	first value of the momentum of the first 250 iterations. Defaults to <code>0.5</code> .
<code>final_momentum</code>	momentum to take into account after the 250 iteration. Defaults to <code>0.8</code> .
<code>eta</code>	is the learning rate of the optimization algorithm. Optional param, defaults to <code>200</code> .
<code>early_exaggeration</code>	the first 100 iterations results are exaggerated by this factor. Optional parameter, defaults to <code>4.0</code> .
<code>colors</code>	list with as many elements as observations are, only valid when visualization is true. Optional parameter, defaults to <code>NULL</code> .
<code>show_prog</code>	defines the number of iterations skipped when reporting the progress. Defaults to <code>100</code> , i.e., only multiples of <code>100</code> are reported. If <code>FALSE</code> , no progress is shown at all.
<code>show_plots</code>	show convergence plots? If <code>TRUE</code> (default), a plot is shown: after $2 * \text{show\_prog}$ iterations and at the end of the search.
<code>tol</code>	is the tolerance, when is below this value it is considered that a good solution has been obtained. Defaults to <code>1e-6</code> .
<code>parallel_cores</code>	number of cores to use concurrently for the calculation of the gradient. Defaults to <code>parallel::detectCores() - 1</code> , that means that uses the total number of cores of the computer except one of them.
<code>init</code>	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 <code>maxit / 100</code> iterations), or a matrix. Defaults to "equispaced".
<code>N</code>	number of differently-initialized searches (see above). Defaults to <code>10</code> .

## 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 of 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	<i>Radial projection onto the sphere</i>
-------------------	--

---

**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	<i>Concurrent optimization of the <math>\rho</math> concentration parameters (matrix version)</i>
----------------	---

---

### Description

Calculate the rho list values based on a fixed perplexity and a given data in  $(S^p)^r$ . 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 $(S^p)^r$ . 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 ( $\rho$ ) with the values optimized for the given perplexity.

---

rho_optim_bst	<i>Binary search rho optimization for each observation</i>
---------------	--

---

### Description

Calculate the rho list values based on a fixed perplexity and a given data in  $(S^p)^r$ . 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)
```

---

r_block	<i>Sample correlation matrices with block structure</i>
---------	---

---

**Description**

Sample a zero-mean multivariate normal  $N_{gp}(\mathbf{0}, \Sigma)$  with a diagonal block matrix  $\Sigma$  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 <a href="#">toeplitz</a> 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  $\sqrt{n-1} \times S^{n-1}!$ 
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	<i>Samplers of one-dimensional modes of variation for polyspherical data</i>
--------------	--

---

## Description

Functions for sampling data on  $(\mathcal{S}^d)^r$  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 $(\mathcal{S}^d)^r$ .
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 <b>integer</b> 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	<b>Clélie curve</b> 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 <b>Clélie curve</b> ) 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	<i>Visualize the reduction status</i>
---------------	---------------------------------------

---

## 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 $S^d$ .
$i$	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	<i>Small RNA dataset</i>
----------	--------------------------

---

## 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.
- Zouboulglou, 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)
```

---

to_perp	<i>Perplexity matrix (matrix version)</i>
---------	---

---

## Description

Calculate the perplexity of each observations for a given  $\rho$  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	<i>Perplexity of the <math>i</math>-th observation (scalar version)</i>
-----------------	---

---

**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)
```

# Index

## \* datasets

jdf, [8](#)  
smallrna, [20](#)

d\_i\_psph\_cauchy, [3](#)  
d\_i\_sph\_cauchy, [4](#)  
d\_ij\_psph\_cauchy, [3](#)  
d\_sph\_cauchy, [4](#)  
d\_total\_psph\_cauchy, [5](#)

grid\_sphere, [5](#)

high\_dimension, [6](#), [14](#)

jcondi\_psph, [6](#)  
jcondi\_sph, [7](#)  
jdf, [8](#)

kl\_divergence\_grad, [9](#)  
kms\_dir, [10](#)

low\_dimension\_Q, [11](#)

prob\_cond\_i\_psph, [12](#)  
prob\_cond\_i\_sph, [13](#)  
psc\_sne, [13](#)  
psc\_sne (psc\_sne-package), [2](#)  
psc\_sne-package, [2](#)

r\_block, [17](#)  
r\_path\_slr (samplers\_one), [18](#)  
r\_path\_s2r (samplers\_one), [18](#)  
radial\_projection, [15](#)  
rho\_optim\_bst, [14](#), [16](#)  
rho\_optimize\_1, [16](#)

samplers\_one, [18](#)  
show\_iter\_sol, [20](#)  
smallrna, [20](#)

to\_perp, [21](#)  
to\_perplexity\_P, [22](#)  
toeplitz, [17](#)