Package 'cardinalR'

August 21, 2025

Type Package

Title Collection of Data Structures

Version 0.1.10

Description A collection of functions to generate a large variety of structures in high dimensions. These data structures are useful for testing, validating, and improving algorithms used in dimensionality reduction, clustering, machine learning, and visualization.

License MIT + file LICENSE

URL https://jayanilakshika.github.io/cardinalR/

BugReports https://github.com/JayaniLakshika/cardinalR/issues

Depends R (>= 4.1.0)

Imports cli, dplyr, geozoo, gtools, MASS, mvtnorm, purrr, stats, tibble, tidyr

Suggests knitr, langevitour, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en-GB

LazyData true

RoxygenNote 7.3.2

NeedsCompilation no

Author Jayani P. Gamage [aut, cre] (ORCID: <https://orcid.org/0000-0002-6265-6481>), Dianne Cook [aut] (ORCID: <https://orcid.org/0000-0002-3813-7155>), Paul Harrison [aut] (ORCID: <https://orcid.org/0000-0002-3980-268X>), Michael Lydeamore [aut] (ORCID: <https://orcid.org/0000-0001-6515-827X>), Thiyanga S. Talagala [aut] (ORCID: <https://orcid.org/0000-0002-0656-9789>)

Maintainer Jayani P. Gamage <jayanilakshika76@gmail.com>

Repository CRAN

Date/Publication 2025-08-21 07:10:07 UTC

Contents

gen_bkgnoise *Generate Background Noise Data*

Description

This function generates background noise data with specified parameters such as the number of samples, number of dimensions, mean, and standard deviation.

Usage

```
gen_bkgnoise(n = 500, p = 4, m = rep(0, p), s = rep(2, p))
```

Arguments

n	A numeric value (default: 500) representing the sample size.
p	A numeric value (default: 4) representing the number of dimensions.
m	A numeric vector (default: c(0, 0, 0, 0)) representing the mean along each dimensions.
s	A numeric vector (default: c(2, 2, 2, 2)) representing the standard deviation along each dimensions.

Value

A data containing the generated background noise data.

Examples

```
# Generate background noise with custom mean and standard deviation
set.seed(20240412)
gen_bkgnoise(n = 500, p = 4, m = c(0, 0, 0, 0), s = c(2, 2, 2, 2))
```

gen_circle *Generate Circle in p-d*

Description

This function generates a dataset representing a structure with a circle.

Usage

```
gen_circle(n = 500, p = 4)
```

Arguments

n	A numeric value (default: 500) representing the sample size.
p	A numeric value (default: 4) representing the number of dimensions.

Value

A data containing a circle.

Examples

```
set.seed(20240412)
circle <- gen_circle(n = 500, p = 4)
```

---

gen_clusteredspheres *Generate Small Spheres Within a Big Sphere*

Description

This function generates a dataset representing a structure with a small and big spheres.

Usage

```
gen_clusteredspheres(
  n = c(1000, 100),
  k = 3,
  p = 4,
  r = c(15, 3),
  loc = 10/sqrt(3)
)
```

Arguments

n	A numeric vector (default: c(1000, 100)) representing the sample sizes of the big and small spheres respectively.
k	A numeric value (default: 3) representing the number of small spheres.
p	A numeric value (default: 4) representing the number of dimensions.
r	A numeric vector (default: c(15, 3)) representing the radius of the big and small spheres respectively.
loc	A numeric value (default: 10 / sqrt(3) representing how far the small spheres are placed from each other.

Value

A data containing small spheres within a big sphere.

Examples

```
set.seed(20240412)
clusteredspheres <- gen_clusteredspheres(n = c(1000, 100), k = 3, p = 4,
r = c(15, 3), loc = 10 / sqrt(3))
```

---

gen_clustloc *Generate Cluster Locations*

Description

This function generate locations for any number of clusters in any dimensions.

Usage

```
gen_clustloc(p = 4, k = 3)
```

Arguments

p	A numeric value (default: 4) representing the number of dimensions.
k	A numeric value (default: 3) representing the number of clusters.

Value

A matrix of the locations.

Examples

```
set.seed(20240412)
gen_clustloc(p = 4, k = 3)
```

---

gen_cone *Generate Blunted Cone*

Description

This function generates a dataset representing a cone with the option of a sharp or blunted apex.

Usage

```
gen_cone(n = 500, p = 4, h = 5, ratio = 0.5)
```

Arguments

n	A numeric value (default: 500) representing the sample size.
p	A numeric value (default: 4) representing the number of dimensions.
h	A numeric value (default: 5) representing the h of the cone.
ratio	A numeric value (default: 0.5) representing the radius tip to radius base ratio of the cone. Should be less than 1.

Value

A data containing the cone with the option of a sharp or blunted apex.

Examples

```
set.seed(20240412)
cone <- gen_cone(n = 500, p = 4, h = 5, ratio = 0.5)
```

---

gen_conicspiral *Generate Conical Spiral*

Description

This function generates a dataset representing a conical spiral structure.

Usage

```
gen_conicspiral(n = 500, p = 4, spins = 1)
```

Arguments

n	A numeric value (default: 500) representing the sample size.
p	A numeric value (default: 4) representing the number of dimensions.
spins	A numeric value (default: 1) representing the number of loops of the spiral.

Value

A data containing a conical spiral structure.

Examples

```
set.seed(20240412)
conicspiral <- gen_conicspiral(n = 500, p = 4, spins = 1)
```

gen_crescent *Generate Crescent*

Description

This function generates a dataset representing a structure with a Crescent pattern.

Usage

```
gen_crescent(n = 500, p = 4)
```

Arguments

n	A numeric value (default: 500) representing the sample size.
p	A numeric value (default: 4) representing the number of dimensions.

Value

A data containing a Crescent structure.

Examples

```
set.seed(20240412)
crescent <- gen_crescent(n = 500, p = 4)
```

gen_cubehole *Generate Cube with Hole*

Description

This function generates a dataset representing a cube with a hole.

Usage

```
gen_cubehole(n = 500, p = 4)
```

Arguments

n	A numeric value (default: 500) representing the sample size.
p	A numeric value (default: 4) representing the number of dimensions.

Value

A data containing the cube data with a hole.

Examples

```
set.seed(20240412)
cubehole <- gen_cubehole(n = 1000, p = 4)
```

---

gen_cubic *Generate Cubic*

Description

This function generates a dataset representing a structure with a cubic pattern.

Usage

```
gen_cubic(n = 500, p = 4, range = c(-1, 2))
```

Arguments

n	A numeric value (default: 500) representing the sample size.
p	A numeric value (default: 4) representing the number of dimensions.
range	A numeric vector (default: c(-1, 2)) representing the range along x1 axis.

Value

A data containing a cubic structure.

Examples

```
set.seed(20240412)
cubic <- gen_cubic(n = 500, p = 4)
```

---

gen_curvybranches *Generate data with curvy shaped branches*

Description

This function generates a dataset representing a structure with non-linear shaped branches.

Usage

```
gen_curvybranches(n = 400, p = 4, k = 4)
```

Arguments

n	A numeric value (default: 400) representing the sample size.
p	A numeric value (default: 4) representing the number of dimensions.
k	A numeric value (default: 4) representing the number of branches.

Value

A data containing non-linear shaped branches.

Examples

```
set.seed(20240412)
curvybranches <- gen_curvybranches(n = 400, p = 4, k = 4)
```

`gen_curvycycle` *Generate Curvy Cell Cycle in p-d*

Description

This function generates a dataset representing a structure with a curvy cell cycle.

Usage

```
gen_curvycycle(n = 500, p = 4)
```

Arguments

n	A numeric value (default: 500) representing the sample size.
p	A numeric value (default: 4) representing the number of dimensions.

Value

A data containing a curvy cell cycle.

Examples

```
set.seed(20240412)
curvycycle <- gen_curvycycle(n = 500, p = 4)
```

`gen_curvycylinder` *Generate Curvy Cylinder*

Description

This function generates a dataset representing a structure with a curvy cylinder.

Usage

```
gen_curvycylinder(n = 500, p = 4, h = 10)
```

Arguments

n	A numeric value (default: 500) representing the sample size.
p	A numeric value (default: 4) representing the number of dimensions.
h	A numeric value (default: 10) representing the height of the cylinder.

Value

A data containing a curvy cylinder.

Examples

```
set.seed(20240412)
curvycylinder <- gen_curvycylinder(n = 500, p = 4, h = 10)
```

---

gen_expbranches *Generate data with exponential shaped branches*

Description

This function generates a dataset representing a structure with exponential shaped branches.

Usage

```
gen_expbranches(n = 400, p = 4, k = 4)
```

Arguments

n	A numeric value (default: 400) representing the sample size.
p	A numeric value (default: 4) representing the number of dimensions.
k	A numeric value (default: 4) representing the number of branches.

Value

A data containing exponential shaped branches.

Examples

```
set.seed(20240412)
expbranches <- gen_expbranches(n = 400, p = 4, k = 4)
```

gen_gaussian *Generate Gaussian*

Description

This function generates a dataset representing a structure with a Gaussian.

Usage

```
gen_gaussian(n = 500, p = 4, s = diag(p) * 0.01)
```

Arguments

n	A numeric value (default: 500) representing the sample size.
p	A numeric value (default: 4) representing the number of dimensions.
s	A numeric matrix (default: diag(4) * 0.01) representing the variance of along each dimension.

Value

A data containing a Gaussian.

Examples

```
set.seed(20240412)
gaussian <- gen_gaussian(n = 500, p = 4, s = diag(4))
```

gen_gridcube *Generate Cube with grid points*

Description

This function generates a grid dataset with specified grid points along each axes.

Usage

```
gen_gridcube(n = 500, p = 4)
```

Arguments

n	A numeric vector (default: 500) representing the sample size.
p	A numeric value (default: 4) representing the number of dimensions.

Value

A data containing the cube with grid points.

Examples

```
set.seed(20240412)
gridcube <- gen_gridcube(n = 500, p = 4)
```

`gen_gridedsphere` *Generate Grided Sphere*

Description

This function generates a dataset representing a structure with a grided sphere.

Usage

```
gen_gridedsphere(n = 500, p = 4)
```

Arguments

`n`	A numeric value (default: 500) representing the sample size.
`p`	A numeric value (default: 4) representing the number of dimensions.

Value

A data containing a grided sphere.

Examples

```
set.seed(20240412)
gridedsphere <- gen_gridedsphere(n = 500, p = 4)
```

`gen_helicalspiral` *Generate Helical Hyper Spiral*

Description

This function generates a dataset representing a structure with a helical hyper spiral.

Usage

```
gen_helicalspiral(n = 500, p = 4)
```

Arguments

`n`	A numeric value (default: 500) representing the sample size.
`p`	A numeric value (default: 4) representing the number of dimensions.

Value

A data containing a helical hyper spiral.

Examples

```
set.seed(20240412)
helicalspiral <- gen_helicalspiral(n = 500, p = 4)
```

`gen_hemisphere` *Generate Hemisphere*

Description

This function generates a dataset representing a structure with a hemisphere.

Usage

```
gen_hemisphere(n = 500, p = 4)
```

Arguments

n	A numeric value (default: 500) representing the sample size.
p	A numeric value (default: 4) representing the number of dimensions.

Value

A data containing a hemisphere.

Examples

```
set.seed(20240412)
hemisphere <- gen_hemisphere(n = 500, p = 4)
```

`gen_linearbranches` *Generate data with linear shaped branches*

Description

This function generates a dataset representing a structure with linear shaped branches.

Usage

```
gen_linearbranches(n = 400, p = 4, k = 4)
```

Arguments

n	A numeric value (default: 400) representing the sample size.
p	A numeric value (default: 4) representing the number of dimensions.
k	A numeric value (default: 4) representing the number of branches.

Value

A data containing linear shaped branches.

Examples

```
set.seed(20240412)
linearbranches <- gen_linearbranches(n = 400, p = 4, k = 4)
```

---

gen_longlinear *Generate Long Linear Data*

Description

This function generates a dataset consisting of long linear data.

Usage

```
gen_longlinear(n = 500, p = 4)
```

Arguments

n	A numeric value (default: 500) representing the sample size.
p	A numeric value (default: 4) representing the number of dimensions.

Value

A data containing the long linear data.

Examples

```
set.seed(20240412)
longlinear <- gen_longlinear(n = 500, p = 4)
```

gen_mobius *Generate a 3-D Mobius in High Dimensions*

Description

This function generates a dataset representing a structure with a mobius.

Usage

```
gen_mobius(n = 500, p = 4)
```

Arguments

n	A numeric value (default: 500) representing the sample size.
p	A numeric value (default: 4) representing the number of dimensions.

Value

A data containing a mobius structure.

Examples

```
set.seed(20240412)
mobius <- gen_mobius(n = 500, p = 4)
```

gen_multicluster *Generate Multiple Clusters*

Description

This function generates a dataset with multiple clusters.

Usage

```
gen_multicluster(
  n = c(200, 300, 500),
  p = 4,
  k = 3,
  loc = matrix(c(0, 0, 0, 0, 5, 9, 0, 0, 3, 4, 10, 7), nrow = 3, byrow = TRUE),
  scale = c(3, 1, 2),
  shape = c("gaussian", "bluntedcorn", "unifcube"),
  rotation = NULL,
  is_bkg = FALSE
)
```

Arguments

n	A numeric vector (default: c(200, 500, 300)) representing the sample sizes.
p	A numeric value (default: 4) representing the number of dimensions.
k	A numeric value (default: 3) representing the number of clusters.
loc	A numeric matrix (default: matrix(c(0, 0, 0, 0, 5, 9, 0, 0, 3, 4, 10, 7), nrow = 3, byrow = TRUE)) representing the locations/centroids of clusters.
scale	A numeric vector (default: c(3, 1, 2)) representing the scaling factors of clusters.
shape	A character vector (default: c("gen_gaussian", "gen_cone", "gen_unifcube")) representing the shapes of clusters.
rotation	A numeric list which contains plane and the corresponding angle along that plane for each cluster.
is_bkg	A Boolean value (default: FALSE) representing the background noise should exist or not.

Value

A data containing same/different shaped clusters.

Examples

```
set.seed(20240412)
rotations_4d <- list(
cluster1 = list(
  list(plane = c(1, 2), angle = 60), # Rotation in the (1, 2) plane
  list(plane = c(3, 4), angle = 90)  # Rotation in the (3, 4) plane
  ),
cluster2 = list(
  list(plane = c(1, 3), angle = 30) # Rotation in the (1, 3) plane
  ),
cluster3 = list(
  list(plane = c(2, 4), angle = 45) # Rotation in the (2, 4) plane
  )
)
clust_data <- gen_multicluster(n = c(200, 300, 500), p = 4, k = 3,
loc = matrix(c(
  0, 0, 0, 0,
  5, 9, 0, 0,
  3, 4, 10, 7
), nrow = 3, byrow = TRUE),
scale = c(3, 1, 2),
shape = c("gaussian", "cone", "unifcube"),
rotation = rotations_4d,
is_bkg = FALSE)
```

gen_noisedims *Generate Random Noise Dimensions*

Description

This function generates random noise dimensions to be added to the coordinates of a data structure.

Usage

```
gen_noisedims(n = 500, p = 4, m = rep(0, p), s = rep(2, p))
```

Arguments

n	A numeric value (default: 500) representing the sample size.
p	A numeric value (default: 4) representing the number of dimensions.
m	A numeric vector (default: c(0, 0, 0, 0)) representing the mean along each dimensions.
s	A numeric vector (default: c(2, 2, 2, 2)) representing the standard deviation along each dimensions.

Value

A data containing the generated random noise dimensions.

Examples

```
set.seed(20240412)
gen_noisedims(n = 500, p = 4, m = c(0, 0, 0, 0), s = c(2, 2, 2, 2))
```

gen_nonlinear *Generate Nonlinear Hyperbola*

Description

This function generates a dataset representing a nonlinear hyperbola structure.

Usage

```
gen_nonlinear(n = 500, p = 4, hc = 1, non_fac = 0.5)
```

Arguments

n	A numeric value (default: 500) representing the sample size.
p	A numeric value (default: 4) representing the number of dimensions.
hc	A numeric value (default: 1) representing the hyperbolic component which define the steepness and vertical scaling of the hyperbola. Larger values of this make the curve more pronounced (sharper dips/rises near 0), while smaller values make it flatter.
non_fac	A numeric value (default: 1) representing the nonlinear factor which describes the strength of this sinusoidal effect. When this is 0, the curve is purely hyperbolic; as it increases, the wave-like fluctuations become more prominent.

Value

A data containing a nonlinear hyperbola structure.

Examples

```
set.seed(20240412)
nonlinear <- gen_nonlinear(n = 500, p = 4, hc = 1, non_fac = 0.5)
```

---

gen_nproduct *Generates a vector of positive integers whose product is approximately equal to a target value.*

Description

This function takes a target integer ‘n‘ and the number of dimensions ‘p‘, and returns a vector ‘n_vec‘ of length ‘p‘ containing positive integers. The goal is to have the product of the elements in ‘n_vec‘ be as close as possible to ‘n‘, especially when ‘n‘ is not a perfect p-th power.

Usage

```
gen_nproduct(n = 500, p = 4)
```

Arguments

n	The target positive integer value for the product of the output vector.
p	The number of dimensions (the length of the output vector). Must be a positive integer.

Value

A sorted vector of positive integers of length ‘p‘. The product of the elements in this vector will be approximately equal to ‘n‘. If ‘n‘ is a perfect p-th power, the elements will be equal.

Examples

```
gen_nproduct(500, 6) # Example with n=500, p=6
gen_nproduct(700, 4) # Example with n=700, p=4
gen_nproduct(625, 4) # Example with n=625 (perfect power)
gen_nproduct(30, 3)  # Example with n=30, p=3
gen_nproduct(7, 2)   # Example where exact product might be hard
```

`gen_nsum` *Generates a vector of positive integers whose summation is approximately equal to a target value.*

Description

This function takes a target integer ‘n‘ and the number of clusters ‘k‘, and returns a vector ‘n_vec‘ of length ‘k‘ containing positive integers. The goal is to have the summation of the elements in ‘n_vec‘ be as close as possible to ‘n‘, especially when ‘n‘ is not a perfect multiplier of ‘k‘.

Usage

```
gen_nsum(n = 500, k = 4)
```

Arguments

`n`	The target positive integer value for the summation of the output vector.
`k`	The number of dimensions (the length of the output vector). Must be a positive integer.

Value

A sorted vector of positive integers of length ‘k‘. The summation of the elements in this vector will be approximately equal to ‘n‘. If ‘n‘ is a perfectly divisible by ‘k‘, the elements will be equal.

Examples

```
gen_nsum(500, 6) # Example with n=500, p=6
gen_nsum(700, 4) # Example with n=700, p=4
gen_nsum(625, 5) # Example with n=625 (perfect division)
gen_nsum(30, 3)  # Example with n=30, p=3
```

gen_orgcurvybranches *Generate data with curvy shaped branches in a initial point*

Description

This function generates a dataset representing a structure with curvy shaped branches.

Usage

```
gen_orgcurvybranches(n = 400, p = 4, k = 4)
```

Arguments

n	A numeric value (default: 400) representing the sample size.
p	A numeric value (default: 4) representing the number of dimensions.
k	A numeric value (default: 4) representing the number of branches.

Value

A data containing curvy shaped branches originated in one point.

Examples

```
set.seed(20240412)
orgcurvybranches <- gen_orgcurvybranches(n = 400, p = 4, k = 4)
```

gen_orglinearbranches *Generate data with linear shaped branches in a initial point*

Description

This function generates a dataset representing a structure with linear shaped branches.

Usage

```
gen_orglinearbranches(n = 400, p = 4, k = 4)
```

Arguments

n	A numeric value (default: 400) representing the sample size.
p	A numeric value (default: 4) representing the number of dimensions.
k	A numeric value (default: 4) representing the number of branches.

Value

A data containing linear shaped branches originated in one point.

Examples

```
set.seed(20240412)
orglinearbranches <- gen_orglinearbranches(n = 400, p = 4, k = 4)
```

`gen_pyrholes` *Generate p-D Triangular Pyramid With Triangular Pyramid shaped holes*

Description

This function generates p-D triangular pyramid with triangular pyramid shaped holes.

Usage

```
gen_pyrholes(n = 500, p = 4)
```

Arguments

`n`	A numeric value (default: 500) representing the sample size.
`p`	A numeric value (default: 4) representing the number of dimensions.

Value

A data containing a triangular pyramid with triangular pyramid shaped holes.

Examples

```
set.seed(20240412)
pyrholes <- gen_pyrholes(n = 500, p = 3)
```

`gen_pyrrect` *Generate Rectangular Based Pyramid*

Description

This function generates a dataset representing a rectangular based pyramid.

Usage

```
gen_pyrrect(n = 500, p = 4, h = 5, l_vec = c(3, 2), rt = 0.5)
```

Arguments

n	A numeric value (default: 500) representing the sample size.
p	A numeric value (default: 4) representing the number of dimensions.
h	A numeric value (default: 5) representing the height of the pyramid.
l_vec	A numeric vector (default: c(3, 2)) representing the base lengths along the and y of the pyramid.
rt	A numeric value (default: 0.5) representing the tip radius of the pyramid.

Value

A data containing the rectangular based pyramid.

Examples

```
set.seed(20240412)
pyrrect <- gen_pyrrect(n = 500, p = 4, h = 5, l_vec = c(3, 2), rt = 0.5)
```

---

gen_pyrstar *Generate Star Based Pyramid*

Description

This function generates a dataset representing a star based pyramid.

Usage

```
gen_pyrstar(n = 500, p = 4, h = 5, rb = 3)
```

Arguments

n	A numeric value (default: 500) representing the sample size.
p	A numeric value (default: 4) representing the number of dimensions.
h	A numeric value (default: 5) representing the height of the pyramid.
rb	A numeric value (default: 3) representing the base radius of the pyramid.

Value

A data containing the star based pyramid.

Examples

```
set.seed(20240412)
pyrstar <- gen_pyrstar(n = 500, p = 4, h = 5, rb = 3)
```

gen_pyrtri *Generate Triangular Based Pyramid*

Description

This function generates a dataset representing a triangular based pyramid.

Usage

```
gen_pyrtri(n = 500, p = 4, h = 5, l = 3, rt = 0.5)
```

Arguments

n	A numeric value (default: 500) representing the sample size.
p	A numeric value (default: 4) representing the number of dimensions.
h	A numeric value (default: 5) representing the height of the pyramid.
l	A numeric value (default: 3) representing the base length of the pyramid.
rt	A numeric value (default: 0.5) representing the tip radius of the pyramid.

Value

A data containing the triangular based pyramid.

Examples

```
set.seed(20240412)
pyrtri <- gen_pyrtri(n = 500, p = 4, h = 5, l = 3, rt = 0.5)
```

gen_quadratic *Generate Quadratic*

Description

This function generates a dataset representing a structure with a quadratic pattern.

Usage

```
gen_quadratic(n = 500, p = 4, range = c(-1, 1))
```

Arguments

n	A numeric value (default: 500) representing the sample size.
p	A numeric value (default: 4) representing the number of dimensions.
range	A numeric vector (default: c(-1, 1)) representing the range along x1 axis.

Value

A data containing a quadratic structure.

Examples

```
set.seed(20240412)
quadratic <- gen_quadratic(n = 500, p = 4)
```

`gen_rotation` *Generate Rotations*

Description

This function generates a rotation matrix.

Usage

```
gen_rotation(p = 4, planes_angles)
```

Arguments

`p`	A numeric value (default: 4) representing the number of dimensions.
`planes_angles`	A numeric list which contains plane and the corresponding angle along that plane.

Value

A matrix containing the rotations.

Examples

```
set.seed(20240412)
rotations_4d <- list(
  list(plane = c(1, 2), angle = 60), # Rotation in the (1, 2) plane
  list(plane = c(3, 4), angle = 90)  # Rotation in the (3, 4) plane
)
gen_rotation(p = 4, planes_angles = rotations_4d)
```

gen_scurve *Generate S-curve Data*

Description

This function generates S-curve data.

Usage

```
gen_scurve(n = 500, p = 4)
```

Arguments

n	A numeric value (default: 500) representing the sample size.
p	A numeric value (default: 4) representing the number of dimensions.

Value

A data containing the generated S-curve data.

References

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., ... & Varoquaux, G. (2013). API design for machine learning software: experiences from the scikit-learn project. *arXiv preprint arXiv:1309.0238*.

Examples

```
set.seed(20240412)
scurve <- gen_scurve(n = 500, p = 4)
```

gen_scurvehole *Generate S-curve Data with a Hole*

Description

This function generates S-curve data with a hole by filtering out samples that are not close to a specified anchor point.

Usage

```
gen_scurvehole(n = 500, p = 4)
```

Arguments

n	A numeric value (default: 500) representing the sample size.
p	A numeric value (default: 4) representing the number of dimensions.

Value

A data containing the generated S-curve data with a hole.

References

Wang, Y., Huang, H., Rudin, C., & Shaposhnik, Y. (2021). Understanding how dimension reduction tools work: an empirical approach to deciphering t-SNE, UMAP, TriMAP, and PaCMAP for data visualization. *J Mach. Learn. Res*, *22*, 1-73.

See Also

the PaCMAP homepage.

Examples

```
set.seed(20240412)
scurvehole <- gen_scurvehole(n = 500, p = 4)
```

---

gen_sphericalspiral *Generate Spherical Spiral*

Description

This function generates a dataset representing a structure with a spherical spiral.

Usage

```
gen_sphericalspiral(n = 500, p = 4, spins = 1)
```

Arguments

n	A numeric value (default: 500) representing the sample size.
p	A numeric value (default: 4) representing the number of dimensions.
spins	A numeric value (default: 1) representing the number of loops of the spiral.

Value

A data containing a spherical spiral.

Examples

```
set.seed(20240412)
sphericalspiral <- gen_sphericalspiral(n = 500, p = 4, spins = 1)
```

gen_swissroll *Generate Swiss Roll Data*

Description

This function generates swiss roll data.

Usage

```
gen_swissroll(n = 500, p = 4, w = c(-1, 1))
```

Arguments

n	A numeric value (default: 500) representing the sample size.
p	A numeric value (default: 4) representing the number of dimensions.
w	A numeric vector (default: c(-1, 1)) representing the vertical variation.

Value

A data containing the generated swiss roll data.

References

Agrafiotis, D. K., & Xu, H. (2002). A self-organizing principle for learning nonlinear manifolds. *Proceedings of the National Academy of Sciences*, *99*(25), 15869-15872.

Roweis, S. T., & Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, *290*(5500), 2323-2326.

Examples

```
set.seed(20240412)
swissroll <- gen_swissroll(n = 500, p = 4)
```

gen_trefoil3d *Generate 3-D Trefoil Knot Coordinates (Stereographic Projection)*

Description

This function generates coordinates for a 3-D trefoil knot by applying a stereographic projection from 4-D space.

Usage

```
gen_trefoil3d(n = 500, p = 4, steps = 5)
```

Arguments

n	A numeric value (default: 500) representing the sample size.
p	A numeric value (default: 4) representing the number of dimensions.
steps	A numeric value (default: 5) representing the number of steps for the theta parameter.

Value

A data containing 3-D trefoil knot.

Examples

```
set.seed(20240412)
trefoil3d <- gen_trefoil3d(n = 500, p = 4, steps = 5)
```

---

gen_trefoil4d *Generate 4-D Trefoil Knot Coordinates*

Description

This function generates coordinates for a 4-D trefoil knot. The number of points is determined by the length of the theta and phi sequences.

Usage

```
gen_trefoil4d(n = 500, p = 4, steps = 5)
```

Arguments

n	A numeric value (default: 500) representing the sample size.
p	A numeric value (default: 4) representing the number of dimensions.
steps	A numeric value (default: 5) representing the number of steps for the theta parameter.

Value

A data containing 4-D trefoil knot.

Examples

```
set.seed(20240412)
trefoil4d <- gen_trefoil4d(n = 500, p = 4, steps = 5)
```

gen_unifcube *Generate Cube with uniform points*

Description

This function generates a grid dataset with specified uniform points along each axes..

Usage

```
gen_unifcube(n = 500, p = 4)
```

Arguments

n	A numeric vector (default: 500) representing the sample size.
p	A numeric value (default: 4) representing the number of dimensions.

Value

A data containing the cube with uniform points.

Examples

```
set.seed(20240412)
unifcube <- gen_unifcube(n = 500, p = 4)
```

gen_unifsphere *Generate Uniform Sphere*

Description

This function generates a dataset representing a structure with a uniform sphere.

Usage

```
gen_unifsphere(n = 500, p = 4, r = 1)
```

Arguments

n	A numeric value (default: 500) representing the sample size.
p	A numeric value (default: 4) representing the number of dimensions.
r	A numeric vector (default: 1) representing the radius of the sphere.

Value

A data containing a uniform sphere.

Examples

```
set.seed(20240412)
unifsphere <- gen_unifsphere(n = 500, p = 4)
```

---

gen_wavydims1 *Generate Random Noise Dimensions With Wavy Pattern*

Description

This function generates random noise dimensions by adding wavy patterns.

Usage

```
gen_wavydims1(n = 500, p = 4, theta = seq(pi/6, 12 * pi/6, length.out = 500))
```

Arguments

n	A numeric value (default: 500) representing the sample size.
p	A numeric value (default: 4) representing the number of dimensions.
theta	A numeric vector representing the nonlinearity along each dimensions.

Value

A data containing the generated random noise dimensions.

Examples

```
set.seed(20240412)
gen_wavydims1(n = 500, p = 4, theta = seq(pi / 6, 12 * pi / 6, length.out = 500))
```

---

gen_wavydims2 *Generate Random Noise Dimensions With Wavy Pattern*

Description

This function generates random noise dimensions by adding wavy patterns.

Usage

```
gen_wavydims2(n = 500, p = 4, x1_vec)
```

Arguments

n	A numeric value (default: 500) representing the sample size.
p	A numeric value (default: 4) representing the number of dimensions.
x1_vec	A numeric vector representing the first dimension of the data structure.

Value

A data containing the generated random noise dimensions.

Examples

```
set.seed(20240412)
theta <- seq(0, 2 * pi, length.out = 500)
x1 <- sin(pi) * cos(theta)
gen_wavydims2(n = 500, p = 4, x1_vec = x1)
```

---

gen_wavydims3 *Generate Random Noise Dimensions With Wavy Pattern*

Description

This function generates random noise dimensions by adding wavy patterns.

Usage

```
gen_wavydims3(n = 500, p = 4, data)
```

Arguments

n	A numeric value (default: 500) representing the sample size.
p	A numeric value (default: 4) representing the number of dimensions.
data	A matrix representing the first three dimensions of the data structure.

Value

A data containing the generated random noise dimensions.

Examples

```
set.seed(20240412)
df <- gen_scurve(n = 500, p = 4) |> as.matrix()
gen_wavydims3(n = 500, p = 4, data = df)
```

make_chain_circles *Generate a Chain of Interlocked Circles in High-Dimensional Space*

Description

This function generates k interlocked circular clusters in a p-dimensional space. Unlike `make_klink_circles()`, the circles are arranged in a **chain-like structure**, where each circle interlocks only with its immediate neighbor, resembling links in a chain.

Usage

```
make_chain_circles(n = c(200, 100), p = 4, k = 2, offset = 0.5, angle = 90)
```

Arguments

`n`	An integer vector of length k giving the number of points in each circle. Default is `c(200, 100)`.
`p`	Integer, the dimensionality of the embedding space. Must be at least 3. Default is `4`.
`k`	Integer, the number of circles to generate. Default is `2`.
`offset`	Numeric, the positional shift applied to each circle along its linking axis to ensure interlocking instead of overlap. Default is `0.5`.
`angle`	Numeric, the rotation angle (in degrees) used when placing each subsequent circle into its respective plane. Default is `90`.

Value

A data frame (or tibble, depending on `gen_multicluster()`) containing the generated points and cluster assignments.

Examples

```
# Generate two chain-linked circles in 4-D
twochain_circles <- make_chain_circles()
```

make_chain_curvycycle *Generate a Chain of Interlocked curvycycle in High-Dimensional Space*

Description

This function generates k interlocked circular clusters in a p-dimensional space. Unlike `make_klink_curvycycle()`, the curvycycle are arranged in a **chain-like structure**, where each curvycycle interlocks only with its immediate neighbor, resembling links in a chain.

Usage

```
make_chain_curvycycle(n = c(200, 100), p = 4, k = 2, offset = 0.5, angle = 90)
```

Arguments

n	An integer vector of length k giving the number of points in each curvycycle. Default is c(200, 100).
p	Integer, the dimensionality of the embedding space. Must be at least 3. Default is 4.
k	Integer, the number of curvycycle to generate. Default is 2.
offset	Numeric, the positional shift applied to each curvycycle along its linking axis to ensure interlocking instead of overlap. Default is 0.5.
angle	Numeric, the rotation angle (in degrees) used when placing each subsequent curvycycle into its respective plane. Default is 90.

Value

A data frame (or tibble, depending on gen_multicluster()) containing the generated points and cluster assignments.

Examples

```
# Generate two chain-linked curvycycle in 4-D
twochain_curvycycle <- make_chain_curvycycle()
```

---

make_curvygau *Generate Curvy Quadratic and Gaussian Clusters*

Description

This function generates synthetic high-dimensional data containing two clusters: one quadratic-shaped cluster and one Gaussian-shaped cluster. The clusters are positioned apart in feature space with different scaling factors.

Usage

```
make_curvygau(n = c(200, 100), p = 4)
```

Arguments

n	A numeric vector of length 2, specifying the number of observations in each cluster. All values must be positive.
p	Integer. Number of dimensions. Must be at least 3.

Value

A tibble containing $n[1] + n[2]$ rows and p columns, with generated features (`x1`, `x2`, `...`, `xp`) and a `cluster` label.

Examples

```
# Generate 2 clusters in 4D: one quadratic, one Gaussian
curvygau <- make_curvygau()
```

`make_gaucircles` *Generate Concentric Circles with a Gaussian Cluster in High Dimensions*

Description

This function generates a dataset consisting of multiple circular clusters together with a single Gaussian cluster in a p-dimensional space. The circles are placed concentrically at the origin with varying scales, while the Gaussian cluster serves as an additional background or center cluster.

Usage

```
make_gaucircles(
  n = c(200, 100, 100),
  p = 4,
  num_circles = 2,
  scale_circles = c(1, 2)
)
```

Arguments

`n`	An integer vector of length `num_circles + 1`, giving the number of points in each cluster (circles first, followed by the Gaussian). Default is `c(200, 100, 100)`.
`p`	Integer, the dimensionality of the embedding space. Must be at least 3. Default is 4.
`num_circles`	Integer, the number of circular clusters to generate. Default is 2.
`scale_circles`	Numeric vector of length `num_circles`, giving the scale (radius) of each circular cluster. Default is `c(1, 2)`.

Value

A data frame (or tibble, depending on `gen_multicluster()`) containing the generated dataset with cluster assignments.

Examples

```
# Two circles (radii 1 and 2) plus one Gaussian cluster in 4-D
gaucircles <- make_gaucircles()
```

`make_gaucurvycycle` *Generate Concentric Curvycycles with a Gaussian Cluster in High Dimensions*

Description

This function generates a dataset consisting of multiple circular clusters together with a single Gaussian cluster in a p-dimensional space. The curvycycle are placed concentrically at the origin with varying scales, while the Gaussian cluster serves as an additional background or center cluster.

Usage

```
make_gaucurvycycle(
  n = c(200, 100, 100),
  p = 4,
  num_curvycycle = 2,
  scale_curvycycle = c(1, 2)
)
```

Arguments

`n`	An integer vector of length `num_curvycycle + 1`, giving the number of points in each cluster (curvycycle first, followed by the Gaussian). Default is `c(200, 100, 100)`.
`p`	Integer, the dimensionality of the embedding space. Must be at least 3. Default is 4.
`num_curvycycle`	Integer, the number of circular clusters to generate. Default is 2.
`scale_curvycycle`	Numeric vector of length `num_curvycycle`, giving the scale (radius) of each circular cluster. Default is `c(1, 2)`.

Value

A data frame (or tibble, depending on `gen_multicluster()`) containing the generated dataset with cluster assignments.

Examples

```
# Two curvycycle (radii 1 and 2) plus one Gaussian cluster in 4-D
gaucurvycycle <- make_gaucurvycycle()
```

make_klink_circles *Generate Multiple Interlocked Circles in High-Dimensional Space*

Description

This function generates k interlocked circular clusters in a p-dimensional space. The circles are constructed using `gen_multicluster()`, with each circle positioned in a different coordinate plane and slightly offset so that they interlock with a central circle (hub-like structure).

Usage

```
make_klink_circles(n = c(200, 100), p = 4, k = 2, offset = 0.5)
```

Arguments

`n`	An integer vector of length k giving the number of points in each circle. Default is `c(200, 100)`.
`p`	Integer, the dimensionality of the embedding space. Must be at least 3. Default is `4`.
`k`	Integer, the number of circles to generate. Default is `2`.
`offset`	Numeric, the amount of positional shift applied to each circle along the second coordinate axis to prevent complete overlap. Default is `0.5`.

Value

A data frame (or tibble, depending on `gen_multicluster()`) containing the generated points and cluster assignments.

Examples

```
# Generate two interlocked circles in 4-D
twolink_circles <- make_klink_circles()
```

make_klink_curvycycle *Generate Multiple Interlocked curvycycle in High-Dimensional Space*

Description

This function generates k interlocked circular clusters in a p-dimensional space. The curvycycle are constructed using `gen_multicluster()`, with each curvycycle positioned in a different coordinate plane and slightly offset so that they interlock with a central curvycycle (hub-like structure).

Usage

```
make_klink_curvycycle(n = c(200, 100), p = 4, k = 2, offset = 0.5)
```

Arguments

n	An integer vector of length k giving the number of points in each curvycycle. Default is `c(200, 100)`.
p	Integer, the dimensionality of the embedding space. Must be at least 3. Default is 4.
k	Integer, the number of curvycycle to generate. Default is 2.
offset	Numeric, the amount of positional shift applied to each curvycycle along the second coordinate axis to prevent complete overlap. Default is `0.5`.

Value

A data frame (or tibble, depending on `gen_multicluster()`) containing the generated points and cluster assignments.

Examples

```
# Generate two interlocked curvycycle in 4-D
twolink_curvycycle <- make_klink_curvycycle()
```

---

make_mobiusgau *Generate Gaussian cluster with the Mobius Cluster*

Description

This function generates a dataset consisting of a mobius cluster and Gaussian cluster.

Usage

```
make_mobiusgau(n = c(200, 100), p = 4)
```

Arguments

n	A numeric vector (default: c(200, 100)) representing the sample sizes.
p	A numeric value (default: 4) representing the number of dimensions.

Value

A data containing the mobius cluster and Gaussian cluster.

Examples

```
mobgau <- make_mobiusgau(n = c(200, 100), p = 4)
```

make_multigau *Generate Multiple Gaussian Clusters*

Description

This function generates a dataset consisting of multiple Gaussian clusters.

Usage

```
make_multigau(n = c(300, 200, 500), p = 4, k = 3, loc = NULL, scale = NULL)
```

Arguments

n	A numeric vector (default: c(300, 200, 500)) representing the sample sizes.
p	A numeric value (default: 4) representing the number of dimensions.
k	A numeric value (default: 5) representing the number of clusters.
loc	A numeric matrix (default: NULL) representing the locations/centroids of clusters.
scale	A numeric vector (default: NULL) representing the scaling factors of clusters.

Value

A data containing the Gaussian clusters.

Examples

```
loc_matrix <- matrix(c(0, 0, 0, 0,
5, 9, 0, 0,
3, 4, 10, 7
), nrow = 3, byrow = TRUE)
multigau <- make_multigau(n = c(300, 200, 500),
p = 4, k = 3,
loc = loc_matrix, scale = c(0.2, 1.5, 0.5))
```

make_onegrid *Generate a Single Grid Cluster in High Dimensions*

Description

This function generates a dataset consisting of one grid-like cluster (a structured cube grid) in 2D, with optional Gaussian noise dimensions added to extend the dataset into higher dimensions.

Usage

```
make_onegrid(n = 500, p = 4)
```

Arguments

n	Integer, the number of points in the grid cluster. Must be positive. Default is 500.
p	Integer, the dimensionality of the embedding space. Must be at least 2. Default is 4.

Value

A tibble containing the generated dataset with columns:

- x1, x2, ..., xp — coordinates of the data points.
- cluster — cluster assignment (always 1 for the grid).

Examples

```
# Default: 500 points, 4D space (grid in 2D + 2 noise dimensions)
onegrid <- make_onegrid()
```

---

make_shape_para *Generate Parallel Multi-Shape Clusters*

Description

This function generates synthetic high-dimensional data consisting of k clusters of a specified shape (e.g., crescents), arranged in parallel along alternating dimensions. The first cluster is shifted along the first dimension, the second along the third dimension, the third along the first dimension again, and so on.

Usage

```
make_shape_para(n = c(500, 300), k = 2, p = 4, shift = 0.4, shape = "crescent")
```

Arguments

n	A numeric vector of length k, specifying the number of observations in each cluster. All values must be positive.
k	Integer. Number of clusters to generate. Must be greater than 1.
p	Integer. Number of dimensions. Must be at least 3.
shift	Numeric. The distance between cluster centers along the alternating dimensions (default is ‘0.4‘).
shape	Character string. Shape of the clusters to generate (e.g., ‘"crescent"‘, ‘"grid-cube"‘, etc.). Must be a single value.

Value

A tibble containing $\sum n$ rows and p columns, with the generated features ('x1, x2, ..., xp') and a 'cluster' label.

Examples

```
# Generate 2 crescent-shaped clusters in 4D
twocrescent <- make_shape_para(n = c(500, 300), k = 2, p = 4, shape = "crescent")
```

---

make_three_clust_01 *Generate a Three-Cluster Dataset in High Dimensions*

Description

This function generates a dataset consisting of three distinct clusters in a p-dimensional space. Each cluster is generated with a specified shape and location using the underlying 'gen_multicluster()' function.

Usage

```
make_three_clust_01(n = c(700, 300, 500), p = 4)
```

Arguments

n	An integer vector of length 3 specifying the number of points in each cluster. Default is c(700, 300, 500).
p	Integer, the dimensionality of the space. Default is 4.

Value

A data frame (or tibble, depending on 'gen_multicluster()') containing the generated dataset with cluster assignments.

Examples

```
# Generate default three clusters in 4-D
three_clust_01 <- make_three_clust_01()
```

make_three_clust_02 *Generate a Three-Cluster Dataset in High Dimensions*

Description

This function generates a dataset consisting of three distinct clusters in a p-dimensional space. Each cluster is generated with a specified shape and location using the underlying 'gen_multicluster()' function.

Usage

```
make_three_clust_02(n = c(700, 300, 500), p = 4)
```

Arguments

n	An integer vector of length 3 specifying the number of points in each cluster. Default is `c(700, 300, 500)`.
p	Integer, the dimensionality of the space. Default is `4`.

Value

A data frame (or tibble, depending on 'gen_multicluster()') containing the generated dataset with cluster assignments.

Examples

```
# Generate default three clusters in 4-D
three_clust_02 <- make_three_clust_02()
```

make_three_clust_03 *Generate a Three-Cluster Dataset in High Dimensions*

Description

This function generates a dataset consisting of three distinct clusters in a p-dimensional space. Each cluster is generated with a specified shape and location using the underlying 'gen_multicluster()' function.

Usage

```
make_three_clust_03(n = c(700, 300, 500), p = 4)
```

Arguments

n	An integer vector of length 3 specifying the number of points in each cluster. Default is c(700, 300, 500).
p	Integer, the dimensionality of the space. Default is 4.

Value

A data frame (or tibble, depending on 'gen_multicluster()') containing the generated dataset with cluster assignments.

Examples

```
# Generate default three clusters in 4-D
three_clust_03 <- make_three_clust_03()
```

---

make_three_clust_04 *Generate a Three-Cluster Dataset in High Dimensions*

Description

This function generates a dataset consisting of three distinct clusters in a p-dimensional space. Each cluster is generated with a specified shape and location using the underlying 'gen_multicluster()' function.

Usage

```
make_three_clust_04(n = c(700, 300, 500), p = 4)
```

Arguments

n	An integer vector of length 3 specifying the number of points in each cluster. Default is c(700, 300, 500).
p	Integer, the dimensionality of the space. Default is 4.

Value

A data frame (or tibble, depending on 'gen_multicluster()') containing the generated dataset with cluster assignments.

Examples

```
# Generate default three clusters in 4-D
three_clust_04 <- make_three_clust_04()
```

make_three_clust_05 *Generate a Three-Cluster Dataset in High Dimensions*

Description

This function generates a dataset consisting of three distinct clusters in a p-dimensional space. Each cluster is generated with a specified shape and location using the underlying 'gen_multicluster()' function.

Usage

```
make_three_clust_05(n = c(700, 300, 500), p = 4)
```

Arguments

n	An integer vector of length 3 specifying the number of points in each cluster. Default is `c(700, 300, 500)`.
p	Integer, the dimensionality of the space. Default is `4`.

Value

A data frame (or tibble, depending on 'gen_multicluster()') containing the generated dataset with cluster assignments.

Examples

```
# Generate default three clusters in 4-D
three_clust_05 <- make_three_clust_05()
```

make_three_clust_06 *Generate a Three-Cluster Dataset in High Dimensions*

Description

This function generates a dataset consisting of three distinct clusters in a p-dimensional space. Each cluster is generated with a specified shape and location using the underlying 'gen_multicluster()' function.

Usage

```
make_three_clust_06(n = c(700, 300, 500), p = 4)
```

Arguments

n	An integer vector of length 3 specifying the number of points in each cluster. Default is c(700, 300, 500).
p	Integer, the dimensionality of the space. Default is 4.

Value

A data frame (or tibble, depending on 'gen_multicluster()') containing the generated dataset with cluster assignments.

Examples

```
# Generate default three clusters in 4-D
three_clust_06 <- make_three_clust_06()
```

---

make_three_clust_07 *Generate a Three-Cluster Dataset in High Dimensions*

Description

This function generates a dataset consisting of three distinct clusters in a p-dimensional space. Each cluster is generated with a specified shape and location using the underlying 'gen_multicluster()' function.

Usage

```
make_three_clust_07(n = c(700, 300, 500), p = 4)
```

Arguments

n	An integer vector of length 3 specifying the number of points in each cluster. Default is c(700, 300, 500).
p	Integer, the dimensionality of the space. Default is 4.

Value

A data frame (or tibble, depending on 'gen_multicluster()') containing the generated dataset with cluster assignments.

Examples

```
# Generate default three clusters in 4-D
three_clust_07 <- make_three_clust_07()
```

make_three_clust_08 *Generate a Three-Cluster Dataset in High Dimensions*

Description

This function generates a dataset consisting of three distinct clusters in a p-dimensional space. Each cluster is generated with a specified shape and location using the underlying 'gen_multicluster()' function.

Usage

```
make_three_clust_08(n = c(700, 300, 500), p = 4)
```

Arguments

n	An integer vector of length 3 specifying the number of points in each cluster. Default is `c(700, 300, 500)`.
p	Integer, the dimensionality of the space. Default is `4`.

Value

A data frame (or tibble, depending on 'gen_multicluster()') containing the generated dataset with cluster assignments.

Examples

```
# Generate default three clusters in 4-D
three_clust_08 <- make_three_clust_08()
```

make_three_clust_09 *Generate a Three-Cluster Dataset in High Dimensions*

Description

This function generates a dataset consisting of three distinct clusters in a p-dimensional space. Each cluster is generated with a specified shape and location using the underlying 'gen_multicluster()' function.

Usage

```
make_three_clust_09(n = c(700, 300, 500), p = 4)
```

Arguments

n	An integer vector of length 3 specifying the number of points in each cluster. Default is c(700, 300, 500).
p	Integer, the dimensionality of the space. Default is 4.

Value

A data frame (or tibble, depending on 'gen_multicluster()') containing the generated dataset with cluster assignments.

Examples

```
# Generate default three clusters in 4-D
three_clust_09 <- make_three_clust_09()
```

---

make_three_clust_10 *Generate a Three-Cluster Dataset in High Dimensions*

Description

This function generates a dataset consisting of three distinct clusters in a p-dimensional space. Each cluster is generated with a specified shape and location using the underlying 'gen_multicluster()' function.

Usage

```
make_three_clust_10(n = c(700, 300, 500), p = 4)
```

Arguments

n	An integer vector of length 3 specifying the number of points in each cluster. Default is c(700, 300, 500).
p	Integer, the dimensionality of the space. Default is 4.

Value

A data frame (or tibble, depending on 'gen_multicluster()') containing the generated dataset with cluster assignments.

Examples

```
# Generate default three clusters in 4-D
three_clust_10 <- make_three_clust_10()
```

make_three_clust_11 *Generate a Three-Cluster Dataset in High Dimensions*

Description

This function generates a dataset consisting of three distinct clusters in a p-dimensional space. Each cluster is generated with a specified shape and location using the underlying 'gen_multicluster()' function.

Usage

```
make_three_clust_11(n = c(700, 300, 500), p = 4)
```

Arguments

n	An integer vector of length 3 specifying the number of points in each cluster. Default is `c(700, 300, 500)`.
p	Integer, the dimensionality of the space. Default is `4`.

Value

A data frame (or tibble, depending on 'gen_multicluster()') containing the generated dataset with cluster assignments.

Examples

```
# Generate default three clusters in 4-D
three_clust_11 <- make_three_clust_11()
```

make_three_clust_12 *Generate a Three-Cluster Dataset in High Dimensions*

Description

This function generates a dataset consisting of three distinct clusters in a p-dimensional space. Each cluster is generated with a specified shape and location using the underlying 'gen_multicluster()' function.

Usage

```
make_three_clust_12(n = c(700, 300, 500), p = 4)
```

Arguments

n	An integer vector of length 3 specifying the number of points in each cluster. Default is c(700, 300, 500).
p	Integer, the dimensionality of the space. Default is 4.

Value

A data frame (or tibble, depending on 'gen_multicluster()') containing the generated dataset with cluster assignments.

Examples

```
# Generate default three clusters in 4-D
three_clust_12 <- make_three_clust_12()
```

---

make_three_clust_13 *Generate a Three-Cluster Dataset in High Dimensions*

Description

This function generates a dataset consisting of three distinct clusters in a p-dimensional space. Each cluster is generated with a specified shape and location using the underlying 'gen_multicluster()' function.

Usage

```
make_three_clust_13(n = c(700, 300, 500), p = 4)
```

Arguments

n	An integer vector of length 3 specifying the number of points in each cluster. Default is c(700, 300, 500).
p	Integer, the dimensionality of the space. Default is 4.

Value

A data frame (or tibble, depending on 'gen_multicluster()') containing the generated dataset with cluster assignments.

Examples

```
# Generate default three clusters in 4-D
three_clust_13 <- make_three_clust_13()
```

make_three_clust_14 *Generate a Three-Cluster Dataset in High Dimensions*

Description

This function generates a dataset consisting of three distinct clusters in a p-dimensional space. Each cluster is generated with a specified shape and location using the underlying 'gen_multicluster()' function.

Usage

```
make_three_clust_14(n = c(700, 300, 500), p = 4)
```

Arguments

n	An integer vector of length 3 specifying the number of points in each cluster. Default is `c(700, 300, 500)`.
p	Integer, the dimensionality of the space. Default is `4`.

Value

A data frame (or tibble, depending on 'gen_multicluster()') containing the generated dataset with cluster assignments.

Examples

```
# Generate default three clusters in 4-D
three_clust_14 <- make_three_clust_14()
```

make_three_clust_15 *Generate a Three-Cluster Dataset in High Dimensions*

Description

This function generates a dataset consisting of three distinct clusters in a p-dimensional space. Each cluster is generated with a specified shape and location using the underlying 'gen_multicluster()' function.

Usage

```
make_three_clust_15(n = c(700, 300, 500), p = 4)
```

Arguments

n	An integer vector of length 3 specifying the number of points in each cluster. Default is c(700, 300, 500).
p	Integer, the dimensionality of the space. Default is 4.

Value

A data frame (or tibble, depending on 'gen_multicluster()') containing the generated dataset with cluster assignments.

Examples

```
# Generate default three clusters in 4-D
three_clust_15 <- make_three_clust_15()
```

---

make_three_clust_16 *Generate a Three-Cluster Dataset in High Dimensions*

Description

This function generates a dataset consisting of three distinct clusters in a p-dimensional space. Each cluster is generated with a specified shape and location using the underlying 'gen_multicluster()' function.

Usage

```
make_three_clust_16(n = c(700, 300, 500), p = 4)
```

Arguments

n	An integer vector of length 3 specifying the number of points in each cluster. Default is c(700, 300, 500).
p	Integer, the dimensionality of the space. Default is 4.

Value

A data frame (or tibble, depending on 'gen_multicluster()') containing the generated dataset with cluster assignments.

Examples

```
# Generate default three clusters in 4-D
three_clust_16 <- make_three_clust_16()
```

make_three_clust_17 *Generate a Three-Cluster Dataset in High Dimensions*

Description

This function generates a dataset consisting of three distinct clusters in a p-dimensional space. Each cluster is generated with a specified shape and location using the underlying 'gen_multicluster()' function.

Usage

```
make_three_clust_17(n = c(700, 300, 500), p = 4)
```

Arguments

n	An integer vector of length 3 specifying the number of points in each cluster. Default is `c(700, 300, 500)`.
p	Integer, the dimensionality of the space. Default is `4`.

Value

A data frame (or tibble, depending on 'gen_multicluster()') containing the generated dataset with cluster assignments.

Examples

```
# Generate default three clusters in 4-D
three_clust_17 <- make_three_clust_17()
```

make_three_clust_18 *Generate a Three-Cluster Dataset in High Dimensions*

Description

This function generates a dataset consisting of three distinct clusters in a p-dimensional space. Each cluster is generated with a specified shape and location using the underlying 'gen_multicluster()' function.

Usage

```
make_three_clust_18(n = c(700, 300, 500), p = 4)
```

Arguments

n	An integer vector of length 3 specifying the number of points in each cluster. Default is c(700, 300, 500).
p	Integer, the dimensionality of the space. Default is 4.

Value

A data frame (or tibble, depending on 'gen_multicluster()') containing the generated dataset with cluster assignments.

Examples

```
# Generate default three clusters in 4-D
three_clust_18 <- make_three_clust_18()
```

---

make_three_clust_19 *Generate a Three-Cluster Dataset in High Dimensions*

Description

This function generates a dataset consisting of three distinct clusters in a p-dimensional space. Each cluster is generated with a specified shape and location using the underlying 'gen_multicluster()' function.

Usage

```
make_three_clust_19(n = c(700, 300, 500), p = 4)
```

Arguments

n	An integer vector of length 3 specifying the number of points in each cluster. Default is c(700, 300, 500).
p	Integer, the dimensionality of the space. Default is 4.

Value

A data frame (or tibble, depending on 'gen_multicluster()') containing the generated dataset with cluster assignments.

Examples

```
# Generate default three clusters in 4-D
three_clust_19 <- make_three_clust_19()
```

make_three_clust_20 *Generate a Three-Cluster Dataset in High Dimensions*

Description

This function generates a dataset consisting of three distinct clusters in a p-dimensional space. Each cluster is generated with a specified shape and location using the underlying 'gen_multicluster()' function.

Usage

```
make_three_clust_20(n = c(700, 300, 500), p = 4)
```

Arguments

n	An integer vector of length 3 specifying the number of points in each cluster. Default is `c(700, 300, 500)`.
p	Integer, the dimensionality of the space. Default is `4`.

Value

A data frame (or tibble, depending on 'gen_multicluster()') containing the generated dataset with cluster assignments.

Examples

```
# Generate default three clusters in 4-D
three_clust_20 <- make_three_clust_20()
```

make_twogrid_overlap *Generate Two Overlapping Grid Clusters in High Dimensions*

Description

This function generates a dataset consisting of two overlapping grid-like clusters in a 2D space, with optional noise dimensions added to reach higher-dimensional spaces. The overlap is controlled by scaling factors for the grids.

Usage

```
make_twogrid_overlap(n = c(500, 500), p = 4)
```

Arguments

n	A numeric vector of length 2 specifying the number of points in each grid cluster.
p	An integer specifying the total number of dimensions. Must be greater than or equal to 2. If p > 2, additional noise dimensions are appended.

Value

A tibble with n[1] + n[2] rows and p + 1 columns:

- x1, ..., xp — coordinates of the generated points.
- cluster — cluster membership label.

Examples

```
# Generate two overlapping grid clusters in 4-D
df <- make_twogrid_overlap()
```

---

make_twogrid_shift *Generate Two Shifted Grid Clusters in High Dimensions*

Description

This function generates two grid-shaped clusters in a 2D space, where one grid is shifted relative to the other. Optionally, additional noise dimensions can be added to embed the structure in a higher-dimensional space.

Usage

```
make_twogrid_shift(n = c(500, 500), p = 4)
```

Arguments

n	A numeric vector of length 2 specifying the number of points in each cluster. Default is c(500, 500).
p	An integer specifying the total number of dimensions for the output dataset. Must be at least 2. If p > 2, additional noise dimensions will be added. Default is 4.

Value

A tibble with n[1] + n[2] rows and p + 1 columns:

- x1, x2, ..., xp: Numeric coordinates of the points.
- cluster: Cluster membership label (factor with 2 levels).

Examples

```
# Generate 2 shifted grid clusters in 4-D
make_twogrid_shift <- make_twogrid_shift()
```

---

mobiusgau *Mobius clust dataset with a noise dimension*

Description

The 'mobiusgau' dataset contains a 3-dimensional Mobius and Gaussian cluster with added noise dimension. Each data point is represented by five dimensions (x1 to x4).

Usage

```
data(mobiusgau)
```

Format

A data frame with 1000 rows and 4 columns:

x1, x2, x3, x4 High-dimensional coordinates

Source

This dataset is generated for illustrative purposes.

Examples

```
# Load the mobiusgau dataset
data(mobiusgau)

# Display the first few rows of the dataset
head(mobiusgau)
```

mobiusgau_tsne1 *tSNE embedding for mobiusgau dataset which with noise dimensions tSNE parameters set to perplexity: 15.*

Description

The 'mobiusgau_tsne1' dataset contains the tSNE (t-distributed Stochastic Neighbor Embedding) embeddings of a five-dimensional mobiusgau. Each data point is represented by two tSNE coordinates (emb1 and emb2).

Usage

```
data(mobiusgau_tsne1)
```

Format

'mobiusgau_tsne1' A data frame with 1000 rows and 4 columns:

emb1 Numeric, first tSNE 2D embeddings.

emb2 Numeric, second tSNE 2D embeddings.

Source

This dataset is generated for illustrative purposes.

Examples

```
# Load the mobiusgau_tsne1 dataset
data(mobiusgau_tsne1)

# Display the first few rows of the dataset
head(mobiusgau_tsne1)
```

mobiusgau_tsne2 *tSNE embedding for mobiusgau dataset which with noise dimensions tSNE parameters set to perplexity: 30.*

Description

The 'mobiusgau_tsne2' dataset contains the tSNE (t-distributed Stochastic Neighbor Embedding) embeddings of a five-dimensional mobiusgau. Each data point is represented by two tSNE coordinates (emb1 and emb2).

Usage

```
data(mobiusgau_tsne2)
```

Format

‘mobiusgau_tsne2‘ A data frame with 1000 rows and 4 columns:

emb1 Numeric, first tSNE 2D embeddings.

emb2 Numeric, second tSNE 2D embeddings.

Source

This dataset is generated for illustrative purposes.

Examples

```
# Load the mobiusgau_tsne2 dataset
data(mobiusgau_tsne2)

# Display the first few rows of the dataset
head(mobiusgau_tsne2)
```

`mobiusgau_tsne3` *tSNE embedding for mobiusgau dataset which with noise dimensions tSNE parameters set to perplexity: 5.*

Description

The ‘mobiusgau_tsne3‘ dataset contains the tSNE (t-distributed Stochastic Neighbor Embedding) embeddings of a five-dimensional mobiusgau. Each data point is represented by two tSNE coordinates (emb1 and emb2).

Usage

```
data(mobiusgau_tsne3)
```

Format

‘mobiusgau_tsne3‘ A data frame with 1000 rows and 4 columns:

emb1 Numeric, first tSNE 2D embeddings.

emb2 Numeric, second tSNE 2D embeddings.

Source

This dataset is generated for illustrative purposes.

Examples

```
# Load the mobiusgau_tsne1 dataset
data(mobiusgau_tsne3)

# Display the first few rows of the dataset
head(mobiusgau_tsne3)
```

`mobiusgau_umap1` *UMAP embedding for mobiusgau dataset which with noise dimensions UMAP parameters set to n-neigbors: 15 and min-dist: 0.1.*

Description

The ‘mobiusgau_umap1‘ dataset contains the UMAP (Uniform Manifold Approximation and Projection) embeddings of a five-dimensional mobiusgau. Each data point is represented by two UMAP coordinates (emb1 and emb2).

Usage

```
data(mobiusgau_umap1)
```

Format

‘mobiusgau_umap1‘ A data frame with 1000 rows and 4 columns:

emb1 Numeric, first UMAP 2D embeddings.

emb2 Numeric, second UMAP 2D embeddings.

Source

This dataset is generated for illustrative purposes.

Examples

```
# Load the mobiusgau_umap1 dataset
data(mobiusgau_umap1)

# Display the first few rows of the dataset
head(mobiusgau_umap1)
```

mobiusgau_umap2 *UMAP embedding for mobiusgau dataset which with noise dimensions UMAP parameters set to n-neigbors: 30 and min-dist: 0.08.*

Description

The 'mobiusgau_umap2' dataset contains the UMAP (Uniform Manifold Approximation and Projection) embeddings of a five-dimensional mobiusgau. Each data point is represented by two UMAP coordinates (emb1 and emb2).

Usage

```
data(mobiusgau_umap2)
```

Format

'mobiusgau_umap2' A data frame with 1000 rows and 4 columns:

emb1 Numeric, first UMAP 2D embeddings.

emb2 Numeric, second UMAP 2D embeddings.

Source

This dataset is generated for illustrative purposes.

Examples

```
# Load the mobiusgau_umap2 dataset
data(mobiusgau_umap2)

# Display the first few rows of the dataset
head(mobiusgau_umap2)
```

mobiusgau_umap3 *UMAP embedding for mobiusgau dataset which with noise dimensions UMAP parameters set to n-neigbors: 5 and min-dist: 0.9.*

Description

The 'mobiusgau_umap3' dataset contains the UMAP (Uniform Manifold Approximation and Projection) embeddings of a five-dimensional mobiusgau. Each data point is represented by two UMAP coordinates (emb1 and emb2).

Usage

```
data(mobiusgau_umap3)
```

Format

'mobiusgau_umap3' A data frame with 1000 rows and 4 columns:

emb1 Numeric, first UMAP 2D embeddings.

emb2 Numeric, second UMAP 2D embeddings.

Source

This dataset is generated for illustrative purposes.

Examples

```
# Load the mobiusgau_umap3 dataset
data(mobiusgau_umap3)

# Display the first few rows of the dataset
head(mobiusgau_umap3)
```

---

normalize_data *Generate Normalized data*

Description

This function normalize the data by absolute value

Usage

```
normalize_data(data)
```

Arguments

data	A tibble representing the data which needed to be normalized.

Value

A normalized data.

Examples

```
set.seed(20240412)
data1 <- gen_gaussian(n= 500, p = 4)
normalize_data(data = data1)
```

randomize_rows *Randomize Rows of a Data Frame*

Description

This function randomly shuffles the rows of a given data frame.

Usage

```
randomize_rows(data)
```

Arguments

data	A data frame to be randomized.

Value

A data frame with rows randomly shuffled.

Examples

```
randomize_rows(mobiusgau)
```

relocate_clusters *Relocate Clusters in High-Dimensional Space*

Description

This function relocates clusters in a dataset by centering each cluster and shifting it based on a given transformation matrix.

Usage

```
relocate_clusters(data, vert_mat)
```

Arguments

data	A tibble or data frame containing clustered data. It must have a ‘cluster‘ column indicating cluster membership.
vert_mat	A matrix specifying the translation vectors for each cluster. The number of rows must match the number of clusters.

Value

A tibble containing the relocated clusters with randomized row order.

Examples

```
set.seed(20240412)
df <- tibble::tibble(
x1 = rnorm(12),
x2 = rnorm(12),
x3 = rnorm(12),
x4 = rnorm(12),
cluster = rep(1:3, each = 4)
)

# Create a 3x4 matrix to define new cluster centers
vert_mat <- matrix(c(
  5, 0, 0, 0,   # Shift cluster 1
  0, 5, 0, 0,   # Shift cluster 2
  0, 0, 5, 0    # Shift cluster 3
), nrow = 3, byrow = TRUE)
# Apply relocation
relocated_df <- relocate_clusters(df, vert_mat)
```

Index