Package ‘SyScSelection’

October 26, 2020

Type Package
Title Systematic Scenario Selection for Stress Testing
Version 1.0.2
Author Merlin Kopfmann
Maintainer Merlin Kopfmann <mghncd+cran@posteo.jp>
Description Quasi-Monte-Carlo algorithm for systematic generation of shock scenarios from an arbitrary multivariate elliptical distribution. The algorithm selects a systematic mesh of arbitrary fineness that approximately evenly covers an isoprobability ellipsoid in d dimensions (Flood, Mark D. & Korenko, George G. (2013) <doi:10.1080/14697688.2014.926018>). This package is the 'R' analogy to the 'Matlab' code published by Flood & Korenko in above-mentioned paper.
License CC0
Encoding UTF-8
LazyData true
RoxygenNote 7.1.0
Imports pracma, stats
Suggests knitr, rmarkdown
VignetteBuilder knitr
NeedsCompilation no
Repository CRAN
Date/Publication 2020-10-26 14:10:02 UTC

R topics documented:

baseb_expansion .................................................. 2
calc_mesh_size .................................................... 3
center_at_origin .................................................. 3
fill_adj_2Dface ................................................... 4
fill_adj_2Dface_beta ............................................. 4
fill_corners ..................................................... 5
get ................................................................. 5
baseb_expansion

Description

Adds the next base-b element to an existing base-b sequence

Usage

baseb_expansion(ain, b)

Arguments

ain
Either an array containing an existing base-b expansion, or a scalar integer indicating the length for a new base-b expansion

b
Base for integer expansions used in the sequence

Value

An expanded base-b expansion
calc_mesh_size

Calculates the number of points in a mesh of fineness phi, covering a hypercube in d dimensions

Description
Calculates the number of points in a mesh of fineness phi, covering a hypercube in d dimensions.

Usage

calc_mesh_size(phi, d)

Arguments

phi
The scalar fineness of the mesh

d
The number of dimensions for the unit spheroid

Value

A list of: corner_pts - Count of points extreme (+/- 1) in all dim, edge_pts - Count of points extreme in all but one dimen, face_pts - Count of points extreme in all but two dimen, total_pts - Sum of: corner_pts + edge_pts + face_pts

center_at_origin

Creates a new ellipsoid object equivalent to the given hyperellipsoid (hellipse), but centered at the origin.

Description

Creates a new ellipsoid object equivalent to the given hyperellipsoid (hellipse), but centered at the origin.

Usage

center_at_origin(hellip)

Arguments

hellip
The original object, to be shifted

Value

List of two: hellip2 - the re-centered hyperellipsoid and mu - the amount of the translation
fill_adj_2Dface  Creates a phi x phi grid (i.e., the mesh on a single two-dimensional face of a larger hypercube) of d-dimensional points, where the regularity of the grid has been adjusted to avoid clustering in the corners.

Description

Creates a phi x phi grid (i.e., the mesh on a single two-dimensional face of a larger hypercube) of d-dimensional points, where the regularity of the grid has been adjusted to avoid clustering in the corners.

Usage

fill_adj_2Dface(d, phi)

Arguments

d  The number of dimensions for the unit spheroid
phi  Fineness of the mesh along each dimension of the 2D face

Value

A phi x phi x d array of points. The points (each facemesh2D(i,j,:)) are identically equal to one in the first d-2 dimensions, so that the mesh varies only in the final two dimensions.

fill_adj_2Dface_beta  Calculates the factor, beta in [0, 1], that interpolates the pth equidistant point between the two endpoints, z_one and z_phi, for an adjusted 2D mesh of fineness phi in d dimensions.

Description

Calculates the factor, beta in [0, 1], that interpolates the pth equidistant point between the two endpoints, z_one and z_phi, for an adjusted 2D mesh of fineness phi in d dimensions.

Usage

fill_adj_2Dface_beta(p, phi, z_one, z_phi)

Arguments

p  ...
phi  Fineness of the mesh along each dimension of the 2D face
z_one  ...
z_phi  ...
**Value**

beta

---

**fill_corners**

Systematically fills a given mesh array (cmesh) with d-dimensional points representing every corner of a d-dimensional hypercube. The function fills the successive dimensions of each point via depth-first recursion across all d dimensions.

**Description**

Systematically fills a given mesh array (cmesh) with d-dimensional points representing every corner of a d-dimensional hypercube. The function fills the successive dimensions of each point via depth-first recursion across all d dimensions.

**Usage**

fill_corners(cmesh, shock, shk_curs, dim_curs)

**Arguments**

- **cmesh**: The mesh to be filled with corner points
- **shock**: The current shock vector being filled
- **shk_curs**: Index in cmesh of the shock currently being filled
- **dim_curs**: Index in the current shock of the dimension being filled

**Value**

A list of: cmesh - d x 2^d array of corner points being filled, shk_curs - last point in cmesh that was filled

---

**get**

Get hyperellipsoid property from the specified object and return the value. Property names are: center, shape, and size

**Description**

Get hyperellipsoid property from the specified object and return the value. Property names are: center, shape, and size

**Usage**

get(hellip, propName)
Arguments

- **hellip**: A valid hyperellipsoid object
- **propName**: A string of the desired property

Value

- The value of the indicated property

---

**hypercube_mesh**

Generates a Cartesian mesh of $d$-dimensional scenarios based on the given ellipsoid. This function does not assume that the ellipsoid is centered at the origin.

Description

Generates a Cartesian mesh of $d$-dimensional scenarios based on the given ellipsoid. This function does not assume that the ellipsoid is centered at the origin.

Usage

```
hypercube_mesh(phi, hellip, normalize)
```

Arguments

- **phi**: The scalar fineness of the mesh
- **hellip**: The basis for the shocks; it must have measurable width in every dimension
- **normalize**: Whether to normalize points from the cube onto the sphere or not (TRUE/FALSE)

Value

- A $d \times N$ array, with each column a scenario

Examples

```
hellip <- hyperellipsoid()
hypercube_mesh(3, hellip, TRUE)
```
hyperellipsoid

Hyperellipsoid class constructor

Description

Hyperellipsoid class constructor

Usage

hyperellipsoid(...)

Arguments

...  
mu - The vector for the center point, sig - The matrix determining the shape; for elliptical probability distributions, sig will be the inverse dispersion matrix, c - The scalar determining the size

Value

A new hyperellipsoid object

Examples

hyperellipsoid()

make_corners

Fills a mesh (corn_mesh) with d-dimensional points representing all corners of a d-dimensional cube encompassing a d-dimensional unit spheroid.

Description

Fills a mesh (corn_mesh) with d-dimensional points representing all corners of a d-dimensional cube encompassing a d-dimensional unit spheroid.

Usage

make_corners(d, normalize)

Arguments

d  
The number of dimensions for the unit spheroid

normalize  
Whether to scale the corner points onto the sphere or not

Value

A d x 2^d array of corner points
**make_edges**

Fills a mesh with d-dimensional points representing all non-corner edge points of a d-dimensional cube encompassing a d-dimensional unit spheroid.

**Description**

Fills a mesh with d-dimensional points representing all non-corner edge points of a d-dimensional cube encompassing a d-dimensional unit spheroid.

**Usage**

make_edges(d, phi, normalize)

**Arguments**

- **d**
  - The number of dimensions for the unit spheroid
- **phi**
  - Fineness of the mesh along the edge (i.e., the total number of points, "including" the corners)
- **normalize**
  - Whether to scale the corner points onto the sphere or not

**Value**

A d x d*2^(d-1)*(phi-2) array of edge points

**make_ellipsoid_from_vertices**

Constructs a new d-dimensional ellipsoid with the given "positive vertices", and size parameter, c. The constructed ellipsoid is centered at the origin. Note that the input vertices (i.e., the columns of V) must therefore be orthogonal vectors, themselves centered at the origin. The size parameter, c, may be needed because the points alone only determine the eigenvalues up to a positive constant. For vertices which fall on the constructed ellipsoid, choose as the size parameter c = 1. The new ellipsoid is centered at the origin.

**Description**

Constructs a new d-dimensional ellipsoid with the given "positive vertices", and size parameter, c. The constructed ellipsoid is centered at the origin. Note that the input vertices (i.e., the columns of V) must therefore be orthogonal vectors, themselves centered at the origin. The size parameter, c, may be needed because the points alone only determine the eigenvalues up to a positive constant. For vertices which fall on the constructed ellipsoid, choose as the size parameter c = 1. The new ellipsoid is centered at the origin.
**Usage**

```r
make_ellipsoid_from_vertices(V, c)
```

**Arguments**

- **V**: A d x d array of positive vertices (in columns)
- **c**: The size parameter of the new ellipsoid

**Value**

A new ellipsoid, centered at the origin, with the given vertices

**Examples**

```r
eellip <- hyperellipsoid()
V <- vertices(eellip)
c <- 4
make_ellipsoid_from_vertices(V, c)
```

---

**Usage**

```r
make_faces(d, phi, normalize)
```

**Arguments**

- **d**: The number of dimensions for the unit spheroid
- **phi**: Fineness of the mesh along each dimension of the 2D face
- **normalize**: Whether to scale the corner points onto the sphere or not

**Value**

A d x d*(d-1)*2^(d-3)*(phi-2)^2 array of face points

**Description**

Fills a mesh with d-dimensional points representing all non-edge face points of a d-dimensional cube encompassing a d-dimensional unit spheroid.
**new_baseb_expansion**  
*Creates a new base-b sequence of a designated length*

---

### Description

Creates a new base-b sequence of a designated length

### Usage

```
new_baseb_expansion(k, b)
```

### Arguments

- **k**: The integer to expand
- **b**: Base for integer expansions used in the sequence

### Value

The expansion of the integer k

---

**rotate_to_coordaxes**  
*Rotates the ellipsoid (hellip) so its principal axes align with the coordinate axes. Both ellipsoids are centered at the origin. Note that there are \(2^d\times d!\) valid ways to rotate the ellipsoid to the axes. This algorithm does not prescribe which solution will be provided.*

---

### Description

Rotates the ellipsoid (hellip) so its principal axes align with the coordinate axes. Both ellipsoids are centered at the origin. Note that there are \(2^d\times d!\) valid ways to rotate the ellipsoid to the axes. This algorithm does not prescribe which solution will be provided.

### Usage

```
rotate_to_coordaxes(hellip)
```

### Arguments

- **hellip**: The shape to be rotated, must be centered at the origin

### Value

A list of: hellip2 - A new hyperellipsoid, rotated to the coordinate axes and tfm - the transformation matrix that creates the rotation
**sizeparam_normal_distn**

*Calculates the size parameter for a d-dimensional hyperellipsoid conforming to a normal (i.e., Gaussian) distribution.*

---

**Description**

Calculates the size parameter for a d-dimensional hyperellipsoid conforming to a normal (i.e., Gaussian) distribution.

**Usage**

```
sizeparam_normal_distn(prob, d)
```

**Arguments**

- **prob**  
  The target probability threshold
- **d**  
  Number of dimensions in the multivariate distribution

**Value**

The appropriate (scalar) size parameter

**Examples**

```
sizeparam_normal_distn(0.95, 6)
```

---

**sizeparam_t_distn**

*Calculates the size parameter for a d-dimensional hyperellipsoid conforming to a Student’s t distribution.*

---

**Description**

Calculates the size parameter for a d-dimensional hyperellipsoid conforming to a Student’s t distribution.

**Usage**

```
sizeparam_t_distn(prob, d, nu)
```

**Arguments**

- **prob**  
  The target probability threshold
- **d**  
  Number of dimensions in the multivariate distribution
- **nu**  
  Degrees of freedom parameter for the t distribution
**Value**

The appropriate (scalar) size parameter

**Examples**

```
sizeparam_t_distn(0.95, 6, 5)
```

---

**spheroid_mesh**  
*Generates a Cartesian mesh of d-dimensional scenarios based on the given ellipsoid. This function does not assume that the ellipsoid is centered at the origin.*

**Description**

Generates a Cartesian mesh of d-dimensional scenarios based on the given ellipsoid. This function does not assume that the ellipsoid is centered at the origin.

**Usage**

```
spheroid_mesh(d, phi, normalize)
```

**Arguments**

- **d**  
The number of dimensions for the unit spheroid
- **phi**  
The scalar fineness of the mesh
- **normalize**  
Whether to normalize points from the cube onto the sphere or not (TRUE/FALSE)

**Value**

A d x N array with each column a scenario

---

**stretch_to_unitspheroid**  
*Stretches the ellipsoid (hellip) to the unit spheroid of the same dimension. Both the input ellipsoid and unit spheroid are centered at the origin.*

**Description**

Stretches the ellipsoid (hellip) to the unit spheroid of the same dimension. Both the input ellipsoid and unit spheroid are centered at the origin.

**Usage**

```
stretch_to_unitspheroid(hellip)
```
**transform_ellipsoid**

Arguments

- **hellip**: The original shape to be stretched

Value

A list of: **hellip** - a new unit spheroid, mapped from the ellipsoid and **tfm** - transformation matrix that creates the stretching

---

**univariate_shocks**

Calculates 2d d-dimensional univariate shocks (up and down in each of the d dimensions) based on the given ellipsoid. Univariate shocks are points on the surface of the ellipsoid that differ from the center of the ellipsoid in only one dimension. Thus, for an ellipsoid centered at the origin, only one element of a d-dimensional shock will be non-zero. This function does not assume that the ellipsoid is centered at the origin.

Description

Calculates 2d d-dimensional univariate shocks (up and down in each of the d dimensions) based on the given ellipsoid. Univariate shocks are points on the surface of the ellipsoid that differ from the center of the ellipsoid in only one dimension. Thus, for an ellipsoid centered at the origin, only one element of a d-dimensional shock will be non-zero. This function does not assume that the ellipsoid is centered at the origin.
Usage

univariate_shocks(hellip)

Arguments

hellip the basis for the shocks; it must have measurable width in every dimension

Value

A d x 2d array, [dx2d], with each column a shock; the first d columns are positive univariate shocks, and final d columns are matching negative univariate shocks

Examples

hellip <- hyperellipsoid()
univariate_shocks(hellip)

Description

Finds the d d-dimensional positive vertices for the given ellipsoid. A "positive" vertex is one where a principal axis for the ellipsoid intersects the surface of the ellipsoid in the direction of the corresponding eigenvector. (Recall that each of the eigenvectors of the ellipsoid’s shape matrix is collinear with one of the principal axes.) This function does not assume that the ellipsoid is centered at the origin. Because the direction of each unit eigenvector is arbitrary (i.e., multiplication by -1 still yields a unit eigenvector), a simple algorithm is used to pick a consistent orientation for the vertex points.

Usage

vertices(hellip)

Arguments

hellip defines the polar vertices
vertices

Value

A d x d array with each column a positive vertex

Examples

hellip <- hyperellipsoid()
vertices(hellip)
Index

baseb_expansion, 2
calc_mesh_size, 3
center_at_origin, 3
fill_adj_2Dface, 4
fill_adj_2Dface_beta, 4
fill_corners, 5
get, 5
hypercube_mesh, 6
hyperellipsoid, 7
make_corners, 7
make_edges, 8
make_ellipsoid_from_vertices, 8
make_faces, 9
new_baseb_expansion, 10
rotate_to_coordaxes, 10
sizeparam_normal_distn, 11
sizeparam_t_distn, 11
spheroid_mesh, 12
stretch_to_unitspheroid, 12
transform_ellipsoid, 13
univariate_shocks, 13
vertices, 14