Package 'volesti'

October 1, 2018

Type Package

License LGPL-3

Title Volume Approximation and Sampling of Convex Polytopes

Description Package provides an R interface for volesti C++ package. Volesti computes approximations of volume of polytopes given as a set of points or linear inequalities or Minkowski sum of segments (zonotopes). There are two algorithms for volume approximation as well as algorithms for sampling, rounding and rotating polytopes.

Version 0.0.0

Date 2018-09-05

BugReports https://github.com/vissarion/volume_approximation/issues

Imports Rcpp (>= 0.12.17)

LinkingTo Rcpp, RcppEigen, BH

Suggests testthat

Encoding UTF-8

RoxygenNote 6.0.1

R topics documented:

---

copula1 *Construct a copula using uniform sampling from the unit simplex*

Description

Given two families of parallel hyperplanes (or a family of parallel hyperplanes and a family of concentric ellispoids centered at the origin) intersecting the canonical simplex, this function samples from the canonical simplex and construct an approximation of the bivariate probability distribution, called copula.

Usage

```
copula1(h1, h2, numSlices, N)
```

Arguments

h1	A d-dimensional vector that describes the direction of the first family of parallel hyperplanes.
h2	A d-dimensional vector that describes the direction of the second family of parallel hyperplanes.
numSlices	The number of the slices for the copula. Default value is 100.
N	The number of points to sample. Default value is $4 \cdot 10^6$.
E	The $d \times d$ symmetric positive define matrix of the family of concentric ellipsoids centered at the origin.

Value

A $numSlices \times numSlices$ copula.

References

L. Cales, A. Chalkis, I.Z. Emiris, V. Fisikopoulos, "Practical volume computation of structured convex bodies, and an application to modeling portfolio dependencies and financial crises," Proc. of Symposium on Computational Geometry, Budapest, Hungary, *2018.*

Examples

```
# compute a copula for two families of parallel hyperplanes
h1 = runif(n = 10, min = 1, max = 1000)
h1 = h1 / 1000
h2=runif(n = 10, min = 1, max = 1000)
h2 = h2 / 1000
cop = copula(h1=h1, h2=h2, numSlices = 10, N = 100000)

# compute a copula for a family of parallel hyperplanes and a family of conentric ellipsoids
h1 = runif(n = 10, min = 1, max = 1000)
```

```
h1 = h1 / 1000
E = replicate(10, rnorm(20))
E = cov(E)
cop = copula(h1=h1, E=E, numSlices=10, N=100000)
```

---

exact_vol *Compute the exact volume of a zonotope*

Description

Given the $m \times d$ matrix that containes the m segments that define the d-dimensional zonotope, this function computes the sum of the absolute values of the determinants of all the $d \times d$ submatrices.

Usage

```
exact_vol(Z, exact)
```

Arguments

ZonoMat	The $m \times d$ matrix that containes the segments that define the zonotope.

Value

The exact volume of the zonotope

Examples

```
# compute the exact volume of a 5-dimensional zonotope defined by the Minkowski sum of 10 segments
ZonoMat = GenZonotope(5, 10)
vol = ExactZonoVol(ZonoMat)
```

---

fileToMatrix *function to get a ine file and returns a numerical matrix A*

Description

This function takes the path for an ine or an ext file and returns the corresponding numerical matrix and vector that are compatible with volesti package's functions.

Usage

```
fileToMatrix(path, zonotope)
```

Arguments

path	A string that containes the path to an ine or a ext file. The ine file desrcibes a H-polytope and ext file describes a V-polytope or a zonotope.

Value

If the path corresponds to an ine file then the return value is a list that containes elements "A" and "b", i.e. the numerical $m \times d$ matrix A and the numerical m-dimensional vector b, defining H-polytope P, s.t.: $Ax \leq b$. If it corresponds to an ext file (V-polytopes or zonotopes) then the return value is a $m \times d$ matrix that containes row-wise the vertices or the segments respectively.

Examples

```
# give the path to birk4.ine
path = system.file('extdata', package = 'volesti')
ListPoly = fileToMatrix(paste0(path,'/birk4.ine'))
```

GenCross *Generator function for cross polytopes*

Description

This function can be used to generate a d-dimensional cross polytope in H or V representation.

Usage

```
GenCross(dimension, repr)
```

Arguments

`dimension`	The dimension of the cross polytope.
`repr`	A string to declare the representation. It has to be 'H' for H-representation or 'V' for V-representation.

Value

A cross polytope in H or V-representation. For an H polytope the return value is a list with two elements: the "matrix" containing a $2^d \times d$ matrix A and the "vector" containing a 2^d -dimensional vector b, s.t. $Ax \leq b$. When the V-representation is chosen the return value is a $2d \times d$ matrix that containes the vertices row-wise.

Examples

```
# generate a 10-dimensional cross polytope in H-representation
PolyList = GenCross(10, 'H')

# generate a 15-dimension cross polytope in V-representation
PolyList = GenCross(15, 'V')
```

GenCube — *Generator function for hypercubes*

Description

This function can be used to generate a d-dimensional Hypercube $[-1, 1]^d$ in H or V representation.

Usage

```
GenCube(dimension, repr)
```

Arguments

`dimension`	The dimension of the hypercube
`repr`	A string to declare the representation. It has to be 'H' for H-representation or 'V' for V-representation.

Value

A hypercube in H or V-representation. For an H polytope the return value is a list with two elements: the "matrix" containing a $2d \times d$ matrix A and the "vector" containing a $2d$-dimensional vector b, s.t. $Ax \leq b$. When the V-representation is chosen the return value is a $2^d \times d$ matrix that containes the vertices row-wise.

Examples

```
# generate a 10-dimensional hypercube in H-representation
PolyList = GenCube(10, 'H')

# generate a 15-dimension hypercube in V-representation
PolyList = GenCube(15, 'V')
```

GenProdSimplex — *Generator function for product of simplices*

Description

This function can be used to generate a $2d$-dimensional polytope that is defined as the product of two d-dimensional unit simplices in H-representation.

Usage

```
GenProdSimplex(dimension)
```

Arguments

`dimension`	The dimension of the simplices.

Value

A polytope defined as the product of two unit simplices in H-representation. The retutn value is a list with two elements: the "matrix" containing a $(2d+1) \times 2d$ matrix A and the "vector" containing a $(2d+1)$-dimensional vector b, s.t. $Ax \leq b$.

Examples

```
# generate a product of two 5-dimensional simplices.
PolyList = GenProdSimplex(5)
```

`GenSimplex` *Generator function for simplices*

Description

This function can be used to generate a d-dimensional unit simplex in H or V representation.

Usage

```
GenSimplex(dimension, repr)
```

Arguments

`dimension`	The dimension of the simplex.
`repr`	A string to declare the representation. It has to be 'H' for H-representation or 'V' for V-representation.

Value

A simplex in H or V-representation. For an H polytope the return value is a list with two elements: the "matrix" containing a $(d+1) \times d$ matrix A and the "vector" containing a $(d+1)$-dimensional vector b, s.t. $Ax \leq b$. When the V-representation is chosen the return value is a $(d+1) \times d$ matrix that containes the vertices row-wise.

Examples

```
# generate a 10-dimensional simplex in H-representation
PolyList = GenSimplex(10, 'H')

# generate a 20-dimensional simplex in V-representation
PolyList = GenSimplex(20, 'V')
```

GenSkinnyCube *Generator function for skinny hypercubes*

Description

This function can be used to generate a d-dimensional skinny hypercube only in H-representation.

Usage

```
GenSkinnyCube(dimension)
```

Arguments

dimension	The dimension of the skinny hypercube.

Value

A d-dimensional skinny hypercube in H-representation. The retutn value is a list with two elements: the "matrix" containing a $2d \times d$ matrix A and the "vector" containing a $2d$-dimensional vector b, s.t. $Ax \leq b$.

Examples

```
# generate a 10-dimensional skinny hypercube.
PolyList = GenSkinnyCube(10)
```

GenZonotope *Generator function for zonotopes*

Description

This function can be used to generate a d-dimensional zonotope defined by the Minkowski sum of m segments. We consider the $e_1, \dots, e_d$ generators and $m-d$ random generators. Then we shift the zonotope in order to contain the origin. The origin is the center of symmetry as well. It might needs rounding before the volume approximation using SequenceOfBalls or CoolingGaussian algorithms.

Usage

```
GenZonotope(dimension, NumGen)
```

Arguments

dimension	The dimension of the zonotope.
NumGen	The number of segments that generate the zonotope.

Value

A $m \times d$ matrix that containes the m d-dimensional segments.

Examples

```
# generate a 10-dimensional zonotope defined by the Minkowski sum of 20 segments
zonotope = GenZonotope(10, 20)
```

InnerBall — *Compute the Chebychev ball of a H-polytope*

Description

For a H-polytope described by a $m \times d$ matrix A and a m-dimensional vector b, s.t.: $Ax \leq b$, this function computes the largest inscribed ball (Chebychev ball) of that polytope by solving the corresponding linear program.

Usage

```
InnerBall(P)
```

Arguments

A	The matrix of the H-polytope.
b	The m-dimensional vector b that containes the constants of the m facets.

Value

A $(d+1)$-dimensional vector that containes the Chebychev ball. The first d coordinates corresponds to the center and the last one to the radius of the Chebychev ball.

Examples

```
# compute the Chebychev ball of a 2d unit simplex
A = matrix(c(-1,0,0,-1,1,1), ncol=2, nrow=3, byrow=TRUE)
b = c(0,0,1)
ball_vec = CheBall(A,b)

# compute the Chebychev ball of 10-dimensional cross polytope
PolyList = GenCross(10, 'H')
ball_vec = CheBall(PolyList$A, PolyList$b)
```

rand_rotate *Apply a random rotation to a convex polytope (H-polytope, V-polytope or a zonotope)*

Description

Given a convex H or V polytope or a zonotope as input this function applies a random rotation.

Usage

```
rand_rotate(P)
```

Arguments

A	Only for H-polytopes. The $m \times d$ matrix A that containes the directions of the m facets.
b	Only for H-polytopes. The m-dimensional vector b that containes the constants of the m facets s.t.: $Ax \leq b$.
V	Only for V-polytopes. The $m \times d$ matrix V that containes row-wise the m d-dimensional vertices of the polytope.
G	Only for zonotopes. The $m \times d$ matrix G that containes row-wise the m d-dimensional segments that define a zonotope.

Value

A random rotation of the polytope that is given as an input. For H-polytopes the return value is a list that containes a $m \times d$ matrix A and a m-dimensional vector b s.t.: $Ax \leq b$. For V-polytopes and zonotopes the return value is a $m \times d$ matrix that containes row-wise the d-dimensional vertices or segments respectively.

Examples

```
# rotate a H-polytope (2d unit simplex)
A = matrix(c(-1,0,0,-1,1,1), ncol=2, nrow=3, byrow=TRUE)
b = c(0,0,1)
listHpoly = rand_rotate(A=A, b=b)

# rotate a V-polytope (3d cube)
Vmat = GenCube(3, 'V')
matVpoly = rand_rotate(V=Vmat)

# rotate a 5-dimensional zonotope defined by the Minkowski sum of 15 segments
Zmat = GenZonotope(5,15)
MatZono = rand_rotate(G=Zmat)
```

`round_polytope` *Apply rounding to a convex polytope (H-polytope, V-polytope or a zonotope)*

Description

Given a convex H or V polytope or a zonotope as input this function computes a rounding based on minimum volume enclosing ellipsoid of a pointset.

Usage

```
round_polytope(P, method)
```

Arguments

`A`	Only for H-polytopes. The $m \times d$ matrix A that containes the directions of the m facets.
`b`	Only for H-polytopes. The m-dimensional vector b that containes the constants of the m facets s.t.: $Ax \leq b$.
`V`	Only for V-polytopes. The $m \times d$ matrix V that containes row-wise the m d-dimensional vertices of the polytope.
`G`	Only for zonotopes. The $m \times d$ matrix G that containes row-wise the m d-dimensional segments that define a zonotope.
`walk_length`	Optional. The number of the steps for the random walk. Default value is $\lfloor 10 + d/10 \rfloor$.
`ball_walk`	Optional. Boolean parameter to use ball walk, only for CG algorithm. Default value is false.
`delta`	Optional. The radius for the ball walk.
`coordinate`	Optional. A boolean parameter for the hit-and-run. True for Coordinate Directions HnR, false for Random Directions HnR. Default value is true.

Value

For H-polytopes the return value is a list that containes a $m \times d$ matrix A and a m-dimensional vector b s.t.: $Ax \leq b$. For V-polytopes and zonotopes the return value is a list with first element a $m \times d$ matrix that containes row-wise the d-dimensional vertices or segments respectively. For all the representations the returned list containes element "round_value" which is the determinant of the square matrix of the linear transformation that was applied on the polytope that is given as input.

Examples

```
# rotate a H-polytope (2d unit simplex)
A = matrix(c(-1,0,0,-1,1,1), ncol=2, nrow=3, byrow=TRUE)
b = c(0,0,1)
```

```
listHpoly = round_polytope(A=A, b=b)

# rotate a V-polytope (3d cube) using Random Directions HnR
Vmat = GenCube(3, 'V')
ListVpoly = round_polytope(V=Vmat, coordinate=FALSE)

# rotate a 10-dimensional zonotope defined by the Minkowski sum of 20 segments
Zmat = GenZonotope(10,20)
ListZono = round_polytope(G=Zmat)
```

`sample_points` *Sample points from a convex Polytope (H-polytope, V-polytope or a zonotope)*

Description

Sample N points from a H or a V-polytope or a zonotope with uniform or spherical gaussian - centered in an internal point- target distribution.

Usage

```
sample_points(P, N, distribution, method, InnerPoint)
```

Arguments

`N`	The number of points that the function is going to sample from the convex polytope. Default value is 100.
`A`	Only for H-polytopes. The $m \times d$ matrix A that containes the directions of the m facets.
`b`	Only for H-polytopes. The m-dimensional vector b that containes the constants of the m facets s.t.: $Ax \leq b$.
`V`	Only for V-polytopes. The $m \times d$ matrix V that containes row-wise the m d-dimensional vertices of the polytope.
`G`	Only for zonotopes. The $m \times d$ matrix G that containes row-wise the m d-dimensional segments that define a zonotope.
`walk_length`	Optional. The number of the steps for the random walk. Default value is $\lfloor 10 + d/10 \rfloor$.
`internal_point`	Optional. A d-dimensional vector that containes the coordinates of an internal point of the polytope. If it is not given then for H-polytopes the Chebychev center is computed, for V-polytopes $d+1$ vertices are picked randomly and the Chebychev center of the defined simplex is computed. For a zonotope that is defined by the Minkowski sum of m segments we use the origin.
`gaussian`	Optional. A boolean parameter to sample with gaussian target distribution. Default value is false.
`variance`	Optional. The variance for the spherical gaussian. Default value is 1.

`ball_walk`	Optional. Boolean parameter to use ball walk for the sampling. Default value is false.
`delta`	Optional. The radius for the ball walk.
`coordinate`	Optional. A boolean parameter for the hit-and-run. True for Coordinate Directions HnR, false for Random Directions HnR. Default value is true.

Value

A $d \times N$ matrix that containes, column-wise, the sampled points from the convex polytope.

Examples

```
# uniform distribution from a 3d cube described by a set of vertices
Vmat = GenCube(3, 'V')
points = sample_points(V=Vmat, N=1000)

# gaussian distribution from a 2d unit simplex in H-representation with variance = 2
A = matrix(c(-1,0,0,-1,1,1), ncol=2, nrow=3, byrow=TRUE)
b = c(0,0,1)
points = sample_points(A=A, b=b, gaussian=TRUE, variance=2)
```

`sliceOfSimplex` *Compute the percentage of the volume of the unit simplex that is contained in the intersection of a half-space and the unit simplex*

Description

When a half-space H is given as a pair of a vector $c \in R^d$ and a scalar $z0 \in R$ s.t.: $c^T x \leq z0$ this function calls the Ali's version of the Varsi formula.

Usage

```
sliceOfSimplex(c, z0)
```

Arguments

`c`	A d-dimensional vector that defines the direction of the hyperplane.
`z0`	The scalar that defines the half-space.

Value

The percentage of the volume of the unit simplex that is contained in the intersection of the given half-space and the unit simplex

References

Varsi, Giulio, "The multidimensional content of the frustum of the simplex," Pacific J. Math. 46, no. 1, 303–314, *1973.*

Ali, Mir M., "Content of the frustum of a simplex," Pacific J. Math. 48, no. 2, 313–322, *1973.*

Examples

```
# compute the frustum of H: -x+y<=0
c=c(-1,1)
z0=0
frustum = sliceOfSimplex(c, z0)
```

`volume` *The main R function for volume approximation of a convex Polytope (H-polytope, V-polytope or a zonotope)*

Description

For the volume approximation can be used two algorithms. Either SequenceOfBalls or CoolingGaussian. A H-polytope with m facets is described by a $m \times d$ matrix A and a m-dimensional vector b, s.t.: $Ax \leq b$. A V-polytope is described as a set of d-dimensional points. A zonotope is desrcibed by the Minkowski sum of d-dimensional segments.

Usage

```
volume(P, walk_length, error, InnerBall, Algo, WalkType, rounding)
```

Arguments

`P`	Only for H-polytopes. The $m \times d$ matrix A that containes the directions of the m facets.
`walk_length`	Optional. The number of the steps for the random walk. Default value is $\lfloor 10 + d/10 \rfloor$ for SequenceOfBalls and 1 for CoolingGaussian.
`error`	Optional. Declare the goal for the approximation error. Default value is 1 for SequenceOfBalls and 0.2 for CoolingGaussian.
`InnerBall`	Optional. A $d+1$ vector that containes an inner ball. The first d coordinates corresponds to the center and the last one to the radius of the ball. If it is not given then for H-polytopes the Chebychev ball is computed, for V-polytopes $d+1$ vertices are picked randomly and the Chebychev ball of the defined simplex is computed. For a zonotope that is defined by the Minkowski sum of m segments we compute the maximal r s.t.: $re_i \in Z$ for all $i = 1, \ldots, d$, then the ball centered at the origin with radius $r/\sqrt{d}$ is an internal ball.
`Algo`	Optional. A list that contains parameters for the CoolingGaussian algorithm. When it is null SequenceOfBalls is used as the default. • CGA boolean element. When it is true CoolingGaussian algorithm is used. • win_lenThe size of the window for the ratios' approximation in CG algorithm. Default value is $4 \cdot dimension^2 + 500$. • CA constant for the lower bound of $variance/mean^2$ in schedule annealing of CG algorithm. Default value is 2. • NThe number of points we sample in each step of schedule annealing in CG algorithm. Default value is $500C + dimension^2/2$.

- ratioParameter of schedule annealing of CG algorithm, larger ratio means larger steps in schedule annealing. Default value is $1 - 1/dimension$.
- fracThe fraction of the total error to spend in the first gaussian in CG algorithm. Default value is 0.1.

`WalkType` Optional. A list that contains parameters for the random walk method.

- methodA string that declares the method: (a) "hnr" for Hit-and-Run or (b) "bw" for ball walk. Default method is Hit-and-Run.
- coordinateA boolean parameter for Hit-and-Run. It has to be TRUE for Cordinate Directions Hit-and-Run or FALSE for Random Directions Hit-and-Run. Default method is Coordinate Directions Hnr.
- deltaThe radius for the ball walk.

`rounding` Optional. A boolean parameter to activate the rounding option. Default value is false.

Value

The approximation of the volume of a convex polytope.

References

I.Z.Emiris and V. Fisikopoulos, "Practical polytope volume approximation," ACM Trans. Math. Soft., *2014.*,

B. Cousins and S. Vempala, "A practical volume algorithm," Springer-Verlag Berlin Heidelberg and The Mathematical Programming Society, *2015.*

Examples

```
# calling SOB algorithm for a H-polytope (2d unit simplex)
A = matrix(c(-1,0,0,-1,1,1), ncol=2, nrow=3, byrow=TRUE)
b = c(0,0,1)
vol = volume(A=A, b=b)

# calling CG algorithm for a V-polytope (3d cube)
Vmat = GenSimplex(2,'V')
vol = volume(V=Vmat, CG=TRUE)

# calling CG algorithm for a 5-dimensional zonotope defined as the Minkowski sum of 10 segments
zonotope = GenZonotope(2, 4)
vol = volume(G=zonotope, rounding=TRUE, CG=TRUE)
```

Index