

Package ‘volesti’

June 22, 2020

Type Package

License LGPL-3

Title Volume Approximation and Sampling of Convex Polytopes

Author Vissarion Fisikopoulos <vissarion.fisikopoulos@gmail.com> [aut, cph, cre],
Apostolos Chalkis <tolis.chal@gmail.com> [cph, aut],
contributors in file inst/AUTHORS

Copyright file inst/COPYRIGHTS

Description

Provides an R interface for 'volesti' C++ package. 'volesti' computes estimations of volume of polytopes given by (i) a set of points, (ii) linear inequalities or (iii) Minkowski sum of segments (a.k.a. zonotopes). There are three algorithms for volume estimation as well as algorithms for sampling, rounding and rotating polytopes. Moreover, 'volesti' provides algorithms for estimating copulas useful in computational finance.

Version 1.1.1

Date 2020-06-22

Maintainer Vissarion Fisikopoulos <vissarion.fisikopoulos@gmail.com>

Depends Rcpp (>= 0.12.17)

Imports methods, stats

LinkingTo Rcpp, RcppEigen, BH

Suggests testthat

Encoding UTF-8

RoxygenNote 7.1.0

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

NeedsCompilation yes

Repository CRAN

Date/Publication 2020-06-22 19:10:02 UTC

R topics documented:

compute_indicators	2
copula	4
direct_sampling	5
exact_vol	6
file_to_polytope	7
frustum_of_simplex	8
gen_cross	8
gen_cube	9
gen_prod_simplex	10
gen_rand_hpoly	10
gen_rand_vpoly	11
gen_rand_zonotope	11
gen_simplex	12
gen_skinny_cube	13
Hpolytope	13
inner_ball	14
loadSdpaFormatFile	14
Rcpp_Spectrahedron	15
readSdpaFormatFile	15
rotate_polytope	16
round_polytope	17
sample_points	18
Spectrahedron	19
volume	20
Vpolytope	21
VpolytopeIntersection	22
writeSdpaFormatFile	22
Zonotope	23
zonotope_approximation	23
Index	25

compute_indicators	<i>Compute an indicator for each time period that describes the state of a market.</i>
--------------------	--

Description

Given a matrix that contains row-wise the assets' returns and a sliding window `win_length`, this function computes an approximation of the joint distribution (copula, e.g. see [https://en.wikipedia.org/wiki/Copula_\(probability_theory\)](https://en.wikipedia.org/wiki/Copula_(probability_theory))) between portfolios' return and volatility in each time period defined by `win_len`. For each copula it computes an indicator: If the indicator is large it corresponds to a crisis period and if it is small it corresponds to a normal period. In particular, the periods over which the indicator is greater than 1 for more than 60 consecutive sliding windows are warnings and for more than 100 are crisis. The sliding window is shifted by one day.

Usage

```
compute_indicators(
  returns,
  win_length = NULL,
  m = NULL,
  n = NULL,
  nwarning = NULL,
  ncrisis = NULL,
  seed = NULL
)
```

Arguments

returns	A d -dimensional vector that describes the direction of the first family of parallel hyperplanes.
win_length	Optional. The length of the sliding window. The default value is 60.
m	Optional. The number of slices for the copula. The default value is 100.
n	Optional. The number of points to sample. The default value is $5 \cdot 10^5$.
nwarning	Optional. The number of consecutive indicators larger than 1 required to declare a warning period. The default value is 60.
ncrisis	Optional. The number of consecutive indicators larger than 1 required to declare a crisis period. The default value is 100.
seed	Optional. A fixed seed for the number generator.

Value

A list that contains the indicators and the corresponding vector that label each time period with respect to the market state: a) normal, b) crisis, c) warning.

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

```
# simple example on random asset returns
asset_returns = replicate(10, rnorm(14))
market_states_and_indicators = compute_indicators(asset_returns, 10, 10, 10000, 2, 3)
```

 copula

 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 ellipsoids centered at the origin intersecting the canonical simplex, this function uniformly samples from the canonical simplex and constructs an approximation of the bivariate probability distribution, called copula (see [https://en.wikipedia.org/wiki/Copula_\(probability_theory\)](https://en.wikipedia.org/wiki/Copula_(probability_theory))). At least two families of hyperplanes or one family of hyperplanes and one family of ellipsoids have to be given as input.

Usage

```
copula(r1, r2 = NULL, sigma = NULL, m = NULL, n = NULL, seed = NULL)
```

Arguments

r1	The d -dimensional normal vector of the first family of parallel hyperplanes.
r2	Optional. The d -dimensional normal vector of the second family of parallel hyperplanes.
sigma	Optional. The $d \times d$ symmetric positive semidefinite matrix that describes the family of concentric ellipsoids centered at the origin.
m	The number of the slices for the copula. The default value is 100.
n	The number of points to sample. The default value is $5 \cdot 10^5$.
seed	Optional. A fixed seed for the number generator.

Value

A $m \times m$ numerical matrix that corresponds to a 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 random 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(r1 = h1, r2 = h2, m = 10, n = 100000)
```

```
# compute a copula for a family of parallel hyperplanes and a family of concentric ellipsoids
```

```

h = runif(n = 10, min = 1, max = 1000)
h = h / 1000
E = replicate(10, rnorm(20))
E = cov(E)
cop = copula(r1 = h, sigma = E, m = 10, n = 100000)

```

direct_sampling	<i>Sample perfect uniformly distributed points from well known convex bodies: (a) the unit simplex, (b) the canonical simplex, (c) the boundary of a hypersphere or (d) the interior of a hypersphere.</i>
-----------------	--

Description

The d -dimensional unit simplex is the set of points $\vec{x} \in \mathbb{R}^d$, s.t.: $\sum_i x_i \leq 1, x_i \geq 0$. The d -dimensional canonical simplex is the set of points $\vec{x} \in \mathbb{R}^d$, s.t.: $\sum_i x_i = 1, x_i \geq 0$.

Usage

```
direct_sampling(body, n, seed = NULL)
```

Arguments

body	A list to request exact uniform sampling from special well known convex bodies through the following input parameters: <ul style="list-style-type: none"> • <code>type</code> A string that declares the type of the body for the exact sampling: a) 'unit_simplex' for the unit simplex, b) 'canonical_simplex' for the canonical simplex, c) 'hypersphere' for the boundary of a hypersphere centered at the origin, d) 'ball' for the interior of a hypersphere centered at the origin. • <code>dimension</code> An integer that declares the dimension when exact sampling is enabled for a simplex or a hypersphere. • <code>radius</code> The radius of the d-dimensional hypersphere. The default value is 1.
n	The number of points that the function is going to sample.
seed	Optional. A fixed seed for the number generator.

Value

A $d \times n$ matrix that contains, column-wise, the sampled points from the convex polytope P.

References

R.Y. Rubinstein and B. Melamed, “Modern simulation and modeling” Wiley Series in Probability and Statistics, 1998.

A Smith, Noah and W Tromble, Roy, “Sampling Uniformly from the Unit Simplex,” Center for Language and Speech Processing Johns Hopkins University, 2004.

Examples

```
# 100 uniform points from the 2-d unit ball
points = direct_sampling(n = 100, body = list("type" = "ball", "dimension" = 2))
```

exact_vol	<i>Compute the exact volume of (a) a zonotope (b) an arbitrary simplex in V-representation or (c) if the volume is known and declared by the input object.</i>
-----------	--

Description

Given a zonotope (as an object of class Zonotope), this function computes the sum of the absolute values of the determinants of all the $d \times d$ submatrices of the $m \times d$ matrix G that contains row-wise the m d -dimensional segments that define the zonotope. For an arbitrary simplex that is given in V-representation this function computes the absolute value of the determinant formed by the simplex's points assuming it is shifted to the origin.

Usage

```
exact_vol(P)
```

Arguments

P A polytope

Value

The exact volume of the input polytope, for zonotopes, simplices in V-representation and polytopes with known exact volume

References

E. Gover and N. Krikorian, "Determinants and the Volumes of Parallelotopes and Zonotopes," Linear Algebra and its Applications, 433(1), 28 - 40, 2010.

Examples

```
# compute the exact volume of a 5-dimensional zonotope defined by the Minkowski sum of 10 segments
Z = gen_rand_zonotope(2, 5)
vol = exact_vol(Z)

# compute the exact volume of a 2-d arbitrary simplex
V = matrix(c(2,3,-1,7,0,0),ncol = 2, nrow = 3, byrow = TRUE)
P = Vpolytope$new(V)
vol = exact_vol(P)
```

```
# compute the exact volume the 10-dimensional cross polytope
P = gen_cross(10, 'V')
vol = exact_vol(P)
```

file_to_polytope	<i>function to get an ine or an ext file and returns the corresponding polytope</i>
------------------	---

Description

For an ".ine" file it generates the corresponding H-polytope. For an ".ext" file it generates the corresponding V-polytope or zonotope. For more details on those file formats see https://github.com/GeomScale/volume_approximation/blob/develop/doc/cpp_interface.md#polytope-input.

Usage

```
file_to_polytope(path, zonotope)
```

Arguments

path	A string that contains the path to an ine or a ext file. The ine file describes a H-polytope and ext file describes a V-polytope or a zonotope.
zonotope	A boolean parameter. It has to be TRUE when the path leads to an .ext file that describes a zonotope.

Value

A polytope class. If the path corresponds to an ine file then the return value represents a H-polytope. If it corresponds to an ext file the return value represents a V-polytope (default choice) or a zonotope if the second argument is TRUE.

Examples

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

frustum_of_simplex	<i>Compute the percentage of the volume of the simplex that is contained in the intersection of a half-space and the simplex.</i>
--------------------	---

Description

A half-space H is given as a pair of a vector $a \in R^d$ and a scalar $z0 \in R$ s.t.: $a^T x \leq z0$. This function calls the Ali's version of the Varsi formula to compute a frustum of the simplex.

Usage

```
frustum_of_simplex(a, z0)
```

Arguments

a	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 simplex that is contained in the intersection of a given half-space and the 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: -x1+x2<=0
a=c(-1,1)
z0=0
frustum = frustum_of_simplex(a, z0)
```

gen_cross	<i>Generator function for cross polytopes</i>
-----------	---

Description

This function generates the d -dimensional cross polytope in H- or V-representation.

Usage

```
gen_cross(dimension, representation)
```

Arguments

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

Value

A polytope class representing a cross polytope in H- or V-representation.

Examples

```
# generate a 10-dimensional cross polytope in H-representation
P = gen_cross(5, 'H')

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

 gen_cube

Generator function for hypercubes

Description

This function generates the d -dimensional unit hypercube $[-1, 1]^d$ in H- or V-representation.

Usage

```
gen_cube(dimension, representation)
```

Arguments

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

Value

A polytope class representing the unit d -dimensional hypercube in H- or V-representation.

Examples

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

# generate a 15-dimension hypercube in V-representation
P = gen_cube(5, 'V')
```

gen_prod_simplex *Generator function for product of simplices*

Description

This function generates a $2d$ -dimensional polytope that is defined as the product of two d -dimensional unit simplices in H-representation.

Usage

```
gen_prod_simplex(dimension)
```

Arguments

dimension The dimension of the simplices.

Value

A polytope class representing the product of the two d -dimensional unit simplices in H-representation.

Examples

```
# generate a product of two 5-dimensional simplices.
P = gen_prod_simplex(5)
```

gen_rand_hpoly *Generator function for random H-polytopes*

Description

This function generates a d -dimensional polytope in H-representation with m facets. We pick m random hyperplanes tangent on the d -dimensional unit hypersphere as facets.

Usage

```
gen_rand_hpoly(dimension, nfacets, seed = NULL)
```

Arguments

dimension The dimension of the convex polytope.
nfacets The number of the facets.
seed Optional. A fixed seed for the generator.

Value

A polytope class representing a H-polytope.

Examples

```
# generate a 10-dimensional polytope with 50 facets
P = gen_rand_hpoly(10, 50)
```

gen_rand_vpoly *Generator function for random V-polytopes*

Description

This function generates a d -dimensional polytope in V-representation with m vertices. We pick m random points from the boundary of the d -dimensional unit hypersphere as vertices.

Usage

```
gen_rand_vpoly(dimension, nvertices, generator = NULL, seed = NULL)
```

Arguments

dimension	The dimension of the convex polytope.
nvertices	The number of the vertices.
generator	The body that the generator samples uniformly the vertices from: (a) 'cube' or (b) 'sphere'.
seed	Optional. A fixed seed for the generator.

Value

A polytope class representing a V-polytope.

Examples

```
# generate a 10-dimensional polytope defined as the convex hull of 25 random vertices
P = gen_rand_vpoly(10, 25)
```

gen_rand_zonotope *Generator function for zonotopes*

Description

This function generates a random d -dimensional zonotope defined by the Minkowski sum of m d -dimensional segments. The function considers m random directions in R^d . There are three strategies to pick the length of each segment: a) it is uniformly sampled from $[0, 100]$, b) it is random from $\mathcal{N}(50, (50/3)^2)$ truncated to $[0, 100]$, c) it is random from $Exp(1/30)$ truncated to $[0, 100]$.

Usage

```
gen_rand_zonotope(dimension, nsegments, generator = NULL, seed = NULL)
```

Arguments

dimension	The dimension of the zonotope.
nsegments	The number of segments that generate the zonotope.
generator	The distribution to pick the length of each segment from $[0, 100]$: (a) 'uniform', (b) 'gaussian' or (c) 'exponential'.
seed	Optional. A fixed seed for the generator.

Value

A polytope class representing a zonotope.

Examples

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

gen_simplex

Generator function for simplices

Description

This function generates the d -dimensional unit simplex in H- or V-representation.

Usage

```
gen_simplex(dimension, representation)
```

Arguments

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

Value

A polytope class representing the d -dimensional unit simplex in H- or V-representation.

Examples

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

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

gen_skinny_cube	<i>Generator function for skinny hypercubes</i>
-----------------	---

Description

This function generates a d -dimensional skinny hypercube $[-1, 1]^{d-1} \times [-100, 100]$.

Usage

```
gen_skinny_cube(dimension)
```

Arguments

dimension The dimension of the skinny hypercube.

Value

A polytope class representing the d -dimensional skinny hypercube in H-representation.

Examples

```
# generate a 10-dimensional skinny hypercube.
P = gen_skinny_cube(10)
```

Hpolytope	<i>An R class to represent H-polytopes.</i>
-----------	---

Description

A H-polytope is a convex polytope defined by a set of linear inequalities or equivalently a d -dimensional H-polytope with m facets is defined by a $m \times d$ matrix A and a m -dimensional vector b , s.t.: $P = \{x \mid Ax \leq b\}$.

Fields

- A A $m \times d$ numerical matrix A
- b m -dimensional vector b
- type An integer that declares the representation of the polytope. For H-representation the default value is 1.
- dimension The dimension of the polytope.
- volume The volume of the polytope, if it is known.

inner_ball *Compute an inscribed ball of a convex polytope*

Description

For a H-polytope described by a $m \times d$ matrix A and a m -dimensional vector b , s.t.: $P = \{x \mid Ax \leq b\}$, this function computes the largest inscribed ball (Chebychev ball) by solving the corresponding linear program. For both zonotopes and V-polytopes the function computes the minimum r s.t.: $re_i \in P$ for all $i = 1, \dots, d$. Then the ball centered at the origin with radius r/\sqrt{d} is an inscribed ball.

Usage

```
inner_ball(P)
```

Arguments

P A convex polytope. It is an object from class (a) Hpolytope or (b) Vpolytope or (c) Zonotope or (d) VpolytopeIntersection.

Value

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

Examples

```
# compute the Chebychev ball of the 2d unit simplex
P = gen_simplex(2, 'H')
ball_vec = inner_ball(P)

# compute an inscribed ball of the 3-dimensional unit cube in V-representation
P = gen_cube(3, 'V')
ball_vec = inner_ball(P)
```

loadSdpaFormatFile *Read a SDPA format file*

Description

Read a SDPA format file

Usage

```
loadSdpaFormatFile(inputFile = NULL)
```

Arguments

inputFile Name of the input file

Value

A list with two named items: an item "matrices" which is a list of the matrices and an vector "objFunction"

Examples

```
path = system.file('extdata', package = 'volesti')
l = loadSdpaFormatFile(paste0(path, '/sdpa_n2m3.txt'))
```

Rcpp_Spectrahedron *An Rcpp class to represent spectrahedra, exposed to R via modules.*

Description

A spectrahedron is a convex body defined by a linear matrix inequality of the form $A_0 + x_1 A_1 + \dots + x_n A_n \preceq 0$. The matrices A_i are symmetric $m \times m$ real matrices and $\preceq 0$ denoted negative semidefiniteness.

Details

matrices A list with the matrices A_0, A_1, \dots, A_n

readSdpaFormatFile *Read a SDPA format file*

Description

Read a SDPA format file and return a spectrahedron (an object of class Spectrahedron) which is defined by the linear matrix inequality in the input file, and the objective function.

Usage

```
readSdpaFormatFile(path)
```

Arguments

path Name of the input file

Value

A list with two named items: an item "matrices" which is an object of class Spectrahedron and an vector "objFunction"

Examples

```

path = system.file('extdata', package = 'volesti')
l = readSdpaFormatFile(paste0(path, '/sdpa_n2m3.txt'))
Spectrahedron = l$spectrahedron
objFunction = l$objFunction

```

rotate_polytope	<i>Apply a random rotation to a convex polytope (H-polytope, V-polytope, zonotope or intersection of two V-polytopes)</i>
-----------------	---

Description

Given a convex H- or V- polytope or a zonotope or an intersection of two V-polytopes as input, this function applies (a) a random rotation or (b) a given rotation by an input matrix T .

Usage

```
rotate_polytope(P, T = NULL, seed = NULL)
```

Arguments

P	A convex polytope. It is an object from class (a) Hpolytope, (b) Vpolytope, (c) Zonotope, (d) intersection of two V-polytopes.
T	Optional. A $d \times d$ rotation matrix.
seed	Optional. A fixed seed for the random linear map generator.

Details

Let P be the given polytope and Q the rotated one and T be the matrix of the linear transformation.

- If P is in H-representation and A is the matrix that contains the normal vectors of the facets of Q then AT contains the normal vectors of the facets of P .
- If P is in V-representation and V is the matrix that contains column-wise the vertices of Q then $T^T V$ contains the vertices of P .
- If P is a zonotope and G is the matrix that contains column-wise the generators of Q then $T^T G$ contains the generators of P .
- If M is a matrix that contains column-wise points in Q then $T^T M$ contains points in P .

Value

A list that contains the rotated polytope and the matrix T of the linear transformation.

Examples

```
# rotate a H-polytope (2d unit simplex)
P = gen_simplex(2, 'H')
poly_matrix_list = rotate_polytope(P)

# rotate a V-polytope (3d cube)
P = gen_cube(3, 'V')
poly_matrix_list = rotate_polytope(P)

# rotate a 5-dimensional zonotope defined by the Minkowski sum of 15 segments
Z = gen_rand_zonotope(3,6)
poly_matrix_list = rotate_polytope(Z)
```

round_polytope	<i>Apply rounding to a convex polytope (H-polytope, V-polytope or a zonotope)</i>
----------------	---

Description

Given a convex H or V polytope or a zonotope as input this function brings the polytope in rounded position based on minimum volume enclosing ellipsoid of a pointset.

Usage

```
round_polytope(P, seed = NULL)
```

Arguments

P	A convex polytope. It is an object from class (a) Hpolytope or (b) Vpolytope or (c) Zonotope.
seed	Optional. A fixed seed for the number generator.

Value

A list with 4 elements: (a) a polytope of the same class as the input polytope class and (b) the element "T" which is the matrix of the inverse linear transformation that is applied on the input polytope, (c) the element "shift" which is the opposite vector of that which has shifted the input polytope, (d) the element "round_value" which is the determinant of the square matrix of the linear transformation that is applied on the input polytope.

References

I.Z.Emiris and V. Fisikopoulos, "Practical polytope volume approximation," ACM Trans. Math. Soft., 2018.,

Michael J. Todd and E. Alper Yildirim, "On Khachiyan's Algorithm for the Computation of Minimum Volume Enclosing Ellipsoids," Discrete Applied Mathematics, 2007.

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)
P = Hpolytope$new(A, b)
listHpoly = round_polytope(P)

# rotate a V-polytope (3d unit cube) using Random Directions HnR with step equal to 50
P = gen_cube(3, 'V')
ListVpoly = round_polytope(P)

# round a 2-dimensional zonotope defined by 6 generators using ball walk
Z = gen_rand_zonotope(2,6)
ListZono = round_polytope(Z)
```

sample_points	<i>Sample uniformly or normally distributed points from a convex Polytope (H-polytope, V-polytope, zonotope or intersection of two V-polytopes).</i>
---------------	--

Description

Sample n points with uniform or multidimensional spherical gaussian -with a mode at any point- as the target distribution.

Usage

```
sample_points(P, n, random_walk = NULL, distribution = NULL, seed = NULL)
```

Arguments

- | | |
|-------------|--|
| P | A convex polytope. It is an object from class (a) Hpolytope or (b) Vpolytope or (c) Zonotope or (d) VpolytopeIntersection. |
| n | The number of points that the function is going to sample from the convex polytope. |
| random_walk | Optional. A list that declares the random walk and some related parameters as follows: <ul style="list-style-type: none"> • walk A string to declare the random walk: i) 'CDHR' for Coordinate Directions Hit-and-Run, ii) 'RDHR' for Random Directions Hit-and-Run, iii) 'BaW' for Ball Walk, iv) 'BiW' for Billiard walk, v) 'BCDHR' boundary sampling by keeping the extreme points of CDHR or vi) 'BRDHR' boundary sampling by keeping the extreme points of RDHR. The default walk is 'BiW' for the uniform distribution or 'CDHR' for the Gaussian distribution. • walk_length The number of the steps per generated point for the random walk. The default value is 5 for 'BiW' and $\lfloor 10 + d/10 \rfloor$ otherwise. • nburns The number of points to burn before start sampling. |

- `starting_point` A d -dimensional numerical vector that declares a starting point in the interior of the polytope for the random walk. The default choice is the center of the ball as that one computed by the function `inner_ball()`.
 - `BaW_rad` The radius for the ball walk.
 - `L` The maximum length of the billiard trajectory.
- `distribution` Optional. A list that declares the target density and some related parameters as follows:
- `density` A string: (a) 'uniform' for the uniform distribution or b) 'gaussian' for the multidimensional spherical distribution. The default target distribution is uniform.
 - `variance` The variance of the multidimensional spherical gaussian. The default value is 1.
 - `mode` A d -dimensional numerical vector that declares the mode of the Gaussian distribution. The default choice is the center of the as that one computed by the function `inner_ball()`.
- `seed` Optional. A fixed seed for the number generator.

Value

A $d \times n$ matrix that contains, column-wise, the sampled points from the convex polytope P .

Examples

```
# uniform distribution from the 3d unit cube in H-representation using ball walk
P = gen_cube(3, 'H')
points = sample_points(P, n = 100, random_walk = list("walk" = "BaW", "walk_length" = 5))

# gaussian distribution from the 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)
P = Hpolytope$new(A,b)
points = sample_points(P, n = 100, distribution = list("density" = "gaussian", "variance" = 2))

# uniform points from the boundary of a 2-dimensional random H-polytope
P = gen_rand_hpoly(2,20)
points = sample_points(P, n = 100, random_walk = list("walk" = "BRDHR"))
```

Spectrahedron

An R class to represent spectrahedra.

Description

A spectrahedron is a convex body defined by a linear matrix inequality of the form $A_0 + x_1 A_1 + \dots + x_n A_n \preceq 0$. The matrices A_i are symmetric $m \times m$ real matrices and $\preceq 0$ denoted negative semidefiniteness.

Fields

- matrices A list with the matrices A_0, A_1, \dots, A_n

volume	<i>The main function for volume approximation of a convex Polytope (H-polytope, V-polytope, zonotope or intersection of two V-polytopes)</i>
--------	--

Description

For the volume approximation can be used three algorithms. Either CoolingBodies (CB) or SequenceOfBalls (SOB) or CoolingGaussian (CG). An H-polytope with m facets is described by a $m \times d$ matrix A and a m -dimensional vector b , s.t.: $P = \{x \mid Ax \leq b\}$. A V-polytope is defined as the convex hull of m d -dimensional points which correspond to the vertices of P . A zonotope is described by the Minkowski sum of m d -dimensional segments.

Usage

```
volume(P, settings = NULL, rounding = NULL, seed = NULL)
```

Arguments

P	A convex polytope. It is an object from class a) Hpolytope or b) Vpolytope or c) Zonotope or d) VpolytopeIntersection.
settings	Optional. A list that declares which algorithm, random walk and values of parameters to use, as follows: <ul style="list-style-type: none"> • algorithm A string to set the algorithm to use: a) 'CB' for CB algorithm, b) 'SoB' for SOB algorithm or b) 'CG' for CG algorithm. The default algorithm for H-polytopes is 'CB' when $d \leq 200$ and 'CG' when $d > 200$. For the other representations the default algorithm is 'CB'. • error A numeric value to set the upper bound for the approximation error. The default value is 1 for SOB algorithm and 0.1 otherwise. • random_walk A string that declares the random walk method: a) 'CDHR' for Coordinate Directions Hit-and-Run, b) 'RDHR' for Random Directions Hit-and-Run, c) 'BaW' for Ball Walk, or 'BiW' for Billiard walk. For CB and SOB algorithms the default walk is 'CDHR' for H-polytopes and 'BiW' for the other representations. For CG algorithm the default walk is 'CDHR' for H-polytopes and 'RDHR' for the other representations. • walk_length An integer to set the number of the steps for the random walk. The default value is $\lfloor 10 + d/10 \rfloor$ for 'SOB' and 1 otherwise. • win_len The length of the sliding window for CB or CG algorithm. The default value is $400 + 3d^2$ for CB or $500 + 4d^2$ for CG. • hpoly A boolean parameter to use H-polytopes in MMC of CB algorithm when the input polytope is a zonotope. The default value is TRUE when the order of the zonotope is < 5, otherwise it is FALSE.
rounding	Optional. A boolean parameter for rounding. The default value is TRUE for V-polytopes and FALSE otherwise.
seed	Optional. A fixed seed for the number generator.

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., 2018.,

A. Chalkis and I.Z.Emiris and V. Fisikopoulos, "Practical Volume Estimation by a New Annealing Schedule for Cooling Convex Bodies," CoRR, abs/1905.05494, 2019.,

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 (3d unit simplex)
HP = gen_cube(3, 'H')
vol = volume(HP)

# calling CG algorithm for a V-polytope (2d simplex)
VP = gen_simplex(2, 'V')
vol = volume(VP, settings = list("algorithm" = "CG"))

# calling CG algorithm for a 2-dimensional zonotope defined as the Minkowski sum of 4 segments
Z = gen_rand_zonotope(2, 4)
vol = volume(Z, settings = list("random_walk" = "RDHR", "walk_length" = 2))
```

Vpolytope

An R class to represent V-polytopes.

Description

A V-polytope is defined as the convex hull of m d -dimensional points which corresponds to its vertices.

Fields

- V A $m \times d$ numerical matrix that contains the vertices row-wise
- `type` An integer that declares the representation of the polytope. For V-representation the default value is 2.
- `dimension` The dimension of the polytope.
- `volume` The volume of the polytope, if it is known.

VpolytopeIntersection *An R class to represent the intersection of two V-polytopes.*

Description

An intersection of two V-polytopes, P_1 , P_2 , is defined by the intersection of the two corresponding convex hulls.

Fields

- V1 The numerical matrix that contains the vertices of P_1 row-wise.
- V2 The numerical matrix that contains the vertices of P_2 row-wise.
- type An integer that declares the representation of the polytope. For this polytope the default value is 4.
- dimension The dimension of the polytope.
- volume The volume of the polytope, if it is known.

writeSdpaFormatFile *Write a SDPA format file*

Description

Outputs a spectrahedron (the matrices defining a linear matrix inequality) and a vector (the objective function) to a SDPA format file.

Usage

```
writeSdpaFormatFile(
  spectrahedron = NULL,
  objectiveFunction = NULL,
  outputFile = NULL
)
```

Arguments

spectrahedron A spectrahedron in n dimensions; must be an object of class Spectrahedron
 objectiveFunction A numerical vector of length n
 outputFile Name of the output file

Zonotope	<i>An R class to represent zonotopes.</i>
----------	---

Description

A zonotope is a convex polytope defined by the Minkowski sum of m d -dimensional segments.

Fields

- `G` A $m \times d$ numerical matrix that contains the segments (or generators) row-wise
- `type` An integer that declares the representation of the polytope. For zonotopes the default value is 3.
- `dimension` The dimension of the polytope.
- `volume` The volume of the polytope, if it is known.

zonotope_approximation

A function to over-approximate a zonotope with PCA method and to evaluate the approximation by computing a ratio of fitness.

Description

For the evaluation of the PCA method the exact volume of the approximation body is computed and the volume of the input zonotope is computed by CoolingBodies algorithm. The ratio of fitness is $R = vol(P)/vol(P_{red})$, where P_{red} is the approximate polytope.

Usage

```
zonotope_approximation(Z, fit_ratio = NULL, settings = NULL, seed = NULL)
```

Arguments

<code>Z</code>	A zonotope.
<code>fit_ratio</code>	Optional. A boolean parameter to request the computation of the ratio of fitness.
<code>settings</code>	Optional. A list that declares the values of the parameters of CB algorithm as follows: <ul style="list-style-type: none"> • <code>error</code> A numeric value to set the upper bound for the approximation error. The default value is 0.1. • <code>walk_length</code> An integer to set the number of the steps for the random walk. The default value is 1. • <code>win_len</code> The length of the sliding window for CB algorithm. The default value is 200. • <code>hpoly</code> A boolean parameter to use H-polytopes in MMC of CB algorithm. The default value is TRUE when the order of the zonotope is < 5, otherwise it is FALSE.
<code>seed</code>	Optional. A fixed seed for the number generator.

Value

A list that contains the approximation body in H-representation and the ratio of fitness

References

A.K. Kopetzki and B. Schurmann and M. Althoff, "Methods for Order Reduction of Zonotopes," IEEE Conference on Decision and Control, 2017.

Examples

```
# over-approximate a 2-dimensional zonotope with 10 generators and compute the ratio of fitness
Z = gen_rand_zonotope(2,8)
retList = zonotope_approximation(Z = Z)
```

Index

[,Rcpp_Spectrahedron,ANY,ANY,ANY-method
(Rcpp_Spectrahedron), [15](#)
[,Rcpp_Spectrahedron-method
(Rcpp_Spectrahedron), [15](#)
\$,Rcpp_Spectrahedron-method
(Rcpp_Spectrahedron), [15](#)
\$<-,Rcpp_Spectrahedron-method
(Rcpp_Spectrahedron), [15](#)

compute_indicators, [2](#)
copula, [4](#)

direct_sampling, [5](#)

exact_vol, [6](#)

file_to_polytope, [7](#)
filepaths<-,Rcpp_Spectrahedron-method
(Rcpp_Spectrahedron), [15](#)
frustum_of_simplex, [8](#)

gen_cross, [8](#)
gen_cube, [9](#)
gen_prod_simplex, [10](#)
gen_rand_hpoly, [10](#)
gen_rand_vpoly, [11](#)
gen_rand_zonotope, [11](#)
gen_simplex, [12](#)
gen_skinny_cube, [13](#)

Hpolytope, [13](#)

inner_ball, [14](#)

loadSdpaFormatFile, [14](#)

Rcpp_Spectrahedron, [15](#)
Rcpp_Spectrahedron-class
(Rcpp_Spectrahedron), [15](#)
readSdpaFormatFile, [15](#)
rotate_polytope, [16](#)
round_polytope, [17](#)
sample_points, [18](#)
Spectrahedron, [19](#)
volume, [20](#)
Vpolytope, [21](#)
VpolytopeIntersection, [22](#)
writeSdpaFormatFile, [22](#)
Zonotope, [23](#)
zonotope_approximation, [23](#)