volesti: Volume Approximation and Sampling for Convex Polytopes in R

by Apostolos Chalkis and Vissarion Fisikopoulos

Abstract Sampling from high-dimensional distributions and volume approximation of convex bodies are fundamental operations that appear in optimization, finance, engineering, artificial intelligence, and machine learning. In this paper, we present **volesti**, an R package that provides efficient, scalable algorithms for volume estimation, uniform, and Gaussian sampling from convex polytopes. **volesti** scales to hundreds of dimensions, handles efficiently three different types of polyhedra and provides non existing sampling routines to R. We demonstrate the power of **volesti** by solving several challenging problems using the R language.

Introduction

High-dimensional sampling from multivariate distributions with Markov Chain Monte Carlo (MCMC) algorithms is a fundamental problem with many applications in science and engineering (Iyengar, 1988; Somerville, 1998; Genz and Bretz, 2009; Schellenberger and Palsson, 2009; Venzke et al., 2021). In particular, multivariate integration over a convex set and volume approximation of such sets —a special case of integration— have accumulated a broad amount of effort over the last decades. Nevertheless, those problems are computationally hard for general dimensions (Dyer and Frieze, 1988). MCMC algorithms have made remarkable progress efficiently solving the problems of sampling and volume estimation of convex bodies while enjoying great theoretical guarantees (Chen et al., 2018; Lee and Vempala, 2018; Mangoubi and Vishnoi, 2019). However, theoretical algorithms cannot be applied efficiently to real-life computations. For example, the asymptotic analysis by Lovász and Vempala (2006) hides some large constants in the complexity, and in Lee and Vempala (2018), the step of the random walk used for sampling is too small to be an efficient choice in practice. Therefore, practical algorithms have been designed by relaxing the theoretical guarantees and applying new algorithmic and statistical techniques to perform efficiently while at the same time meeting the requirements for high accuracy results (Emiris and Fisikopoulos, 2014; Cousins and Vempala, 2016; Chalkis et al., 2019).

In this paper, we present **volesti** (Fisikopoulos et al., 2020), an R package containing a variety of high-dimensional MCMC methods for sampling from multivariate distributions restricted to a convex polytope and randomized algorithms for volume estimation of convex polytopes. In particular, it includes efficient implementations of three practical volume algorithms—Sequence of Balls (SoB) (Emiris and Fisikopoulos, 2014), Cooling Gaussians (CG) (Cousins and Vempala, 2016), and Cooling convex Bodies (CB) (Chalkis et al., 2019). In addition to volume estimation, **volesti** provides efficient implementations for Random-Directions and Coordinate-Directions Hit and Run (RDHR and CDHR) (Smith, 1984), Ball Walk (BaW) (Hastings, 1970), Billiard Walk (BiW) (Polyak and Gryazina, 2014). The first three can be used to sample from multivariate uniform or spherical Gaussian distributions (centered at any point), while BiW can be employed, by definition, only for uniform sampling. On the whole, **volesti** is the first R package that:

(a) performs high-dimensional volume estimation,

(b) efficiently handles three different types of polyhedra in high dimensions, namely H-polytopes, V-polytopes, and Z-polytopes,

(c) provides—previously absent from R—MCMC sampling algorithms for uniform and truncated Gaussian distributions, namely BaW, CDHR, and BiW,

(d) solves some challenging problems in finance, engineering, and applied mathematics.

On top of **volesti** presentation, we illustrate the usage of **volesti** in the study of convergence of various random walks (e.g., Figure 3) and accuracy of volume estimation methods. Regarding applications, in the last section, we illustrate how one can (a) exploit **volesti** to detect shock events in stock markets following the results by Calès et al. (2018), (b) evaluate zonotope approximation in engineering (Kopetzki et al., 2017), and (c) approximate the number of linear extensions of a partially ordered set, which is useful in various applications in artificial intelligence and machine learning.

To improve the presentation of the current paper, detailed comparisons and benchmarking of R packages–including **volesti**–for solving the problems of MCMC sampling, volume computation, and numerical integration are presented in a separate blog post (Chalkis and Fisikopoulos, 2021).

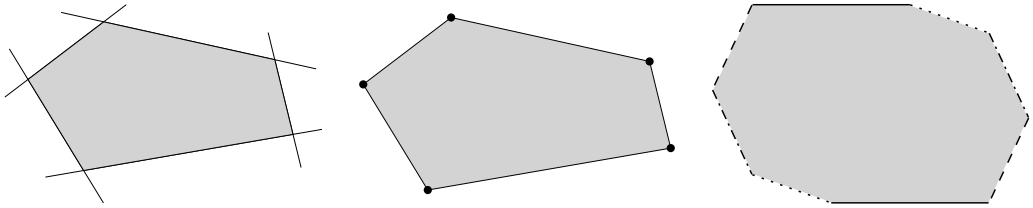

Figure 1: Examples of three different polytope representations. From left to right: an H-polytope, a V-polytope, and a Z-polytope (a sum of four segments).

Related R software and applications

Considering MCMC methods to sample from multivariate distributions are divided into two main categories: truncated to a convex body and untruncated distributions. For the first category—which clearly is the main focus of this paper—an important case is the truncated Gaussian distribution which arises in several applications in statistics. Bolin and Lindgren (2015) sample from truncated Gaussian distributions in a novel importance sampling method to study Markov processes that exceed a certain level. Wadsworth and Tawn (2014) use sampling from a specific truncated Gaussian distribution to develop a novel method for likelihood inference, while Huser and Davison (2013) sample from the same distribution for likelihood estimation for max-stable processes. In curve prediction, they exploit Gaussian sampling to compute simultaneous confidence bands to forecast a full curve from explanatory variables (Azaïs et al., 2010). Grün and Hornik (2012) study the posterior distribution for Bayesian inference on mixed regression models to represent human immunodeficiency virus ribonucleic acid levels, a Gaussian restricted to a convex polytope. In Albert and Chib (1993), the probit regression model for binary outcomes have an underlying normal regression structure on latent continuous data; sampling from the posterior distribution of the parameters involves sampling from a truncated Gaussian distribution.

Another important special case is the truncated uniform distribution. In systems biology the flux space of a metabolic network is represented by a convex polytope (Haraldsdòttir et al., 2017); uniform sampling from the interior of that polytope could lead to important biological insights. In computational finance, the set of all possible portfolios in a stock market is in general a convex polytope. Volume computation and uniform sampling from that set is useful for crises detection (Calès et al., 2018) and efficient portfolio allocation and analysis (Pouchkarev et al., 2004; Hallerbach et al., 2002).

Considering R packages for the truncated case, there is **tmg** (Pakman, 2015) implementing exact Hamiltonian Monte Carlo (HMC) with boundary reflections as well as **multinomineq** (Heck, 2019), **lineqGPR** (Lopez, 2019), **restrictedMVN** (Taylor and Benjamini, 2016), **tmvmixnorm** (Ma et al., 2020) implementing variations of the Gibbs sampler. To our knowledge, the only two R packages for uniform sampling is **hitandrun** (van Valkenhoef and Tervonen, 2019) and **limSolve** (den Meersche K. et al., 2009), which exposes the R function `xsample()` (den Meersche et al., 2009). For the untruncated case, packages **HybridMC** (Morey, 2009), **rhmc** (Sartório, 2018), **mcmc** (Geyer and Johnson, 2020), and **MHadaptive** (Chivers, 2012) provide implementations for HMC and Metropolis Hastings algorithms, respectively. For volume computation, the only existing package, **geometry** (Roussel et al., 2019), computes the volume of the convex hull of a set of points and is based on the C++ library, qhull (Barber et al., 1996).

Algorithms and polytopes

Convex polytopes

Convex polytopes are a special case of convex bodies with special interest in many scientific fields and applications. For example, in optimization, the feasible region of a linear program is a polytope, and in finance, the set of portfolios is usually expressed by a polytope (i.e., the simplex). More formally, an H-polytope is defined as

$$P := \{x \mid Ax \leq b\} \subseteq \mathbb{R}^d,$$

where $A \in \mathbb{R}^{m \times d}$ and $b \in \mathbb{R}^m$, and we say that P is given in H-representation. Each row $a_i^T \in \mathbb{R}^d$ of matrix A corresponds to a normal vector that defines a halfspace $a_i^T x \leq b_i$, $i = [m]$. The intersection of those halfspaces defines the polytope P, and the hyperplanes $a_i^T x = b_i$, $i = [m]$ are called facets of P. A V-polytope is given by a matrix $V \in \mathbb{R}^{d \times n}$, which contains n points column-wise, and we say that P is given in V-representation. The points of P that cannot be written as convex combinations of

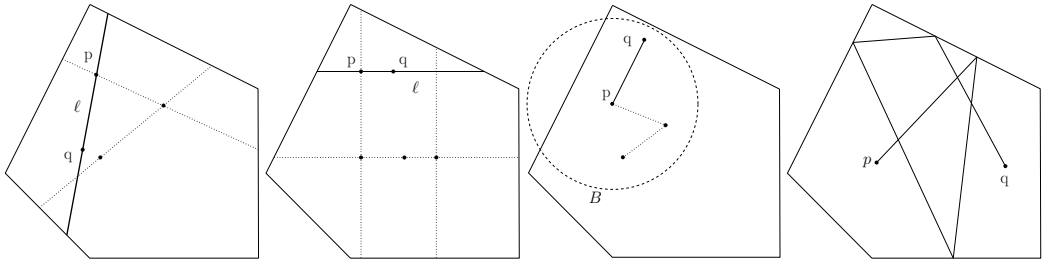

Figure 2: Examples of random walks. From left to right: RDHR, CDHR, BaW, BiW; p is the point at current step and q the new point computed; ℓ is the line computed by RDHR and CDHR; B is the ball computed by BaW. Dotted lines depict previous steps.

other points of P are called vertices. The polytope P is defined as the convex hull of those vertices, i.e., the smallest convex set that contains them. Equivalently, a V-polytope can be seen as the linear map of the canonical simplex $\Delta^{n-1} := \{x \in \mathbb{R}^n \mid x_i \geq 0,\ \sum_{i=1}^n x_i = 1\}$ according to matrix V, i.e.,

$$P := \{x \in \mathbb{R}^d \mid \exists y \in \Delta^{n-1} : x = Vy\}$$

A Z-polytope (or zonotope) is given by a matrix $G \in \mathbb{R}^{d\times k}$, which contains k segments column-wise, which are called generators. In this case, P is defined as the Minkowski sum of those segments and we say that it is given in Z-representation. We call *order* of a Z-polytope the ratio between the number of segments over the dimension. Equivalently, P can be expressed as the linear map of the hypercube $[-1,1]^k$ with matrix G, i.e.

$$P := \{x \in \mathbb{R}^d \mid \exists y \in [-1,1]^k : x = Gy\}.$$

Thus, a Z-polytope is a centrally symmetric convex body, as a linear map of an other centrally symmetric convex body. Examples of an H-polytope, a V-polytope and a Z-polytope in two dimensions are illustrated in Figure 1. For an excellent introduction to polytope theory, we recommend the book of Ziegler (1995).

MCMC sampling and geometric random walks

We more formally define here the four geometric random walks implemented in **volesti**, namely, Hit and Run (two variations, RDHR, and CDHR), Ball walk (Baw), and Billiard walk (BiW). They are illustrated in Figure 2 for two dimensions.

In general, if $f : \mathbb{R}^n \to \mathbb{R}_+$ is a non-negative integrable function, then it defines a measure π_f on any measurable subset A of $\mathbb{R}^d$,

$$\pi_f(A) = \frac{\int_A f(x)dx}{\int_{\mathbb{R}^d} f(x)dx}$$

Let ℓ be a line in $\mathbb{R}^d$ and let $\pi_{\ell,f}$ be the restriction of π to ℓ,

$$\pi_{\ell,f}(P) = \frac{\int_{p+tu\in P} f(p+tu)dt}{\int_\ell f(x)dx},$$

where p is a point on ℓ and u a unit vector parallel to ℓ.

Algorithm 1 describes the general Hit and Run procedure. When the line ℓ in line (1.) of the pseudocode is chosen uniformly at random from all possible lines passing through p, then the walk is called Random-Directions Hit and Run (Smith, 1984). To pick a random direction through point $p \in \mathbb{R}^d$, we could sample d numbers $g_1, \ldots, g_d$ from $\mathcal{N}(0,1)$, and then the vector $u = (g_1, \ldots, g_d)/\sqrt{\sum g_i^2}$ is uniformly distributed on the surface of the d-dimensional unit ball. A special case is called Coordinate-Directions Hit and Run (Smith, 1984), where we pick ℓ uniformly at random from the set of d lines that passing through p and are parallel to the coordinate axes.

Algorithm 1: Hit_and_run(P, p, f)

Input : Polytope $P \subset \mathbb{R}^d$, point $p \in P$, $f : \mathbb{R}^d \to \mathbb{R}_+$
Output: A point $q \in P$

1. Pick a line ℓ through p.
2. **return** a random point on the chord $\ell \cap P$ chosen from the distribution $\pi_{\ell,f}$.

The Ball walk (Algorithm 2) needs, additionally to Hit and Run, a radius δ as input. In particular, Ball walk is a special case of Metropolis Hastings (Hastings, 1970) when the target distribution is truncated. For both Hit and Run and Ball walk π_f, is the stationary distribution of the random walk. If $f(x) = e^{-||x-x_0||^2/2\sigma^2}$, then the target distribution is the multidimensional spherical Gaussian with variance σ^2 and its mode at x_0. When f is the indicator function of P, then the target distribution is the uniform distribution.

Algorithm 2: Ball_walk(P, p, δ, f)

Input : Polytope $P \subset \mathbb{R}^d$, point $p \in P$, radius δ, $f\mathbb{R}^d \to \mathbb{R}_+$
Output: A point $q \in P$

1. Pick a uniform random point x from the ball of radius δ centered at p
2. **return** x with probability $\min\left\{1, \frac{f(x)}{f(p)}\right\}$; **return** p with the remaining probability.

Billiard walk is a random walk for sampling from the uniform distribution (Polyak and Gryazina, 2014). It tries to emulate the movement of a gas particle during the physical phenomena of filling uniformly a vessel. Algorithm 3 implements Billiard walk, where $\langle \cdot, \cdot \rangle$ is the inner product between two vectors, $|| \cdot ||$ is the ℓ^2 norm, and $| \cdot |$ is the length of a segment.

Algorithm 3: Billiard_walk(P, p, τ, R)

Input : Polytope $P \subset \mathbb{R}^d$, current point of the random walk $p \in P$, length of trajectory parameter $\tau \in \mathbb{R}_+$, upper bound on the number of reflections $R \in \mathbb{N}$
Output: A point $q \in P$

1. Set the length of the trajectory $L \leftarrow -\tau \ln \eta$, $\eta \sim \mathcal{U}(0,1)$;
   Set the number of reflections $n \leftarrow 0$ and $p_0 \leftarrow p$;
   Pick a uniformly distributed direction on the unit sphere, v;
2. Update $n \leftarrow n + 1$; **If** $n > R$ **return** p_0;
3. Set $\ell \leftarrow \{p + tv, 0 \leq t \leq L\}$;
4. **If** $\partial P \cap \ell = \emptyset$ **return** $p + Lv$;
5. Update $p \leftarrow \partial P \cap \ell$; Let s be the inner normal vector of the tangent plane on p, s.t. $||s|| = 1$;
   Update $L \leftarrow L - |P \cap \ell|$, $v \leftarrow v - 2\langle v, s \rangle s$; **goto** 2;

Every random walk starts from a point in the convex body and perform a number of steps called *walk length*. The larger the walk length is, the less correlated the final with the starting point will be. The number of steps to get an uncorrelated point, that is, a point approximately drawn from π_f is called *mixing time*. The number of operations performed to generate a point is called *cost per step*. Hence, the total cost to generate a random point is the mixing time multiplied by the cost per step.

random walk	mixing time	cost/step H-polytope	cost/step V- & Z-polytope
RDHR (Lovász and Vempala, 2006)	$O^*(d^3)$	$O(md)$	2 LPs
CDHR (Laddha and Vempala, 2020)	$O^*(d^{10})$	$O(m)$	2 LPs
BaW (Lee and Vempala, 2017)	$O^*(d^{2.5})$	$O(md)$	1 LP
BiW (Polyak and Gryazina, 2014)	**?**	$O((d+R)m)$	R LPs

Table 1: Overview of the random walks implemented in **volesti**. LP for linear program; R for the number of reflections per point in BiW; D for the diameter of the polytope.

Table 1 displays known complexities for mixing time and cost per step. For the mixing time of RDHR, we assume that P is well rounded, i.e., $B_d \subseteq P \subseteq C\sqrt{d}B_d$, where B_d is the unit ball and C a constant. In general, if $rB_d \subseteq P \subseteq RB_d$ then RDHR mixing time is $O^*(d^2(R/r)^2)$. For the mixing

time of Ball walk in Table 1, we assume that P is in isotropic position and the radius of the ball is $\delta = \Theta(1/\sqrt{d})$ (Lee and Vempala, 2017). There are no theoretical bounds on mixing time for CDHR and BiW. Polyak and Gryazina (2014) experimentally shows that BiW converges faster than RDHR when $\tau \approx diam(P)$, i.e., the diameter of P. CDHR is the main paradigm for sampling in practice from H-polytopes, e.g., in volume computation (Emiris and Fisikopoulos, 2014) and biology (Haraldsdòttir et al., 2017). The main reason behind this is the small cost per step and the same convergence in practice as RDHR (Emiris and Fisikopoulos, 2014). For V- and Z-polytopes, the cost per step of BiW is comparable with that of CDHR. Moreover, it converges fast to the uniform distribution (Chalkis et al., 2019). The fact that all above walks are implemented in **volesti** enable us to empirically evaluate their mixing time using R (e.g., Figure 3).

Volume estimation

As mentioned before, volume computation is a hard problem, so given a polytope P, we have to employ randomized algorithms to approximate $\mathrm{vol}(P)$ within some target relative error ϵ and high probability. The keys to the success of those algorithms are the Multiphase Monte Carlo (MMC) technique and sampling from multivariate distributions with geometric random walks.

In particular, we define a sequence of functions $\{f_0, \ldots, f_q\}$, $f_i : \mathbb{R}^d \to \mathbb{R}$. Then, $\mathrm{vol}(P)$ is given by the following telescopic product:

$$\mathrm{vol}(P) = \int_P dx = \int_P f_q(x)dx \frac{\int_P f_{q-1}(x)dx}{\int_P f_q(x)dx} \cdots \frac{\int_P dx}{\int_P f_0(x)dx} \tag{1}$$

Then, we need to:

- Fix the sequence such that q is as small as possible.
- Select f_i such that each integral ratio can be efficiently estimated.
- Estimate $\int_P f_q(x)dx$.

For a long time researchers, e.g., Lovász et al. (1997), set f_i to be indicator functions of concentric balls intersecting P. It follows that $\int_P f_i(x)dx = \mathrm{vol}(B_i \cap P)$, and the sequence of convex bodies $P = P_1 \supseteq \cdots \supseteq P_q$, $P_i = B_i \cap P$ forms a telescopic product of ratios of volumes, while for $\mathrm{vol}(P_q)$ there is a closed formula. Assuming $rB_d \subset P \subset RB_d$, then $q = O(d \lg R/r)$. The trick now is that we do not have to compute the exact value of each ratio $r_i = \mathrm{vol}(P_i)/\mathrm{vol}(P_{i+1})$, but we can use sampling-rejection to estimate it within some target relative error ϵ_i. If r_i is bounded, then $O(1/\epsilon_i^2)$ uniformly distributed points in P_{i+1} suffices. Another crucial aspect is the sandwiching ratio R/r of P which has to be as small as possible. This was tackled by a rounding algorithm, that is bringing P to nearly isotropic position (Lovász et al., 1997).

The SoB algorithm follows this paradigm and deterministically defines the sequence of P_i such that $0.5 \leq \mathrm{vol}(P_i)/\mathrm{vol}(P_{i+1}) \leq 1$. In the CG algorithm, each f_i is a spherical multidimensional Gaussian distribution, and the algorithm uses an annealing schedule (Lovász and Vempala, 2006) to fix the sequence of those Gaussians. The SoB algorithm uses a similar annealing schedule but to fix a sequence of convex bodies P_i. As far as performance is concerned, the CB algorithm is the most efficient choice for H-polytopes in less than 200 dimensions and for V- and Z-polytopes in any dimension. For the rest of the cases, the user should choose CG algorithm.

Package

The package **volesti** combines the efficiency of C++ and the popularity and usability of R. The package uses the eigen library (Guennebaud et al., 2010) for linear algebra, lpsolve library (Berkelaar et al., 2004) for solving linear programs, and boost random library (Maurer and Watanabe, 2017) (part of Boost C++ libraries) for random numbers and distributions. All the code development is performed on github platform. The package is available in Comprehensive R Archive Network (CRAN) and is regularly updated with new features and bug fixes. We employ continuous integration to test the package on various systems and deploy environments. There is detailed documentation of all the exposed R classes and functions publicly available. We maintain a contribution tutorial to help users and researchers who want to contribute to the development or propose a bug-fix. The package is shipped under the `LGPL-3` license to be open to all the scientific communities. We use **Rcpp** (Eddelbuettel et al., 2020b) to interface C++ with R. In particular, we create one **Rcpp** function for each procedure (such as sampling, volume estimation etc.) and we export it as an R function.

In the following sections, we demonstrate the use of **volesti**. The R scripts in the following sections use only standard R functions, **volesti**. In a single script in Section 2.4, we use **Rfast** to compute the assets' compound return in a stock market.

Polytope classes and generators

The package **volesti** comes with three classes to handle different representations of polytopes. Table 2 demonstrates the exposed R classes. The names of the classes are the names of polytope representations as defined in the previous section. Each polytope class has a few variable members that describe a specific polytope, demonstrated in Table 2. The matrices and the vectors in Table 2 correspond to those in the polytope definitions. The `integer` variable `type` implies the representation: `1` is for H-polytopes, `2` for V-polytopes, `3` for Z-polytopes. The `numerical` variable `volume` corresponds to the volume of the polytopes if it is known. **volesti** provides standard and random polytope generators. The first produce well-known polytopes such as cubes, cross polytopes, and simplices and assign the value of the exact volume to `volume` variable. The second are random generators using various probability distributions and methods to produce a variety of different random polytopes; notably the generated polytopes have unknown volume.

Class	Constructor	Variable members
`"Hpolytope"`	`Hpolytope(A,b)`	$A \in \mathbb{R}^{m\times d}$, $b \in \mathbb{R}^m$, `integer type`, `numerical volume`
`"Vpolytope"`	`Vpolytope(V)`	$V \in \mathbb{R}^{n\times d}$, `integer type`, `numerical volume`
`"Zonotope"`	`Zonotope(G)`	$G \in \mathbb{R}^{k\times d}$, `integer type`, `numerical volume`

Table 2: Overview of the polytopes' classes in **volesti**.

Uniform sampling from polytopes

A core feature of **volesti** is approximate sampling from convex bodies with uniform or spherical Gaussian target distribution using the four geometric random walks defined above.

The following R script samples 1000 points from the 100-dimensional hypercube $[-1,1]^{100}$ defined as P and stores them in a list.

```
R> d = 100
R> P = gen_cube(d, 'H')

R> samples = sample_points(P, random_walk = list(
                           "walk" = "RDHR", "burn-in"=1000, "walk_length" = 5),
                           n = 1000)
```

We use the Random Directions Hit-and-Run (RDHR) walk. Other choices are: Coordinate Directions Hit-and-Run (CDHR), Ball Walk (BaW), and Billiard Walk (BiW). Setting the parameter `burn-in` to 1000 means that **volesti** burns the first 1000 points RDHR generates; setting walk_length to 5 means that we keep in the list, one every five generated points. The default choice for the target distribution is the uniform distribution.

To evaluate the efficiency of **volesti** sampling routines, one could measure the run-time and estimate the effective sample size (Geyer, 2011) per second. To estimate the effective sample size in R, a standard choice is the package **coda** (Plummer et al., 2020). In Chalkis and Fisikopoulos (2021), benchmarks show that **volesti** can be up to $\sim 2\,500$ times faster than **hitandrun** for uniform sampling from a polytope.

Moreover, using **volesti** and R, we can empirically study the mixing time of the geometric random walks implemented in **volesti**. To this end, we uniformly sample from a random rotation of the 200-dimensional hypercube $[-1,1]^{200}$. First, we generate the hypercube and use `rotate_polytope()` that returns the rotated polytope and the matrix of the linear transformation.

```
R> d = 200
R> num_of_points = 1000
R> P = gen_cube(d, 'H')
R> retList = rotate_polytope(P, rotation = list("seed" = 5))
R> T = retList$T
R> P = retList$P
```

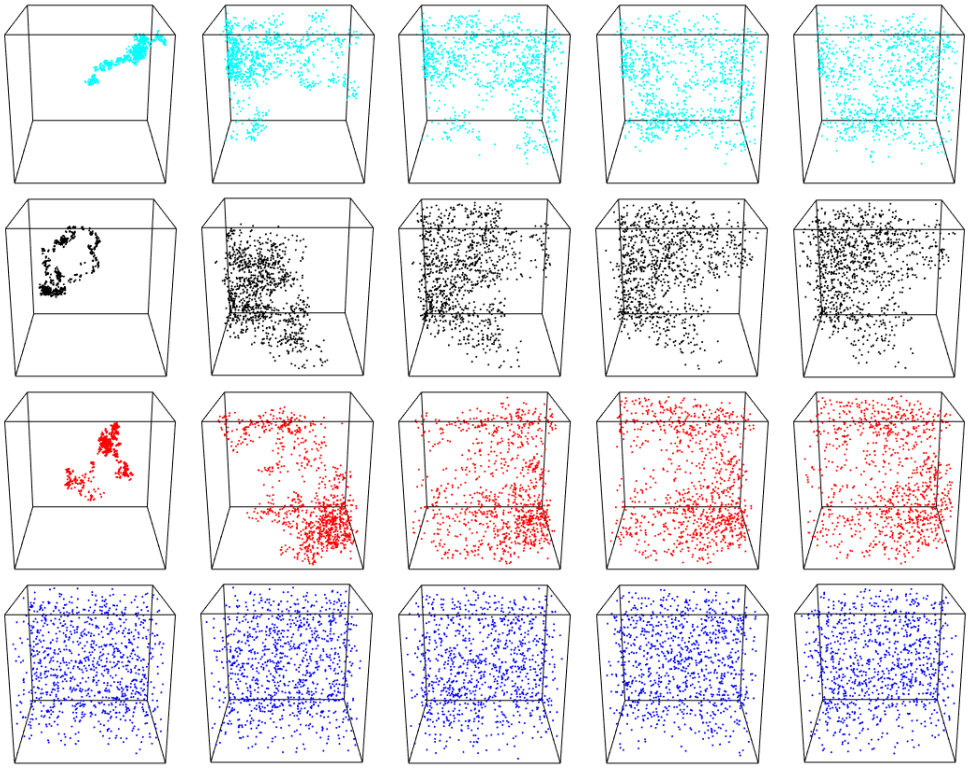

Figure 3: Uniform sampling from a random rotation of the hypercube $[-1, 1]^{200}$. We map the sample back to $[-1, 1]^{200}$, and then we project them on $\mathbb{R}^3$ by keeping the first three coordinates. Each row corresponds to a different walk: BaW, CDHR, RDHR, BiW. Each column to a different walk length: {1, 50, 100, 150, 200}. That is, the sub-figure in the third row and the forth column corresponds to RDHR with 150 walk length.

Then, we use `sample_points()` to sample from the rotated cube with various walk lengths to test the practical mixing of the random walk.

```
R> for (i in c(1, seq(from = 50, to = 200, by = 50))){
     points1 = t(T) %*% sample_points(P, n = num_of_points, random_walk = list(
                                      "walk" = "BaW", "walk_length" = i, "seed" = 5))
     points2 = t(T) %*% sample_points(P, n = num_of_points, random_walk = list(
                                      "walk" = "CDHR", "walk_length" = i, "seed" = 5))
     points3 = t(T) %*% sample_points(P, n = num_of_points, random_walk = list(
                                      "walk" = "RDHR", "walk_length" = i, "seed" = 5))
     points4 = t(T) %*% sample_points(P, n = num_of_points, random_walk = list(
                                      "walk" = "BiW", "walk_length" = i, "seed" = 5))
   }
```

Finally, we map the points back to $[-1, 1]^{200}$ using the inverse transformation, and then we project all the sample points on $\mathbb{R}^3$, or equivalently on the 3D cube $[-1, 1]^3$, by keeping the first three coordinates. We plot the results in Figure 3.

Note that, in general, perfect uniform sampling in the rotated polytope would result to perfect uniformly distributed points in the 3D cube $[-1, 1]^3$. Hence, Figure 3 shows an advantage of BiW in mixing time for this scenario compared to the other walks—it mixes relatively well even with one step (i.e. walk length). Notice also that the mixing of both CDHR and RDHR seem similar while it is slightly better than the mixing of BaW.

Gaussian sampling from polytopes

In many Bayesian models, the posterior distribution is a multivariate Gaussian distribution restricted to a specific domain. We illustrate the usage of **volesti** for the case of the truncation being the canonical simplex $\Delta^n = \{x \in R^n \mid x_i \geq 0, \sum_i x_i = 1\}$, which is of special interest. This situation typically occurs whenever the unknown parameters can be interpreted as fractions or probabilities. Thus, it appears in

many important applications (Altmann et al., 2014). In particular, we consider the following density,

$$f(x|\mu,\Sigma) \propto \begin{cases} exp[-\frac{1}{2}(x-\mu)^T\Sigma(x-\mu)], & \text{if } x \in \Delta^n, \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

Clearly, the support of the density in Equation (2) is defined by a convex subset of a linear subspace of $\mathbb{R}^n$. Thus, to sample from $f(x|\mu,\Sigma)$, we apply a proper linear transformation, induced by a matrix $N \in \mathbb{R}^{n\times(n-1)}$ that maps the support to a full-dimensional polytope in $\mathbb{R}^{n-1}$, while the covariance matrix changes accordingly to $\Sigma' = N^T\Sigma N$. Then, we apply a Cholesky decomposition to $\Sigma' = LL^T$ and employ the linear transformation induced by L to transform the distribution into a spherical Gaussian distribution.

In the following R script we first generate a random 100-dimensional positive definite matrix Σ. Then, we sample from the multivariate Gaussian distribution with the covariance matrix being Σ and the mode being the center of the canonical simplex Δ^n. To achieve this goal, we first apply all the necessary linear transformations to both the probability density function and the Δ^n to obtain the standard Gaussian distribution, $\mathcal{N}(0, I_n)$, restricted to a general full-dimensional simplex.

```
R> d = 100
R> S = matrix( rnorm(d*d,mean=0,sd=1), d, d) #random covariance matrix
R> S = S %*% t(S)
R> shift = rep(1/d, d)
R> A = -diag(d)
R> b = rep(0,d)
R> b = b - A %*% shift
R> Aeq = t(as.matrix(rep(1,d), 10,1))
R> N = pracma::nullspace(Aeq)
R> A = A %*% N #transform the truncation into a full dimensional polytope
R> S = t(N) %*% S %*% N
R> A = A %*% t(chol(S)) #Cholesky decomposition to transform to the standard Gaussian
R> P = Hpolytope(A=A, b=as.numeric(b)) #new truncation
```

Next, we use the `sample_points()` function to sample from the standard Gaussian distribution restricted to the computed simplex, and we apply the inverse transformations to obtain a sample in the initial space.

```
R> samples = sample_points(P, n = 100000, random_walk =
                           list("walk"="CDHR", "burn-in"=1000,
                           "starting_point" = rep(0, d-1),
                           distribution = list("density" = "gaussian",
                           "mode" = rep(0, d-1))))
R> samples_initial_space = N %*% samples +
   kronecker(matrix(1, 1, 100000), matrix(shift, ncol = 1))
```

In the previous script, we set the starting point of the walk to the mode of the Gaussian, i.e., the origin. Note that the default choice in **volesti** for the target distribution in the case of Gaussian sampling is the standard Gaussian; that is, the target distribution in the above script.

Considering comparisons, **volesti** is at least one order of magnitude faster than **restrictedMVN** and **tmg** for computing a sample of similar quality. For more details on comparison with other packages, we refer to (Chalkis and Fisikopoulos, 2021).

Volume estimation

Let us now give an example of how we approximate the volume of a polytope in **volesti**. Since this is a randomized algorithm, it makes sense to compute some statistics for the output values using R when approximating the volume of the 10-dimensional cube $[-1,1]^{10}$ generated as an H-polytope.

```
R> P = gen_cube(10, 'H')
R> volumes = list()
R> for (i in seq_len(20)) {
     volumes[[i]] = volume(P, settings = list("error" = 0.2))
   }
```

By changing the error to 0.02, we can obtain more accurate results. The results are illustrated in Figure 4. Note that the exact volume is 1024.

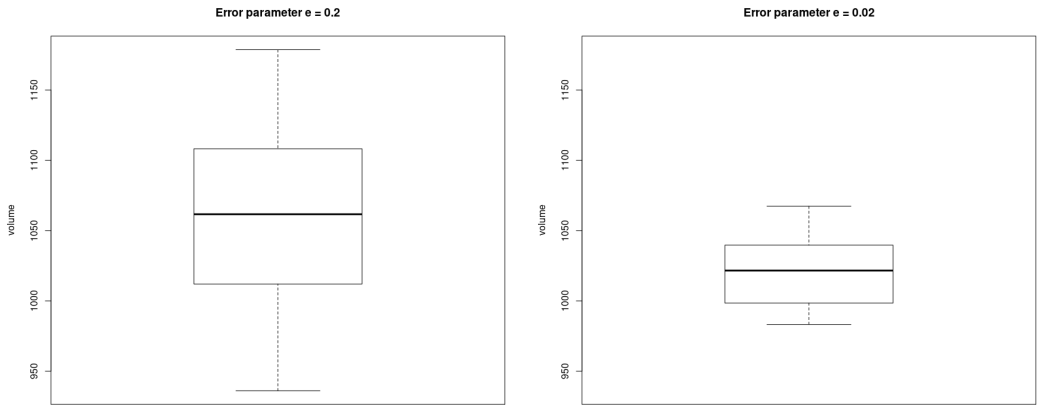

Figure 4: The boxplot of the estimated volumes of the hypercude $[-1,1]^{10}$ by **volesti**. Left, the input error parameter is $\epsilon = 0.2$, and right is $\epsilon = 0.02$.

To understand the need for randomized computation in high dimensions implemented in **volesti**, we can consider the state-of-the-art volume computation in R today, namely, **geometry**. It implements a deterministic algorithm in which run-time grows exponentially with the dimension. Because of the later property, **geometry** generally, fails to terminate for polytope in dimension $d \geq 20$. See (Chalkis and Fisikopoulos, 2021) for comparison details with **geometry**.

The following script illustrates the usage and efficiency of **volesti** to compute the volume of high-dimensional polytopes. In particular, a V-polytope, namely the cross-polytope, and an H-polytope, namely the hypercube.

```
R> d = 80
R> P = gen_cross(80, 'V')  #generate a cross polytope in V-representation

R> time = system.time({
         volume_estimation = volume(P, settings = list(
                                    "algorithm" = "CB", "random_walk" = "BiW",
                                    "seed" = 127)) })
R> exact_volume = 2^d/prod(1:d)
R> cat(time[1], abs(volume_estimation - exact_volume) / exact_volume)

82.874   0.074434

R> P = gen_cube(d, 'H')   #generate a hypercube polytope in H-representation

R> time = system.time({
         volume_estimation = volume(P, settings = list(
                                    "algorithm" = "CB", "random_walk" = "CDHR",
                                    "seed" = 23)) })
R> exact_volume = 2^d
R> cat(time[1], abs(volume_estimation - exact_volume) / exact_volume)

0.657   0.067633
```

For V- and Z- polytopes the most efficient choice of random walk is BiW, while for H-polytopes is CDHR. This explains why we use different random walks in the previous script. However, notice that the run-time for the H-polytope is two order of magnitude smaller. This happens because the cost per step of a random walk in a V-polytope increases comparing to H-polytopes.

Last but not least, **volesti** provides random polytope generators. The following command estimates the volume of a randomly generated V-polytope that is the convex hull of 40 uniformly generated random points from the 20-dimensional cube.

```
R> P = gen_rand_vpoly(20, 40, generator = list("body" =  "cube", "seed" = 1729))
R> volume_estimation = volume(P)
```

The next call estimates the volume of an H-polytope randomly generated as an intersection of 180 linear halfspaces computed by random tangent hyperplanes on an 60-dimensional hypersphere.

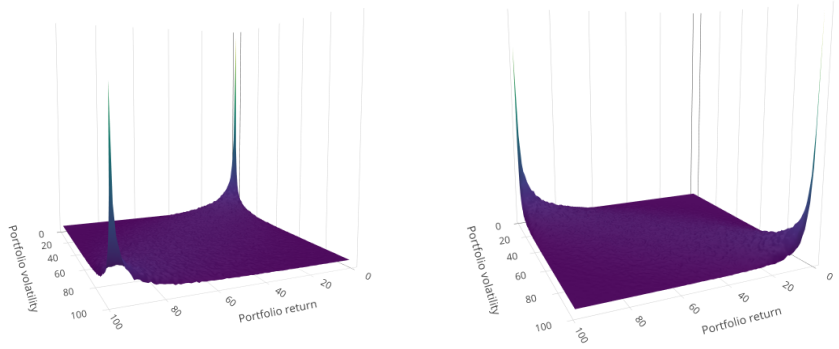

Figure 5: Left, a copula that corresponds to normal period ($07/03/2007 - 31/05/2007$), $I = 0.2316412$. Right, a copula that corresponds to a crisis period ($18/12/2008 - 13/03/2009$), $I = 5.610785$; x axis is for return and y axis is for volatility.

```
R> P = gen_rand_hpoly(60, 180, generator = list('constants' = 'sphere'))
R> volume_estimation = volume(P)
```

Since the exact volume of those polytopes is unknown, the accuracy of the computed estimation is unknown and statistical methods such as the effective sample size (Geyer, 2011) could be used.

Applications

We demonstrate **volesti**'s potential to solve challenging problems. More specifically, we provide detailed use-cases for applications in finance (crises detection and portfolio scoring), decision and control, multivariate integration, and artificial intelligence.

Financial crises detection and portfolio scoring

In this subsection, we present how one could employ **volesti** to detect financial crises or shock events in stock markets by following the method of Calès et al. (2018). For all the examples in the sequel, we use a set of 52 popular exchange-traded funds (ETFs) and the US central bank (FED) rate of return publicly available from `https://stanford.edu/class/ee103/portfolio.html`. The following script is used to load the data.

```
R> MatReturns = read.table("https://stanford.edu/class/ee103/data/returns.txt",
                           sep = ",")
R> MatReturns = MatReturns[-c(1, 2), ]
R> dates = as.character(MatReturns$V1)
R> MatReturns = as.matrix(MatReturns[ ,-c(1, 54)])
R> MatReturns = matrix(as.numeric(MatReturns), nrow = dim(MatReturns )[1], ncol =
                       dim(MatReturns )[2], byrow = FALSE)
R> nassets = dim(MatReturns)[2]
```

The method uses the copula representation to capture the dependence between portfolios' returns and volatility. A copula is an approximation of the bivariate joint distribution while both marginals follow the uniform distribution. In normal times, portfolios are characterized by slightly positive returns and moderate volatility, in up-market times (typically bubbles) by high returns and low volatility, and during financial crises by strongly negative returns and high volatility. Thus, when a copula implies a positive dependence (see Figure 5 left), then it probably comes from a normal period. On the other side, when the dependence between portfolios' return and volatility is negative (see Figure 5 right), the copula probably comes from a crisis period. The first case occurs when the indicator that computes the ratio between the red mass over the blue mass is smaller than 1, and the second case when that indicator is larger than 1. The function `copula()` can be used to compute such copulas. When two vectors of returns are given as input by the user, then the computed copula is related to the problem of the momentum effect in stock markets.

The following script produces Figure 5 by setting the starting and the stopping date for the left and the right plot, respectively. To compute the copula, we use the compound asset return, which is the rate of return for capital over a cumulative series of time (Calès et al., 2018).

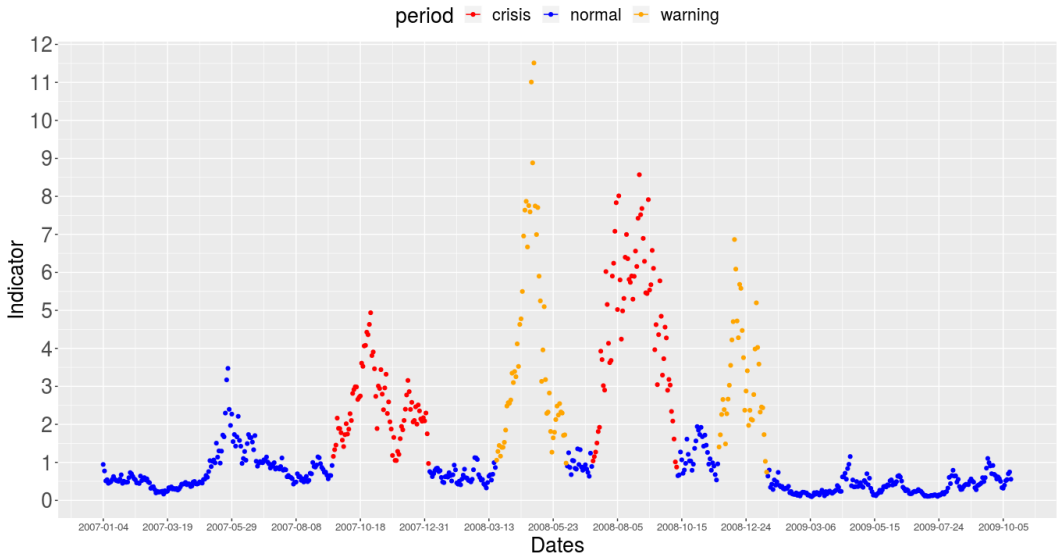

Figure 6: The values of the indicators from 2007-01-04 until 2010-01-04. We mark the crisis periods with red, the warning periods with orange, and the normal periods with blue that **volesti** identifies.

```
R> row1 = which(dates %in% "2008-12-18")
R> row2 = which(dates %in% "2009-03-13")
R> compound_asset_return = Rfast::colprods(1 + MatReturns[row1:row2, ]) - 1
R> mass = copula(r1 = compound_asset_return, sigma = cov(MatReturns[row1:row2, ]),
                 m = 100, n = 1e+06, seed = 5)
```

Moreover, the function `compute_indicators()` computes the copulas of all the sets of `win_len` consecutive days and returns the corresponding indicators and the states of the market during the given time period. The next script takes as input the daily returns of all the 52 assets from 01/04/2007 until 04/01/2010. When the indicator is ≥ 1 for more than 30 days, we issue a warning, and when it is for more than 60 days, we mark this period as a crisis (see Figure 6).

```
R> row1 = which(dates %in% "2007-01-04")
R> row2 = which(dates %in% "2010-01-04")
R> market_analysis = compute_indicators(returns =  MatReturns[row1:row2, ],
                                        parameters = list("win_len" = 60, "m" = 100,
                                        "n" = 1e+06, "nwarning" = 30, "ncrisis" = 60,
                                        "seed" = 5))
R> I = market_analysis$indicators
R> market_states = market_analysis$market_states
```

We compare the results with the database for financial crises in European countries proposed in Duca et al. (2017). The only listed crisis for this period is the sub-prime crisis (from December 2007 to June 2009). Notice that Figure 6 successfully points out 4 crisis events in that period (2 crisis and 2 warning periods) and detects sub-prime crisis as a W-shape crisis.

As a second financial application, we will use **volesti** to evaluate the performance of a given portfolio. In particular, **volesti** computes the proportion of all possible allocations that the given portfolio outperforms. This score independently introduced in Pouchkarev (2005); Guegan et al. (2011); Banerjee and Hung (2011), and is an alternative to more classical choices for the evaluation of the performance of a portfolio as the Sharpe-like ratios proposed in the 1960's by Jensen (1967); Sharpe (1966); Treynor (2015). However, the efficient computation of that score was uncertain until Calès et al. (2018) notice that Varsi's algorithm (Varsi, 1973) can be used to perform robust computations in high dimensions. Varsi's algorithm is implemented in **volesti** by the function `frustum_of_simplex()` and computes volumes in thousands of dimensions in just a few milliseconds on modest hardware. As an example, the following R script let us know that on 03/13/2009, any portfolio with a return of 0.002 outperforms almost 48% of all possible portfolios.

```
R> R = MatReturns[which(dates %in% "2009-03-13"), ]
R> R0 = 0.002
R> tim = system.time({ exact_score = frustum_of_simplex(R, R0) })
R> cat(exact_score, tim[3])
```

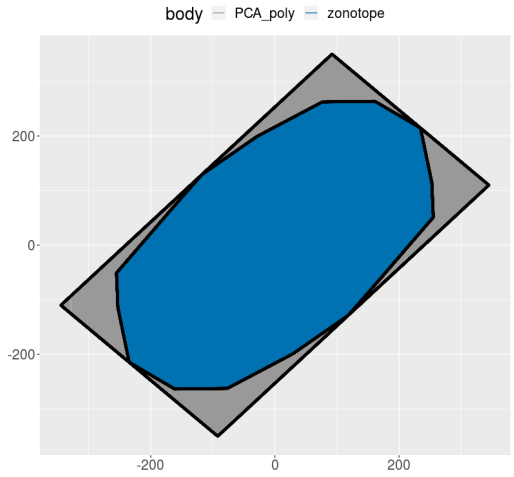

Figure 7: Blue color represents a 2D Z-polytope. Grey color represents the over-approximation of P computed with PCA method.

```
0.4773961  0.001
```

Zonotope volumes in decision and control

Volume approximation for Z-polytopes (or zonotopes) could be very useful in several applications in decision and control (Kopetzki et al., 2017), in autonomous driving (Althoff and Dolan, 2014), or human-robot collaboration (Pereira and Althoff, 2015). The complexity of algorithms that manipulate Z-polytopes strongly depends on their order. Thus, to achieve efficient computations, the common approach in practice is to over-approximate the Z-polytope at hand P, as tight as possible, with a second Z-polytope P_{red} of a smaller order. Then, the ratio of fitness $\rho = (\text{vol}(P_{red})/\text{vol}(P))^{1/d}$ is a good measure for the quality of the approximation. However, this ratio cannot be computed for dimensions typically larger than 10 (see (Kopetzki et al., 2017)). **volesti** is the first software to the best of our knowledge that efficiently approximates the ratio of fitness of a high dimensional Z-polytope–typically up to 100 and order 200–or a Z-polytope of very high order in lower dimensions–e.g., order 1500 in 10 dimensions.

As an illustration, the following R script generates a random 2D zonotope, computes the over-approximation with the PCA method, and estimates the ratio of fitness. The `sample_points` function is then used to plot the two polygons (Figure 7).

```
R> Z = gen_rand_zonotope(2, 8, generator = list("distribution" = "uniform",
                         "seed" = 1729))
R> points1 = sample_points(Z, random_walk = list("walk" = "BRDHR"), n = 10000)
R> retList = zonotope_approximation(Z = Z, fit_ratio = TRUE,
                                    generator = list("seed" = 5))
R> P = retList$P
R> cat(retList$fit_ratio)

1.116799539

R> points2 = sample_points(P, random_walk = list("walk" = "BRDHR", "seed" = 5),
                           n = 10000)
```

High-dimensional integration

Computing the integral of a function over a convex set (i.e., convex polytope) is a hard fundamental problem with numerous applications. **volesti** can be used to approximate the value of such an integral by a simple MCMC integration method, which employs the vol(P) and a uniform sample in P. In particular, let

$$I = \int_P f(x)dx. \tag{3}$$

dimension	*Exact value*	*Estimated value*	*Rel. error*	*Exact Time (sec)*	*Est. Time (sec)*
5	0.02738404	0.02446581	0.1065667	0.023	3.983
10	3.224286e-06	3.204522e-06	0.00612976	3.562	11.95
15	4.504834e-11	4.867341e-11	0.08047068	471.479	33.256
20	-	1.140189e-16	-	-	64.058

Table 3: We compute the integral of the function in Equation (5) over a random generated V-polytope. *Exact value*: the exact value of the integral using **SimplicialCubature** and **geometry**; *Estimated value*: the estimation of the integral with **volesti**; *Rel. error*: the relative error of **volesti**; *Exact Time*: the sum of run-times of **geometry** and **SimplicialCubature**; *Est. Time*: the run-time of **volesti**; "-" indicates that the program halts.

Then, sample N uniformly distributed points $x_1, \ldots, x_N$ from P and,

$$I \approx \text{vol}(P) \frac{1}{N} \sum_{i=1}^{N} f(x_i). \tag{4}$$

The following R script generates a V-polytope for $d = 5, 10, 15, 20$, and estimates the integral of

$$f(\mathbf{x}) = \sum_{i=1}^{n} x_i + 2x_1^2 + x_2 + x_3, \tag{5}$$

over the generated V-polytope P.

Considering the efficiency of **volesti**, Table 3 reports the exact value of I computed by **SimplicialCubature** (Nolan et al., 2016). It computes multivariate integrals over simplices. Hence, to compute an integral of a function over a convex polytope P in R, one should compute the Delaunay triangulation with package **geometry** and then use the package **SimplicialCubature** to sum the values of all the integrals over the simplices computed by the triangulation. The pattern is similar to volume computation. For $d = 5, 10$ the exact computation is faster than the approximate. For $d = 15$, **volesti** is 13 times faster. For $d = 20$, the exact approach halts, while **volesti** returns an estimation in less than a minute.

```
R> num_of_points = 5000
R> f = function(x) { sum(x^2) + (2 * x[1]^2 + x[2] + x[3]) }
R> for (d in seq(from = 5, to = 20, by = 5)) {
     P = gen_rand_vpoly(d, 2 * d, generator = list("seed" = 127))

     points = sample_points(P, random_walk = list("walk" = "BiW",
                            "walk_length" = 1, "seed" = 5), n = num_of_points)
     sum_f = 0
     for (i in seq_len(num_of_points)){
       sum_f = sum_f + f(points[, i])
     }
     V = volume(P, settings = list("error" = 0.05, "seed" = 5))
     I2 = (sum_f * V) / num_of_points
   }
```

Combinatorics and artificial intelligence

We focus now on a different problem, namely, counting the linear extensions of a given partially ordered set (poset), which arises in various applications in artificial intelligence and machine learning, such as partial order plans (Muise et al., 2016) and learning graphical models (Niinimäki et al., 2016).

Let $G = (V, E)$ be an acyclic digraph with $V = [n] := \{1, 2, \ldots, n\}$. One might want to consider G as a representation of the poset $V : i > j$ if and only if there is a directed path from node i to node j. A permutation π of $[n]$ is called a linear extension of G (or the associated poset V) if $\pi^{-1}(i) > \pi^{-1}(j)$ for every edge $i \to j \in E$.

Let $P_{LE}(G)$ be the polytope in $\mathbb{R}^n$ defined by

$$P_{LE}(G) = \{x \in \mathbb{R}^n \mid 1 \geq x_i \geq 0 \text{ for all } i = 1, 2, \ldots, n\},$$

and $x_i \geq x_j$ for all directed edges $i \to j \in E$. It is well known (Stanley, 1986) that the number of linear

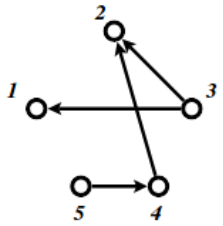

Figure 8: An acyclic directed graph with 5 nodes, 4 edges, and 9 linear extensions.

extensions of G equals the normalized volume of $P_{LE}(G)$, i.e.,

$$\#_{LE}G = \text{vol}(P_{LE}(G))\ n!$$

It is also well known that counting linear extensions is #P-complete (Brightwell and Winkler, 1991). Thus, as the number of graph nodes (i.e., the dimension of $P_{LE}(G)$) grows, the problem becomes intractable for exact methods. Interestingly, **volesti** provides an efficient approximation method that could be added to the ones surveyed by Talvitie et al. (2018).

As a simple example, consider the graph in Figure 8 that has 9 linear extensions[1]. This number can be estimated in milliseconds using **volesti** as in the following script, where the estimated number of linear extensions is 9.014706.

```
R> A = matrix(c(
              -1,0,1,0,0,0,
              -1,1,0,0,0,-1,
              0,1,0,0,0,0,-1,
              1,1,0,0,0,0,0,
              1,0,0,0,0,0,1,
              0,0,0,0,0,1,0,
              0,0,0,0,1,-1,
              0,0,0,0,0,-1,
              0,0,0,0,0,-1,
              0,0,0,0,0,-1,
              0,0,0,0,0,-1),
              ncol = 5, nrow = 14, byrow = TRUE)
R> b = c(0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1 , 1, 1, 1)
R> P_LE = Hpolytope(A = A, b = b)
R> time = system.time({ LE = volume(P_LE, settings = list("error" = 0.01,
                                    "seed" = 1927)) * factorial(5) })
```

Concluding remarks and future work

volesti is an R package that provides MCMC sampling routines for multivariate distributions restricted to convex polytopes and volume estimation. It supports three different polytope representations, and thus, it is useful for several applications. We illustrate the usage of **volesti** with simple, reproducible examples and show how **volesti** can be used to address challenging problems in modern applications.

Regarding future work, the expansion of **volesti** to support general log-concave sampling methods would be of special interest for several applications. Efficient log-concave sampling could also lead to additional sophisticated methods to estimate a multivariate integral over a convex polytope (Lovasz and Vempala, 2006).

Computational details

The results in this paper were obtained using R 3.4.4, R 3.6.3, and **volesti** 1.1.2-2. The versions of the imported by **volesti** packages are **stats** 3.4.4 (R Core Team, 2020b) and **methods** 3.4.4 (R Core Team, 2020a); of the linked by **volesti** packages, **Rcpp** 1.0.3, **BH** 1.69.0.1 (Eddelbuettel et al., 2020a), **RcppEigen** 0.3.3.7.0 (Bates and Eddelbuettel, 2013). The suggested package is **testthat** 2.0.1 (Wickham, 2011). For comparison to **volesti** and for plots, this paper uses **geometry** 0.4.5, **hitandrun** 0.5.5,

[1]Example taken from https://people.inf.ethz.ch/fukudak/lect/pclect/notes2016/expoly_order.pdf

SimplicialCubature 1.2, **Rfast** 2.0.3 (Papadakis et al., 2021), **ggplot2** 3.1.0 (Wickham, 2016), **plotly** 4.8.0 (Sievert, 2020), **rgl** 0.100.50 (Adler et al., 2021), **coda** 0.19.4. All packages used are available from CRAN.

All computations were performed on a PC with `Intel® Pentium(R) CPU G4400 @ 3.30GHz × 2 CPU` and `16GB RAM`.

Acknowledgments

The main part of the work has been done, while A.C. was supported by Google Summer of Code 2018 and 2019 grants, and V.F. was his mentor. The authors acknowledge fruitful discussions with Ioannis Emiris, Ludovic Calès, Elias Tsigaridas, the R-project for statistical computing, and the R community.

Bibliography

D. Adler, D. Murdoch, et al. *rgl: 3D Visualization Using OpenGL*, 2021. URL https://CRAN.R-project.org/package=rgl. R package version 0.105.13. [p656]

J. Albert and S. Chib. Bayesian analysis of binary and polychotomous response data. *Journal of The American Statistical Association - J AMER STATIST ASSN*, 88:669–679, 06 1993. URL https://doi.org/10.1080/01621459.1993.10476321. [p643]

M. Althoff and J. M. Dolan. Online verification of automated road vehicles using reachability analysis. *IEEE Transactions on Robotics*, 30(4):903–918, Aug 2014. ISSN 1552-3098. URL https://doi.org/10.1109/TRO.2014.2312453. [p653]

Y. Altmann, S. McLaughlin, and N. Dobigeon. Sampling from a multivariate gaussian distribution truncated on a simplex: A review. In *2014 IEEE Workshop on Statistical Signal Processing (SSP)*, pages 113–116, 2014. URL https://doi.org/10.1109/SSP.2014.6884588. [p649]

J. M. Azaïs, S. Bercu, J. C. Fort, A. Lagnoux, and P. Lé. Simultaneous confidence bands in curve prediction applied to load curves. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 59(5):889–904, 2010. URL https://doi.org/10.1111/j.1467-9876.2010.00727.x. [p643]

A. Banerjee and C.-H. Hung. Informed momentum trading versus uninformed "naive" investors strategies. *Journal of Banking & Finance*, 35(11):3077–3089, 2011. ISSN 0378-4266. URL https://doi.org/10.1016/j.jbankfin.2011.04.005. [p652]

C. B. Barber, D. P. Dobkin, and H. Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22(4):469–483, Dec. 1996. ISSN 0098-3500. URL https://doi.org/10.1145/235815.235821. [p643]

D. Bates and D. Eddelbuettel. Fast and elegant numerical linear algebra using the RcppEigen package. *Journal of Statistical Software*, 52(5):1–24, 2013. URL http://www.jstatsoft.org/v52/i05/. [p655]

M. Berkelaar, K. Eikland, and P. Notebaert. **lp_solve** 5.5, open source (mixed-integer) linear programming system. Software, May 1 2004. URL http://lpsolve.sourceforge.net/5.5/. [p646]

D. Bolin and F. Lindgren. Excursion and contour uncertainty regions for latent gaussian models. *Journal of the Royal Statistical Society Series B (Statistical Methodology)*, 77:85–106, 01 2015. URL https://doi.org/10.1111/rssb.12055. [p643]

G. Brightwell and P. Winkler. Counting linear extensions. *Order*, 8:225–242, 1991. URL https://doi.org/10.1007/BF00383444. [p655]

L. Calès, A. Chalkis, I. Emiris, and V. Fisikopoulos. Practical Volume Computation of Structured Convex Bodies, and an Application to Modeling Portfolio Dependencies and Financial Crises. In B. Speckmann and C. D. Tóth, editors, *34th International Symposium on Computational Geometry (SoCG 2018)*, volume 99 of *LIPIcs*, pages 19:1–19:15, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL https://doi.org/10.4230/LIPIcs.SoCG.2018.19. [p642, 643, 651, 652]

A. Chalkis and V. Fisikopoulos. Blog post on sampling, integration and volume computations in R, 2021. URL https://geomscale.github.io/Sampling-integration-volume-in-R. [p642, 647, 649, 650]

A. Chalkis, I. Z. Emiris, and V. Fisikopoulos. Practical volume estimation by a new annealing schedule for cooling convex bodies, 2019. URL http://arxiv.org/abs/1905.05494. [p642, 646]

Y. Chen, R. Dwivedi, M. Wainwright, and B. Yu. Fast MCMC sampling algorithms on polytopes. *Journal of Machine Learning Research*, 19(55):1–86, 2018. URL http://jmlr.org/papers/v19/18-158.html. [p642]

C. Chivers. **MHadaptive**: *General Markov Chain Monte Carlo for Bayesian Inference using adaptive Metropolis-Hastings sampling*, 2012. R package version 1.1-8. [p643]

B. Cousins and S. Vempala. A practical volume algorithm. *Mathematical Programming Computation*, 8 (2), Jun 2016. URL https://doi.org/10.1007/s12532-015-0097-z. [p642]

K. V. den Meersche, K. Soetaert, and D. V. Oevelen. xsample(): An r function for sampling linear inverse problems. *Journal of Statistical Software, Code Snippets*, 30(1):1–15, 2009. ISSN 1548-7660. doi: 10.18637/jss.v030.c01. URL https://www.jstatsoft.org/v030/c01. [p643]

V. den Meersche K., S. K., and van Oevelen D. **limSolve**: *Solving Linear Inverse Models*, 2009. URL https://CRAN.R-project.org/package=limSolve. R package version 1.5.1. [p643]

M. Duca, A. Koban, M. Basten, E. Bengtsson, B. Klaus, P. Kusmierczyk, J. Lang, C. Detken, and T. Peltonen. A new Database for Financial Crises in European Countries. Occasional Paper Series 194, European Central Bank, July 2017. URL https://ideas.repec.org/p/ecb/ecbops/2017194.html. [p652]

M. Dyer and A. Frieze. On the complexity of computing the volume of a polyhedron. *SIAM Journal on Computing*, 17(5):967–974, 1988. URL https://doi.org/10.1137/0217060. [p642]

D. Eddelbuettel, J. W. Emerson, and M. J. Kane. *BH: Boost C++ Header Files*, 2020a. URL https://CRAN.R-project.org/package=BH. R package version 1.72.0-3. [p655]

D. Eddelbuettel, R. Francois, J. Allaire, K. Ushey, Q. Kou, N. Russell, D. Bates, and J. Chambers. **Rcpp**: *Seamless R and C++ Integration*, 2020b. URL https://CRAN.R-project.org/package=Rcpp. R package version 1.0.5. [p646]

I. Emiris and V. Fisikopoulos. Practical polytope volume approximation. *ACM Transactions of Mathematical Software, 2018*, 44(4):38:1–38:21, 2014. ISSN 0098-3500. URL https://doi.org/10.1145/3194656. [p642, 646]

V. Fisikopoulos, A. Chalkis, and contributors in file inst/AUTHORS. *volesti: Volume Approximation and Sampling of Convex Polytopes*, 2020. R package version 1.1.2. [p642]

A. Genz and F. Bretz. *Computation of Multivariate Normal and t Probabilities*. Springer Publishing Company, Incorporated, 1st edition, 2009. ISBN 364201688X, 9783642016882. [p642]

C. Geyer. Introduction to markov chain monte carlo. *Handbook of markov chain monte carlo*, 20116022:45, 2011. [p647, 651]

C. J. Geyer and L. T. Johnson. **mcmc**: *Markov Chain Monte Carlo*, 2020. R package version 0.9-7. [p643]

B. Grün and K. Hornik. Modelling human immunodeficiency virus ribonucleic acid levels with finite mixtures for censored longitudinal data. *Journal of the Royal Statistical Society. Series C, Applied statistics*, 61:201–218, 03 2012. URL https://doi.org/10.1111/j.1467-9876.2011.01007.x. [p643]

D. Guegan, L. Calès, and M. Billio. A Cross-Sectional Score for the Relative Performance of an Allocation. Université Paris 1 Panthéon-Sorbonne (Post-Print and Working Papers) halshs-00646070, HAL, 2011. URL https://ideas.repec.org/p/hal/cesptp/halshs-00646070.html. [p652]

G. Guennebaud, B. Jacob, et al. *Eigen v3*, 2010. URL http://eigen.tuxfamily.org. [p646]

W. Hallerbach, C. Hundack, I. Pouchkarev, and J. Spronk. A broadband vision of the development of the dax over time. Technical Report ERS-2002-87-F&A, Erasmus University Rotterdam, 2002. [p643]

H. Haraldsdòttir, B. Cousins, I. Thiele, R. Fleming, and S. Vempala. CHRR: Coordinate Hit-and-Run with Rounding for Uniform Sampling of Constraint-Based Models. *Bioinformatics*, 33(11):1741–1743, 01 2017. URL https://doi.org/10.1093/bioinformatics/btx052. [p643, 646]

W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970. ISSN 00063444. URL http://www.jstor.org/stable/2334940. [p642, 645]

D. W. Heck. **multinomineq**: *Bayesian Inference for Multinomial Models with Inequality Constraints*, 2019. R package version 0.2.1. [p643]

R. Huser and A. Davison. Composite likelihood estimation for the brown-resnick process. *Biometrika*, 2, 06 2013. URL https://doi.org/10.1093/biomet/ass089. [p643]

S. Iyengar. Evaluation of normal probabilities of symmetric regions. *SIAM Journal on Scientific and Statistical Computing*, 9(3):418–423, 1988. URL https://doi.org/10.1137/0909028. [p642]

M. C. Jensen. The Performance of Mutual Funds in the Period 1945-1964. SSRN Scholarly Paper ID 244153, Social Science Research Network, Rochester, NY, May 1967. URL https://papers.ssrn.com/abstract=244153. [p652]

A. Kopetzki, B. Schürmann, and M. Althoff. Methods for order reduction of zonotopes. In *IEEE Conference on Decision and Control*, pages 5626–5633, 2017. URL https://doi.org/10.1109/CDC.2017.8264508. [p642, 653]

A. Laddha and S. Vempala. Convergence of Gibbs Sampling: Coordinate Hit-and-Run Mixes Fast, 2020. [p645]

Y. Lee and S. Vempala. Eldan's stochastic localization and the KLS hyperplane conjecture: An improved lower bound for expansion. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 998–1007, 2017. URL https://doi.org/10.1109/FOCS.2017.96. [p645, 646]

Y. Lee and S. Vempala. Convergence rate of Riemannian Hamiltonian Monte Carlo and faster polytope volume computation. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2018, pages 1115–1121, 2018. ISBN 978-1-4503-5559-9. URL https://doi.org/10.1145/3188745.3188774. [p642]

A. F. Lopez. **lineqGPR**: *Gaussian Process Regression Models with Linear Inequality Constraints*, 2019. R package version 0.1.1. [p643]

L. Lovász and S. Vempala. Simulated annealing in convex bodies and an $O^*(n^4)$ volume algorithm. *Journal of Computer and System Sciences*, 72:392–417, 2006. URL https://doi.org/10.1016/j.jcss.2005.08.004. [p646]

L. Lovász and S. Vempala. Hit-and-run from a corner. *SIAM Journal on Computing*, 35(4):985–1005, 2006. ISSN 0097-5397. URL https://doi.org/10.1137/S009753970544727X. [p642, 645]

L. Lovasz and S. Vempala. Fast algorithms for logconcave functions: Sampling, rounding, integration and optimization. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 57–68, 2006. doi: 10.1109/FOCS.2006.28. [p655]

L. Lovász, R. Kannan, and M. Simonovits. Random walks and an $o^*(n^5)$ volume algorithm for convex bodies. *Random Structures and Algorithms*, 11:1–50, 1997. URL https://doi.org/10.1002/(SICI)1098-2418(199708)11:1<1::AID-RSA1>3.0.CO;2-X. [p646]

T. F. R. Ma, S. K. Ghosh, and Y. Li. **tmvmixnorm**: *Sampling from Truncated Multivariate Normal and t Distributions*, 2020. R package version 1.1.1. [p643]

O. Mangoubi and N. K. Vishnoi. Faster polytope rounding, sampling, and volume computation via a sub-linear ball walk. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1338–1357, 2019. URL https://doi.org/10.1109/FOCS.2019.00082. [p642]

J. Maurer and S. Watanabe. Boost random number library. Software, 2017. URL https://www.boost.org/doc/libs/1_73_0/doc/html/boost_random.html. [p646]

R. D. Morey. **HybridMC**: *Implementation of the Hybrid Monte Carlo and Multipoint Hybrid Monte Carlo sampling techniques*, 2009. R package version 0.2. [p643]

C. Muise, J. Beck, and S. McIlraith. Optimal partial-order plan relaxation via maxsat. *Journal of Artificial Intelligence Research*, 57:113–149, 09 2016. URL https://doi.org/10.1613/jair.5128. [p654]

T. Niinimäki, P. Parviainen, and M. Koivisto. Structure discovery in bayesian networks by sampling partial orders. *Journal of Machine Learning Research*, 17(57):1–47, 2016. URL http://jmlr.org/papers/v17/15-140.html. [p654]

J. P. Nolan, with parts adapted from Fortran, and matlab code by Alan Genz. **SimplicialCubature**: *Integration of Functions Over Simplices*, 2016. URL https://CRAN.R-project.org/package=SimplicialCubature. R package version 1.2. [p654]

A. Pakman. **tmg**: *Truncated Multivariate Gaussian Sampling*, 2015. R package version 0.3. [p643]

M. Papadakis, M. Tsagris, M. Dimitriadis, S. Fafalios, I. Tsamardinos, M. Fasiolo, G. Borboudakis, J. Burkardt, C. Zou, K. Lakiotaki, and C. Chatzipantsiou. *Rfast: A Collection of Efficient and Extremely Fast R Functions*, 2021. URL https://CRAN.R-project.org/package=Rfast. R package version 2.0.3. [p656]

A. Pereira and M. Althoff. Safety control of robots under computed torque control using reachable sets. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 331–338, May 2015. URL https://doi.org/10.1109/ICRA.2015.7139020. [p653]

M. Plummer, N. Best, K. Cowles, K. Vines, D. Sarkar, D. Bates, R. Almond, and A. Magnusson. **coda**: *Output Analysis and Diagnostics for MCMC*, 2020. URL https://CRAN.R-project.org/package=coda. R package version 0.19-4. [p647]

B. Polyak and E. Gryazina. Billiard walk - a new sampling algorithm for control and optimization. *IFAC Proceedings Volumes*, 47(3):6123 – 6128, 2014. ISSN 1474-6670. URL https://doi.org/10.3182/20140824-6-ZA-1003.02312. 19th IFAC World Congress. [p642, 645, 646]

I. Pouchkarev. *Performance Evaluation of Constrained Portfolios*. PhD thesis, Erasmus Research Institute of Management, The Netherlands, 2005. [p652]

I. Pouchkarev, J. Spronk, and J. Trinidad. Dynamics of the spanish stock market through a broadband view of the IBEX 35 index. *Estudios Econom. Aplicada*, 22(1):7–21, 2004. [p643]

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2020a. URL https://www.R-project.org/. [p655]

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2020b. URL https://www.R-project.org/. [p655]

J.-R. Roussel, C. B. Barber, K. Habel, R. Grasman, R. B. Gramacy, P. Mozharovskyi, and D. C. Sterratt. **geometry**: *Mesh Generation and Surface Tessellation*, 2019. R package version 0.4.5. [p643]

V. Sartório. **rhmc**: *Hamiltonian Monte Carlo*, 2018. R package version 1.0.0. [p643]

J. Schellenberger and B. Palsson. Use of randomized sampling for analysis of metabolic networks. *The Journal of biological Chemistry*, 284 9:5457–61, 2009. URL https://doi.org/10.1074/jbc.R800048200. [p642]

W. F. Sharpe. Mutual Fund Performance. *The Journal of Business*, 39(1):119–138, 1966. ISSN 0021-9398. URL https://www.jstor.org/stable/2351741. [p652]

C. Sievert. *Interactive Web-Based Data Visualization with R, plotly, and shiny*. Chapman and Hall/CRC, 2020. ISBN 9781138331457. URL https://plotly-r.com. [p656]

R. L. Smith. Efficient monte carlo procedures for generating points uniformly distributed over bounded regions. *Operations Research*, 32(6):1296–1308, 1984. ISSN 0030364X, 15265463. URL http://www.jstor.org/stable/170949. [p642, 644]

P. Somerville. Numerical computation of multivariate normal and multivariate-t probabilities over convex regions. *Journal of Computational and Graphical Statistics*, 7(4):529–544, 1998. URL https://doi.org/10.1080/10618600.1998.10474793. [p642]

R. P. Stanley. Two poset polytopes. *Discrete & Computational Geometry*, 1(1):9–23, Mar. 1986. ISSN 1432-0444. URL https://doi.org/10.1007/BF02187680. [p654]

T. Talvitie, K. Kangas, T. Niinimäki, and M. Koivisto. Counting linear extensions in practice: Mcmc versus exponential monte carlo. In *AAAI Conference on Artificial Intelligence*, 2018. URL https://aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16957. [p655]

J. Taylor and Y. Benjamini. **restrictedMVN**: *Multivariate Normal Restricted by Affine Constraints*, 2016. R package version 1.0. [p643]

J. L. Treynor. How to Rate Management of Investment Funds. In *Treynor on Institutional Investing*, pages 69–87. John Wiley & Sons, Ltd, 2015. ISBN 978-1-119-19667-9. URL https://doi.org/10.1002/9781119196679.ch10. [p652]

G. van Valkenhoef and T. Tervonen. **hitandrun**: *"Hit and Run" and "Shake and Bake" for Sampling Uniformly from Convex Shapes*, 2019. URL https://CRAN.R-project.org/package=hitandrun. R package version 0.5-5. [p643]

G. Varsi. The multidimensional content of the frustum of the simplex. *Pacific Journal of Mathematics*, 46 (1):303–314, 1973. URL https://projecteuclid.org:443/euclid.pjm/1102946623. [p652]

A. Venzke, D. Molzahn, and S. Chatzivasileiadis. Efficient creation of datasets for data-driven power system applications. *Electric Power Systems Research*, 190:106614, 2021. ISSN 0378-7796. URL https://doi.org/10.1016/j.epsr.2020.106614. [p642]

J. Wadsworth and J. Tawn. Efficient inference for spatial extreme value processes associated to log-gaussian random functions. *Biometrika*, 1, 03 2014. URL https://doi.org/10.1093/biomet/ast042. [p643]

H. Wickham. testthat: Get started with testing. *The R Journal*, 3:5–10, 2011. URL https://journal.r-project.org/archive/2011-1/RJournal_2011-1_Wickham.pdf. [p655]

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. ISBN 978-3-319-24277-4. URL https://ggplot2.tidyverse.org. [p656]

G. Ziegler. *Lectures on Polytopes*. Springer-Verlag, New York, 1995. URL https://doi.org/10.1007/978-1-4613-8431-1. [p644]

Apostolos Chalkis
Department of Informatics & Telecommunications
National & Kapodistrian University of Athens
Greece
GeomScale org.
ORCiD: 0000-0002-4628-1907
achalkis@di.uoa.gr

Vissarion Fisikopoulos
Department of Informatics & Telecommunications
National & Kapodistrian University of Athens
Greece
GeomScale org.
ORCiD: 0000-0002-0780-666X
vfisikop@di.uoa.gr