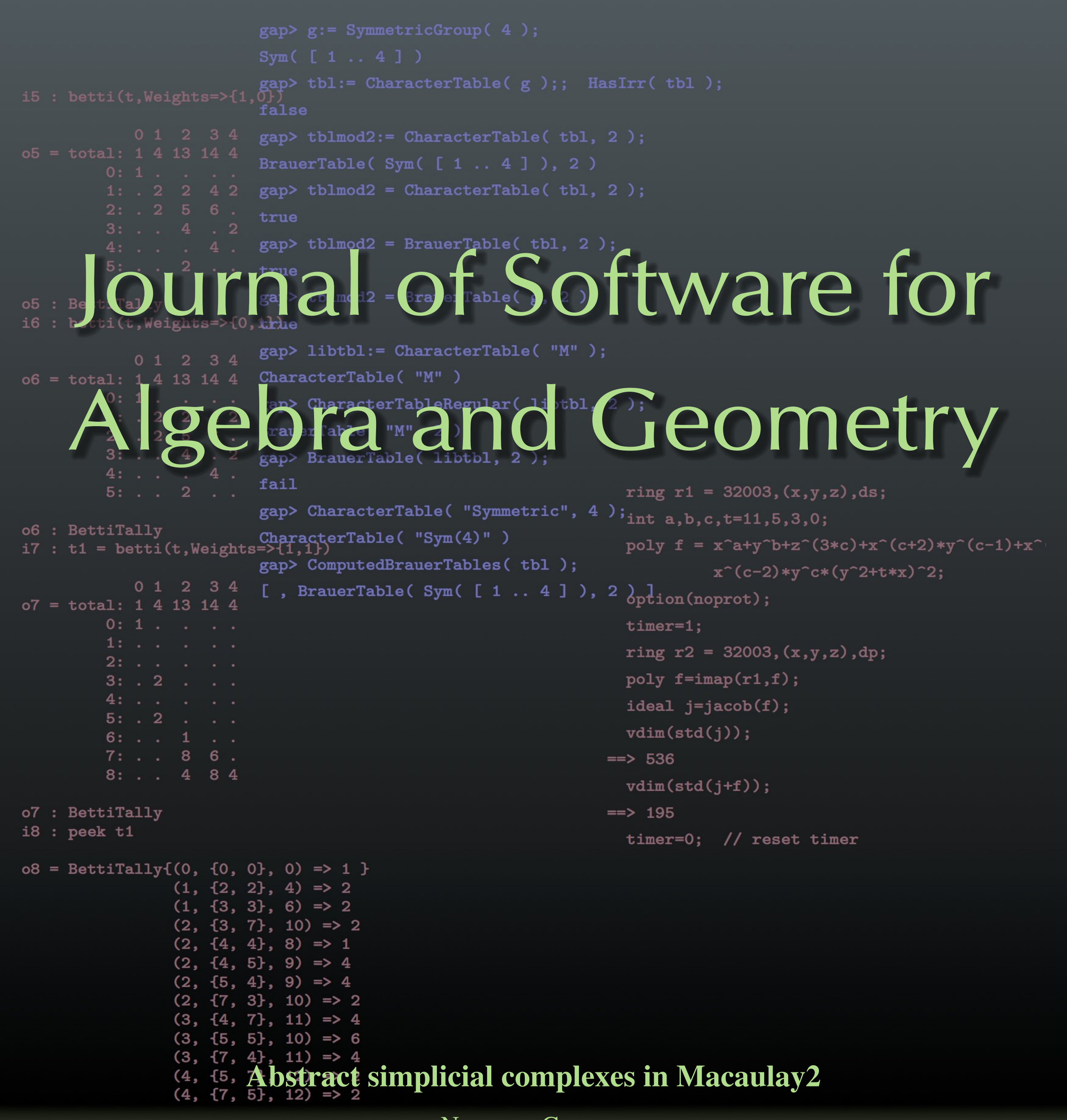

Journal of Software for Algebra and Geometry

Abstract simplicial complexes in Macaulay2

Nathan Grieve

vol 15 2025

JSAG **15** (2025), 29–39
https://doi.org/10.2140/jsag.2025.15.29

The Journal of Software for
Algebra and Geometry

Abstract simplicial complexes in Macaulay2

NATHAN GRIEVE

ABSTRACT: `AbstractSimplicialComplexes.m2` is a package written for Macaulay2. It provides new infrastructure to work with abstract simplicial complexes and related homological constructions. Its key novel feature is to implement each given abstract simplicial complex as a certain graded list in the form of a hash table with integer keys. Among other features, this allows for a direct implementation of the associated reduced and nonreduced simplicial chain complexes. Further, it facilitates construction of random simplicial complexes. The approach that we employ here builds on Stillman and Smith's Macaulay2 package `Complexes.m2`; It complements and is entirely different from the simplicial complexes framework made possible by the Macaulay2 package `SimplicialComplexes.m2` of Smith, Hersey and Zotine.

1. INTRODUCTION.

`AbstractSimplicialComplexes.m2` is a package for Macaulay2 (version 1.24.11). It provides new infrastructure to work with abstract simplicial complexes. Its key novel feature is to implement each given abstract simplicial complex as a graded list (in the form of a hash table with integer keys). This approach is entirely different from the one used in Smith, Hersey and Zotine's Macaulay2 package `SimplicialComplexes.m2` [8; 13].

Our approach, which builds on Stillman and Smith's package `Complexes.m2` [12], allows for a direct implementation of simplicial and reduced simplicial chain complexes. Another feature of independent interest is a method that outputs the induced chain complex maps arising via subsimplicial complexes. Further we provide methods for producing random simplicial complexes and, which also allows for calculation of random simplicial homology complexes. (See Section 4.)

To place matters into perspective, while the existing implementation of simplicial complexes within `SimplicialComplexes.m2` is well suited for simplicial complex commutative algebra questions that arise via the celebrated *Stanley–Reisner correspondence*, a key limitation occurs when working with simplicial complexes and related homological questions. Indeed, when working within `SimplicialComplexes.m2` there is the need to introduce a suitable polynomial ring and then to regard the vertices of each given simplicial complex as variables of that polynomial ring.

By contrast, `AbstractSimplicialComplexes.m2` implements a framework for simplicial complexes

The author thanks the Natural Sciences and Engineering Research Council of Canada for their support through his grants DGECR-2021-00218 and RGPIN-2021-03821.

It is the author's pleasure to thank colleagues for their interest and discussions on related topics. The author also thanks Mike Stillman and Greg Smith for helpful feedback on this work and for correspondence about their package [12]. The author especially thanks an anonymous referee for a careful reading, testing of the code and thoughtful suggestions.

MSC2020: primary 05E45; secondary 55U10, 62R40, 19-04.

Keywords: simplicial complexes, simplicial chain complexes, random simplicial complexes, topological data analysis.

and related homological questions that is based on graded lists as a primitive data type and does not need to introduce auxiliary constructs such as polynomial rings. (See [4] or [10] for the Stanley–Reisner correspondence and [1; 7; 11] for the topological theory of abstract simplicial complexes.)

The advantages and drawbacks to such design choice considerations — depending on the applications that a user has in mind — have been well known to Macaulay2 developers for some time. For example, they were highlighted early in the development of the Macaulay2 package `SpectralSequences.m2` [2].

Indeed, the primary impetus for what we do here arose after revisiting the design choice decisions that were made in creating the package `SpectralSequences.m2`. The aim was to extend its functionality and to give a good forward compatible integration with the package `Complexes.m2`. An overview of the main features of the package `SpectralSequences.m2` is provided in [3]. Such decision choice considerations for the package `AbstractSimplicialComplexes.m2`, which are compatible with the package `Complexes.m2`, are a key novel feature to our present work here.

There are also pedagogical reasons for being able to interact with computational tools for understanding homological questions related to simplicial complexes that are independent of the correspondence of Stanley and Reisner. This is illustrated in the next example.

Example 1.1 (calculating the homology groups of $\mathbb{P}^2_{\mathbb{R}}$ and the Klein bottle). The following code illustrates the calculation of the nonreduced simplicial homology groups of $\mathbb{P}^2_{\mathbb{R}}$ and the Klein bottle, using simplicial complex realizations found, for instance, in Armstrong's *Basic Topology* [1] (see `i2` and `i4`). Related calculations are illustrated in [2] within the package `SpectralSequences.m2`, using realizations compatible with ours, and in [13] within the package `SimplicialComplexes.m2`, using realizations based on [11].

We refrain from producing such comparable calculations here; instead we focus on illustrating how to perform the calculations using the package `AbstractSimplicialComplexes.m2`.

```
M2-host % M2
Macaulay2, version 1.24.11
with packages: ConwayPolynomials, Elimination, IntegralClosure, InverseSystems,
Isomorphism, LLLBases, MinimalPrimes, OnlineLookup,
PrimaryDecomposition, ReesAlgebra, Saturation, TangentCone, Truncations, Varieties

i1 : needsPackage"AbstractSimplicialComplexes";

i2 : -- A simplicial complex realization of PP^2_RR
     K = abstractSimplicialComplex {{1,2,3},{2,3,4},{3,4,5}, {1,5,4},
     {5,2,1},{5,6,2},{4,6,2},
     {1,6,4},{3,6,5},{1,6,3}};

i3 : prune HH simplicialChainComplex K

       1
o3 = ZZ  <-- cokernel | 2 |
     0       1

o3 : Complex

i4 : --  A simplicial complex realization of the Klein bottle
     L = abstractSimplicialComplex {{1,2,7}, {7,8,2}, {4,7,8},
     {4,8,5}, {1,4,5}, {1,5,2}, {2,8,3}, {8,3,9},
     {5,8,9}, {5,9,6}, {2,5,6},  {2,6,3}, {3,9,1}, {9,1,4},
     {6,9,4},{6,7,4}, {3,6,7}, {3,7,1}};

i5 : prune HH simplicialChainComplex L
```

```
          1
o5 = ZZ  <-- cokernel | 2 |
                      | 0 |
     0       1
o5 : Complex
```

The design methodology utilized here, to implement a framework for abstract simplicial complexes within `AbstractSimplicialComplexes.m2`, is based on graded lists. Compared with the approach that is employed in the package `SimplicialComplexes.m2`, the one adopted there has a number of merits.

As an example, inputing the data of a simplicial complex within `AbstractSimplicialComplexes.m2` is less cumbersome than that of `SimplicialComplexes.m2`. Working within the new package, there is no need to introduce a polynomial ring and then represent faces as products of the relevant variables. Instead faces are defined directly as subsets of $[n] := \{1, \ldots, n\}$. Further, the associated simplicial chain complexes (both reduced and nonreduced) are defined over $\mathbb{Z}$ by default. By extension of scalars one obtains the reduced and nonreduced chain complexes with coefficients in other rings.

As another example, our approach facilitates a direct construction of random simplicial complexes. This is an additional novel aspect to the present work which is of an independent interest. It builds, for example, on the approach from [9]. Such methods to construct random simplicial complexes are currently not implemented, directly, within the package `SimplicialComplexes.m2`. However, complementary constructions of random monomial ideals are implemented within the package `RandomIdeals.m2`. In either case, our approach here provides a new point of departure for such themes.

A comparison of computational benchmarks between the packages `AbstractSimplicialComplexes.m2` and `SimplicialComplexes.m2` suggests that their computable thresholds are comparable (depending on the application). Also, it is expected that the package `AbstractSimplicialComplexes.m2` will see enhanced computational efficiencies once the main features of the package `Complexes.m2`, [12], on which the present package `AbstractSimplicialComplexes.m2` builds upon, become part of the core Macaulay2 infrastructure.

Example 1.2 below helps give a sense for the most basic computational comparable data between `AbstractSimplicialComplexes.m2` and `SimplicialComplexes.m2`. In considering this example, note that the methods used in `SimplicialComplexes.m2` to construct the maps in the simplicial chain complexes that it produces ultimately rely on methods that are not internal to the package itself. Rather they rely on methods that are internal to the core Macaulay2 infrastructure. In doing so, some added efficiency, in terms of construction of simplicial chain complexes, is achieved.

By contrast, in `AbstractSimplicialComplexes.m2` the simplicial chain complexes are constructed directly within the package itself. Aside from these computational considerations, in terms of construction of simplicial chain complexes, Example 1.2 suggests that one should expect there to be no major computational efficiency differences in terms of calculating the homology of such chain complexes and similar kinds of homological constructions.

Example 1.2 (calculations on the 12-simplex). In the following runs, the homology of the 12-simplex is calculated using the methods of the package `AbstractSimplicialComplexes.m2` (run **ASC**) and

SimplicialComplexes.m2 (run **SC**). The header with the system prompt, the version number and the packages loaded is the same as on the previous page and will not be repeated.

Run **ASC**:

```
i1 : needsPackage"AbstractSimplicialComplexes";
i2 : time K = abstractSimplicialComplex(12);
 -- used 0.038916s (cpu); 0.0389109s (thread); 0s (gc)
i3 : time k = reducedSimplicialChainComplex(K);
 -- used 9.78308s (cpu); 7.7467s (thread); 0s (gc)
i4 : time h = HH k;
 -- used 38.1503s (cpu); 21.9464s (thread); 0s (gc)
i5 : time apply(13,i-> i => rank(h_i));
 -- used 102.599s (cpu); 50.6054s (thread); 0s (gc)
```

Run **SC**:

```
i1 : needsPackage"SimplicialComplexes";
i2 : R = ZZ[x_1..x_12];
i3 : K = simplexComplex(11,R);
i4 : time k = complex K;
 -- used 0.104141s (cpu); 0.104134s (thread); 0s (gc)
i5 : time h = HH k;
 -- used 40.9266s (cpu); 22.6574s (thread); 0s (gc)
i6 : time apply(13,i-> i => rank(h_i));
 -- used 90.7713s (cpu); 47.5847s (thread); 0s (gc)
```

The conclusion is that, aside from apparent differences in creating the simplicial chain complexes themselves (i3 in run **ASC** and i4 in run **SC**), the computational thresholds are similar (i4, i5 in run **ASC** and i5, i6 in run **SC**). This is not surprising.

An additional feature of the package AbstractSimplicialComplexes.m2 is that it can produce the induced chain complex maps that arise from nested pairs of simplicial complexes with one simplicial complex a subsimplicial complex of another. The relevant methods are inducedSimplicialChainComplexMap and reducedInducedSimplicialChainComplexMap.

As one possible direction for ongoing future development, such methods may find applications towards creating an interface for the current package AbstractSimplicialComplexes.m2 to integrate with the package SpectralSequences.m2. Such an interface would also provide a simplification of the main constructors that are currently used within the package SpectralSequences.m2 to create filtered chain complexes arising from filtered simplicial complexes.

Another possible future development would expand the package's infrastructure in the direction of topological data analysis (see for instance [5] and [6]). The design considerations that we have employed here will facilitate with this. A first example in this direction is illustrated in Example 4.2. Further, the question of homological features for models of random simplicial complexes is quite attractive. In Example 4.1 we indicate one such model which arises in the work [9] and the references therein.

Organization. In Section 3 we give a brief overview of the `AbstractSimplicialComplexes.m2` package and its syntax. In Section 4 we discuss selected illustrative examples that pertain to homological calculations with random simplicial complexes. In Section 2 we provide a brief discussion of the relevant mathematical concepts that underlie what we do here. It also serves to fix the conventions and notation that are used implicitly within the package itself.

2. Mathematical preliminaries.

2.1. ***Abstract simplicial complexes.*** For our purposes, by an *abstract simplicial complex* K on the vertex set

$$[n] := \{1, \dots, n\}$$

we mean a collection of subsets

$$\sigma \subseteq [n]$$

that is closed under taking subsets. Thus,

$$\tau \subseteq \sigma \subseteq [n] \text{ and } \sigma \in K \quad \implies \quad \tau \in K.$$

2.2. ***Reduced and nonreduced simplicial chain complexes.*** Fixing a commutative ring R, our conventions about reduced and nonreduced simplicial chain complexes with coefficients in R are similar to those which can be found, for example, in [10].

Let K be an abstract simplicial complex on the vertex set $[n]$. Recall that

$$\dim K := \max_{\sigma \in K} \#\sigma - 1.$$

Define K's *reduced simplicial chain complex* with coefficients in R to be the chain complex

$$0 \leftarrow \mathcal{C}_{-1}(K; R) \leftarrow \cdots \leftarrow \mathcal{C}_{i-1}(K; R) \xleftarrow{\partial_i} \mathcal{C}_i(K; R) \leftarrow \dots : \mathcal{C}^{\mathrm{red}}_{\bullet}(K; R);$$

here

$$\mathcal{C}_i(K; R) := \begin{cases} \bigoplus\limits_{\substack{\sigma \in K \\ \#\sigma = i+1}} R\boldsymbol{e}_\sigma & \text{if } -1 \leqslant i \leqslant \dim K, \\ 0 & \text{otherwise,} \end{cases}$$

$$\partial_i(\boldsymbol{e}_\sigma) := \sum_{j=0}^{i} (-1)^j \boldsymbol{e}_{\{\ell_0 < \cdots < \hat{\ell}_j < \cdots < \ell_i\}} \text{ for } 0 \leqslant i \leqslant \dim K,$$

and

$$\sigma = \{\ell_0 < \cdots < \ell_i\} \in K.$$

(The notation $\{\ell_0 < \cdots < \hat{\ell}_j < \cdots < \ell_i\}$ means $\{\ell_0 < \cdots < \ell_i\}$ with ℓ_j removed.)

The *nonreduced simplicial chain complex* with coefficients in R is defined to be the chain complex

$$0 \leftarrow \mathcal{C}_0(K;\, R) \leftarrow \cdots \leftarrow \mathcal{C}_{i-1}(K;\, R) \xleftarrow{\partial_i} \mathcal{C}_i(K;\, R) \leftarrow \ldots : \mathcal{C}_\bullet(K;\, R).$$

Denote the homology modules of these chain complexes by $H_i^{\mathrm{red}}(K;\, R)$ and $H_i(K;\, R)$ for $i \in \mathbb{Z}$. Thus

$$H_i(K;\, R) = \ker \partial_i / \operatorname{image} \partial_{i+1} \quad \text{for } i \geqslant 0,$$
$$H_i^{\mathrm{red}}(K;\, R) = \ker \partial_i / \operatorname{image} \partial_{i+1} \quad \text{for } i \geqslant -1.$$

3. Brief overview of the package. In the package `AbstractSimplicialComplexes.m2` abstract simplicial complexes are implemented by the type `AbstractSimplicialComplexes`. Users who wish to use the package `AbstractSimplicialComplexes.m2` to study simplicial complexes on vertex sets different from $[n]$ should first fix a suitable order-preserving bijection compatible with the standard lexicographic ordering.

Example 3.1 (constructing abstract simplicial complexes). Simplicial complexes can be constructed within the package `AbstractSimplicialComplexes.m2` using the method `abstractSimplicialComplexes`. As input this method takes either an integer n, in which case the output is the n-simplex, or a collection of subsets of $[n]$, in which case the output is the simplicial complex generated by these subsets.

As another consideration to be aware of, if $m \leqslant n$ and if K is a simplicial complex with vertices supported on $[m]$ then automatically K is considered as a subsimplicial complex of the n-simplex on $[n]$.

Such features are illustrated via the following code. More details about reduced and nonreduced simplicial chain complexes are given in Example 3.2.

```
i1 : needsPackage"AbstractSimplicialComplexes";

i2 : --- make the simplicial complex on [5]
     --- generated by {1,2,3,4},{1,3,4},{2,5},{2,4,5}
     K = abstractSimplicialComplex({{1,2,3,4},{1,3,4},{2,5},{2,4,5}});

i3 : -- vertices
     K_0

o3 = {{1}, {2}, {3}, {4}, {5}}
o3 : List
i4 : -- facets
     abstractSimplicialComplexFacets K

o4 = {{2, 4, 5}, {1, 2, 3, 4}}
o4 : List
i5 : -- make the simplex on [6]
     L = abstractSimplicialComplex(6);

i6 : -- we regard K as a subsimplicial complex of L
     -- the induced simplicial chain complex map is then
     f = inducedSimplicialChainComplexMap(L,K)

            6                      5
o6 = 0 : ZZ  <----------------- ZZ  : 0
               | 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 |
               | 0 0 0 0 0 |
         15                              8
     1 : ZZ   <----------------------- ZZ  : 1
                 | 1 0 0 0 0 0 0 0 |
                 | 0 1 0 0 0 0 0 0 |
                 | 0 0 1 0 0 0 0 0 |
                 | 0 0 0 0 0 0 0 0 |
                 | 0 0 0 0 0 0 0 0 |
                 | 0 0 0 1 0 0 0 0 |
                 | 0 0 0 0 1 0 0 0 |
                 | 0 0 0 0 0 1 0 0 |
                 | 0 0 0 0 0 0 0 0 |
                 | 0 0 0 0 0 0 1 0 |
                 | 0 0 0 0 0 0 0 0 |
                 | 0 0 0 0 0 0 0 0 |
                 | 0 0 0 0 0 0 0 1 |
                 | 0 0 0 0 0 0 0 0 |
                 | 0 0 0 0 0 0 0 0 |
         20                        5
     2 : ZZ   <----------------- ZZ  : 2
                 | 1 0 0 0 0 |
                 | 0 1 0 0 0 |
                 | 0 0 0 0 0 |
                 | 0 0 0 0 0 |
                 | 0 0 1 0 0 |
                 | 0 0 0 0 0 |
                 | 0 0 0 0 0 |
                 | 0 0 0 0 0 |
                 | 0 0 0 0 0 |
                 | 0 0 0 0 0 |
                 | 0 0 0 1 0 |
                 | 0 0 0 0 0 |
                 | 0 0 0 0 0 |
                 | 0 0 0 0 1 |
                 | 0 0 0 0 0 |
```

[last line repeats 5 more times]

```
         15              1
     3 : ZZ   <--------- ZZ  : 3
                 | 1 |
                 | 0 |
```

[last line repeats 13 more times]

```
o6 : ComplexMap
i7 : isWellDefined f
o7 = true
```

Example 3.2 (constructing reduced and nonreduced simplicial chain complexes). There are two simplicial chain complex constructors. The following code illustrates the key points.

```
i2 : K = abstractSimplicialComplex({{1,2,3,4}, {2,3,5}, {1,5}});
i3 : k = simplicialChainComplex K
       5       9       5       1
o3 = ZZ  <-- ZZ  <-- ZZ  <-- ZZ
     0       1       2       3
o3 : Complex
```

```
i4 : k.dd

              5                                     9
o4 = 0 : ZZ  <---------------------------------- ZZ  : 1
                | -1 -1 -1 -1 0  0  0  0  0  |
                | 1  0  0  0  -1 -1 -1 0  0  |
                | 0  1  0  0  1  0  0  -1 -1 |
                | 0  0  1  0  0  1  0  1  0  |
                | 0  0  0  1  0  0  1  0  1  |
              9                         5
     1 : ZZ  <---------------------- ZZ  : 2
                | 1  1  0  0  0  |
                | -1 0  1  0  0  |
                | 0  -1 -1 0  0  |
                | 0  0  0  0  0  |
                | 1  0  0  1  1  |
                | 0  1  0  -1 0  |
                | 0  0  0  0  -1 |
                | 0  0  1  1  0  |
                | 0  0  0  0  1  |
              5               1
     2 : ZZ  <---------- ZZ  : 3
                | -1 |
                | 1  |
                | -1 |
                | 1  |
                | 0  |

o4 : ComplexMap

i5 : kRed = reducedSimplicialChainComplex K

       1       5       9       5       1
o5 = ZZ  <-- ZZ  <-- ZZ  <-- ZZ  <-- ZZ
     -1      0       1       2       3

o5 : Complex

i6 : kRed.dd

               1                        5
o6 = -1 : ZZ  <----------------- ZZ  : 0
                 | 1 1 1 1 1 |
              5                                     9
     0 : ZZ  <---------------------------------- ZZ  : 1
                | -1 -1 -1 -1 0  0  0  0  0  |
                | 1  0  0  0  -1 -1 -1 0  0  |
                | 0  1  0  0  1  0  0  -1 -1 |
                | 0  0  1  0  0  1  0  1  0  |
                | 0  0  0  1  0  0  1  0  1  |
              9                         5
     1 : ZZ  <---------------------- ZZ  : 2
                | 1  1  0  0  0  |
                | -1 0  1  0  0  |
                | 0  -1 -1 0  0  |
                | 0  0  0  0  0  |
                | 1  0  0  1  1  |
                | 0  1  0  -1 0  |
                | 0  0  0  0  -1 |
                | 0  0  1  1  0  |
                | 0  0  0  0  1  |
           5                1
     2 : ZZ  <---------- ZZ  : 3
                | -1 |
                | 1  |
                | -1 |
                | 1  |
                | 0  |
o6 : ComplexMap
```

Example 3.3 (constructing chain complex maps induced via subsimplicial complexes.). Given a subsimplicial complex $L \subseteq K$ there is support for computing the induced maps of the respective simplicial and reduced simplicial chain complexes. The following code illustrates the key points.

```
i2 : K = abstractSimplicialComplex({{1,2},{3}})
o2 = AbstractSimplicialComplex{-1 => {{}}          }
                               0 => {{1}, {2}, {3}}
                               1 => {{1, 2}}

o2 : AbstractSimplicialComplex
i3 : J = ambientAbstractSimplicialComplex(K)
o3 = AbstractSimplicialComplex{-1 => {{}}                        }
                               0 => {{1}, {2}, {3}}
                               1 => {{1, 2}, {1, 3}, {2, 3}}
                               2 => {{1, 2, 3}}
o3 : AbstractSimplicialComplex
i4 : f = inducedSimplicialChainComplexMap(J,K)
           3                 3
o4 = 0 : ZZ  <------------- ZZ  : 0
                | 1 0 0 |
                | 0 1 0 |
                | 0 0 1 |
           3               1
     1 : ZZ  <--------- ZZ  : 1
                | 1 |
                | 0 |
                | 0 |
o4 : ComplexMap
i5 : isWellDefined f
o5 = true
```

4. Random simplicial complexes.

Within the package `AbstractSimplicialComplexes.m2` we provide three ways of producing random simplicial complexes.

The first produces a random simplicial complex with support on $[n]$. The second method produces a random simplicial complex with support on $[n]$ and having r-skeleton. The final method produces a random simplicial complex $Y_d(n,m)$ having vertex set $[n]$, complete $(d-1)$ skeleton and exactly m dimension d faces chosen at random from all $\binom{\binom{n}{d+1}}{m}$ possibilities.

Such random simplicial complexes appear in many different contexts; see [9] and the references therein.

Example 4.1 (generating random simplicial complexes). The following code illustrates how to work with these methods using the package `AbstractSimplicialComplexes.m2`.

```
i2 : setRandomSeed(currentTime());
i3 : randomAbstractSimplicialComplex(5)
o3 = AbstractSimplicialComplex{-1 => {{}}     }
                               0 => {{1}, {3}}
                               1 => {{1, 3}}
o3 : AbstractSimplicialComplex
i4 : randomAbstractSimplicialComplex(5,3)
o4 = AbstractSimplicialComplex{-1 => {{}}                  }
                               0 => {{2}, {3}, {4}}
                               1 => {{2, 3}, {2, 4}, {3, 4}}
                               2 => {{2, 3, 4}}
o4 : AbstractSimplicialComplex
i5 : tally(for i from 1 to 10000 list (facets randomAbstractSimplicialComplex(5,3,2)))
o5 = Tally{{{1, 2, 3}, {1, 2, 4}, {2, 4, 5}} => 391 }
           {{1, 2, 3}, {1, 3, 4}, {3, 4, 5}} => 1338
           {{1, 2, 3}, {1, 3, 5}, {2, 3, 4}} => 1293
           {{1, 2, 3}, {1, 4, 5}, {2, 4, 5}} => 403
           {{1, 2, 4}, {1, 2, 5}} => 1710
           {{1, 4, 5}, {2, 4, 5}} => 1774
           {{2, 3, 4}, {2, 3, 5}, {3, 4, 5}} => 1557
           {{2, 3, 4}, {2, 4, 5}, {3, 4, 5}} => 1534
o5 : Tally
```

Example 4.2 (homological calculations on random Vietoris–Rips complexes). Motivated by the perspective of topological data analysis (see [6], for example), we model an n-dimensional *random point cloud distance matrix* as a random $n \times n$ upper triangular matrix with random real entries above the main diagonal. The (i, j)-entry represents the distance $d(x_i, x_j)$ between the points x_i and x_j; the corresponding *Vietoris–Rips complex* with vertex set supported on $[n]$ and depending on a parameter $\epsilon > 0$ is the simplicial complex that has as k-faces those $(k+1)$-element subsets

$$\{i_0 < \cdots < i_k\} \subseteq [n]$$

such that

$$d(x_{i_j}, x_{i_\ell}) \leqslant \epsilon \quad \text{for all } i_j, i_\ell \in \{i_0 < \cdots < i_k\}.$$

See [6, p. 104] and [5, Section 2.2] for details.

```
i2 : -- A random such VR complex on [n] and depending on a given
     -- parameter e can be modelled in the following way.
     randomVRcomplex := (n,e) -> (
     -- return a random VR-complex supported on [n] and depending on parameter e --
         L := for i from 1 to n list i;
         setRandomSeed(currentTime());
         M := fillMatrix(mutableMatrix(RR,n,n), UpperTriangular => true);
         myFaces := select(subsets(L), i-> all(apply(subsets(i,2),
         j-> M_(j#0-1,j#1-1) <= e)));
         K = abstractSimplicialComplex myFaces;
         return K);
i3 : -- For example, the facets of a random VR complex on the vertex set [10]
     -- and depending on a parameter e = .4 is described as
     setRandomSeed(currentTime());
```

```
i4 : K = randomVRcomplex(10,.4);
i5 : facets K
o5 = {{5, 8}, {2, 4, 9}, {3, 4, 6}, {4, 6, 9}, {4, 7, 10}, {6, 8, 9}, {7, 8, 9},
     {1, 4, 7, 9}, {2, 3, 4, 10}, {3, 4, 5, 10}}
o5 : List
i6 : prune HH reducedSimplicialChainComplex K
        1
o6 = ZZ
     1
o6 : Complex
```

Supplement. The online supplement contains version 1.0 of `AbstractSimplicialComplexes.m2`.

References.

[1] M. A. Armstrong, *Basic topology*, Springer, 1983.

[2] D. Berlekamp, A. Boocher, N. Grieve, E. Grifo, G. G. Smith, and T. Vu, "A spectral sequence package for Macaulay2", 2016, available at https://macaulay2.com/doc/Macaulay2/share/doc/Macaulay2/SpectralSequences/html/index.html.

[3] A. Boocher, N. Grieve, and E. Grifo, "The software package `SpectralSequences`", preprint. arXiv 1610.05338

[4] W. Bruns and J. Herzog, *Cohen–Macaulay rings*, Cambridge Studies in Advanced Mathematics **39**, Cambridge University Press, 1993.

[5] G. Carlsson, "Topology and data", *Bull. Amer. Math. Soc.* (*N.S.*) **46**:2 (2009), 255–308.

[6] G. Carlsson and M. Vejdemo-Johansson, *Topological data analysis with applications*, Cambridge University Press, 2022.

[7] A. Hatcher, *Algebraic topology*, Cambridge University Press, 2002.

[8] B. Hersey, G. G. Smith, and A. Zotine, "Simplicial complexes in Macaulay2", *J. Softw. Algebra Geom.* **13**:1 (2023), 53–59.

[9] M. Kahle, F. H. Lutz, A. Newman, and K. Parsons, "Cohen–Lenstra heuristics for torsion in homology of random complexes", *Exp. Math.* **29**:3 (2020), 347–359.

[10] E. Miller and B. Sturmfels, *Combinatorial commutative algebra*, Graduate Texts in Mathematics **227**, Springer, 2005.

[11] J. R. Munkres, *Elements of algebraic topology*, Addison-Wesley, Menlo Park, CA, 1984.

[12] G. G. Smith and M. Stillman, "Complexes: development package for beta testing new version of chain complexes", Macaulay2 package, available at https://github.com/Macaulay2/M2/blob/master/M2/Macaulay2/packages/Complexes.m2.

[13] G. G. Smith, B. Hersey, and A. Zotine, "SimplicialComplexes: exploring abstract simplicial complexes within commutative algebra", Macaulay2 package, available at https://github.com/Macaulay2/M2/blob/master/M2/Macaulay2/packages/SimplicialComplexes.m2.

Received: 30 Sep 2024 Revised: 13 Mar 2025 Accepted: 14 Apr 2025

Nathan Grieve: Department of Mathematics and Statistics, Acadia University, Wolfville, NS, B4P 2R6, Canada

and

School of Mathematics and Statistics, Carleton University, Ottawa, ON, K1S 5B6, Canada

and

Département de mathématiques, Université du Québec à Montréal, Montréal, QC, H2X 3Y7, Canada

and

Department of Pure Mathematics, University of Waterloo, Waterloo, ON, N2L 3G1, Canada

nathan.m.grieve@gmail.com