Journal of Software for Algebra and Geometry

Random Monomial Ideals: a Macaulay2 package

Sonja Petrović, Despina Stasi and Dane Wilburne

vol 9 2019

JSAG **9** (2019), 65–70
dx.doi.org/10.2140/jsag.2019.9.65

The Journal of Software for
Algebra and Geometry

Random Monomial Ideals: a Macaulay2 package

SONJA PETROVIĆ, DESPINA STASI AND DANE WILBURNE

ABSTRACT: The Macaulay2 package `RandomMonomialIdeals.m2` provides users with a set of tools that allow for the systematic generation and study of random monomial ideals. It also introduces new objects, `Sample` and `Model`, to allow for streamlined handling of random objects and their statistics in Macaulay2.

ERDŐS–RÉNYI RANDOM MONOMIAL IDEALS. Given their central role in commutative algebra and their inherent combinatorial structure (see, e.g., [Miller and Sturmfels 2005; Stanley 1996]), monomial ideals are a natural class of object to study probabilistically. This study was initiated in [De Loera et al. 2019], where random monomial ideals were produced from random sets of monomial generators in a manner inspired by the Erdős–Rényi model of random graphs. Working within the framework of such a model allows one to ask well-posed questions about the distributions of various algebraic invariants of interest.

The [Macaulay2] package `RandomMonomialIdeals.m2` implements several basic probabilistic models for monomial ideals based on the work in [De Loera et al. 2019]; it also computes summary statistics of (algebraic properties of) samples of monomial ideals, and sets up the framework to define new probabilistic models and generate samples from them using two new Macaulay2 types, `Sample` and `Model`.

The fundamental method in the package is `randomMonomialSets`, which randomly generates sets of monomials in a fixed number of variables up to a given degree from the Erdős–Rényi-type distribution $\mathcal{B}(n, D, p)$ defined in [De Loera et al. 2019], as well as various other related distributions.

In the following examples, we set the number of variables to `n=3` and sample size to `N=5`.

Each command in Table 1 returns a list of five sets of randomly generated monomials. In the case of $\mathcal{B}(n, D, M)$, if M is larger than the total number of monomials in n variables of degree at most D, all such monomials are returned.

MSC2010: primary 13F20; secondary 05E40.
Keywords: random monomial ideals, random commutative algebra, combinatorial commutative algebra, Macaulay2 package.
`RandomMonomialIdeals.m2` version 1.0

Model	Example of `M2` command
$\mathcal{B}(n, D, p)$: select each monomial of degree $\leq D$ independently with probability $p \in [0, 1]$	`D=3; p=0.2;` `randomMonomialSets(n,D,p,N)`
$\mathcal{B}(n, D, \boldsymbol{p})$: select each monomial of degree $1 \leq d \leq D$ independently with probability $p_d \in [0, 1]$, where $\boldsymbol{p} = (p_1, \ldots, p_D)$ is a list of probabilities	`D=3;` `p = {0.5,0.0,0.1};` `randomMonomialSets(n,D,p,N)`
$\mathcal{B}(n, D, M)$: select a set of M monomials of degree $\leq D$ uniformly at random all among such sets	`D=3; M=2;` `randomMonomialSets(n,D,M,N)`
$\mathcal{B}(n, D, \boldsymbol{M})$: randomly select M_d monomials of each degree $1 \leq d \leq D$	`D=4; M={1,0,3,0};` `randomMonomialSets(n,D,M,N)`

Table 1. Examples of models of random monomial ideals implemented in the `RandomMonomialIdeals.m2` Macaulay2 package (left) and the corresponding M2 commands to produce random instances for each model (right).

One can also force the monomial sets to be minimal generating sets as follows.

```
i1 : n=3; D=4; N=4; p={0.5,0.0,1.0,0.0};
     netList pack(4,randomMonomialSets(n,D,p,N, Strategy=>"Minimal"))

       +-----------+----------------------+----------------------+-----------+
       |          3|      3   2       2   3|      3   2       2   3|           |
o1 =   |{x , x , x }|{x , x , x x , x x , x }|{x , x , x x , x x , x }|{x , x , x }|
       | 1   2   3 | 3   1   1 2   1 2   2 | 1   2   2 3   2 3   3 | 1   2   3 |
       +-----------+----------------------+----------------------+-----------+
```

In the above sample of four sets of monomials, there are no monomials of degrees 2 or 4. Each of the variables was selected with probability 0.5. All monomials of degree 3 were selected, but of course some are not included as they are not minimal generators of the corresponding monomial ideal.

The distributions that we have described above and, in fact, any probability distribution on sets of monomials, naturally induce distributions on monomial ideals. The distribution on ideals induced by $\mathcal{B}(n, D, p)$ is denoted by $\mathcal{I}(n, D, p)$. Samples of monomial ideals for each of the models in Table 1 can be generated as follows:

```
i2 : n=2;D=5;p=0.2;N=3; randomMonomialIdeals(n,D,p,N)

                                            3
o2 = {monomialIdeal(x ), monomialIdeal (x , x x ), monomialIdeal(x x )}
                     2                  1   1 2                   1 2

o2 : List
```

SUMMARY STATISTICS. Given a list of monomial ideals, the package contains an array of methods for computing and summarizing various algebraic invariants of the ideals in the list: Krull dimension, degree, projective dimension, Castelnuovo–Mumford regularity, Betti tables and Betti shapes, and the proportion of ideals which are Borel-fixed or Cohen–Macaulay.

For instance, `dimStats` is a method which computes the Krull dimension of $k[x_1, \dots, x_n]/I$ for each monomial ideal I in the list and returns the mean and standard deviation of the sample. When the optional input `ShowTally` is set to `true`, `dimStats` also returns a histogram of the Krull dimensions of the ideals in the list.

```
i3 : B = randomMonomialIdeals(3,10,0.01,1000);
i4 : dimStats(B, ShowTally=>true)
o4 = (1.92, .46, Tally{0 => 1  })
                       1 => 146
                       2 => 785
                       3 => 68
o4 : Sequence
```

In this sample of $N = 1000$ monomial ideals from the distribution $\mathcal{I}(3, 10, 0.01)$, the proportion of ideals with Krull dimension 0, 1, 2, 3 was 0.001, 0.146, 0.785, 0.068, respectively. By [De Loera et al. 2019, Theorem 3.2], the probability that a monomial ideal from this distribution has Krull dimension t for $t = 0, 1, 2, 3$ is 0.0009, 0.1458, 0.7963, 0.570, respectively. By the same theorem, the expected Krull dimension in this case is 1.9094 whereas the observed sample mean in the example is 1.92.

Each method that computes sample statistics of a particular invariant or property of a list monomial ideals can also be applied more generally to any list of algebraic objects for which that invariant or property is defined, whether or not the objects were generated using the ER-model.

NEW TYPES FOR CREATING AND STORING PROBABILISTIC MODELS AND SAMPLES IN MACAULAY2. The package comes equipped with the predefined Erdős–Rényi-type model. For example, let us consider the graded ER-type model with $\boldsymbol{p} = (0.1, 0, 0.2)$ in four variables and degree bound $D = 3$. We store this in an object of class `Model`:

```
i5 : myModel = ER(ZZ/101[a..d],3,{0.1,0.0,0.2})
o5 = Model{Generate => {*Function[RandomMonomialIdeals.m2:168:22-168:46]*}}
           Name => Erdos-Renyi
                            ZZ
           Parameters => (---[a, b, c, d], 3, {.1, 0, .2})
                           101
o5 : Model
```

It is now easy to obtain a sample of 1000 monomial ideals from this model. Note that model parameters are also stored with the sample object for easy access.

```
i6 : time mySample = sample(myModel,1000);
     -- used 2.32541 seconds
i7 : mySample.ModelName
o7 = Erdos-Renyi
i8 : mySample.Parameters
        ZZ
o8 = (---[a, b, c, d], 3, {.1, 0, .2})
       101
o8 : Sequence
i9 : mySample.SampleSize
o9 = 1000
```

The raw data (i.e., actual sets of monomials without the parameter values etc.) from the sample can be loaded as follows:

```
i10 :  time myIdeals = getData mySample;
     -- used 0.00005 seconds
i11 : myIdeals_0
        2      2     2
o11 = {b d, b*d , c*d }
o11 : List
```

To obtain statistics on any algebraic property of interest for the given sample, one simply runs the `statistics` command on the sample. Let us look at the distribution of Krull dimensions for this sample of 1000 monomial ideals:

```
i12 : time statistics(mySample, dim@@ideal)
     -- used 0.325259 seconds
                        2053
o12 = HashTable{Mean => ----, StdDev=> .654363, Histogram=> Tally{0 => 5  }}
                        1000                                       1 => 166
                                                                   2 => 608
                                                                   3 => 213
                                                                   4 => 8
o12 : HashTable
```

In addition, the command `writeSample(Sample,String)` can be used to write a sample to disk (the string is the filename), a feature that will be useful when large samples are generated and their statistics take a lot of computational time (e.g., Gröbner bases or free resolutions). This command creates a folder in which the model and data are stored. The sample can then be read via calls to the `sample(String)` method; for more details, the user is referred to the package documentation.

The new types, `Model` and `Sample`, along with the `statistics` function, allow one to define a new way to sample random algebraic objects, store the data as a proper statistical sample, and study their algebraic properties under the probabilistic regime. Here is a simple example of a model that generates M polynomials in n variables of degree D randomly using Macaulay2's built-in `random` function:

```
i13: f=(D,n,M)->(R=QQ[x_1..x_n];apply(M,i->random(D,R)))
o13=f
o13: FunctionClosure
i14: myModel = model({2,3,4},f,"rand(D,n,M):
        M random polynomials in n variables of degree D")
o14=Model{Generate=>{*Function[RandomMonomialIdeals.m2:107:22-107:37]*}    }
          Name=>rand(D,n,M): M random polynomials in n variables of degree D
          Parameters=>{2, 3, 4}
o14: Model
i15: mySample = sample(myModel,10);
```

The last line produces a sample of size 10 from `myModel`. One can use the method `statistics` to generate an ideal from each of the ten sets in the sample, compute the Gröbner basis, and report its size:

```
i16 : statistics(mySample, numcols@@gens@@gb@@ideal)
o16 = HashTable{Histogram => Tally{6 => 10}}
                Mean => 6
                StdDev => 0
o16 : HashTable
```

Any function that can be run through `tally` can also serve as input for the `statistics` method; for more extensive examples, the user is directed to the package documentation.

ACKNOWLEDGEMENTS. The following undergraduate students made valuable contributions to the development of the package during summer of 2017: Genevieve Hummel, Parker Joncus, Daniel Kosmas, Richard Osborn, Monica Yun, and Tanner Zielinski. The project is funded by the NSF grant NSF-DMS-1522662, the Illinois Institute of Technology College of Science student summer research stipend, and the McMorris student summer research stipend from the Department of Applied Mathematics at Illinois Institute of Technology.

SUPPLEMENT. Version 1.0 of `RandomMonomialIdeals.m2` is contained in the online supplement.

REFERENCES.

[De Loera et al. 2019] J. A. De Loera, S. Petrović, L. Silverstein, D. Stasi, and D. Wilburne, "Random monomial ideals", *J. Algebra* **519** (2019), 440–473. MR Zbl

[Macaulay2] D. R. Grayson and M. E. Stillman, "Macaulay2, a software system for research in algebraic geometry", available at https://faculty.math.illinois.edu/Macaulay2/.

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

[Stanley 1996] R. P. Stanley, *Combinatorics and commutative algebra*, 2nd ed., Progress in Mathematics **41**, Birkhäuser, Boston, 1996. MR Zbl

RECEIVED: 27 Nov 2017 REVISED: 27 Feb 2019 ACCEPTED: 11 Apr 2019

SONJA PETROVIĆ:
sonja.petrovic@iit.edu
Department of Applied Mathematics, Illinois Institute of Technology, Chicago, IL, United States

DESPINA STASI:
stasdes@iit.edu
Department of Applied Mathematics, Illinois Institute of Technology, Chicago, IL, United States

DANE WILBURNE:
drw@yorku.ca
Department of Mathematics and Statistics, York University, Toronto ON, Canada

JOURNAL OF SOFTWARE FOR ALGEBRA AND GEOMETRY vol 9, no 1, 2019