Séminaire Lotharingien de Combinatoire **80B** (2018)
Article #97, 6 pp.

Proceedings of the 30th Conference on Formal Power Series and Algebraic Combinatorics (Hanover)

Visualizing combinatorial objects in Macaulay2

Brett Barwick*[1], Thomas Enkosky†[2], Branden Stone‡[3], and Jim Vallandingham§[4]

[1]*Division of Mathematics and Computer Science, University of South Carolina Upstate, 800 University Way, Spartanburg, SC 29303*
[2]*Mathematics and Statistics, Boston University, 111 Cummington Mall, Boston, MA 02215*
[3]*Mathematics Department, Hamilton College, 198 College Hill Road, Clinton, NY 13323*
[4]*Data Visualization Engineer,* `http://vallandingham.me/`

Abstract. Using JavaScript, this package creates interactive visualizations of a variety of objects in a modern browser. The user has the ability to manipulate the object, run various tests, and compute invariants. Once finished, the user can export the result back to the Macaulay2 session.

Keywords: Macaulay2, Graph, Poset, Simplicial Complex, Newton Polytope

1 Introduction and Current State

The computer algebra system (CAS) *Macaulay2* [5] has been funded by the National Science Foundation since 1992 and focuses mainly on algebraic geometry and commutative algebra. However, the *Macaulay2* community has also created many great packages to aid in the study of algebraic combinatorics. For example, the *SimplicialComplexes* package allows users to compute the f-vector, homology, and Alexander dual of a given simplicial complex, among other important invariants. The *Posets* package provides functionality for analyzing poset structures such as the intersection lattice of a hyperplane arrangement or an LCM lattice. In this note we discuss a new package, *Visualize*, which allows users to visually interact with combinatorial objects created by previously established packages such as these. This interaction has the potential to aid in both teaching and research.

The *Visualize* package for *Macaulay2* utilizes the JavaScript library D3 [1] to bind mathematical objects to a document object model (DOM). Once bound, these objects can be displayed and interacted with in a modern browser. The browser then becomes a portal to *Macaulay2*, allowing the user to visually edit, test, and manipulate the object

*bbarwick@uscupstate.edu. B.B. was partially supported by NSA Grant No. H98230-10-1-0361.
†tomenk@bu.edu
‡bstone@hamilton.edu
§vlandham@gmail.com

before sending it back to *Macaulay2* for analysis with more sophisticated tools. For example, if the user wants to delete an edge from a graph, they can simply select the edge with the cursor and press the delete key. Adding an edge to a graph is as simple as dragging and dropping the edge from one existing vertex to another. Once the desired graph is constructed, the object can be sent back to *Macaulay2* for further study.

Currently *Visualize* supports interaction with the *Graphs*, *Posets*, and *SimplicialComplexes* packages and a stable version is integrated in *Macaulay2*, version 1.10. The package also has the ability to create a Newton polytope associated to a monomial ideal in a polynomial ring in 2 or 3 variables and allows the user to export each visualization as TikZ code, suitable for inclusion in a manuscript. Unfortunately *Visualize* will not work on the *Macaulay2* server [6] or with CoCalc [7] as there is no display environment. As such, *Macaulay2* must be installed on the user's machine in order to use *Visualize*; this process will be discussed in Section 4.

In Section 2 we give examples of the functionality of *Visualize* and a basic workflow. Further, we provide comparisons to other programs with similar functionality in Section 3.

2 Main Functionalities and Applications

The primary function of the *Visualize* package is to create an interactive user interface with *Macaulay2* via a modern browser. The user experience is centered around 3 methods: `openPort`, `visualize`, and `closePort`. The basic workflow for this package begins by requiring the user to open a port for communication between *Macaulay2* and the web browser. Once the port is open, the `visualize` method can be used freely. The general workflow is as follows:

1. Load or install the *Visualize* package.
2. Open a port with the `openPort` method for communication with the browser. It is up to the user to choose a port and also to close the port when finished.
3. Define an object to visualize. For example, a graph, poset, digraph, etc.
4. Run the `visualize` method, passing the previously created object as an input. This will open the browser with an interactive interface. This session is in communication with *Macaulay2* through the open port above. At this point, the user can edit and manipulate the created object.
5. End the session, which automatically exports any edits made to the object back to *Macaulay2*.

6. Continue manipulating and studying the object in *Macaulay2*, repeating steps 3–5 as necessary.

7. When finished, either close the port with `closePort` or restart *Macaulay2*.

2.1 Visualizing Data

In the following we give an application of the `visualize` method in conjunction with the *Graphs* package. As mentioned above, the *Posets* and *SimplicialComplexes* packages are also supported.

In order to use the `visualize` method, the user must first open a port on their machine. This could potentially be dangerous as someone could be listening on that port. However, the open port is only accessible to the localhost and is most vulnerable if the package is being run on a server. The user may choose any port number they wish. Common ports are 8080 and 8081. For example,

```
i1 : needsPackage "Visualize"
o1 = Visualize
o1 : Package

i2 : openPort "8080"
--Port $localhost:8080 is now open.
```

With an open port, constructing an object in the browser is not much different than constructing the object normally. For example, to create a graph in *Macaulay2* we would use the following commands.

```
i3 : needsPackage"Graphs"
o3 = Graphs
o3 : Package

i4 : G = graph({{1,2},{1,3},{2,3},{3,4}},Singletons=>{5});
```

To visualize this in the browser, we use the `visualize` method.

```
i5 : visualize G
-- Visualizing. Your browser should open automatically.
-- Click 'End Session' in the browser when finished.
```

At this point the user's default browser will open with an interactive session similar to what is shown in Figure 1. With the graph displayed in the browser, the user has many options available. While the specific options vary depending on the object being visualized, the general concepts are all the same. Using the mouse, the vertices of the

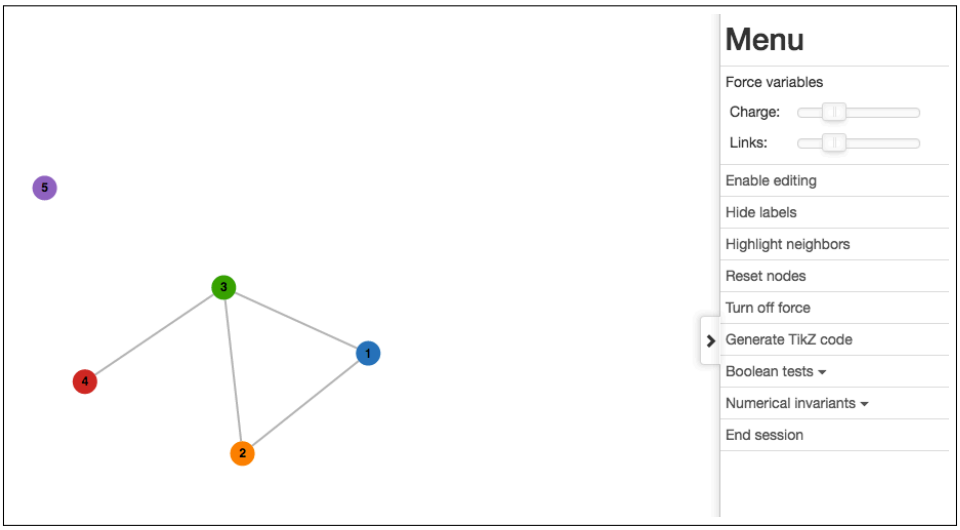

Figure 1: An interactive visualization of the graph in the browser.

graph can be moved around and pinned to the canvas. Further, creation (and deletion) of edges and vertices is possible by simply clicking and dragging.

The menu on the right side of the session provides options to enable/disable editing, show/hide labels, highlight neighbors of vertices, reset the positions of the nodes, and turn off the D3 force environment (which causes the vertices to move and naturally settle into an attractive layout). The browser interface also allows for exporting TikZ code of the graph as seen on the screen (currently only in grayscale) as well as the ability to run boolean tests or calculate numerical invariants that the *Graphs* package supports.

Once the graph has been modified and/or tested, clicking 'End session' will export the graph in the browser back to *Macaulay2* for more sophisticated calculations and constructions. When finished, the `closePort` command will close the port that was opened.

```
i6 : closePort()
--Port $localhost:8080 is now closing. This could take a few seconds.
```

Restarting or quitting *Macaulay2* will also close the port.

3 Comparison with Existing Software

Graphing and interacting with objects is not a new feature for computer algebra software. In fact, almost all major platforms (Maple, Mathematica, MATLAB, SageMath, etc.) have interactive plotting capabilities. However, these interactions are usually centered around a sliding bar which may be used to adjust a parameter, causing a static

image to refresh. Using JavaScript, the combinatorial objects pictured by *Visualize* are not static and move across the screen when created. The user is then able to tweak the particular layout of the image, directly manipulating the object without a frame of reference to center the image. Also, the object can be edited and tested directly from the browser interface, creating a new object that can be sent back to *Macaulay2* for further investigation. To the best of the authors' knowledge, there is no other current program with this functionality that supports all of the structures supported by the *Visualize* package. The closest comparable open source software package we could find is GraphViz [3]. This program has been in development for almost two decades, and as such provides more robust visualization options than our package for graphs and digraphs, especially with a large number of vertices and edges. However this software cannot interface directly with *Macaulay2* and also does not natively handle posets or simplicial complexes. While some applets and software packages have been developed for visualizing posets and/or simplicial complexes (for example [8], [2], and [10]), these software packages also do not interface natively with any computer algebra system and most do not allow the user to interact with and manipulate the objects in real-time.

Before *Visualize*, *Macaulay2* did not have any interactive visualizations for the supported packages. While some packages did support the generation of static images of graphs, posets, and Newton polytopes, the user did not have any control over the design of the image or any way to interact visually with the object.

4 Ease of Use and Sustainability

The *Visualize* package has been available with *Macaulay2* starting with version 1.10. The user only needs to run `needsPackage "Visualize"` as detailed above to have access to the package. Funded by the NSF since 1992, *Macaulay2* is open source and available for download at http://macaulay2.com for Linux and MacOS platforms. Full documentation for *Visualize* is available on the *Macaulay2* website as well as locally by running the following in *Macaulay2*.

```
i7 : installPackage "Visualize"


i8 : viewHelp Visualize
```

The documentation contains detailed examples for each of the supported packages. Currently *Visualize* only supports three packages: *Graphs*, *Posets*, and *SimplicialComplexes*. The authors plan to continue development and extend support to other packages in the future.

As of version 0.8, *Visualize* does not have any tests implemented. Most of the code is written in JavaScript and relies on connection to the *Macaulay2* server for the visualizations to be created. As far as we know, the unit testing framework available in *Macaulay2*

is not able to handle this situation. As such, we currently rely on users to post issues to the development repository on GitHub [13].

Acknowledgements

We are grateful to the following people who have generously contributed code or worked on our code at various *Macaulay2* workshops: Ata Firat Pir, Elliot Korte, Will Smith, and Julio Urenda. We are also extremely thankful to Dan Grayson and Mike Stillman for their help in facilitating communication between *Macaulay2* and the browser.

Finally, we are thankful to the authors of the open source JavaScript libraries [1, 4, 9, 11, 12]. These libraries greatly enhanced the aesthetics and functionality of our package.

References

[1] M. Bostock. "D3.js Version 3". Available at https://d3js.org.

[2] R. Freese. "Lattice Drawing Component". Available at http://www.latdraw.org/.

[3] E.R. Gansner and S.C. North. "An open graph visualization system and its applications to software engineering". *SOFTWARE - PRACTICE AND EXPERIENCE* **30**.11 (2000), pp. 1203–1233.

[4] L. Gersen. "noUiSlider.js v8.5.1". Available at http://refreshless.com/nouislider/.

[5] D.R. Grayson and M.E. Stillman. "Macaulay2, a software system for research in algebraic geometry". Available at http://www.math.uiuc.edu/Macaulay2/.

[6] F. Hinkelmann, L. Kastner, and M.E. Stillman. "Macaulay2 server". Available at http://web.macaulay2.com.

[7] SageMath Inc. *CoCalc Collaborative Computation Online*. Available at https://cocalc.com.

[8] P. Jipsen. "Interactive Poset and Lattice Drawing Java Applet". Available at http://www1.chapman.edu/ jipsen/gap/posets.html.

[9] A. Lombardo. "BootSideMenu.js". Available at https://github.com/AndreaLombardo/BootSideMenu/.

[10] A. Nikolaev. "visualsc: A simplicial complex visualization tool similar to GraphViz". Available at https://github.com/a-nikolaev/visualsc/.

[11] M. Otto and J. Thornton. "Bootstrap.js v3.3.6". Available at http://getbootstrap.com/.

[12] Z. Rocha. "clipboard.js v1.5.10". Available at https://clipboardjs.com.

[13] B. Stone, B. Barwick, T. Enkosky, and J. Vallandingham. "Visualize Package for Macaulay2". Available at https://github.com/b-stone/Visualize-M2/.