References

[CR21] Edgar Costa and David Roe, *Zen and the art of database maintenance*, accepted to Simons Symp.

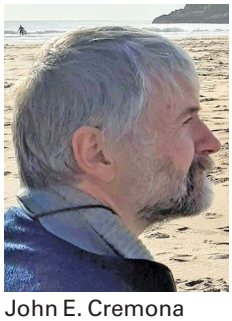
[Cre21] John Cremona, *The L-functions and modular forms database project*, Found. Comput. Math. **16** (2016), no. 6, 1541–1553.

[LMFDB] The LMFDB Collaboration, *The L-functions and modular forms database*, http://www.lmfdb.org, 2021 [Online; accessed 18 May 2021].

[Pari] The PARI Group, PARI/GP version 2.13.2, Univ. Bordeaux, 2021, https://pari.math.u-bordeaux.fr.

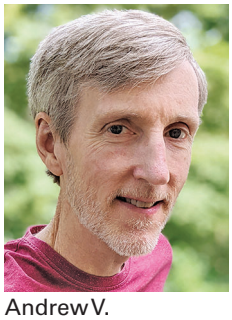
[Magma] Wieb Bosma, John Cannon, and Catherine Playoust, *The Magma algebra system. I. The user language*, J. Symbolic Comput. **24** (3–4), 1997, 235–265.

[Sage] SageMath, the Sage Mathematics Software System (Version 9.3), The Sage Developers, 2021, https://www.sagemath.org.



John E. Cremona

John W. Jones



Andrew V. Sutherland

John Voight

Credits

Figures 1 and 2 are courtesy of LMFDB.
Photo of John E. Cremona is courtesy of the author.
Photo of John W. Jones is courtesy of the author.
Photo of Andrew V. Sutherland is courtesy of the author.
Photo of John Voight is courtesy of Brian Kennedy.

Collaborative Calculation in Your Web Browser

William Stein

I greatly appreciate the opportunity to write about online resources for mathematics in this theme of the Early Career Section, since such resources are something that I have long enjoyed sharing with the community. I first started creating websites for number theorists in the late 1990s when I was a graduate student at UC Berkeley. I started by creating tables of modular forms, inspired by a question from Ken Ribet, then encouraged by other people who told me that my data inspired their research (e.g., [2b]). I later became involved with the *L*-functions and modular forms database (LMFDB) project [3b], which organized dozens of mathematicians to make amazing tables of number-theoretic data.

I also created interactive online calculators, in which you could input the parameters of some mathematical object (e.g., an elliptic curve) into a web page, and the web server would compute and display extensive information. Later, when teaching undergraduate number theory courses at Harvard, I made similar online calculators that enabled my students to run calculations using PARI and Magma (see [4b]), so that they could explore nontrivial computations in algebraic number theory. I wanted something similar to Magma that was free and open source, so I started the Python-based software SageMath in 2004 (see [5b]).

SageMath was difficult to install and only had a command line interface, so I started working to make it possible to use in a web browser. At the University of Washington (UW) in 2006, some students and I undertook the creation of a web-based interactive notebook interface called the Sage Notebook, which was inspired by Mathematica's desktop notebook. The Sage Notebook was the first serious web-based computational notebook, and it was challenging for us to implement because of the primitive Javascript and HTML technology of the day, and the dangers arising from running arbitrary user code on my server. I hosted the notebook publicly for anybody to use starting in 2007, and my students and others frequently used it in courses, summer student research programs, and SageMath development workshops. A few years later the IPython project created a new notebook called Jupyter that looked and felt similar, but with a more modern underlying architecture that was built to work with a wide range of programming environments. The arrival of the Jupyter notebook was fantastic news, because by that time the Sage Notebook's design began to feel antiquated; moreover, Jupyter benefited from

William Stein is founder and CEO of SageMath, Inc. His email address is wstein@gmail.com.

DOI: https://dx.doi.org/10.1090/noti2348

a multimillion dollar grant that resulted in substantial open source software development and publicity.

To address more general needs related to my teaching, I started dreaming about creating another web application called CoCalc in 2012 (see [1b]). CoCalc is an abbreviation for "Collaborative Calculation," and I hoped it could provide a unified home for everything I was using for my teaching and research. My favorite undergraduate course to teach at UW was called "Mathematical Computation"; it covered LaTeX, R, Python, and computational statistics, abstract algebra, graph theory, symbolic calculus, and cryptography using SageMath, and today it would significantly overlap with an introductory data science course. I wanted a web application that would be ideal for hosting this course.

Prior to launching CoCalc in 2013, I taught Mathematical Computation many times using the Sage Notebook. However, the Sage Notebook did not adequately support important topics that I wanted to teach, including "How to create LaTeX documents," "How to use a Linux terminal," and "How to develop code in files instead of a notebook." Furthermore, it did not possess the crucial ability to do simultaneous multiuser editing, which is a key feature of popular web applications today. Over the years I noticed two things about my students: (1) they appreciated working collaboratively with others, especially on final projects, and (2) they wanted to work collaboratively with themselves via a "time machine," in the sense that they wanted easy access to exactly what they were doing 15 minutes ago before they messed everything up. Having mulled over these issues, I decided I wanted to create a single web application that would support collaborative editing of Jupyter notebooks and LaTeX documents, multiuser Linux terminals, integrated chat, and a detailed browsable time travel history of editing documents. The first release incorporated some of these things, but with stripped down functionality, and it was hosted on a single desktop in my office.

That being said, I made CoCalc public and free just like the Sage Notebook, and a few other professors were interested in trying it out. Several students in my course were skilled at programming and began helping with its development. However, the instructors who first tried the application reported being annoyed with the tedium involved with uploading and downloading assignments. To combat this, we implemented a basic course management system, which automated distribution and collection of assignments.

With this course management system in place, we received much more usage. Initially, I bought a few dozen servers using a grant, and hosted them in data centers on the UW campus. As usage grew, I received free credits on Google Cloud, and moved hosting to Google, which made dynamic scaling in response to variations in load much more efficient. We ended up facing a number of challenges due to the nature of our users, who were mainly people in courses making heavy use of Python and R. In one course, the students would regularly create massive plots, so we had to come up with better ways of dealing with huge output, especially in the context of simultaneous multiuser editing. Over a period of several years, we reimplemented the Jupyter stack for our purposes, while attempting to preserve the functionality, look, and feel of the official Jupyter notebook.

We also found that instructors needed access to a broad software stack, and we accumulated hundreds of gigabytes of installed software, along with automated scripts to install and test it. It was a major technical challenge for us to make this software quickly available for use across the cluster, periodically upgrade it with automated testing, and provide stable images over time. Along the way, some of our solutions to these problems failed at scale, which resulted in lost data, difficult weeks of hard work, and many sleepless nights. Fortunately, we eventually engineered robust solutions to these problems, which have been working for several years.

In 2019, I officially resigned my Full Professor position at UW to work full time on CoCalc as a business. Today this business is reasonably stable, with a manageable number of issues, and enough paying customers so that our team can proactively improve CoCalc, rather than reacting to crises. The company's daily operations are funded entirely by customers. Much of our current development effort on CoCalc is driven by the following question, which can be asked about every relevant piece of software: "What would *X* look like if one were to focus hard for 10 years on perfecting it?" This principle guides our development efforts, and we strive to make CoCalc faster to load and navigate, and to improve its efficiency, stability, and usability. As an example, we are currently working to provide beginner-friendly graphical user interfaces for editing Jupyter notebooks and other documents. Most CoCalc users are beginners at scientific computation, so a graphical editor, with an easily browsable library of code snippets, makes CoCalc significantly more accessible to them.

During the last 20 years, over one thousand people have contributed to the online tools I have described here. Today these tools are relatively mature and powerful. I hope you find them as useful as I do, and that maybe you will help improve them over the coming decade.

William Stein

References

[1b] https://cocalc.com
[2b] https://link.springer.com/article/10.1007/s00208-003-0449-2
[3b] https://www.lmfdb.org/
[4b] http://magma.maths.usyd.edu.au/calc/
[5b] https://sagemath.org

Credits

Photo of William Stein is courtesy of the author.