# (Mis)Adventures with open source in mathematics

Alex Ghitza

The University of Melbourne

OpenScienceWorkshop, Saturday 19 July 2014

# No flame wars today

Live and let live:

- git *vs* mercurial
- github *vs* bitbucket
- emacs *vs* vim *vs* sublime text *vs* notepad (really?)
- linux *vs* mac os *vs* windows
- mathematics *vs* physics

# Why open source is the right fit for mathematics

# Campbell et al, Symmetric presentations and orthogonal groups 1998

[... description of marvelous theorems in the introduction ...]

"We achieve many of our results by systematic and substantial use of implementations of algorithms. Access to group-theoretic algorithms is provided via the computer algebra systems Cayley [6], GAP [25] and MAGMA [2]; the packages Quotpic [18] and the ANU p-Quotient Program [16, 21]; and various stand-alone programs, see for example [17]."

# 15-minute assessment of the tools

- Quotpic is available from Derek Holt's web page; it was last touched in 2000, and I haven't been able to (quickly) compile it
- the ANU p-Quotient Program used to be available from the ftp server of the Algebra group at ANU; said ftp server is no more; it's also packaged in GAP 3 (yay!) but hasn't been ported to the backward-incompatible GAP 4 (no!)
- GAP is free and open-source
- Cayley is the precursor of MAGMA, which is not free and not open source
- ... and even if we had easy access to all of the above, we would still not be able to easily verify the proof of Theorem 4

**Theorem 4** *$\langle S_{5,2}(4)\rangle$ is an infinite group. It has a homomorphism onto $O_5(4)$, and the kernel of this map has a homomorphism onto an infinite 2-group.*

**Proof:** Using the low index subgroup algorithm in Cayley we quickly find a subgroup $\langle x_1,\ x_2,\ x_3,\ x_4x_1x_2^{-1}x_3^{-1}x_2^{-1}x_1^{-1}x_4^{-1},\ x_4x_2^{-1}x_3x_4x_2^{-1}x_1^{-1}x_4^{-1}\rangle$ of index 136. Using MAGMA, we show that the permutation action on the cosets of this subgroup generates the group $O_5(4)$ of order 979 200.

Using Quotpic, we construct the homomorphism onto this group, considered as a permutation group on 85 points, and obtain a presentation for the pre-image $H$ of the point stabilizer, $2^6{:}(A_5 \times 3)$. Then, using the low index subgroups algorithm inside Quotpic, we find a subgroup $K$ of index 4 in $H$, whose core, $\bigcap_{g\in G} K^g$, has index $2^8|O_5(4)|$ in $G$.

The presentation of $K$ obtained by Quotpic is then simplified by Tietze transformations (also in Quotpic) to obtain a presentation with 3 generators and 17 relations, of total length 265. We then restart Quotpic with this presentation as input.

The abelian quotient of $K$ is cyclic of order 3, and the derived subgroup has a single homomorphism onto $A_5$ which is found by Quotpic, leading to a presentation of a subgroup $L$ of index 180 in $K$.

We simplify this presentation, first using the Tietze transformation program in Quotpic and then (to further improve the presentation) the one in GAP, to a presentation with 61 generators, 624 relations and total length 16244. The ANU $p$-Quotient Program, applied to this presentation, quickly reveals that the largest exponent-2 class-2 quotient of this group has central factors of orders $2^{61}$ and $2^{1529}$. A theorem of M. F. Newman [20] (adjusted for 2-groups as in [22]) then implies that this group must have an infinite 2-quotient.

# Joachim Neubüser 1993

"You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but [...] for many computer algebra systems license fees have to be paid regularly for the total time of their use."

# Joachim Neubüser 1993

"In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case."

# Joachim Neubüser 1993

"With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking."

# Joachim Neubüser 1993

"Not applying these rules to computer algebra systems that are made for mathematical research [...] means moving in a most undesirable direction."

# Joachim Neubüser 1993

"Most important: Can we expect somebody to believe a result of a program that he is not allowed to see?"

# Joachim Neubüser 1993

"Moreover: Do we really want to charge colleagues in Moldova several years of their salary for a computer algebra system?"

# Joachim Neubüser 1993

"And even: if O'Nan and Scott would have to pay a license fee for using an implementation of their ideas about primitive groups, should not they in turn be entitled to charge a license fee for using their ideas in the implementation?"

# Mathematical research using open source software

# Rapid prototyping

I often get some crazy idea and want to quickly see if it has any chance of being true.

Sage is particularly well-suited for this.

# Untested code is broken with probability $1 - \epsilon$

Doctests are tests embedded in the documentation. Use them wisely:

- to ferret out bugs
- to future-proof your code
- to give other users interesting examples

You may even want to use Test-Driven Development: first write the tests, then write the code.

# Untested code is broken with probability $1 - \epsilon$

Doctests are tests embedded in the documentation. Use them wisely:

- to ferret out bugs
- to future-proof your code
- to give other users interesting examples

You may even want to use Test-Driven Development: first write the tests, then write the code.

# What to do when Sage becomes too slow/memory-intensive

- go parallel: Sage/Python have very easy poor-man's parallel processing
- use a better algorithm (duh!)
- can get amazing speed-ups using Cython
- go old school: C with some low-level math libraries (e.g. GMP/MPIR, FLINT)

# Writing up the results

- Sage has LaTeX output for some of its data types, e.g. matrices
- even better: I hear SageTeX is awesome
- function graphs in Sage are okay, although you might need to dive into the matplotlib backend to achieve publication-quality results
- automate the creation of LaTeX tables of results to avoid data entry errors and simplify changes to presentation

# You need version control

# From the Git manual

"A Version Control System (VCS) allows you to:

- revert files back to a previous state
- revert the entire project back to a previous state
- review changes made over time
- see who last modified something that might be causing a problem, who introduced an issue and when
- ...

Using a VCS also means that if you screw things up or lose files, you can generally recover easily. In addition, you get all this for very little overhead."

# Flying solo

Learn a tiny bit of git and start using it for something you're writing: paper, thesis, assignment, teaching materials, code, etc.

You'll eventually wish for more advanced features. They probably already exist!

# Flying solo

Learn a tiny bit of git and start using it for something you're writing: paper, thesis, assignment, teaching materials, code, etc.

You'll eventually wish for more advanced features. They probably already exist!

# Collaboration

This is the bread-and-butter of tools like git: several people working on the same files at the same time, with merge conflicts automatically resolved as much as possible

git blame

# Hybrid approach

Not every academic you work with is willing to learn version control.

Don’t let this stop you!

# Idea/wish

Automate the process of going from *email from git-impaired collaborator* to *commit to the repository*

- include the email body as commit message
- use a separate branch for each collaborator?

# Version control and TEX

Since TEX uses text files, it's well-suited for version control.

- .gitignore: hide the myriad pesky auxiliary files
- gitinfo2: which version am I reading?
- (git-)latexdiff: great potential but buggy

# Dissemination

- make your paper's source file available on arXiv
- make your code and data available *somewhere* (your web page/github/bitbucket/etc.), no matter how messy they are
- as much as possible, use text files for data
- try to lower the barrier to using your code and data

# Scalability: two examples

- the stacks project: TEX+ git + scripting
- the polymath projects: math research via blogs; no version control; significant number of contributors

Go make something awesome and share it with the world