PlasmaFAIR

Embedding FAIRness in Plasma Science

Peter Hill, Liam Pattinson

University of York

Overview

- How open is plasma science?
- Improving software sustainability
- FAIRer simulation data

How open is plasma science?

- Other more sophisticated comparisons have been made
  - Schindler *et al* 2021, 2022
  - Federer *et al* 2018
- But either missing plasma science, or less popular plasma journals
- So, ArXiv:
  - very commonly used by plasma and similar fields
  - Enables comparing across multiple journals without worrying about how to get papers
  - Papers available from (free) cloud data dump
  - BUT: not everyone uses ArXiv, so possibly some bias?
  - ALSO: raw LaTeX not available, only PDFs (and PS), so OCR/text conversion required

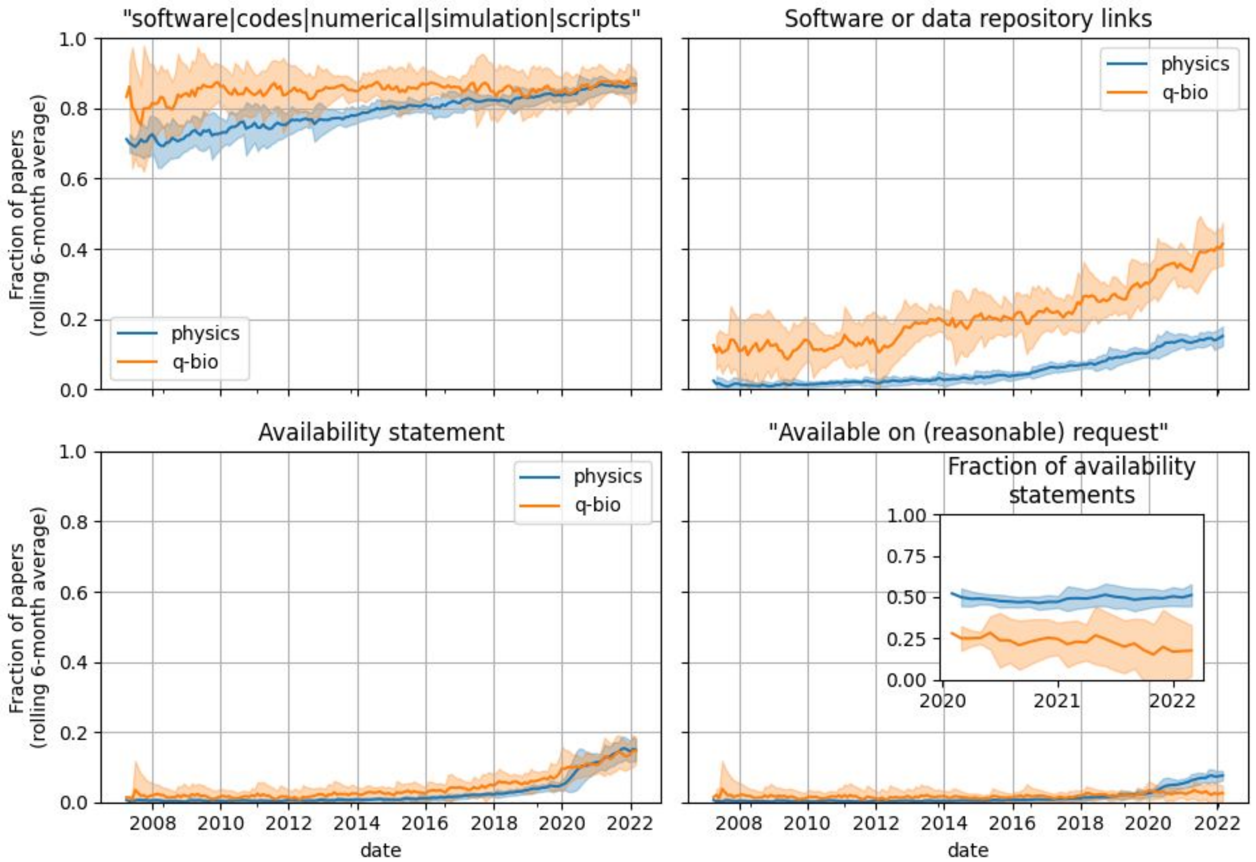
Cross-community Comparison

- Full ArXiv archive way too big: multiple TBs
- Limit analysis to "**`physics`**" (general physics) and "**`q-bio`**" (quantitative biology)
  - physics includes: Atmospheric and Oceanic Physics; Atomic Physics; Biological Physics; Chemical Physics; Computational Physics; Data Analysis, Statistics and Probability; Fluid Dynamics; History and Philosophy of Physics; Physics Education; **Plasma Physics**
  - q-bio includes: Biomolecules; Cell Behavior; Genomics; Molecular Networks; Neurons and Cognition
- Total: 255,727 papers, 0.4 TB

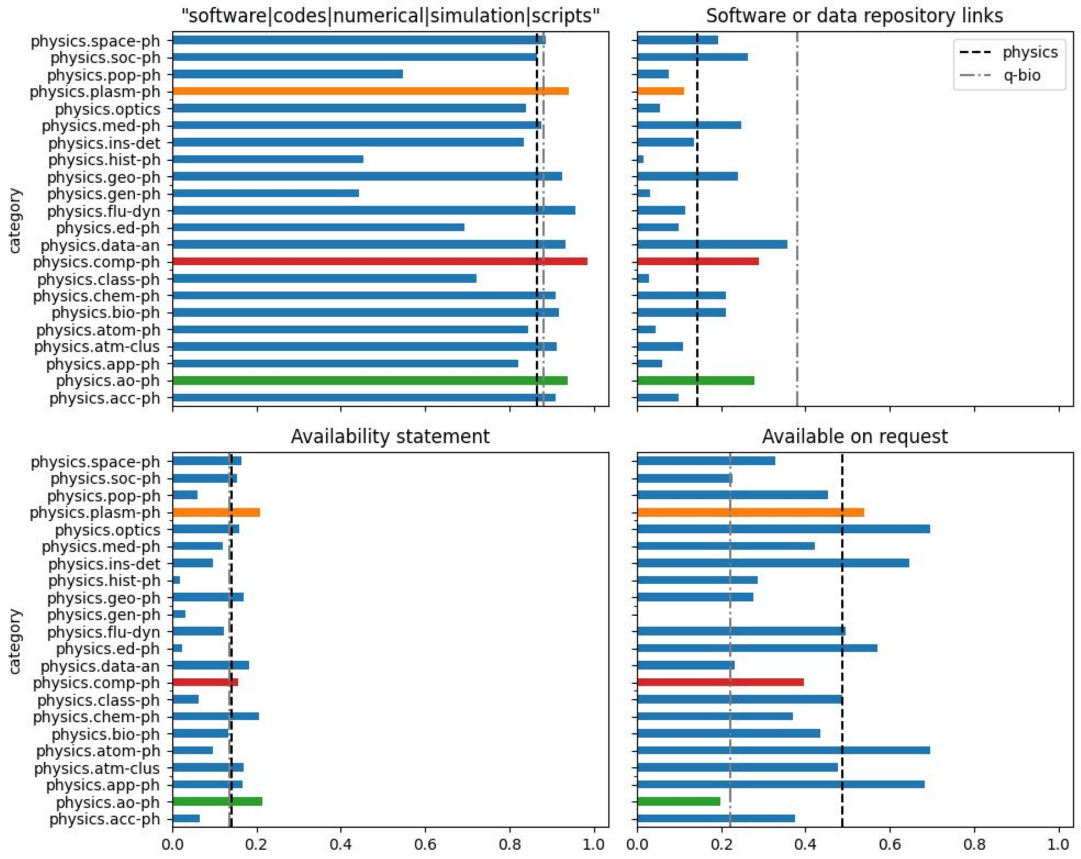
Analysis method

- Regular expression based searching
  - Using software: "software|\bcodes?\b|numerical|simulation|\bscripts?\b"
  - Data/code availability statement: "available (?:up)?on (?:reasonable)? ?request|reasonable request|(data|code) ?availability"
- Regex system far from perfect, required lots of tweaking on subset of data, looking at context
- Plan to read through sample of papers to check reliability of regex searches

Physics vs Biology trends



Physics category comparisons 2021

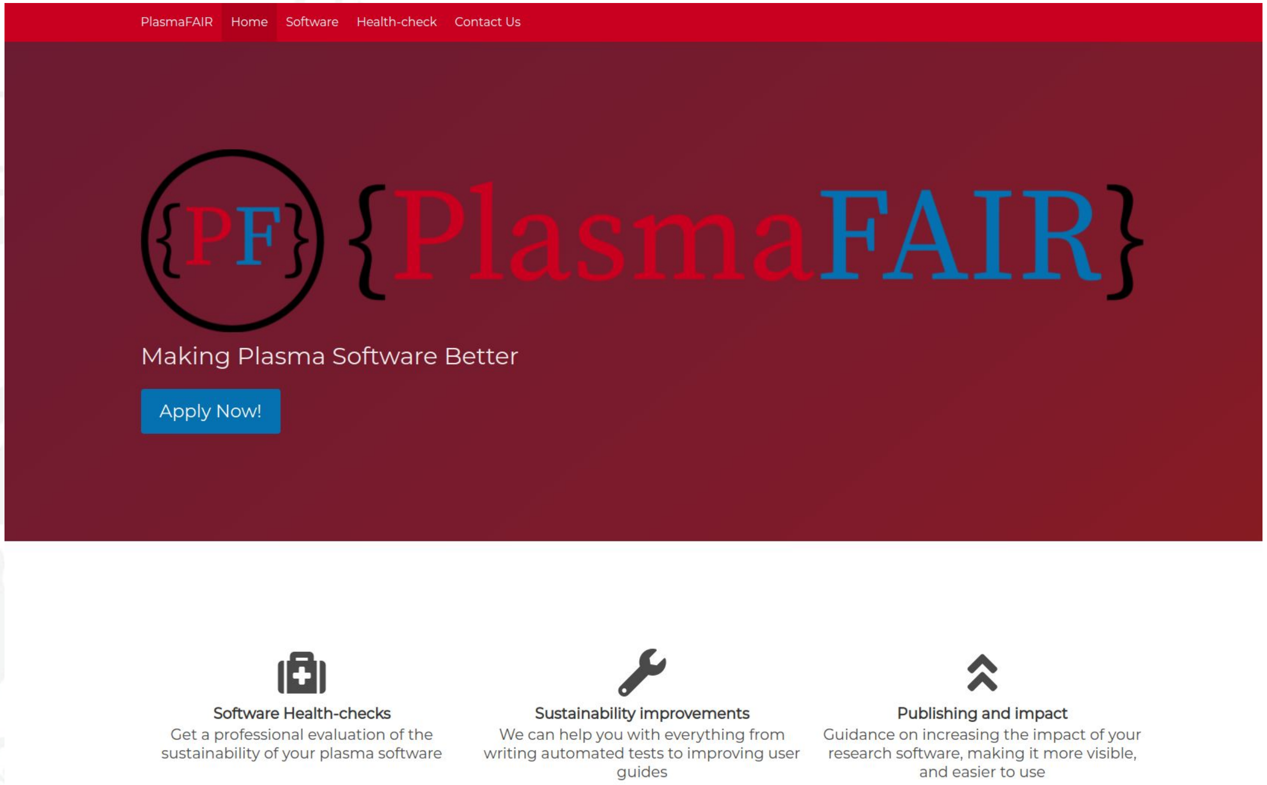


Conclusions

- Sharing of data/code has increased over time across all communities, particularly in the last two-three years
- Other communities share (relatively) a lot more
- Biology seems to benefit from domain-specific data repositories
    - => Expand existing efforts in plasma?

Usability and Sustainability Projects

- Why do people not share data/code? Stodden 2010 offers insights
- Most do want to
    - Unsurprising! People share results in papers, often in Open Access
- Most common reason:
    - **The time and effort required to clean it up**
- PlasmaFAIR: bring RSE resources to researchers to clean up code
- End goals:
    - improve sustainability of plasma research software ecosystem as a whole
    - introduce FAIR principles to researchers
- Lots of narrowly focused projects => direct interaction with more researchers



Making Plasma Software Better

Software Health-checks
Get a professional evaluation of the sustainability of your plasma software

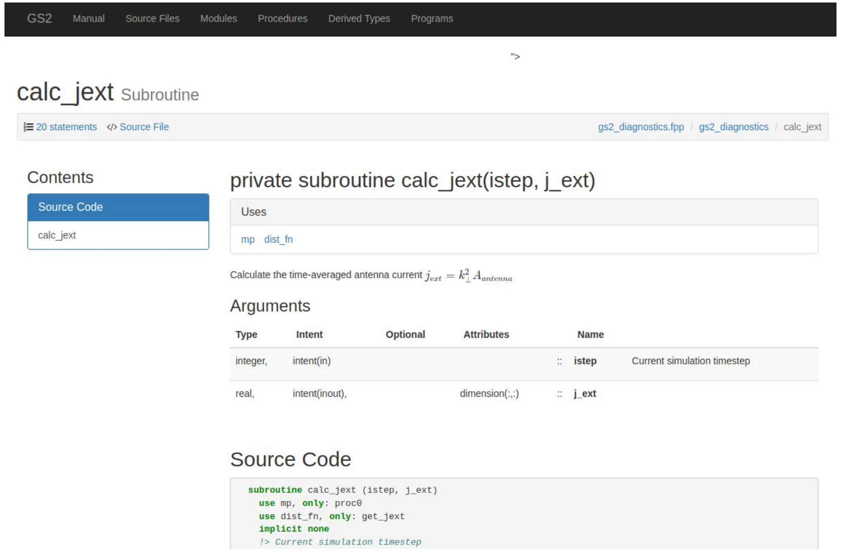
Sustainability improvements
We can help you with everything from writing automated tests to improving user guides

Publishing and impact
Guidance on increasing the impact of your research software, making it more visible, and easier to use

https://plasmafair.github.io

Case study 1: FORD

- Fortran documentation generator — Doxygen/Sphinx for Fortran
- Generates HTML pages from in-source comments (and markdown files)
- Original author no longer had time to support project
- More than two years since last release
- Documentation an essential part of sustainable software
- Fortran heavily used in plasma science
- First PlasmaFAIR project!



calc_jext Subroutine

private subroutine calc_jext(istep, j_ext)

Calculate the time-averaged antenna current $j_{ext} = k_\perp^2 A_{antenna}$

Arguments

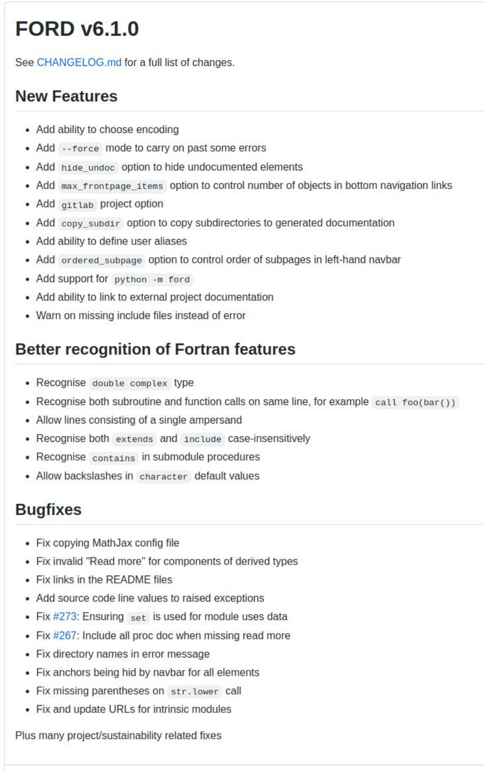
Type	Intent	Optional	Attributes		Name	
integer,	intent(in)			::	**istep**	Current simulation timestep
real,	intent(inout),		dimension(:,:)	::	**j_ext**	

Source Code

```
subroutine calc_jext (istep, j_ext)
  use mp, only: proc0
  use dist_fn, only: get_jext
  implicit none
  !> Current simulation timestep
```

Case study 1: FORD

- Took over maintainership
- Reviewed ~30 outstanding PRs
- Merged bug fixes and new features => new release
- Implemented modern Python packaging best practices
- Added CI, unit tests, automated packaging
- Now merged 54 PRs, fixed >30 bugs, added >100 tests, made 12 releases



FORD v6.1.0

See CHANGELOG.md for a full list of changes.

New Features

- Add ability to choose encoding
- Add `--force` mode to carry on past some errors
- Add `hide_undoc` option to hide undocumented elements
- Add `max_frontpage_items` option to control number of objects in bottom navigation links
- Add `gitlab` project option
- Add `copy_subdir` option to copy subdirectories to generated documentation
- Add ability to define user aliases
- Add `ordered_subpage` option to control order of subpages in left-hand navbar
- Add support for `python -m ford`
- Add ability to link to external project documentation
- Warn on missing include files instead of error

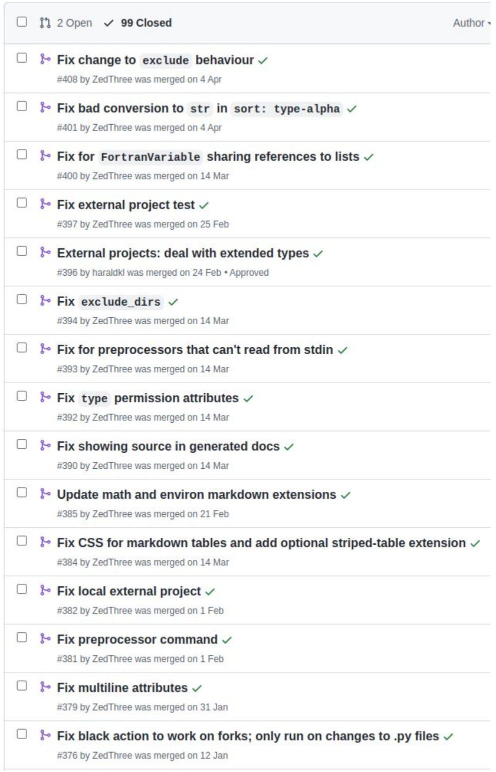
Better recognition of Fortran features

- Recognise `double complex` type
- Recognise both subroutine and function calls on same line, for example `call foo(bar())`
- Allow lines consisting of a single ampersand
- Recognise both `extends` and `include` case-insensitively
- Recognise `contains` in submodule procedures
- Allow backslashes in `character` default values

Bugfixes

- Fix copying MathJax config file
- Fix invalid "Read more" for components of derived types
- Fix links in the README files
- Add source code line values to raised exceptions
- Fix #273: Ensuring `set` is used for module uses data
- Fix #267: Include all proc doc when missing read more
- Fix directory names in error message
- Fix anchors being hid by navbar for all elements
- Fix missing parentheses on `str.lower` call
- Fix and update URLs for intrinsic modules

Plus many project/sustainability related fixes



2 Open ✓ 99 Closed Author

- **Fix change to `exclude` behaviour** ✓
  #408 by ZedThree was merged on 4 Apr
- **Fix bad conversion to `str` in `sort: type-alpha`** ✓
  #401 by ZedThree was merged on 4 Apr
- **Fix for `FortranVariable` sharing references to lists** ✓
  #400 by ZedThree was merged on 14 Mar
- **Fix external project test** ✓
  #397 by ZedThree was merged on 25 Feb
- **External projects: deal with extended types** ✓
  #396 by haraldkl was merged on 24 Feb • Approved
- **Fix `exclude_dirs`** ✓
  #394 by ZedThree was merged on 14 Mar
- **Fix for preprocessors that can't read from stdin** ✓
  #393 by ZedThree was merged on 14 Mar
- **Fix `type` permission attributes** ✓
  #392 by ZedThree was merged on 14 Mar
- **Fix showing source in generated docs** ✓
  #390 by ZedThree was merged on 14 Mar
- **Update math and environ markdown extensions** ✓
  #385 by ZedThree was merged on 21 Feb
- **Fix CSS for markdown tables and add optional striped-table extension** ✓
  #384 by ZedThree was merged on 14 Mar
- **Fix local external project** ✓
  #382 by ZedThree was merged on 1 Feb
- **Fix preprocessor command** ✓
  #381 by ZedThree was merged on 1 Feb
- **Fix multiline attributes** ✓
  #379 by ZedThree was merged on 31 Jan
- **Fix black action to work on forks; only run on changes to .py files** ✓
  #376 by ZedThree was merged on 12 Jan

Case study 2: Neasy-f

- Wrapper for NetCDF Fortran API
- Designed to make common patterns simple and enable piecewise use (i.e. plays nice with the standard Fortran API)
- Makes use of NetCDF-4 features
  - Backed by HDF5: can enable compression (faster IO, smaller file size)
  - No need to separate defining and writing variables
- Removes need to keep variable handles around for program lifetime
- Built-in error checking (aborts if error detected)
- Handles some conventional attributes and metadata
- Fortran 2008 features to reduce interface explosion
- Used in GS2: enabled removal of net 1200 lines

Case study 2: Neasy-f

- Left: official NetCDF Fortran example; Right: rewritten with Neasy-f
- No need for user-written check subroutine
- Variable definition, conventional metadata, and write done in same call

```fortran
call check( nf90_create("my_file.nc", &
            ior(nf90_clobber, nf90_netcdf4), ncid) )

call check( nf90_def_dim(ncid, "x", NX, x_dimid) )
call check( nf90_def_dim(ncid, "y", NY, y_dimid) )

call check( nf90_def_var(ncid, "data", NF90_INT, &
            [y_dimid, x_dimid], varid)
call check( nf90_put_att(ncid, varid, "units", "Pa") )
call check( nf90_put_att(ncid, varid, &
            "long_name", "Synthetic pressure") )

call check( nf90_enddef(ncid) )

call check( nf90_put_var(ncid, varid, data_out) )

call check( nf90_close(ncid) )
```

```fortran
ncid = neasyf_open("my_file.nc", "w")

call neasyf_dim(ncid, "x", dim_size=NX)
call neasyf_dim(ncid, "y", dim_size=NY)

call neasyf_write(ncid, "data", data_out, ["y", "x"], &
       units="Pa", long_name="Synthetic pressure")

call neasyf_close(ncid)
```

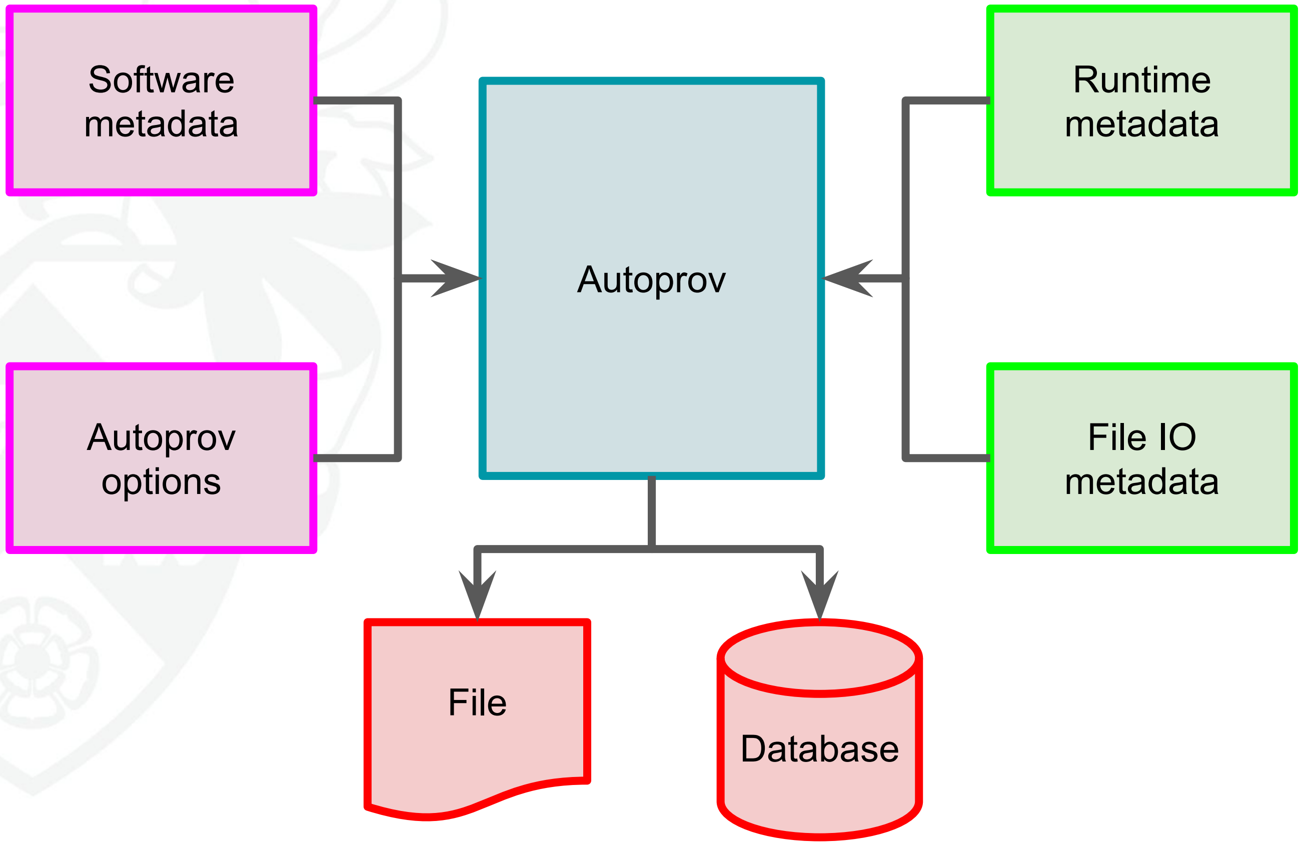
Automatic Provenance Capturing

- Many tools exist for capturing provenance information
- But, either:
  - For dynamic languages (e.g. Python)
  - Requires use of workflow management tool
- Want something:
  - that works with C/C++/Fortran and MPI/HPC
  - simple to integrate with existing software
  - that requires little interaction from end user
- Use cases:
  - Finding simulations that match some parameters
  - Enabling machine learning
  - Easier, FAIRer archiving

Autoprov

- New tool, still in development, pre-alpha
- Two function calls: `autoprov_init` and `autoprov_finish`
- Implemented in C++ with C and Fortran APIs
- Automatic capturing of runtime information
- Automatic capturing of file IO metadata
- Output to metadata file and/or database
- MPI compatible

Autoprov


Software metadata
Autoprov options
Autoprov
Runtime metadata
File IO metadata
File
Database

Autoprov

```c
#include <autoprov/autoprov.h>

int main(int argc, char* argv[]){
    AutoprovOpts opts = autoprov_default_opts();
    AutoprovMetadata metadata = {"test", "1.2.3"};
    autoprov_init(argc, argv, &opts, &metadata);
    /* do things */
    autoprov_finalize();
    return 0;
}
```

Conclusions

- Plasma community could share more
- PlasmaFAIR providing usability and sustainability projects
- https://plasmafair.github.io
- Autoprov: automatic provenance capturing