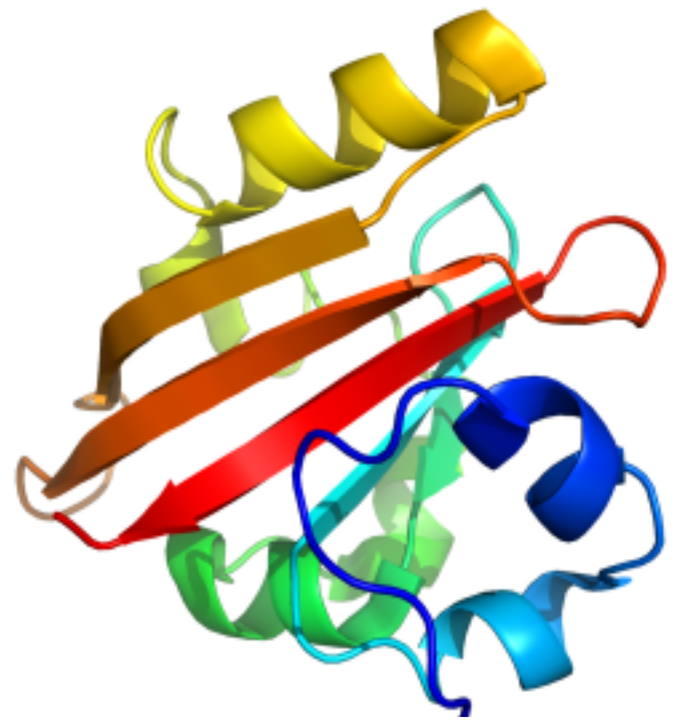
Serial crystallography

Data processing and thoughts about "The Digital Scientific Method"



Thomas White
7. MT Meeting
3. Feb 2021

HELMHOLTZ RESEARCH FOR GRAND CHALLENGES

CFEL SCIENCE

DESY.

Serial crystallography

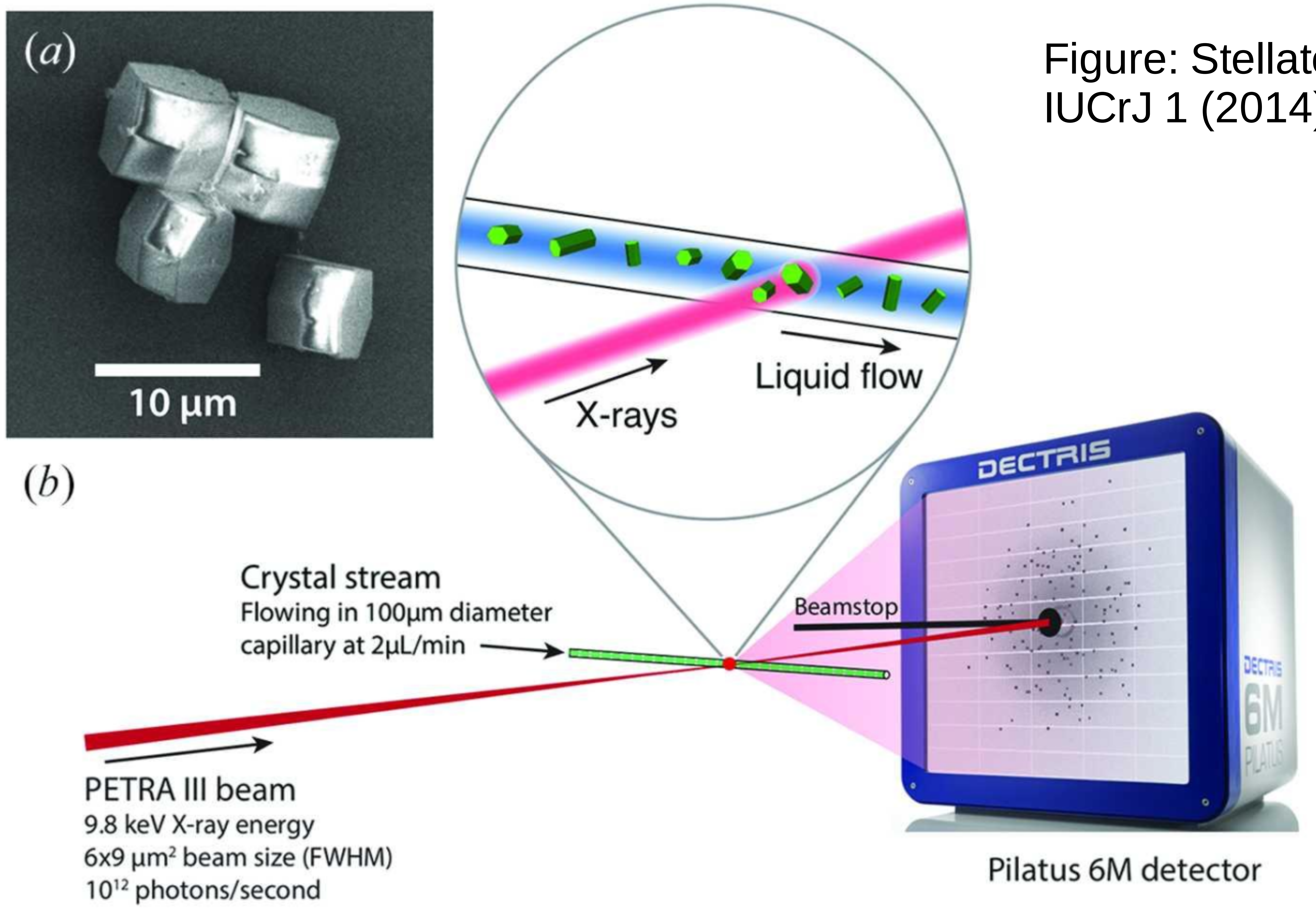
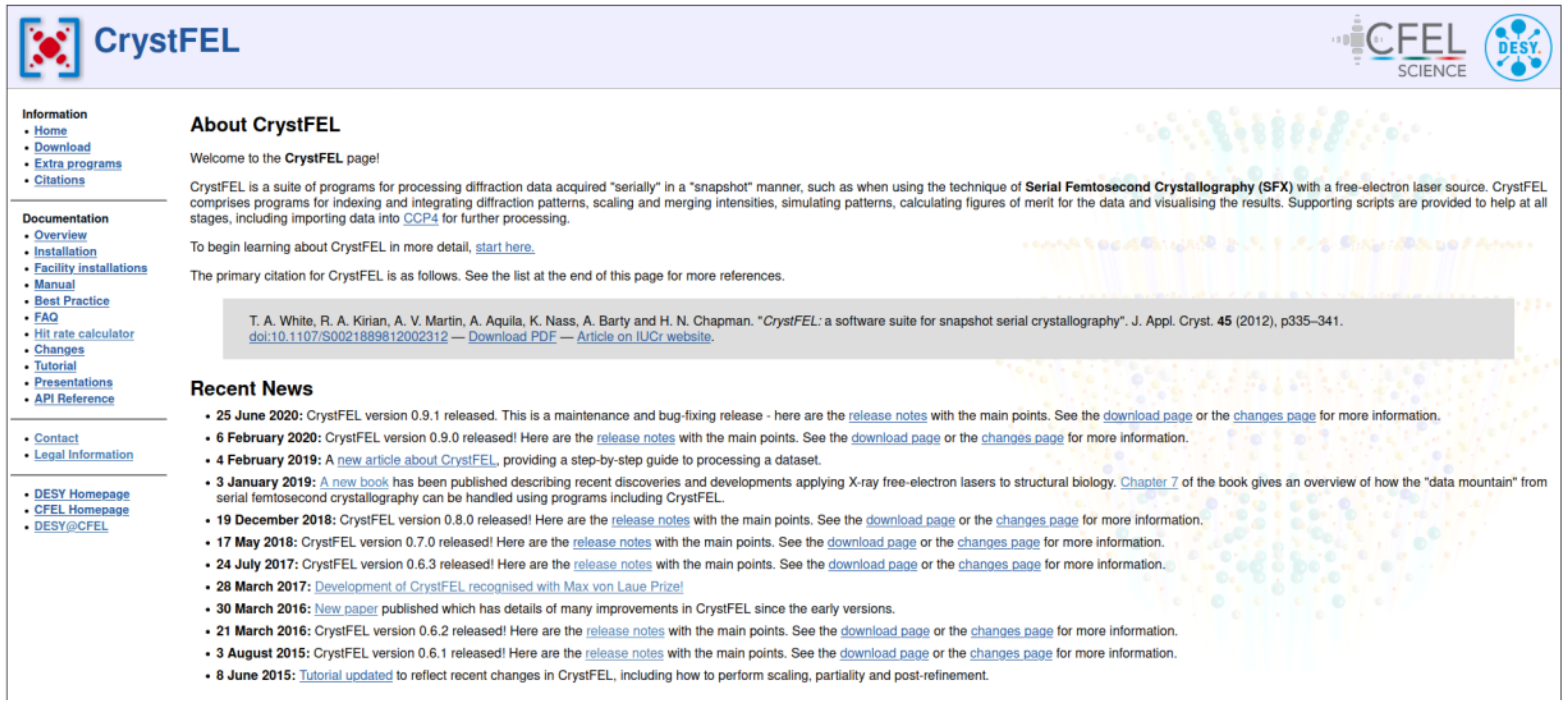


Figure: Stellato et al., IUCrJ 1 (2014) p204-212

CrystFEL: data processing for serial crystallography



CrystFEL

CFEL SCIENCE

DESY.

Information
- Home
- Download
- Extra programs
- Citations

Documentation
- Overview
- Installation
- Facility installations
- Manual
- Best Practice
- FAQ
- Hit rate calculator
- Changes
- Tutorial
- Presentations
- API Reference

- Contact
- Legal Information

- DESY Homepage
- CFEL Homepage
- DESY@CFEL

About CrystFEL

Welcome to the **CrystFEL** page!

CrystFEL is a suite of programs for processing diffraction data acquired "serially" in a "snapshot" manner, such as when using the technique of **Serial Femtosecond Crystallography (SFX)** with a free-electron laser source. CrystFEL comprises programs for indexing and integrating diffraction patterns, scaling and merging intensities, simulating patterns, calculating figures of merit for the data and visualising the results. Supporting scripts are provided to help at all stages, including importing data into CCP4 for further processing.

To begin learning about CrystFEL in more detail, start here.

The primary citation for CrystFEL is as follows. See the list at the end of this page for more references.

> T. A. White, R. A. Kirian, A. V. Martin, A. Aquila, K. Nass, A. Barty and H. N. Chapman. "*CrystFEL:* a software suite for snapshot serial crystallography". J. Appl. Cryst. **45** (2012), p335–341.
> doi:10.1107/S0021889812002312 — Download PDF — Article on IUCr website.

Recent News

- **25 June 2020:** CrystFEL version 0.9.1 released. This is a maintenance and bug-fixing release - here are the release notes with the main points. See the download page or the changes page for more information.
- **6 February 2020:** CrystFEL version 0.9.0 released! Here are the release notes with the main points. See the download page or the changes page for more information.
- **4 February 2019:** A new article about CrystFEL, providing a step-by-step guide to processing a dataset.
- **3 January 2019:** A new book has been published describing recent discoveries and developments applying X-ray free-electron lasers to structural biology. Chapter 7 of the book gives an overview of how the "data mountain" from serial femtosecond crystallography can be handled using programs including CrystFEL.
- **19 December 2018:** CrystFEL version 0.8.0 released! Here are the release notes with the main points. See the download page or the changes page for more information.
- **17 May 2018:** CrystFEL version 0.7.0 released! Here are the release notes with the main points. See the download page or the changes page for more information.
- **24 July 2017:** CrystFEL version 0.6.3 released! Here are the release notes with the main points. See the download page or the changes page for more information.
- **28 March 2017:** Development of CrystFEL recognised with Max von Laue Prize!
- **30 March 2016:** New paper published which has details of many improvements in CrystFEL since the early versions.
- **21 March 2016:** CrystFEL version 0.6.2 released! Here are the release notes with the main points. See the download page or the changes page for more information.
- **3 August 2015:** CrystFEL version 0.6.1 released! Here are the release notes with the main points. See the download page or the changes page for more information.
- **8 June 2015:** Tutorial updated to reflect recent changes in CrystFEL, including how to perform scaling, partiality and post-refinement.

https://www.desy.de/~twhite/crystfel

Throughout 2019 and 2020, a new application of CrystFEL came out in peer-reviewed literature every 10 days (on average).

Development since 2009 at DESY.

First "public" release in 2012.

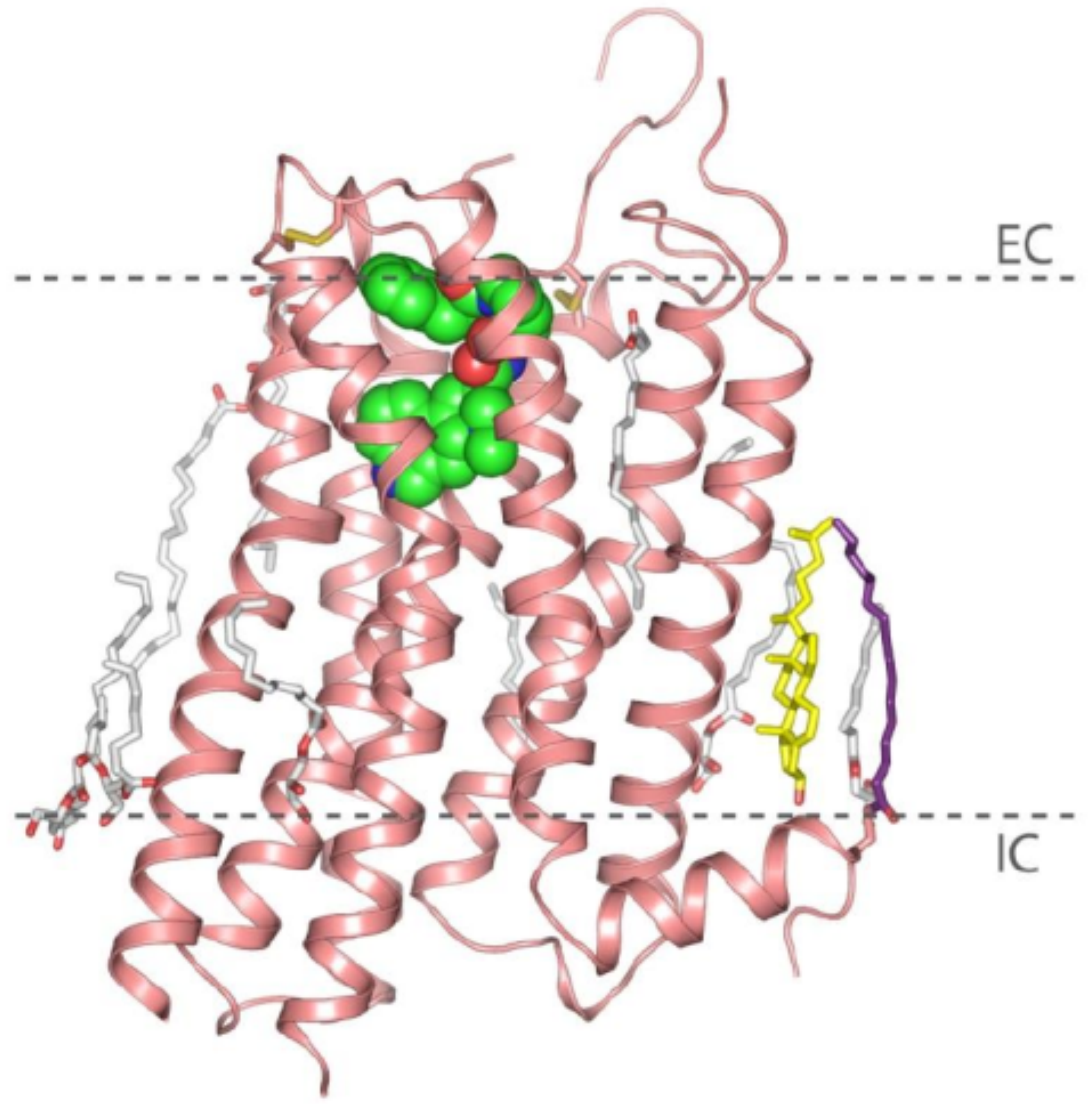
Project led by me (Thomas White)

Tools for indexing, integration, merging, calculating figures of merit, simulating test data and more.

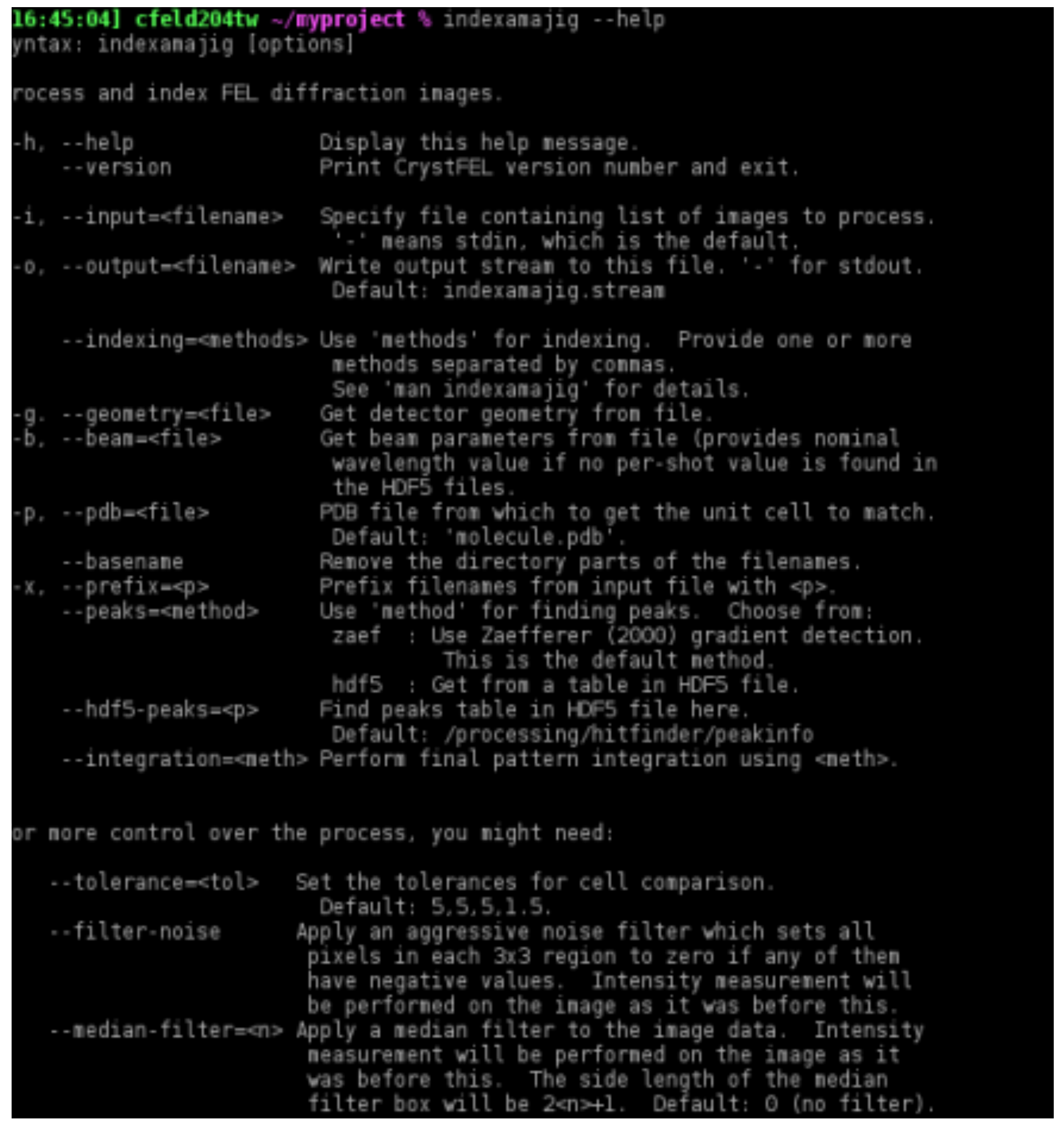
CrystFEL processing pipeline

Data preparation if data is not already in a suitable format, or needs detector calibration.
Cheetah, CASS, psocake or OnDA
Detector readout
Detector corrections / calibrations
Hit finding
HDF5 or CBF file(s)
Key
File
External program
CrystFEL program
program name
Geometry file (includes wavelength etc)
Peak search, indexing and integration
indexamajig
Stream
Determination of lattice parameters
cell_explorer
Resolve indexing ambiguity (if any)
ambigator
Re-indexed stream
If indexing ambiguity is known to be present, or if there are signs of twinning in the final data.
Fast and simple merging
process_hkl
Merging with scaling/ partialities
partialator
--even-only
--odd-only
Merged data (.hkl)
Merged half-dataset A (.hkl1)
Merged half-dataset B (.hkl2)

What can we expect "users" to know?

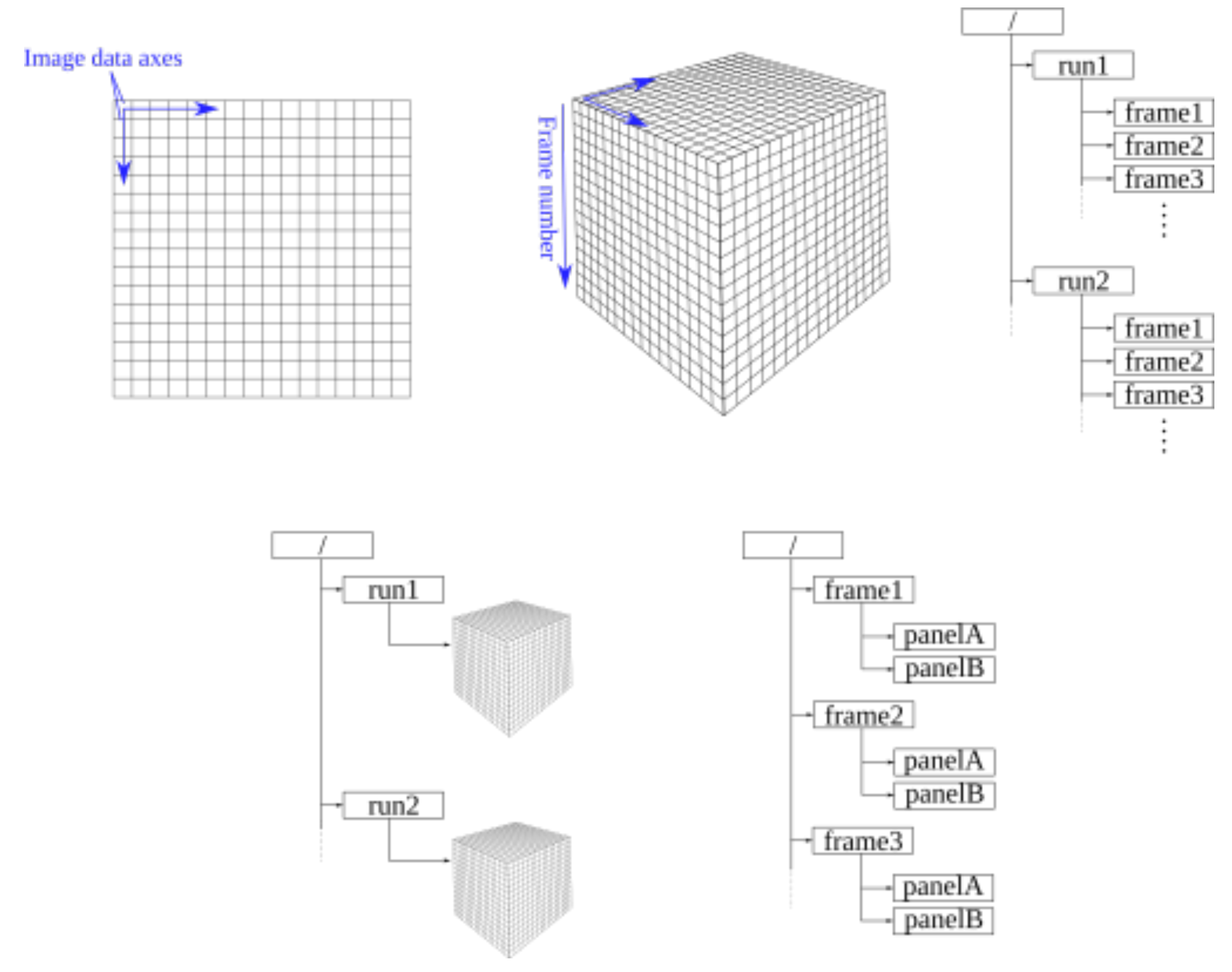


Structural biology

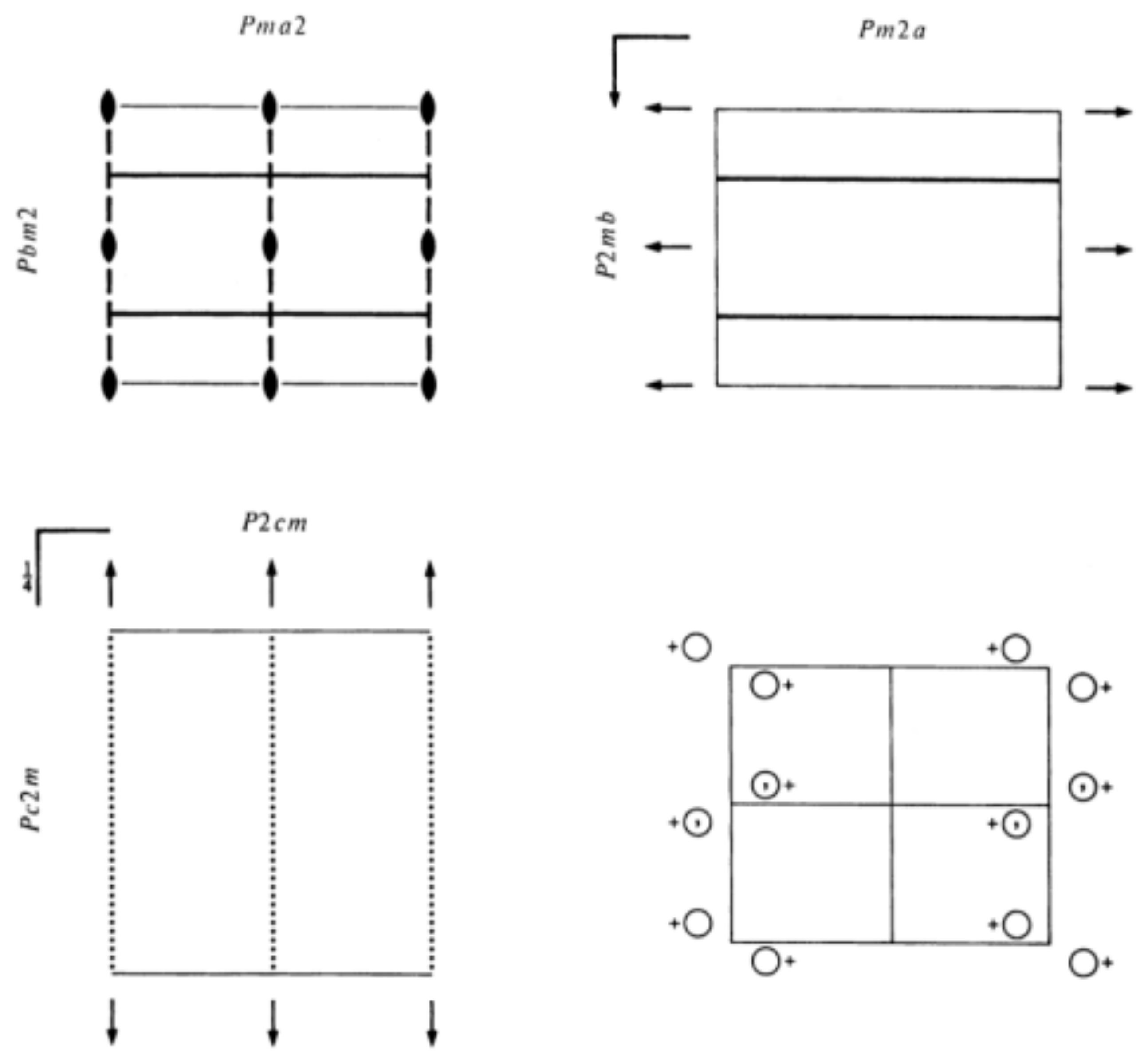


```
16:45:04] cfeld204tw ~/myproject % indexamajig --help
yntax: indexamajig [options]

rocess and index FEL diffraction images.

-h, --help               Display this help message.
    --version            Print CrystFEL version number and exit.

-i, --input=<filename>   Specify file containing list of images to process.
                          '-' means stdin, which is the default.
-o, --output=<filename>  Write output stream to this file. '-' for stdout.
                          Default: indexamajig.stream

    --indexing=<methods> Use 'methods' for indexing.  Provide one or more
                          methods separated by commas.
                          See 'man indexamajig' for details.
-g. --geometry=<file>    Get detector geometry from file.
-b, --beam=<file>        Get beam parameters from file (provides nominal
                          wavelength value if no per-shot value is found in
                          the HDF5 files.
-p, --pdb=<file>         PDB file from which to get the unit cell to match.
                          Default: 'molecule.pdb'.
    --basename           Remove the directory parts of the filenames.
-x, --prefix=<p>         Prefix filenames from input file with <p>.
    --peaks=<method>     Use 'method' for finding peaks.  Choose from:
                          zaef  : Use Zaefferer (2000) gradient detection.
                                  This is the default method.
                          hdf5  : Get from a table in HDF5 file.
    --hdf5-peaks=<p>     Find peaks table in HDF5 file here.
                          Default: /processing/hitfinder/peakinfo
    --integration=<meth> Perform final pattern integration using <meth>.


or more control over the process, you might need:

   --tolerance=<tol>   Set the tolerances for cell comparison.
                         Default: 5,5,5,1.5.
   --filter-noise      Apply an aggressive noise filter which sets all
                        pixels in each 3x3 region to zero if any of them
                        have negative values.  Intensity measurement will
                        be performed on the image as it was before this.
   --median-filter=<n> Apply a median filter to the image data.  Intensity
                        measurement will be performed on the image as it
                        was before this.  The side length of the median
                        filter box will be 2<n>+1.  Default: 0 (no filter).
```

Unix command line



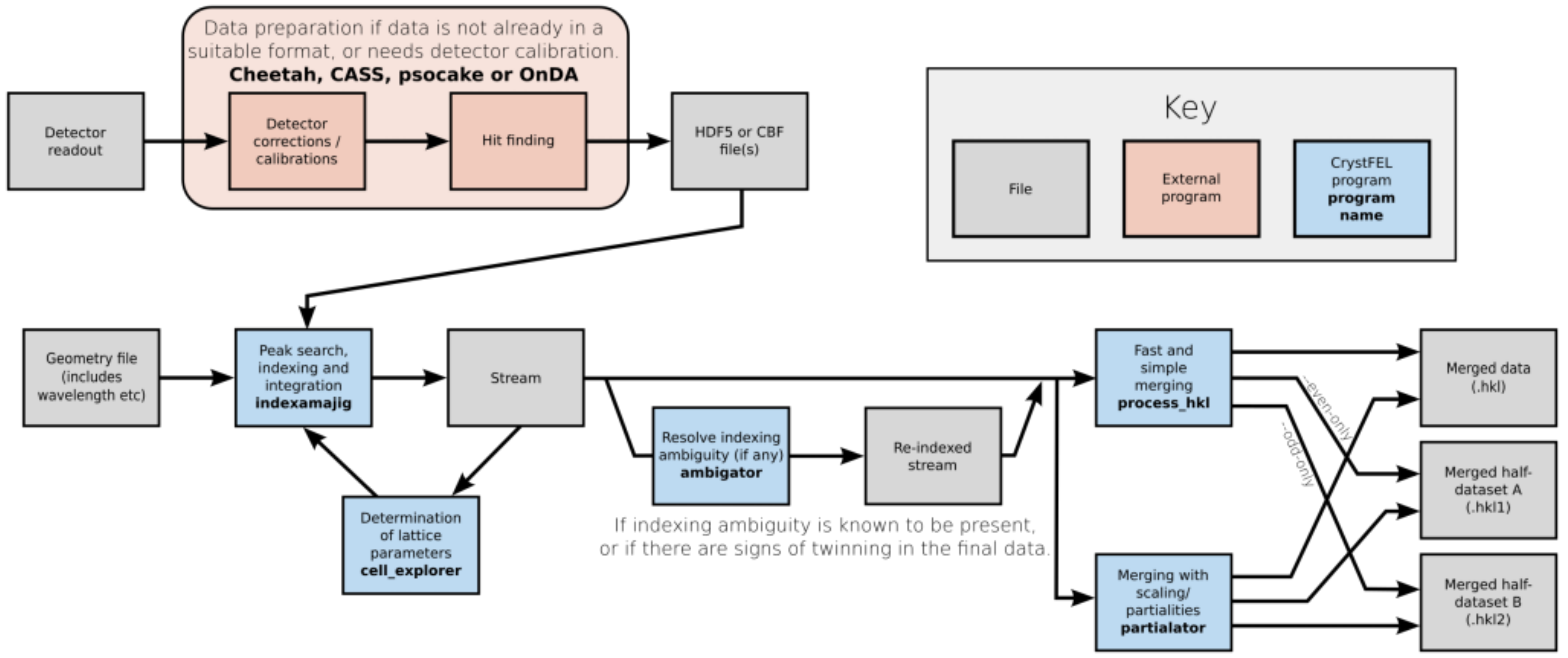
Data formats



Crystallographic theory

slurm
workload manager

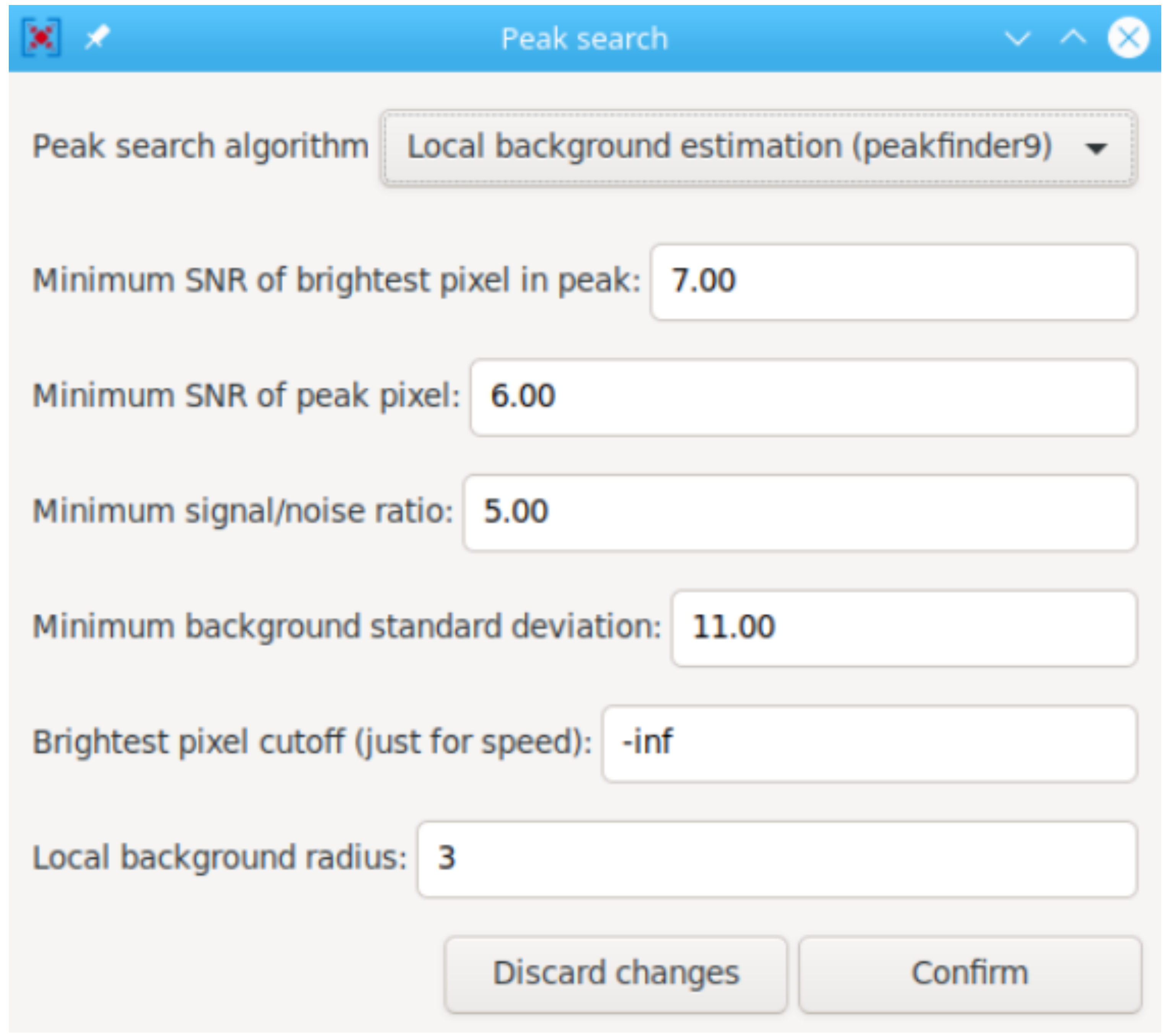
Cluster / batch system



Data preparation if data is not already in a suitable format, or needs detector calibration.
Cheetah, CASS, psocake or OnDA
Detector readout
Detector corrections / calibrations
Hit finding
HDF5 or CBF file(s)
Key
File
External program
CrystFEL program
program name
Geometry file (includes wavelength etc)
Peak search, indexing and integration
indexamajig
Stream
Determination of lattice parameters
cell_explorer
Resolve indexing ambiguity (if any)
ambigator
Re-indexed stream
If indexing ambiguity is known to be present, or if there are signs of twinning in the final data.
Fast and simple merging
process_hkl
Merging with scaling/ partialities
partialator
--even-only
--odd-only
Merged data (.hkl)
Merged half-dataset A (.hkl1)
Merged half-dataset B (.hkl2)

CrystFEL

The CrystFEL GUI



Peak search algorithm: Local background estimation (peakfinder9)

Minimum SNR of brightest pixel in peak: 7.00

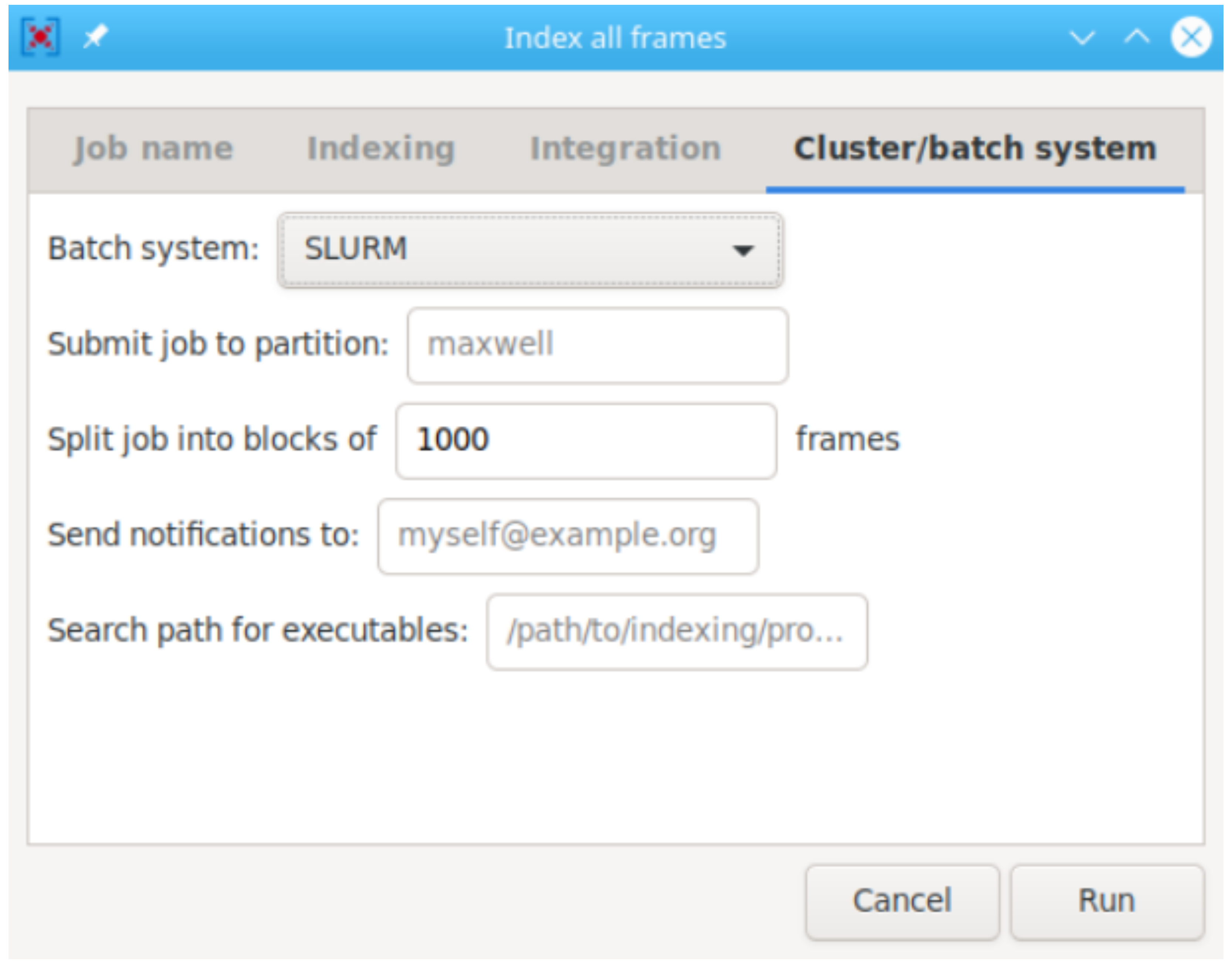
Minimum SNR of peak pixel: 6.00

Minimum signal/noise ratio: 5.00

Minimum background standard deviation: 11.00

Brightest pixel cutoff (just for speed): -inf

Local background radius: 3



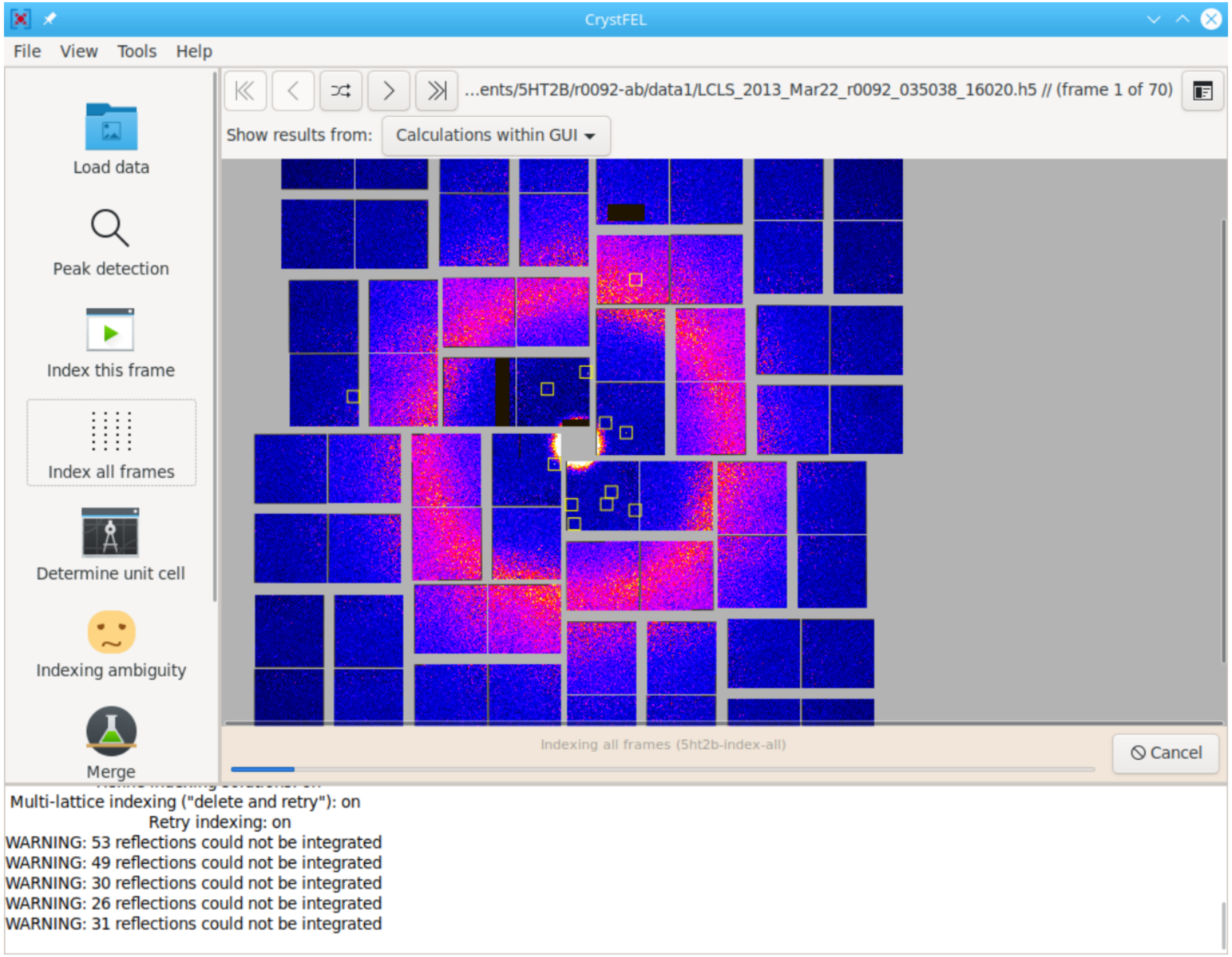
Batch system: SLURM

Submit job to partition: maxwell

Split job into blocks of 1000 frames

Send notifications to: myself@example.org

Search path for executables: /path/to/indexing/pro...



...ents/5HT2B/r0092-ab/data1/LCLS_2013_Mar22_r0092_035038_16020.h5 // (frame 1 of 70)

Show results from: Calculations within GUI

Indexing all frames (5ht2b-index-all)

```
Multi-lattice indexing ("delete and retry"): on
Retry indexing: on
WARNING: 53 reflections could not be integrated
WARNING: 49 reflections could not be integrated
WARNING: 30 reflections could not be integrated
WARNING: 26 reflections could not be integrated
WARNING: 31 reflections could not be integrated
```

Benefits of real-time data processing

Faster results and publication

No need to store raw data (except for audit/accountability)

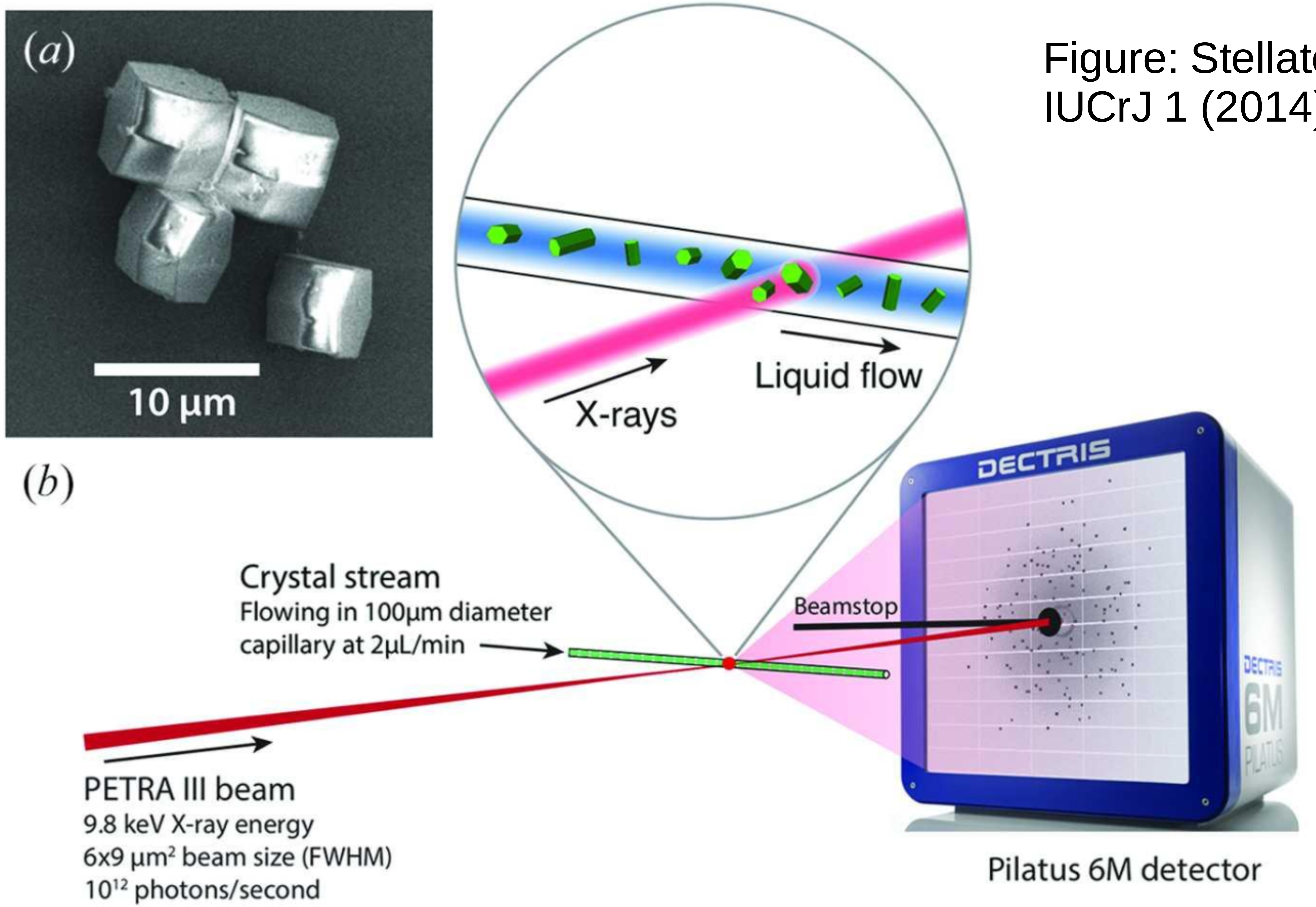
Better situational awareness during experiment ("Do we have enough data yet?")

Faster diagnosis of experiment problems (e.g. pump laser misaligned)

Less scope for "fiddling and taking the result you like best"

Serial crystallography

Figure: Stellato et al., IUCrJ 1 (2014) p204-212



Benefits of real-time data processing

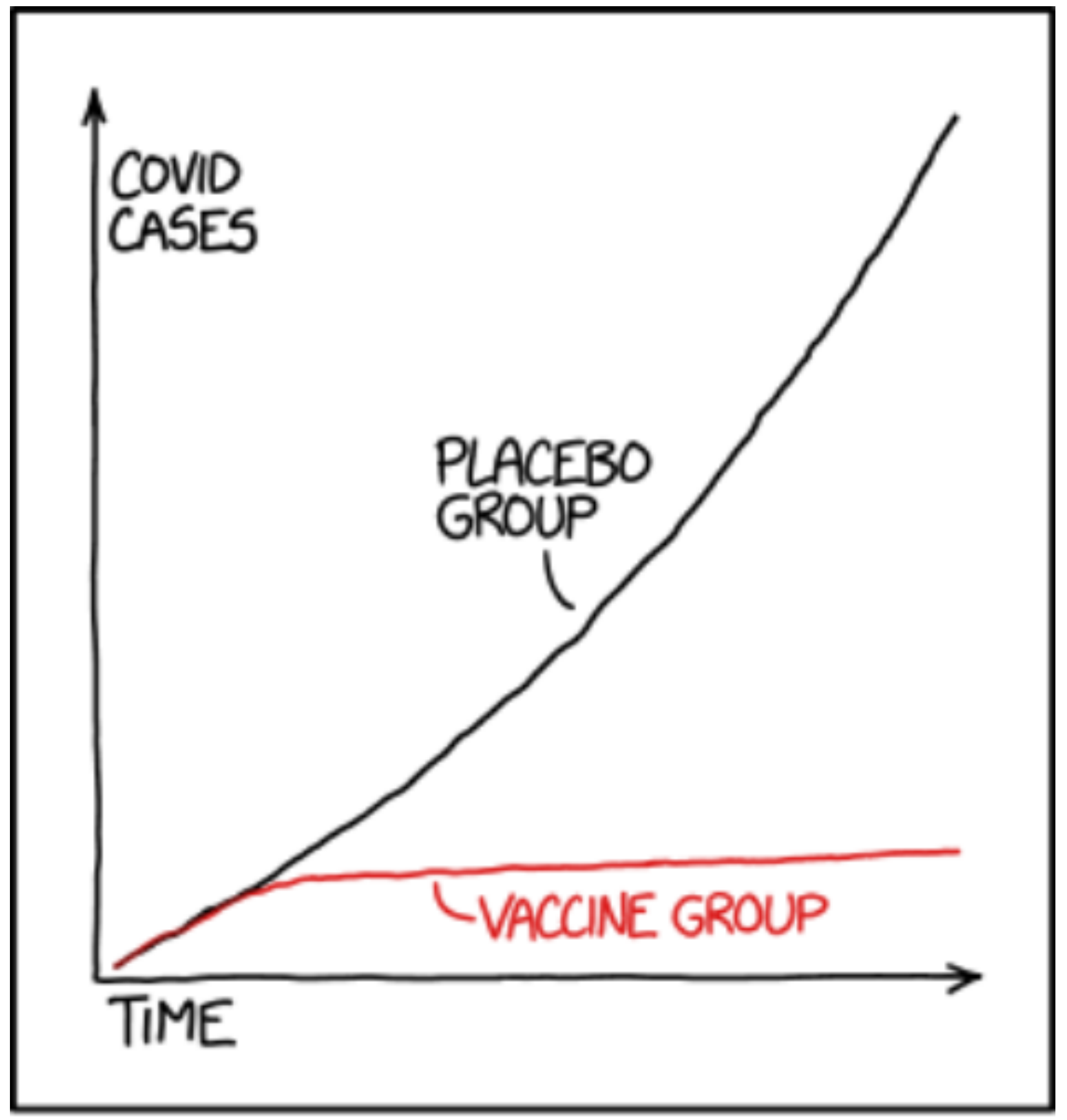
Faster results and publication

No need to store raw data (except for audit/accountability)

Better situational awareness during experiment ("Do we have enough data yet?")

Faster diagnosis of experiment problems (e.g. pump laser misaligned)

Less scope for "fiddling and taking the result you like best" ——————————>

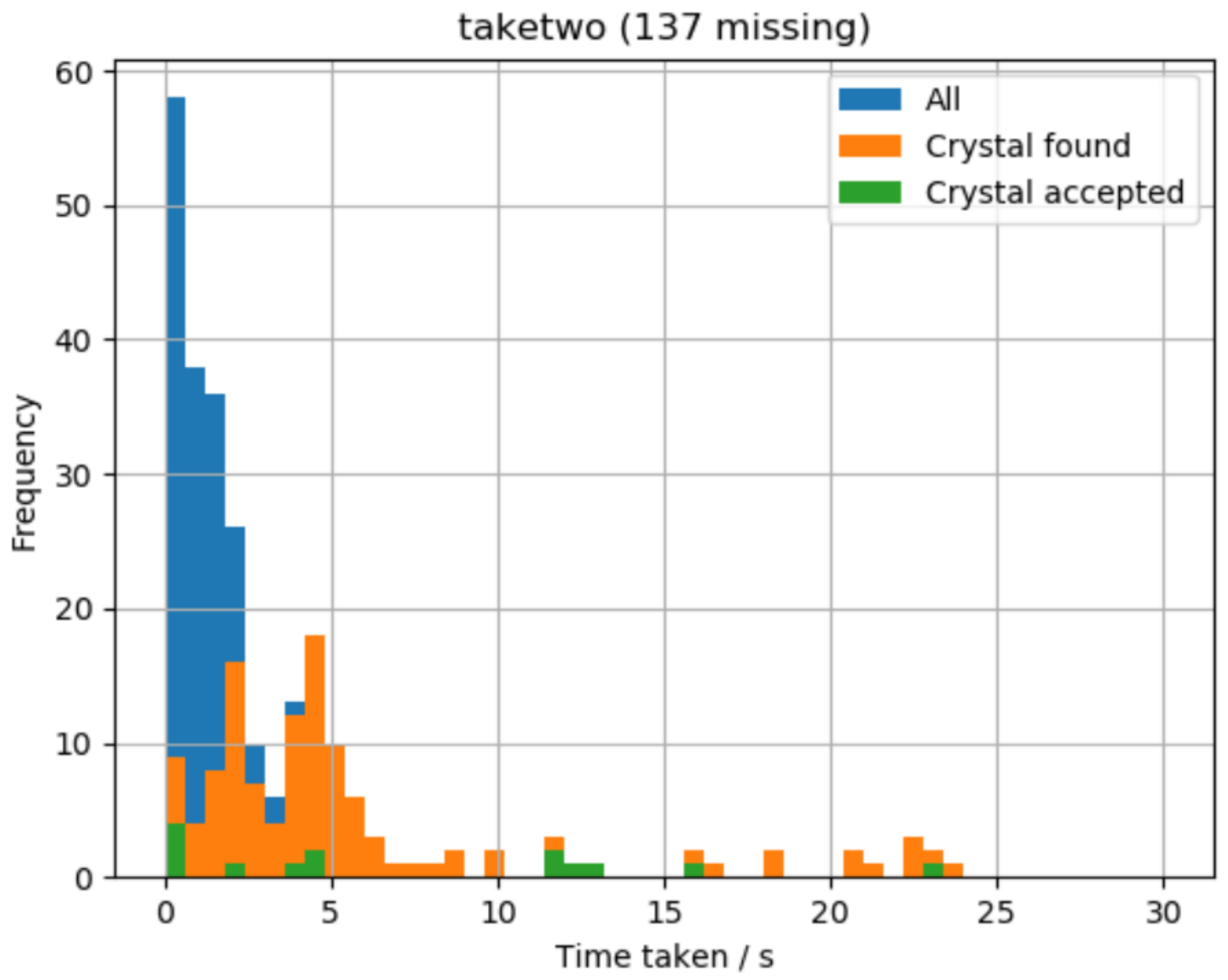


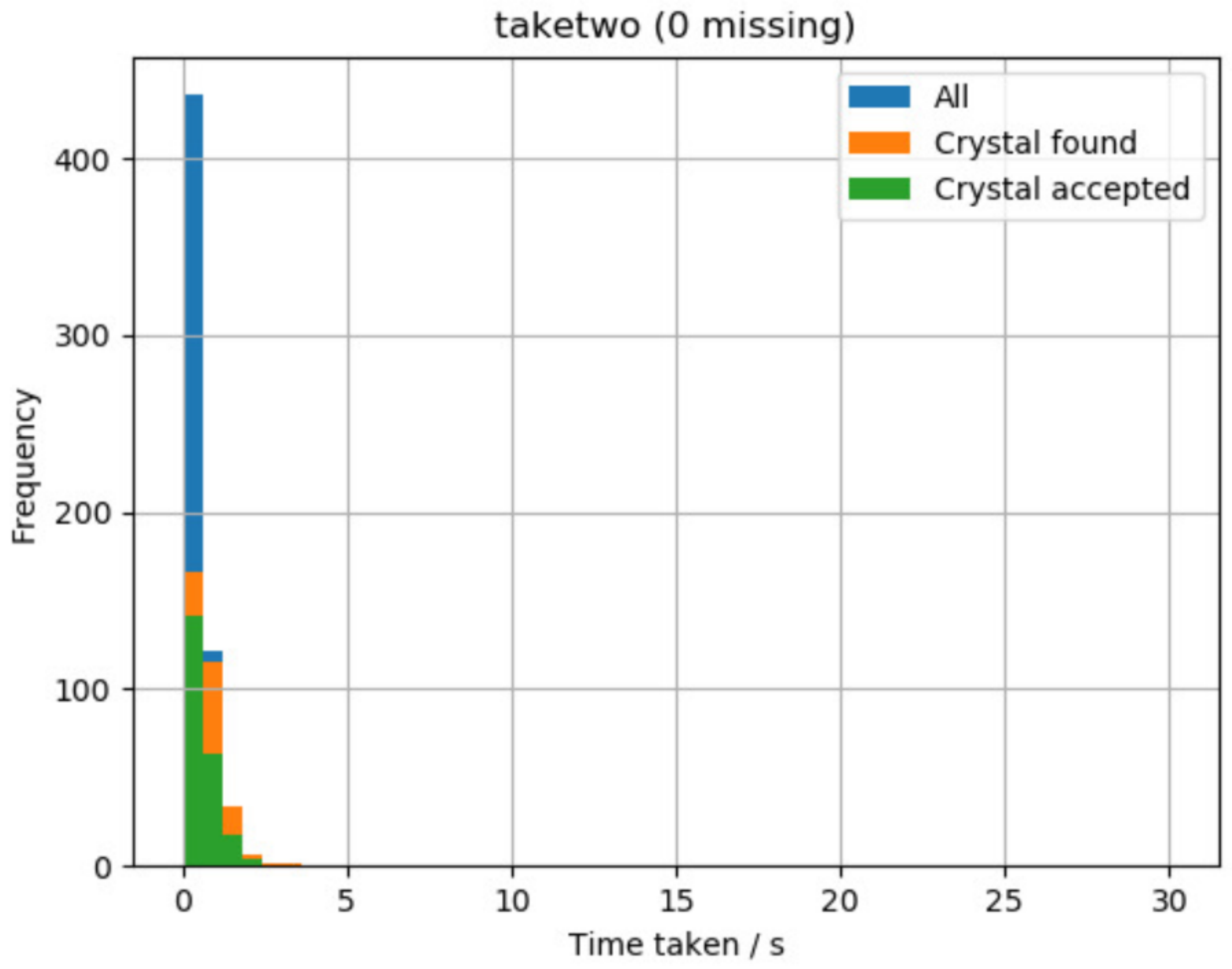
STATISTICS TIP: ALWAYS TRY TO GET DATA THAT'S GOOD ENOUGH THAT YOU DON'T NEED TO DO STATISTICS ON IT

Image: XKCD (2400)

Towards real-time data processing

Remove bottlenecks, "make it go faster", e.g. by algorithm tuning:





Further topics

Are there better ways to write code?

- Better level of abstraction
- Consider "old" computer science concepts,
  e.g. functional or declarative programming style

Making code part of the scientific record:

- Code vs. written methods section
- Notebook-oriented user interface
  (... for large batch processes?)
- Recognition of code as important scientific output
- Institutional software repository and DOIs for code

Summary / take "home" points

To me, "The Digital Scientific Method" is about how we can use computers to improve the reliability and efficiency of our science.

Carefully considered user interfaces can make a big difference

- Embed domain-specific knowledge so that not everyone has to learn it
  (.... reduce the chance to make mistakes ....)
- Reach into the "guts" of the data processing software
- Close feedback loops in data processing workflow

Real-time processing will make a huge difference

- Speed
- Don't store raw data
- Better situational awareness during experiment
- Faster diagnosis of problems
- Less scope for selecting "nice" results ——> more reliable science