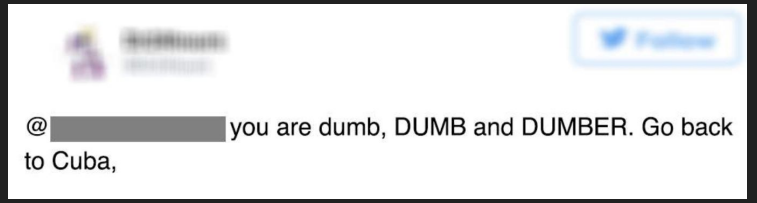
Software engineering in research

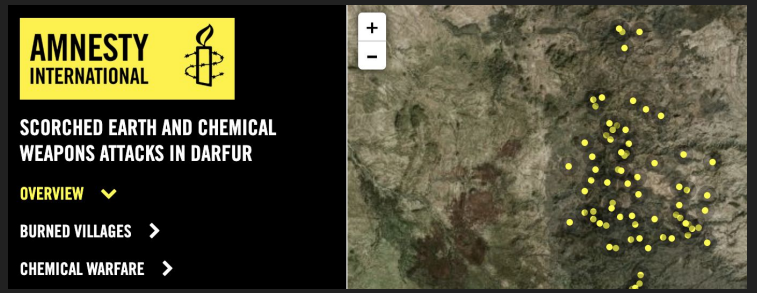
ELEMENT AI

Finding abuse

Spotting destruction

Tracking objects of violence





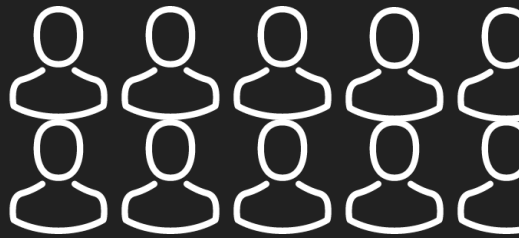
Data analysis pipeline

Data access API

Data collection software

Data analysis pipeline

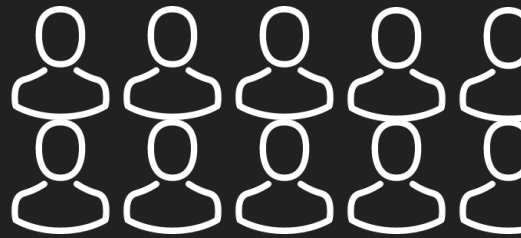
Data access API



Data collection software

Data analysis pipeline

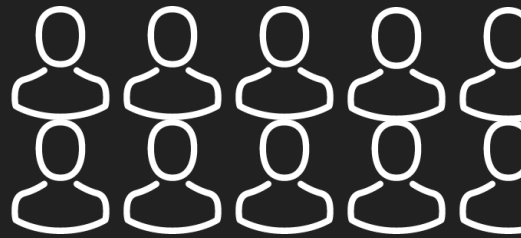
Data access API



Data collection software

Data analysis pipeline

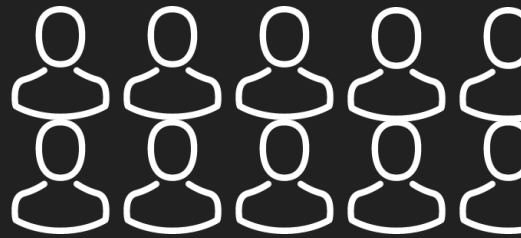
Data access API



Data collection software

Data analysis pipeline

Data access API



Data collection software

So how does this change how we write this software?

Depending on how software is used, we want to change how it is accessed.

1. For experiment setups specific to hardware, code needs to be templated.
2. For standardised data ingestion pipelines, code needs to be installable.
3. For shared statistical analysis pipelines, code just needs to be readable - although novel methods should be installable.

This is because these things map onto different classes of software.

1. Experiments and data collection can be considered an application
2. Data access and standardised statistical analysis methods are libraries
3. Data analysis is an “artifact” of the data and the code

Some concrete examples

1. A web application for collecting labels from volunteers
2. The python BIDS tools (github.com/bids-standard/pybids)
3. A library such as Scikit-Learn
4. A paper / R Markdown document

Putting code on the internet is often not enough

To make a template data collection app, you can use cookiecutter to create lab- or department-wide templates.

To make analysis tools available, you can package them up as installable libraries

To make your analysis output available, you can publish your notebooks.

Putting code on the internet is always the first step :)

git

Some terminology

A git repository is a “software project” that can be distributed

A git commit is the unit of change on your software

A git branch is a separate history of changes to your software

Merges and pull requests are ways of putting together different histories

So how does it work in practice?

GitHub

GitHub is effectively a (nice) online interface to a git repository.

It’s the most popular code sharing platform, but there are many others as well.

So how does it work in practice?