

Research and Academic Software Projects at the Institute for Quantitative Social Science

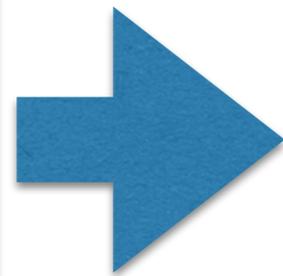
Mercè Crosas, Ph.D.
Chief Data Science and Technology Officer
IQSS, Harvard University
twitter: @mercecrosas web: mercecrosas.com

The Big Picture

Identify a
problem or
need in
research or
academia

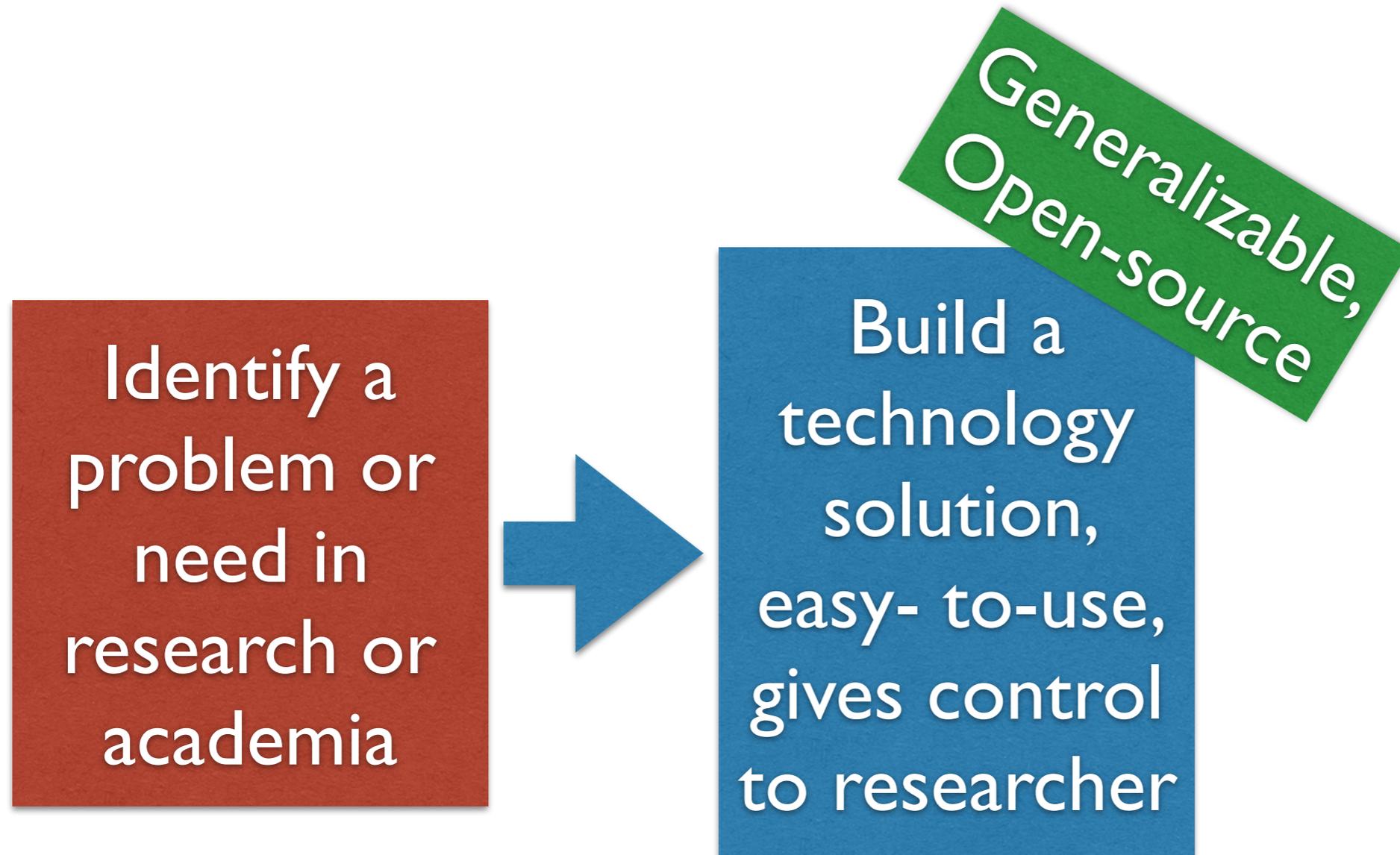
The Big Picture

Identify a
problem or
need in
research or
academia



Build a
technology
solution,
easy- to-use,
gives control
to researcher

The Big Picture



The Big Picture



Example: Dataverse

Example: Dataverse

- How do we increase data sharing to improve research transparency and replication with incentives to researchers?

Example: Dataverse

- How do we increase data sharing to improve research transparency and replication with incentives to researchers?
- Provide a repository solution, where researchers have control of branding and access of their data, and get credit through data citation.

Example: OpenScholar

Example: OpenScholar

- How do we enable scholars to build their academic web sites in a cost effective way?

Example: OpenScholar

- How do we enable scholars to build their academic web sites in a cost effective way?
- Provide a web site builder with pre-set features for academics, where a single hosting serves thousands of sites.

Example: Zelig

Example: Zelig

- How do we simplify using thousands of R statistical methods built by different authors?

Example: Zelig

- How do we simplify using thousands of R statistical methods built by different authors?
- Provide a statistical package that uses the same three commands for all methods, with consistent documentation.

Example: Consilience

Example: Consilience

- How do we make sense of thousands (or millions!) of texts?

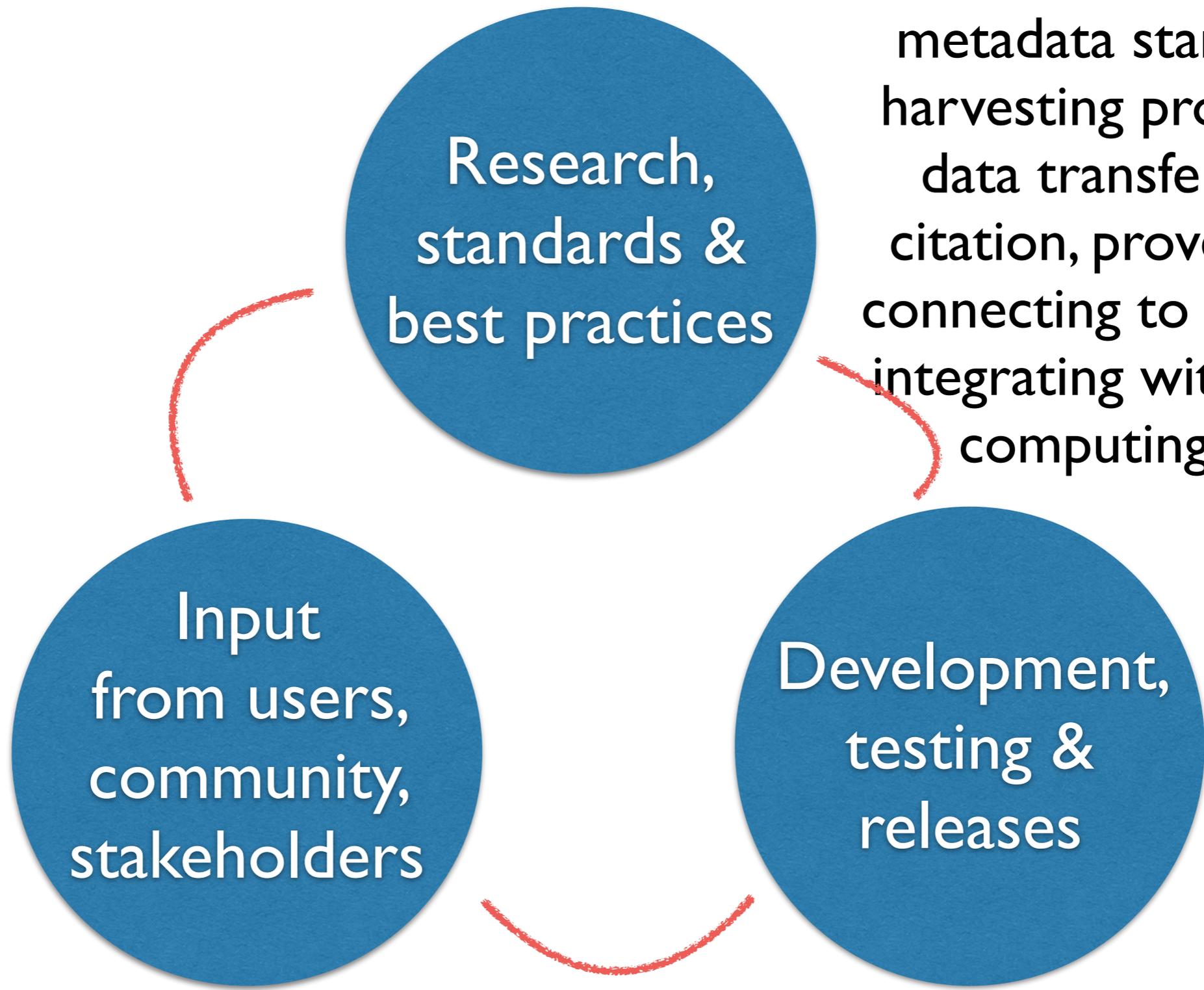
Example: Consilience

- How do we make sense of thousands (or millions!) of texts?
- Provide an application that helps researchers explore many possible ways of categorizing documents.

The Process

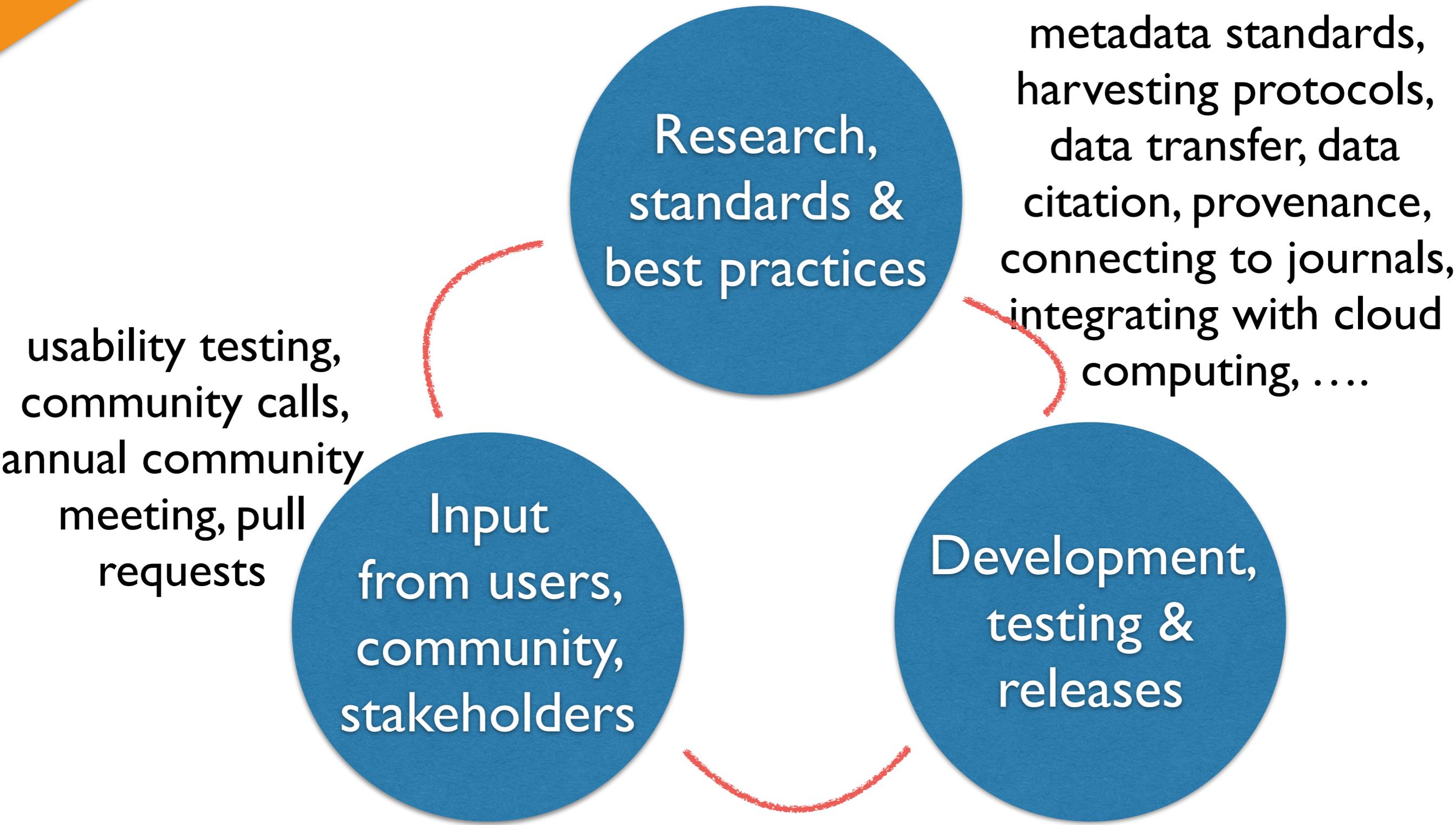


The Process



metadata standards,
harvesting protocols,
data transfer, data
citation, provenance,
connecting to journals,
integrating with cloud
computing,

The Process



Dataverse Case Study

The Process Details



Jenkins

IQSS/dataverse

Filter Board

Backlog 799 108

Ready 5 49

Development 5 39

Code Review 2

QA 7 10

Usability Testing 0

Polishing 0

Done 14 5

3238 Provenance Integration for Files
4.6 - File Replace

3245 Provenance Integration for Datasets
4.6 - File Replace

2003 Customising the Dataverse application is undocumented / hard
4.6 - File Replace
Component: Dataverse General Info
Component: UX & UI Status: Triaged
Type: Suggestion

3391 Guides - Building epub version results in warnings
Component: Documentation

3241 View File Versions
4.6 - File Replace

3116 File Upload: Batch upload problems: Drag and Drop, and "select files" feature is skipping files, zip not unpacking.
4.6 - File Replace
Component: File Upload & Handling
Priority 3: Serious Type: Bug

3352 Add new APIs in support of rsync
Component: File Upload & Handling

3347 Administration Changes to Support rsync
Component: File Upload & Handling

2290 Files: Ability to Replace a File Instead of Deleting
4.6 - File Replace
Component: File Upload & Handling
Component: Publishing & Versions
Component: UX & UI Type: Feature

2465 File: Individual Landing Pages for Files
4.6 - File Replace
Component: File Upload & Handling
Component: UX & UI Type: Feature

3338 support OAuth2 (ORCID) authentication within dataverse
4.6 - File Replace

3307 Harvest: Investigate OAI-PMH

2437 Handles: Restore handle registration functionality in 4.x
Component: Code Infrastructure
Priority: High Status: Triaged
Type: Feature

3146

209 Internationalization
Component: Code Infrastructure
Component: UX & UI Priority: Medium
Status: Triaged Type: Feature

3327

3365 Version: Update version from 4.5 to v4.5.1 just prior to tagging.
4.5.1 - IP Groups fixes
Component: Documentation
Component: Publishing & Versions
Component: UX & UI Status: Dev
Type: Feature

3392

3083 Create Dataset: add support for migrating/harvesting (with existing DOIs) via native API
4.6 - File Replace Component: API
Component: Migration
Effort 2: Medium Type: Feature

3377

3291 Install script fails if postgres master password and user password differ

Usability Testing
Does this actually solve the problem?

Polishing
Any quick changes from Usability testing

3181 Documentation: Resolve build warnings for HTML version of doc
Component: Code Infrastructure
Component: Documentation
Type: Bug

3389

3112 Install Guide: Explain how to configure Google Analytics
Component: Documentation
Type: Suggestion

3393

2905 Footer: Make Copyright Date Dynamic
4.5.1 - IP Groups fixes
Component: UX & UI Effort 1: Small
Priority 1: Low Type: Bug

3380

Dataverse Case Study

The Process Details



Jenkins

An agile process, integrating Waffle + GitHub + Jenkins, including these steps:

Dataverse Case Study

The Process Details



Jenkins

An agile process, integrating Waffle + GitHub + Jenkins, including these steps:

Backlog > Ready > Dev > Code Review > QA > Usability Test > Polishing > Done

The Process Details



The screenshot shows a Jira board for the project 'IQSS/dataverse'. The board is organized into columns representing different stages of the development process: Backlog (799 items), Ready (5 items), Development (5 items), Code Review (2 items), QA (7 items), Usability Testing (0 items), Polishing (0 items), and Done (14 items). A red arrow points to a callout box labeled 'Pull Requests' which is positioned over the Code Review column. The board contains various issues with details such as titles, components, priorities, and statuses.

An agile process, integrating Waffle + GitHub + Jenkins, including these steps:

Backlog > Ready > Dev > Code Review > QA > Usability Test > Polishing > Done

**Not only Best Practices in
Process, but also in Coding**

Not only Best Practices in Process, but also in Coding

Best Practices for Scientific Computing



Greg Wilson , D. A. Aruliah, C. Titus Brown, Neil P. Chue Hong, Matt Davis, Richard T. Guy, Steven H. D. Haddock, Kathryn D. Huff, Ian M. Mitchell, Mark D. Plumbley, Ben Waugh, Ethan P. White, Paul Wilson

Published: January 7, 2014 • <http://dx.doi.org/10.1371/journal.pbio.1001745>

Not only Best Practices in Process, but also in Coding

Best Practices for Scientific Computing



Greg Wilson , D. A. Aruliah, C. Titus Brown, Neil P. Chue Hong, Matt Davis, Richard T. Guy, Steven H. D. Haddock, Kathryn D. Huff, Ian M. Mitchell, Mark D. Plumbley, Ben Waugh, Ethan P. White, Paul Wilson

Published: January 7, 2014 • <http://dx.doi.org/10.1371/journal.pbio.1001745>

- I. Write programs for people, not computers.

Not only Best Practices in Process, but also in Coding

Best Practices for Scientific Computing



Greg Wilson , D. A. Aruliah, C. Titus Brown, Neil P. Chue Hong, Matt Davis, Richard T. Guy, Steven H. D. Haddock, Kathryn D. Huff, Ian M. Mitchell, Mark D. Plumbley, Ben Waugh, Ethan P. White, Paul Wilson

Published: January 7, 2014 • <http://dx.doi.org/10.1371/journal.pbio.1001745>

1. Write programs for people, not computers.
2. Let the computer do the work.

Not only Best Practices in Process, but also in Coding

Best Practices for Scientific Computing



Greg Wilson , D. A. Aruliah, C. Titus Brown, Neil P. Chue Hong, Matt Davis, Richard T. Guy, Steven H. D. Haddock, Kathryn D. Huff, Ian M. Mitchell, Mark D. Plumbley, Ben Waugh, Ethan P. White, Paul Wilson

Published: January 7, 2014 • <http://dx.doi.org/10.1371/journal.pbio.1001745>

1. Write programs for people, not computers.
2. Let the computer do the work.
3. Make incremental changes.

Not only Best Practices in Process, but also in Coding

Best Practices for Scientific Computing



Greg Wilson , D. A. Aruliah, C. Titus Brown, Neil P. Chue Hong, Matt Davis, Richard T. Guy, Steven H. D. Haddock, Kathryn D. Huff, Ian M. Mitchell, Mark D. Plumbley, Ben Waugh, Ethan P. White, Paul Wilson

Published: January 7, 2014 • <http://dx.doi.org/10.1371/journal.pbio.1001745>

1. Write programs for people, not computers.
2. Let the computer do the work.
3. Make incremental changes.
4. Don't repeat yourself (or others).

Not only Best Practices in Process, but also in Coding

Best Practices for Scientific Computing



Greg Wilson , D. A. Aruliah, C. Titus Brown, Neil P. Chue Hong, Matt Davis, Richard T. Guy, Steven H. D. Haddock, Kathryn D. Huff, Ian M. Mitchell, Mark D. Plumbley, Ben Waugh, Ethan P. White, Paul Wilson

Published: January 7, 2014 • <http://dx.doi.org/10.1371/journal.pbio.1001745>

1. Write programs for people, not computers.
2. Let the computer do the work.
3. Make incremental changes.
4. Don't repeat yourself (or others).
5. Plan for mistakes.

Not only Best Practices in Process, but also in Coding

Best Practices for Scientific Computing



Greg Wilson , D. A. Aruliah, C. Titus Brown, Neil P. Chue Hong, Matt Davis, Richard T. Guy, Steven H. D. Haddock, Kathryn D. Huff, Ian M. Mitchell, Mark D. Plumbley, Ben Waugh, Ethan P. White, Paul Wilson

Published: January 7, 2014 • <http://dx.doi.org/10.1371/journal.pbio.1001745>

1. Write programs for people, not computers.
2. Let the computer do the work.
3. Make incremental changes.
4. Don't repeat yourself (or others).
5. Plan for mistakes.
6. Optimize software only after it works correctly.

Not only Best Practices in Process, but also in Coding

Best Practices for Scientific Computing



Greg Wilson , D. A. Aruliah, C. Titus Brown, Neil P. Chue Hong, Matt Davis, Richard T. Guy, Steven H. D. Haddock, Kathryn D. Huff, Ian M. Mitchell, Mark D. Plumbley, Ben Waugh, Ethan P. White, Paul Wilson

Published: January 7, 2014 • <http://dx.doi.org/10.1371/journal.pbio.1001745>

1. Write programs for people, not computers.
2. Let the computer do the work.
3. Make incremental changes.
4. Don't repeat yourself (or others).
5. Plan for mistakes.
6. Optimize software only after it works correctly.
7. Document design and purpose, not mechanics.

Not only Best Practices in Process, but also in Coding

Best Practices for Scientific Computing



Greg Wilson , D. A. Aruliah, C. Titus Brown, Neil P. Chue Hong, Matt Davis, Richard T. Guy, Steven H. D. Haddock, Kathryn D. Huff, Ian M. Mitchell, Mark D. Plumbley, Ben Waugh, Ethan P. White, Paul Wilson

Published: January 7, 2014 • <http://dx.doi.org/10.1371/journal.pbio.1001745>

1. Write programs for people, not computers.
2. Let the computer do the work.
3. Make incremental changes.
4. Don't repeat yourself (or others).
5. Plan for mistakes.
6. Optimize software only after it works correctly.
7. Document design and purpose, not mechanics.
8. Collaborate.

Impact at Harvard

Open**Scholar**[®]

6,833

OpenScholar sites created

13,904

Registered users

75,378

Publications posted

24

Academic departments

Impact at Harvard



243 Dataverses from Harvard affiliates

1,226 Datasets by Harvard affiliates as authors

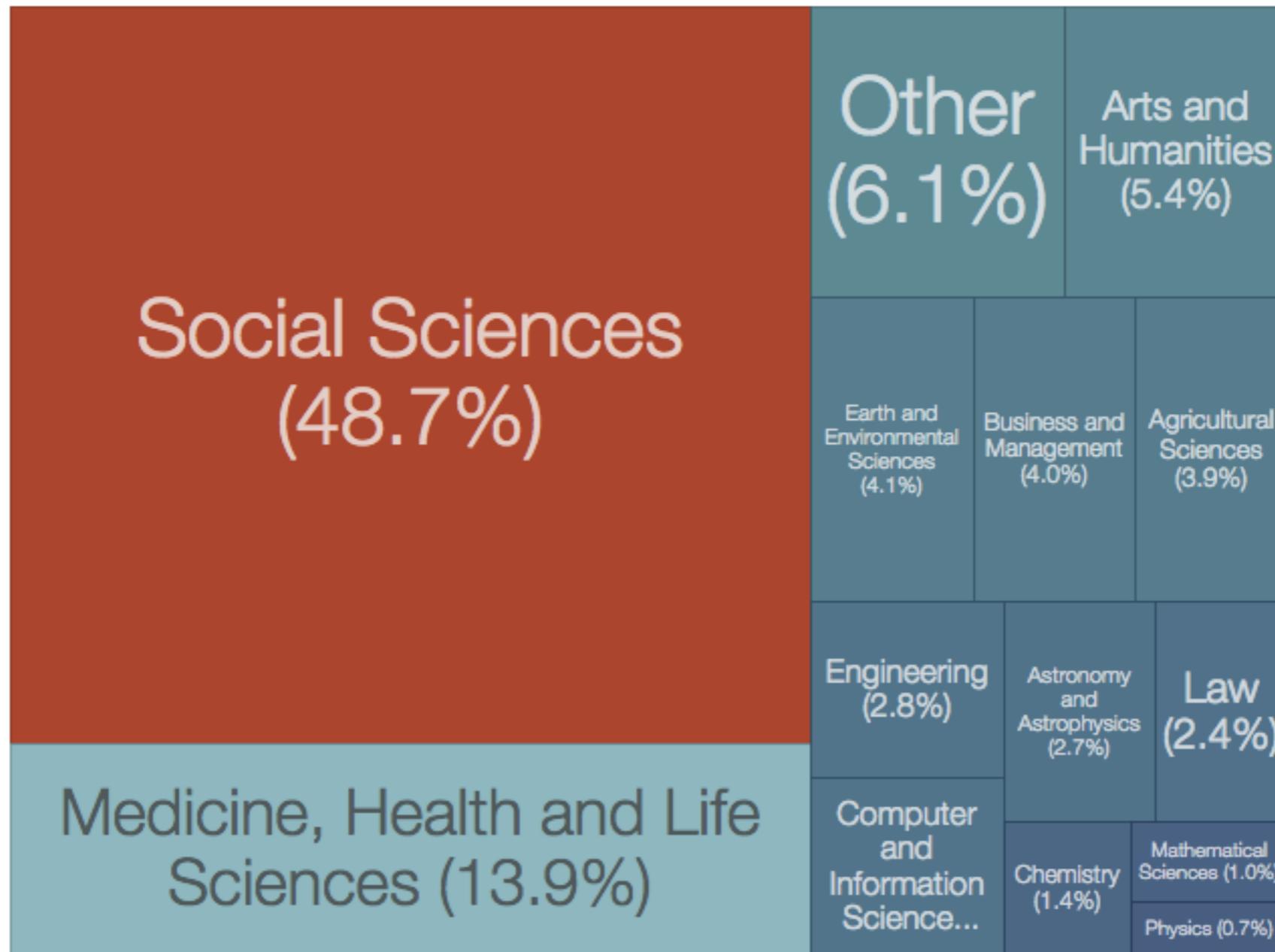
1,427 Registered Harvard users

Broader Impact

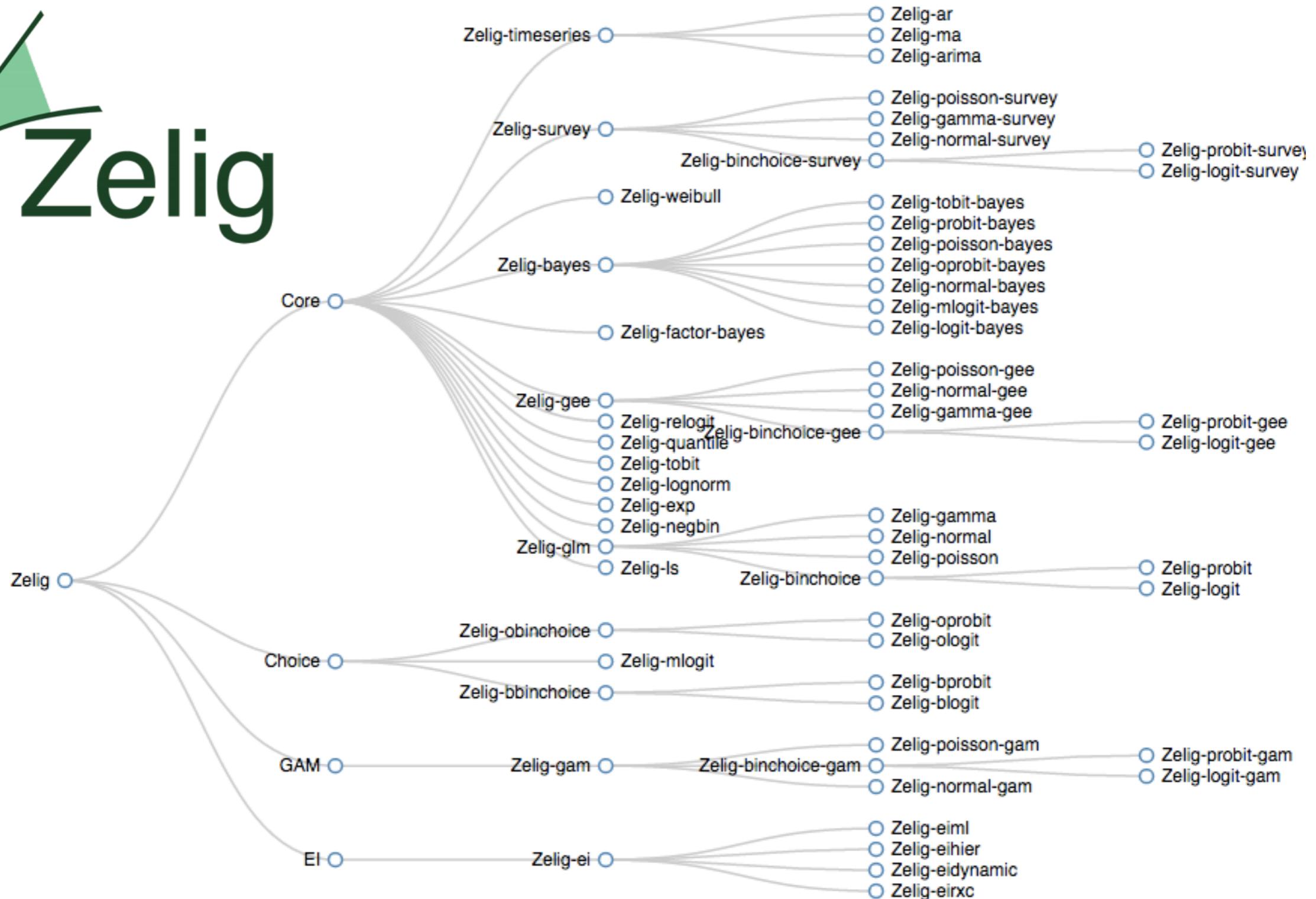
Dataverses by Category



Datasets by Subject



53 Stats Models, easy-to-use



Thank you!



iq.harvard.edu



dataverse.org
dataverse.harvard.edu



zeligproject.org



openscholar.org
projects.iq.harvard.edu
scholar.harvard.edu



Coming soon!