Incorporating Software Curation into Research Data Management Services: Lessons Learned

Fernando Rios
University of Arizona Libraries
Office of Digital Innovation & Stewardship
@riosfrnd

IDCC 2018
February 19-22, Barcelona
"We should probably think about including research software into our RDM services"

"Hmmm... we have no idea where to start or what exactly we should be looking at"

"Alright, let’s try a few things and see what happens"
Software is different?

Can we treat software like data?

Yes. But...

“we are more interested in what software does rather than what software is.” (Matthews et al., 2010)

Software as data:

- Functionality: execution
- Relationships: workflows, dependencies
- Other aspects: code repositories, intellectual property
- Attribution: software visibility is diminished
Software in RDM

Research Software

http://www.dcc.ac.uk/resources/curation-lifecycle-model

Jones et al. (2013) How to Develop Research Data Management Services - a guide for HEIs
Software in RDM – Plain English version

Consulting
- Best practices
- Funder & publisher reqs.
- IP
- Repository selection
- Metadata
- DMP

Archiving
- Institutional data repo
- Dataset curation

Education
- Workshops
- 1-on-1 training
- Online training
- Web resources

How does software fit?
Developing Services -- Overview

Rapid prototyping, Iterative approach

Constraints:
- Build it into existing services
- Do it with existing tech infrastructure.
- Keep extra work minimal

Sustainability, Scalability
Knowledge-building

What’s out there? What are people doing to preserve software?

Literature review
- Software preservation
- Sharing
- Software engineering
- Citation & credit
- Metadata
- Intellectual property

Science/research + cultural heritage viewpoints


Knowledge-building + planning took most time
Planning

Publishing research openly as a condition of funding

Re-analyzing data in light of new theories

Promoting good software

Rios (2016) The pathways of research software preservation
Planning outcome

- Expand Archiving workflow
  - Treat software separately from data in the JHU Data Archive
  - Scope: Only software produced as part of the research. Limited curation: Don’t try to capture everything, do full re-execution.

- Expand consulting expertise
  - Elevate team knowledge on research software in RDM
  - Scope: focus on the fundamentals, funder reqs, publisher reqs/offerings

- Education
  - Develop educational resources related to research software
  - Scope: Audience is campus researchers. Mix of in-person training and online modules related to the above.
Sharing in the JHU Data Archive

- How should archiving workflows be modified to include software?
- How can software be made more visible in the JHU Data Archive?
- How should it be curated?

- Describe the most general level which still allows for replication or reuse
- Include enough info for others to replicate/reuse
- Include enough info for others to give credit
Mediated deposit – no major workflow changes. Curation: collect SW metadata, organize datasets for better software visibility (where appropriate), check for docs, link data, paper, and code

Lesson: biggest change was in thinking of software as something worthy of separate consideration.
What does it look like?


Description
The Martian polar atmosphere is known to have a potential vorticity (PV) near the winter pole, with a finding is surprising since an isolated band of PV is going back to Rayleigh. Here we investigate the use of numerical integrations of the rotating shallow mode of instability and its growth rate depends on annulus. By introducing thermal relaxation toward a time scale similar to that of the instability, we are annular vortex with similar characteristics as that of the atmosphere. This time scale, typically 0.5-2 so, scales for Mars’ polar atmosphere. We also discuss

How to Cite

Software Title

License
MIT license

Description
‘BOB’ shallow water model code, along with model output and analysis code used to study the dynamics of Mars’ polar vortex. Codes are available in 1_Model.zip file. It is based on Git-hub commit 4a27f/d3. See the ‘Readme’ file in 1_Model.zip for details.

Artifact Type
Source Code

Data Published
2017-02

Code Repository Link
https://github.com/wseviour/sxwbob_mars

Programming Language
Fortran, Python

Function
Simulation / modeling

Interaction Method
Programmatic

Software Contributors
Seviour, William J. M. Developer
Waugh, Darryn W. Principal Investigator
Scott, Richard K. Developer
Consulting – improving quality

Recommend best practices

- Version control (git, SVN, etc.)
- Code organization & style
- What to document
- Comments – Focus on “what” and “why”
- Use open tools
- Licensing

Advise on reproducibility

- Availability: Source code is better than nothing.
- GitHub ≠ archive!
- Capture workflow (& dependencies if possible) – Emulation, virtualization, containers

Explain cost/benefits

- Code + data + paper = more impact
- Get credit for all parts
- YOU will be the first to benefit
  - Save time: easily reuse
  - Build on prior research
  - New students

www.dcc.ac.uk/webfm_send/2398
Lesson: Helping with making educational material was deemed by the team as the most useful way to learn the material themselves.
Other Lessons

- There is a need for educational resources from RDM professionals around research software
  - Good response from colleagues and social media on the software modules

- Providing more complete curation and preservation out of reach for many library-based RDM services
  - e.g., quality eval, checking results, enable re-execution
  - Labor intensive, cyberinfrastructure costs
Future

- Refine metadata
  - Software-related metadata is more mature now. Revisit existing efforts
    - CodeMeta
  - Software Citation Implementation group (FORCE11)

- Identify if lib. should provide more intense curation & what is needed
  - E.g., verify quality, check results, enable re-execution

- How can partners and tools be leveraged for more complete preservation service
  - Capturing dependencies.
  - Enable re-execution over time
Attribution


Document & tag icons: https://thenounproject.com/artworkbean

Fernando Rios
@riosfrnd
Thank you
## Metadata

Surveyed: 11 schemas, 42 fields. Refined to 16 concepts

<table>
<thead>
<tr>
<th>Concept</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SoftwareTitle</td>
<td>Name of the software</td>
</tr>
<tr>
<td>SoftwareIdentifier</td>
<td>E.g., DOI or other PID</td>
</tr>
<tr>
<td>License</td>
<td>Licensing terms.</td>
</tr>
<tr>
<td>SoftwareContributor</td>
<td>Individuals or organizations (developers, programmers, testers, etc.)</td>
</tr>
<tr>
<td>ContributorsId</td>
<td>A PID associated with each contributor (e.g., email address)</td>
</tr>
<tr>
<td>Description</td>
<td>Human-readable synopsis</td>
</tr>
<tr>
<td>datePublished</td>
<td>Date this software was published</td>
</tr>
<tr>
<td>Version</td>
<td>Software version. Some software (e.g., one-off programs) may not require a version number</td>
</tr>
<tr>
<td>ObjectType</td>
<td>The type of software artifact being described (source code, executable, etc.)</td>
</tr>
<tr>
<td>ProgrammingLanguage</td>
<td>The programming language used.</td>
</tr>
<tr>
<td>CodeRepositoryLink</td>
<td>Link to software repository where software is developed (e.g., GitHub, SourceForge, etc.)</td>
</tr>
<tr>
<td>RelatedObjects</td>
<td>Related papers, website, documentation, other software (except dependencies).</td>
</tr>
<tr>
<td>Dependencies</td>
<td>Hardware (e.g., instruments, CPU, etc.), operating system, environment (e.g., Python 2.7, etc.) libraries/other SW.</td>
</tr>
<tr>
<td>InteractionMethod</td>
<td>Function</td>
</tr>
</tbody>
</table>
| GUI, command line, etc. | General functionality (plotting, analysis, data collection, simulation, etc.) | Inputs and outputs.
What is research software?

Research software – software that supports research claims

Building blocks

- Excel
- R
- MySQL
Consulting

Recommend best practices

- Version control (git, SVN, etc.)
- Code organization & style
- What to document
- Comments – Focus on "what" and "why"
- Use open tools
- Licensing

Advise on reproducibility

- Availability: Source code is better than nothing.
- GitHub ≠ archive!
- Capture workflow (& dependencies if possible) – Emulation, virtualization, containers

Explain cost/benefits

- Code + data + paper = more impact
- Get credit for all parts
- YOU will be the first to benefit
  - Save time: easily reuse
  - Build on prior research
  - New students

Lesson learned: it's an avalanche of information