RDA Global Adoption week
15 - 19 June 2020
The RDA Global Adoption Week: 15-19 June 2020
focused on five areas of the research data lifecycle

<table>
<thead>
<tr>
<th>Day &amp; Topic</th>
<th>Sessions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monday, 15th June 2020 - Data Management Planning</td>
<td>14:00 UTC + 23:00 UTC</td>
</tr>
<tr>
<td>Tuesday, 16th June 2020 - Data Description</td>
<td>06:00 UTC + 14:00 UTC</td>
</tr>
<tr>
<td>Wednesday, 17th June 2020 - Identify, Store and Preserve</td>
<td>07:00 UTC + 14:00 UTC</td>
</tr>
<tr>
<td>Thursday, 18th June 2020 - Disseminate, Link and Find</td>
<td>07:00 UTC + 12:00 UTC</td>
</tr>
<tr>
<td>Friday, 19th June 2020 - Policy, Legal Compliance and Capacity</td>
<td>05:00 UTC + 13:00 UTC</td>
</tr>
</tbody>
</table>
Originally planned for the RDA 15th Plenary, the Adoption Week aims to demonstrate the wide variety of RDA adoptable and adopted solutions to data sharing challenges across research practices, domains and geographies.

Purpose of the week:
- Learn about RDA Outputs
- Converse with speakers from all around the world who have created and implemented them
- Determine how best to integrate those data sharing solutions into your own projects
• RDA Outputs are classified as **RDA Recommendations** (official, endorsed results of RDA Groups), **Supporting Outputs** (useful solutions from our RDA Working and Interest Groups) or **other Outputs**

• They can be searched according to their status, **Data Life Cycle topics** or **scientific domain**

rd-alliance.org/recommendations-and-outputs/catalogue
Tell your adoption story

• Are you an adopter? RDA is actively seeking new adoption stories to inspire the further uptake of RDA outputs.

• Submit your story here: https://www.rd-alliance.org/tell-your-rda-adoption-story
CODATA Data Science Journal CfP

• RDA special collection themes:
  o Results produced by an IG or WG;
  o Description of an Adoption Case outlining how a specific recommendation or output has been implemented;
  o Other types of work related to RDA activities.

• RDA Europe 4.0 still has funds available for the publication of articles in DSJ

• Open to all interested applicants regardless of their geographical provenance.

• Deadline 17 July
Thursday 18th June
07:00 UTC

1. Data Discovery Paradigms IG
   • Survey on the practices in data search services
     Mingfang Wu (ARDC)
   • Eleven quick tips and User requirements and recommendations
     Fotis Psomopoulos (INAB CERTH)

2. FAIR data maturity model: specification and guidelines
   Keith Russell (ARDC)

   Recommendation - Introducing Maneage: customizable framework for managing data lineage
   Mohammad Akhlaghi (IAC)

Followed by Q&A
Investigate what data search systems and ranking models have been deployed.

Serve as a benchmark to be looked back on in future to assess how much and in what ways data search has improved.

Identify potential collaborative projects from the Survey.
1. What are characteristics of each repositories? (5)
2. What are system configurations (e.g., ranking model, index methods, query methods)? (7)
3. What are evaluation methods and benchmark? (10)
4. What methods have been used to boost search-ability to web search engines? (2)
5. What other technologies or system configurations have been employed? (5)
6. Wish list for future activities for the RDA relevance task force (2)
Participants background

N = 98

- Life Science: 35%
- Earth Science: 34%
- Social Science: 28%
- Computer Science: 3%
- Physical Science: 17%
- Health Science: 12%
- All Domains: 16%
Survey result highlights ...
Data repositories use common search systems

- Google Custom Search, 4%
- MarkLogic, 1%
- Elastic Search, 20%
- Lucene, 18%
- Solr, 31%
- I don't know, 10%

N = 96

- I Don't know, 54%
- TF-IDF, 12%
- Language Model, 1%
- Okapi BM25 Model, 3%
- Vector Space Model, 6%
- Others, 29%

N = 90
Open source and available skills are top reasons for choosing a search system
Majority didn’t conduct any kind of evaluations

- Created test collection: 9
- Informal evaluation: 11
- Log analysis: 6

No performance measure was provided
Repositories desire guidelines for improving relevancy ranking in their data search system, with small repositories having the greatest need. Repositories understand that their search systems need to be evaluated and improved, but often lack the resources (time and/or expertise) to explore and evaluate the available options. The study concludes that there is an opportunity for people working in the search space to collaborate, to build test collections and other efforts that offer the greatest improvements in search services at the lowest cost.

Khalsa, SiriJodha; Cotroneo, Peter; Wu, Mingfang (2018), “A survey of current practices in data search services”, Mendeley Data, v1 http://dx.doi.org/10.17632/7j43z6n22z.1
Thank you

Contact:
mingfang.wu@ardc.edu.au
sjsk@nsidc.org
fpsom@certh.gr
Helping to make research data **Findable** to support users in discovering data.
DDP Interest Group: Objective

- Provide a forum where representatives across the spectrum of stakeholders and roles can explore how to improve data discovery.
- Produce actionable recommendations for data producers, data repositories, data services providers and data seekers.
Tip 1: Think about the data you need and why you need them.
Tip 2: Select the most appropriate resource.
Tip 3: Construct your query strategically.
Tip 4: Make the repository work for you.
Tip 5: Refine your search.
Tip 6: Assess data relevance and fitness-for-use.
Tip 7: Save your search and data-source details.
Tip 8: Look for data services, not just data.
Tip 9: Monitor the latest data.
Tip 10: Treat sensitive data responsibly.
Tip 11: Give back (cite and share data).

Output I - Eleven quick tips for finding research data

(8124 views, 2345 downloads)
Output 2 - User Requirements for a data repository

Nine requirements (from 79 use cases)

- Indication of data availability
- Connection of data with person/institution/paper/citations/grants
- Fully annotated data
- Filtering of data based on specific criteria on multiple fields at the same time
- Cross-referencing of data
- Visual analytics/inspections of data/thumbnail preview
- Sharing data in a collaborative environment
- Accompanying educational/training material
- Portal functionality similar to other established academic portals

Data repository operators can use the requirements for the following purposes:

- As a checklist for designing and implementing a data service portal.
- For existing data discovery services, the list of requirements can be used as guidelines for heuristic evaluation of a specific data discovery service, and therefore plan for future improvements when necessary.
- In the era of big data, research on data discovery paradigms is at an all-time high. A user’s perspective provides a strong foundation on which to construct the paradigms of the future.
Output 2 - Recommendations for Data Repositories to make data discovery

Recommendations:
- Multiple query interfaces
- Multiple access points
- Accessible search results
- Readable and analysable metadata records
- Available bibliographic references
- Available data usage statistics
- Consistent interface
- Identifiable duplicates
- Findable from web search engines
- Interoperability with other repositories

Data repositories can take the ten recommendations:
- As guidelines when implementing a new repository
- As a checklist when conducting heuristic evaluation of an existing repository.

Data repositories can implement all or prioritise their implementation based on their user needs and available resources.

Use cases published to Zenodo
https://doi.org/10.5281/zenodo.1050976 (124 views, 73 downloads)
### Output 2 - User Requirements and Recommendations for Data Repositories

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Data Availability</th>
<th>Connection of Data</th>
<th>Annotation</th>
<th>Filtering</th>
<th>Cross-referencing</th>
<th>Inspection of Data</th>
<th>Similarity across portals</th>
<th>Training material</th>
</tr>
</thead>
<tbody>
<tr>
<td>REC 1: Query interfaces</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>REC 2: Multiple access points</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>REC 3: Summarize search results</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>REC 4: Metadata records readable</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>REC 5: Bibliographic references</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>REC 6: Usage statistics</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>REC 7: Consistency</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>REC 8: Identify duplicates</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>REC 9: Findability from web SEs</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>REC 10: Interoperability</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>

Support data searches from web search engines

The Fair Data Principles

---

Contact

fpsom@certh.gr
mingfang.wu@ardc.edu.au
sjsk@nsidc.org

https://www.rd-alliance.org/groups/data-discovery-paradigms-ig
Adoption of the FAIR Data Maturity Model

18 June 2020
The principles are NOT strict

- Ambiguity
- Wide range of interpretations of FAIRness

Different FAIR Assessment Frameworks

- Different metrics
- No comparison of results
- No benchmark

**SOLUTION** is to bring together stakeholders to build on existing approaches and expertise

- Set of core assessment criteria for FAIRness
- FAIR data maturity model & toolset
- FAIR data checklist
- RDA recommendation

Join the RDA Working Group: [RDA WG web page](https://www.rd-alliance.org) | [GitHub](https://github.com)

16/06/2020
Public review period complete now to council


THANKS TO ALL REVIEWERS

3600+ page views

14 comments
Adoption examples
Early adopters – Experience sharing

• Ge Peng | NOAA
• Anusuriya Devaraju | FAIRsFAIR

... will share their relevant experience with regard to the adoption of the FDMM and answer to the following questions:

1. What is the level of adoption at your organisation? (E.g., pilot, production, ...)
2. Do you plan to continue to use the Recommendation?
3. Did you need to modify the Recommendation for your use?
4. Can you give an estimate of how much time / effort you have spent on the adoption so far?
5. What’s your overall experience? (E.g., Very Good, Good, Fair, Poor)
6. Would you do it again?
Evaluating the FAIRness of Environmental Data
– Application of the RDA FAIR Data Maturity Indicators

Ge Peng, PhD
Cooperative Institute for Satellite Earth System Studies (CISESS) Between
U.S. National Oceanic and Atmospheric Administration (NOAA) and North Carolina State University
at NOAA National Centers for Environmental Information (NCEI)

#9 Workshop of the RDA FAIR Data Maturity Model Working Group, May 20–21, 2020
Purposes of Pilot Application

- Examine the relevancy of the RDA FAIR DMIs (v0.04)
- Baseline the FAIRness of NCEI managed data
  - In particular, OneStop-Ready datasets,
    - OneStop project was initiated in 2015 to improve discovery and access services for NOAA datasets.
  - What worked?
- Identify potential gaps & define path forward in NCEI data sharing practices
Adopting OAIS RM & DSMM Helped!

Mapping FAIR Data Principles to NCEI/CICS-NC Data Stewardship Maturity Matrix (DSMM)

<table>
<thead>
<tr>
<th>FAIR Data Principles</th>
<th>DSMM Key Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Wilkinson et al. 2016)</td>
<td>Preservability</td>
</tr>
<tr>
<td>F1. (meta)data are assigned a globally unique and eternally persistent identifier</td>
<td>L3</td>
</tr>
<tr>
<td>F2. data are described with rich metadata (defined by R1 below)</td>
<td>L3</td>
</tr>
<tr>
<td>F3. metadata clearly and explicitly include the identifier of the data it describes</td>
<td>L3</td>
</tr>
<tr>
<td>F4. (meta)data are registered or indexed in a searchable resource</td>
<td>L2 &amp; L3</td>
</tr>
<tr>
<td>A1. (meta)data are retrievable by their identifier using a standardised communications protocol</td>
<td>L2</td>
</tr>
<tr>
<td>A1.1. the protocol is open, free, and universally implementable</td>
<td>L3</td>
</tr>
<tr>
<td>A1.2. the protocol allows for an authentication and authorization procedure, where necessary</td>
<td>L3</td>
</tr>
<tr>
<td>A2. metadata are accessible, even when the data are no longer available</td>
<td>L2</td>
</tr>
<tr>
<td>I1. (meta)data use a formal, accessible, shared, and broadly applicable language for knowledge representation</td>
<td>L3</td>
</tr>
<tr>
<td>I2. (meta)data use vocabularies that follow FAIR principles</td>
<td>L4</td>
</tr>
<tr>
<td>I3. (meta)data include qualified references to other (meta)data</td>
<td>L3</td>
</tr>
<tr>
<td>R1. (meta)data are richly described with a plurality of accurate and relevant attributes</td>
<td>L3</td>
</tr>
<tr>
<td>R1.1. (meta)data are released with a clear and accessible data usage licence</td>
<td>L3</td>
</tr>
<tr>
<td>R1.2. (meta)data are associated with detailed provenance</td>
<td>L3</td>
</tr>
<tr>
<td>R1.3. (meta)data meet domain-relevant community standards</td>
<td>L3</td>
</tr>
</tbody>
</table>

* Can be easily implemented via relevant metadata entity and modified document template

Many data stewardship quality attributes are not explicitly addressed by the FAIR Data Principles.

- Most of data are open by default,
- Use agreements or use constraints,
- CC license not yet explicitly included.
Path Forward

Improving the FAIRness of NCEI & NOAA Data

- Explicitly include a data usage license, e.g. CC-BY 4.0; CC0, in the metadata record:
  - Discussions are on-going,
  - Procedure is under development.

Extending the Application Scope – under discussion

- Assess: 200+ additional NCEI datasets,
  - produced by NCEI’s Center for Weather and Climate, various stages of OneStop-ready.
- Examine the scalability of the evaluation.

Integrating Assessment Results - Fairly

- Community guidelines – consistently curating and representing dataset quality information,
- Virtual workshop on July 13, 2020 – bringing together international domain experts,
- Contact me at gpeng@ncsu.edu if interested in participating or contributing.
Repository Certification

- CoreTrustSeal follows a self-assessment and peer review model
- FAIRsFAIR is offering support with a CoreTrustSeal+FAIR angle
- Map object characteristics to where repositories can enable FAIR
Repository Certification

- CoreTrustSeal follows a self-assessment and peer review model
- FAIRsFAIR is offering support with a CoreTrustSeal+FAIR angle
- Map object characteristics to where repositories can enable FAIR

Later:
- Integrate object evaluation outcomes
Overall Adoption Experience

• The recommendation should be used as a starting reference point for data FAIRness assessment.

• Presentation - specification and guidelines are well structured!

• ‘What’ aspect of FAIR assessment
  • Descriptions of indicators are very helpful!
  • Suggestion - Include priority level next to each of the indicators.
  • Essential I-indicators missing (needs further work or not important?)

• ‘How’ aspect of FAIR assessment
  • Context matters (e.g., practices, data types)
  • Assessment details not always provide sufficient detail to implement tests.
  • Potential supporting technologies and services should be described.
Next steps

• Reach out to your communities as for the publishing of the **FAIR data maturity model: specification and guidelines** (i.e. RDA recommendation)

• Continuously provide feedback to the Editorial Team and pass on information with regards to the use of the **FAIR data maturity model: specification and guidelines** (i.e. RDA recommendation)

The editorial team will look into a release calendar and change management schedule

WORKSHOP #10

Possibly **September 2020**
Thank you!
Introducing Maneage:
Customizable framework for managing data lineage

[RDA Europe Adoption grant recipient. Submitted to IEEE CiSE (arXiv:2006.03018), Comments welcome]

Mohammad Akhlaghi
Instituto de Astrofísica de Canarias (IAC), Tenerife, Spain

RDA Global Adoption week
June 18th, 2020

Most recent slides available in link below (this PDF is built from Git commit d1faba6):
Challenges (also relevant to researchers, not just repositories)

- **Bi-directional linking**: how to link data and publications.
- **Software management**: how to manage, preserve, publish and cite software?
- **Metrics**: how often are data used.
- **Incentives to researchers**: how to communicate benefits of following good practices to researchers.

“We would like to see a workflow that results in all **scholarly objects** being connected, linked, citable, and persistent to allow researchers to navigate smoothly and to enable **reproducible research**. This includes **linkages** between documentation, code, data, and journal articles in an integrated **environment**. Furthermore, in the ideal workflow, all of these objects need to be **well documented** to enable other researchers (or citizen scientists etc) to reuse the data for new discoveries.”
General outline of a project (after data collection)

Existing solutions:
- Virtual machines
- Containers (e.g., Docker)
- OSs (e.g., Nix, GNU Guix)

**Software**
- Config environment?
- Config options?
- Dep. versions?
- Dependencies?
- What version?

**Hardware/data**
- Data base, or PID?
- Calibration/version?
- Integrity?

**Build**
- Run software on data
  - Environment update?
  - In sync with coauthors?

**Paper**
- Confirmation bias?
- Human error?
- Runtime options?
- What order?

**History recorded?**
- Cited software?
- Report this info?
- Sync with analysis?

---

**Green boxes** with sharp corners: source/input components/files.
**Blue boxes** with rounded corners: built components.
**Red boxes** with dashed borders: questions that must be clarified for each phase.

https://heywhatwhatdidyousay.wordpress.com
http://pngimages.net
Science is a tricky business

Data analysis [...] is a human behavior. Researchers who hunt hard enough will turn up a result that fits statistical criteria, but their discovery will probably be a false positive.

Founding criteria

Basic/simple principle:

Science is defined by its METHOD, not its result.

- **Complete/self-contained:**
  - Only dependency should be POSIX tools (discards Conda or Jupyter which need Python).
  - Must not require root permissions (discards tools like Docker or Nix/Guix).
  - Should be non-interactive or runnable in batch (user interaction is an incompleteness).
  - Should be usable without internet connection.

- **Modularity:** Parts of the project should be re-usable in other projects.

- **Plain text:** Project’s source should be in plain-text (binary formats need special software)
  - This includes high-level analysis.
  - It is easily publishable (very low volume of $\times 100$KB), archivable, and parse-able.
  - Version control (e.g., with Git) can track project’s history.

- **Minimal complexity:** Occum’s rasor: “Never posit pluralities without necessity”.
  - Avoiding the fashionable tool of the day: tomorrow another tool will take its place!
  - Easier learning curve, also doesn’t create a generational gap.
  - Is compatible and extensible.

- **Verifiable inputs and outputs:** Inputs and Outputs must be automatically verified.

- **Free and open source software:** Free software is essential: non-free software is not configurable, not distributable, and dependent on non-free provider (which may discontinue it in N years).
General outline of a project (after data collection)

- **Software**
  - Build
    - Config environment?
    - Config options?
    - Dep. versions?
    - Dependencies?
  - Repository?
  - What version?
  - Data base, or PID?
  - Calibration/version?
  - Integrity?

- **Hardware/data**
  - Run software on data
  - Environment update?
  - In sync with coauthors?

- **Paper**
  - Confirmation bias?
  - Human error?
  - Runtime options?
  - What order?
  - History recorded?
  - Cited software?
  - Report this info?
  - Sync with analysis?

Green boxes with sharp corners: source/input components/files.
Blue boxes with rounded corners: built components.
Red boxes with dashed borders: questions that must be clarified for each phase.
Predefined/exact software tools

Reproducibility & software

Reproducing the environment (specific software versions, build instructions and dependencies) is also critically important for reproducibility.

- Containers or Virtual Machines are a binary black box.

- Maneage installs fixed versions of all necessary research software and their dependencies.

- Installs similar environment on GNU/Linux, or macOS systems.

- Works very much like a package manager (e.g., apt or brew).
Example: Matplotlib (a Python visualization library) build dependencies

From “Attributing and Referencing (Research) Software: Best Practices and Outlook from Inria” (Alliez et al. 2020, CiSE, DOI: 10.1109/MCSE.2019.2949413).
Advantages of this build system

- Project runs in fixed/controlled environment: custom build of Bash, Make, GNU Coreutils (ls, cp, mkdir and etc), AWK, or SED, \LaTeX, etc.

- No need for root/administrator permissions (on servers or super computers).

- Whole system is built automatically on any Unix-like operating system (less 2 hours).

- Dependencies of different projects will not conflict.

- Everything in plain text (human & computer readable/archivable).
removes the necessity to add further dependencies to start the project. There are high-level language libraries like NumPy which also provide plots. However, the problem is that they require many dependencies (PyQt, Matplotlib, etc.). installing these dependencies from source is not easy and will hinder the reproducibility of your paper. Your work after several years, the binary files of these high level libraries, that you usually install non-dynamically, will no longer be available in common repositories. Therefore building the libraries from source is the only option to reproduce your results.

Furthermore, since PyPI/RPMs is built by RStudio it respects all the properties of your tool (for example line width and font size). Therefore, the final plot looks in your paper much more nicely. It also has a wonderful filename.

This template also defines two RMarkdown macros allowing you to reuse text within your document in your paper. For example, this text has been marked once. If you comment the line by adding a `#` at the start of the line, or simply deleting the line that defines `highlight-white`, then the first line marked `#` will become black (indicated here with the text in the front of this section) and the line marked `highlight-white` will be in the final PDF. You can use these `highlight-white` to easily make copies of your research (by editing abstracts who are pasted into the new project or example) and new co-authors (who don’t want to be disturbed by these issues in their first reading).

2. NOTICE AND CITATIONS
To encourage other scientists to publish similarly reproducible papers, please add a notice above the start of your paper or in your final draft. The notice is set to display all your papers work is fully reproducible.

For this time being, we haven’t written a specific paper for this template. Instead, we would be grateful if you could cite the first paper that start the final version of this template: Astropy Collaboration (2013).

After publication, don’t forget to upload all the necessary data, software source code and the project’s source to a long-lasting host like Zenodo (https://zenodo.org).
General outline of a project (after data collection)

**Software**
- Build
- Config options?
- Config environment?
- Dep. versions?
- Dependencies?
- What version?
- Repository?

**Hardware/data**
- Data base, or PID?
- Calibration/version?
- Integrity?

**Run software on data**
- What order?
- Runtime options?
- Confirmation bias?
- Human error?
- Environment update?
- In sync with coauthors?

**Paper**
- History recorded?
- Cited software?
- Report this info?
- Sync with analysis?

Green boxes with sharp corners: source/input components/files.
Blue boxes with rounded corners: built components.
Red boxes with dashed borders: questions that must be clarified for each phase.
Input data source and integrity is documented and checked

Stored information about each input file:

- **PID** (where available).
- Download **URL**.
- **MD5**-sum to check integrity.

All inputs are **downloaded** from the given PID/URL when necessary (during the analysis).

MD5-sums are **checked** to make sure the download was done properly or the file is the same (hasn’t changed on the server/source).

Example from the reproducible paper **arXiv:1909.11230**.
This paper needs three input files (two images, one catalog).
General outline of a project (after data collection)

Software
- Build
  - Config environment?
  - Config options?
  - Dep. versions?
  - Dependencies?

Hardware/data
- Data base, or PID?
- Calibration/version?
- Integrity?
- Repository?
- What version?

Run software on data
- What order?
- Confirmation bias?
- Runtime options?
- Human error?
- Environment update?

Paper
- History recorded?
- Cited software?
- Report this info?
- Sync with analysis?
- In sync with coauthors?

Green boxes with sharp corners: source/input components/files.
Blue boxes with rounded corners: built components.
Red boxes with dashed borders: questions that must be clarified for each phase.
Reproducible science: Maneage is managed through a Makefile

All steps (downloading and analysis) are managed by Makefiles (example from zenodo.1164774):

- Unlike a script which always starts from the top, a Makefile starts from the end and steps that don’t change will be left untouched (not remade).

- A single rule can manage any number of files.

- Make was designed for complex projects with thousands of files (all major Unix-like components), so it is highly evolved and efficient.

- Make is a very simple and small language, thus easy to learn with great and free documentation (for example GNU Make’s manual).
General outline of a project (after data collection)

- **Software**
  - Config environment?
  - Config options?
  - Dep. versions?
  - Dependencies?
  - Repository?
  - What version?
  - Data base, or PID?
  - Calibration/version?
  - Integrity?
  - Build
  - Run software on data
  - History recorded?
  - Cited software?
  - Report this info?
  - Sync with analysis?
  - Environment update?
  - In sync with coauthors?

**Green boxes** with sharp corners: source/input components/files.
**Blue boxes** with rounded corners: built components.
**Red boxes** with dashed borders: questions that must be clarified for each phase.
Values in final report/paper

All analysis results (numbers, plots, tables) written in paper’s PDF as \LaTeX macros. They are thus updated automatically on any change.

Shown here is a portion of the NoiseChisel paper and its \LaTeX source (arXiv:1505.01664).

\begin{equation}
\mathbf{S/N}_\text{r} = \frac{\text{NF-NS}_a}{\text{F-S}_a}\frac{\sqrt{\text{NF}+\text{N}}}{\sqrt{\text{F}+\text{sigma}_s^2}}.
\end{equation}

\noindent See Section \ref{SNegmodif} for the modifications required when the input image is not in units of counts or has already been Sky subtracted. The distribution of $\text{S/N}_r$ for the three examples in Figure \ref{dettf} can be seen in column 5 (top) of that figure. Image processing effects, mainly due to shifting, rotating, and re-sampling the images for co-adding, on the real data further increase the size and count, and hence, the $\text{S/N}_\text{r}$ of false detections in real, reduced/co-added images. A comparison of scales on the $\text{S/N}$ histograms between the mock ((a.5.1) and (b.5.1)) and real (c.5.1) examples in Figure \ref{dettf} shows the effect quantitatively. In the histograms of Figure \ref{dettf}, the bin with the largest number of false pseudo-detections respectively has an $\text{S/N}_\text{r}$ of $\text{one large dettf max}$, $\text{sensitivity cdettf max}$, and $\text{four dettf max}$.  

\begin{align*}
\frac{S}{N_T} &= \frac{NF - NS_d}{\sqrt{NF + N\sigma_s^2}} = \sqrt{\frac{N(F - S_0)}{F + \sigma_s^2}}.
\end{align*}

See Section 3.3 for the modifications required when the input image is not in units of counts or has already been Sky subtracted. The distribution of $S/N_T$ from the objects in $R_t$ for the three examples in Figure 7 can be seen in column 5 (top) of that figure. Image processing effects, mainly due to shifting, rotating, and re-sampling the images for co-adding, on the real data further increase the size and count, and hence, the $S/N$ of false detections in real, reduced/co-added images. A comparison of scales on the $S/N$ histograms between the mock ((a.5.1) and (b.5.1)) and real (c.5.1) examples in Figure 7 shows the effect quantitatively. In the histograms of Figure 7, the bin with the largest number of false pseudo-detections respectively has an $S/N$ of 1.89, 2.37, and 4.77.

The $S/N_T$ distribution of detections in $R_t$ provides a very ro-
Analysis step results/values concatenated into a single file.

All LATEX macros come from a single file.
Analysis results stored as \LaTeX\ macros

The analysis scripts write/update the \LaTeX\ macro values automatically.

```latex
\begin{verbatim}
# Numbers for dettf.tex:
sqnt=9999999
function detthist
{
  # Set the file name.
  if [ $2 == 4 ]; then obase=four;
elif [ $2 == sensitivity3 ]; then obase=sensitivityc;
  else obase=$2;
  fi
  if [ $2 == onelarge ]; then ind="_7"; else ind="_12"; fi
  name=$1$2$ind"_detsn"$txt

dettfnun=$ (awk '/points binned in/{print $4; exit(0)}' $name)
dettfqnt=$ (awk '/quantile has a value of/{
  printf("%.2f", $9); exit(0);}' $name)
dettfnmax=$ (awk 'BEGIN { max=-999999 } /\^\#/ { if($2>max){max=$2; mv=$1} } END { printf("%.2f", mv) }' $name)
addtexmacro $obase"dettfnun" $dettfnun
addtexmacro $obase"dettfnmax" $dettfnmax
addtexmacro $obase"dettfqnt" $dettfqnt

# Find the smallest S/N quantile:
sqnt=$(echo " " | awk '{if($dettfqnt<$sqnt) print "$dettfqnt"}')
}
for base in 4 onelarge sensitivity3
do detthist $texdir/dett/ $base; done
addtexmacro dettsmallestsqnt $sqnt
\end{verbatim}
```
Let’s look at the data lineage to replicate Figure 1C (green/tool) of Menke+2020 (DOI:10.1101/2020.01.15.908111), as done in arXiv:2006.03018 for a demo.

**ORIGINAL PLOT**
The Green plot shows the fraction of papers mentioning software tools from 1997 to 2019.

**OUR enhanced REPLICATION**
The green line is same as above but over their full historical range.
Red histogram is the number of papers studied in each year.
All analysis steps cascade down to paper.pdf (URL and checksum of input in INPUTS.conf).

Green boxes with sharp corners: source files (hand written).
Blue boxes with rounded corners: built files (automatically generated), built files are shown in the Makefile that contains their build instructions.
It is very easy to expand the project and add new analysis steps (this solution is scalable).

Basic project info (e.g., Git commit). Also defines project structure (for *.mk files).

**Green boxes** with sharp corners: *source* files (hand written).

**Blue boxes** with rounded corners: *built* files (automatically generated), built files are shown in the Makefile that contains their build instructions.
All questions have an answer now (in plain text: human & computer readable/archivable).

- **Software**
  - Repository?
  - What version?
  - Config environment?
  - Config options?
  - Dep. versions?
  - Dependencies?

- **Hardware/data**
  - Data base, or PID?
  - Calibration/version?
  - Integrity?

- **Build**
  - Runtime options?
  - Human error?
  - Confirmation bias?
  - What order?

- **Run software on data**
  - Environment update?
  - In sync with coauthors?

- **Paper**
  - History recorded?
  - Cited software?
  - Report this info?
  - Sync with analysis?

**Green boxes** with sharp corners: source/input components/files.
**Blue boxes** with rounded corners: built components.
**Red boxes** with dashed borders: questions that must be clarified for each phase.
All questions have an answer now (in plain text: so we can use Git to keep its history).

- **Green boxes** with sharp corners: source/input components/files.
- **Blue boxes** with rounded corners: built components.
- **Red boxes** with dashed borders: questions that must be clarified for each phase.
New projects branch from Maneage

- Template’s history is recorded in Git.
- New project: a branch from the template. Recall that every commit contains the following:
  - Instructions to download, verify and build software.
  - Instructions to download and verify input data.
  - Instructions to run software on data (do the analysis).
  - Narrative description of project’s purpose/context.

- Research progresses in the project branch.
- Template will evolve (improved infrastructure).
- Template can be imported/merged back into project.
- The template and project will evolve.
- During research this encourages creative tests (previous research states can easily be retrieved).
- Coauthors can work on same project in parallel (separate project branches).
- Upon publication, the Git checksum is enough to verify the integrity of the result.
Carving out the low surface brightness universe with NoiseChisel

Mohammad Akhlaghi1,2

1Instituto de Astrofísica de Canarias, C/ Vía Láctea, 38200 La Laguna, Tenerife, Spain.
email: mohammad@iac.es

2Facultad de Física, Universidad de La Laguna, Avda. Jardín Botánico s/n, 38200 La Laguna, Tenerife, Spain.

Abstract. NoiseChisel is a program to detect very low signal-to-noise ratio (S/N) features with minimal assumptions on their morphology. It was introduced in 2015 and released within a collection of data analysis programs and libraries known as GSTe Astronomy Taxonomy (Gauzewicz). Over the last ten stable releases of Gauzewicz, NoiseChisel has significantly improved; detecting even fainter signal, enabling better user control over its inner workings, and many bug fixes. The most important change may be that NoiseChisel’s selection features have been moved into a new program called Segment. Another major change is the final growth strategy of its true detections, for example NoiseChisel is able to resolve the outer wings of M31 down to 0.18 or 22.57 mag/arcsec2 on a single-exposure frames of r-band. Segment is also able to detect the localized HII regions as ‘Violet’ structure. Finally, to demonstrate a controlled analysis, the concept of a ‘template’ has been released: this paper itself is exactly a template of the object being processed.

Keywords: galaxies, halos, galaxy populations, galaxy structure; methods: data analysis methods: reproducible; techniques: image processing; techniques: photometric

1. Introduction

Signal from the low surface brightness universe is buried deep in the data noise and thus requires accurate detection methods. In Akhlaghi and Ichikawa (2015) (hereinafter A15) a new method was introduced to detect such very low signal-to-noise ratio (S/N) signal from the images in a non-parametric manner. It allows accurate detection of the diffuse outer features of galaxies (that often have a different morphology from the centers). The software implementation of this method (NoiseChisel) is released as part of a larger collection of data analysis software known as GSTe Astronomy Utilities (Gauzewicz).

It was the first professional astronomical software to be independently released by an independent panel (GNU Evaluation committee) and fully conforms with the GNU Coding Standards.

Since its release, NoiseChisel has been used in many studies. For example, Bacon et al. (2017) used it to identify objects that were missed by Riddle et al. (2011) (hereinafter R11). A15 used a combination of six Skrutter and Arnaboldi’s (2006) runs with different configurations to avoid deblending problems, but still missed many sources with significant signal, such as Figure 1, Bierold et al. (2018), Miller et al. (2019), and Trujillo et al. (2019) used it to find Lyman-$\alpha$ emitters in spectra. For future studies, Laine et al. also characterized and used the extended PISF of the H-I gas towards Galactic Center (GCT) telescope to model and remove the scattered light in the observation of the G32.59 galaxy. (In some cases, the signal-to-noise ratio is very low) from the images of the Canada-France-Hawaii Telescope (CFHT) to do a census of these galaxies and star-forming regions in the large-scale structures and finally, the extended PISF of the H-I gas towards Galactic Center (GCT) telescope. The main result of this study is that the signal-to-noise ratio is very low, which makes the analysis of these galaxies very difficult. (In some cases, the signal-to-noise ratio is very low)
Publication of the project

A reproducible project using Maneage will have the following (plain text) components:

- Makefiles.
- \LaTeX source files.
- Configuration files for software used in analysis.
- Scripts/programming files (e.g., Python, Shell, AWK, C).

The volume of the project’s source will thus be negligible compared to a single figure in a paper (usually $\sim 100$ kilo-bytes).

The project’s pipeline (customized Maneage) can be published in

- \textbf{arXiv}: uploaded with the \LaTeX source to always stay with the paper (for example arXiv:1505.01664 or arXiv:2006.03018).
- \textbf{Zenodo}: Along with all the input datasets (many Gigabytes) and software (for example zenodo.3872248) and given a unique DOI.
General outline of using Maneage (for example arXiv:2006.03018)

$ git clone https://gitlab.com/makhlaghi/maneage-paper  # Import the project.

$ ./project configure  # You will specify the build directory on your system,
                       # and it will build all software (about 1.5 hours).

$ ./project make  # Does all the analysis and makes final PDF.
Future prospects...

Adoption of reproducibility by many researchers will enable the following:

- A repository for education/training (PhD students, or researchers in other fields).
- Easy **verification/understanding** of other research projects (when necessary).
- Trivially **test** different steps of others’ work (different configurations, software and etc).
- Science can progress **incrementally** (shorter papers actually building on each other!).
- Extract **meta-data** after the publication of a dataset (for future ontologies or vocabularies).
- Applying **machine learning** on reproducible research projects will allow us to solve some Big Data Challenges:
  - Extract the relevant parameters automatically.
  - **Translate the science to enormous samples.**
  - **Believe the results when no one will have time to reproduce.**
  - **Have confidence in results derived using machine learning or AI.**
Maneage and its principles are described in arXiv:2006.03018. It is a customizable template that will do the following steps/instructions (all in simple plain text files).

- **Automatically downloads** the necessary *software* and *data*.
- **Builds** the software in a *closed environment*.
- **Runs the software on data** to **generate** the final *research results*.
- **Modification** of part of the analysis will only result in re-doing that part, not the whole project.
- **Using LaTeX macros**, paper’s figures, tables and numbers will be **Automatically updated** after a change in analysis. Allowing the scientist to focus on the scientific interpretation.
- The whole project is under **version control** (Git) to allow easy reversion to a previous state. This encourages tests/experimentation in the analysis.
- The **Git commit hash** of the project source, is **printed** in the published paper and **saved on output data products**. Ensuring the integrity/reproducibility of the result.


For a technical description of Maneage’s implementation, as well as a checklist to customize it, and tips on good practices, please see this page:

[https://gitlab.com/maneage/project/-/blob/maneage/README-hacking.md](https://gitlab.com/maneage/project/-/blob/maneage/README-hacking.md)