

BIG Data, BIG responsibility

Maneage: *Managing data lineage for long-term and archivable reproducibility*

(Published in CiSE 23 (3), pp 82-91: DOI:10.1109/MCSE.2021.3072860, arXiv:2006.03018)

Mohammad Akhlaghi

Centro de Estudios de Física del Cosmos de Aragón (CEFCA), Teruel, Spain

Royal Observatory Coffee talk; Edinburgh
23rd of May 2023

Most recent slides available in link below (this PDF is built from Git commit c747d78-dirty):

<https://maneage.org/pdf/slides-intro.pdf>



Financiado por la Unión Europea-NextGenerationEU



Let's start with this nice image of the Whirlpool galaxy (M51): <https://i.redd.it/jfqgpqg0hfk11.jpg>

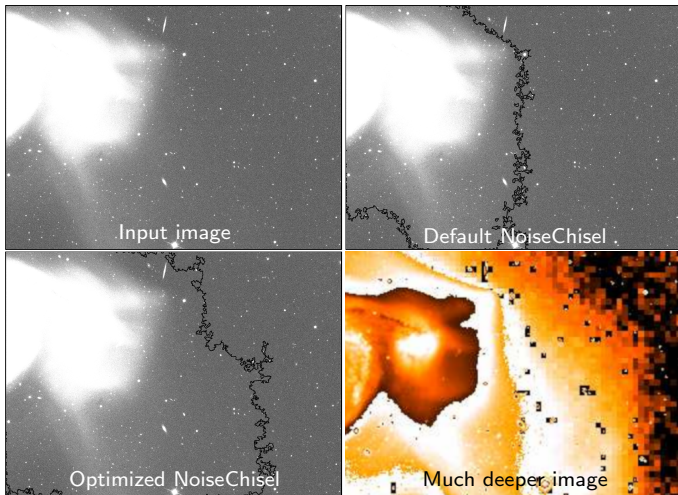


Now, let's assume you want to study M51's outer structure, but you'll have to detect it first.

Example: Using a **single exposure** SDSS image with NoiseChisel (a program that is part of 'GNU Astronomy Utilities').

- ▶ When optimized, outskirts detected down to $S/N = 1/4$, or **28.3** mag/arcsec². By default, it only reaches $S/N > 1/2$.
- ▶ Akhlaghi 2019 ([arXiv:1909.11230](https://arxiv.org/abs/1909.11230)) describes optimized result:
 - ▶ **Run-time** options/configuration.
 - ▶ Steps **before/after** NoiseChisel.
- ▶ Deep/orange image from Watkins+2015 ([arXiv:1501.04599](https://arxiv.org/abs/1501.04599)) shown for reference.
- ▶ Therefore:
 - ▶ Default settings not enough.
 - ▶ Final number not just from NoiseChisel (more software involved).

Simply reporting in your paper that "**we used NoiseChisel**" is **not enough** to reproduce, understand, or verify your result.



Reproducibility crisis in the sciences/astronomy

Snakes on a Spaceship – An Overview of Python in Heliophysics

“...inadequate analysis descriptions and loss of scientific data have made scientific studies difficult or impossible to replicate”. From Burrell+2018, ([arXiv:1901.00143](https://arxiv.org/abs/1901.00143)).

Reproducibility crisis in the sciences/astronomy

Snakes on a Spaceship – An Overview of Python in Heliophysics

“...**inadequate analysis descriptions** and loss of scientific data have made scientific studies **difficult** or **impossible** to replicate”. From Burrell+2018, ([arXiv:1901.00143](#)).

Perspectives on Reproducibility and Sustainability of Open-Source Scientific Software

“It is our interest that NASA adopt an open-code policy because without it, reproducibility in computational science is **needlessly hampered**”. From Oishi+2018, ([arXiv:1801.08200](#)).

Reproducibility crisis in the sciences/astronomy

Snakes on a Spaceship – An Overview of Python in Heliophysics

“...**inadequate analysis descriptions** and loss of scientific data have made scientific studies **difficult** or **impossible** to replicate”. From Burrell+2018, ([arXiv:1901.00143](#)).

Perspectives on Reproducibility and Sustainability of Open-Source Scientific Software

“It is our interest that NASA adopt an open-code policy because without it, reproducibility in computational science is **needlessly hampered**”. From Oishi+2018, ([arXiv:1801.08200](#)).

Schroedinger's code: source code availability and link persistence in astrophysics

“We were **unable to find source code** online ... for 40.4% of the codes used in the research we looked at”. From Allen+2018, ([arXiv:1801.02094](#)).



Original image from <https://www.redbubble.com>

This problem isn't just limited to astronomy

Repeatability of published microarray gene expression analyses

Ioannidis+2009 evaluated the replication of data analyses in 18 articles ... in Nature Genetics and reproduced only 2 in principle." . DOI:10.1038/ng.295.

This problem isn't just limited to astronomy

Repeatability of published microarray gene expression analyses

Ioannidis+2009 evaluated the replication of data analyses in **18 articles** ... in Nature Genetics and reproduced **only 2** in principle." . DOI:10.1038/ng.295.

Is Economics Research Replicable? 60 papers from Thirteen Journals Say "Usually Not"

Chang&Li2015 were able to **replicate less than half** of 67 papers in well-regarded journals. Even *with help* from the authors. They "assert that **economics research is usually not replicable**". DOI:10.17016/FEDS.2015.083

This problem isn't just limited to astronomy

Repeatability of published microarray gene expression analyses

Ioannidis+2009 evaluated the replication of data analyses in **18 articles** ... in *Nature Genetics* and reproduced **only 2** in principle". DOI:10.1038/ng.295.

Is Economics Research Replicable? 60 papers from Thirteen Journals Say "Usually Not"

Chang&Li2015 were able to **replicate less than half** of 67 papers in well-regarded journals. Even *with help* from the authors. They "assert that **economics research is usually not replicable**". DOI:10.17016/FEDS.2015.083

An empirical analysis of journal policy effectiveness for computational reproducibility

Stodden+2018 studied a random sample of **204** scientific papers in *Science* and were able to obtain **artifacts from 44%** and **reproduce the findings for 26%**. DOI:10.1073/pnas.1708290115

“Reproducibility crisis” in the sciences? (Baker 2016, Nature 533, 452, DOI:10.1038/533452a)

1576 researchers participated in a survey by Nature, 90% believed in a crisis!

Status	% agreed
Yes, a significant crisis	52
Yes, a slight crisis	38
Don't know	7
No, there is no crisis	3

Full PDF available at <https://www.nature.com/articles/533452a.pdf>

EDITORS: Lorena A. Barba, lbarba@gsa.es
Lorena Barba, lorena.garcia@gsa.es

SPECIAL TRACK: REPRODUCIBLE RESEARCH

Toward Long-Term and Archivable Reproducibility

Mohammad Akhlaghi , Instituto de Astrofísica de Canarias, La Laguna, Tenerife, 38205, Spain
Raul Infante-Sainz , Universidad de La Laguna, La Laguna, Tenerife, 38205, Spain
Boudewijn F. Roukema , Nicolaus Copernicus University, Toruń 87-100, Poland
Mohammadreza Khellat , Ideal-Information, PC 133/Al Khawair, Muscat, Oman
David Valls-Gabaud, Paris Observatory, Paris 75014, France
Roberto Bana-Galla , Universidad Internacional de La Rioja, Logroño 26006, Spain

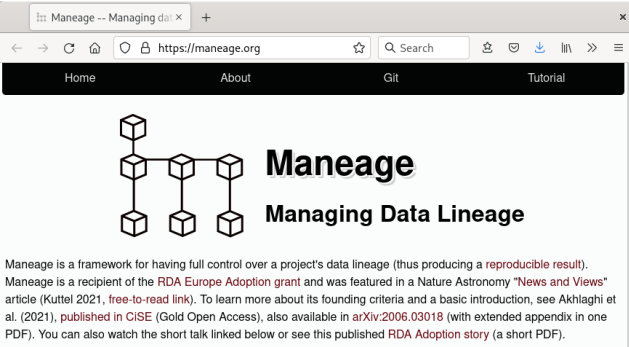
Analysis pipelines commonly use high-level technologies that are popular when created, but are unlikely to be readable, executable, or sustainable in the long term. A set of criteria is introduced to address this problem: completeness (no execution requirement beyond a minimal Unix-like operating system, no administrator privileges, no network connection, and storage primarily in plain text); modular design; minimal complexity; scalability; verifiable inputs and outputs; version control; linking analysis with narrative; and free and open-source software. As a proof of concept, we introduce "Maneage" (managing data lineage), enabling cheap archiving, provenance extraction, and peer verification that has been tested in several research publications. We show that longevity is a realistic requirement that does not sacrifice immediate or short-term reproducibility. The caveats (with proposed solutions) are then discussed and we conclude with the benefits for the various stakeholders. This article is itself a Maneage'd project (commit 313db0b). Appendices—Two comprehensive appendices that review the longevity of existing solutions are available as supplementary "Web extras," which are available in the IEEE Computer Society Digital Library at <http://dx.doi.org/10.1109/MCSE.2021.3072860>. Reproducibility—All products available in zenodo: 4913277, the Git history of this paper's source is at github.com/maneage/maneage, which is also archived in Software Heritage: swh1.dli:33feab87068c1612da071f161b977b9a0d3f. Clicking on the SWHIDs in the digital format will provide more "context" for same content.

Reproducible research has been discussed in the sciences for at least 30 years.^{1,2} Many reproducible workflow solutions (hereafter, "solutions") have been proposed, which mostly rely on the common technology of the day, starting

with Make and Matlab libraries in the 1990s, Java in the 2000s, and mostly shifting to Python during the past decade.

However, these technologies develop fast, e.g., code written in Python 2 (which is no longer officially maintained) often cannot run with Python 3. The cost of staying up to date within this rapidly evolving landscape is high. Scientific projects, in particular, suffer the most: Scientists have to focus on their own research domain, but to some degree, they need to understand the technology of their tools because it determines their results

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see <https://creativecommons.org/licenses/by/4.0/>.
Digital Object Identifier 10.1109/MCSE.2021.3072860
Date of publication 13 April 2021; date of current version 15 June 2021.

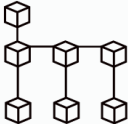


The screenshot shows a web browser window with the address bar displaying "https://maneage.org". The website has a dark blue header with navigation links: Home, About, Git, and Tutorial. Below the header is a large graphic featuring a network diagram of interconnected cubes and the text "Maneage Managing Data Lineage". The main content area contains a paragraph about Maneage, its funding, and its availability in various formats.

Maneage -- Managing data x

← → ↻ 🏠 🔒 https://maneage.org ☆ 🔍 Search 📄 📄 📄 📄 📄 📄

Home About Git Tutorial

 **Maneage**
Managing Data Lineage

Maneage is a framework for having full control over a project's data lineage (thus producing a **reproducible result**). Maneage is a recipient of the **RDA Europe Adoption grant** and was featured in a Nature Astronomy "**News and Views**" article (Kuttel 2021, **free-to-read link**). To learn more about its founding criteria and a basic introduction, see Akhlaghi et al. (2021), **published in CiSE** (Gold Open Access), also available in **arXiv:2006.03018** (with extended appendix in one PDF). You can also watch the short talk linked below or see this published **RDA Adoption story** (a short PDF).

<https://maneage.org>

Recognition 1: RDA adoption grant (2019) to IAC for Maneage



For Maneage, the **IAC** is selected as a **Top European organization** funded to adopt RDA Recommendations and Outputs.

- ▶ Research Data Alliance was launched by the **European Commission**, NSF, National Institute of Standards and Technology, and the Australian Government's Department of Innovation.
- ▶ RDA Outputs are the technical and social infrastructure solutions developed by RDA Working Groups or Interest Groups that enable data sharing, exchange, and interoperability.

news & views



REPRODUCIBILITY

No expiration date

The short lifespan of software puts a time limit on the reproducibility of computational research. To extend software longevity, guidelines and tools to preserve scientific workflows and analysis are helpful, but the challenge is to get researchers to use them.

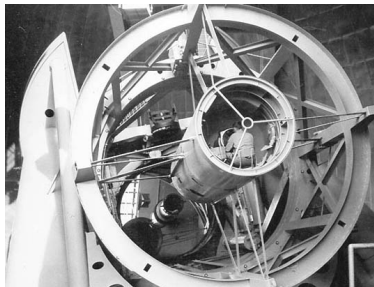
Michelle M. Kuttel

Free-to-read link: <https://rdcu.be/cmYVx>

DOI: [10.1038/s41550-021-01402-3](https://doi.org/10.1038/s41550-021-01402-3)

Replicability (hardware/statistical)

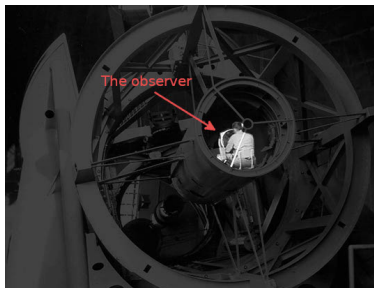
- ▶ Involves data **collection**.
- ▶ Inherently includes **measurements errors** (can never be exactly reproduced).
- ▶ Example: Raw telescope image/spectra.
- ▶ **NOT DISCUSSED HERE.**



<http://slittlefair.staff.shef.ac.uk>

Replicability (hardware/statistical)

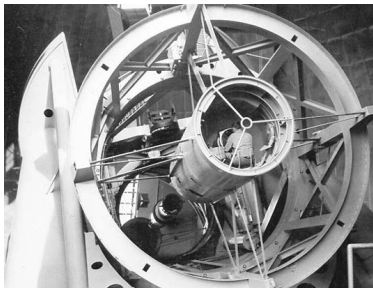
- ▶ Involves data **collection**.
- ▶ Inherently includes **measurements errors** (can never be exactly reproduced).
- ▶ Example: Raw telescope image/spectra.
- ▶ **NOT DISCUSSED HERE.**



<http://slittlefair.staff.shef.ac.uk>

Replicability (hardware/statistical)

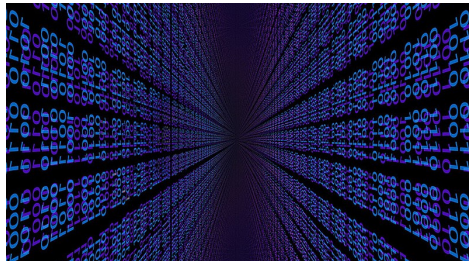
- ▶ Involves data **collection**.
- ▶ Inherently includes **measurements errors** (can never be exactly reproduced).
- ▶ Example: Raw telescope image/spectra.
- ▶ **NOT DISCUSSED HERE.**



<http://slittlefair.staff.shef.ac.uk>

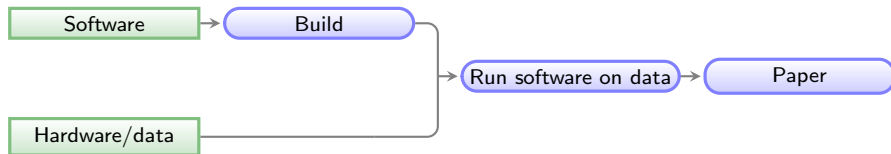
Reproducibility (Software/Deterministic)

- ▶ Involves data **analysis**, or simulations.
- ▶ Starts **after** data is collected/digitized.
- ▶ Example: $2 + 2 = 4$ (i.e., sum of datasets).
- ▶ **DISCUSSED HERE.**



Wikimedia Commons

General outline of a project (after data collection)

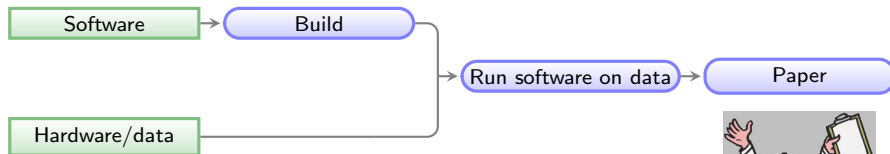


Green boxes with sharp corners: *source*/input components/files.

Blue boxes with rounded corners: *built* components.

Red boxes with dashed borders: questions that must be clarified for each phase.

General outline of a project (after data collection)



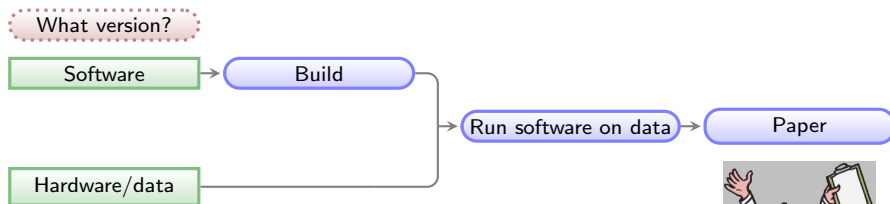
Green boxes with sharp corners: *source*/input components/files.

Blue boxes with rounded corners: *built* components.

Red boxes with dashed borders: questions that must be clarified for each phase.

<https://heywhatwhatdidyousay.wordpress.com>

General outline of a project (after data collection)



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>

Different package managers have different versions of software (repology.org, 2021/12/02)

Astropy

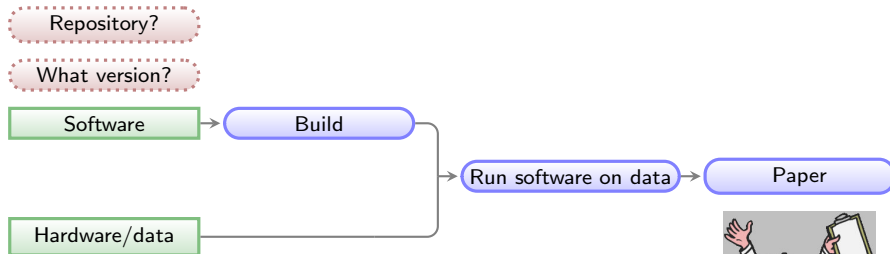
Packaging status	
Debian 10	3.1.2
Debian 11	4.2
Debian 12	4.3.1
Debian Unstable	5.0
Debian Experimental	5.0-rc2
Deepin	3.1.2
Devuan 3.0	3.1.2
Devuan 4.0	4.2
Devuan Unstable	5.0
Kali Linux Rolling	4.3.1
Pardus 19	3.1.2
Pardus 21	4.2
Parrot	4.2
PureOS Amber	3.1.2
PureOS landing	4.2
Raspbian Oldstable	3.1.2
Raspbian Stable	4.2
Raspbian Testing	4.3.1
Trisquel 9.0	3.0
Trisquel 10.0	4.0
Ubuntu 18.04	3.0
Ubuntu 20.04	4.0
Ubuntu 20.10	4.0.1+post1
Ubuntu 21.04	4.2
Ubuntu 21.10	4.2
Ubuntu 22.04	4.2
Ubuntu 22.04 Proposed	4.3.1

GNU Astronomy Utilities (Gnuastro)

Packaging status	
Debian 9	0.2.33
Debian 10	0.8
Debian 11	0.14
Debian 12	0.16.1
Debian Unstable	0.16.1
Deepin	0.8
Devuan 2.0	0.2.33
Devuan 3.0	0.8
Devuan 4.0	0.14
Devuan Unstable	0.16.1
DPorts	0.15
FreeBSD Ports	0.16
Funtoo 1.4	0.3
Gentoo	0.3
GNU Guix	0.16
Kali Linux Rolling	0.16.1
LiGurOS stable	0.3
LiGurOS develop	0.3
OpenBSD Ports	0.15
openSUSE Leap 15.1	0.8
openSUSE Leap 15.2	0.8
openSUSE Leap 15.3	0.8
openSUSE Tumbleweed	0.16
openSUSE Science Tumbleweed	0.16
Pardus 17	0.2.33
Pardus 19	0.8
Pardus 21	0.14
Parrot	0.14
PLD Linux	0.15

PureOS Amber	0.8
PureOS landing	0.14
Raspbian Oldstable	0.8
Raspbian Stable	0.14
Raspbian Testing	0.16.1
RPM Sphere	0.16.1
Trisquel 9.0	0.5
Trisquel 10.0	0.11
Ubuntu 18.04	0.5
Ubuntu 20.04	0.11
Ubuntu 20.10	0.12
Ubuntu 21.04	0.14
Ubuntu 21.10	0.14
Ubuntu 22.04	0.14
Ubuntu 22.04 Proposed	0.16.1

General outline of a project (after data collection)



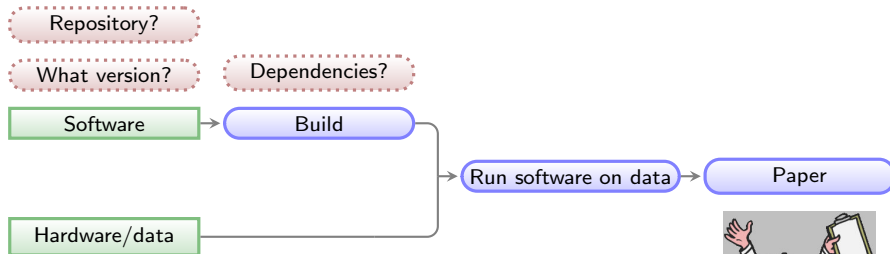
Green boxes with sharp corners: *source*/input components/files.

Blue boxes with rounded corners: *built* components.

Red boxes with dashed borders: questions that must be clarified for each phase.

<https://heywhatwhatdidyousay.wordpress.com>

General outline of a project (after data collection)

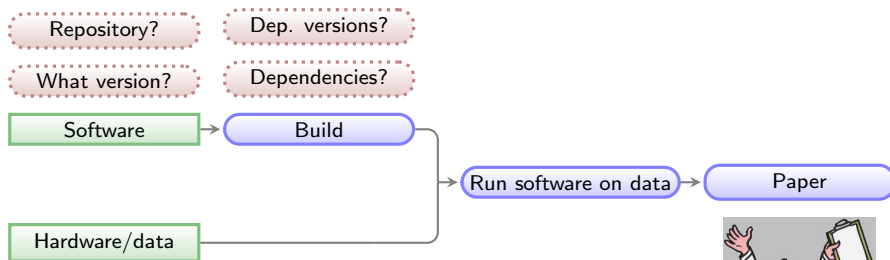


Green boxes with sharp corners: *source*/input components/files.

Blue boxes with rounded corners: *built* components.

Red boxes with dashed borders: questions that must be clarified for each phase.

General outline of a project (after data collection)

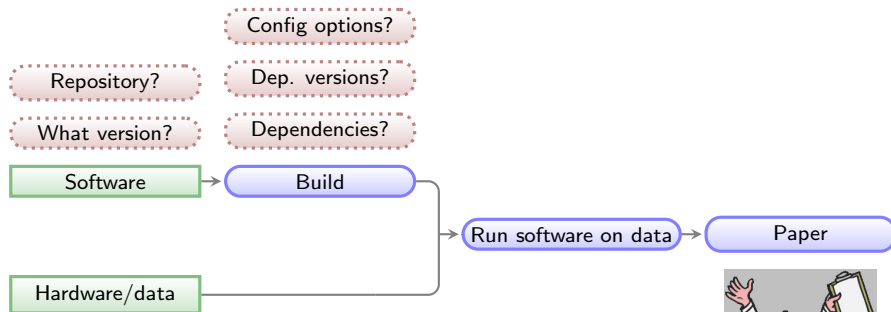


Green boxes with sharp corners: *source*/input components/files.

Blue boxes with rounded corners: *built* components.

Red boxes with dashed borders: questions that must be clarified for each phase.

General outline of a project (after data collection)

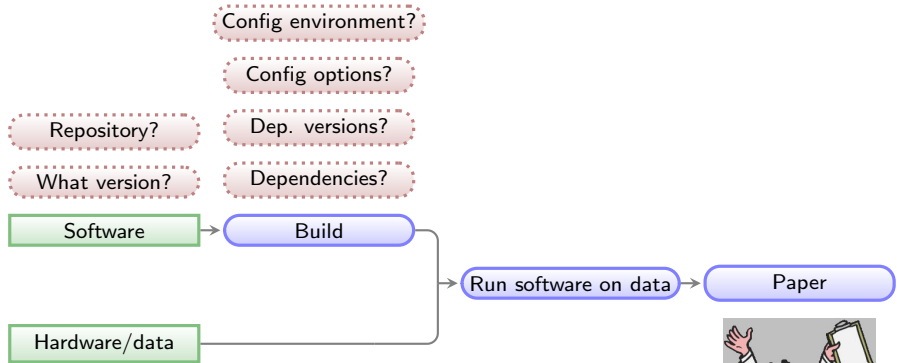


Green boxes with sharp corners: *source*/input components/files.

Blue boxes with rounded corners: *built* components.

Red boxes with dashed borders: questions that must be clarified for each phase.

General outline of a project (after data collection)



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.

Example: Matplotlib (a Python visualization library) build dependencies

Matplotlib library

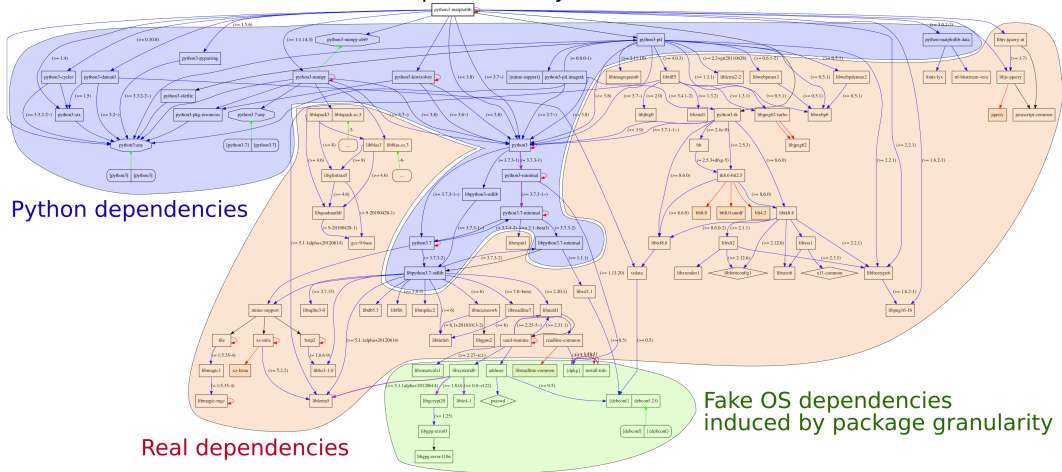


Fig. 1. Transitive dependencies of the software environment required by a simple "import matplotlib" command in the Python 3 interpreter.

Impact of “Dependency hell” on native building in various hardware (CPU architectures), retrieved from Debian on 2021/12/02



Debian Package Auto-Building

Build status for astropy (sid)

Tracker - Changelog - Bugs - packages.d.o - Source

Package(s): Suite:

☐ Compact mode ☐ Co-maintainers

Architecture	Version	Status	For	Buildid	State	Section	Logs	Actions
all	5.0-1	Installed	9d 9h 36m	x86-conova-01		misc	old all (1)	giveback
amd64	5.0-1	Installed	9d 9h 37m	x86-csail-01		misc	old all (1)	giveback
arm64	5.0-1	Installed	9d 9h 8m	arm-ubc-02		misc	old all (1)	giveback
armel	5.0-1	Installed	9d 6h 52m	antheil		misc	old all (1)	giveback
armhf	5.0-1	Installed	9d 8h 8m	hoiby		misc	old all (1)	giveback
i386	5.0-1	Installed	9d 9h 57m	x86-grnet-01		misc	old all (1)	giveback
mips64el	5.0-1	Build-Attempted	8d 18h 46m	mipsel-osuosl-04	out-of-date	misc	old all (3)	giveback
mipsel	5.0-1	Installed	9d 9h 37m	mipsel-manda-05		misc	old all (1)	giveback
ppc64el	5.0-1	Installed	9d 9h 37m	ppc64el-unicamp-01		misc	old all (1)	giveback
s390x	5.0-1	Installed	9d 9h 57m	zandonai		misc	old all (1)	giveback
alpha	5.0-1	BD-Uninstallable	9d 10h 22m		out-of-date	misc	old no log	giveback
hppa	5.0-1	Build-Attempted	2d 17h 20m	c8000	out-of-date	misc	old all (3)	giveback
hurd-i386	5.0-1	BD-Uninstallable	9d 10h 22m		uncompiled	misc	old no log	giveback
ia64	5.0-1	BD-Uninstallable	9d 10h 22m		uncompiled	misc	old no log	giveback
kfreebsd-amd64	5.0-1	BD-Uninstallable	9d 10h 22m		uncompiled	misc	old no log	giveback
kfreebsd-i386	5.0-1	BD-Uninstallable	9d 10h 22m		uncompiled	misc	old no log	giveback
m68k	5.0-1	BD-Uninstallable	9d 10h 22m		out-of-date	misc	old no log	giveback
powerpc	5.0-1	BD-Uninstallable	9d 10h 22m		uncompiled	misc	old no log	giveback
ppc64	5.0-1	Installed	9d 9h 31m	kapitsa		misc	old all (1)	giveback
risCV64	5.0-1	Installed	9d 6h 11m	rv-osuosl-02		misc	old all (1)	giveback
sh4	5.0-1	BD-Uninstallable	9d 10h 21m		out-of-date	misc	old no log	giveback
sparc64	5.0-1	BD-Uninstallable	9d 10h 21m		out-of-date	misc	old no log	giveback
x32	5.0-1	BD-Uninstallable	9d 10h 21m		out-of-date	misc	old no log	giveback

Astropy depends on Matplotlib



Debian Package Auto-Building

Build status for gnuastro (sid)

Tracker - Changelog - Bugs - packages.d.o - Source

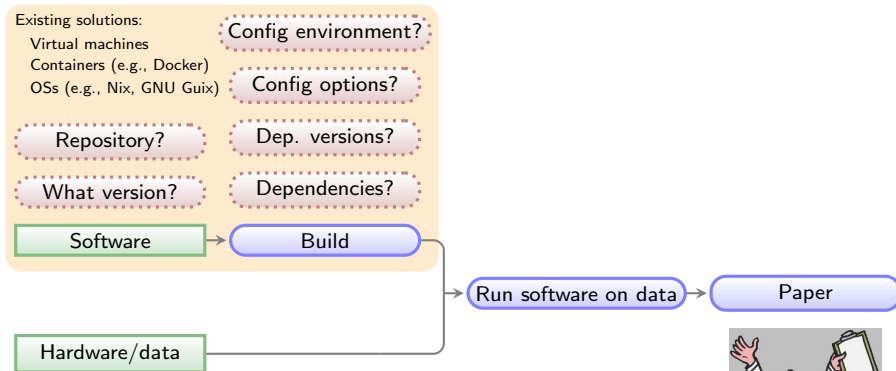
Package(s): Suite:

☐ Compact mode ☐ Co-maintainers

Architecture	Version	Status	For	Buildid	State	Section	Logs	Actions
all is not present in the architecture list set by the maintainer								
amd64	0.16.1-1	Installed	14d 6h 8m	x86-csail-01		misc	old all (1)	giveback
arm64	0.16.1-1	Installed	14d 5h 56m	arm-ubc-03		misc	old all (1)	giveback
armel	0.16.1-1	Installed	14d 5h 26m	henze		misc	old all (1)	giveback
armhf	0.16.1-1	Installed	14d 5h 56m	arm-conova-02		misc	old all (1)	giveback
i386	0.16.1-1	Installed	14d 5h 56m	x86-ubc-01		misc	old all (1)	giveback
mips64el	0.16.1-1	Installed	14d 5h 26m	mipsel-aql-03		misc	old all (1)	giveback
mipsel	0.16.1-1	Installed	11d 15h 26m	mipsel-osuosl-04		misc	old all (1)	giveback
ppc64el	0.16.1-1	Installed	14d 6h 8m	ppc64el-unicamp-01		misc	old all (1)	giveback
s390x	0.16.1-1	Installed	14d 6h 7m	zani		misc	old all (1)	giveback
alpha	0.16.1-1	Installed	7d 6h 11m	imago		misc	old all (2)	giveback
hppa	0.16.1-1	Installed	14d 5h 31m	c8000b		misc	old all (1)	giveback
hurd-i386	0.16.1-1	Installed	12d 19h 21m	ironforge		misc	old all (1)	giveback
ia64	0.16.1-1	Installed	14d 5h 41m	ilfshitz2		misc	old all (1)	giveback
kfreebsd-amd64	0.16.1-1	Installed	13d 22h 31m	kamp		misc	old all (1)	giveback
kfreebsd-i386	0.16.1-1	Installed	11d 11h 31m	kamp		misc	old all (1)	giveback
m68k	0.16.1-1	Installed	14d 4h 21m	vs92		misc	old all (1)	giveback
powerpc	0.16.1-1	Installed	14d 5h 31m	blauw		misc	old all (1)	giveback
ppc64	0.16.1-1	Installed	14d 6h	blauw2		misc	old all (1)	giveback
risCV64	0.16.1-1	Installed	14d 5h 30m	rv-osuosl-01		misc	old all (1)	giveback
sh4	0.16.1-1	Installed	14d 5h	sh4-do-02		misc	old all (1)	giveback
sparc64	0.16.1-1	Installed	14d 5h 10m	nv5120b		misc	old all (1)	giveback
x32	0.16.1-1	Installed	14d 6h	x32-do-02		misc	old all (1)	giveback

GNU Astronomy Utilities doesn't.

General outline of a project (after data collection)



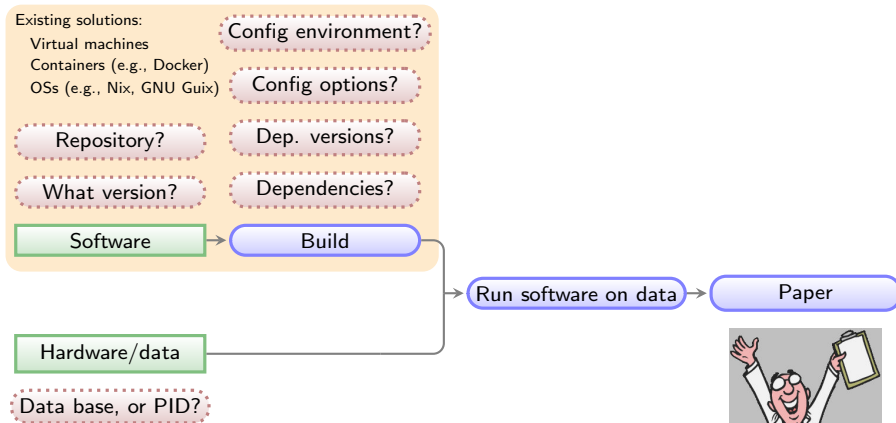
Green boxes with sharp corners: *source*/input components/files.

Blue boxes with rounded corners: *built* components.

Red boxes with dashed borders: questions that must be clarified for each phase.

<https://heywhatwhatdidyousay.wordpress.com>

General outline of a project (after data collection)

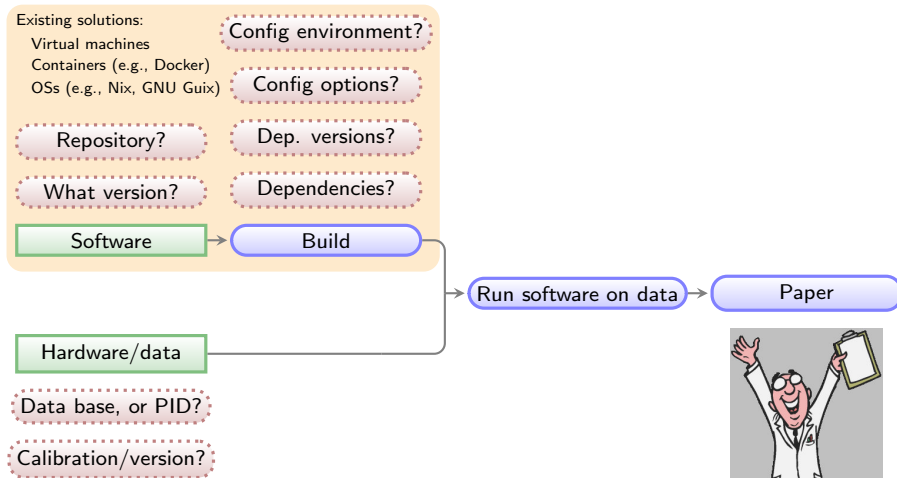


Green boxes with sharp corners: *source*/input components/files.

Blue boxes with rounded corners: *built* components.

Red boxes with dashed borders: questions that must be clarified for each phase.

General outline of a project (after data collection)

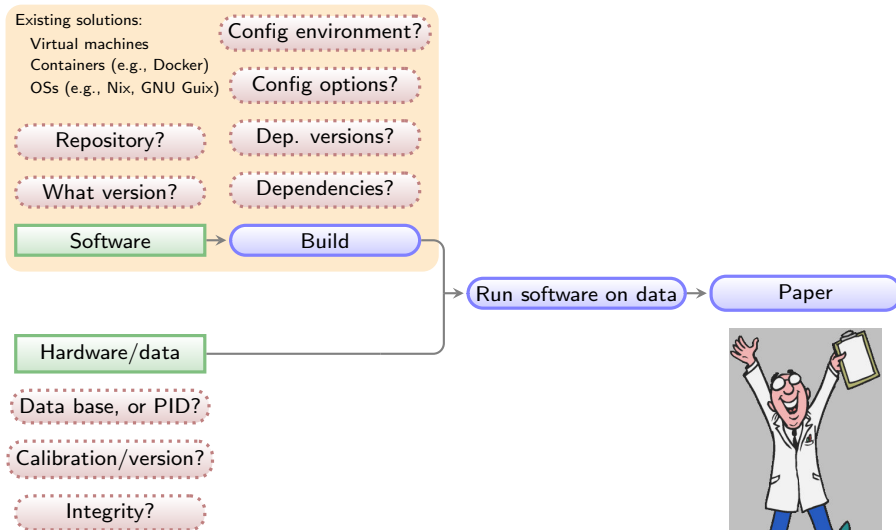


Green boxes with sharp corners: *source*/input components/files.

Blue boxes with rounded corners: *built* components.

Red boxes with dashed borders: questions that must be clarified for each phase.

General outline of a project (after data collection)



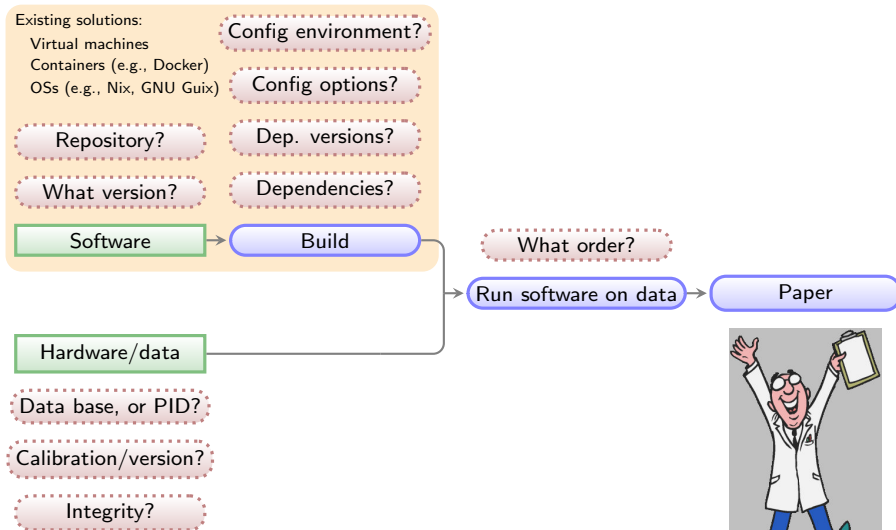
Green boxes with sharp corners: *source*/input components/files.

Blue boxes with rounded corners: *built* components.

Red boxes with dashed borders: questions that must be clarified for each phase.



General outline of a project (after data collection)



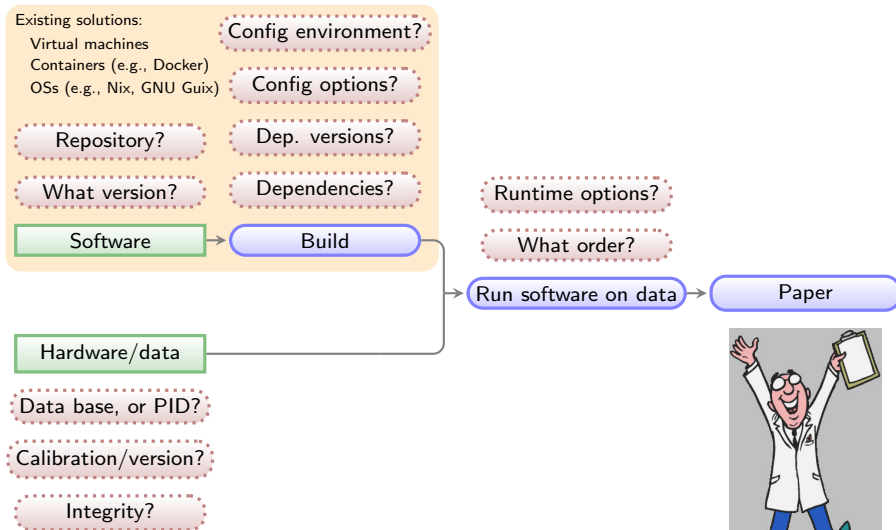
Green boxes with sharp corners: *source*/input components/files.

Blue boxes with rounded corners: *built* components.

Red boxes with dashed borders: questions that must be clarified for each phase.



General outline of a project (after data collection)



Green boxes with sharp corners: *source*/input components/files.

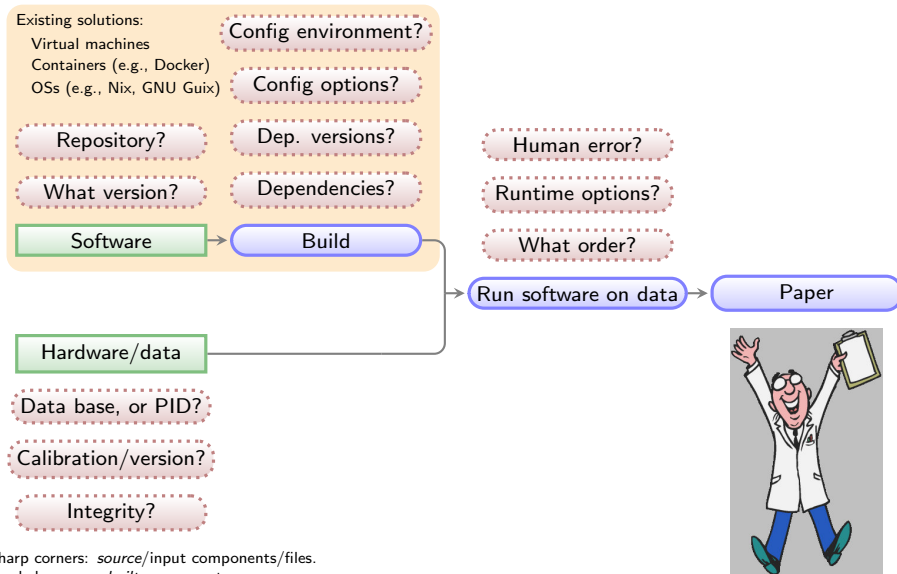
Blue boxes with rounded corners: *built* components.

Red boxes with dashed borders: questions that must be clarified for each phase.

<https://heywhatwhatdidyousay.wordpress.com>



General outline of a project (after data collection)



Green boxes with sharp corners: *source*/input components/files.

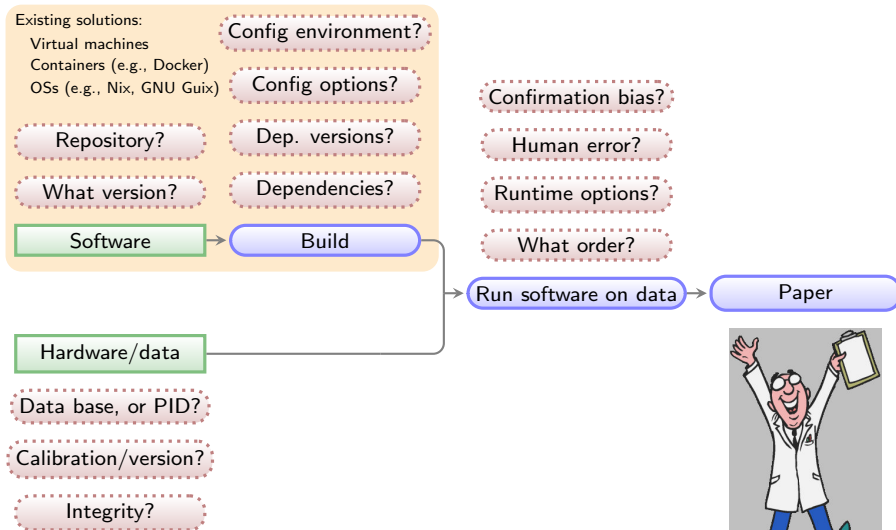
Blue boxes with rounded corners: *built* components.

Red boxes with dashed borders: questions that must be clarified for each phase.

<https://heywhatwhatdidyousay.wordpress.com>



General outline of a project (after data collection)



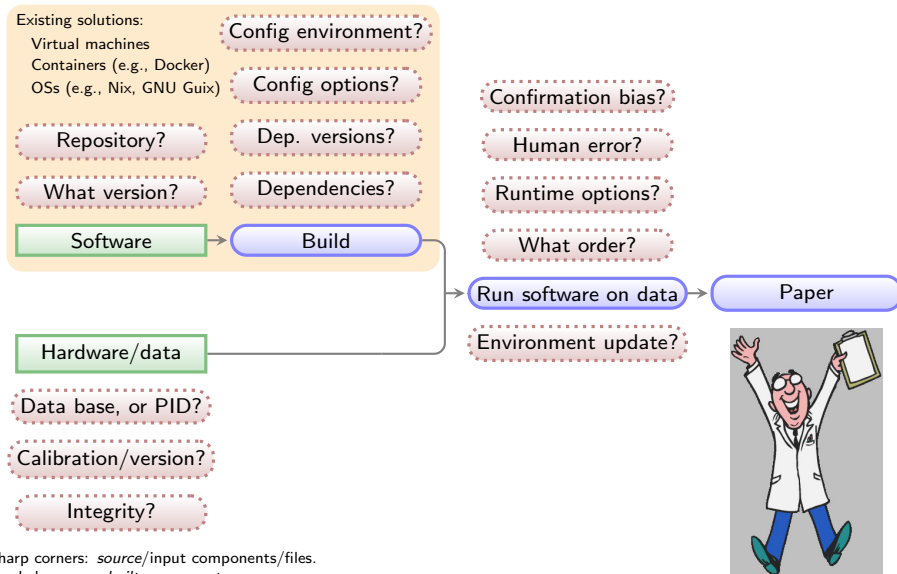
Green boxes with sharp corners: *source*/input components/files.

Blue boxes with rounded corners: *built* components.

Red boxes with dashed borders: questions that must be clarified for each phase.

<https://heywhatwhatdidyousay.wordpress.com>

General outline of a project (after data collection)



Green boxes with sharp corners: *source*/input components/files.

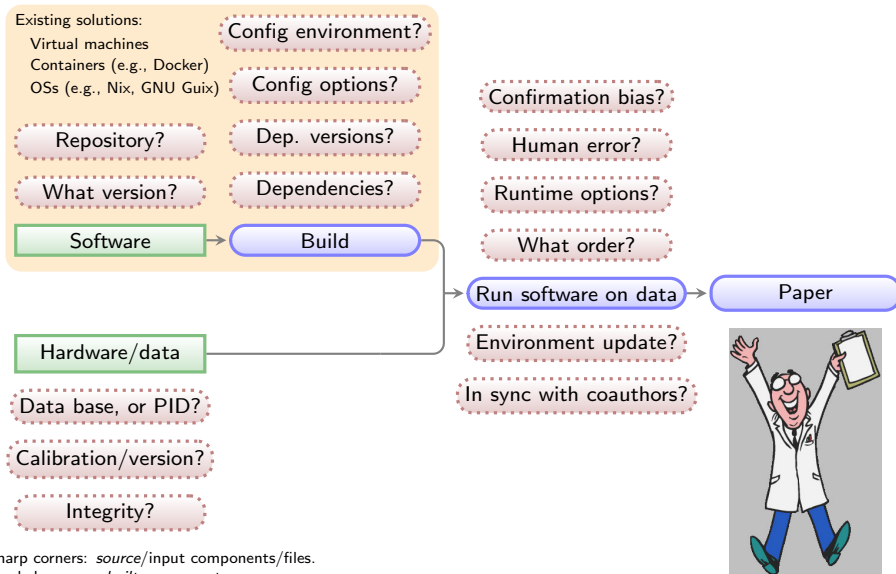
Blue boxes with rounded corners: *built* components.

Red boxes with dashed borders: questions that must be clarified for each phase.

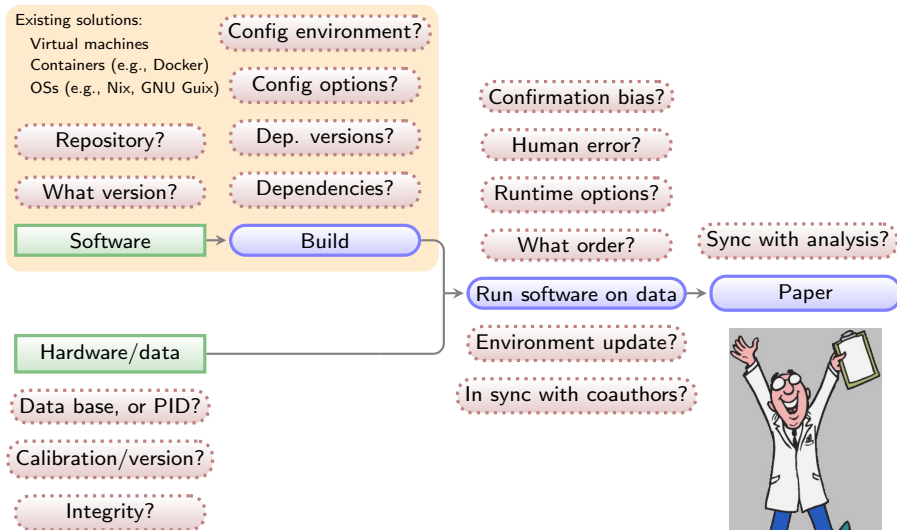
<https://heywhatwhatdidyousay.wordpress.com>



General outline of a project (after data collection)



General outline of a project (after data collection)

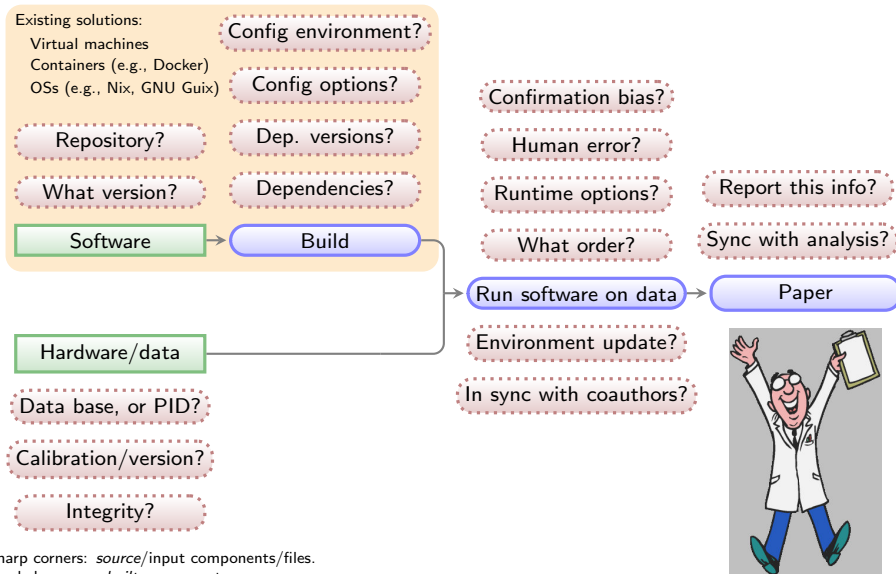


Green boxes with sharp corners: *source*/input components/files.

Blue boxes with rounded corners: *built* components.

Red boxes with dashed borders: questions that must be clarified for each phase.

General outline of a project (after data collection)

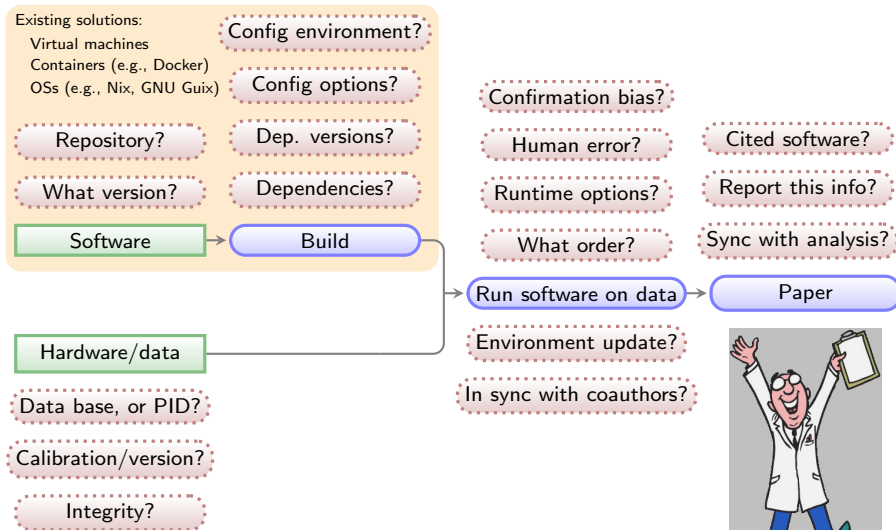


Green boxes with sharp corners: *source*/input components/files.

Blue boxes with rounded corners: *built* components.

Red boxes with dashed borders: questions that must be clarified for each phase.

General outline of a project (after data collection)



Green boxes with sharp corners: *source*/input components/files.

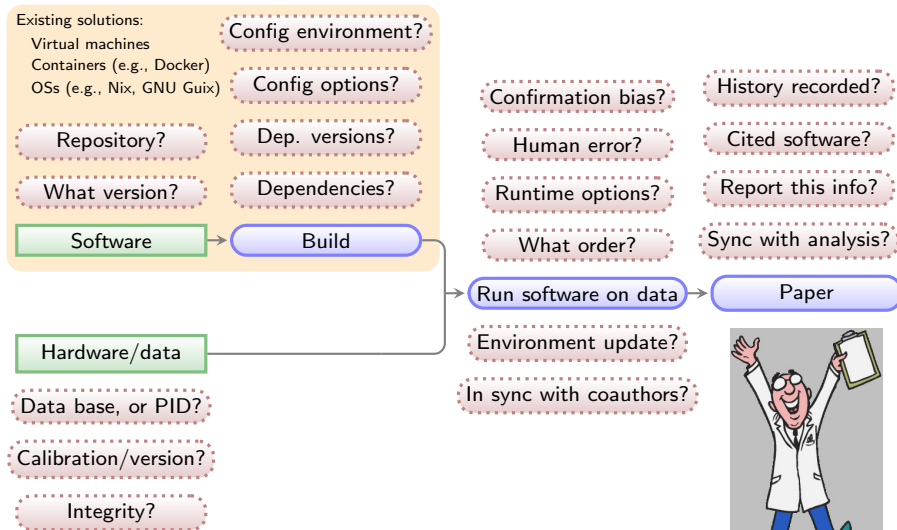
Blue boxes with rounded corners: *built* components.

Red boxes with dashed borders: questions that must be clarified for each phase.

Di Cosmo & Pellegrini (2019) Encouraging a wider usage of software derived from research

“Software is a hybrid object in the world research as it is equally a driving force (as a **tool**), a **result** (as proof of the existence of a solution) and an **object of study** (as an artefact)”.

General outline of a project (after data collection)

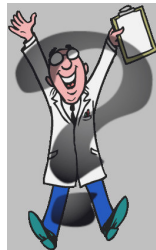
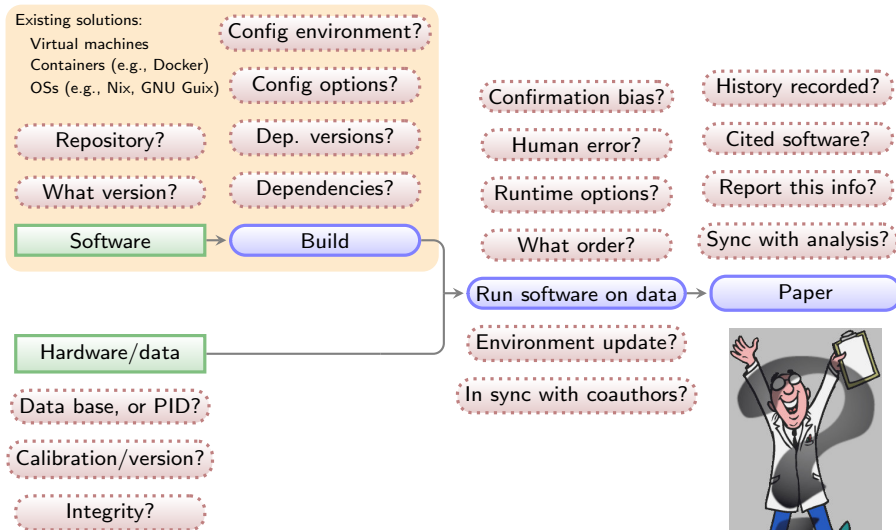


Green boxes with sharp corners: *source*/input components/files.

Blue boxes with rounded corners: *built* components.

Red boxes with dashed borders: questions that must be clarified for each phase.

General outline of a project (after data collection)



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**.

Five ways to fix statistics (Nature, 551, Nov 2017; DOI:[10.1038/d41586-017-07522-z](https://doi.org/10.1038/d41586-017-07522-z)).

Buckheit & Donoho (1996) Lecture Notes in Statistics (vol 103, DOI:10.1007/978-1-4612-2544-7_5)

“An **article** about computational science [*today: almost all sciences*] ... is not the scholarship itself, it is merely **ADVERTISING** of the **SCHOLARSHIP**.”

Buckheit & Donoho (1996) Lecture Notes in Statistics (vol 103, DOI:10.1007/978-1-4612-2544-7_5)

“An **article** about computational science [*today: almost all sciences*] ... is not the scholarship itself, it is merely **ADVERTISING** of the **SCHOLARSHIP**.

The **ACTUAL SCHOLARSHIP** is the **complete software development environment** and the **complete set of instructions** which generated the figures.”

Principles behind proposed solution

Basic/simple principle:

Science is defined by its METHOD, **not** its result.

Principles behind proposed solution

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).

Principles behind proposed solution

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).

Principles behind proposed solution

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).

Principles behind proposed solution

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.

Principles behind proposed solution

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.

Principles behind proposed solution

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 $\times 100\text{KB}$), archivable, and parse-able.
 - ▶ **Version control** (e.g., with Git) can track project's history.

Principles behind proposed solution

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 $\times 100\text{KB}$), archivable, and parse-able.
 - ▶ **Version control** (e.g., with Git) can track project's history.
- ▶ **Minimal complexity:** Occum's razor: "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**.

Principles behind proposed solution

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 $\times 100\text{KB}$), archivable, and parse-able.
 - ▶ **Version control** (e.g., with Git) can track project's history.
- ▶ **Minimal complexity:** Occum's razor: "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**.

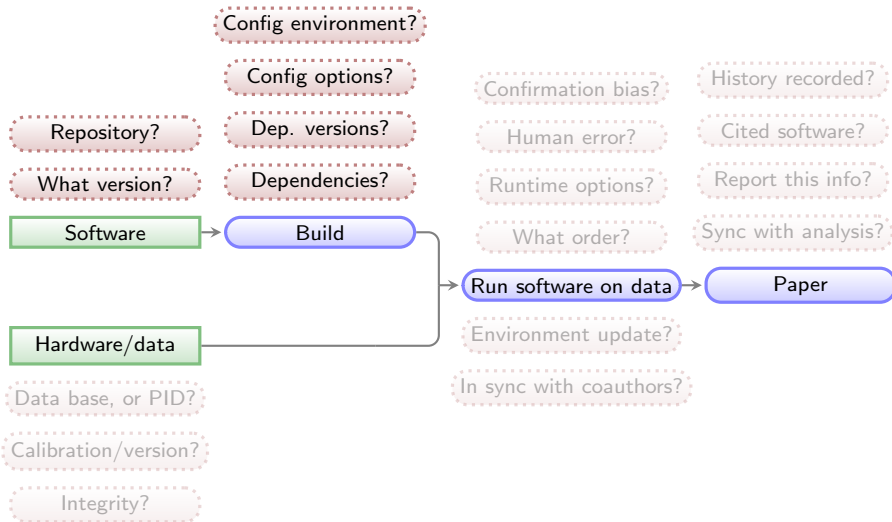
Principles behind proposed solution

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 $\times 100\text{KB}$), archivable, and parse-able.
 - ▶ **Version control** (e.g., with Git) can track project's history.
- ▶ **Minimal complexity:** Occum's razor: "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)



Green boxes with sharp corners: *source*/input components/files.

Blue boxes with rounded corners: *built* components.

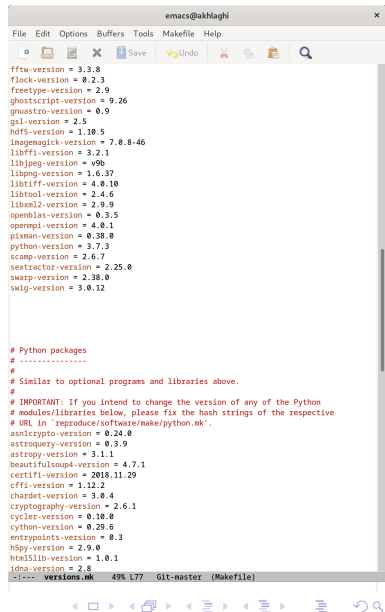
Red boxes with dashed borders: questions that must be clarified for each phase.

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**.
- ▶ Manage **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**).



```
emacs@akhlaghi
File Edit Options Buffers Tools Makefile Help
[Icons] Save Undo [Icons]

fftw-version = 3.3.8
flock-version = 0.2.3
freetype-version = 2.9
ghostscript-version = 9.26
gnuastro-version = 0.9
gsl-version = 2.5
hdf5-version = 1.10.5
imagemagick-version = 7.0.8-46
libffi-version = 3.2.1
libjpeg-version = v9b
libpng-version = 1.6.37
libtiff-version = 4.0.10
libtool-version = 2.4.6
libxml2-version = 2.9.9
openblas-version = 0.3.5
openmpi-version = 4.0.1
pixman-version = 0.38.0
python-version = 3.7.3
scamp-version = 2.6.7
sextractor-version = 2.25.0
swarp-version = 2.38.0
swig-version = 3.0.12

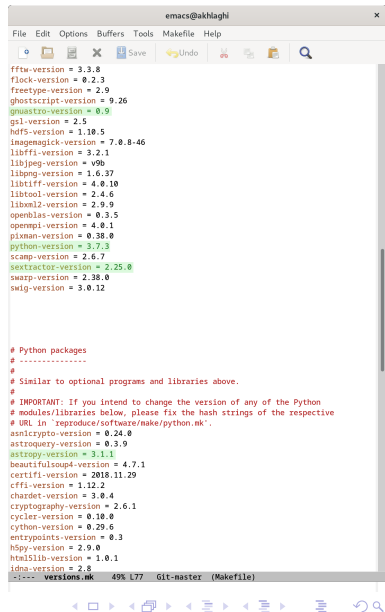
# Python packages
# -----
#
# Similar to optional programs and libraries above.
#
# IMPORTANT: If you intend to change the version of any of the Python
# modules/libraries below, please fix the hash strings of the respective
# URL in 'reproduce/software/make/python.mk'.
asn1crypto-version = 0.24.0
astroquery-version = 0.3.9
astropy-version = 3.1.1
beautifulsoup4-version = 4.7.1
certifi-version = 2018.11.29
cffi-version = 1.12.2
chardet-version = 3.0.4
cryptography-version = 2.6.1
cycler-version = 0.10.0
cython-version = 0.29.6
entrypoints-version = 0.3
h5py-version = 2.9.0
html5lib-version = 1.0.1
idna-version = 2.8
-i--- versions.mk 49% L77 Git-master (Makefile)
```


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**.
- ▶ Manage **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**).



```
emacs@akhlaghi
File Edit Options Buffers Tools Makefile Help
[Icons] Save Undo Redo Find

fftw-version = 3.3.8
flock-version = 0.2.3
freetype-version = 2.9
ghostscript-version = 9.26
gnuastro-version = 0.9
gsl-version = 2.5
hdf5-version = 1.10.5
imagemagick-version = 7.0.8-46
libffi-version = 3.2.1
libjpeg-version = v9b
libpng-version = 1.6.37
libtiff-version = 4.0.10
libtool-version = 2.4.6
libxml2-version = 2.9.9
openblas-version = 0.3.5
openmpi-version = 4.0.1
pixman-version = 0.38.0
python-version = 3.7.3
scamp-version = 2.6.7
sexttractor-version = 2.25.0
swarp-version = 2.38.0
swig-version = 3.0.12

# Python packages
# -----
#
# Similar to optional programs and libraries above.
#
# IMPORTANT: If you intend to change the version of any of the Python
# modules/libraries below, please fix the hash strings of the respective
# URL in 'reproduce/software/make/python.mk'.
asn1crypto-version = 0.24.0
astroquery-version = 0.3.9
astropy-version = 3.1.1
beautifulsoup4-version = 4.7.1
certifi-version = 2018.11.29
cffi-version = 1.12.2
chardet-version = 3.0.4
cryptography-version = 2.6.1
cyclizer-version = 0.10.0
cython-version = 0.29.6
entrypoints-version = 0.3
h5py-version = 2.9.0
html5lib-version = 1.0.1
idna-version = 2.8
urllib3-version = 1.24.2

--- versions.mk 49% L77 Git-master (Makefile)
```


Controlled environment and build instructions

```
emacs@akhlaghi
File Edit Options Buffers Tools Makefile Help

include reproduce/software/config/installation/textlive.mk
include reproduce/software/config/installation/versions.mk

lockdir = $(BDIR)/locks
tdir = $(BDIR)/software/tarballs
ddir = $(BDIR)/software/build-tmp
idir = $(BDIR)/software/installed
lbidir = $(BDIR)/software/installed/bin
lldir = $(BDIR)/software/installed/lib
dtxdir = $(shell pwd)/reproduce/software/bibtex
itidir = $(BDIR)/software/installed/version-info/tex
lctdir = $(BDIR)/software/installed/version-info/cite
ipydir = $(BDIR)/software/installed/version-info/python
lbidir = $(BDIR)/software/installed/version-info/proglib

# Set the top-level software to build.
all: $(foreach p, $(top-level-programs), $(lbidir)/$(p)) \
      $(foreach p, $(top-level-python), $(ipydir)/$(p)) \
      $(itidir)/textlive

# Other basic environment settings: We are only including the host
# operating system's PATH environment variable (after our own!) for the
# compiler and linker. For the library binaries and headers, we are only
# using our internally built libraries.
#
# To investigate:
#
# 1) Set SHELL to '$(lbidir)/env - NAME=VALUE $(lbidir)/bash' and set all
# the parameters defined below as 'NAME=VALUE' statements before
# calling Bash. This will enable us to completely ignore the user's
# native environment.
#
# 2) Add '--noprofile --norc' to '.SHELLFLAGS' so doesn't load the
# user's environment.
.SHELL:
.SHELLFLAGS := --noprofile --norc -ec
export CCACHE_DISABLE := 1
export PATH := $(lbidir)
export SHELL := $(lbidir)/bash
export CPPFLAGS := -I$(idir)/include
export PKG_CONFIG_PATH := $(lldir)/pkgconfig
export PKG_CONFIG_LIBDIR := $(lldir)/pkgconfig
export LD_RUN_PATH := $(lldir):$(lib64dir)
export LD_LIBRARY_PATH := $(lldir):$(lib64dir)
export LDFLAGS := $(rpath_command) -L$(lldir)

# We want the download to happen on a single thread. So we need to define a
# lock, and call a special script we have written for this job. These are
U:--- high-level.mk 4% L81 Git:master (Makefile)
```

```
emacs@akhlaghi
File Edit Options Buffers Tools Makefile Help

# not 'LIBS'.
#
# On Mac systems, the build complains about 'clang' specific
# features, so we can't use our own GCC build here.
if [ x$(on_mac_os) = yes ]; then \
  export CC=clang; \
  export CXX=clang++; \
fi; \
cd $(ddir) \
&& rm -rf cmake-$(cmake-version) \
&& tar xf $< \
&& cd cmake-$(cmake-version) \
&& ./bootstrap --prefix=$(idir) --system-curl --system-zlib \
--system-bzip2 --system-liblzma --no-qt-gui \
&& make -j$(numthreads) LIBS="$LIBS -ls1 -lcrypto -lz" VERBOSE=1 \
&& make install \
&& cd .. \
&& rm -rf cmake-$(cmake-version) \
&& echo "CMake $(cmake-version)" > $@

$(lbidir)/ghostscript: $(tdir)/ghostscript-$(ghostscript-version).tar.gz
$(call gbuild, $<, ghostscript-$(ghostscript-version)) \
&& echo "GPL Ghostscript $(ghostscript-version)" > $@

$(lbidir)/gnustro: $(tdir)/gnustro-$(gnustro-version).tar.lz \
$(lbidir)/ghostscript \
$(lbidir)/libjpeg \
$(lbidir)/libtiff \
$(lbidir)/libgit2 \
$(lbidir)/wcslib \
$(lbidir)/gs1
ifeq ($(static_build),yes)
staticopts="--enable-static=yes --enable-shared=no";
endif
$(call gbuild, $<, gnustro-$(gnustro-version), static, \
$staticopts, -j$(numthreads), \
make check -j$(numthreads)) \
&& cp $(dtxdir)/gnustro.tex $(lctdir) \
&& echo "GNU Astronomy Utilities $(gnustro-version) \cite{gnustro}" > $@

$(lbidir)/imagemagick: $(tdir)/imagemagick-$(imagemagick-version).tar.xz \
$(lbidir)/libjpeg \
$(lbidir)/libtiff \
$(lbidir)/zlib
$(call gbuild, $<, ImageMagick-$(imagemagick-version), static, \
--without-x --disable-openmp, V=1) \
&& echo "ImageMagick $(imagemagick-version)" > $@

U:--- high-level.mk 67% L584 Git:master (Makefile)
```


Controlled environment and build instructions

```
emacs@akhlaghi
File Edit Options Buffers Tools Makefile Help

include reproduce/software/config/installation/textlive.mk
include reproduce/software/config/installation/versions.mk

lockdir = $(BDIR)/locks
tdir = $(BDIR)/software/tarballs
ddir = $(BDIR)/software/build-tmp
idir = $(BDIR)/software/installed
lbidir = $(BDIR)/software/installed/bin
lldir = $(BDIR)/software/installed/lib
dtxdir = $(shell pwd)/reproduce/software/bibtex
itidir = $(BDIR)/software/installed/version-info/tex
lctdir = $(BDIR)/software/installed/version-info/cite
ipydir = $(BDIR)/software/installed/version-info/python
lbidir = $(BDIR)/software/installed/version-info/proglib

# Set the top-level software to build.
all: $(foreach p, $(top-level-programs), $(lbidir)/$(p)) \
      $(foreach p, $(top-level-python), $(ipydir)/$(p)) \
      $(itidir)/textlive

# Other basic environment settings: We are only including the host
# operating system's PATH environment variable (after our own!) for the
# compiler and linker. For the library binaries and headers, we are only
# using our internally built libraries.
#
# To investigate:
#
# 1) Set SHELL to '$(lbidir)/env - NAME=VALUE $(lbidir)/bash' and set all
# the parameters defined below as 'NAME=VALUE' statements before
# calling Bash. This will enable us to completely ignore the user's
# native environment.
#
# 2) Add '--noprofile --norc' to '.SHELLFLAGS' so doesn't load the
# user's environment.
.SHELL:
.SHELLFLAGS := --noprofile --norc -ec
export CCACHE_DISABLE := 1
export PATH := $(lbidir)
export SHELL := $(lbidir)/bash
export CPPFLAGS := -I$(idir)/include
export PKG_CONFIG_PATH := $(lbidir)/pkgconfig
export PKG_CONFIG_LIBDIR := $(lbidir)/pkgconfig
export LD_RUN_PATH := $(lbidir)/$(lldir)
export LD_LIBRARY_PATH := $(lbidir)/$(lldir)
export LDFLAGS := $(lbidir)/$(lldir)

# We want the download to happen on a single thread. So we need to define a
# lock, and call a special script we have written for this job. These are
U:--- high-level.mk 4% L81 Git:master (Makefile)
```

```
emacs@akhlaghi
File Edit Options Buffers Tools Makefile Help

# not 'LIBS'.
#
# On Mac systems, the build complains about 'clang' specific
# features, so we can't use our own GCC build here.
if [ x$(on_mac_os) = yes ]; then \
  export CC=clang; \
  export CXX=clang++; \
fi; \
cd $(ddir) \
&& rm -rf cmake-$(cmake-version) \
&& tar xf < \
&& cd cmake-$(cmake-version) \
&& ./bootstrap --prefix=$(idir) --system-curl --system-zlib \
--system-bzip2 --system-liblzma --no-qt-gui \
&& make -j$(numthreads) LIBS="$LIBS -ls1 -lcrypto -lz" VERBOSE=1 \
&& make install \
&& cd .. \
&& rm -rf cmake-$(cmake-version) \
&& echo "CMake $(cmake-version)" > $@

$(lbidir)/ghostscript: $(tdir)/ghostscript-$(ghostscript-version).tar.gz
$(call gbuild, $*, ghostscript-$(ghostscript-version)) \
&& echo "GPL Ghostscript $(ghostscript-version)" > $@

$(lbidir)/gnustro: $(tdir)/gnustro-$(gnustro-version).tar.lz \
$(lbidir)/ghostscript \
$(lbidir)/libjpeg \
$(lbidir)/libtiff \
$(lbidir)/libgit2 \
$(lbidir)/wcslib \
$(lbidir)/gs1
ifeq ($(static_build),yes)
staticopts="--enable-static=yes --enable-shared=no";
endif
$(call gbuild, $*, gnustro-$(gnustro-version), static, \
$staticopts, -j$(numthreads), \
make check -j$(numthreads)) \
&& cp $(dtxdir)/gnustro.tex $(lctdir) \
&& echo "GNU Astronomy Utilities $(gnustro-version) \cite{gnustro}" > $@

$(lbidir)/imamagick: $(tdir)/imamagick-$(imamagick-version).tar.xz \
$(lbidir)/libjpeg \
$(lbidir)/libtiff \
$(lbidir)/zlib
$(call gbuild, $*, ImageMagick-$(imamagick-version), static, \
--without-x --disable-openssl, V=1) \
&& echo "ImageMagick $(imamagick-version)" > $@

U:--- high-level.mk 67% L584 Git:master (Makefile)
```


Example: Matplotlib (a Python visualization library) build dependencies

Matplotlib library

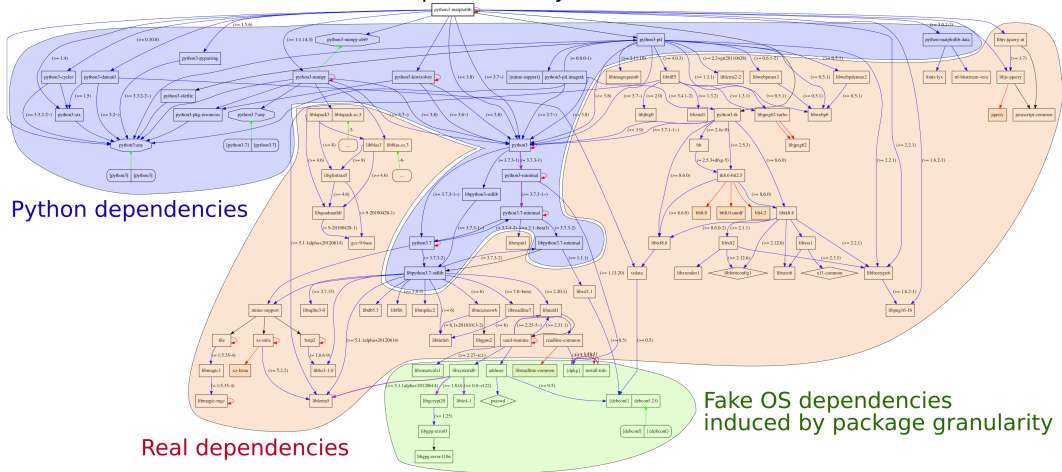


Fig. 1. Transitive dependencies of the software environment required by a simple "import matplotlib" command in the Python 3 interpreter.

All high-level dependencies are under control (e.g., NoiseChisel's dependencies)

GNU/Linux distribution

```
$ ldd .local/bin/astnoisechisel
libgnuastro.so.7 => /PROJECT/libgnuastro.so.7 (0x00007f6745f39000)
libgit2.so.26 => /PROJECT/libgit2.so.26 (0x00007f6745df1000)
libtiff.so.5 => /PROJECT/libtiff.so.5 (0x00007f6745d77000)
liblzma.so.5 => /PROJECT/liblzma.so.5 (0x00007f6745d4f000)
libjpeg.so.9 => /PROJECT/libjpeg.so.9 (0x00007f6745d12000)
libwcs.so.6 => /PROJECT/libwcs.so.6 (0x00007f6745ba8000)
libcfitsio.so.8 => /PROJECT/libcfitsio.so.8 (0x00007f674588b000)
libcurl.so.4 => /PROJECT/libcurl.so.4 (0x00007f6745811000)
libssl.so.1.1 => /PROJECT/libssl.so.1.1 (0x00007f6745777000)
libcrypto.so.1.1 => /PROJECT/libcrypto.so.1.1 (0x00007f6745491000)
libz.so.1 => /PROJECT/libz.so.1 (0x00007f6745474000)
libgsl.so.23 => /PROJECT/libgsl.so.23 (0x00007f67451e3000)
libgslcblas.so.0 => /PROJECT/libgslcblas.so.0 (0x00007f67451a1000)
linux-vdso.so.1 (0x00007ffffdcfbf7000)
libpthread.so.0 => /usr/lib/libpthread.so.0 (0x00007f6745006000)
libm.so.6 => /usr/lib/libm.so.6 (0x00007f6745027000)
libc.so.6 => /usr/lib/libc.so.6 (0x00007f6744e43000)
libdl.so.2 => /usr/lib/libdl.so.2 (0x00007f6744e1e000)
/lib64/ld-linux-x86-64.so.2 => /usr/lib64/ld-linux-x86-64.so.2
```

macOS

```
$ otool -L .local/bin/astnoisechisel
/PROJECT/libgnuastro.7.dylib (comp ver 8.0.0, cur ver 8.0.0)
/PROJECT/libgit2.26.dylib (comp ver 26.0.0, cur ver 0.26.0)
/PROJECT/libtiff.5.dylib (comp ver 10.0.0, cur ver 10.0.0)
/PROJECT/liblzma.5.dylib (comp ver 8.0.0, cur ver 8.4.0)
/PROJECT/libjpeg.9.dylib (comp ver 12.0.0, cur ver 12.0.0)
/PROJECT/libwcs.6.2.dylib (comp ver 6.0.0, cur ver 6.2.0)
/PROJECT/libcfitsio.8.dylib (comp ver 8.0.0, cur ver 8.3.47)
/PROJECT/libcurl.4.dylib (comp ver 10.0.0, cur ver 10.0.0)
/PROJECT/libssl.1.1.dylib (comp ver 1.1.0, cur ver 1.1.0)
/PROJECT/libcrypto.1.1.dylib (comp ver 1.1.0, cur ver 1.1.0)
/PROJECT/libz.1.dylib (comp ver 1.0.0, cur ver 1.2.11)
/PROJECT/libgsl.23.dylib (comp ver 25.0.0, cur ver 25.0.0)
/PROJECT/libgslcblas.0.dylib (comp ver 1.0.0, cur ver 1.0.0)
/usr/lib/libSystem.B.dylib (comp ver 1.0.0, cur ver 1252.50.4)
```

Project libraries: High-level libraries built from source for each project (note the same version in both OSs).
GNU C Library: Project specific build is in progress (<http://savannah.nongnu.org/task/?15390>).
Closed operating system files: We have no control on low-level non-free operating systems components.

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**, **L^AT_EX**, 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).



<https://natemowry2.wordpress.com>

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡

Software citation automatically generated in paper (including Astropy)

Do not reuse, reuse (pp), Your Month day

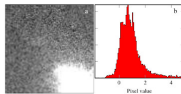


Figure 2: (a) An example image of the Wide-Field Planetary Camera 2, onboard the Hubble Space Telescope from 1993 to 2009. This is one of the sample images from the FITS standard webpage, kept as examples for this file format. (b) Histogram of pixel values in (a).

removes the necessity to add further dependencies (to create the plots) to your project. There are high-level language libraries like Matplotlib which also generate plots. However, the problem is that they require many dependencies (Python, NumPy and etc). Installing these dependencies from source, is not easy and will harm the reproducibility of your paper. Note that after several years, the binary files of these high-level libraries, that you easily install today, will no longer be available in common repositories. Therefore building the libraries from source is the only option to reproduce your results.

Furthermore, since PGPlots is built by IDLX it respects all the properties of your text (for example line width and fonts and etc). Therefore the final plot blends in your paper much more nicely. It also has a wonderful manual².

This template also defines two IDLX macros that allow you to mark text within your document as *new* and *aster*. For example, this text has been marked as *new*. If you comment the line (by adding a “#” at the start of the line or simply deleting the line) that defines *highlightchanges*, then the one that was marked *new* will become black (totally blend in with the rest of the text) and the one marked *aster* will not be in the final PDF. You can thus use *highlightchanges* to easily make copies of your research for existing coauthors (who are just interested in the new parts or notes) and new co-authors (who don't want to be distracted by these issues in their first time reading).

2. NOTICE AND CITATIONS

To encourage other scientists to publish similarly reproducible papers, please add a notice close to the start of your paper or in the end of the abstract clearly mentioning that your work is fully reproducible.

For the time being, we haven't written a specific paper only for this template. Until then, we would be grateful if you could cite the first paper that used the early versions of this template: Akhlaghi and Ichikawa (2015).

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>).

² <http://www.cim.org/graphicviz/FontID/pgsqlbooks/pgplots.pdf>

3. ACKNOWLEDGEMENTS

Please include the following two paragraphs in the Acknowledgement section of your paper. This reproducible paper template was developed in parallel with Gnausto, so it benefited from the same grants. If you don't use Gnausto in your final/customized project, please remove it from the paragraph below, only mentioning the reproducible paper template.

This research was partly done using GNU Astronomy Utilities (Gnausto, mc.net/1801.009), and the reproducible paper template v0.364-2684f5c0-dirty. Work on Gnausto and the reproducible paper template has been funded by the Japanese Ministry of Education, Culture, Sports, Science, and Technology (MEXT) scholarship and its Grant-in-Aid for Scientific Research (21244012, 24251003), the European Research Council (ERC) advanced grant 339659-MUSICOS, European Union's Horizon 2020 research and innovation programme under Marie Skłodowska-Curie grant agreement No 721463 to the SUNSHINE, ITN, and from the Spanish Ministry of Economy and Competitiveness (MINECO) under grant number AYA2016-76210-P.

This research was done with the following free software programs and libraries: Bzip2 1.0.6, Cfitsio 3.45, CMake 3.14.2, cURL 7.63.0, Discovry flock 0.2.3, File 5.36, FreeType 2.9, Git 2.21.0, GNU Astronomy Utilities 0.9 (Akhlaghi and Ichikawa 2015), GNU AWK 5.0.0, GNU Bash 5.0.7, GNU Binutils 2.32, GNU Compiler Collection (GCC) 9.1.0, GNU Coreutils 8.31, GNU Diffutils 3.7, GNU Findutils 4.6.0-199+e3c, GNU Grep 3.3, GNU Gzip 1.10, GNU Integer Set Library 0.18, GNU Libtool 2.4.6, GNU M4 1.4.18, GNU Make 4.2.90, GNU Multiple Precision Arithmetic Library 6.1.2, GNU Multiple Precision Complex Library, GNU Multiple Precision Floating-Point Reliability 4.0.2, GNU NCURSES 6.1, GNU Readline 8.0, GNU Scientific Library 2.5, GNU Sed 4.5, GNU Tar 1.32, GNU Wget 1.20.3, GNU Which 2.21, GPL Ghostscript 9.26, HDF5 library 1.10.5, ImageMagick 7.0.8-46, Libtool 0.9.1, Libtiff 4.2.0, Libjpeg v9b, Libpng 1.6.37, Libtiff 4.0.10, Lisp 1.20, Measure (forked) 1.1.2-2d49f7b8, OpenBLAS 0.3.5, Open MPI 4.0.1, OpenSSL 1.1.1a, Patches 1.0.9, pkg-config 0.29.2, Python 3.7.3, Unzip 6.0, WCSLIB 6.2, XZ Utils 5.2.4, Zip 3.0 and Zlib 1.2.11. Within Python, the following modules were used: Astropy 3.1.1 (Astropy Collaboration et al. 2013; Astropy Collaboration et al. 2018), Cython 0.10.0, Cylind 0.20.6 (Redel et al. 2011), idpy 2.0.0, Kiwisolver 1.0.1, Matplotlib 3.0.2 (Hunter 2007), Numpy 1.16.2 (van der Walt et al. 2011), pkgconfig 1.5.1, PyParsing 2.3.1, python-datatool 2.8.0, Scipy 1.2.1 (Oliphant 2007; Millman and Aivazis 2011), Setuptools 40.8.0, Setuptools-scm 3.2.0 and Sphinx 1.2.0. The IDLX source of the paper was compiled to make the PDF using the following packages: bibex 2.12, bibex 2.12, biblatex 3.12, biblatex 3.12, caption 2018-10-05, caption 2018-10-05, courier 2016-06-24, courier 2016-06-24, courier 5.2d, dateime 2.60, dateime 2.60, ex 1.8, ex 1.0, etoolbox 2.5f, etoolbox 2.5f, fontcabin 3.10, fontcabin 3.10, fontcabin 3.05, fontcabin 3.05, fontcabin 1.0d, fontcabin 1.0d, fontcabin 5.5b, fontcabin 5.5b, fp 2.1d, fp 2.1d, logreq 1.0, logreq 1.0, newts 1.554, newts 1.554, pdf 3.1.2, pdf 3.1.2, pgplots 1.16, pgplots 1.16, preprint 2011, preprint 2011, sequnce 6.7a, sequnce 6.7a, tex 3.14159265, tex 3.14159265, texgym 2.501, texgym 2.501, times 2016-06-24, times 2016-06-24, times 2.10.2, times 2.10.2, ttfoms 2016-

YOUR NAME et al.,

Do not reuse, reuse (pp), Your Month day

06-24, ttfoms 2016-06-24, ttfoms 2016-06-24, ttfoms 2016-06-24, xcolor 2.12, xcolor 2.12, xkeyval 2.7a and xkeyval 2.7a. We are very grateful to all their creators for freely providing this necessary infrastructure. This research (and many others) would not be possible without them.

References

- Akhlaghi, M. and T. Ichikawa (Sept. 2015). *ApJS*, 220, 1.
- Astropy Collaboration et al. (Oct. 2013). *A&A*, 558, A33.
- Astropy Collaboration et al. (Sept. 2018). *AJ*, 156, 125.
- Bacon, R. et al. (Dec. 2015). *A&A*, 576, A1.
- Bellini, S. et al. (Mar. 2011). *CSE*, 13, 31.
- Hunter, J. D. (2007). *CSE*, 9, 90.
- Millman, K. J. and M. Aivazis (Mar. 2011). *CSE*, 13, 9.
- Oliphant, T. E. (May 2007). *CSE*, 9, 90.
- van der Walt, S. et al. (Mar. 2011). *CSE*, 13, 22.

YOUR NAME et al.,

Appendix A: Software acknowledgement

The reproducible paper template that is customized for this project automatically installs all the necessary software. Directly listing all the high-level software and their versions is done with two primary motives: 1) software citation and acknowledgement of the hard work (as part of different software projects) that this project utilized; 2) reproducibility for (future) readers.

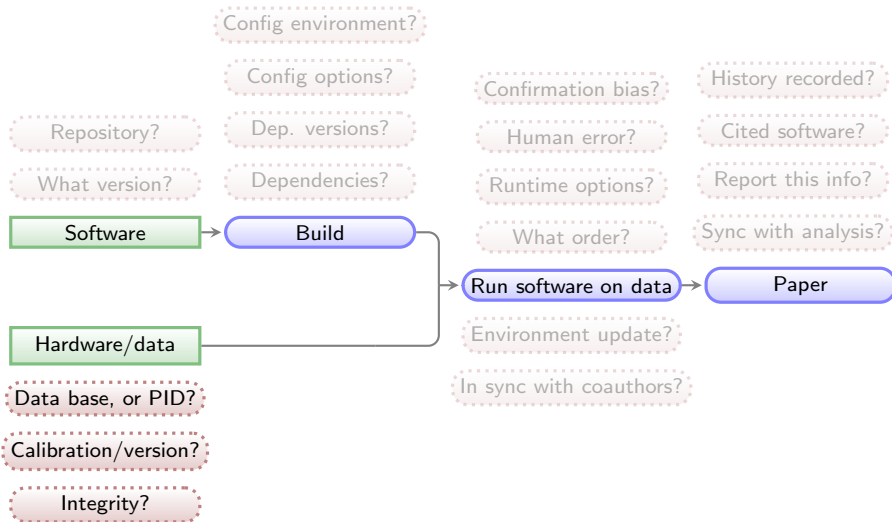
This research was done with the following free software programs and libraries: Brup2 1.0.6, CPITSIO 3.47, CMake 3.14.2, curl 7.65.0, Discotek Book 0.2.3, File 5.36, Git 2.22.0, GNU Astronomy Utilities 0.9.170-16fc (Akhlaghi and Ishikawa, 2015), GNU AWK 5.0.0, GNU Bash 5.0.7, GNU Binutils 2.32, GNU Compiler Collection (GCC) 9.1.0, GNU Coreutils 8.31, GNU Diffutils 3.7, GNU Findutils 4.6.0-199-c6c, GNU Grep 3.3, GNU Gzip 1.10, GNU Integer Set Library 0.18, GNU Libtool 2.4.6, GNU M4 1.4.18, GNU Make 4.2.90, GNU Multiple Precision Arithmetic Library 6.1.2, GNU Multiple Precision Complex Library, GNU Multiple Precision Floating-Point Reliably 4.0.2, GNU NCURSES 6.1, GNU Realtime 8.0, GNU Scientific Library 2.5, GNU Sed 4.7, GNU Tar 1.32, GNU Wget 1.20.3, GNU Which 2.21, GPL Ghostscript 9.26, Libbsd 0.9.1, Libg2 0.28.2, Libjpeg v8, Libtiff 4.0.10, Lzip 1.20, MetaStore (forked) 1.12-23-fa9170b, OpenSSL 1.1.1a, PatchELF 0.9, pkg-config 0.29.2, Unzip 6.0, WCSLIB 6.2, XZ Utils 5.2.4, Zip 3.0 and Zlib 1.2.11. The HUGO source of the paper was compiled to make the PDF using the following packages: hiber 2.12, biblatex 3.12, caption 2018-10-05, charter 2016-06-24, counter 2016-06-24, csquotes 5.2d, datatime 2.60, ec 1.0, ecrimon 0.3, ebookbox 2.5f, etexvars 1.8a, fancyhdr 3.10, fonticore 3.05, fontaxes 1.0d, font-misc 5.5b, fp 2.1d, helvetica 2016-06-24, lineno 4.41, logreq 1.0, newtx 1.554, pdf 3.1.2, pgplots 1.16, preprint 2011, setspace 6.7a, smoke 2.0, scolarbox 4.20, tex 3.14159265, texgym 2.501, times 2016-06-24, titelsec 2.10.2, trimspaces 1.1, txfonts 2016-06-24, ulen 2016-06-24, scolar 2.12 and xkeyval 2.7a. We are very grateful to all their creators for freely providing this necessary infrastructure. This research (and many others) would not be possible without them.

Appendix A: Software acknowledgement

The reproducible paper template that is customized for this project automatically installs all the necessary software. Directly listing all the high-level software and their versions is done with two primary motives: 1) software citation and acknowledgement of the hard work (as part of different software projects) that this project utilized; 2) reproducibility for (future) readers.

This research was done with the following free software programs and libraries: Brp2 1.0.6, Cfitsio 3.47, CMake 3.14.2, curl 7.65.0, Discoteq Box 0.2.3, File 5.36, Git 2.22.0, GNU Astronomy Utilities 0.9.170-1bfc (Akhlaghi and Likhawa 2015), GNU AWK 5.0.0, GNU Bash 5.0.7, GNU Binutils 2.32, GNU Compiler Collection (GCC) 9.1.0, GNU Coreutils 8.31, GNU Diffutils 3.7, GNU Findutils 4.6.0-199-cb6, GNU Grep 3.3, GNU Gzip 1.10, GNU Integer Set Library 0.18, GNU Libtool 2.4.6, GNU M4 1.4.18, GNU Make 4.2.90, GNU Multiple Precision Arithmetic Library 6.1.2, GNU Multiple Precision Complex Library, GNU Multiple Precision Floating-Point Reliability 4.0.2, GNU NCURSES 6.1, GNU Realtime 8.0, GNU Scientific Library 2.5, GNU Sed 4.7, GNU Tar 1.32, GNU Wget 1.20.3, GNU Which 2.21, GPL Ghostscript 9.26, Libbsd 0.9.1, Libg2 0.28.2, Libjpeg v8b, Libtiff 4.0.10, Lzip 1.20, Metasploit (forked) 1.1.2-23-6a9170b, OpenSSL 1.1.1a, PatchELF 0.9, pkg-config 0.29.2, Unzip 6.0, WCSLIB 6.2, XZ Utils 5.2.4, Zip 3.0 and Zlib 1.2.11. The HUGO source of the paper was compiled to make the PDF using the following packages: hiber 2.12, biblatex 3.12, caption 2018-10-05, charter 2016-06-24, counter 2016-06-24, csquotes 5.26, datetime 2.66, ee 1.0, environ 0.3, etoolbox 2.5f, etexsize 1.6a, fancyhdr 3.10, fonticnt 3.05, fontaxes 1.0a, four-misc 5.5b, tp 2.1d, helvetica 2016-06-24, lmodern 4.41, logreq 1.0, newtx 1.554, pdf 3.1.2, pgplots 1.16, preprint 2011, setspace 6.7a, stoken 2.0, xcolorbox 4.2a, xes 3.14159265, xergive 2.501, times 2016-06-24, timesec 2.10.2, trimspaces 1.1, txfonts 2016-06-24, ulen 2016-06-24, scolor 2.12 and xkeyval 2.7a. We are very grateful to all their creators for freely providing this necessary infrastructure. This research (and many others) would not be possible without them.

General outline of a project (after data collection)



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

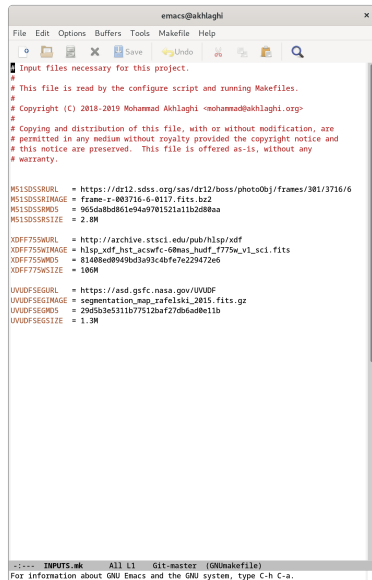
Stored information about each input file:

- ▶ **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](https://arxiv.org/abs/1909.11230).
This paper needs three input files (two images, one catalog).



```
emac@akhlaghi
File Edit Options Buffers Tools Makefile Help
[Icons] Save Undo [Icons] Search

# Input files necessary for this project.
#
# This file is read by the configure script and running Makefiles.
#
# Copyright (C) 2018-2019 Mohammad Akhlaghi <mohammad@akhlaghi.org>
#
# Copying and distribution of this file, with or without modification, are
# permitted in any medium without royalty provided the copyright notice and
# this notice are preserved. This file is offered as-is, without any
# warranty.

W51505SRURL = https://dr12.sdss.org/sas/dr12/boos/photoObj/frames/301/3716/6
W51505SRIMAGE = frame-r-003716-6-0117.fits.bz2
W51505SRMDS = 965da8bd861e94a9701521a11b2d88aa
W51505SRSIZE = 2.8M

XDF775WURL = http://archive.stsci.edu/pub/hlsp/xdff
XDF775WIMAGE = hlsp_xdff_hst_acswfc-60mas_hudf_f775w_v1_sc1.fits
XDF775WMDS = 81408ed0949bd3a93c4bfe7e229472e6
XDF775WSIZE = 106M

UVUDFSEGURL = https://asd.gsfc.nasa.gov/UVUDF
UVUDFSEGIMAGE = segmentation_map_rafelski_2015.fits.gz
UVUDFSEGMDS = 29d5b3e5311b77512ba727db6ad0e11b
UVUDFSEGSIZE = 1.3M

-:--- INPUTS.mk All L1 Git-master (GNUmakefile)
For information about GNU Emacs and the GNU system, type C-h C-a.
```


Input data source and integrity is documented and checked

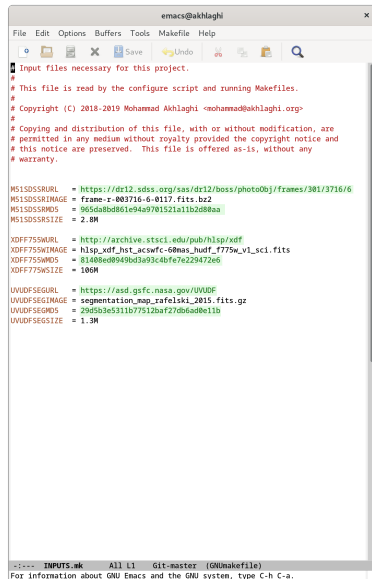
Stored information about each input file:

- ▶ **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](https://arxiv.org/abs/1909.11230).
This paper needs three input files (two images, one catalog).



```
# Input files necessary for this project.
# This file is read by the configure script and running Makefiles.
# Copyright (C) 2018-2019 Mohammad Akhlaghi <mohammad@akhlaghi.org>
# Copying and distribution of this file, with or without modification, are
# permitted in any medium without royalty provided the copyright notice and
# this notice are preserved. This file is offered as-is, without any
# warranty.

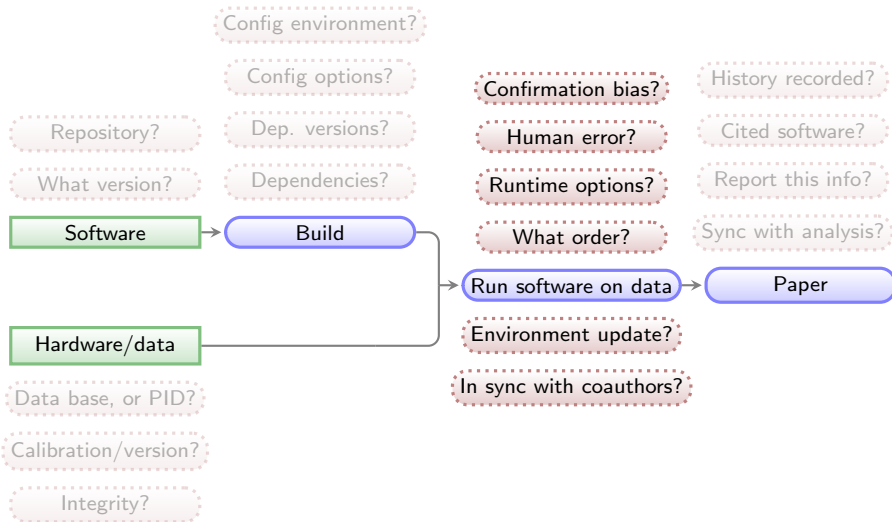
MS1S0SSRURL = https://dr12.sdss.org/sas/dr12/boos/photoObj/frames/301/3716/6
MS1S0SSRIMAGE = frame-r-003716-6-0117.fits.bz2
MS1S0SSRMDS = 965da8bd861e94a9701521a11b2d88aa
MS1S0SSRSIZE = 2.8M

XDFF75SWURL = http://archive.stsci.edu/pub/hlsp/xdff
XDFF75SWIMAGE = hlsp_xdff_hst_acswfc-60mas_hudf_f775w_v1_sc1.fits
XDFF75SWMDS = 81408ed0949bd3a93c4bfe7e229472e6
XDFF75WSIZE = 106M

UVUDFSEGURL = https://asd.gsfc.nasa.gov/UVUDF
UVUDFSEGIMAGE = segmentation_map_rafelski_2015.fits.gz
UVUDFSEGMDS = 29d5b3e5311b77512ba727db6ad0e11b
UVUDFSEGSIZE = 1.3M

-:--- INPUTS.mk All L1 Git-master (GNUmakefile)
For information about GNU Emacs and the GNU system, type C-h C-a.
```


General outline of a project (after data collection)



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](https://zenodo.org/record/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 can identify independent steps internally and do them in **parallel**.
- ▶ 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](#)).



```
# Run NoiseChisel
# -----
# NoiseChisel's output is needed for several things down the line: Its
# Sky and Sky standard deviation outputs will be used in the several
# runs of MakeCatalog. Its detections are also going to be used to
# create a NoiseChisel segmentation map. We also need the Sky values
# for the raw aperture catalogs, so we'll also run NoiseChisel on the
# images with a gradient..
allf = $(acff) $(wfc3f)
ncfdir = $(fdir)/noisechisel
$(ncfdir): | $(fdir); mkdir $@
noisechisel=$(foreach f, $(allfilters), $(ncfdir)/udf_$(f).fits) \
$(foreach f, $(xdfsfc3irf), $(ncfdir)/xdf_$(f).fits) \
$(foreach f, $(xdfsfc3irf), $(ncfdir)/grd_$(f).fits)
$(noisechisel): $(ncfdir)/% $(sdepth)/% .gnuastro/astnoisechisel.conf \
| $(ncfdir)
if [ $* == "udf_f225w.fits" ] || [ $* == "udf_f275w.fits" ] \
|| [ $* == "udf_f336w.fits" ]; then extraopt="--qthresh=0.4"; \
else extraopt=" "; fi;
astnoisechisel $$extraopt --detquant=0.9 --segquant=0.9 $< -o$@

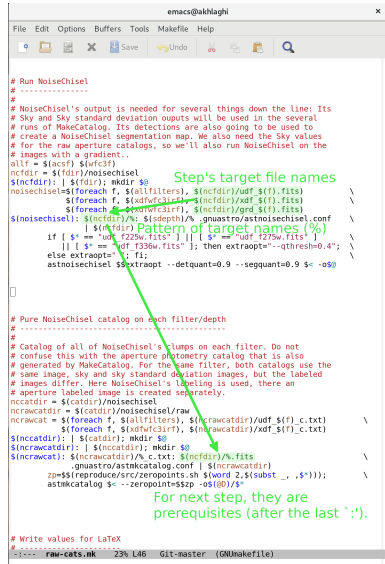
# Pure NoiseChisel catalog on each filter/depth
# -----
# Catalog of all of NoiseChisel's clumps on each filter. Do not
# confuse this with the aperture photometry catalog that is also
# generated by MakeCatalog. For the same filter, both catalogs use the
# same image, sky and sky standard deviation images, but the labeled
# images differ. Here NoiseChisel's labeling is used, there an
# aperture labeled image is created separately.
nccatdir = $(catdir)/noisechisel
ncracatdir = $(catdir)/noisechisel/raw
ncracat = $(foreach f, $(allfilters), $(ncracatdir)/udf_$(f)_c.txt) \
$(foreach f, $(xdfsfc3irf), $(ncracatdir)/xdf_$(f)_c.txt)
$(nccatdir): | $(catdir); mkdir $@
$(ncracatdir): | $(nccatdir); mkdir $@
$(ncracat): $(ncracatdir)/%.c.txt: $(ncfdir)/%.fits \
.gnuastro/astnmkcatalog.conf | $(ncracatdir)
zp=$(reproduce/src/zeropoints.sh $(word 2,$(subst _,,$*))); \
astnmkcatalog $< --zeropoint=$zp -o$(@D)/$*

# Write values for LaTeX
# -----
raw-cats.mk 23% L46 Git-master (GNUmakefile)
```


Reproducible science: Manage is managed through a Makefile

All steps (downloading and analysis) are managed by Makefiles (example from [zenodo.1164774](https://zenodo.org/record/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 can identify independent steps internally and do them in **parallel**.
- ▶ 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](#)).



```
# emacs@akhiaghi
File Edit Options Buffers Tools Makefile Help
[Icons] Save Undo [Icons] [Search]

# Run NoiseChisel
# -----
#
# NoiseChisel's output is needed for several things down the line: Its
# Sky and Sky standard deviation outputs will be used in the several
# runs of MakeCatalog. Its detections are also going to be used to
# create a NoiseChisel segmentation map. We also need the Sky values
# for the raw aperture catalogs, so we'll also run NoiseChisel on the
# images with a gradient..
allf = $(acff) $(wfc3f)
ncfdir = $(fdir)/noisechisel
$(ncfdir): | $(fdir); mkdir $@
noisechisel=$(foreach f, $(allfilters), $(ncfdir)/udf $(f).fits) \
$(foreach f, $(xdfsfc3irf), $(ncfdir)/xdf $(f).fits) \
$(foreach f, $(xdfsfc3irf), $(ncfdir)/grd $(f).fits)
$(noisechisel): $(ncfdir)/%. $(sdepth)/%. gnuastro/astnoisechisel.conf \
| $(ncfdir) Pattern of target names (%)
if [ $* == "udf_f225w.fits" ]; then extraopt="--qthresh=0.4"; \
else extraopt=""; fi;
astnoisechisel $$extraopt --detquant=0.9 --segquant=0.9 $< -o$@

# Pure NoiseChisel catalog on each filter/depth
# -----
#
# Catalog of all of NoiseChisel's clumps on each filter. Do not
# confuse this with the aperture photometry catalog that is also
# generated by MakeCatalog. For the same filter, both catalogs use the
# same image, sky and sky standard deviation images, but the labeled
# images differ. Here NoiseChisel's labeling is used, there an
# aperture labeled image is created separately.
nccatdir = $(catdir)/noisechisel
ncrawcatdir = $(catdir)/noisechisel/raw
ncrawcat = $(foreach f, $(allfilters), $(ncrawcatdir)/udf $(f)_c.txt) \
$(foreach f, $(xdfsfc3irf), $(ncrawcatdir)/xdf $(f)_c.txt)
$(nccatdir): | $(catdir); mkdir $@
$(ncrawcatdir): | $(nccatdir); mkdir $@
$(ncrawcat): $(ncrawcatdir)/%. c.txt: $(ncfdir)/%.fits \
, gnuastro/astmkcatalog.conf | $(ncrawcatdir)
zp=$(reproduce/src/zeropoints.sh $(word 2,$(subst _,,$*))); \
astmkcatalog $< --zeropoint=$zp -o$(@D)/$*

# Write values for LaTeX
# -----
raw-cats.mk 23% L46 Git-master (GNUmakefile)
```

Step's target file names

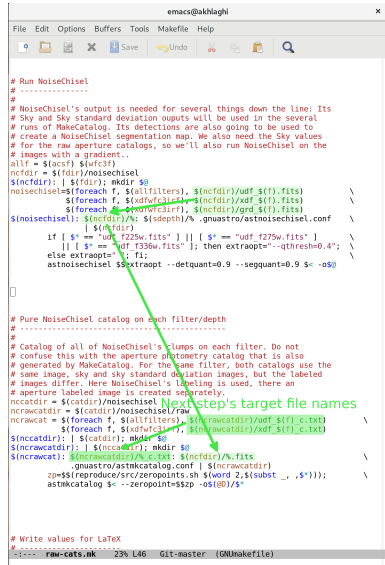
Pattern of target names (%)

For next step, they are prerequisites (after the last `:').

Reproducible science: Maneage is managed through a Makefile

All steps (downloading and analysis) are managed by Makefiles (example from [zenodo.1164774](https://zenodo.org/record/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 can identify independent steps internally and do them in **parallel**.
- ▶ 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](#)).

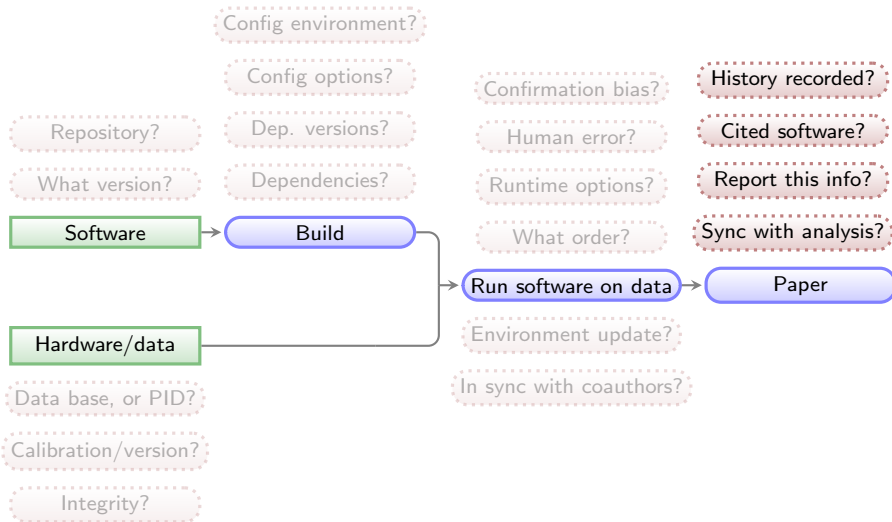


```
# Run NoiseChisel
# -----
#
# NoiseChisel's output is needed for several things down the line: Its
# Sky and Sky standard deviation outputs will be used in the several
# runs of MakeCatalog. Its detections are also going to be used to
# create a NoiseChisel segmentation map. We also need the Sky values
# for the raw aperture catalogs, so we'll also run NoiseChisel on the
# images with a gradient..
allf = $(acff) $(wfc3f)
ncfdir = $(fdir)/noisechisel
$(ncfdir): | $(fdir); mkdir $@
noisechisel=$(foreach f, $(allfilters), $(ncfdir)/udf $(f).fits) \
$(foreach f, $(xdfs3irf), $(ncfdir)/xdf $(f).fits) \
$(foreach f, $(xdfs3irf), $(ncfdir)/grd $(f).fits)
$(noisechisel): $(ncfdir)/% $(sdepth)/% .gnuastro/astnoisechisel.conf \
| $(fdir)
if [ $* == "udf_f225w.fits" ] || [ $* == "udf_f275w.fits" ] \
|| [ $* == "udf_f336w.fits" ]; then extraopt="--qthresh=0.4"; \
else extraopt=""; fi;
astnoisechisel $$extraopt --detquant=0.9 --segquant=0.9 $<-o$@

# Pure NoiseChisel catalog on each filter/depth
# -----
#
# Catalog of all of NoiseChisel's clumps on each filter. Do not
# confuse this with the aperture photometry catalog that is also
# generated by MakeCatalog. For the same filter, both catalogs use the
# same image, sky and sky standard deviation images, but the labeled
# images differ. Here NoiseChisel's labeling is used, there an
# aperture labeled image is created separately.
ncatdir = $(catdir)/noisechisel
ncrcatdir = $(catdir)/noisechisel/raw
ncrcat = $(foreach f, $(allfilters), $(ncrcatdir)/udf $(f)_c.txt) \
$(foreach f, $(xdfs3irf), $(ncrcatdir)/xdf $(f)_c.txt)
$(ncatdir): | $(catdir); mkdir $@
$(ncrcatdir): | $(ncatdir); mkdir $@
$(ncrcat): $(ncrcatdir)/%_c.txt: $(ncfdir)/%.fits \
.gnuastro/astnmkcatalog.conf | $(ncrcatdir)
zp=$(reproduce/src/zeropoints.sh $(word 2,$(subst _,,$*))); \
astnmkcatalog $<- --zeropoint=$zp -o$(@D)/$*

# Write values for LaTeX
# -----
raw-cats.mk 23% L46 Git-master (GNUmakefile)
```


General outline of a project (after data collection)



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 **L^AT_EX macros**. They are thus **updated automatically** on any change.

Shown here is a portion of the NoiseChisel paper and its L^AT_EX source ([arXiv:1505.01664](https://arxiv.org/abs/1505.01664)).

```
\begin{equation}
  \label{tSNeg}
  \mathrm{S/N}_{\mathrm{T}} = \frac{NF - NS_a}{\sqrt{NF + N\sigma_s^2}}
  = \frac{\sqrt{N}(F - S_a)}{\sqrt{F + \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 `\small S/N`_T from the objects in `R_s` 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 `\small S/N` of false detections in real, reduced/co-added images. A comparison of scales on the `\small 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 `\small S/N` of `\onelargedettfmax`, `\sensitivedettfmax`, and `\fourdettfmax`.[□]

smaller than `--detsnminarea` are removed from the analysis in both R_s and R_d . In the examples in this section, it is set to 15. Note that since a threshold approximately equal to the Sky value is used, this is a very weak constraint. For each pseudo-detection, S/N_T can be written as,

$$S/N_T = \frac{NF - NS_a}{\sqrt{NF + N\sigma_s^2}} = \frac{\sqrt{N}(F - S_a)}{\sqrt{F + \sigma_s^2}}. \quad (3)$$

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_s 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_s provides a very ro-

Values in final report/paper

All analysis **results** (numbers, plots, tables) written in paper's PDF as **L^AT_EX macros**. They are thus **updated automatically** on any change.

Shown here is a portion of the NoiseChisel paper and its L^AT_EX source ([arXiv:1505.01664](https://arxiv.org/abs/1505.01664)).

$$\mathrm{S/N}_{-T} = \frac{NF - NS_a}{\sqrt{NF + N\sigma_s^2}} = \frac{\sqrt{N} (F - S_a)}{\sqrt{F + \sigma_s^2}}.$$

\noindent

See Section [\ref{SNeqmodif}](#) for the modifications required when the input image is not in units of counts or has already been Sky subtracted. The distribution of $\{\text{small S/N}\}_T$ from the objects in $\$R_s$ 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 $\{\text{small S/N}\}$ of false detections in real, reduced/co-added images. A comparison of scales on the $\{\text{small S/N}\}$ 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 $\{\text{small S/N}\}$ of $\$ \text{onelargedettfmax}$, $\$ \text{sensitivitycdettfmax}$, and $\$ \text{fourdettfmax}$.

smaller than $-\text{detsminarea}$ are removed from the analysis in both R_s and R_d . In the examples in this section, it is set to 15. Note that since a threshold approximately equal to the Sky value is used, this is a very weak constraint. For each pseudo-detection, S/N_T can be written as,

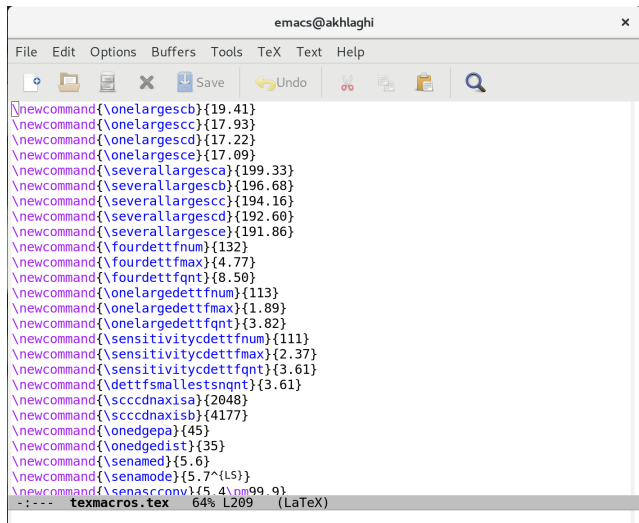
$$S/N_T = \frac{NF - NS_a}{\sqrt{NF + N\sigma_S^2}} = \frac{\sqrt{N}(F - S_a)}{\sqrt{F + \sigma_S^2}}. \quad (3)$$

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_s 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_g provides a very ro-

Analysis step results/values concatenated into a single file.

All \LaTeX macros come from a **single file**.



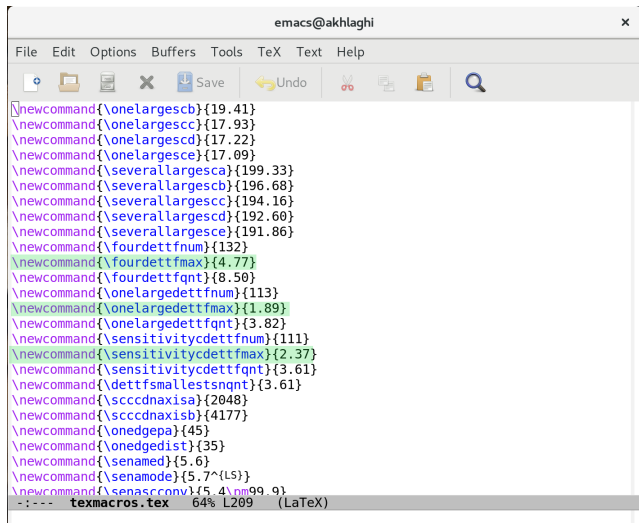
The screenshot shows an Emacs editor window titled "emacs@akhlaghi". The menu bar includes File, Edit, Options, Buffers, Tools, TeX, Text, and Help. The toolbar contains icons for opening a file, saving, undo, redo, and search. The main text area displays a list of LaTeX macros defined in a file named "texmacros.tex". The macros are listed as follows:

```
\newcommand{\onelargescb}{19.41}  
\newcommand{\onelargesccl}{17.93}  
\newcommand{\onelargescd}{17.22}  
\newcommand{\onelargescel}{17.09}  
\newcommand{\severallargescal}{199.33}  
\newcommand{\severallargescb}{196.68}  
\newcommand{\severallargesccl}{194.16}  
\newcommand{\severallargescd}{192.60}  
\newcommand{\severallargescel}{191.86}  
\newcommand{\fourdettfnun}{132}  
\newcommand{\fourdettfmax}{4.77}  
\newcommand{\fourdettfqnt}{8.50}  
\newcommand{\onelargedettfnun}{113}  
\newcommand{\onelargedettfmax}{1.89}  
\newcommand{\onelargedettfqnt}{3.82}  
\newcommand{\sensitivitycdettfnun}{111}  
\newcommand{\sensitivitycdettfmax}{2.37}  
\newcommand{\sensitivitycdettfqnt}{3.61}  
\newcommand{\dettfsmallestsnqnt}{3.61}  
\newcommand{\scccdnaxisa}{2048}  
\newcommand{\scccdnaxisb}{4177}  
\newcommand{\onedgepa}{45}  
\newcommand{\onedgedist}{35}  
\newcommand{\senamed}{5.6}  
\newcommand{\senamode}{5.7^{LS}}  
\newcommand{\senascconv}{5.4^{nm}99.9}
```

The status bar at the bottom of the window shows the file name "texmacros.tex", the encoding "64% L209", and the document type "(LaTeX)".

Analysis step results/values concatenated into a single file.

All \LaTeX macros come from a **single file**.



The screenshot shows an Emacs editor window titled "emacs@akhlaghi". The menu bar includes File, Edit, Options, Buffers, Tools, TeX, Text, and Help. The toolbar contains icons for opening a file, saving, undo, redo, and search. The main text area displays a list of LaTeX macro definitions, each starting with `\newcommand`. The macros are defined with a name and a value in curly braces. The values are numerical or symbolic expressions. The status bar at the bottom shows the file name "texmacros.tex", the cursor position "64%", and the page number "L209".

```
\newcommand{\onelargescb}{19.41}
\newcommand{\onelargescb}{17.93}
\newcommand{\onelargescd}{17.22}
\newcommand{\onelargescd}{17.09}
\newcommand{\severallargescb}{199.33}
\newcommand{\severallargescb}{196.68}
\newcommand{\severallargescb}{194.16}
\newcommand{\severallargescd}{192.60}
\newcommand{\severallargescd}{191.86}
\newcommand{\fourdettfnum}{132}
\newcommand{\fourdettfmax}{4.77}
\newcommand{\fourdettfqnt}{8.50}
\newcommand{\onelargedettfnum}{113}
\newcommand{\onelargedettfmax}{1.89}
\newcommand{\onelargedettfqnt}{3.82}
\newcommand{\sensitivitycdettfnum}{111}
\newcommand{\sensitivitycdettfmax}{2.37}
\newcommand{\sensitivitycdettfqnt}{3.61}
\newcommand{\dettfsmallestsnqnt}{3.61}
\newcommand{\scccdnaxisa}{2048}
\newcommand{\scccdnaxisb}{4177}
\newcommand{\onedgepa}{45}
\newcommand{\onedgedist}{35}
\newcommand{\senamed}{5.6}
\newcommand{\senamode}{5.7^{LS}}
\newcommand{\senascconv}{5.4^{nm}99.9}
```

--- texmacros.tex 64% L209 (LaTeX)

Analysis results stored as \LaTeX macros

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

```
# Numbers for dettf.tex:
sqnt=9999999
function dettfhist
{
  # 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

  dettfnum=$(awk '/points binned in/{print $4; exit(0)}' $name)
  dettfqnt=$(awk '/quantile has a value of/{
    printf("%.2f", $9); exit(0);}' $name)
  dettfmax=$(awk 'BEGIN { max=-999999 }
    !/^#/ { if($2>max){max=$2; mv=$1} }
    END { printf("%.2f", mv) }' $name)
  addtexmacro $obase"dettfnum" $dettfnum
  addtexmacro $obase"dettfmax" $dettfmax
  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 dettfhist $texdir/dettf/ $base; done
addtexmacro dettfsmallestsqnt $sqnt
```


Analysis results stored as \LaTeX macros

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

```
# Numbers for dettf.tex:
sqnt=9999999
function dettfhist
{
  # 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

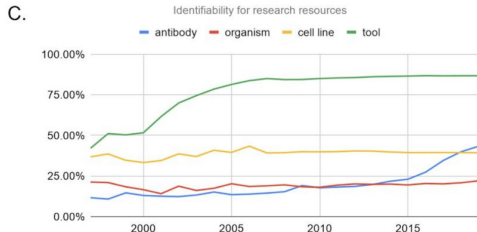
  dettfnum=$(awk '/points binned in/{print $4; exit(0)}' $name)
  dettfqnt=$(awk '/quantile has a value of/{
    printf("%.2f", $9); exit(0);}' $name)
  dettfmax=$(awk 'BEGIN { max=-999999 }
    !/^#/ { if($2>max){max=$2; mv=$1} }
    END { printf("%.2f", mv) }' $name)
  addtexmacro $obase"dettfnum" $dettfnum
  addtexmacro $obase"dettfmax" $dettfmax
  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 dettfhist $texdir/dettf/ $base; done
addtexmacro dettfsmallestsqnt $sqnt
```


Let's look at the data lineage to replicate Figure 1C (green/tool) of Menke+2020
(DOI:10.1101/2020.01.15.908111)

ORIGINAL PLOT

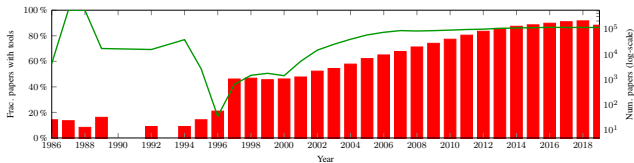
The Green plot shows the fraction of papers mentioning software tools from 1997 to 2019.



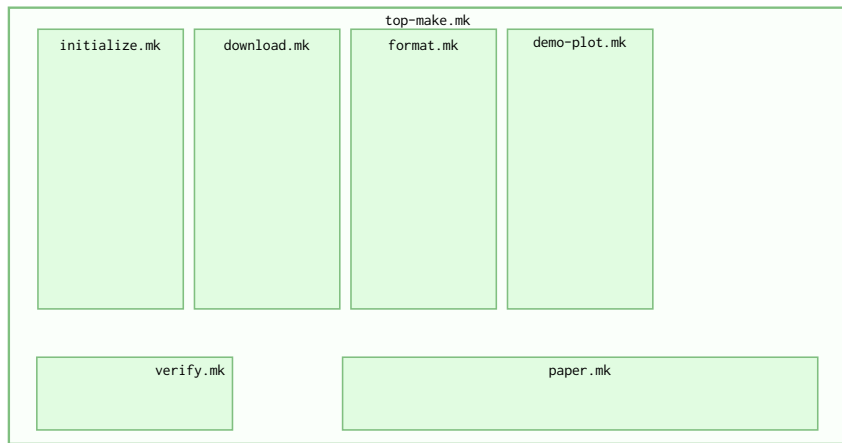
OUR enhanced REPLICATION

The green line is same as above but over their full historical range.

Red histogram is the number of papers studied in each year



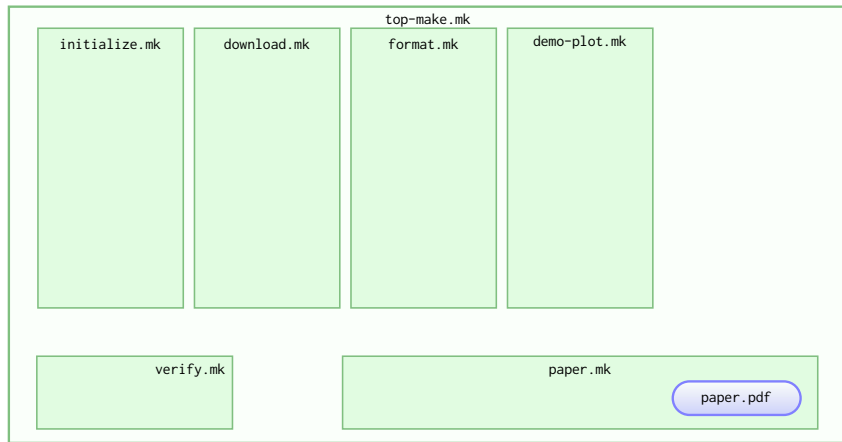
Makefiles (`.mk`) keep contextually separate parts of the project, all imported into `top-make.mk`



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.

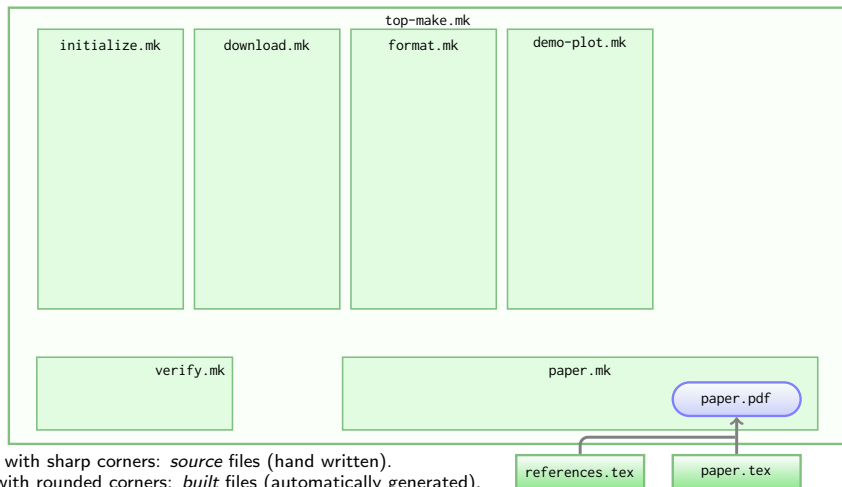
The ultimate purpose of the project is to produce a paper/report (in PDF).



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.

The narrative description, typography and references are in `paper.tex` & `references.tex`.

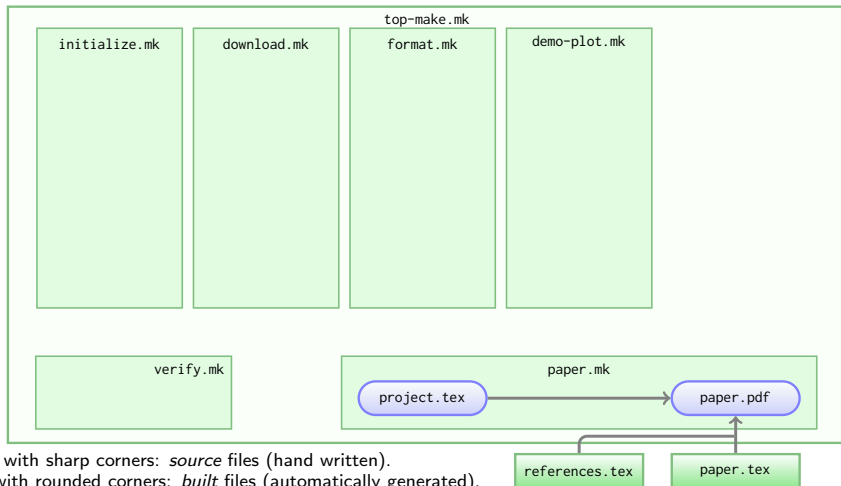


Green boxes with sharp corners: *source* files (hand written).

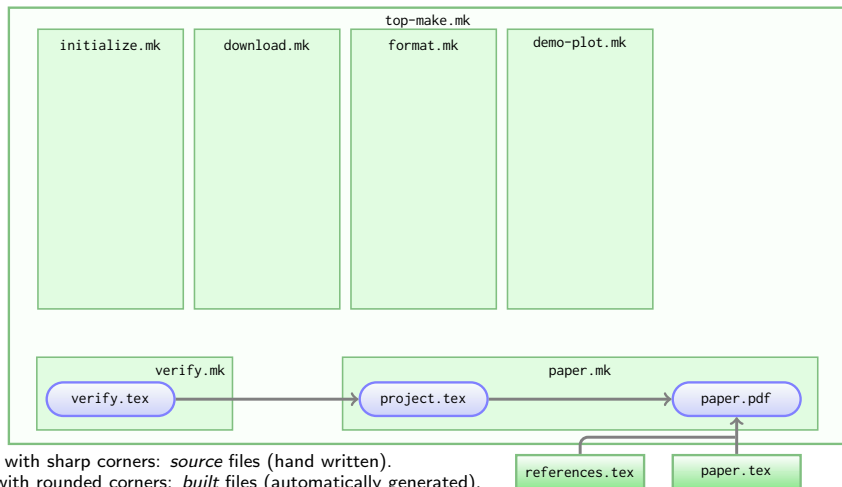
Blue boxes with rounded corners: *built* files (automatically generated),

built files are shown in the Makefile that contains their build instructions.

Analysis outputs (blended into the PDF as \LaTeX macros) come from `project.tex`.



But analysis outputs must first be *verified* (with checksums) before entering the report/paper.

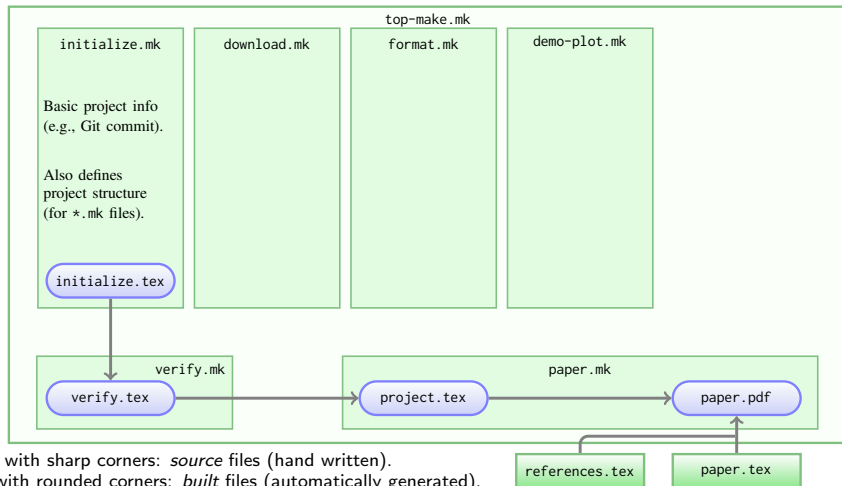


Green boxes with sharp corners: *source* files (hand written).

Blue boxes with rounded corners: *built* files (automatically generated),

built files are shown in the Makefile that contains their build instructions.

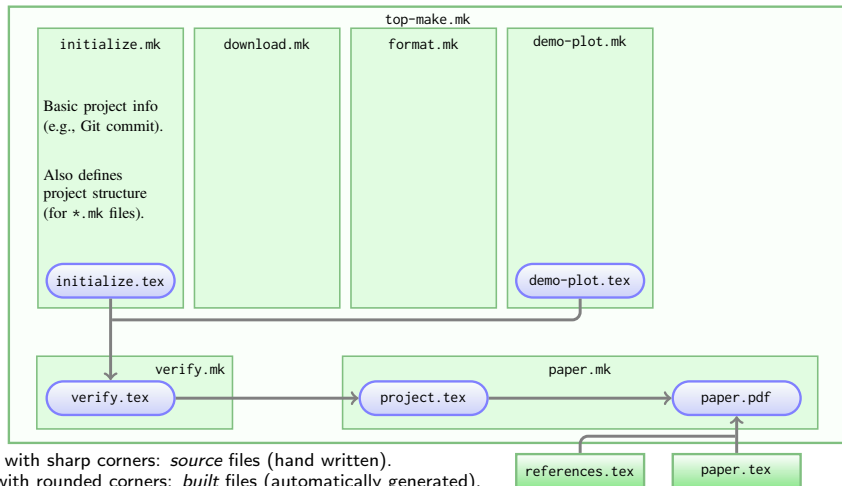
Basic project info comes from `initialize.tex`.



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.

The paper includes some information about the plot.

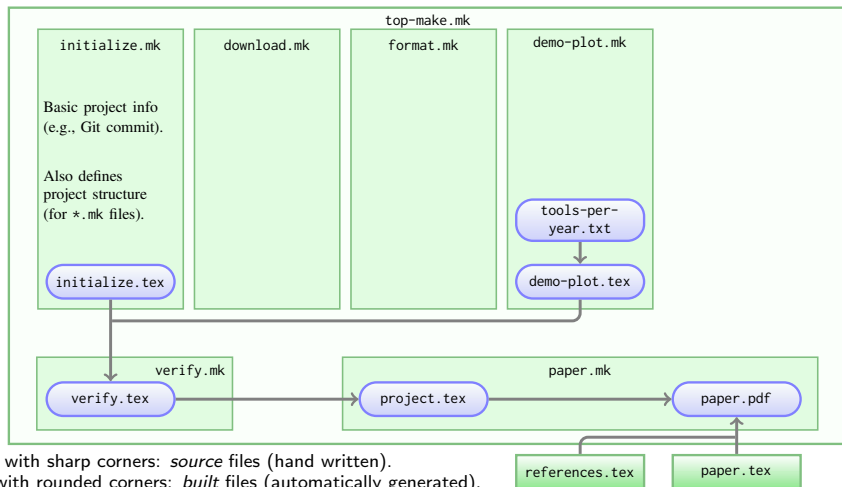


Green boxes with sharp corners: *source* files (hand written).

Blue boxes with rounded corners: *built* files (automatically generated),

built files are shown in the Makefile that contains their build instructions.

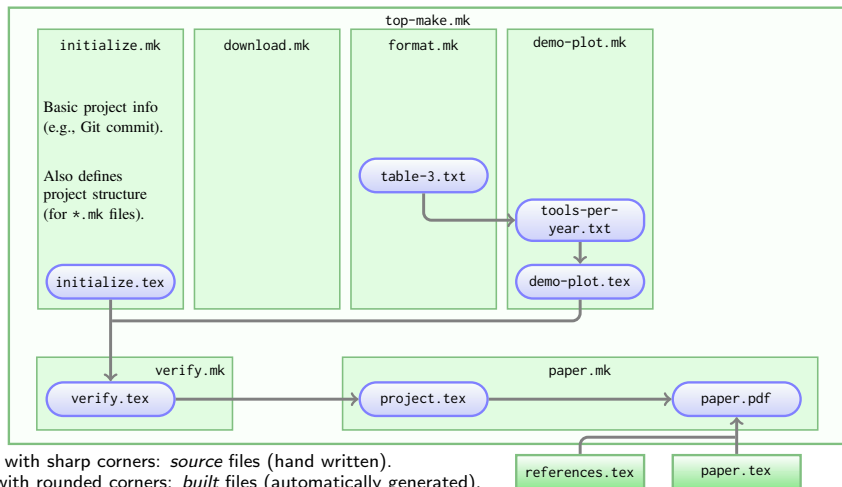
The final plotted data are calculated and stored in `tools-per-year.txt`.



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.

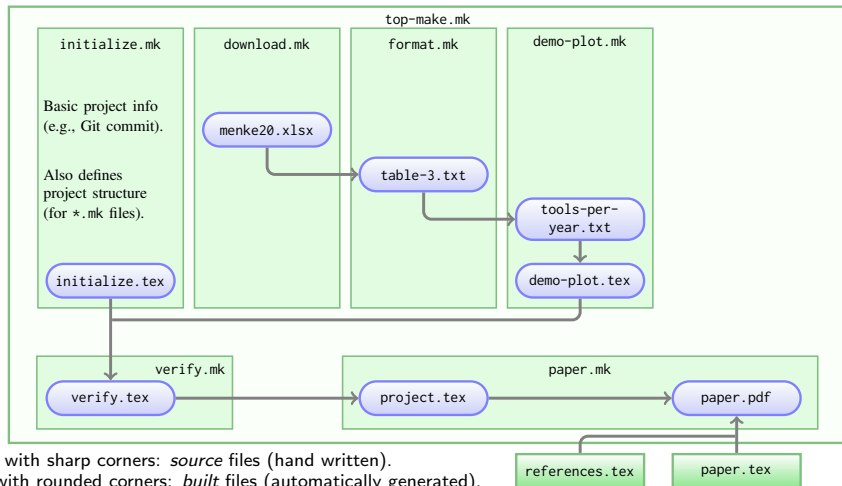
The plot's calculation is done on a formatted sub-set of the raw input data.



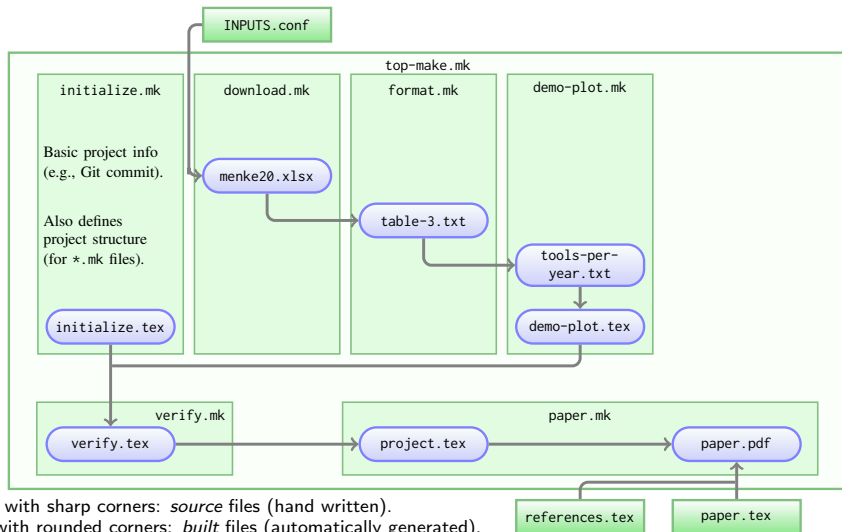
Green boxes with sharp corners: *source* files (hand written).

Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

The raw data that were downloaded are stored in XLSX format.



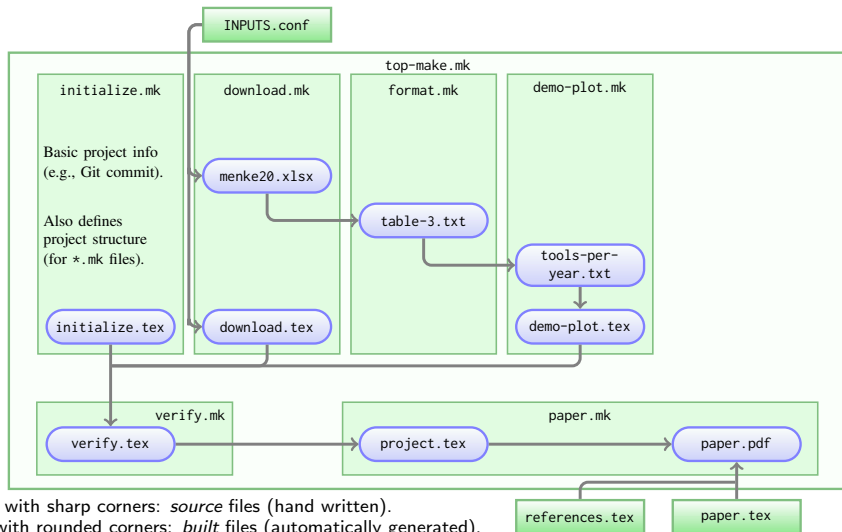
The download URL *and* a checksum to validate the raw inputs, are stored in `INPUTS.conf`.



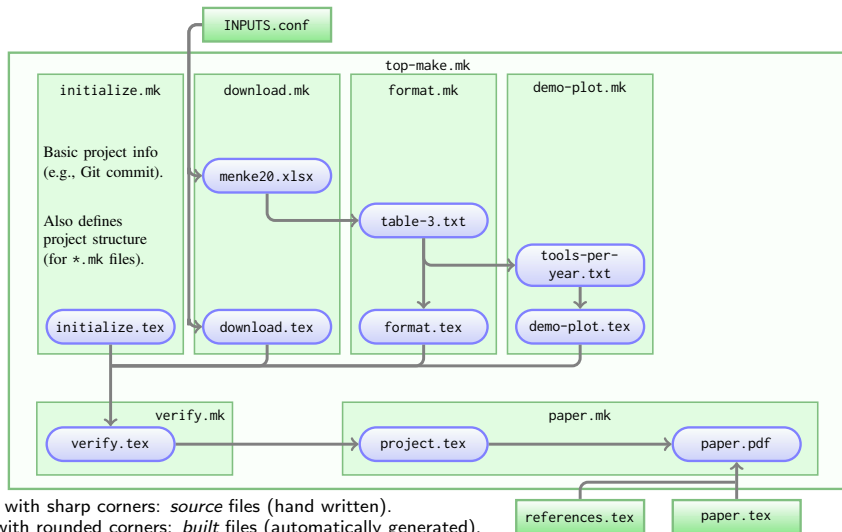
Green boxes with sharp corners: *source* files (hand written).

Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

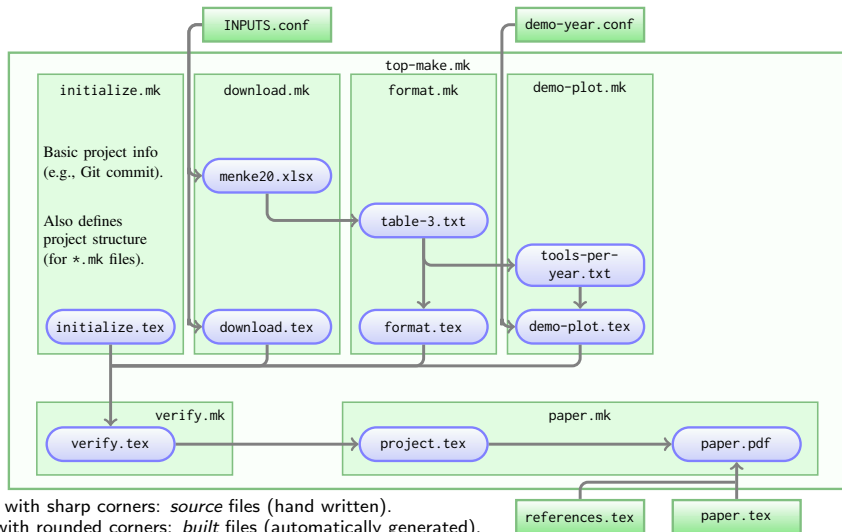
We also need to report the URL in the paper...



Some general info about the full dataset may also be reported.



We report the number of papers studied in a special year, desired year is stored in `.conf` file.

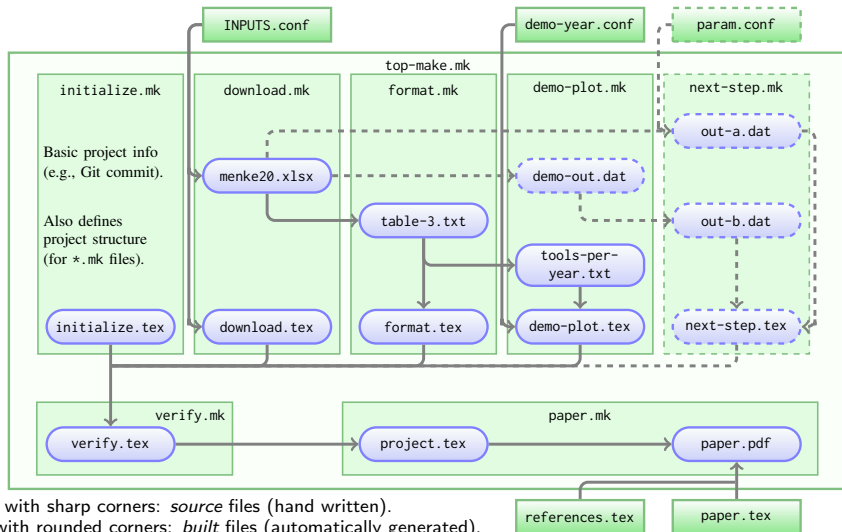


Green boxes with sharp corners: *source* files (hand written).

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)



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.

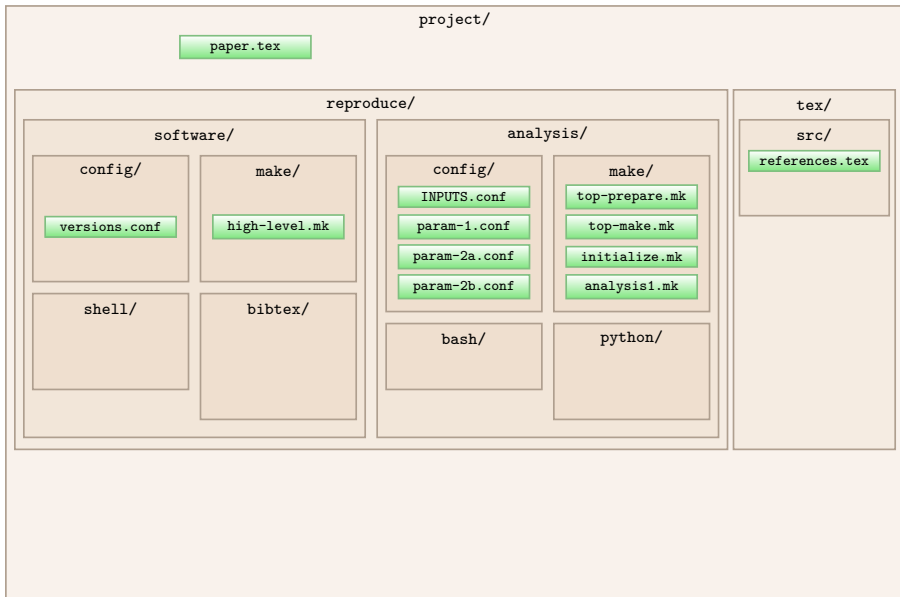
The whole project is a directed graph (codifying the data's lineage).

- ▶ Every **file** (source or built) is a **node** in the graph (connected to others).
(The links/connections/dependencies between the nodes, defined by the Makefiles: `*.mk`)
- ▶ There are two types of nodes/files:
 - ▶ **Source** nodes (`*.conf` and `paper.tex`) only have an **outward** link.
 - ▶ **Built** files always have **inward** *and* (except `paper.pdf`) **outward** link(s).
- ▶ All built files ultimately originate from a `*.conf` file,
... and ultimately conclude in `paper.pdf`.

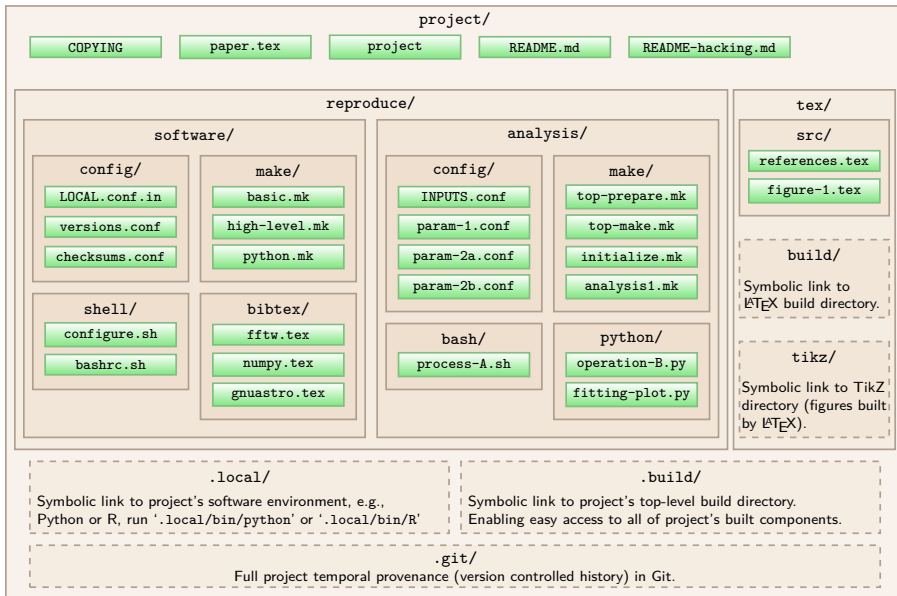
Benefits of using Make

- ▶ Make can **parallelize** the analysis:
Make knows which steps are independent and will run them at the same time.
- ▶ Make can **automatically detect a change** and will re-do *only* the affected steps.
(for example to change the multiple of sigma in a configuration file to see its effect)
- ▶ Easily **backtrace** any step (without needing to remember!).
(very useful to find problems/improvements)
- ▶ The above will speed up your work, and **encourage experimentation** on methods.
- ▶ Make is **available** on any system: many people are **already familiar** with it.
- ▶ And again: its **all in plain text!**
(doesn't take much space, easy to read, distribute, parse automatically, or archive)
- ▶ Recall that the project's **software installation** was also managed in Make.

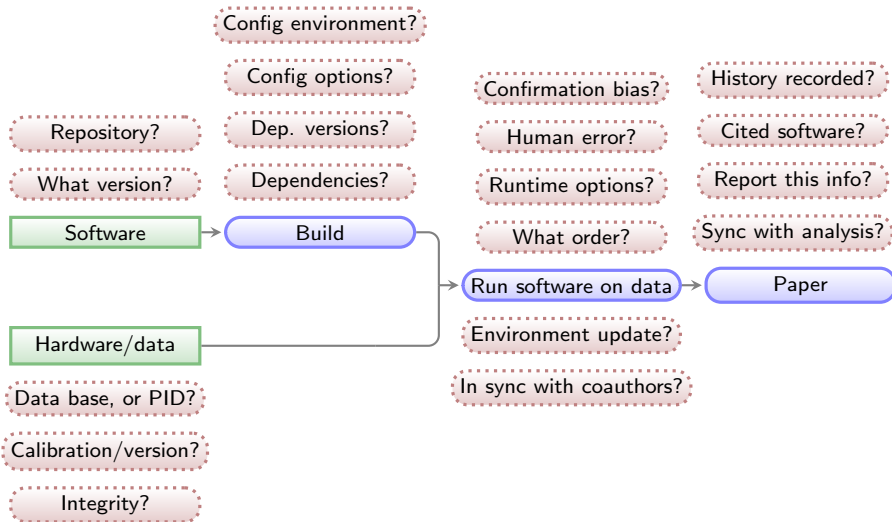
Files organized in directories by context (here are some of the files discussed before)



Files organized in directories by context (now with other project files and symbolic links)



All questions have an answer now (in **plain text**: human & computer readable/archivable).

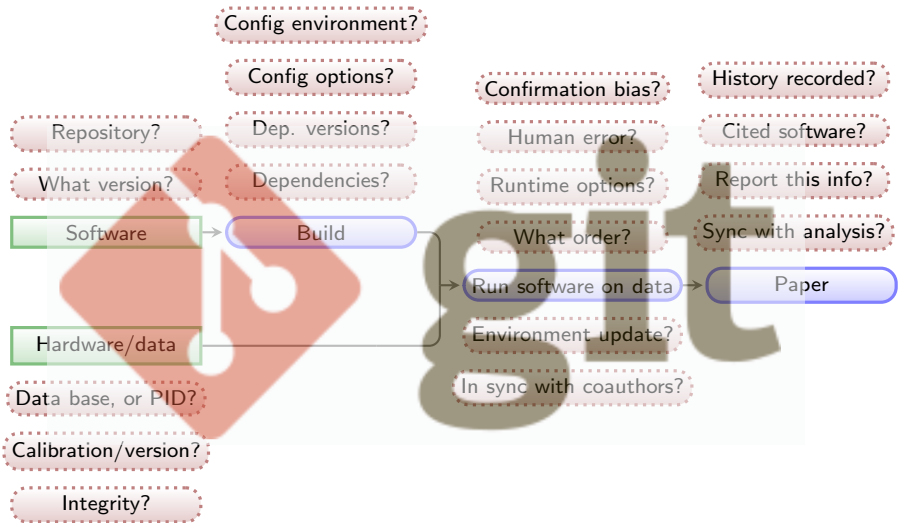


Green boxes with sharp corners: *source*/input components/files.

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

- ▶ The project (answers to questions above) will evolve.



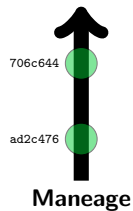
New projects branch from Maneage

- ▶ The project (answers to questions above) will evolve.



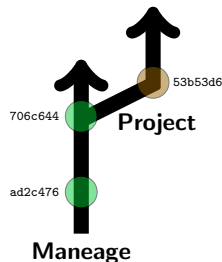
New projects branch from Maneage

- ▶ Each point of project's history is recorded with Git.

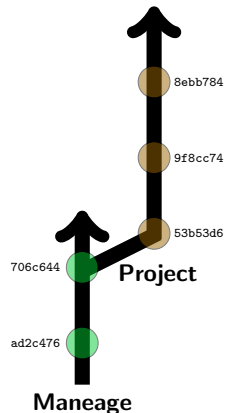


New projects branch from Maneage

- ▶ Each point of project's history is recorded with 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**.

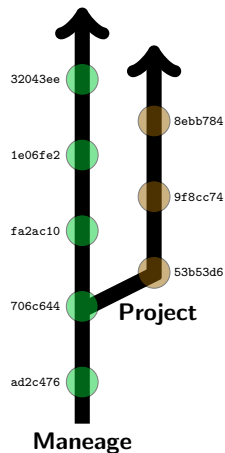


New projects branch from Maneage



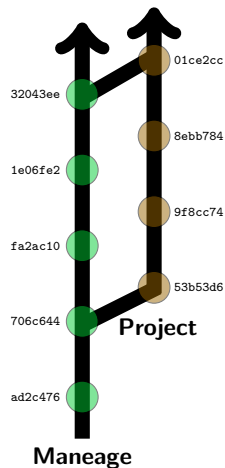
- ▶ Each point of project's history is recorded with Git.
- ▶ New project: a branch from the template.
Recall that **every commit** contains the following:
 - ▶ Instructions to download, verify and build **software**.
 - ▶ Instructions to download and verify input **data**.
 - ▶ Instructions to run software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

New projects branch from Maneage



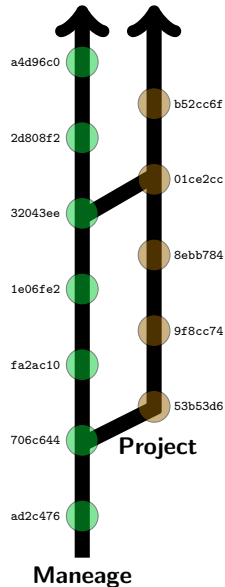
- ▶ Each point of project's history is recorded with Git.
- ▶ New project: a branch from the template.
Recall that **every commit** contains the following:
 - ▶ Instructions to download, verify and build **software**.
 - ▶ Instructions to download and verify input **data**.
 - ▶ Instructions to run software on data (do the **analysis**).
 - ▶ Narrative description of project's purpose/**context**.
- ▶ Research progresses in the project branch.
- ▶ Template will evolve (improved infrastructure).

New projects branch from Maneage



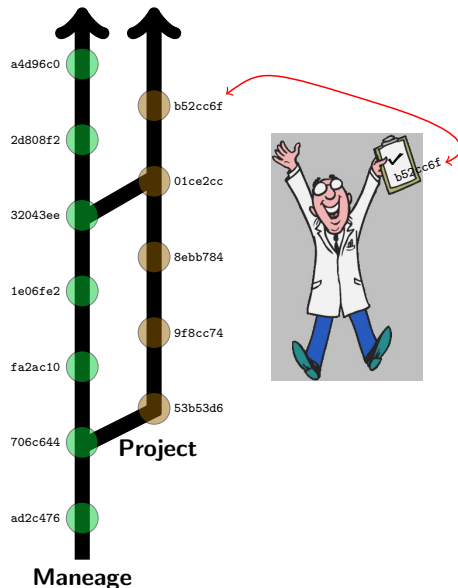
- ▶ Each point of project's history is recorded with Git.
- ▶ New project: a branch from the template.
Recall that **every commit** contains the following:
 - ▶ Instructions to download, verify and build **software**.
 - ▶ Instructions to download and verify input **data**.
 - ▶ Instructions to run software 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.

New projects branch from Maneage



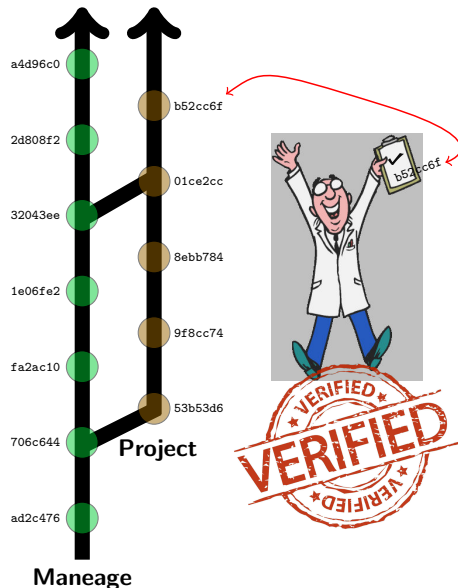
- ▶ Each point of project's history is recorded with Git.
- ▶ New project: a branch from the template.
Recall that **every commit** contains the following:
 - ▶ Instructions to download, verify and build **software**.
 - ▶ Instructions to download and verify input **data**.
 - ▶ Instructions to run software 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).

New projects branch from Maneage



- ▶ Each point of project's history is recorded with 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.

New projects branch from Maneage



- ▶ Each point of project's history is recorded with 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.

Two recent examples (publishing Git checksum in abstract)

arXiv:1909.11230v1 [astro-ph.IM] 24 Sep 2019

The Results of the Low-Surface-Brightness Universe
Proceedings IAU Symposium No. 355, 2019
D. Valls-Gabaud, I. Trujillo & S. Okamoto, eds.

© 2019 International Astronomical Union
DOI: 00.0000/X000000000000000X

Carving out the low surface brightness universe with NoiseChisel

Mohammad Akhlaghi^{1,2}

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

²Facultad de Física, Universidad de La Laguna, Avda. Astrofísico Fco. Sánchez 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 GNU Astronomy Utilities (Gnuastro). Over the last ten stable releases of Gnuastro, 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 segmentation 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 detect the outer wings of M51 down to S/N of 0.25, or 28.27 mag/arcsec² on a single-exposure SDSS image (r-band). Segment is also able to detect the localized HII regions as “clumps” much more successfully. Finally, to orchestrate a controlled analysis, the concept of a “reproducible paper” is discussed: this paper itself is exactly reproducible (snapshot v4.4-g8505c6f).

Keywords. galaxies: halos, galaxies: photometry, galaxies: 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 datasets noise and thus requires accurate detection methods. In Akhlaghi and Ichikawa (2015) (henceforth AI15) 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 GNU Astronomy Utilities (Gnuastro). It was the first professional astronomical software to be independently refereed 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 Rafelski et al. (2015) (henceforth R15), who used a combination of six SExtractor (Bertin and Arnouts 1996) runs with different configurations to avoid deblending problems, but still missed many sources with significant signal, see Figure 1. Borlaff et al. (2019), Miller et al. (2019), and Trujillo et al. (2019) used it for accurate flat field and Sky subtraction to create deeper co-added images in galaxy fields for optimal detection of the low surface brightness features. Calvi et al. (2019) used it to find Lyman- α emitters in spectra. For future studies, Laine et al.

[†] <https://www.gnu.org/s/gnuastro>
[‡] <https://www.gnu.org/prep/standards>

Monthly Notices

mnras.0000000.0000000

MNRAS **00**, 1–11 (2020)

Advance Access publication 2019 November 14

doi:10.1093/mnras/mtz111

The Sloan Digital Sky Survey extended point spread functions

Raúl Infante-Sainz^{1,2*}, Ignacio Trujillo^{0,1,2} and Javier Román^{0,1,2,3}

⁰Instituto de Astrofísica de Canarias, C/ Vía Láctea s/n, E-38205 La Laguna, Tenerife, Spain

¹Departamento de Astrofísica, Universidad de La Laguna, E-38205 La Laguna, Tenerife, Spain

²Instituto de Astrofísica de Andalucía (CSIC), Glorieta de la Astronomía, E-18008 Granada, Spain

Accepted 2019 October 30. Received 2019 October 29; in original form 2019 September 10

ABSTRACT

A robust and extended characterization of the point spread function (PSF) is crucial to extract the photometric information produced by deep imaging surveys. Here, we present the extended PSFs of the Sloan Digital Sky Survey (SDSS), one of the most productive astronomical surveys of all time. By stacking ~1000 images of individual stars with different brightness, we obtain the bidimensional SDSS PSFs extending over 9 arcmin in radius for all the SDSS filters (i.e. g , r , i , z). This new characterization of the SDSS PSFs is near a factor of 10 larger in extension than previous PSFs characterizations of the same survey. We found asymmetries in the shape of the PSFs caused by the drift scanning observing mode. The flux of the PSFs is larger along the drift scanning direction. Finally, we illustrate with an example how the PSF models can be used to remove the scattered light field produced by the brightest stars in the central region of the Coma cluster field. This particular example shows the huge importance of PSFs in the study of the low-surface brightness Universe, especially with the upcoming of ultradep surveys, such as the Large Synoptic Survey Telescope (LSST). Following a reproducible science philosophy, we make all the PSF models and the scripts used to do the analysis of this paper publicly available (snapshot v0.4.4-g4966ad0).

Key words: instrumentation: detectors – methods: data analysis – techniques: image processing – techniques: photometric – galaxies: halos.

1 INTRODUCTION

The point spread function (PSF) describes the response of an imaging system to the light produced by a point source. Real PSFs have complex structures as their shapes depend on the optical path that light takes as it travels through the atmosphere and multiple optical elements, mirrors, lenses, detectors, etc. For the vast majority of astronomical works, only a tiny portion of the PSF (i.e. essentially a few inner arcseconds; see e.g. Trujillo et al. 2004a, b) is characterized. In practice, however, the light of both point and extended sources are spread over the entire detector due to the effect of the PSF at large radii. Therefore, it is necessary to have a good understanding of its structure along the entire detector (typically extended over arcminutes or more).

Extended PSFs have become a vital tool to obtain precise photometric information in modern astronomical surveys. For instance, Stare, Harding & Mibow (2009) modelled the extended PSF and the internal reflections produced by the stars of the Hubble Space Telescope and showed that virtually all the pixels of the image are dominated by the scattered light by both stars and galaxies at 20.5 mag/arcsec² (i-band; Trujillo & Pflanz 2016)

also characterized and used the extended PSF of the 10.4 m Gran Telescopio Canarias (GTC) telescope to model and remove the scattered light in ultradep observations of the UGC 00100 galaxy. Even more troublesome for low-surface brightness studies is the finding (see e.g. Trujillo & Balon 2013; Sandin 2014, 2015) that the outer regions of astronomical objects are severely affected by their own scattered light produced by the convolution with the PSF. In order to correct this effect, Karabal et al. (2017) generated the PSF and models of the internal reflections from images of the Canada-France-Hawaii Telescope (CFHT) to deconvolve a sample of three galaxies and correct them from instrumental scattered light. More recently, Román, Trujillo & Montes (2019) characterized the PSFs of the Strips 82 survey and used them to model and correct the scattered light field produced by stars to study the optical properties of the Galactic cirr. All the above works have shown that having an extended PSF is crucial when accurate photometric and structure properties of astronomical objects at low-surface brightness levels are required.

One of the most commonly used surveys for measuring photometric properties of astronomical objects is the Sloan Sky Digital Survey (SDSS; York et al. 2000), covering 14 555 deg² on the sky (just over 35 per cent of the full sky) in five photometric bands (i.e. g , r , i , and z). Although SDSS is a relatively shallow survey compared

*E-mail: rinfante@gaia.es

Two recent examples (publishing Git checksum in abstract)

arXiv:1909.11230v1 [astro-ph.IM] 24 Sep 2019

The Roads of the Low-Surface-Brightness Universe
Proceedings IAU Symposium No. 355, 2019
D. Valls-Gabaud, J. Trujillo & S. Okamoto, eds.

© 2019 International Astronomical Union
DOI: 00.0000/X000000000000000X

Carving out the low surface brightness universe with NoiseChisel

Mohammad Akhlaghi^{1,2}

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

²Facultad de Física, Universidad de La Laguna, Avda. Astrofísico Fco. Sánchez 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 GNU Astronomy Utilities (Gnuastro). Over the last ten stable releases of Gnuastro, 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 segmentation 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 detect the outer wings of M51 down to S/N of 0.25, or 28.27 mag/arcsec² on a single-exposure SDSS image (r-band). Segment is also able to detect the localized HII regions as “clumps”, which are non-spheroidal. Finally, to orchestrate a controlled analysis, the concept of a “reproducible analysis” is proposed: this paper itself is exactly reproducible (snapshot v4.4.0-g8505c6f).

Keywords. galaxies: halos, galaxies: photometry, galaxies: 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 datasets noise and thus requires accurate detection methods. In Akhlaghi and Ichikawa (2015) (henceforth AI15) 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 GNU Astronomy Utilities (Gnuastro). It was the first professional astronomical software to be independently refereed 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 Rafelski et al. (2015) (henceforth R15), who used a combination of six SExtractor (Bertin and Arnouts 1996) runs with different configurations to avoid deblending problems, but still missed many sources with significant signal, see Figure 1. Borlaff et al. (2019), Miller et al. (2019), and Trujillo et al. (2019) used it for accurate flat field and Sky subtraction to create deeper co-added images in galaxy fields for optimal detection of the low surface brightness features. Calvi et al. (2019) used it to find Lyman- α emitters in spectra. For future studies, Laine et al.

[†] <https://www.gnu.org/s/gnuastro>
[‡] <https://www.gnu.org/prep/standards>

Monthly Notices

mnras.000.0000-00000000

MNRAS **00**, 000–000 (2020)

Advance Access publication 2019 November 14

doi:10.1093/mnras/mtz111

The Sloan Digital Sky Survey extended point spread functions

Raúl Infante-Sainz^{1,2*}, Ignacio Trujillo^{0,1,2} and Javier Román^{0,1,2,3}

⁰Instituto de Astrofísica de Canarias, C/ Vía Láctea s/n, E-38205 La Laguna, Tenerife, Spain

¹Departamento de Astrofísica, Universidad de La Laguna, E-38205 La Laguna, Tenerife, Spain

²Instituto de Astrofísica de Andalucía (CSIC), Glorieta de la Astronomía, E-18008 Granada, Spain

Accepted 2019 October 30. Received 2019 October 29; in original form 2019 September 10

ABSTRACT

A robust and extended characterization of the point spread function (PSF) is crucial to extract the photometric information produced by deep imaging surveys. Here, we present the extended PSFs of the Sloan Digital Sky Survey (SDSS), one of the most productive astronomical surveys of all time. By stacking ~1000 images of individual stars with different brightness, we obtain the bidimensional SDSS PSFs extending over 9 arcmin in radius for all the SDSS filters (i.e. g , r , i , z). This new characterization of the SDSS PSFs is near a factor of 10 larger in extension than previous PSFs characterizations of the same survey. We found asymmetries in the shape of the PSFs caused by the drift scanning observing mode. The flux of the PSFs is larger along the drift scanning direction. Finally, we illustrate with an example how the PSF models can be used to remove the scattered light field produced by the brightest stars in the central region of the Coma cluster field. This particular example shows the huge importance of PSFs in the study of the low-surface brightness Universe, especially with the upcoming of ultra-deep surveys, such as the Large Synoptic Survey Telescope (LSST). Following a reproducible science philosophy, we make all the PSF models and the scripts used to do the analysis of this paper publicly available (snapshot v0.4.0-gd966ad0).

Key words: instrumentation: detectors – methods: data analysis – techniques: image processing – techniques: photometric – galaxies: halos.

1 INTRODUCTION

The point spread function (PSF) describes the response of an imaging system to the light produced by a point source. Real PSFs have complex structures as their shapes depend on the optical path that light takes as it travels through the atmosphere and multiple optical elements, mirrors, lenses, detectors, etc. For the vast majority of astronomical works, only a tiny portion of the PSF (i.e. essentially a few inner arcseconds; see e.g. Trujillo et al. 2004a, b) is characterized. In practice, however, the light of both point and extended sources are spread over the entire detector due to the effect of the PSF at large radii. Therefore, it is necessary to have a good understanding of its structure along the entire detector (typically extending over arcminutes or more).

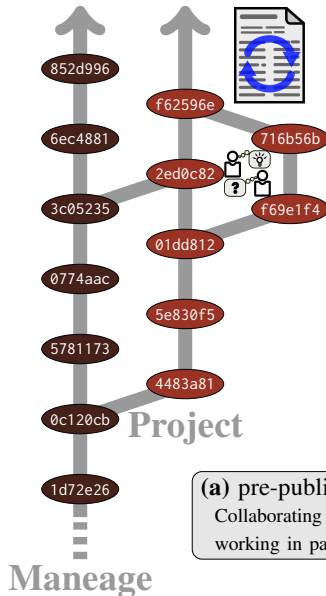
Extended PSFs have become a vital tool to obtain precise photometric information in modern astronomical surveys. For instance, Stare, Harding & Mibow (2009) modelled the extended PSF and the internal reflections produced by the stars of the Burrell Schmidt telescope and showed that virtually all the pixels of the image are dominated by the scattered light by both stars and galaxies at 20.5 mag/arcsec² (r-band; Trujillo & Pflanz 2016)

also characterized and used the extended PSF of the 10.4 m Gran Telescopio Canarias (GTC) telescope to model and remove the scattered light in ultradeep observations of the UGC 00100 galaxy. Even more troublesome for low-surface brightness studies is the finding (see e.g. Trujillo & Balok 2013; Sandin 2014, 2015) that the outer regions of astronomical objects are severely affected by their own scattered light produced by the convolution with the PSF. In order to correct this effect, Karabal et al. (2017) generated the PSF models and the internal reflections from images of the Canada-France-Hawaii Telescope (CFHT) to deconvolve a sample of three galaxies and correct them from instrumental scattered light. More recently, Román, Trujillo & Montes (2019) characterized the PSFs of the Stripe 82 survey and used them to model and correct the scattered light field produced by stars to study the optical properties of the Galactic rim. All the above works have shown that having an extended PSF is crucial when accurate photometric and structure properties of astronomical objects at low-surface brightness levels are required.

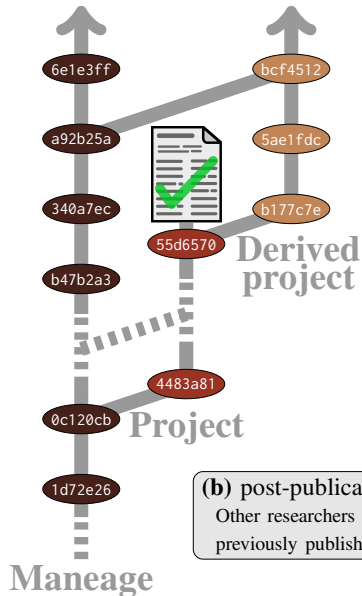
One of the most commonly used surveys for measuring photometric properties of astronomical objects is the Sloan Sky Digital Survey (SDSS; York et al. 2000, covering 14 555 deg² on the sky (just over 35 per cent of the full sky) in five photometric bands (i.e. g , r , i , and z). Although SDSS is a relatively shallow survey compared

*E-mail: rinfante@gaia.es

Any Git-based workflow is possible.



(a) pre-publication:
Collaborating on a project while working in parallel, then merging.



(b) post-publication:
Other researchers building upon previously published work.

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 ~ 100 kilo-bytes).

Publication of the project

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

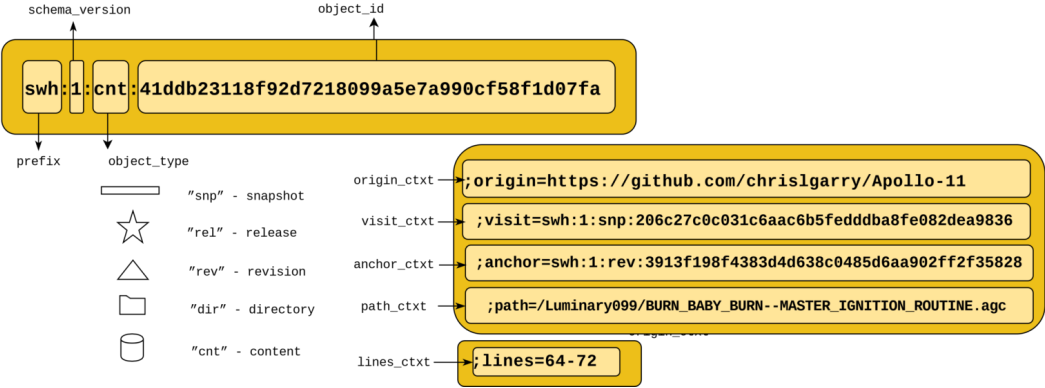
- ▶ Makefiles.
- ▶ \LaTeX source files.
- ▶ Configuration files for software used in analysis.
- ▶ Scripts/programming files (e.g., Python, Shell, AWK, C).

The **volume** of the project's source will thus be **negligible** compared to a single figure in a paper (usually ~ 100 kilo-bytes).

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

- ▶ **arXiv**: uploaded with the \LaTeX source to always stay with the paper (for example [arXiv:1505.01664](https://arxiv.org/abs/1505.01664) or [arXiv:2006.03018](https://arxiv.org/abs/2006.03018)).
- ▶ **Zenodo**: Along with all the input datasets (many Gigabytes) and software (for example [zenodo.3872247](https://zenodo.org/record/3872247)) and given a unique DOI.
 - ▶ ... and put links to data in paper! See ending of caption of Figure 1 in the [Maneage paper](#).
- ▶ **Software Heritage**: to archive the full version-controlled history of the project. (for example [swh:1:dir:33fea87068c1612daf011f161b97787b9a0df39fk](https://sw.here.org/record/1:dir:33fea87068c1612daf011f161b97787b9a0df39fk))
 - ▶ ... and put links to exact parts of the code! See caption of Listing 1 in the [Maneage paper](#).

Software Heritage IDs (SWHID); persistent identifier for source code (or any text!)



For more details, see SoftwareHeritage FAQ (at <https://www.softwareheritage.org/faq>)

Project source and its execution

Programs [here: Scientific projects] must be written for **people to read...**
...and only *incidentally* for machines to *execute*.

Harold Abelson, Structure and Interpretation of Computer Programs

General outline of using this system (for example arXiv:1909.11230)

```
$ git clone http://gitlab.com/makhlaghi/iau-symposium-355    # Import the project.
```


General outline of using this system (for example arXiv:1909.11230)

```
$ git clone http://gitlab.com/makhlaghi/iau-symposium-355    # Import the project.
```

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


General outline of using this system (for example arXiv:1909.11230)

```
$ git clone http://gitlab.com/makhlaghi/iau-symposium-355    # 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.*

Summary:

Maneage is introduced as 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**.
- ▶ Only parts affected by a modification are re-done.
- ▶ Using LaTeX macros, paper's figures, tables and numbers will be **Automatically updated**.
- ▶ The whole project is under **version control** (Git) **encouraging tests/experimentation**.
- ▶ The **Git commit hash** of the project source, is **printed** in the paper and **on output** data products.
- ▶ These slides are available at <https://maneage.org/pdf/slides-intro.pdf>.

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>

Feel free to contact me: mohammad@akhlaghi.org