

# Maneage: a proof-of-concept for rigorous reproducible research paper criteria

(Akhlaghi+2021, CiSE 23, 82 DOI:10.1109/MCSE.2021.3072860, arXiv:2006.03018)

Boud Roukema

Institute of Astronomy, Nicolaus Copernicus University

Ten Years of Guix

16 September 2022 @Université de Sorbonne

Slides' main authors: Akhlaghi + Roukema; pdf built from git commit 348ad50;  
this pdf: <https://cosmo.torun.pl/~boud/Roukema20220916TYG.pdf>



## Reproducibility crisis in scientific research: 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](#)).

### Schrödinger'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](#)).



and in biology, other sciences, economics

### 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 & Li (2015) were able to **replicate less than half** of 67 papers in prestigious journals, *with help* from the authors. 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: 70% of researchers couldn't reproduce another scientist's results; half couldn't reproduce their own. [nature.com/articles/533452a](http://nature.com/articles/533452a)



## Replicability (hardware/statistical)

---

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



(C) CC BY-SA 2006, R. Feiler

## Reproducibility (software/deterministic)

---

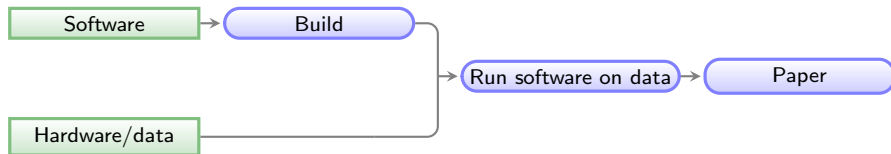
- ▶ Involves data **analysis**, or simulations.
- ▶ Starts **after** data is collected/digitised.
- ▶ Example:  $2 + 2 = 4$  (i.e., sum of datasets).
- ▶ **DISCUSSED HERE.**



(C) 2008 J. Zawinski



## General outline of a project (after data collection)



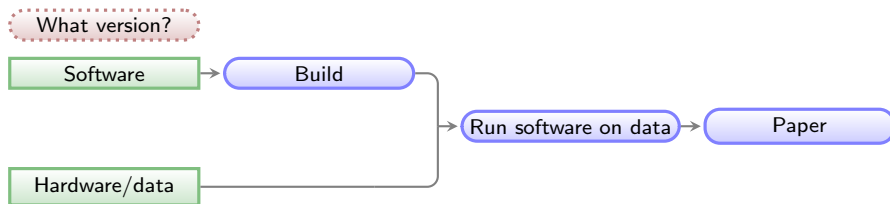
Green boxes with sharp corners: *source*/input components/files.

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.



# 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

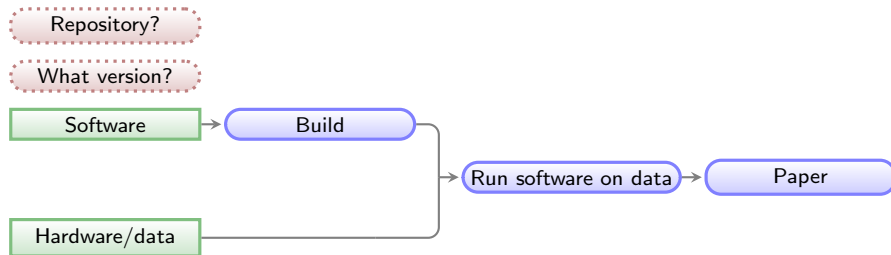
## GNU Astronomy Utilities (Gnuastro)

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)



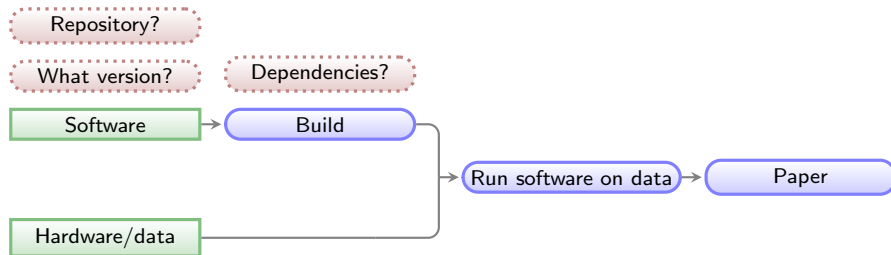
Green boxes with sharp corners: *source*/input components/files.

Blue boxes with rounded corners: *built* components.

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



## General outline of a project (after data collection)



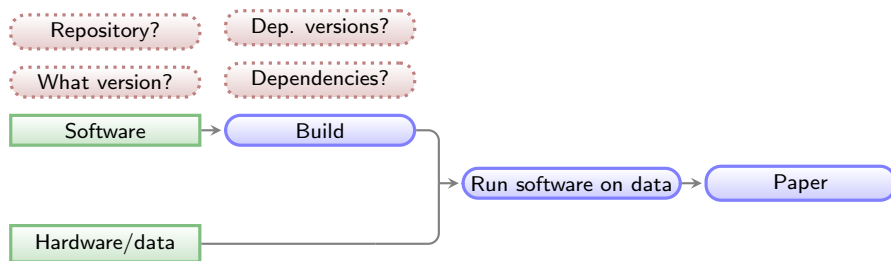
Green boxes with sharp corners: *source*/input components/files.

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)



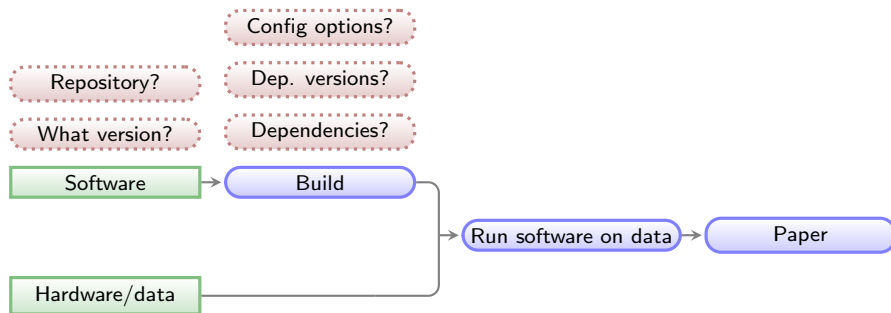
Green boxes with sharp corners: *source*/input components/files.

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)



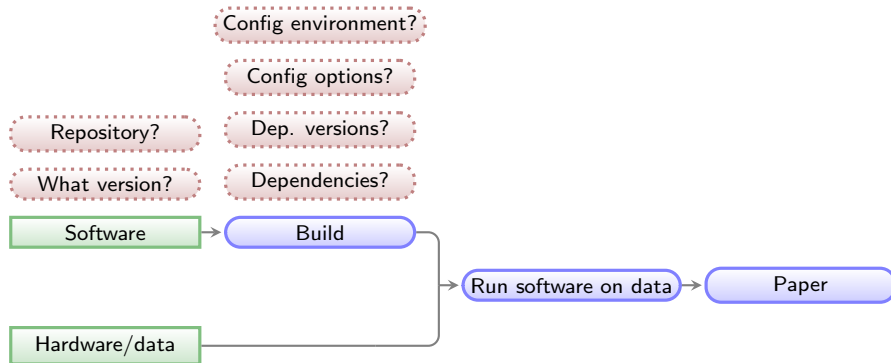
Green boxes with sharp corners: *source*/input components/files.

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.

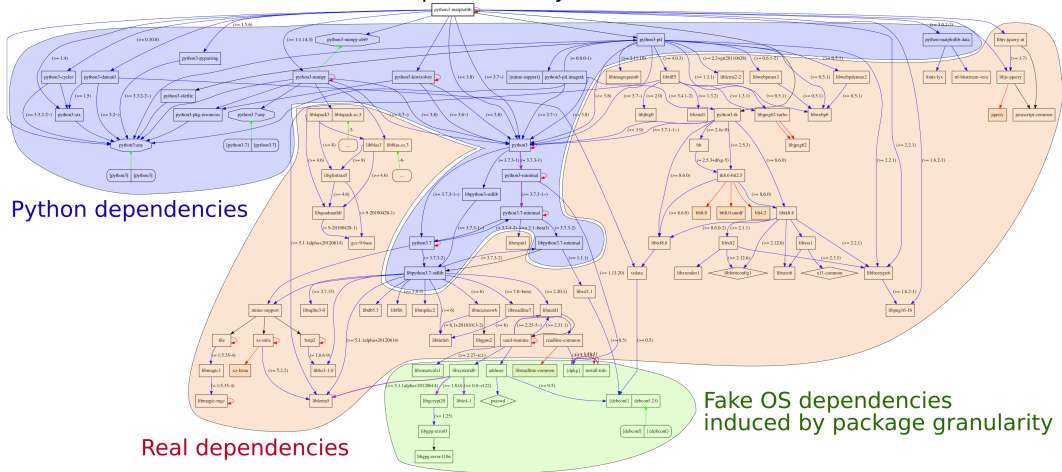
Blue boxes with rounded corners: *built* components.

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



# Example: Matplotlib (a Python visualization library) build dependencies

## Matplotlib library



Fake OS dependencies induced by package granularity

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:

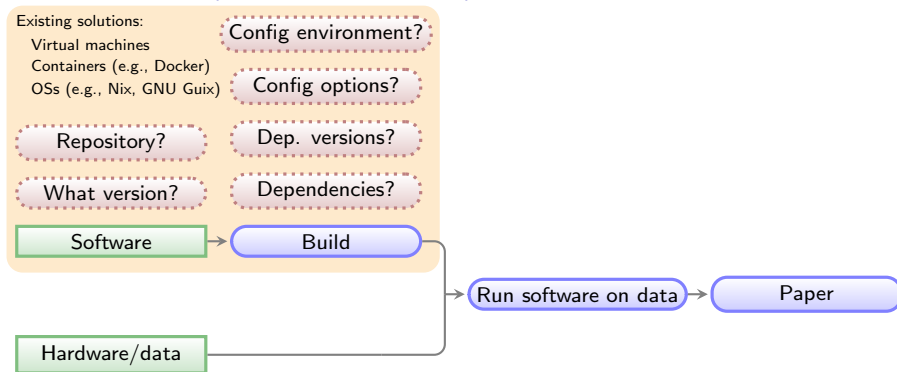
☐ Compact mode ☐ Co-maintainers

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)



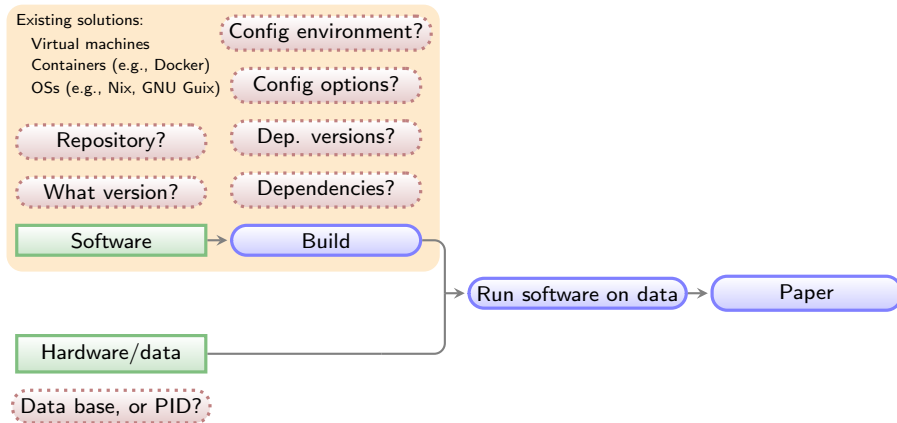
Green boxes with sharp corners: *source*/input components/files.

Blue boxes with rounded corners: *built* components.

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



## General outline of a project (after data collection)



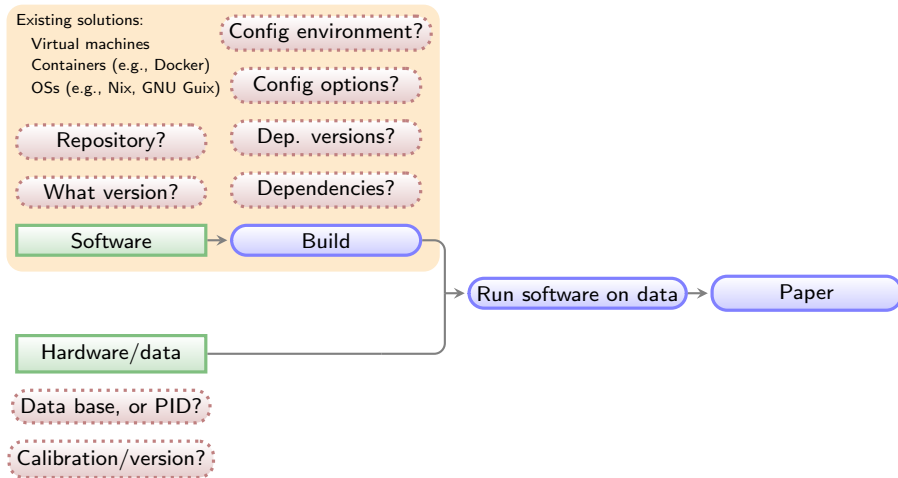
Green boxes with sharp corners: *source*/input components/files.

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)



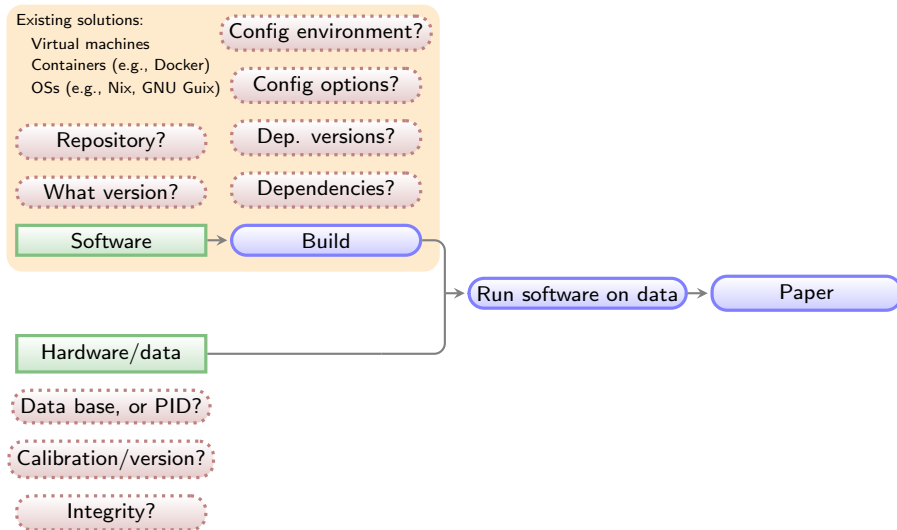
Green boxes with sharp corners: *source*/input components/files.

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)



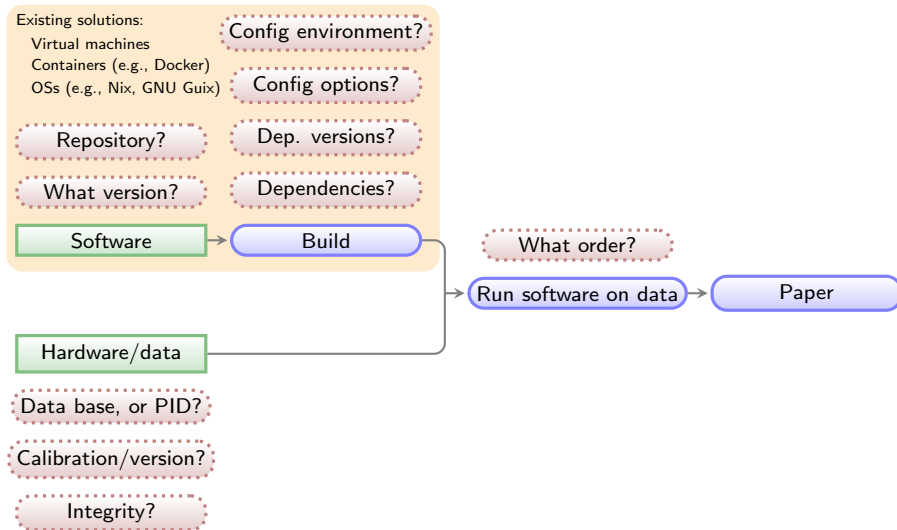
Green boxes with sharp corners: *source*/input components/files.

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)



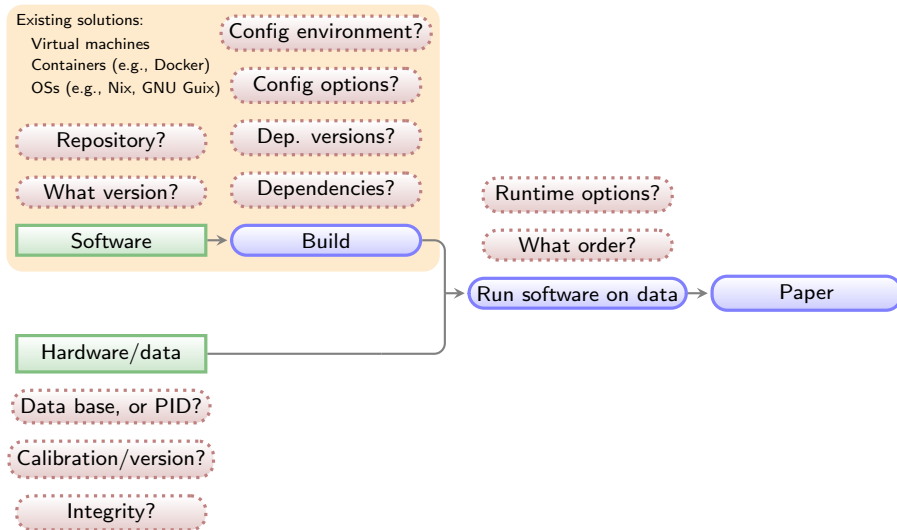
Green boxes with sharp corners: *source*/input components/files.

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)



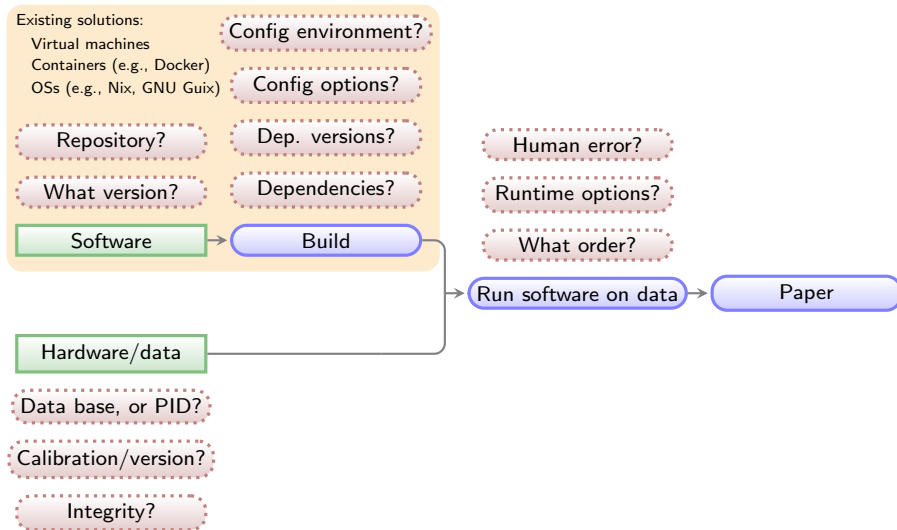
Green boxes with sharp corners: *source*/input components/files.

Blue boxes with rounded corners: *built* components.

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



## General outline of a project (after data collection)



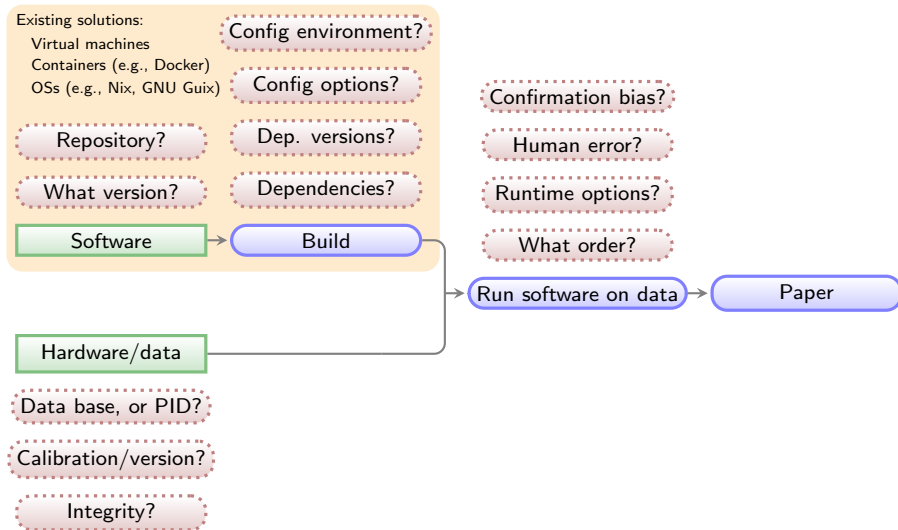
Green boxes with sharp corners: *source*/input components/files.

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)



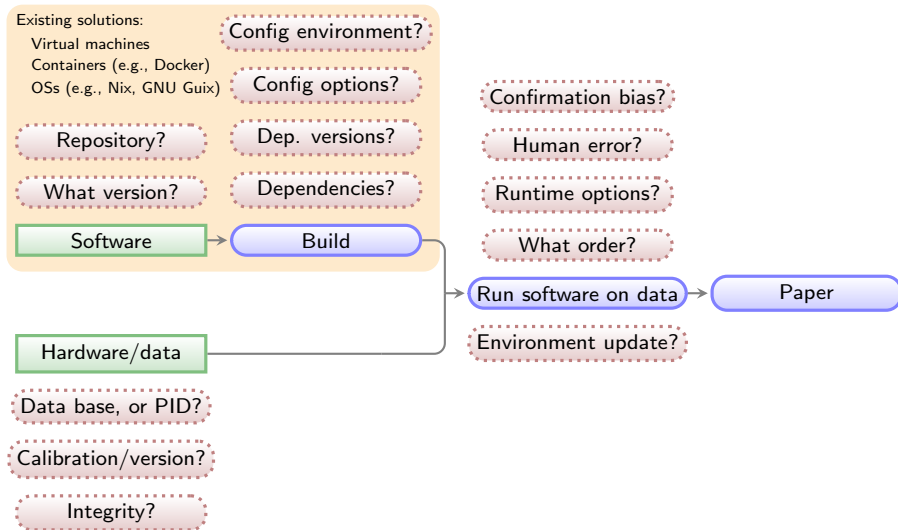
Green boxes with sharp corners: *source*/input components/files.

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)



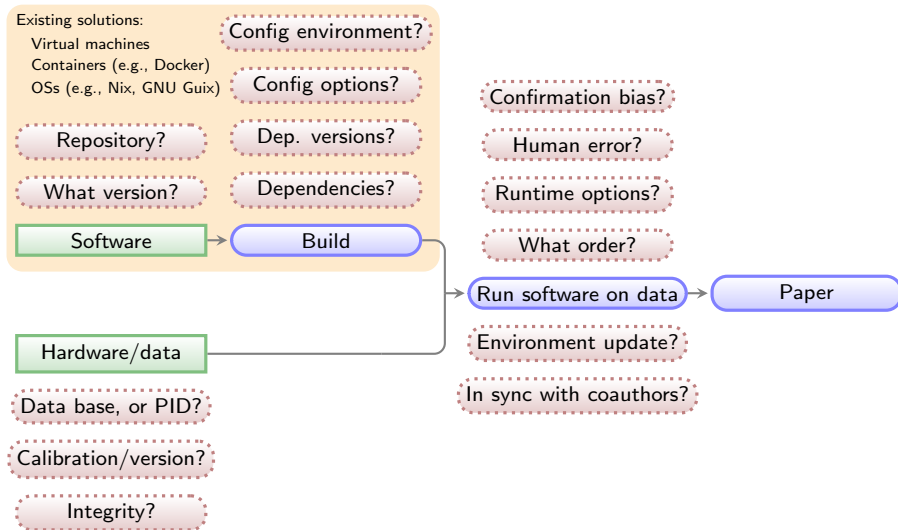
Green boxes with sharp corners: *source*/input components/files.

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)



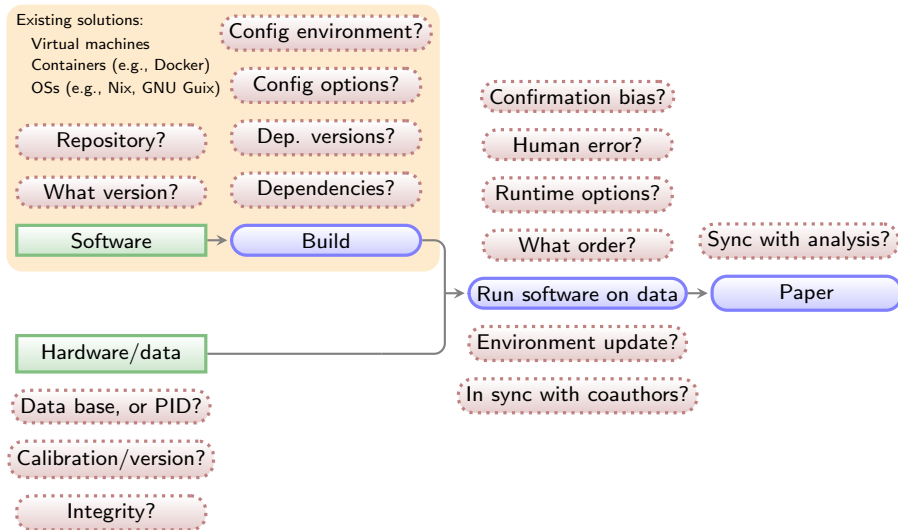
Green boxes with sharp corners: *source*/input components/files.

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)



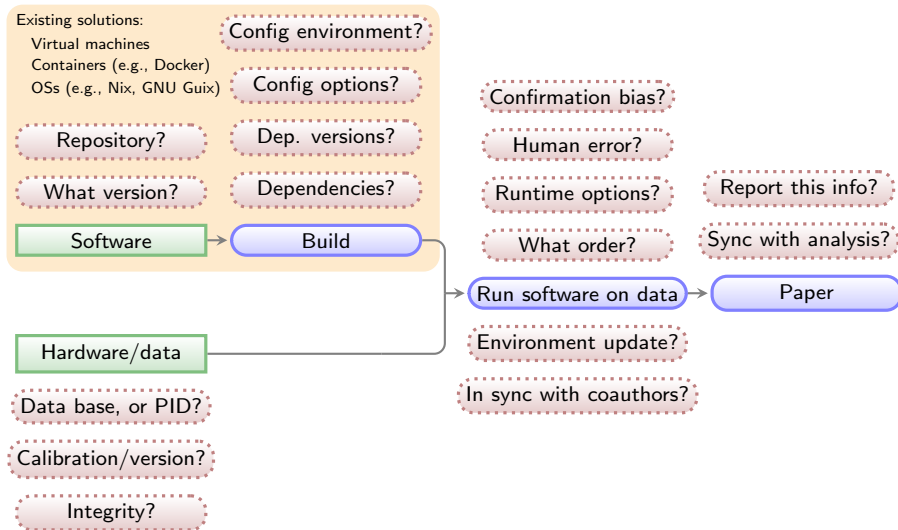
Green boxes with sharp corners: *source*/input components/files.

Blue boxes with rounded corners: *built* components.

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



## General outline of a project (after data collection)



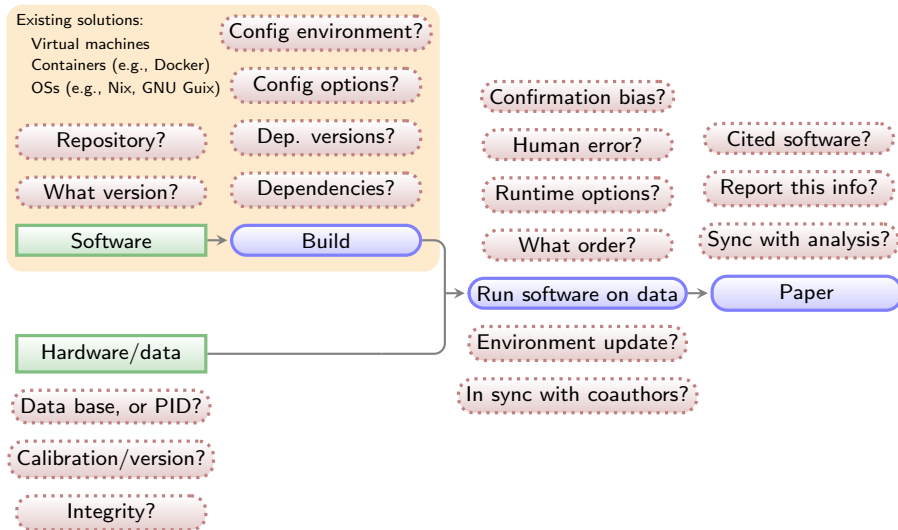
Green boxes with sharp corners: *source*/input components/files.

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.

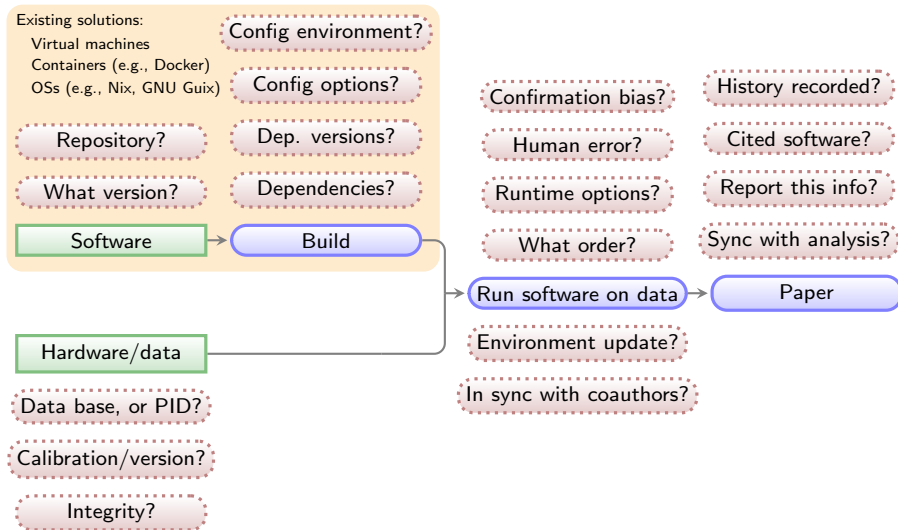


Di Cosmo & Pellegrini (2019) Encouraging a wider usage of software derived from research

**“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.



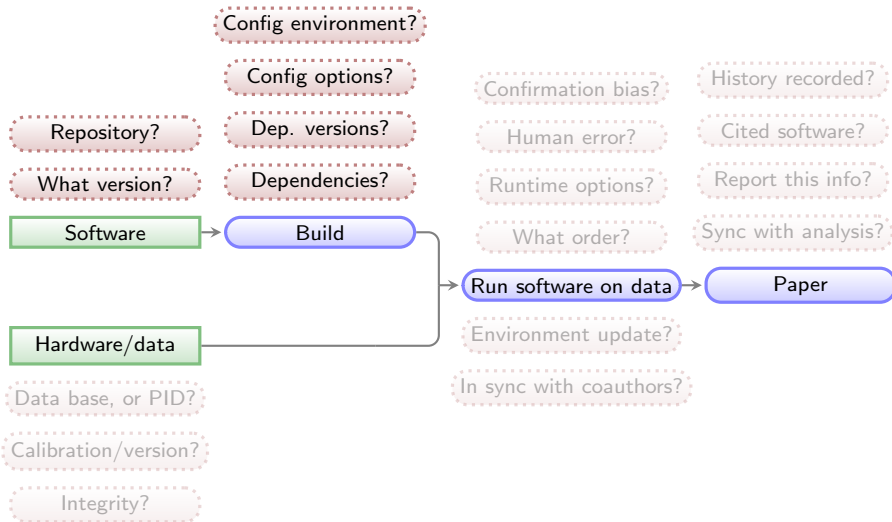
Buckheit & Donoho (1996) Lecture Notes in Statistics (vol 103, DOI:10.1007/978-1-4612-2544-7\_5)

“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.”



## 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.



# Criteria for a solution

## ▶ 1. Completeness (self-contained-ness):

- ▶ (1) **Only dependency** should be **POSIX** tools (discards Conda or Jupyter which need Python).
- ▶ (2) **Plain text**: Project's source should be in **plain-text** (binary formats need special software)
- ▶ (3) **No impact on host** OS libraries, programs, env variables.
- ▶ (4) Must **not require root** permissions (discards tools like Docker or Nix/Guix).
- ▶ (5) Builds its own **controlled software + env variables**.
- ▶ (6) Should be usable **without an internet** connection.
- ▶ (7) Contains full instructions for **inputs, software building, and outputs**: analysis + narrative + graphical output (e.g. pdf or html)
- ▶ (8) Should be **non-interactive** or runnable in batch (user interaction is an incompleteness).

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

## ▶ 3. Minimal complexity: Occam'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**.

## ▶ 4. Scalability: an implementation should easily scale to arbitrarily large, complex projects.



## Criteria for a solution

- ▶ **5. Verifiable inputs and outputs:** Inputs and Outputs must be **automatically verified**.
- ▶ **6. Recorded history:** Exploratory research involves modifying methods, reducing over-ambitious goals, serendipity; hypothesis testing should have a fixed, predefined method — **did** the authors modify the method? **how? what? which? when?**
- ▶ **7. Including the narrative that is associated with the analysis:** a workflow alone lacks **motivations, interpretations**.
- ▶ **8. 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).



EDITORS: Lorena A. Barba, lbarba@gsa.es  
Lorena A. Barba, lbarba@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/AI Khawair, Muscat, Oman  
David Valls-Gabaud, Paris Observatory, Paris 75014, France  
Roberto Bana-Galla , Universidad Internacional de La Rioja, Logroño 26006, Spain

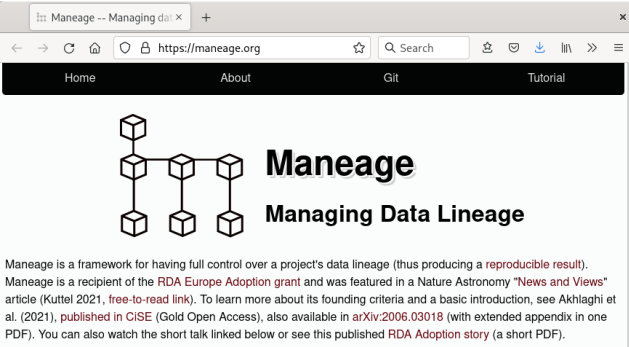
Analysis pipelines commonly use high-level technologies that are popular when created, but are unlikely to be readable, executable, or sustainable in the long term. A set of criteria is introduced to address this problem: completeness (no execution requirement beyond a minimal Unix-like operating system, 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://doi.ieeecomputersociety.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](https://github.com/maneage/maneage), which is also archived in Software Heritage: [swh1.cdn.mozilla.community/urn:uuid:33feab87068c1612da071f161b977b9a0d39f](https://swh1.cdn.mozilla.community/urn:uuid:33feab87068c1612da071f161b977b9a0d39f). 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.<sup>1,2</sup> 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.

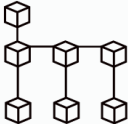


The screenshot shows the Maneage website in a web browser. The browser's address bar displays the URL <https://maneage.org>. The website has a dark blue header with navigation links: Home, About, Git, and Tutorial. Below the header is a large graphic featuring a network diagram of interconnected cubes 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>

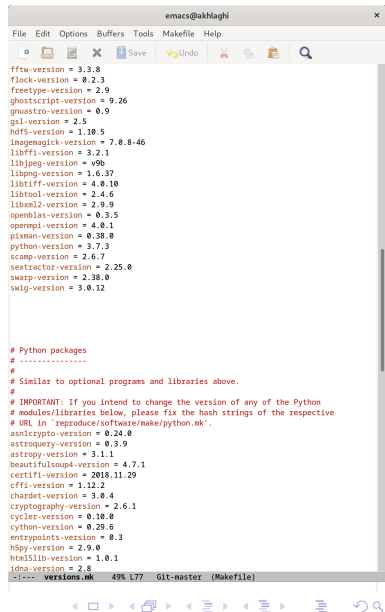


# Predefined/exact software tools

## 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

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
cyclur-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-openssl, 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 := $(lbidir)/$(lib64dir)
export LD_LIBRARY_PATH := $(lbidir)/$(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)
```



Example: Matplotlib (a Python visualization library) build dependencies

## Matplotlib library

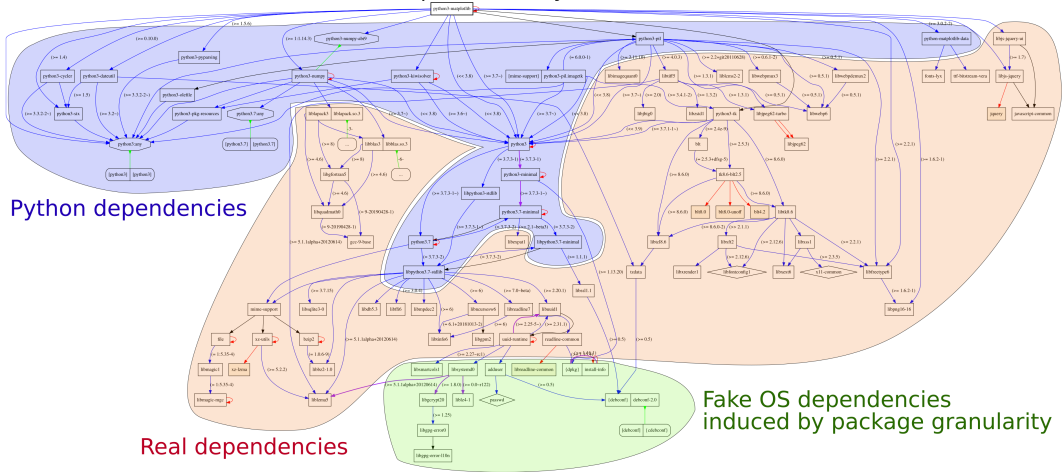


Fig. 1. Transitive dependencies of the software environment required by a simple “import matplotlib” command in the Python 3 interpreter.



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 (0x00007ffffdcbf7000)
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<sup>A</sup>T<sub>E</sub>X**, etc.
- ▶ No need for **root**/administrator **permissions** (on servers or super computers).
- ▶ Whole system is built **automatically** on any Unix-like operating system (~ 2–3 hours; Sep 2022).
- ▶ Dependencies of different projects will **not conflict**.
- ▶ Everything in **plain text** (human & computer readable/archivable).



## YOUR NAME: \_\_\_\_\_



## YOUTH NAME: 000000



## 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-cfc, 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, datetim 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, timesec 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.



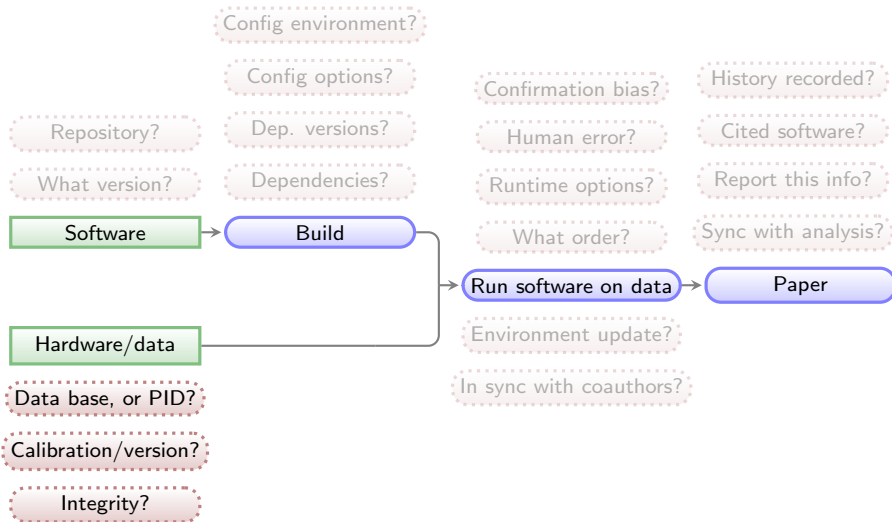
## Appendix A: Software acknowledgement

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 1.0.0, 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-cf6c, 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, datatime 2.66, ee 1.0, environ 0.3, etoolbox 2.5f, etexrsrc 1.6a, fancyhdr 3.10, fmincsint 3.05, fontaxes 1.0d, four-misc 5.5b, tp 2.1d, helvetica 2016-06-24, ltnemo 4.41, logreq 1.0, newtx 1.554, pdf 3.1.2, pgplots 1.16, preprint 2011, setspace 6.7a, stowik 2.0, xcolorbox 4.20, xes 3.14159265, xesgxy 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

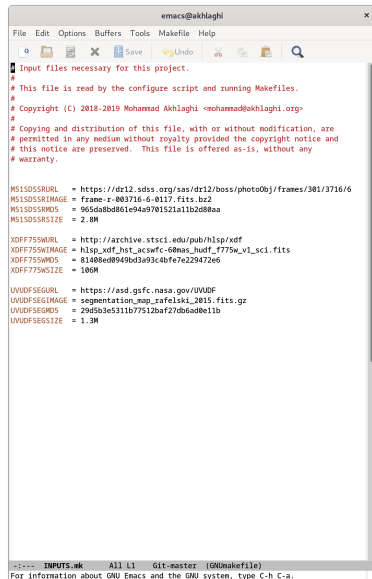
Stored information about each input file:

- ▶ **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.

W51S05SRURL = https://dr12.sdss.org/sas/dr12/boos/photoObj/frames/301/3716/6
W51S05SRIMAGE = frame-r-003716-6-0117.fits.bz2
W51S05SRMDS = 965da8bd861e94a9701521a11b2d88aa
W51S05SRSIZE = 2.8M

XDF775WURL = http://archive.stsci.edu/pub/hlsp/xdff
XDF775WIMGE = 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.

MS1S05SRURL = https://dr12.sdss.org/sas/dr12/boos/photoObj/frames/301/3716/6
MS1S05SRIMAGE = frame-r-003716-6-0117.fits.bz2
MS1S05SRMDS = 965da8bd861e94a9701521a11b2d88aa
MS1S05SRSIZE = 2.8M

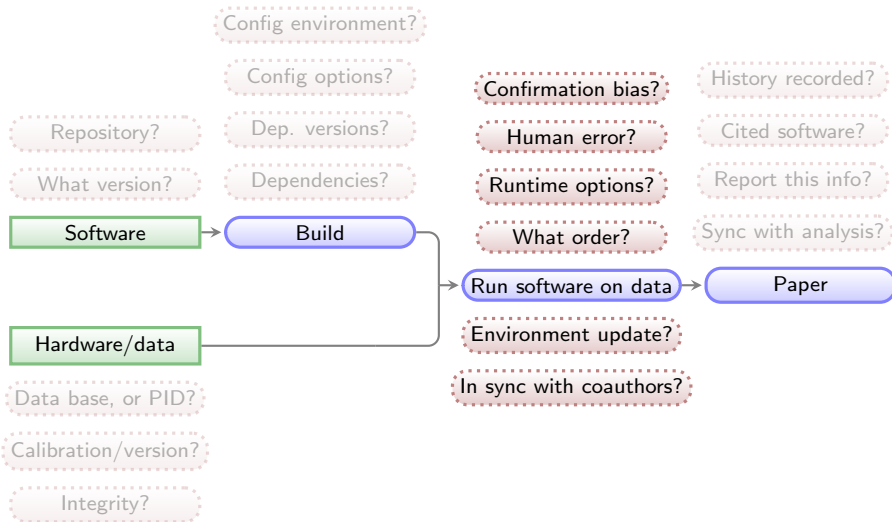
XDF775WURL = http://archive.stsci.edu/pub/hlsp/xdff
XDF775WIMAGE = hlsp_xdff_hst_acswfc-60mas_hudf_f775w_v1_sc1.fits
XDF775WMDS = 81408ed0949bd3a93c4bfe7e229472e6
XDF775WSIZE = 106M

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.
```



## 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)):

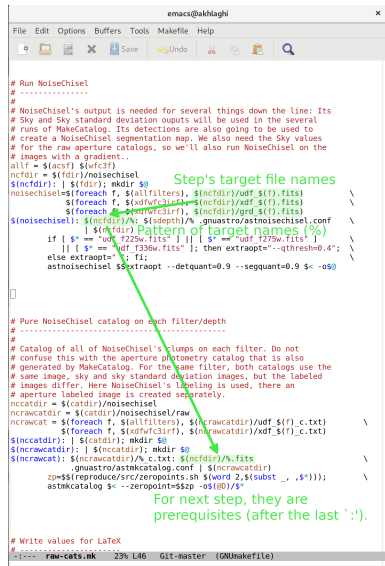
- ▶ Unlike a script which always starts from the top, a Makefile **starts from the end** and steps that don't change will be left untouched (not remade).
- ▶ A single *rule* can **manage any number of files**.
- ▶ Make can identify independent steps internally and do them in **parallel**.
- ▶ Make was **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](#)).



# 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" ] || [ $* == "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
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$(@D)/$*
```

Step's target file names

Pattern of target names (%)

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

# Write values for LaTeX

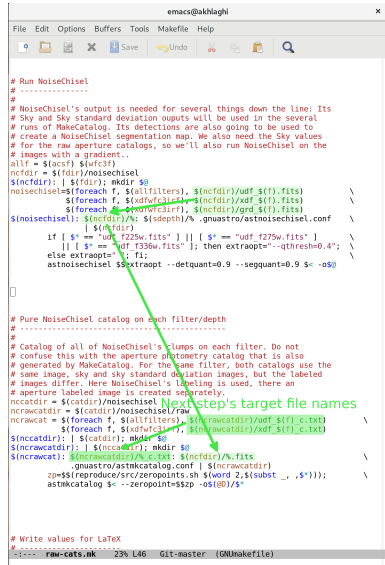
--- raw-cats.mk 23% L46 Git-master (GNUmakefile)



# 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](#)).



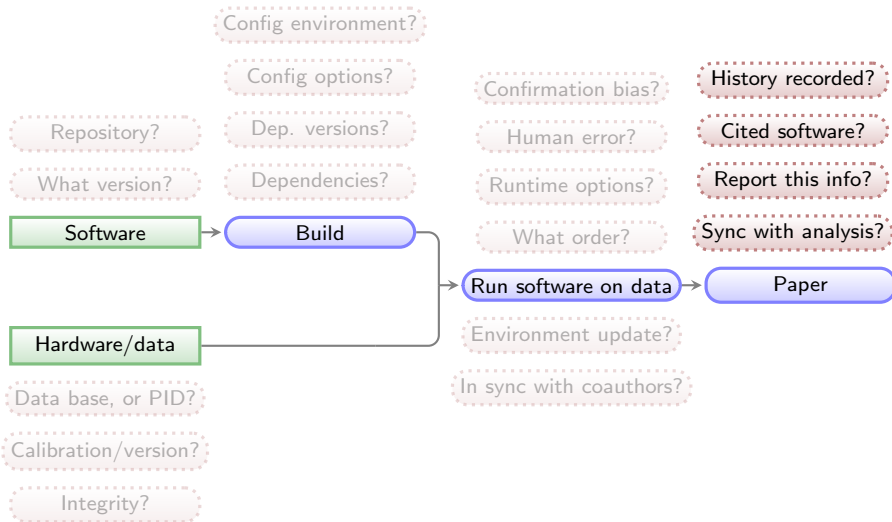
```
# Run NoiseChisel
# -----
#
# NoiseChisel's output is needed for several things down the line: Its
# Sky and Sky standard deviation outputs will be used in the several
# runs of MakeCatalog. Its detections are also going to be used to
# create a NoiseChisel segmentation map. We also need the Sky 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, $(xdfwfc3irf), $(ncfdir)/xdf $(f).fits) \
$(foreach f, $(xdfwfc3irf), $(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, $(xdfwfc3irf), $(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<sup>A</sup>T<sub>E</sub>X macros**. They are thus **updated automatically** on any change.

Shown here is a portion of the NoiseChisel paper and its L<sup>A</sup>T<sub>E</sub>X 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`<sub>T</sub> 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`.<sup>□</sup>

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<sup>A</sup>T<sub>E</sub>X macros**. They are thus **updated automatically** on any change.

Shown here is a portion of the NoiseChisel paper and its L<sup>A</sup>T<sub>E</sub>X 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`<sub>T</sub> 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$`, `\sensitivitycdettfmax$`, and `\fourdettfmax$`.<sup>□</sup>

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,

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

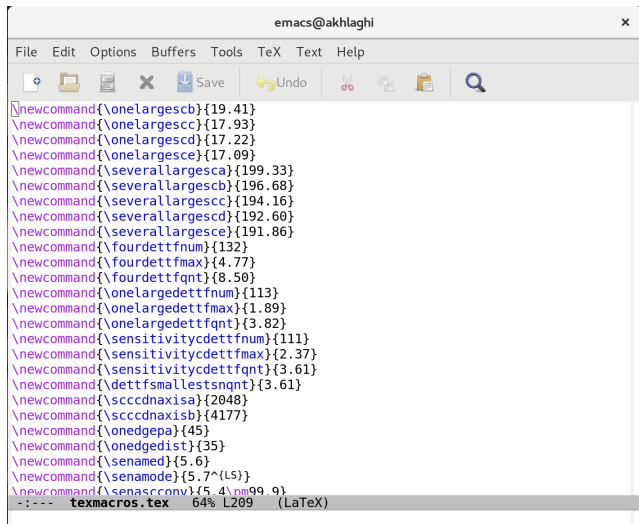
See Section 3.3 for the modifications required when the input image is not in units of counts or has already been Sky subtracted. The distribution of  $S/N_T$  from the objects in  $R_s$  for the three examples in Figure 7 can be seen in column 5 (top) of that figure. Image processing effects, mainly 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-



Analysis step results/values concatenated into a single file.

All  $\text{\LaTeX}$  macros come from a **single file**.



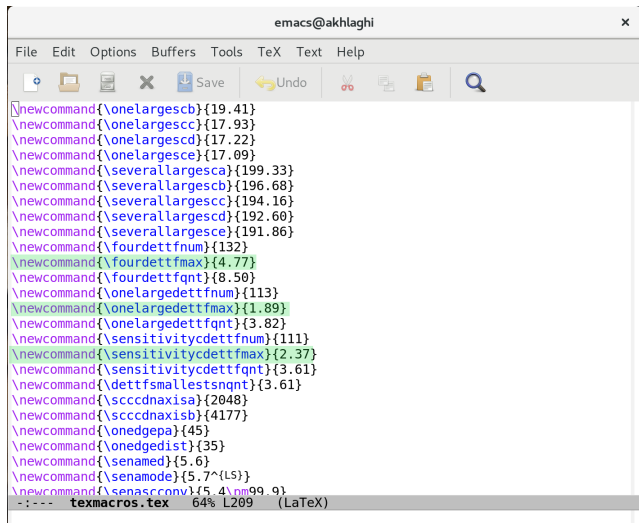
The screenshot shows an Emacs window titled "emacs@akhlaghi". The menu bar includes File, Edit, Options, Buffers, Tools, TeX, Text, and Help. The toolbar contains icons for opening a file, saving, undo, redo, and search. The main text area displays a list of LaTeX macro definitions, each starting with `\newcommand` followed by a macro name and a value in curly braces. The macros are: `\onelargescb` (19.41), `\onelargesc` (17.93), `\onelargescd` (17.22), `\onelargesc` (17.09), `\severallargesc` (199.33), `\severallargescb` (196.68), `\severallargesc` (194.16), `\severallargescd` (192.60), `\severallargesc` (191.86), `\fourdettfnum` (132), `\fourdettfmax` (4.77), `\fourdettfqnt` (8.50), `\onelargedettfnum` (113), `\onelargedettfmax` (1.89), `\onelargedettfqnt` (3.82), `\sensitivitycdettfnum` (111), `\sensitivitycdettfmax` (2.37), `\sensitivitycdettfqnt` (3.61), `\dettfsmallestsnqnt` (3.61), `\scccdnaxisa` (2048), `\scccdnaxisb` (4177), `\onedgepa` (45), `\onedgedist` (35), `\senamed` (5.6), `\senamode` (5.7<sup>LS</sup>), and `\senascconv` (5.4<sup>nm</sup>99.9). The status bar at the bottom shows "-:-- texmacros.tex 64% L209 (LaTeX)".

```
\newcommand{\onelargescb}{19.41}
\newcommand{\onelargesc}{17.93}
\newcommand{\onelargescd}{17.22}
\newcommand{\onelargesc}{17.09}
\newcommand{\severallargesc}{199.33}
\newcommand{\severallargescb}{196.68}
\newcommand{\severallargesc}{194.16}
\newcommand{\severallargescd}{192.60}
\newcommand{\severallargesc}{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}{5.7LS}
\newcommand{\senascconv}{5.4nm99.9}
-:-- texmacros.tex 64% L209 (LaTeX)
```



Analysis step results/values concatenated into a single file.

All  $\text{\LaTeX}$  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, representing various parameters. 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 $\text{\LaTeX}$ macros

The analysis scripts write/update the  $\text{\LaTeX}$  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 $\text{\LaTeX}$ macros

The analysis scripts write/update the  $\text{\LaTeX}$  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

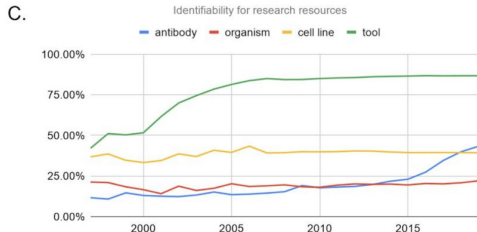
  # Find the smallest S/N quantile:
  sqnt=$(echo " " | awk '{if('$dettfqnt'<'$sqnt') print '$dettfqnt'}}')
}
for base in 4 onelarge sensitivity3
do dettfhist $texdir/dettf/ $base; done
addtexmacro dettfsmallestsqnt $sqnt
```



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

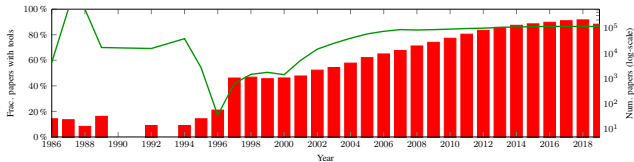
The Green plot shows the fraction of papers mentioning software tools from 1997 to 2019.



### OUR enhanced REPLICATION

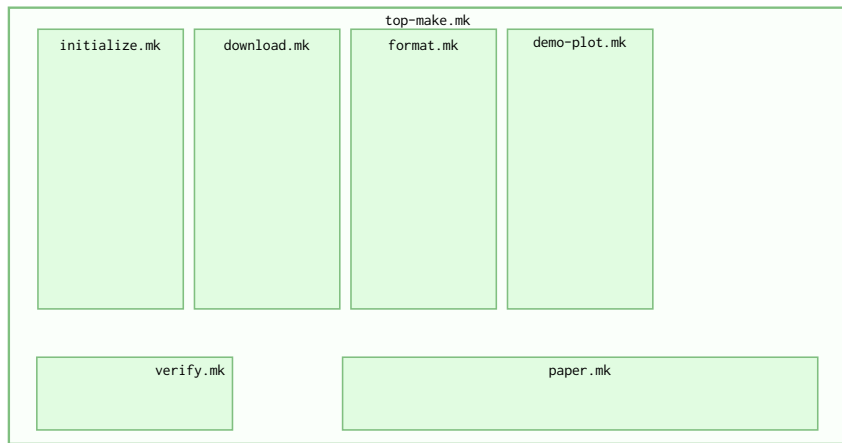
The green line is same as above but over their full historical range.

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`



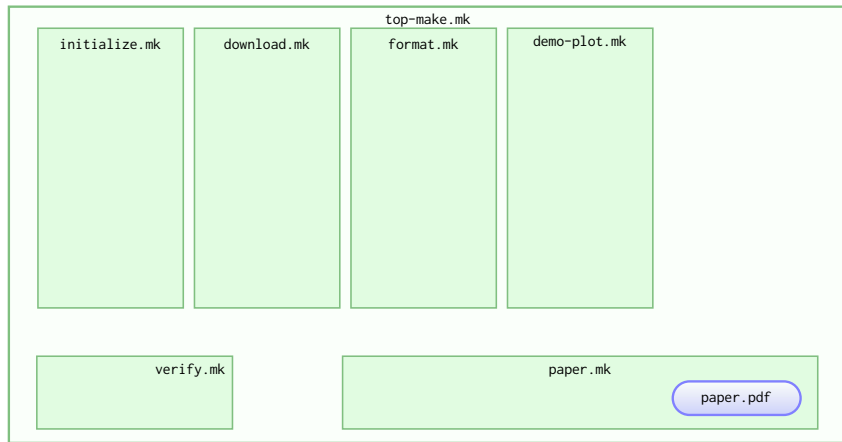
Green boxes with sharp corners: *source* files (hand written).

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).

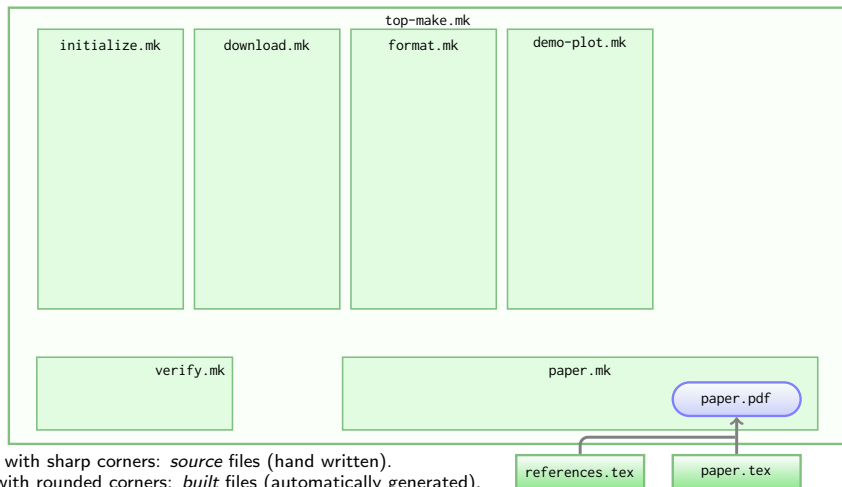


Green boxes with sharp corners: *source* files (hand written).

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`.



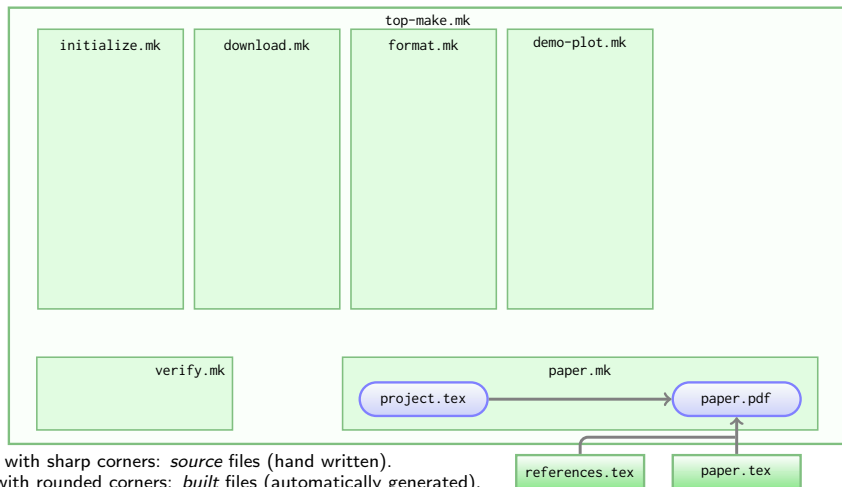
Green boxes with sharp corners: *source* files (hand written).

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`.

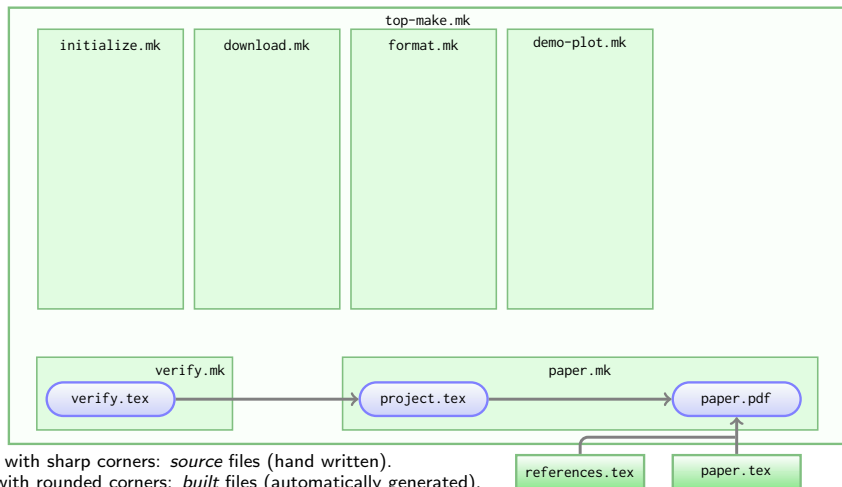


Green boxes with sharp corners: *source* files (hand written).

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



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



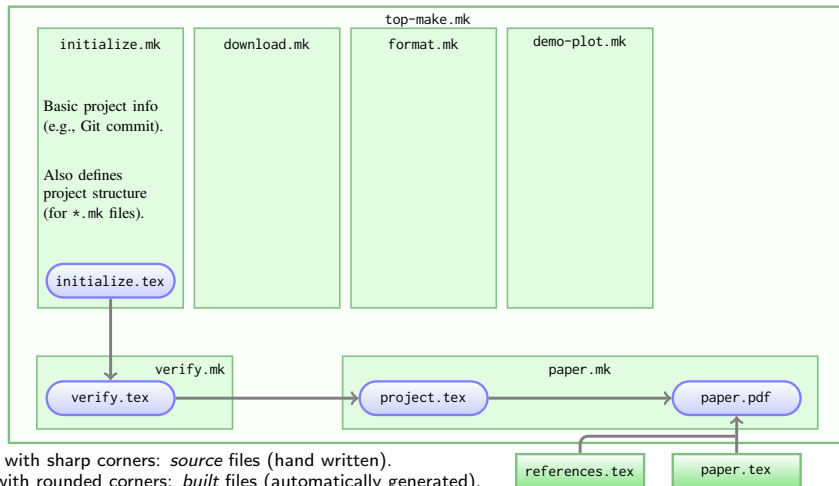
Green boxes with sharp corners: *source* files (hand written).

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`.

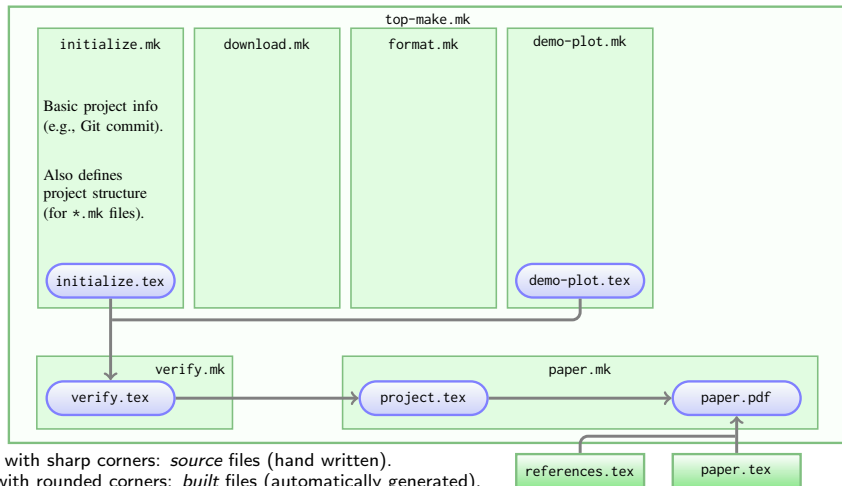


Green boxes with sharp corners: *source* files (hand written).

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.

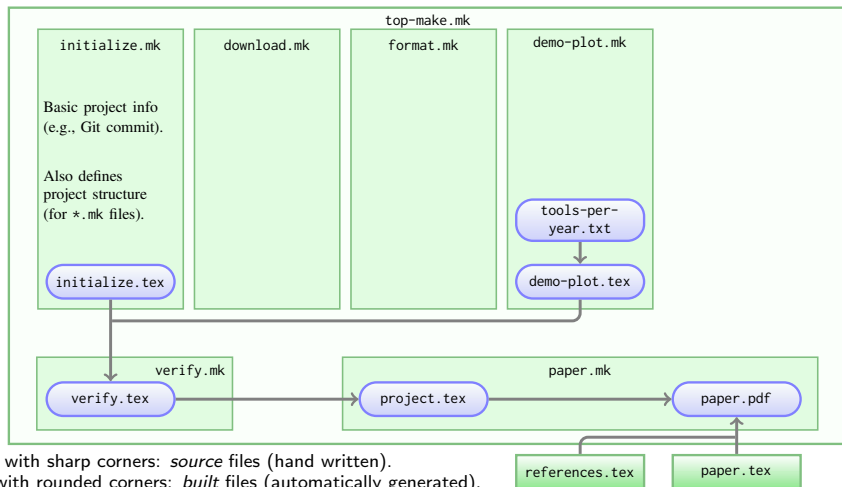


Green boxes with sharp corners: *source* files (hand written).

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

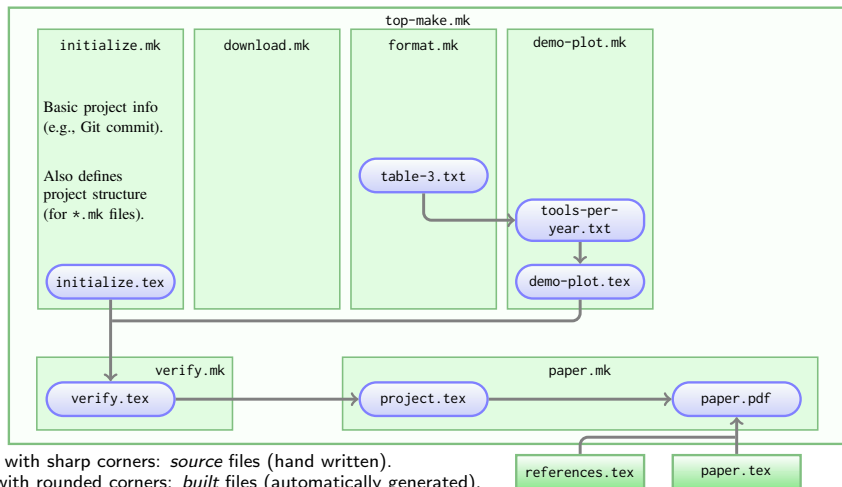


The final plotted data are calculated and stored in `tools-per-year.txt`.





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

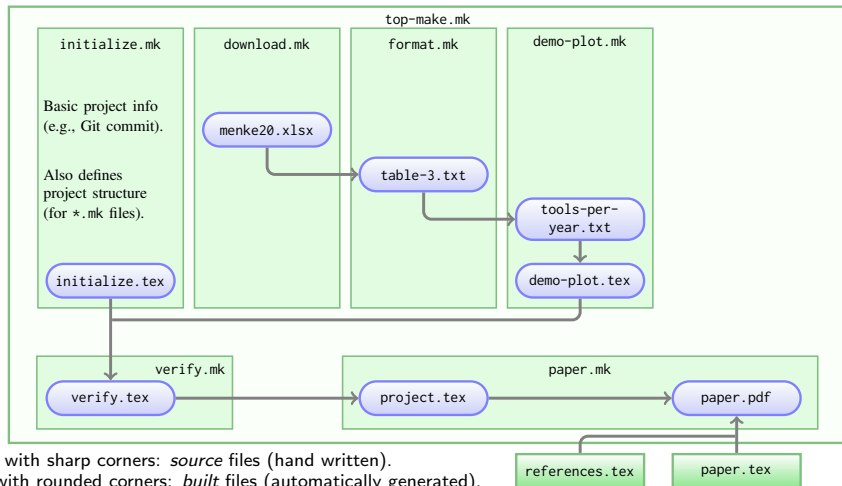


Green boxes with sharp corners: *source* files (hand written).

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

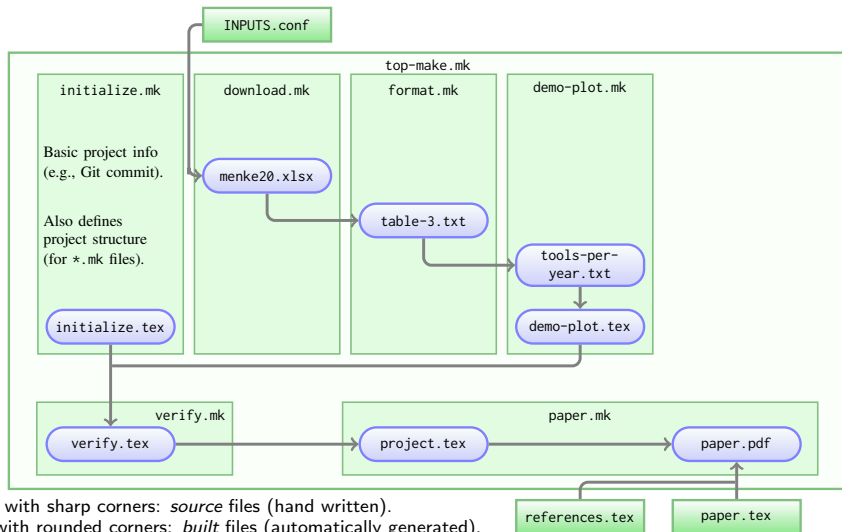


The raw data that were downloaded are stored in XLSX format.





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



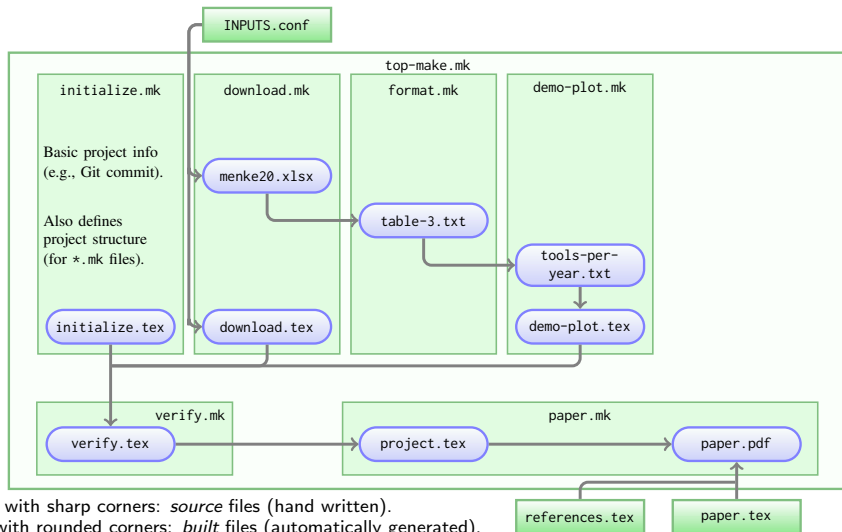
Green boxes with sharp corners: *source* files (hand written).

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

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

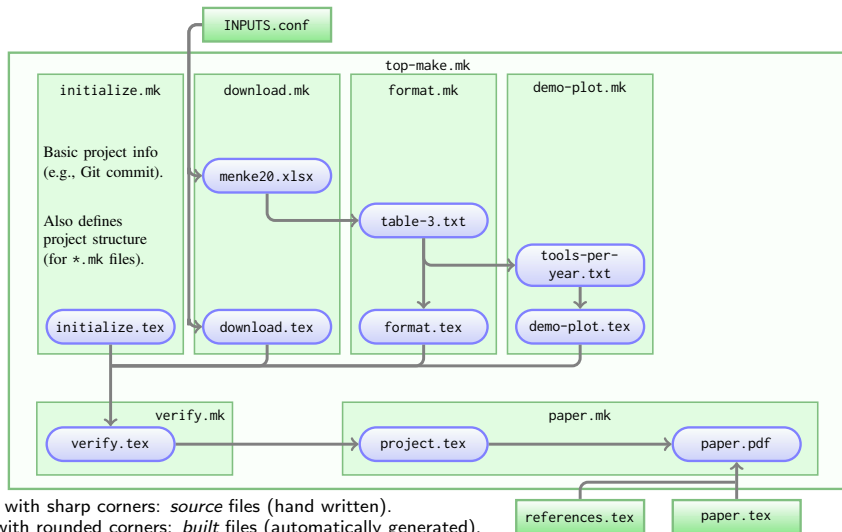


We also need to report the URL in the paper...



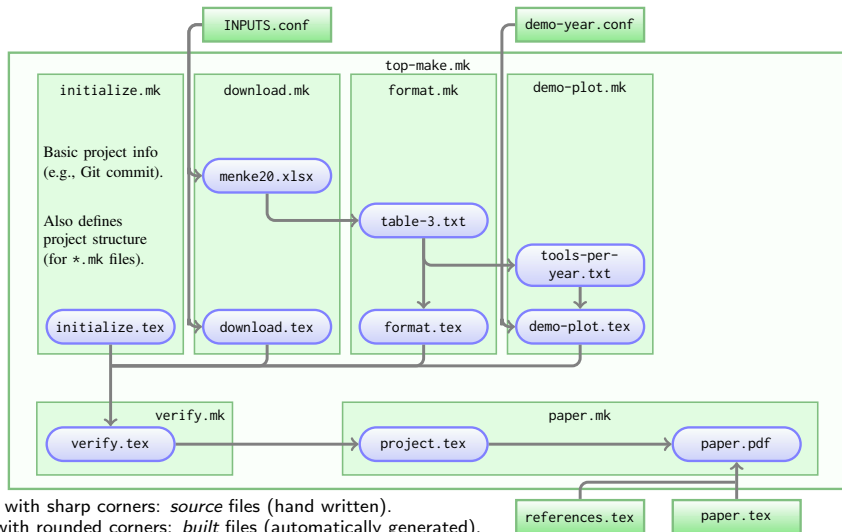


Some general info about the full dataset may also be reported.





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



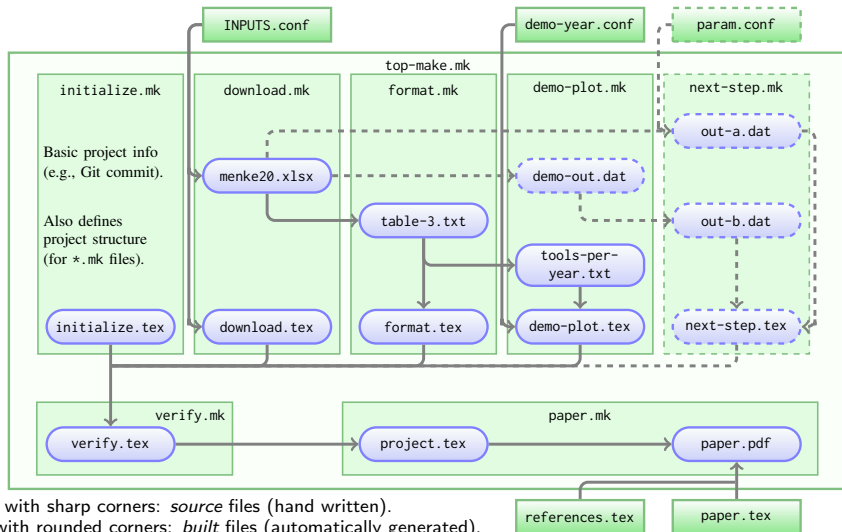
Green boxes with sharp corners: *source* files (hand written).

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.

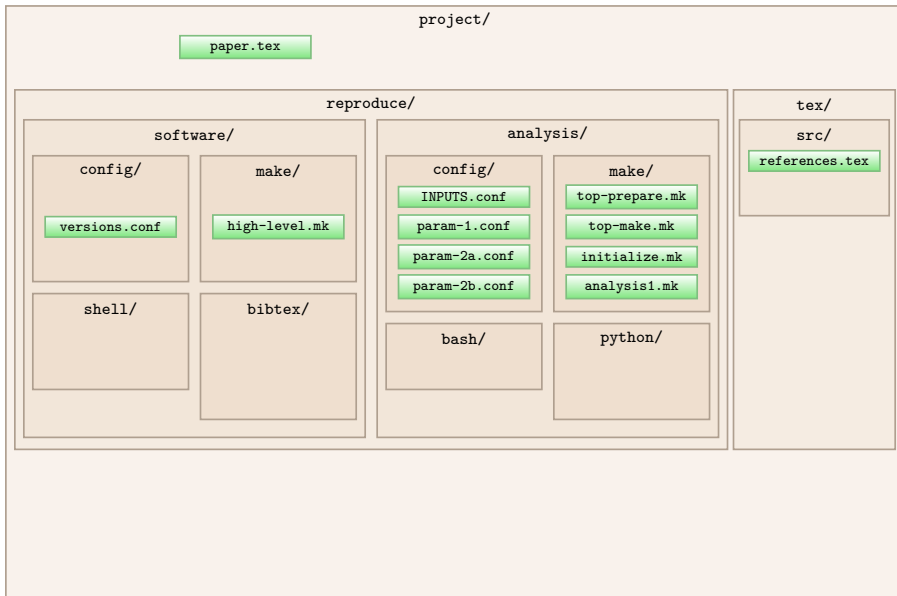


## Benefits of using Make

- ▶ 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.

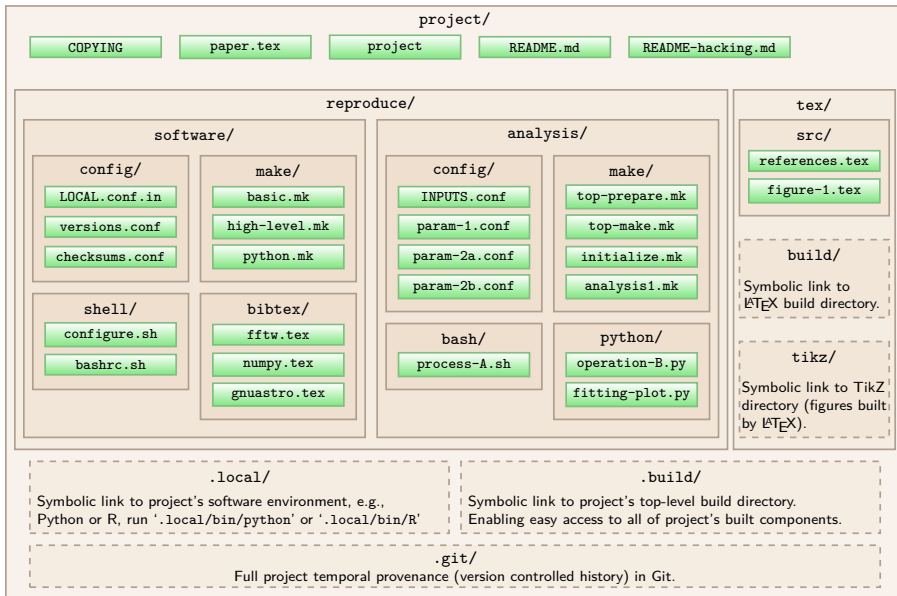


Files organized in directories by context (here are some of the files discussed before)



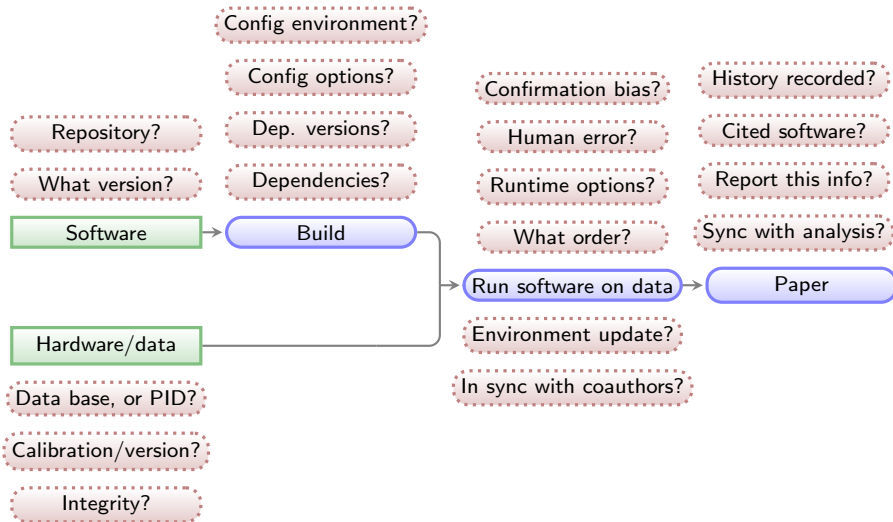


## Files organized in directories by context (now with other project files and symbolic links)





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



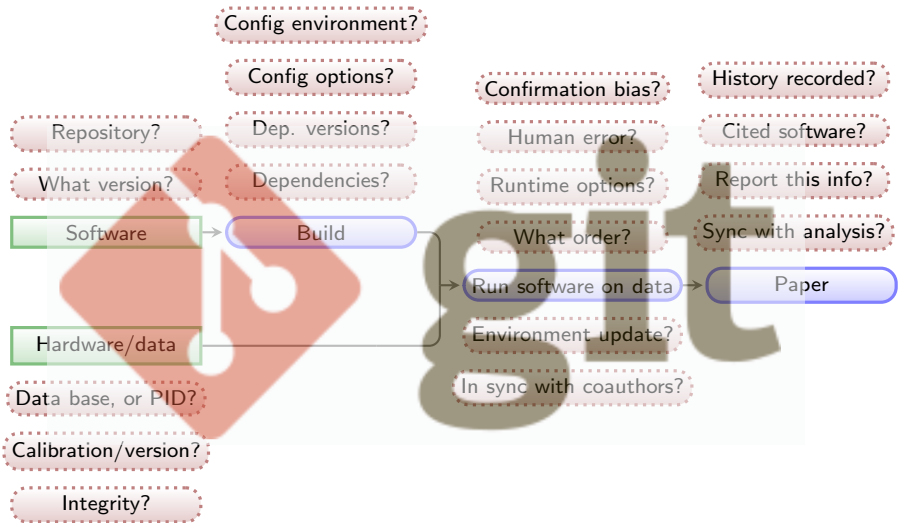
Green boxes with sharp corners: *source*/input components/files.

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

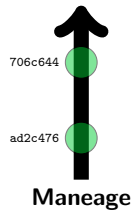
- The project (answers to questions above) will evolve.





## New projects branch from Maneage

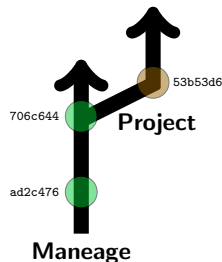
- ▶ Each point of project's history is recorded with Git.





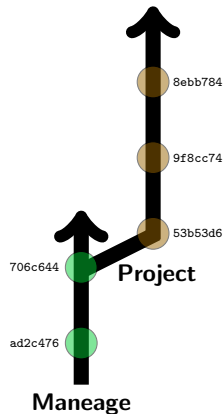
## New projects branch from Maneage

- ▶ Each point of project's history is recorded with Git.
- ▶ New project: a branch from the template.  
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**.





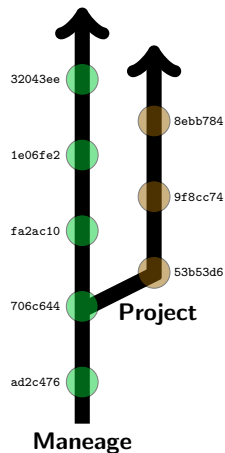
## New projects branch from Maneage



- ▶ Each point of project's history is recorded with Git.
- ▶ New project: a branch from the template.  
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.



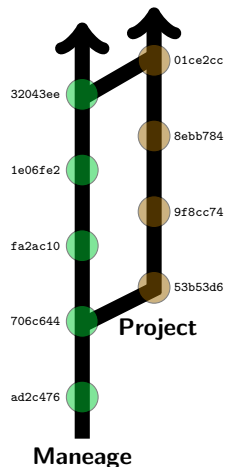
## New projects branch from Maneage



- ▶ Each point of project's history is recorded with Git.
- ▶ New project: a branch from the template.  
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).



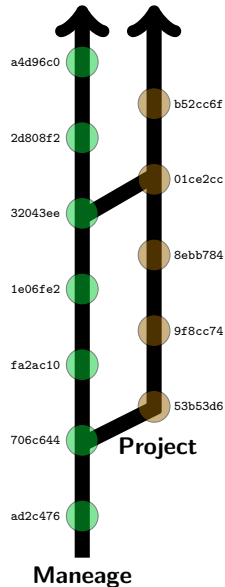
## New projects branch from Maneage



- ▶ Each point of project's history is recorded with Git.
- ▶ New project: a branch from the template.  
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.



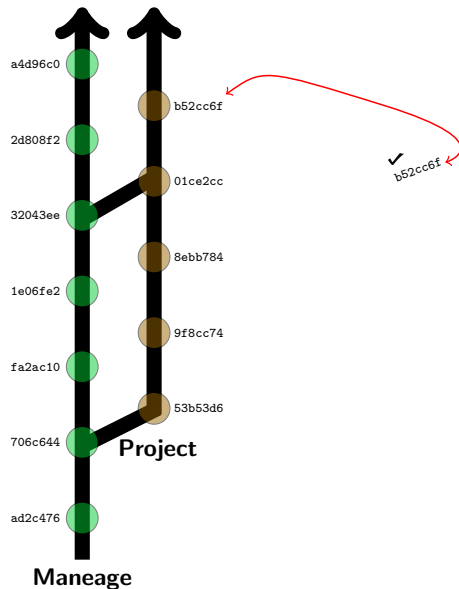
## New projects branch from Maneage



- ▶ Each point of project's history is recorded with Git.
- ▶ New project: a branch from the template.  
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).



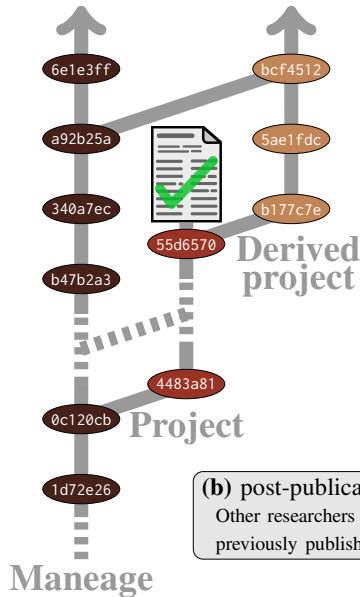
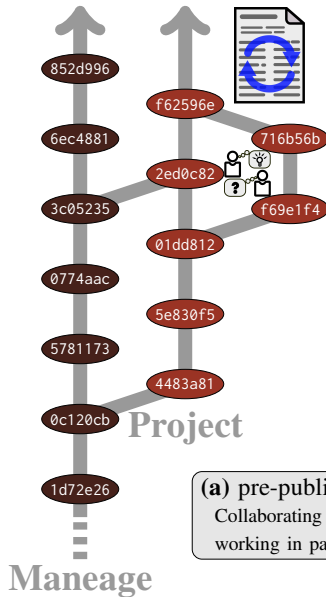
## New projects branch from Maneage



- ▶ Each point of project's history is recorded with Git.
- ▶ New project: a branch from the template.  
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 commit ID identifies the peer-reviewed version of record (apart from journal typesetting, proofreading corrections).



Any Git-based workflow is possible.





## Publication of the project

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

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

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

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

- ▶ **arXiv**: uploaded with the  $\text{\LaTeX}$  source to always stay with the paper (for example [arXiv:2112.14174](#)).
- ▶ **Zenodo**: Along with all the input datasets (many Gigabytes) and software (for example [zenodo.6794222](#)) and given a unique DOI.
  - ▶ ... and put links to data in paper! See the caption of Fig. 3 in [Borkowska & Roukema \(2022\)](#).
- ▶ **Software Heritage**: to archive the full version-controlled history of the project. (for example [swh:1:dir:33fea87068c1612daf011f161b97787b9a0df39fk](#))
  - ▶ ... and put links to exact parts of the code! See caption of Listing 1 in the [Maneage paper](#).



## 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 (e.g. arXiv:2112.14174)

```
$ git clone https://codeberg.org/boud/gevcurvtest      # Import the project.
$ cd gevcurvtest      # Enter the directory.
$ ./project --help    # RTFM. Skip if brave enough.

$ ./project configure      # You will specify the build directory on your system,
                          # and it will build all software (~2-3 hours on 4 cores).

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



## Practical experience: peer-reviewed papers

- ▶ Roukema (2021), PeerJ, 9:e11856, *"Anti-clustering in the national SARS-CoV-2 daily infection counts"*
  - ▶ <https://arXiv.org/abs/2007.11779>
  - ▶ frozen record: <https://zenodo.org/record/4765705>
  - ▶ live git: <https://codeberg.org/boud/subpoisson>
  - ▶ archived git: [sw:h1:rev:72242ca8eade9659031ea00394a30e0cc5cc1c37](https://sw.hic.cc/sw/1:rev:72242ca8eade9659031ea00394a30e0cc5cc1c37)
- ▶ Peper & Roukema (2021), MNRAS, 505, 1223, *"The role of the elaphrocentre in void galaxy formation"*
  - ▶ <https://arXiv.org/abs/2010.03742>
  - ▶ frozen record: <https://zenodo.org/record/4699702>
  - ▶ live git: <https://codeberg.org/boud/elaphrocentre>
  - ▶ archived git: [sw:h1:rev:a029edd32d5cd41dbdac145189d9b1a08421114e](https://sw.hic.cc/sw/1:rev:a029edd32d5cd41dbdac145189d9b1a08421114e)
- ▶ Borkowska & Roukema (2022), CQG, in press, *"Does relativistic cosmology software handle emergent volume evolution?"*
  - ▶ <https://arXiv.org/abs/2112.14174>
  - ▶ frozen record: <https://doi.org/10.5281/zenodo.5806027>
  - ▶ live git: <https://codeberg.org/boud/gevcurvtest>
  - ▶ archived git [sw:h1:rev:d9b47736f81aff9bb8f2f359d9f0331aa923f38d](https://sw.hic.cc/sw/1:rev:d9b47736f81aff9bb8f2f359d9f0331aa923f38d)
- ▶ Was it a lot of work? Yes.
- ▶ Did we feed our variations and fixes upstream to Maneage? Some, yes.
- ▶ Overall feeling: was it worth it? Yes.



## How can you contribute?

- ▶ Try to reproduce Borkowska & Roukema (2022), e.g. from SWH:  
`swh:1:rev:d9b47736f81aff9bb8f2f359d9f0331aa923f38d` (0.5 Mb; fits on a floppy disk)
  - ▶ Did it fully configure, run and verify? If not, then help with a task or a bug:
- ▶ Tasks: <https://savannah.nongnu.org/tasks/?group=reproduce>, e.g.
  - ▶ #15739 debian-verified sources (stable || testing)
  - ▶ #15363 file dates after git checkout
  - ▶ #15997 safe-rm
  - ▶ #15390 glibc within Maneage
- ▶ Bugs: <https://savannah.nongnu.org/bugs/?group=reproduce>, e.g.
  - ▶ #62879 Maneage handling of /dev/shm and required RAM
- ▶ Core Maneage: <https://git.maneage.org/project.git>
- ▶ Merge requests: any git server of your choice
- ▶ Description of implementation, how-to guide, recommendations:  
[https://codeberg.org/boud/maneage\\_dev/src/branch/maneage/README-hacking.md](https://codeberg.org/boud/maneage_dev/src/branch/maneage/README-hacking.md)
- ▶ Interactive discussion: irc at <https://libera.chat>: channel `##maneage`



## Conclusion

Maneage is a customisable template that does the following — all in 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 those components that need to be re-done are re-done.
- ▶ Using  $\text{\LaTeX}$  macros, the paper's figures, tables and numbers will be **Automatically updated**.
- ▶ The whole project is under **version control** (Git) **encouraging tests and experimentation**.
- ▶ The **Git commit hash** of the project source is **printed** in the paper and **on output** data products.

this pdf – full of clickable links: <https://cosmo.torun.pl/~boud/Roukema20220916TYG.pdf>