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

Boud Roukema
Institute of Astronomy, Nicolaus Copernicus University

11 March 2024 @IA NCU

Slides' main authors: Akhlaghi + Roukema; pdf built from git commit ba08d20;
this pdf: https://cosmo.torun.pl/~boud/Roukema20240311IANCU.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 tried to repeat the data analyses in 18 articles in Nature Genetics and only were able to (almost) reproduce only 2 DOI:10.1038/ng.295.

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

Chang & Li (2015) were are able to reproduce just under 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 data or code 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

per US National Academies report, 2019, DOI:10.17226/25303; (pdf)

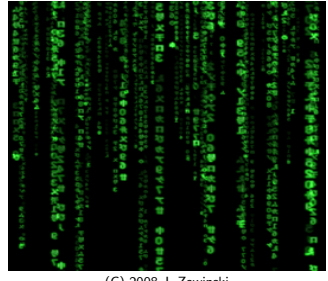
Replicability (hardware/statistical)

- **Similar** data, **similar** method.
- 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

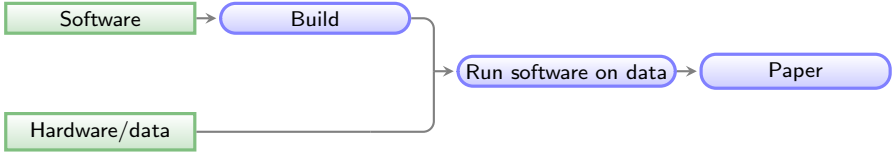
Reproducibility (software/deterministic)

- **Same** data, **same** method.
- May include simulations of data.
- Involves data analysis,
- Example: $2 + 2 = 4$ (i.e., sum of datasets).
- **DISCUSSED HERE.**



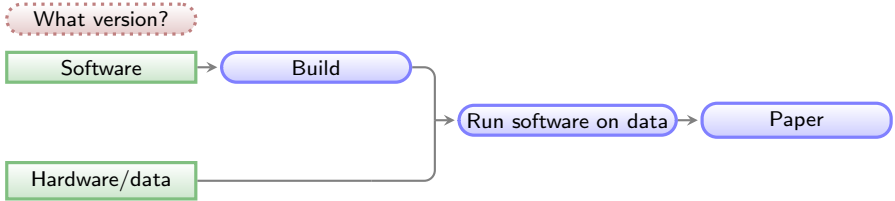
(C) 2008 J. Zawinski

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)

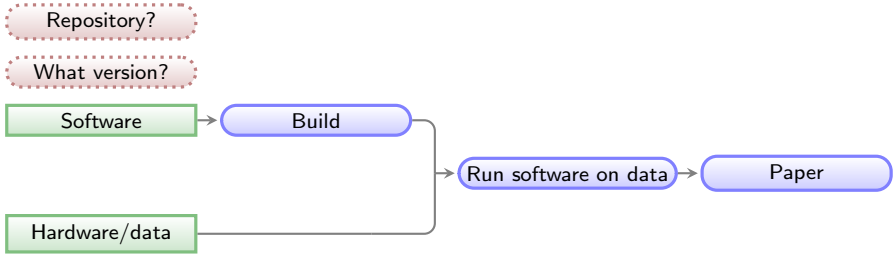
Astropy

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)

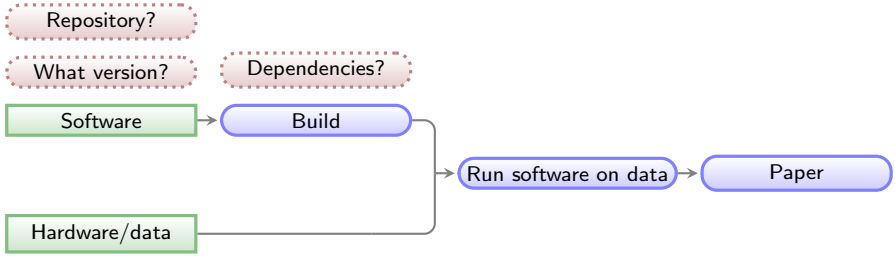
Packaging status	
Debian 9	0.2.33
Debian 10	0.8
Debian 11	0.14
Debian 12	0.16.1
Debian Unstable	0.16.1
Deepin	0.8
Devuan 2.0	0.2.33
Devuan 3.0	0.8
Devuan 4.0	0.14
Devuan Unstable	0.16.1
DPorts	0.15
FreeBSD Ports	0.16
Funtoo 1.4	0.3
Gentoo	0.3
GNU Guix	0.16
Kali Linux Rolling	0.16.1
LiGurOS stable	0.3
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)



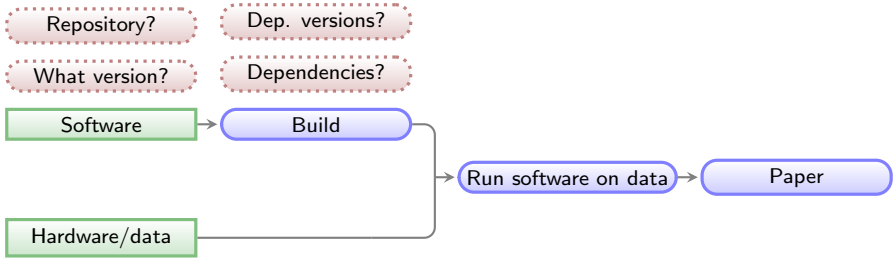
Green boxes with sharp corners: *source*/input components/files.
Blue boxes with rounded corners: *built* components.
Red boxes with dashed borders: questions that must be clarified for each phase.

General outline of a project (after data collection)



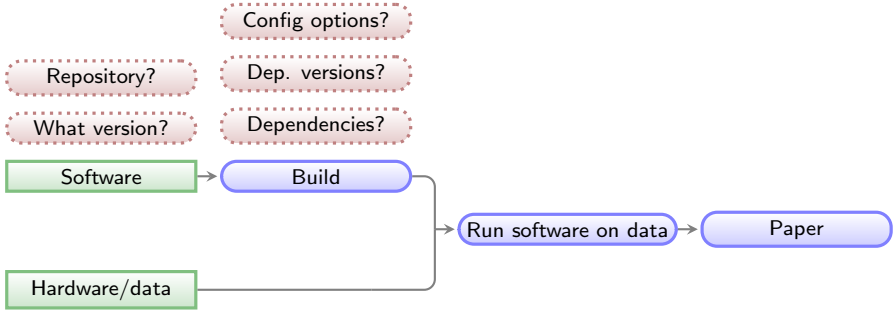
Green boxes with sharp corners: *source*/input components/files.
Blue boxes with rounded corners: *built* components.
Red boxes with dashed borders: questions that must be clarified for each phase.

General outline of a project (after data collection)



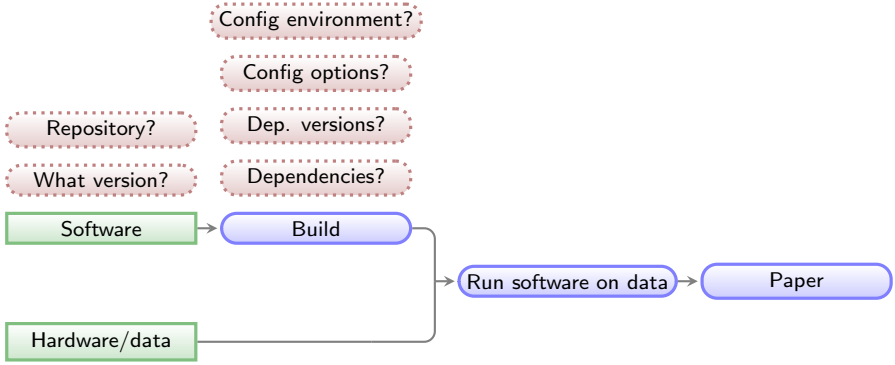
Green boxes with sharp corners: *source*/input components/files.
Blue boxes with rounded corners: *built* components.
Red boxes with dashed borders: questions that must be clarified for each phase.

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

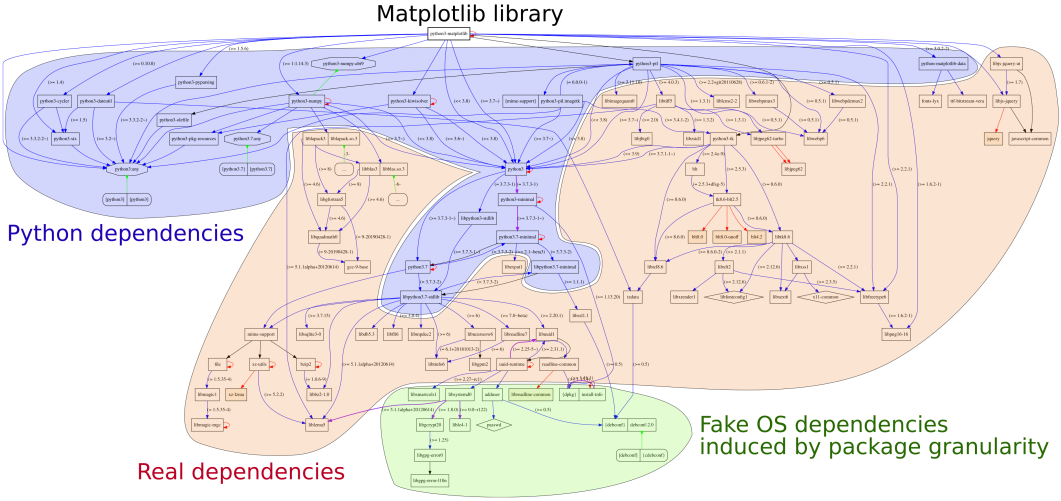
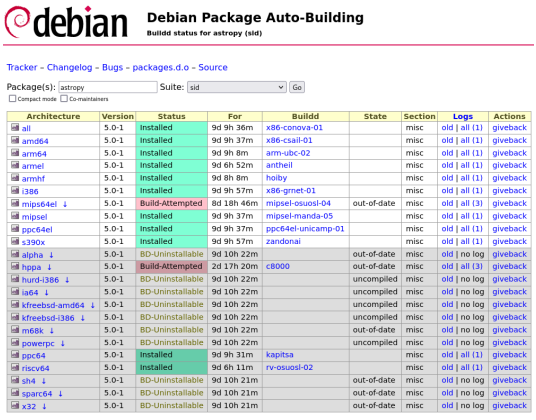


Fig. 1. Transitive dependencies of the software environment required by a simple "`import matplotlib`" command in the Python 3 interpreter.

From "Attributing and Referencing (Research) Software: Best Practices and Outlook from Inria" (Alliez et al. 2020, CiSE, DOI:10.1109/MCSE.2019.2949413).

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



debian **Debian Package Auto-Building**

Buildd status for astropy (sid)

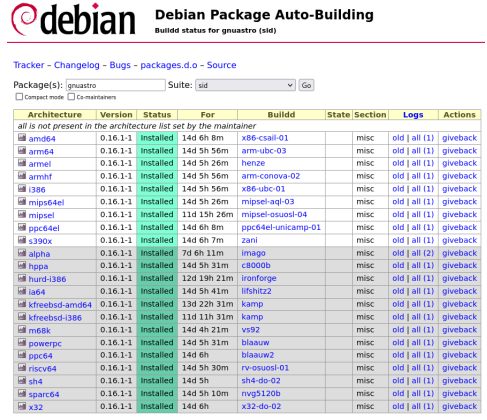
Tracker - Changelog - Bugs - packages.d.o - Source

Package(s): astropy Suite: sid Go

Compact mode Co-maintainers

Architecture	Version	Status	For	Buildd	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 **Debian Package Auto-Building**

Buildd status for gnuastro (sid)

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

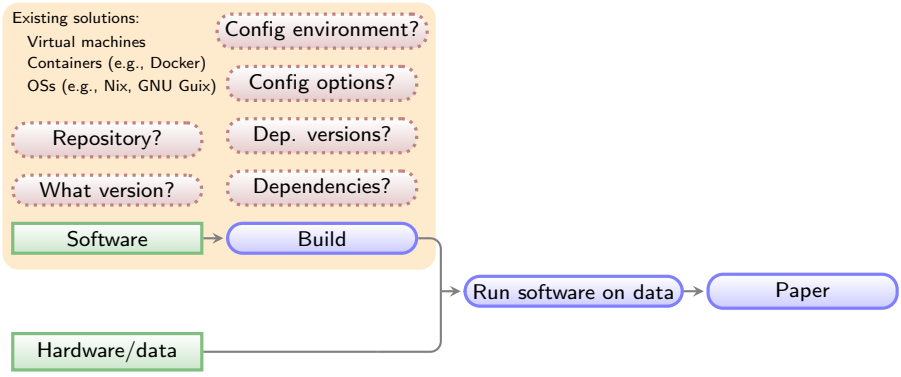
Package(s): gnuastro Suite: sid Go

Compact mode Co-maintainers

Architecture	Version	Status	For	Buildd	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	lifshitz2		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	blaauw		misc	old | all (1)	giveback
ppc64	0.16.1-1	Installed	14d 6h	blaauw2		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	nvg5120b		misc	old | all (1)	giveback
x32	0.16.1-1	Installed	14d 6h	x32-do-02		misc	old | all (1)	giveback

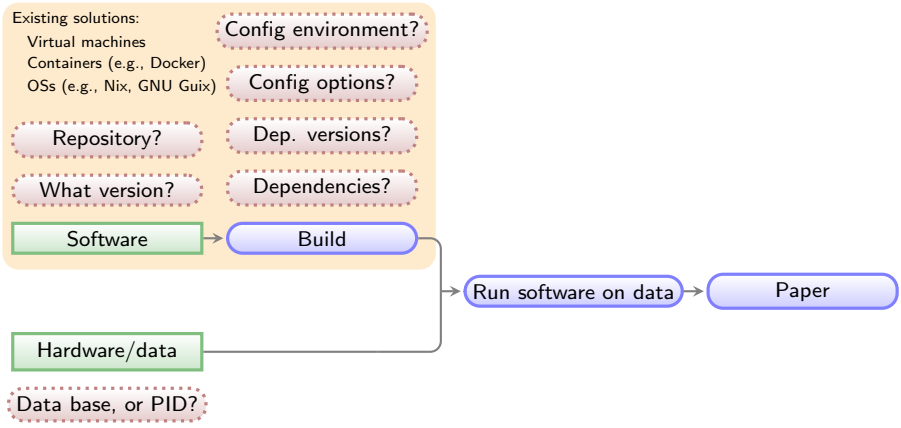
GNU Astronomy Utilities doesn't.

General outline of a project (after data collection)



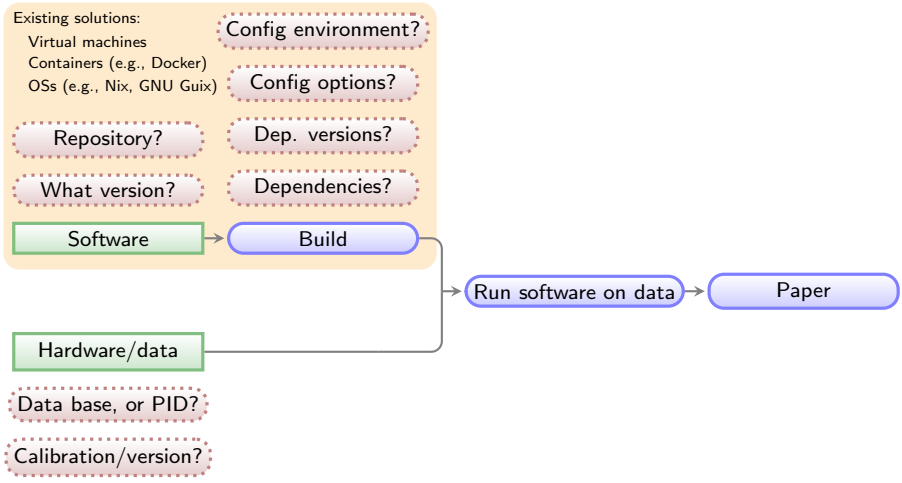
Green boxes with sharp corners: *source*/input components/files.
Blue boxes with rounded corners: *built* components.
Red boxes with dashed borders: questions that must be clarified for each phase.

General outline of a project (after data collection)



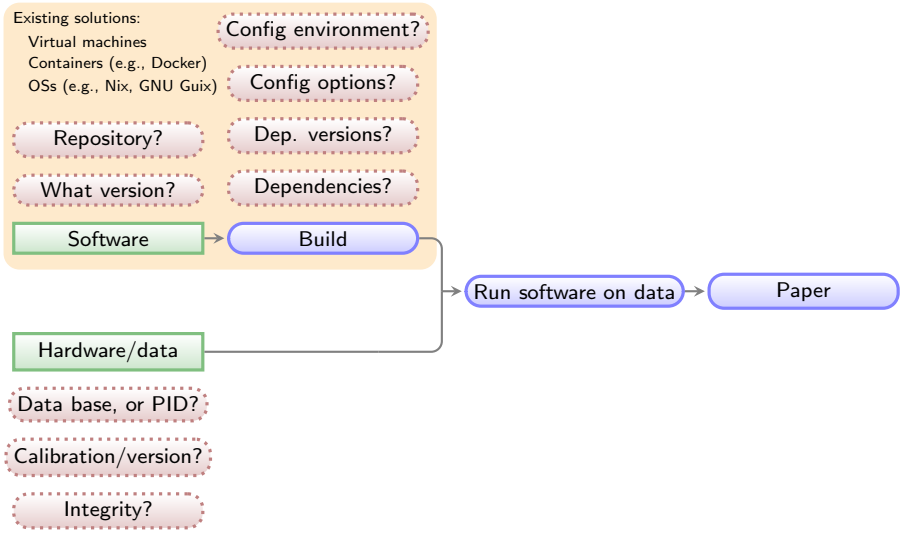
Green boxes with sharp corners: *source*/input components/files.
Blue boxes with rounded corners: *built* components.
Red boxes with dashed borders: questions that must be clarified for each phase.

General outline of a project (after data collection)



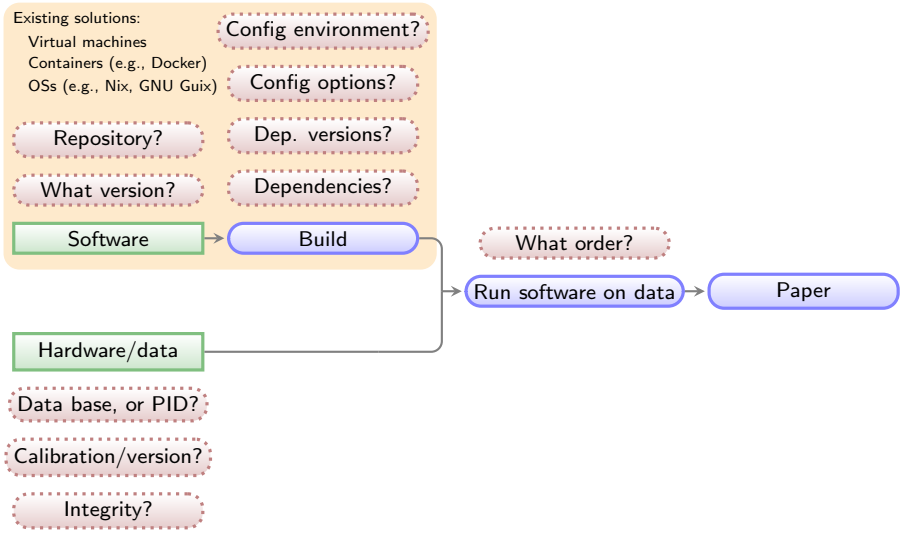
Green boxes with sharp corners: *source*/input components/files.
Blue boxes with rounded corners: *built* components.
Red boxes with dashed borders: questions that must be clarified for each phase.

General outline of a project (after data collection)



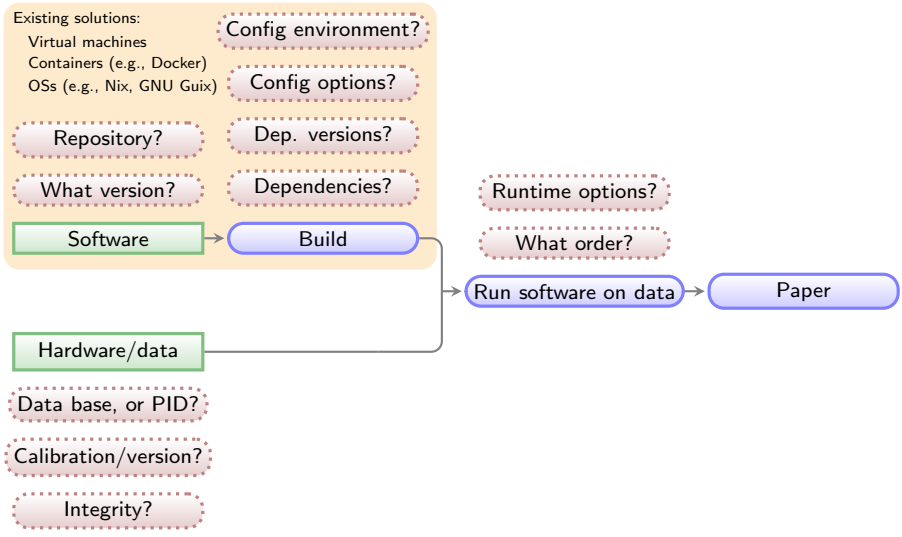
Green boxes with sharp corners: *source*/input components/files.
Blue boxes with rounded corners: *built* components.
Red boxes with dashed borders: questions that must be clarified for each phase.

General outline of a project (after data collection)



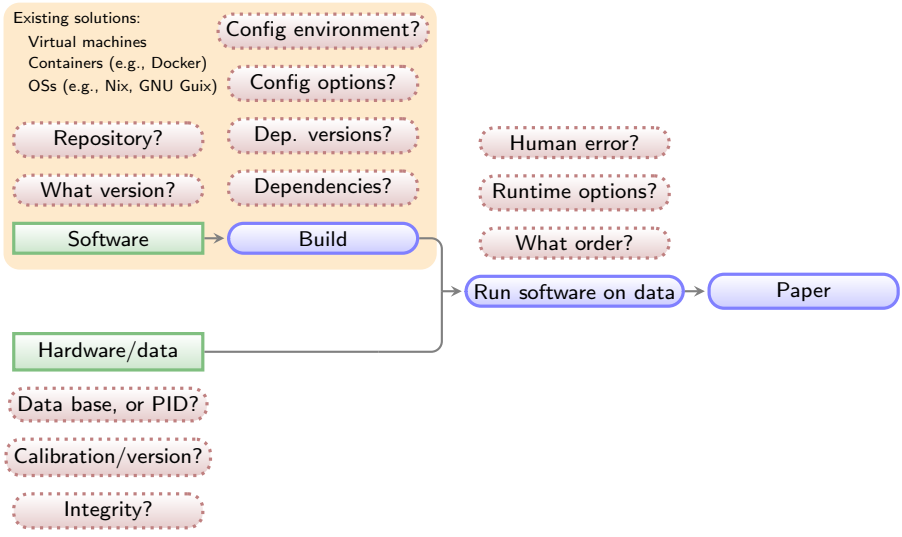
Green boxes with sharp corners: *source*/input components/files.
Blue boxes with rounded corners: *built* components.
Red boxes with dashed borders: questions that must be clarified for each phase.

General outline of a project (after data collection)



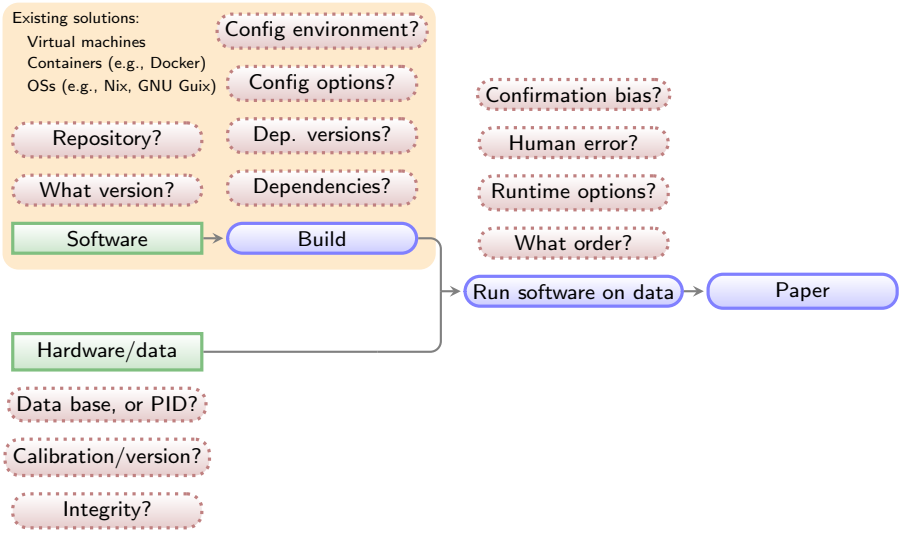
Green boxes with sharp corners: *source*/input components/files.
Blue boxes with rounded corners: *built* components.
Red boxes with dashed borders: questions that must be clarified for each phase.

General outline of a project (after data collection)



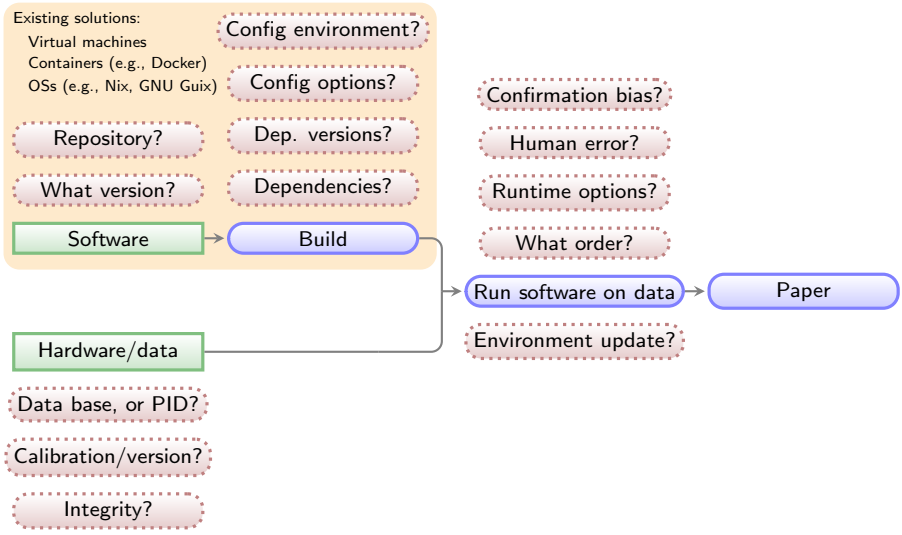
Green boxes with sharp corners: *source*/input components/files.
Blue boxes with rounded corners: *built* components.
Red boxes with dashed borders: questions that must be clarified for each phase.

General outline of a project (after data collection)



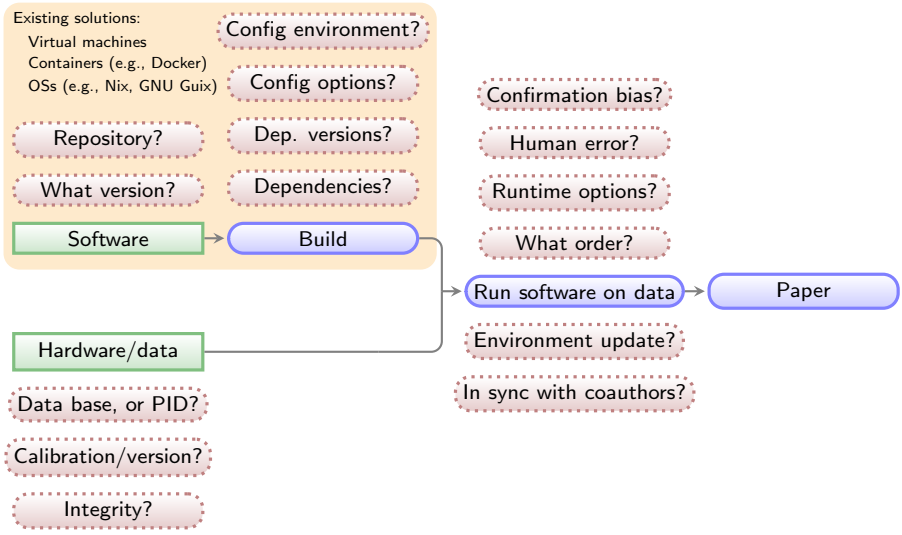
Green boxes with sharp corners: *source*/input components/files.
Blue boxes with rounded corners: *built* components.
Red boxes with dashed borders: questions that must be clarified for each phase.

General outline of a project (after data collection)



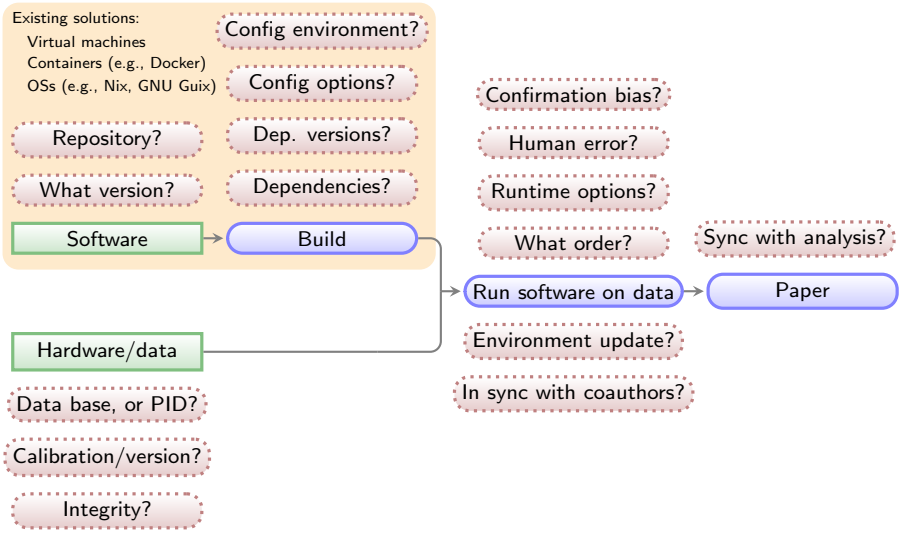
Green boxes with sharp corners: *source*/input components/files.
Blue boxes with rounded corners: *built* components.
Red boxes with dashed borders: questions that must be clarified for each phase.

General outline of a project (after data collection)



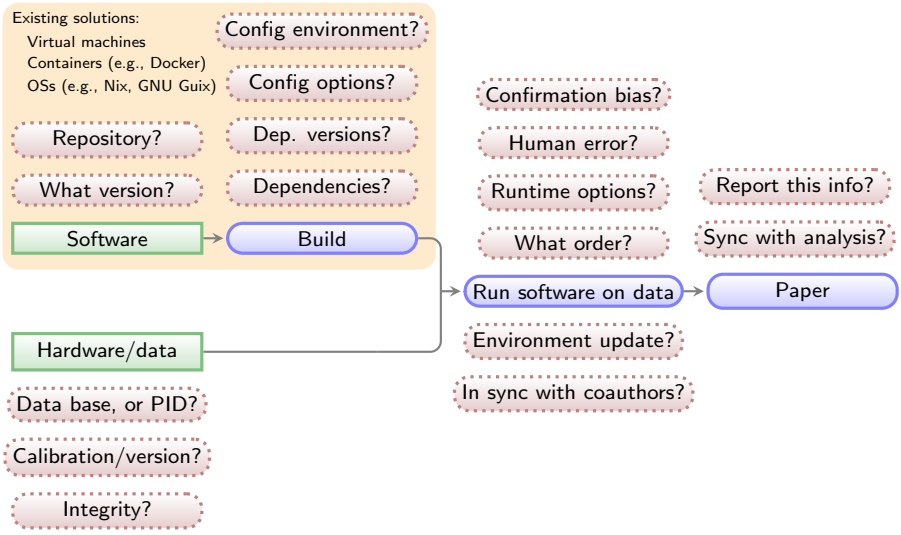
Green boxes with sharp corners: *source*/input components/files.
Blue boxes with rounded corners: *built* components.
Red boxes with dashed borders: questions that must be clarified for each phase.

General outline of a project (after data collection)



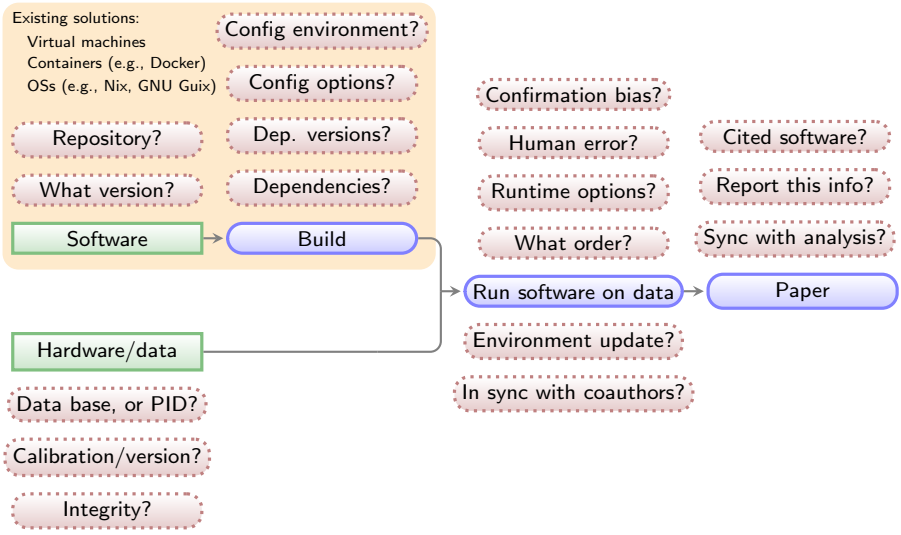
Green boxes with sharp corners: *source*/input components/files.
Blue boxes with rounded corners: *built* components.
Red boxes with dashed borders: questions that must be clarified for each phase.

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)

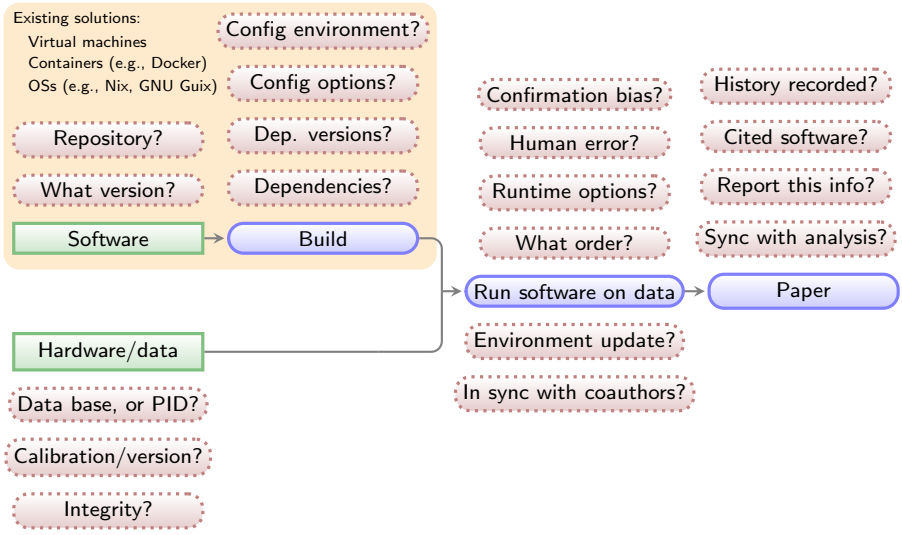


Green boxes with sharp corners: *source*/input components/files.
Blue boxes with rounded corners: *built* components.
Red boxes with dashed borders: questions that must be clarified for each phase.

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

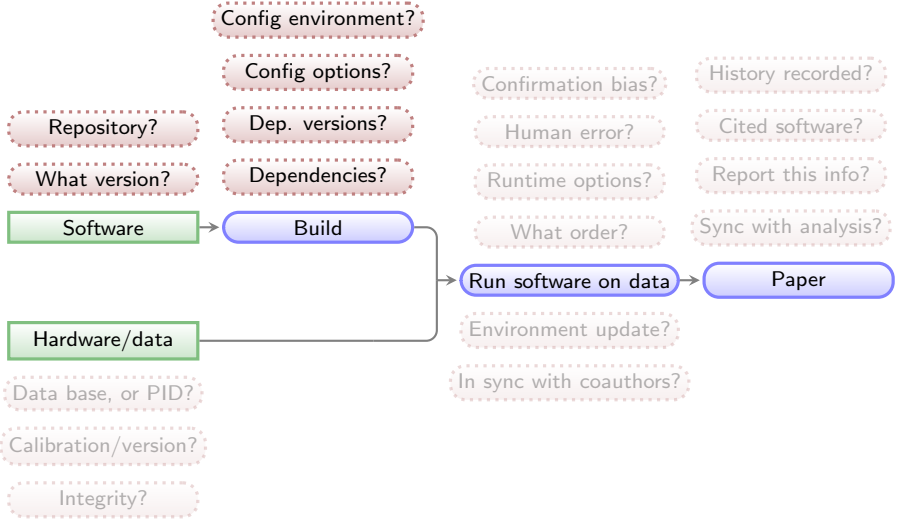
"**Software is a hybrid** object in the world [of] research as it is equally a driving force (as a tool), a result (as proof of the existence of a solution) and an object of study (as an artefact)".

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.

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 (FOSS):** Free software is essential: non-free software is not configurable, not distributable, and is dependent on a non-free provider (who may discontinue it after e.g. 5–10 years).

Reminder: Free Software Definition:

freedom to: **0: use** **1: modify** **2: distribute** **3: distribute modified versions**

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

EDITORS: Lorena A. Barba, labarba@gwu.edu
Sandra Gesing, sandra.gesing@nd.edu

SPECIAL TRACK: REPRODUCIBLE RESEARCH

Toward Long-Term and Archivable Reproducibility

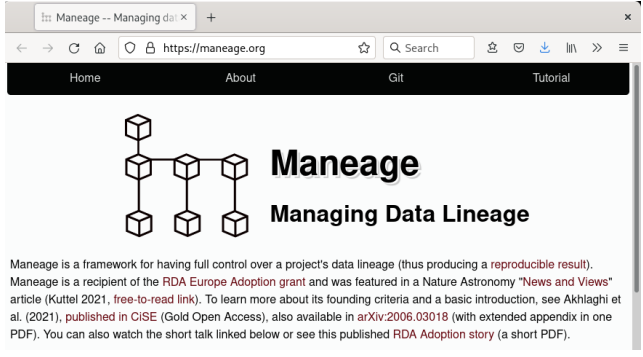
Mohammad Akhlaghi, *Instituto de Astrofísica de Canarias, La Laguna, Tenerife, 38205, Spain*
Raúl Infante-Sainz, *Universidad de La Laguna, La Laguna, Tenerife, 38205, Spain*
Boudewijn F. Roukema, *Nicolaus Copernicus University, Toruń 87-100, Poland*
Mohammadreza Khellat, *Ideal-Information, PC 133 Al Khuwair, Muscat, Oman*
David Valls-Gabaud, *Paris Observatory, Paris 75014, France*
Roberto Baena-Gallé, *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 (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 git.maneage.org/project..git, which is also archived in Software Heritage: swh:1:dir:33fea8706bc1612daf011f161b9787b9a0df39f. 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

Digital Object Identifier 10.1109/MCSE.2021.3072860
Date of publication 13 April 2021; date of current version 15 June 2021.



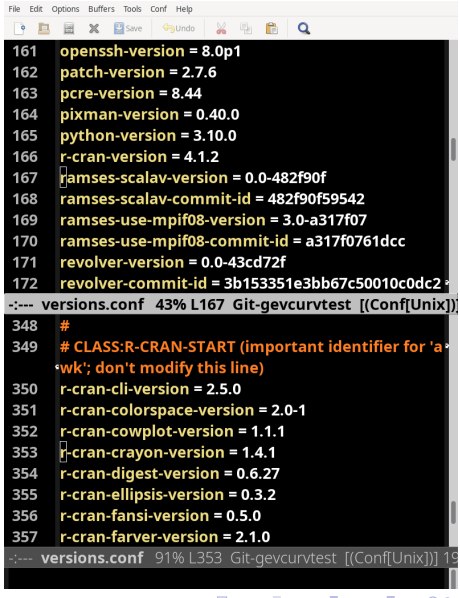
https://maneage.org

Predefined/exact software tools

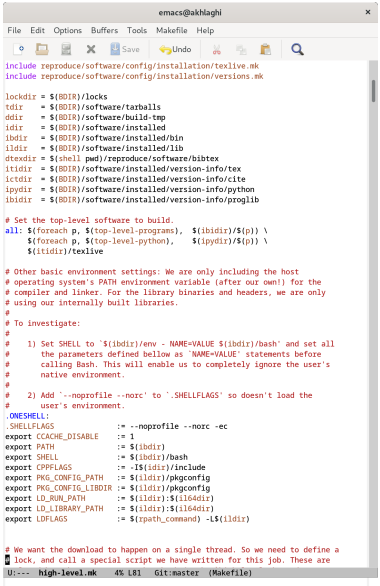
Reproducibility & software

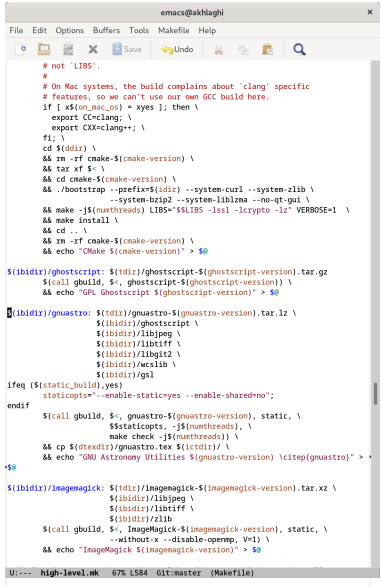
Reproducing the environment (specific software versions, build instructions and dependencies) is also critically important for reproducibility.

- *Containers* or *Virtual Machines* are a binary black box.
- Maneage installs fixed versions of all necessary research software and its dependencies.
- Installs similar environment on GNU/Linux, or Darwin/Xnu (macOS, iOS) systems.
- Works very much like a package manager (e.g., `apt` or `brew`).

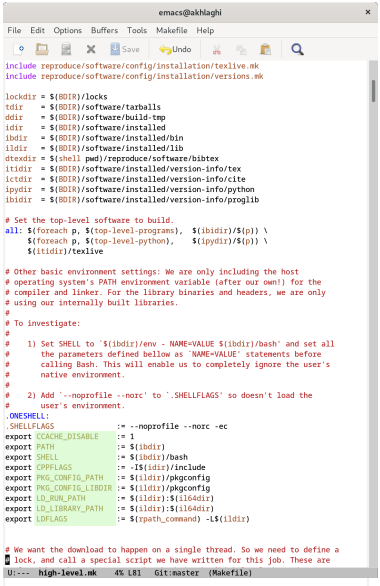


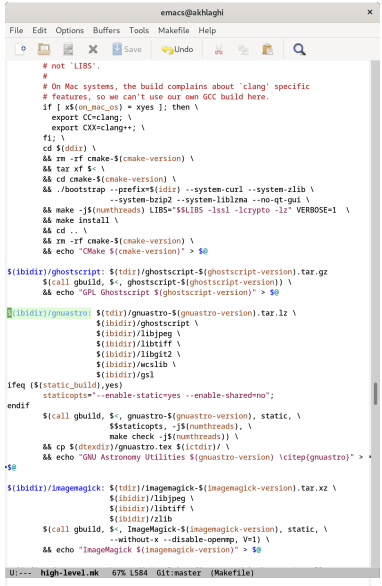
Controlled environment and build instructions





Controlled environment and build instructions





Example: Matplotlib (a Python visualization library) build dependencies

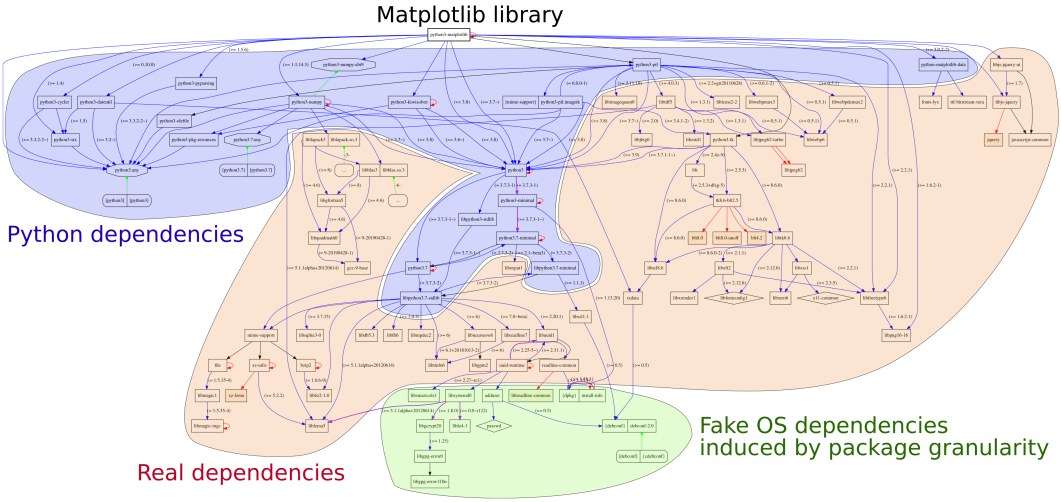


Fig. 1. Transitive dependencies of the software environment required by a simple "`import matplotlib`" command in the Python 3 interpreter.

From "Attributing and Referencing (Research) Software: Best Practices and Outlook from Inria" (Alliez et al. 2019, hal-02135891)

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 (0x00007fffdcbf7000)
    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
```

Darwin/Xnu

```
$ 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, LaTeX, etc.
- No need for root/administrator permissions (on servers or super computers).
- Whole system is built automatically on any Unix-like operating system ($\sim$ 2–3 hours; Sep 2022).
- Dependencies of different projects will not conflict.
- Everything in plain text (human & computer readable/archivable).

Software citation automatically generated in paper

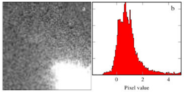


Figure 2: (a) An example image of the Wide-Field Planetary Camera 2, on board the Hubble Space Telescope from 1993 to 2009. This is one of the sample images from the FITS standard webpage, kept as examples for this file format. (b) Histogram of pixel values in (a).

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 PGFPlots is built by LaTeX 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 manual[5].

This template also defines two LaTeX macros that allow you to mark text within your document as *new* and *notes*. 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 `tonote` 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).

[5] http://mirrors.ctan.org/graphics/pgf/contrib/pgfplots/doc/pgfplots.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 Gnuastro, so it benefited from the same grants. If you don't use Gnuastro 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 (Gnuastro, ascl.net/1801.009), and the reproducible paper template v0-364-g268dfc0-dirty. Work on Gnuastro 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, 24253003), the European Research Council (ERC) advanced grant 339659-MUSICOS, European Union's Horizon 2020 research and innovation programme under Marie Sklodowska-Curie grant agreement No 721463 to the SUNDIAL ITN, and from the Spanish Ministry of Economy and Competitiveness (MINECO) under grant number AYA2016-76219-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, Discoteq 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-e3fc, 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 Readline 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, HDF5 library 1.10.5, ImageMagick 7.0.8-46, Libbsd 0.9.1, Libgit2 0.26.0, Libjpeg v9b, Libpng 1.6.37, Libtiff 4.0.10, Lzip 1.20, Metastore (forked) 1.1.2-23-fa9170b, OpenBLAS 0.3.5, Open MPI 4.0.1, OpenSSL 1.1.1a, PatchELF 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), Cycler 0.10.0, Cython 0.29.6 (Behnel et al. 2011), h5py 2.9.0, 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-dateutil 2.8.0, Scipy 1.2.1 (Oliphant 2007; Millman and Aivazis 2011), Setuptools 40.8.0, Setuptools-scm 3.2.0 and Six 1.12.0. The LaTeX source of the paper was compiled to make the PDF using the following packages: biber 2.12, biber 2.12, biblatex 3.12, biblatex 3.12, caption 2018-10-05, caption 2018-10-05, courier 2016-06-24, courier 2016-06-24, csquotes 5.2d, datetime 2.60, datetime 2.60, ec 1.0, ec 1.0, etoolbox 2.5f, etoolbox 2.5f, fancyhdr 3.10, fancyhdr 3.10, fmtcount 3.05, fmtcount 3.05, fontaxes 1.0d, fontaxes 1.0d, footmisc 5.5b, footmisc 5.5b, fp 2.1d, fp 2.1d, logreq 1.0, logreq 1.0, newtx 1.554, newtx 1.554, pgf 3.1.2, pgf 3.1.2, pgfplots 1.16, pgfplots 1.16, preprint 2011, preprint 2011, setspace 6.7a, setspace 6.7a, tex 3.14159265, tex 3.14159265, texgyre 2.501, texgyre 2.501, times 2016-06-24, times 2016-06-24, titlesec 2.10.2, titlesec 2.10.2, txfonts 2016-

06-24, txfonts 2016-06-24, ulem 2016-06-24, ulem 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, 123.
Bacon, R. et al. (Nov. 2017). *A&A*, 608, A1.
Behnel, S. et al. (Mar. 2011). *CiSE*, 13, 31.
Hunter, J. D. (2007). *CiSE*, 9, 90.
Millman, K. J. and M. Aivazis (Mar. 2011). *CiSE*, 13, 9.
Oliphant, T. E. (May 2007). *CiSE*, 9, 10.
van der Walt, S. et al. (Mar. 2011). *CiSE*, 13, 22.

Software citation automatically generated in paper

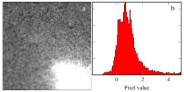


Figure 2: (a) An example image of the Wide-Field Planetary Camera 2, on board the Hubble Space Telescope from 1993 to 2009. This is one of the sample images from the FITS standard webpage, kept as examples for this file format. (b) Histogram of pixel values in (a).

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 PGFPlots is built by LaTeX 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 manual[5].

This template also defines two LaTeX macros that allow you to mark text within your document as *new* and *notes*. 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 `tonote` 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).

[5] http://mirrors.ctan.org/graphics/pgf/contrib/pgfplots/doc/pgfplots.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 Gnuastro, so it benefited from the same grants. If you don't use Gnuastro 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 (Gnuastro, ascl.net/1801.009), and the reproducible paper template v0-364-g268dfc0-dirty. Work on Gnuastro 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, 24253003), the European Research Council (ERC) advanced grant 339659-MUSICOS, European Union's Horizon 2020 research and innovation programme under Marie Sklodowska-Curie grant agreement No 721463 to the SUNDIAL ITN, and from the Spanish Ministry of Economy and Competitiveness (MINECO) under grant number AYA2016-76219-P.

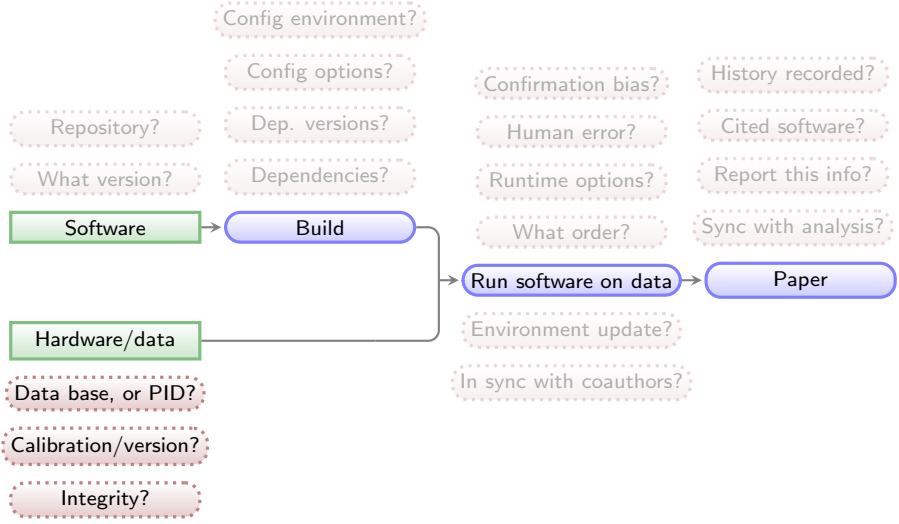
This research was done with the following free software programs and libraries: Bzip2 1.0.6, CFITSIO 3.45, CMake 3.14.2, cURL 7.63.0, Discoteq flock 0.2.3, File 5.36, FreeType 2.9, Git 2.21.0, GNU Astronomy Utilities 0.9 (Akhlaghi and Ichikawa 2015), GNU AWK 5.0.0, GNU Bash 5.0.7, GNU Binutils 2.32, GNU Compiler Collection (GCC) 9.1.0, GNU Coreutils 8.31, GNU Diffutils 3.7, GNU Findutils 4.6.0.199-e3fc, 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 Readline 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, HDF5 library 1.10.5, ImageMagick 7.0.8-46, Libbsd 0.9.1, Libgit2 0.26.0, Libjpeg v9b, Libpng 1.6.37, Libtiff 4.0.10, Lzip 1.20, Metastore (forked) 1.1.2-23-fa9170b, OpenBLAS 0.3.5, Open MPI 4.0.1, OpenSSL 1.1.1a, PatchELF 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), Cycler 0.10.0, Cython 0.29.6 (Behnel et al. 2011), h5py 2.9.0, 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-dateutil 2.8.0, Scipy 1.2.1 (Oliphant 2007; Millman and Aivazis 2011), Setuptools 40.8.0, Setuptools-scm 3.2.0 and Six 1.12.0. The LaTeX source of the paper was compiled to make the PDF using the following packages: biber 2.12, biber 2.12, biblatex 3.12, biblatex 3.12, caption 2018-10-05, caption 2018-10-05, courier 2016-06-24, courier 2016-06-24, csquotes 5.2d, datetime 2.60, datetime 2.60, ec 1.0, ec 1.0, etoolbox 2.5f, etoolbox 2.5f, fancyhdr 3.10, fancyhdr 3.10, fmtcount 3.05, fmtcount 3.05, fontaxes 1.0d, fontaxes 1.0d, footmisc 5.5b, footmisc 5.5b, fp 2.1d, fp 2.1d, logreq 1.0, logreq 1.0, newtx 1.554, newtx 1.554, pgf 3.1.2, pgf 3.1.2, pgfplots 1.16, pgfplots 1.16, preprint 2011, preprint 2011, setspace 6.7a, setspace 6.7a, tex 3.14159265, tex 3.14159265, texgyre 2.501, texgyre 2.501, times 2016-06-24, times 2016-06-24, titlesec 2.10.2, titlesec 2.10.2, txfonts 2016-

06-24, txfonts 2016-06-24, ulem 2016-06-24, ulem 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, 123.
Bacon, R. et al. (Nov. 2017). *A&A*, 608, A1.
Behnel, S. et al. (Mar. 2011). *CiSE*, 13, 31.
Hunter, J. D. (2007). *CiSE*, 9, 90.
Millman, K. J. and M. Aivazis (Mar. 2011). *CiSE*, 13, 9.
Oliphant, T. E. (May 2007). *CiSE*, 9, 10.
van der Walt, S. et al. (Mar. 2011). *CiSE*, 13, 22.

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

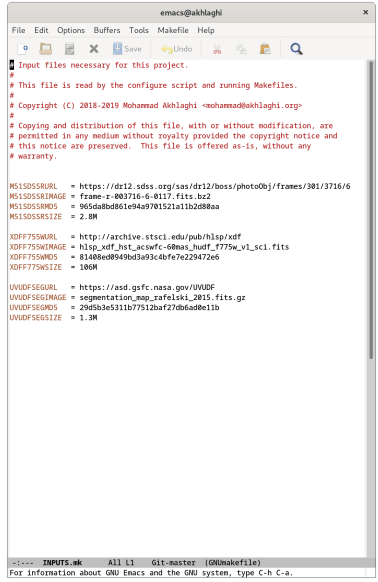
Stored information about each input file:

- PID (Persistent IDentifier if available)
- Download URL (if available).
- checksum (e.g. md5sum, sha512sum) to check integrity (calculate with standard tools)

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

Checksums are checked to make sure the download was done properly or the file is the same (hasn't changed on the server/source).

Example from the reproducible paper arXiv:1909.11230.
This paper needs three input files (two images, one catalog).



Input data source and integrity is documented and checked

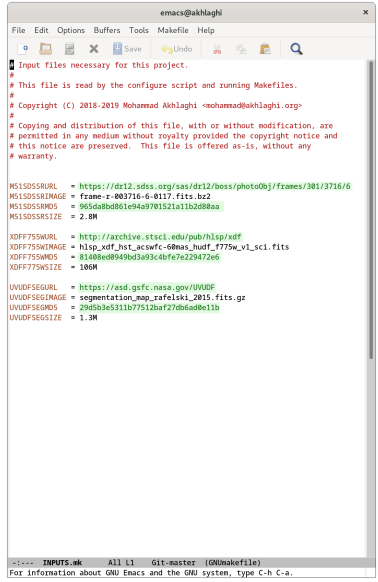
Stored information about each input file:

- PID (Persistent IDentifier if available)
- Download URL (if available).
- checksum (e.g. md5sum, sha512sum) to check integrity (calculate with standard tools)

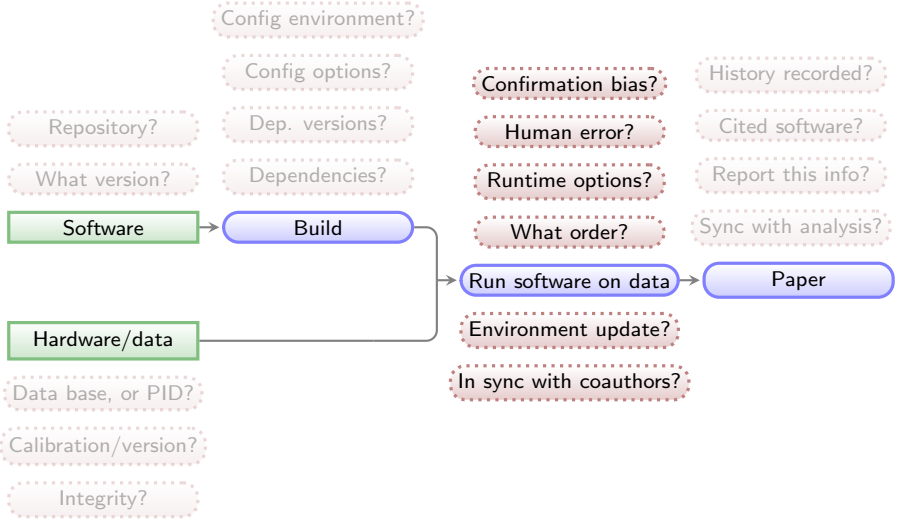
All inputs are downloaded using the given PID/URL when necessary (during the analysis).

Checksums are checked to make sure the download was done properly or the file is the same (hasn't changed on the server/source).

Example from the reproducible paper arXiv:1909.11230.
This paper needs three input files (two images, one catalog).



General outline of a project (after data collection)

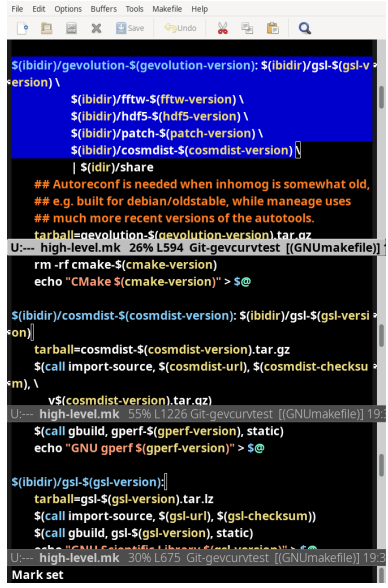


Green boxes with sharp corners: *source*/input components/files.
Blue boxes with rounded corners: *built* components.
Red boxes with dashed borders: questions that must be clarified for each phase.

Reproducible science: Maneage is managed through a Makefile

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

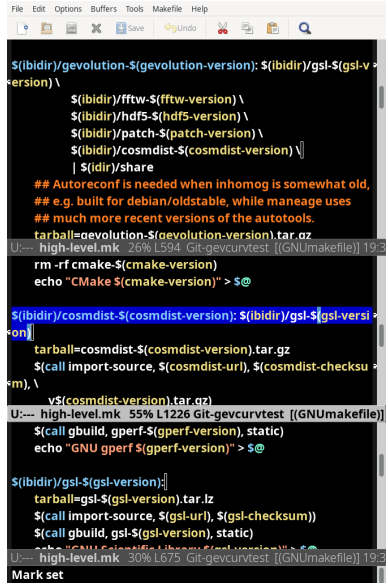
- Unlike a script which always starts from the top, a Makefile starts from the end and steps that don't change will be left untouched (not remade).
- A single *rule* can manage any number of files.
- Make can identify independent steps internally and do them in parallel.
- Make was designed for complex 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 well-checked, free-licensed documentation (for example GNU Make's manual).



Reproducible science: Maneage is managed through a Makefile

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

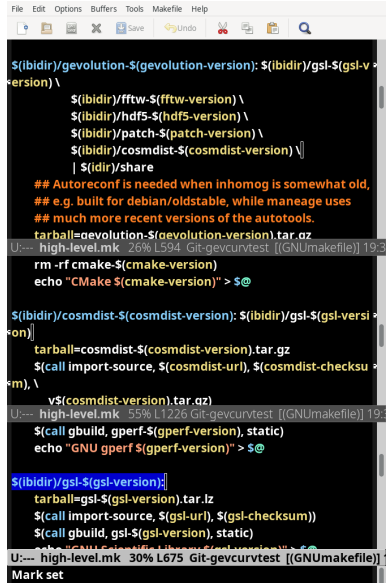
- Unlike a script which always starts from the top, a Makefile starts from the end and steps that don't change will be left untouched (not remade).
- A single *rule* can manage any number of files.
- Make can identify independent steps internally and do them in parallel.
- Make was designed for complex 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 well-checked, free-licensed documentation (for example GNU Make's manual).



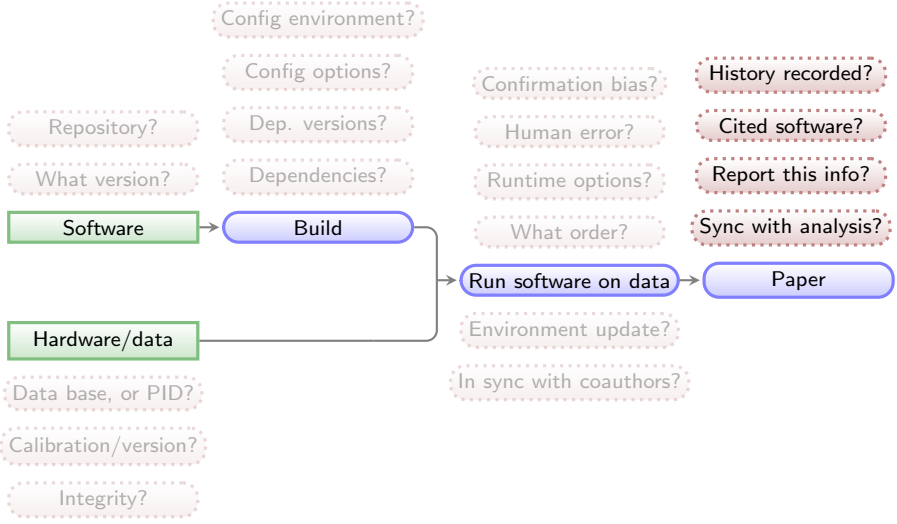
Reproducible science: Maneage is managed through a Makefile

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

- Unlike a script which always starts from the top, a Makefile starts from the end and steps that don't change will be left untouched (not remade).
- A single *rule* can manage any number of files.
- Make 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 well-checked, free-licensed documentation (for example GNU Make's manual).



General outline of a project (after data collection)

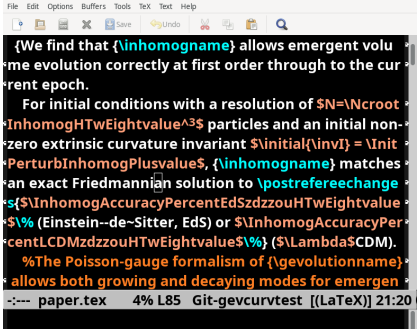


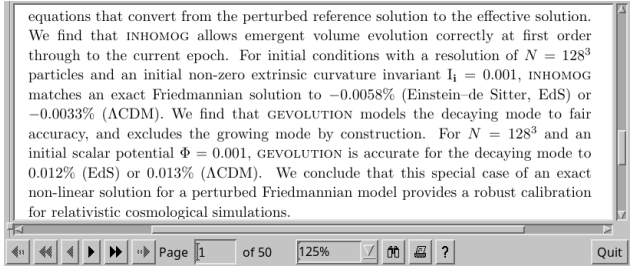
Green boxes with sharp corners: *source*/input components/files.
Blue boxes with rounded corners: *built* components.
Red boxes with dashed borders: questions that must be clarified for each phase.

Values in final report/paper

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

Here is a portion of the Borkowska & Roukema (2022) abstract and its LaTeX source (arXiv:2112.14174).

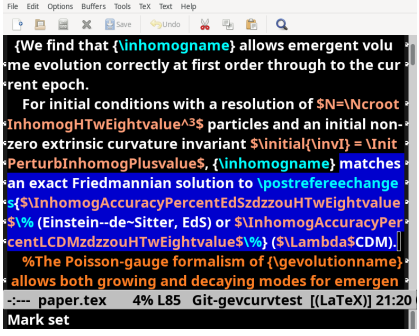


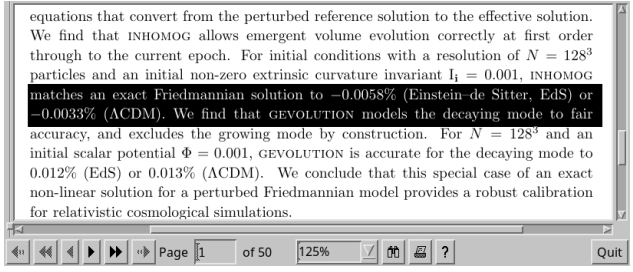


Values in final report/paper

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

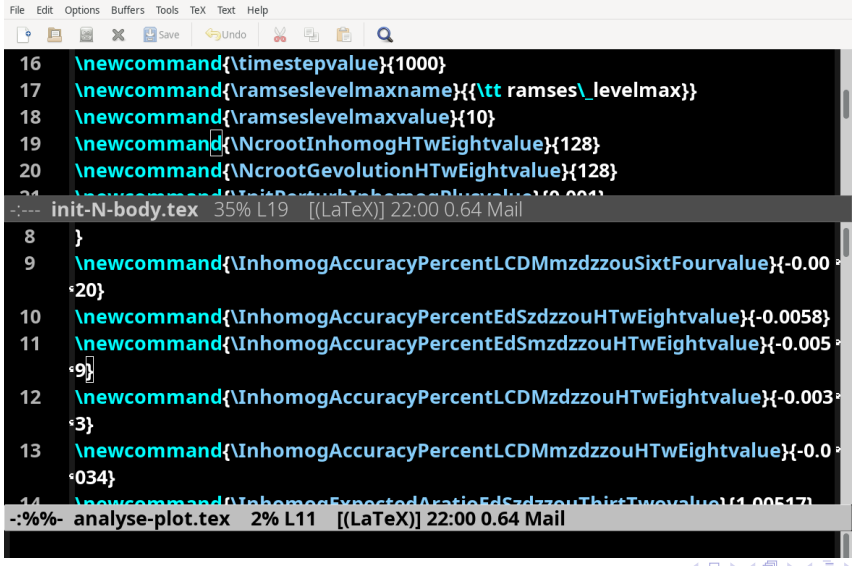
Here is a portion of the Borkowska & Roukema (2022) abstract and its LaTeX source (arXiv:2112.14174).





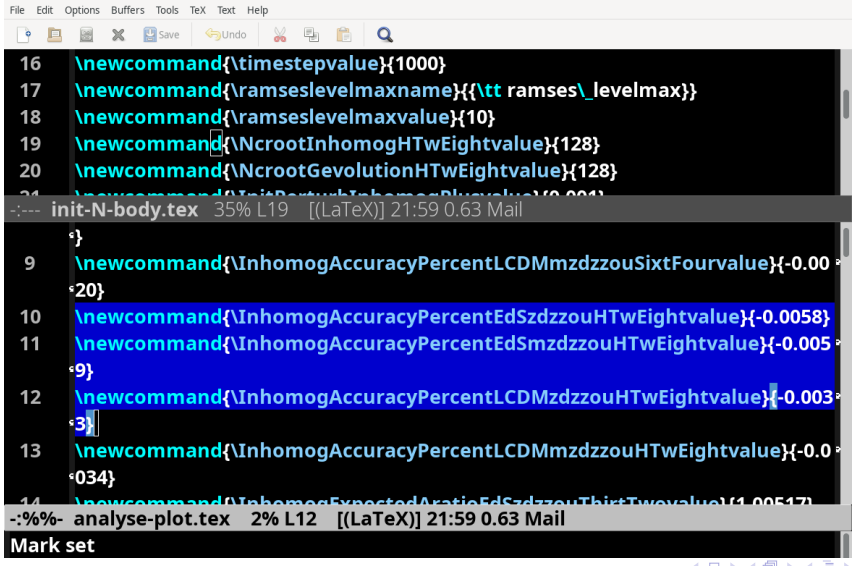
Analysis step results/values stored in a single build directory.

All LaTeX macros are output in a single directory.



Analysis step results/values stored in a single build directory.

All LaTeX macros are output in a single directory.



Analysis results stored as LaTeX macros

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

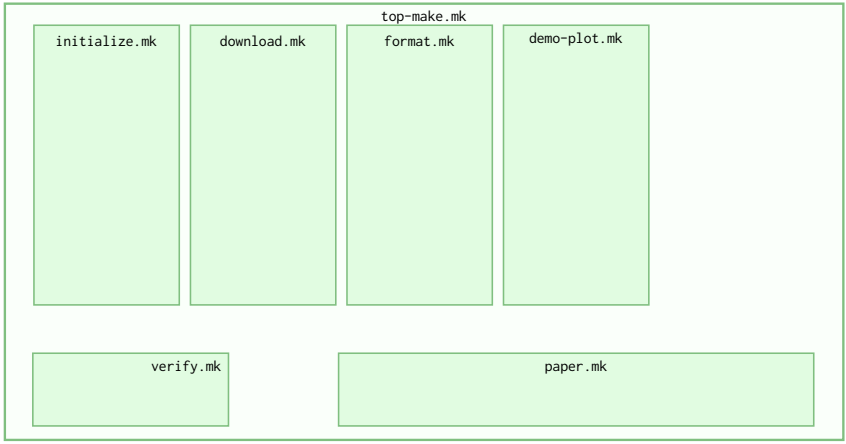
```
# Numbers for dettf.tex:
sqnt=9999999
function dettfhist
{
    # Set the file name.
    if [ $2 == 4 ]; then                obase=four;
    elif [ $2 = sensitivity3 ]; then    obase=sensitivityc;
    else                                obase=$2;
    fi
    if [ $2 == onelarge ]; then ind="_7"; else 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 dettfsmallestsnqnt $sqnt
```

Analysis results stored as LaTeX macros

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

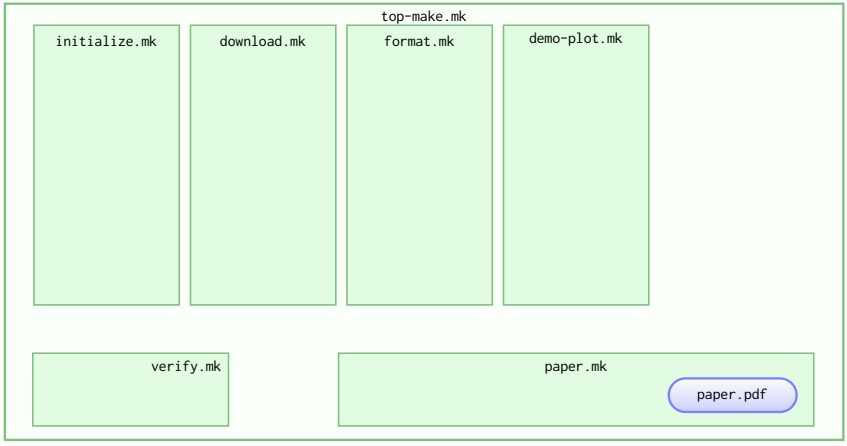
```
# Numbers for dettf.tex:
sqnt=9999999
function dettfhist
{
    # Set the file name.
    if [ $2 == 4 ]; then                obase=four;
    elif [ $2 = sensitivity3 ]; then    obase=sensitivityc;
    else                                obase=$2;
    fi
    if [ $2 == onelarge ]; then ind="_7"; else 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 dettfsmallestsnqnt $sqnt
```

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



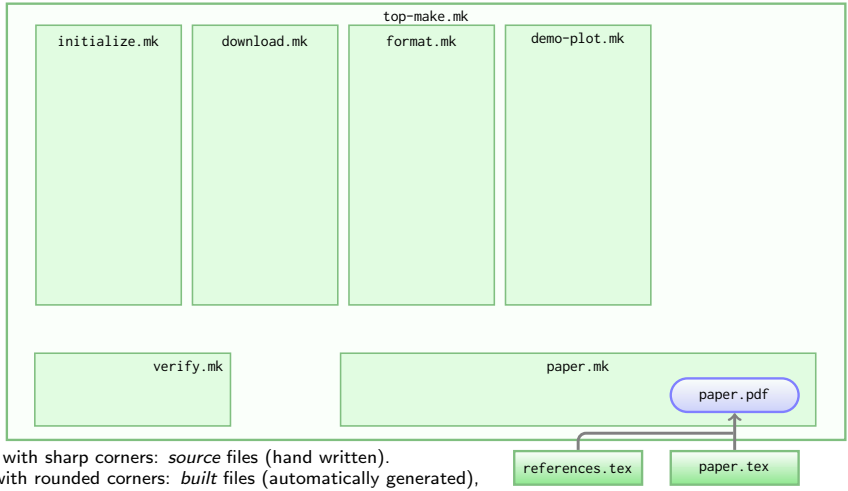
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

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



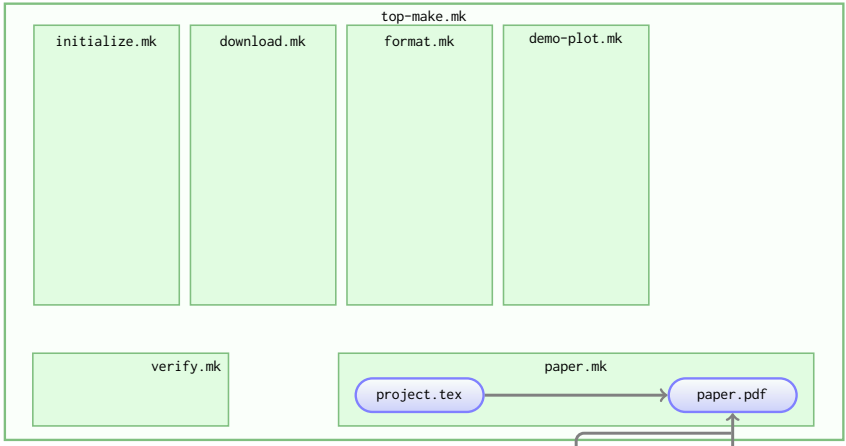
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

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



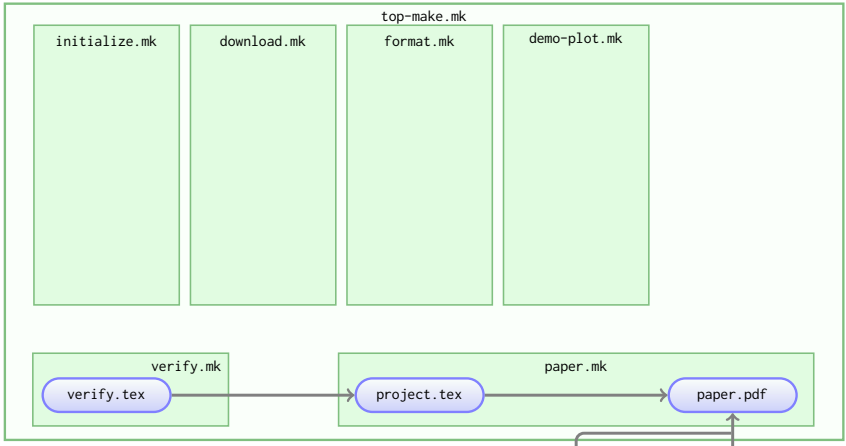
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

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



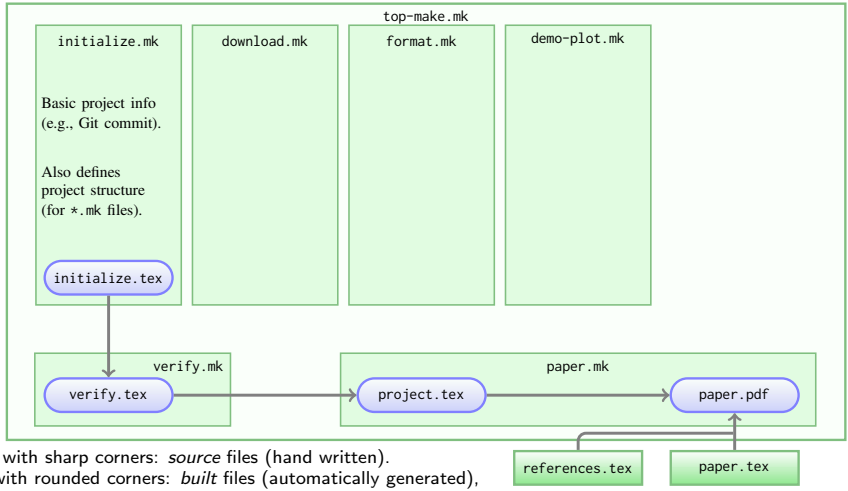
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

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



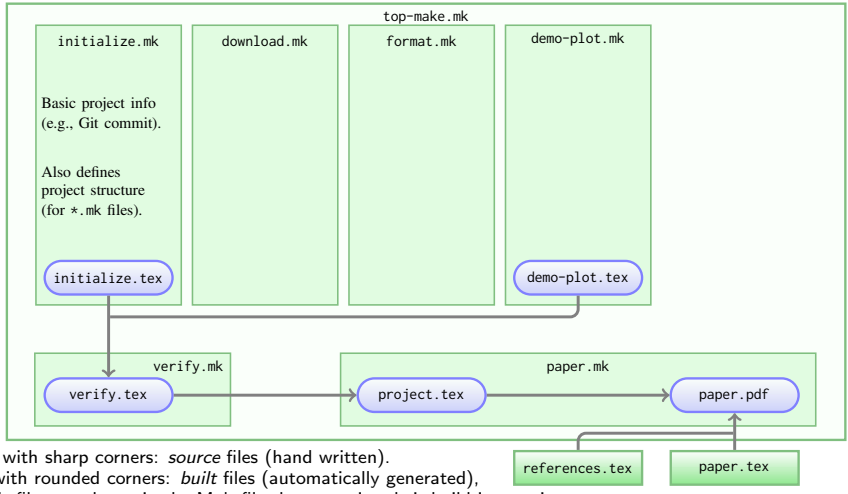
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

Basic project info comes from `initialize.tex`.



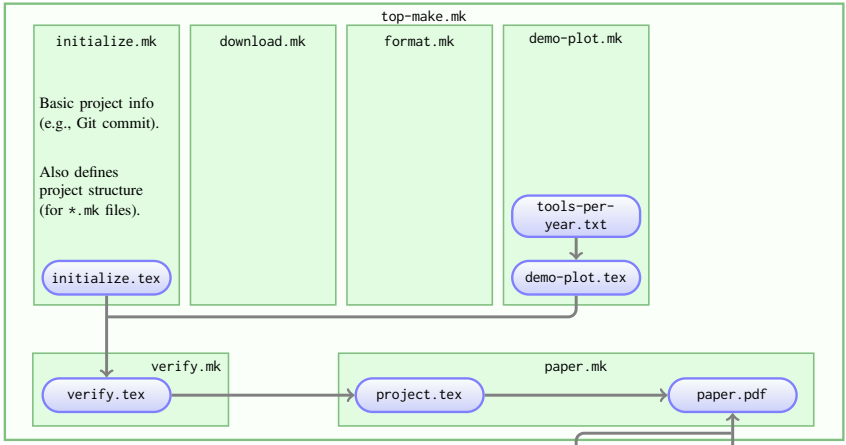
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

The paper includes some information about the plot.



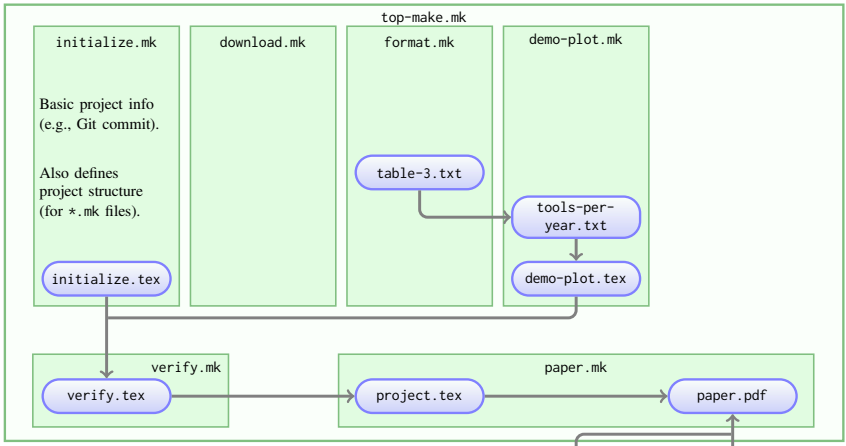
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

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



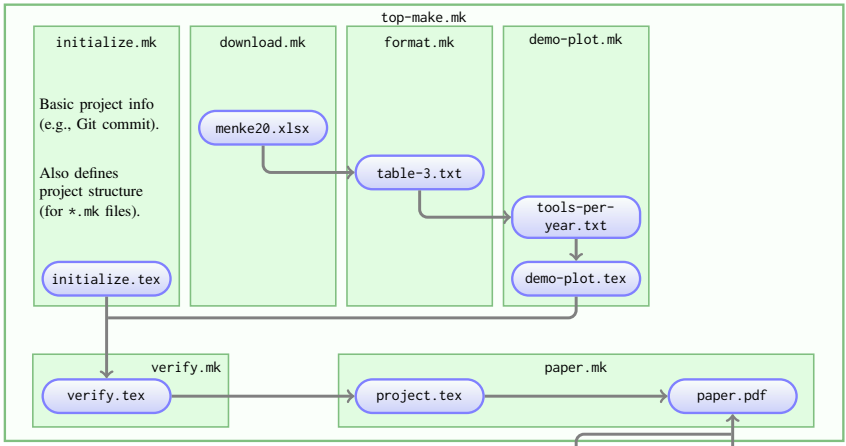
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

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



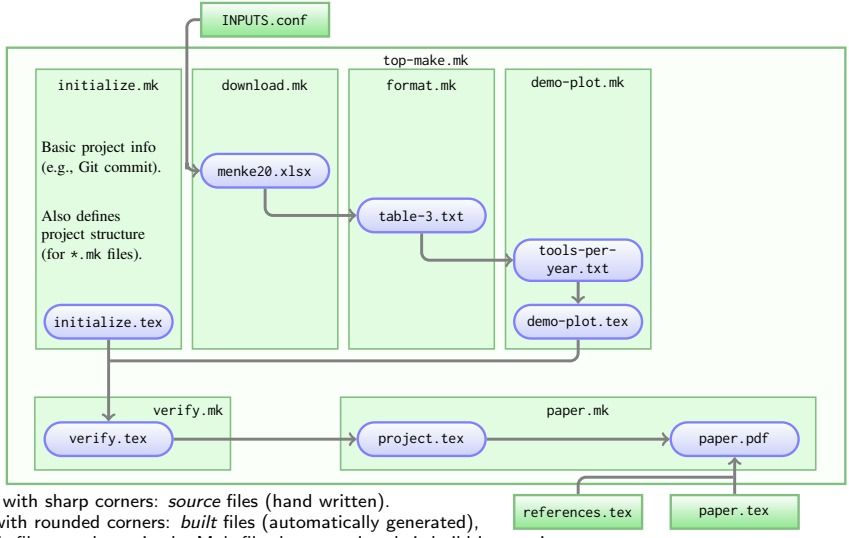
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

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



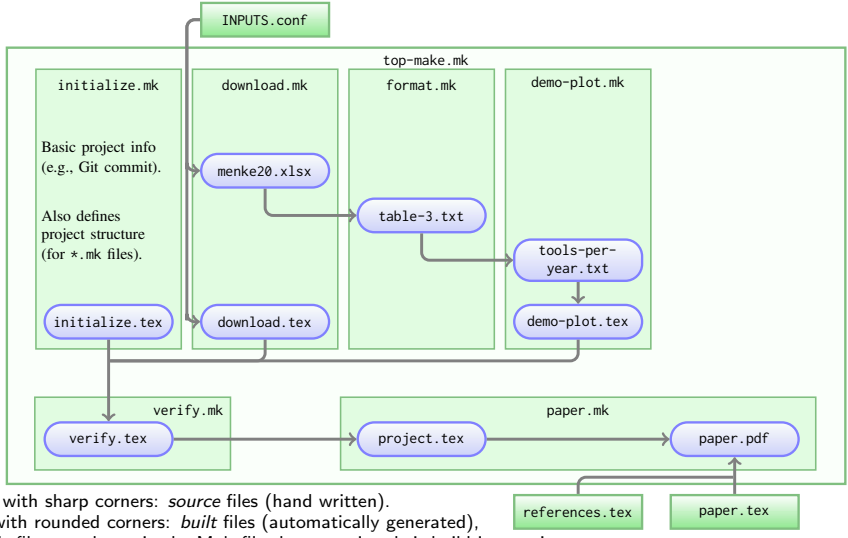
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

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



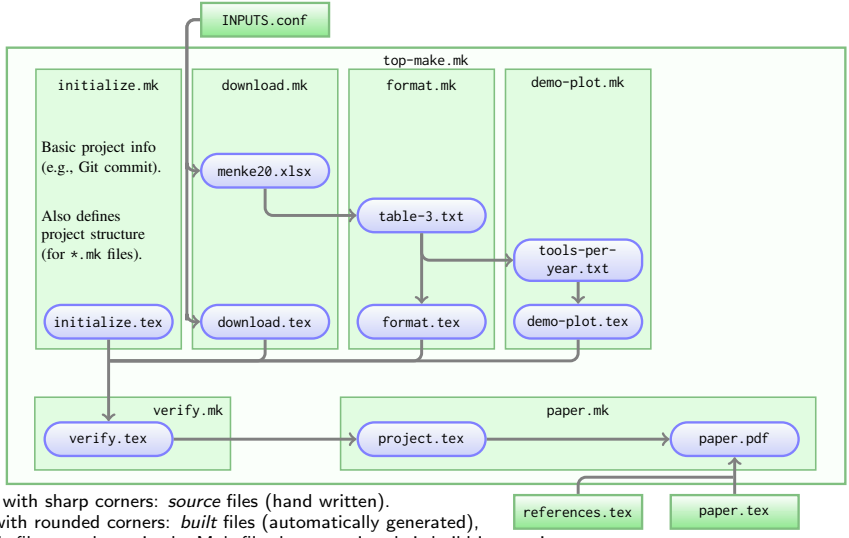
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

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



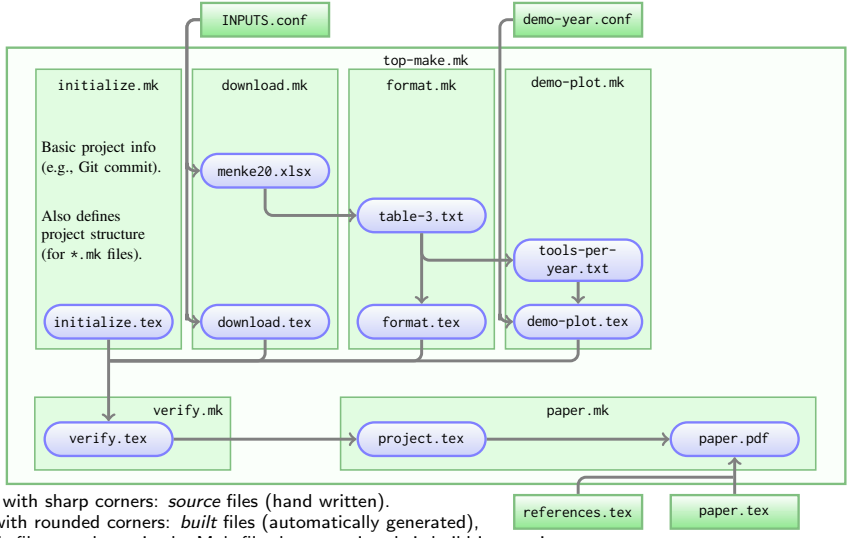
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

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



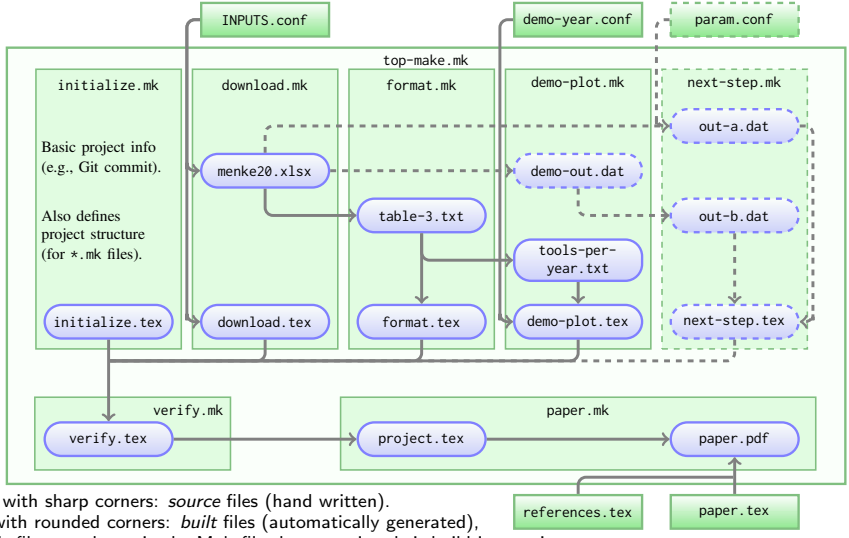
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

We 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`)

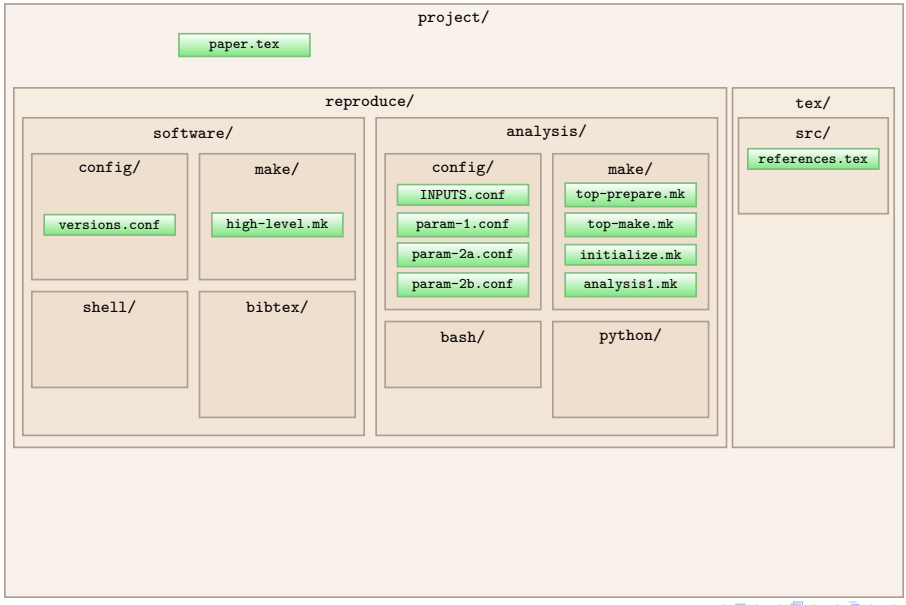
- There are two types of nodes/files:
  - Source nodes (`*.conf` and `paper.tex`) only have an outward link.
  - Built files always have inward *and* (except `paper.pdf`) outward link(s).

- All built files ultimately originate from a `*.conf` file,
  ... and ultimately conclude in `paper.pdf`.

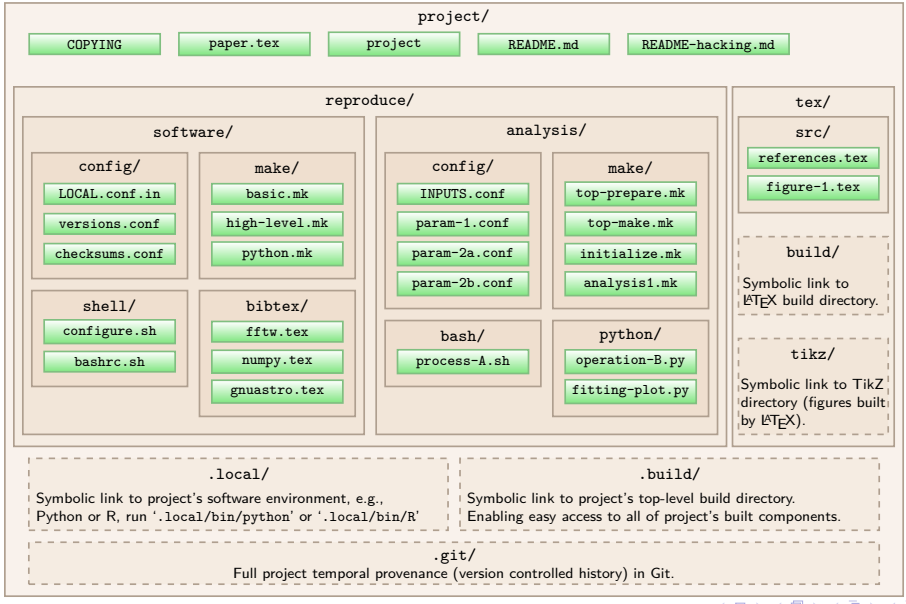
Benefits of using Make

- Make can parallelize the analysis:
  Make knows which steps are indepenent 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.

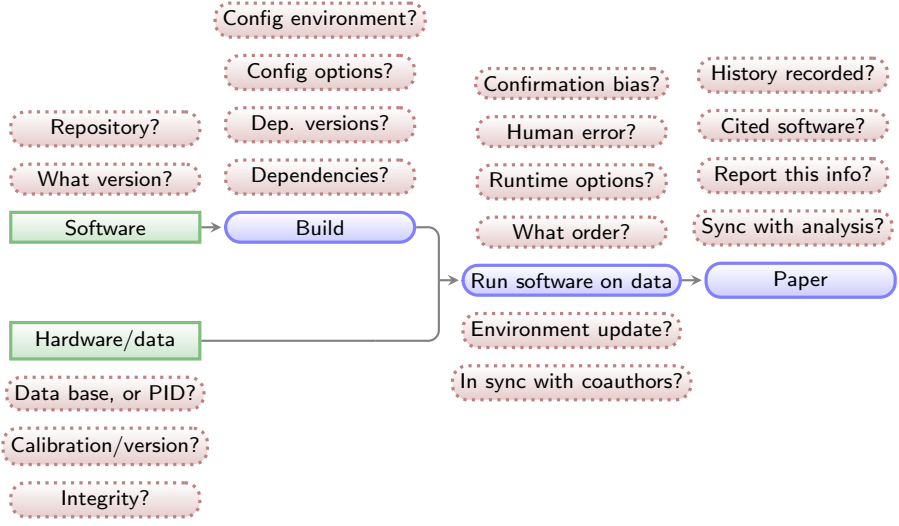
Files organized in directories by context (here are some of the files discussed before)



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

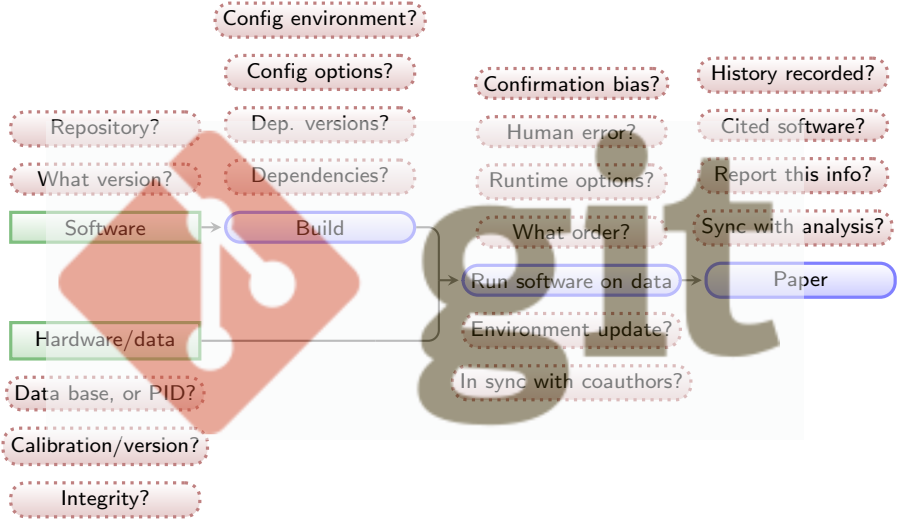


All questions have an answer now (in plain text: human & computer readable/archivable).



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

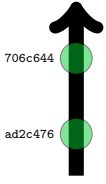
- The project (answers to questions above) will evolve.

New projects branch from Maneage

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

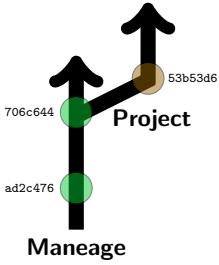
New projects branch from Maneage

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

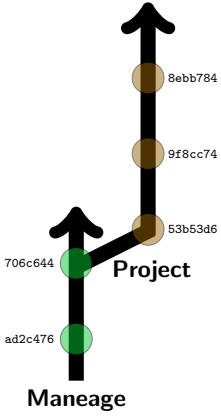


New projects branch from Maneage

- 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.

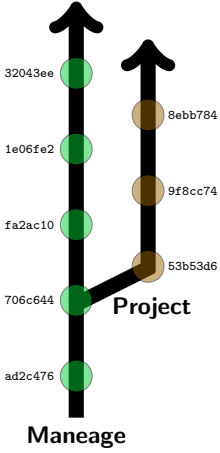


New projects branch from Maneage



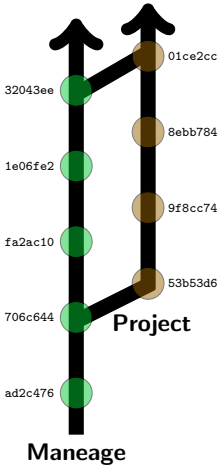
- Each point of project's history is recorded with Git.
- New project: a branch from the template.
  Every commit contains the following:
  - Instructions to download, verify and build software.
  - Instructions to download and verify input data.
  - Instructions to run software on data (do the analysis).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

New projects branch from Maneage



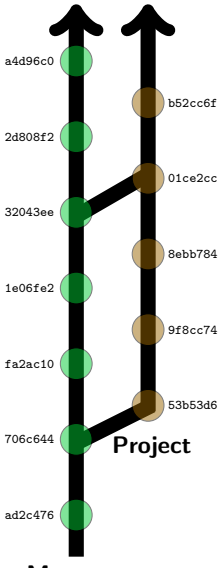
- Each point of project's history is recorded with Git.
- New project: a branch from the template.
  Every commit contains the following:
  - Instructions to download, verify and build software.
  - Instructions to download and verify input data.
  - Instructions to run software on data (do the analysis).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

New projects branch from Maneage



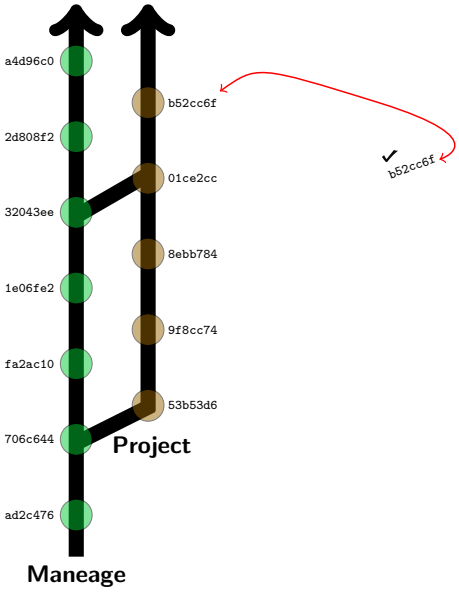
- Each point of project's history is recorded with Git.
- New project: a branch from the template.
  Every commit contains the following:
  - Instructions to download, verify and build software.
  - Instructions to download and verify input data.
  - Instructions to run software on data (do the analysis).
  - 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



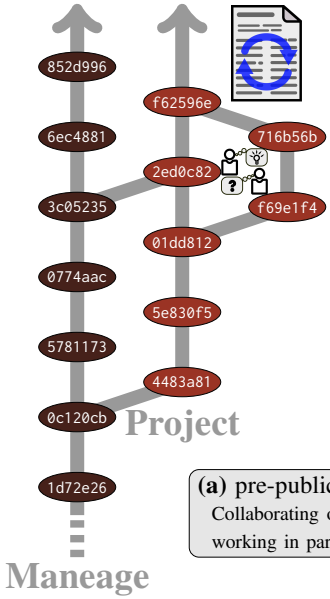
- Each point of project's history is recorded with Git.
- New project: a branch from the template.
  Every commit contains the following:
  - Instructions to download, verify and build software.
  - Instructions to download and verify input data.
  - Instructions to run software on data (do the analysis).
  - 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

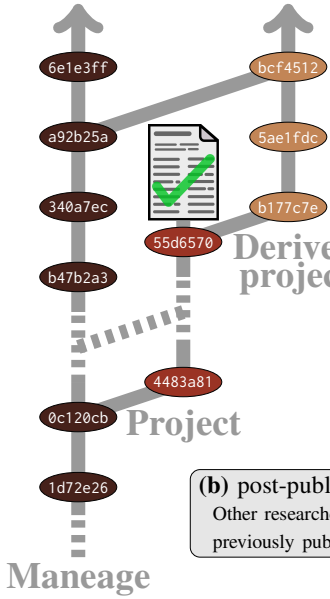


- Each point of project's history is recorded with Git.
- New project: a branch from the template.
  Every commit contains the following:
  - Instructions to download, verify and build software.
  - Instructions to download and verify input data.
  - Instructions to run software on data (do the analysis).
  - 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.



(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 tiny: $\lesssim$1 Mb.
Borkowska & Roukema (2022) 550 kb + Peper, Roukema & Bolejko (2023) 740 kb would fit on a floppy disk

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: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 cosmology papers

- 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: swh:1:rev:a029edd32d5cd41dbdac145189d9b1a08421114e
- Borkowska & Roukema (2022) CQG, 39, 215007 *"Does relativistic cosmology software handle emergent volume evolution?"*
  - `https://arXiv.org/abs/2112.14174`
  - frozen record: `https://zenodo.org/record/5806027`
  - live git: `https://codeberg.org/boud/gevcurvtest`
  - archived git swh:1:rev:d9b47736f81aff9bb8f2f359d9f0331aa923f38d
- Peper, Roukema & Bolejko (2023) MNRAS, 525, 91 *"Detecting cosmic voids via maps of geometric-optics parameters"*
  - `https://arXiv.org/abs/2304.00591`
  - frozen record: `https://zenodo.org/record/8103985`
  - live git: `https://codeberg.org/mpeper/lensing`
  - archived git swh:1:rev:b5dff23ab8ba8c758112d5fd3f737fb6f44cd6fe
- Was it a lot of work? Yes.
- Did we feed our variations and fixes upstream to Maneage? Many, yes.
- Overall feeling: was it worth it? Yes.

How can you try this out?

- 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 post an issue at `https://codeberg.org/boud/gevcurvtest` or, if relevant, post it upstream as 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`
- Interactive discussion: #maneage_community:matrix.org (e.g. `https://matrix.to/#/#maneage_community:matrix.org`)

Conclusion

Maneage (Akhlaghi+2021, CiSE 23, 82 arXiv:2006.03018) 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 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.

Published cosmology papers:

- Peper & Roukema (2021) MNRAS, 505, 1223, *"The role of the elaphrocentre in void galaxy formation"*, `https://doi.org/10.5281/zenodo.4699702`
- Borkowska & Roukema (2022) CQG, 39, 215007 *"Does relativistic cosmology software handle emergent volume evolution?"*, `https://doi.org/10.5281/zenodo.5806027`
- Peper, Roukema & Bolejko (2023) MNRAS, 525, 91 *"Detecting cosmic voids via maps of geometric-optics parameters"*, `https://doi.org/10.5281/zenodo.8103985`

this pdf – full of clickable links: https://cosmo.torun.pl/~boud/Roukema20240311IANCU.pdf