BIG Data, **BIG** responsibility

Introducing *Maneage*: customizable framework for *mana*ging data lin*eage*

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

Most recent slides available in link below (this PDF is built from Git commit 7c49cdd):
https://maneage.org/pdf/slides-intro.pdf

MINISTERIO DE CIENCIA, INNOVACIÓN Y UNIVERSIDADES

SUNDIAL
H2020 Innovative Training Network

This project has received financial support from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No. 721463 to the SUNDIAL ITN network.

Unión Europea
Fondo Europeo de Desarrollo Regional
"Una manera de hacer Europa"

This project received financial support from the European Union's Horizon 2020 research and innovation grant agreement No. 777388 to the RDA EU 4.0 project.

RESEARCH DATA ALLIANCE EUROPE

Universidad de La Laguna

Gobierno de Canarias
Consejería de Economía, Conocimiento y Empleo

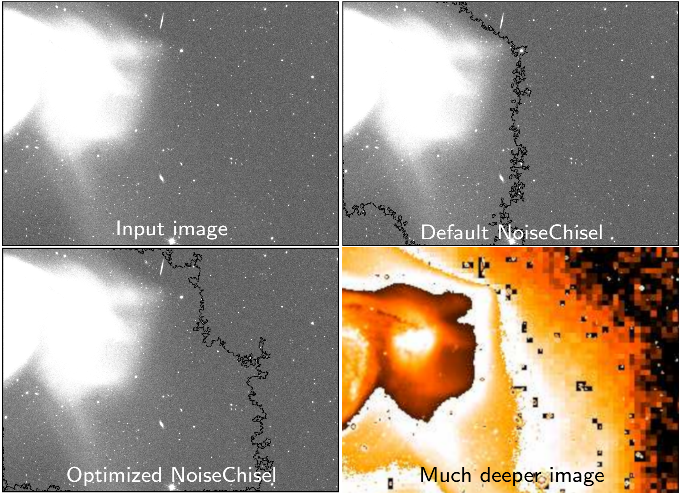
Let's start with this nice image of the Wirlpool galaxy (M51): https://i.redd.it/jfqgpqg0hfk11.jpg

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

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

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

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



Reproducibility crisis in the sciences/astronomy

Snakes on a Spaceship – An Overview of Python in Heliophysics

"...inadequate analysis descriptions and loss of scientific data have made scientific studies difficult or impossible to replicate". From Burrell+2018, (arXiv:1901.00143).

Reproducibility crisis in the sciences/astronomy

Snakes on a Spaceship – An Overview of Python in Heliophysics

"...inadequate analysis descriptions and loss of scientific data have made scientific studies difficult or impossible to replicate". From Burrell+2018, (arXiv:1901.00143).

Perspectives on Reproducibility and Sustainability of Open-Source Scientific Software

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

Reproducibility crisis in the sciences/astronomy

Snakes on a Spaceship – An Overview of Python in Heliophysics

"...inadequate analysis descriptions and loss of scientific data have made scientific studies difficult or impossible to replicate". From Burrell+2018, (arXiv:1901.00143).

Perspectives on Reproducibility and Sustainability of Open-Source Scientific Software

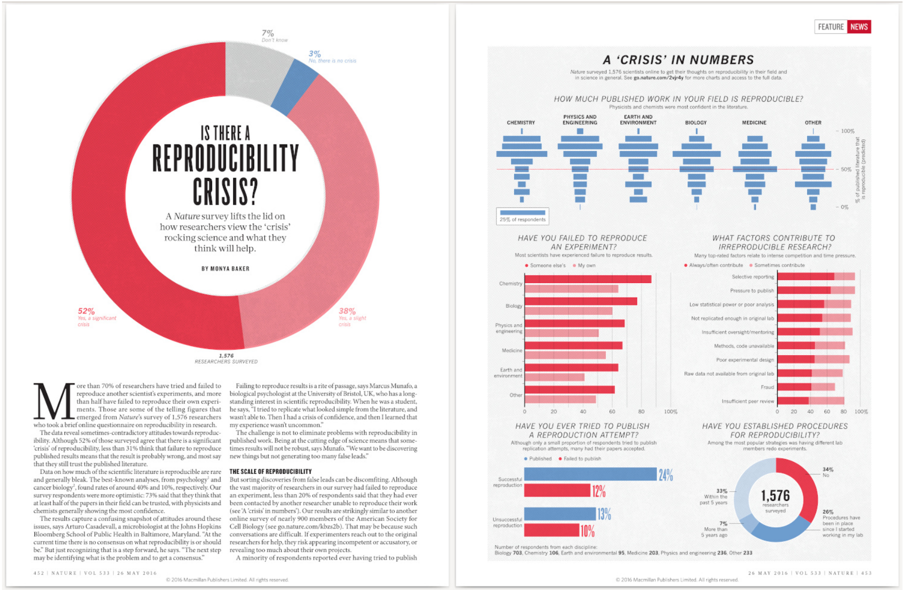
"It is our interest that NASA adopt an open-code policy because without it, reproducibility in computational science is needlessly hampered". From Oishi+2018, (arXiv:1801.08200).

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

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

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

“Reproducibility crisis” in the sciences? (Baker 2016, Nature 533, 452)



IS THERE A REPRODUCIBILITY CRISIS?

A *Nature* survey lifts the lid on how researchers view the ‘crisis’ rocking science and what they think will help.

BY MONYA BAKER

More than 70% of researchers have tried and failed to reproduce another scientist's experiments, and more than half have failed to reproduce their own experiments. Those are some of the telling figures that emerged from *Nature*'s survey of 1,576 researchers who took a brief online questionnaire on reproducibility in research.

The data reveal sometimes-contradictory attitudes towards reproducibility. Although 52% of those surveyed agree that there is a significant ‘crisis’ of reproducibility, less than 31% think that failure to reproduce published results means that the result is probably wrong, and most say that they still trust the published literature.

Data on how much of the scientific literature is reproducible are rare and generally bleak. The best-known analyses, from psychology[1] and cancer biology[2], found rates of around 40% and 10%, respectively. Our survey respondents were more optimistic: 73% said that they think that at least half of the papers in their field can be trusted, with physicists and chemists generally showing the most confidence.

The results capture a confusing snapshot of attitudes around these issues, says Arturo Casadevall, a microbiologist at the Johns Hopkins Bloomberg School of Public Health in Baltimore, Maryland. “At the current time there is no consensus on what reproducibility is or should be.” But just recognizing that is a step forward, he says. “The next step may be identifying what is the problem and to get a consensus.”

Failing to reproduce results is a rite of passage, says Marcus Munafo, a biological psychologist at the University of Bristol, UK, who has a long-standing interest in scientific reproducibility. When he was a student, he says, “I tried to replicate what looked simple from the literature, and wasn't able to. Then I had a crisis of confidence, and then I learned that my experience wasn't uncommon.”

The challenge is not to eliminate problems with reproducibility in published work. Being at the cutting edge of science means that sometimes results will not be robust, says Munafo. “We want to be discovering new things but not generating too many false leads.”

THE SCALE OF REPRODUCIBILITY

But sorting discoveries from false leads can be discomfiting. Although the vast majority of researchers in our survey had failed to reproduce an experiment, less than 20% of respondents said that they had ever been contacted by another researcher unable to reproduce their work (see ‘A ‘crisis’ in numbers’). Our results are strikingly similar to another online survey of nearly 900 members of the American Society for Cell Biology (see go.nature.com/kbzs2b). That may be because such conversations are difficult. If experimenters reach out to the original researchers for help, they risk appearing incompetent or accusatory, or revealing too much about their own projects.

A minority of respondents reported ever having tried to publish

FEATURE NEWS

A ‘CRISIS’ IN NUMBERS

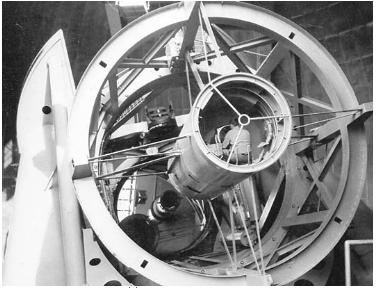
Nature surveyed 1,576 scientists online to get their thoughts on reproducibility in their field and in science in general. See **go.nature.com/2vjr4y** for more charts and access to the full data.

Number of respondents from each discipline:
Biology **703**, Chemistry **106**, Earth and environmental **95**, Medicine **203**, Physics and engineering **236**, Other **233**

 (from the National Academies report in 2019, DOI:10.17226/25303)

Replicability (hardware/statistical)

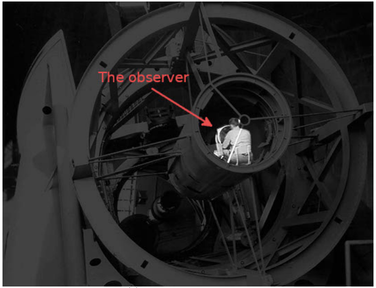
- Involves data collection.
- Inherently includes measurements errors (can never be exactly reproduced).
- Example: Raw telescope image/spectra.
- **NOT DISCUSSED HERE.**



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

Replicability (hardware/statistical)

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



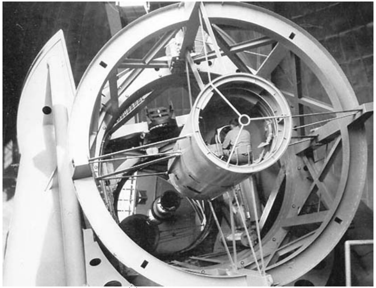
http://slittlefair.staff.shef.ac.uk

Definitions & Clarification

(from the National Academies report in 2019, DOI:10.17226/25303)

Replicability (hardware/statistical)

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



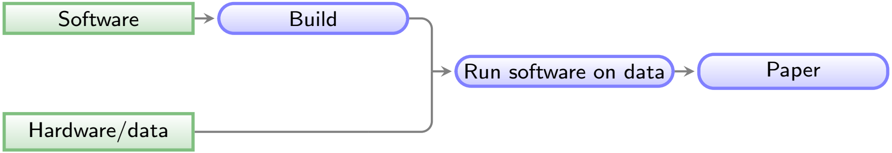
http://slittlefair.staff.shef.ac.uk

Reproducibility (Software/Deterministic)

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

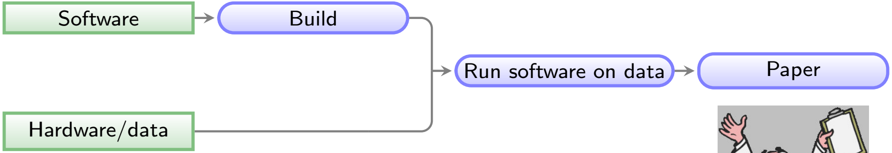
https://tsongas.com

General outline of a project (after data collection)



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

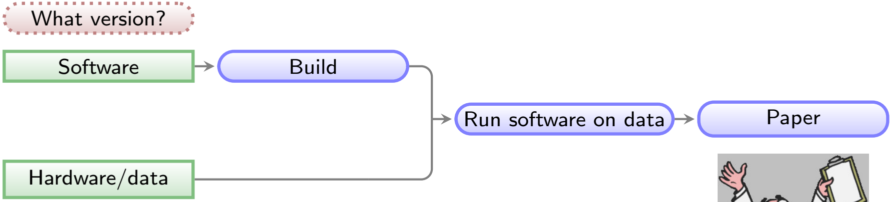
General outline of a project (after data collection)



https://heywhatwhatdidyousay.wordpress.com

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)



https://heywhatwhatdidyousay.wordpress.com

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, 2019/11/20)

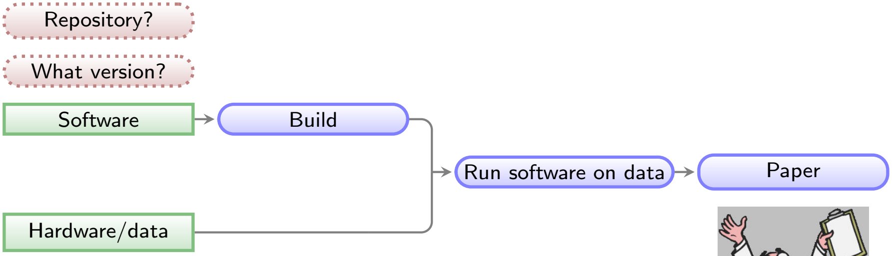
Astropy

Packaging status	
Debian Stable	3.1.2
Debian Testing	3.2.3
Debian Unstable	3.2.3
Deepin	3.0.2
Devuan 3.0 (Beowulf)	3.1.2
Devuan Unstable	3.2.3
Kali Linux Rolling	3.2.3
Parrot	3.2.3
PureOS Amber	3.1.2
PureOS landing	3.2.3
Raspbian Stable	3.1.2
Raspbian Testing	3.2.3
Ubuntu 18.04	3.0
Ubuntu 18.10	3.0.4
Ubuntu 19.04	3.1.1
Ubuntu 19.10	3.2.1
Ubuntu 20.04	3.2.2
Ubuntu 20.04 Proposed	3.2.3

GNU Astronomy Utilities (Gnuastro)

Packaging status	
Debian Oldstable	0.2.33
Debian Stable	0.8
Debian Testing	0.10
Debian Unstable	0.10
Debian Experimental	0.10.39
Deepin	0.5
Devuan 2.0 (ASCII)	0.2.33
Devuan 3.0 (Beowulf)	0.8
Devuan Unstable	0.10
DPorts	0.9
FreeBSD Ports	0.10
Funtoo 1.3	0.3
Funtoo 1.4	0.3
Gentoo	0.3
GNU Guix	0.10
Kali Linux Rolling	0.10
openSUSE Leap 15.1	0.8
openSUSE Leap 15.2	0.8
openSUSE Tumbleweed	0.8
openSUSE Science Tumbleweed	0.10
Pardus	0.2.33
Parrot	0.10
PLD Linux	0.8
PureOS Amber	0.8
PureOS landing	0.10
Raspbian Oldstable	0.2.33
Raspbian Stable	0.8
Raspbian Testing	0.10
Ubuntu 18.04	0.5
Ubuntu 18.10	0.7
Ubuntu 19.04	0.8
Ubuntu 19.10	0.10
Ubuntu 20.04	0.10

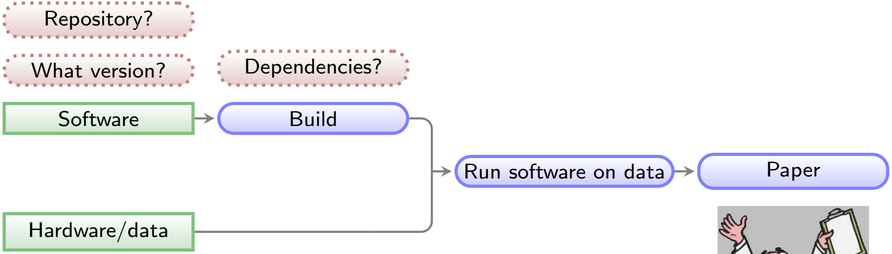
General outline of a project (after data collection)



https://heywhatwhatdidyousay.wordpress.com

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.

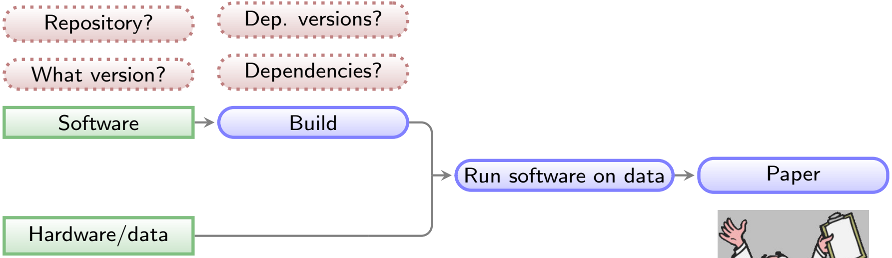
General outline of a project (after data collection)



https://heywhatwhatdidyousay.wordpress.com

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.

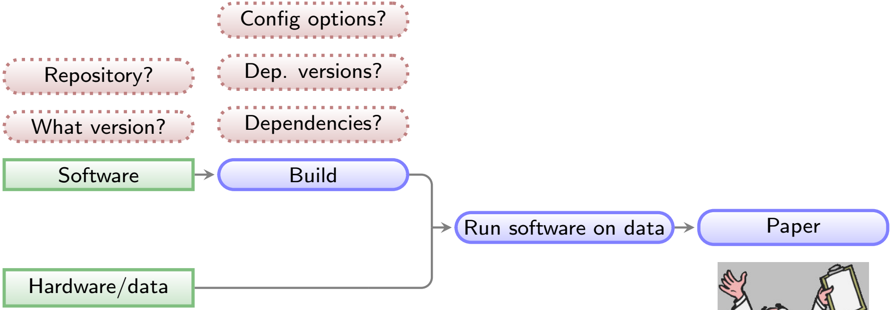
General outline of a project (after data collection)



https://heywhatwhatdidyousay.wordpress.com

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.

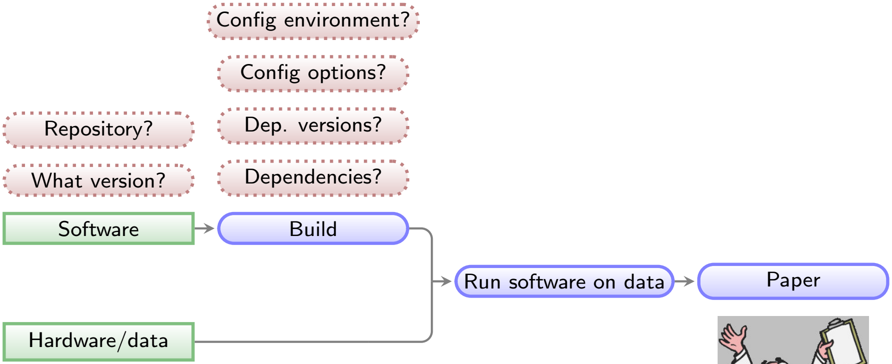
General outline of a project (after data collection)



https://heywhatwhatdidyousay.wordpress.com

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)



https://heywhatwhatdidyousay.wordpress.com

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

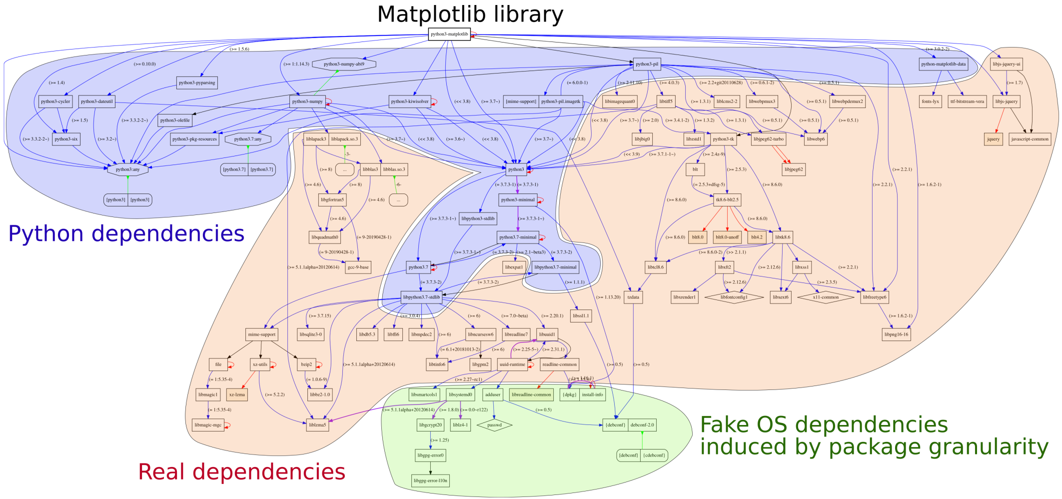


Fig. 1. Transitive dependencies of the software environment required by a simple "`import matplotlib`" command in the Python 3 interpreter.

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

Impact of "Dependency hell" on native building in various hardware (CPU architectures)

debian — **Debian Package Auto-Building**

Buildd status for astropy (sid)

PTS – 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
all	3.2.1-1	Installed	25d 17h 39m	x86-grnet-02		misc	old | all (1)
amd64	3.2.1-1+b1	Installed	2d 10h 45m	x86-ubc-01		misc	old | all (1)
arm64	3.2.1-1+b1	Installed	2d 10h 45m	arm-arm-04		misc	old | all (1)
armel	3.2.1-1+b1	Installed	2d 7h 26m	arnold		misc	old | all (1)
armhf	3.2.1-1+b1	Installed	2d 10h 45m	arm-arm-01		misc	old | all (1)
i386	3.2.1-1+b1	Installed	2d 10h 15m	x86-grnet-01		misc	old | all (1)
mips	3.2.1-1+b1	Installed	2d 9h 21m	mips-manda-01		misc	old | all (1)
mips64el	3.2.1-1+b1	Installed	2d 53m	mipsel-aql-01		misc	old | all (1)
mipsel	3.2.1-1+b1	Installed	2d 5h 38m	mipsel-aql-01		misc	old | all (1)
ppc64el	3.2.1-1+b1	Installed	2d 10h 15m	ppc64el-osuosl-01		misc	old | all (1)
s390x	3.2.1-1+b1	Installed	2d 10h 47m	zandonai		misc	old | all (1)
alpha	3.2.1-1+b1	Installed	2d 36m	imago2		misc	old | all (2)
hppa	3.2.1-1+b1	Installed	2d 1h 4m	phantom		misc	old | all (1)
hurd-i386	3.2.1-1	BD-Uninstallable	25d 18h 34m		uncompiled	misc	old | no log
ia64	3.2.1-1	BD-Uninstallable	25d 18h 32m		uncompiled	misc	old | no log
kfreebsd-amd64	3.2.1-1	BD-Uninstallable	25d 18h 34m		uncompiled	misc	old | no log
kfreebsd-i386	3.2.1-1	BD-Uninstallable	25d 18h 32m		uncompiled	misc	old | no log
m68k	3.2.1-1	BD-Uninstallable	25d 18h 34m		out-of-date	misc	old | no log
powerpc	3.2.1-1	BD-Uninstallable	25d 18h 29m		uncompiled	misc	old | no log
ppc64	3.2.1-1+b1	Installed	2d 10h 7m	kapitsa		misc	old | all (1)
riscv64	3.2.1-1+b1	Installed	2d 5h 23m	rv-aurel32-01		misc	old | all (1)
sh4	3.2.1-1	BD-Uninstallable	25d 18h 29m		out-of-date	misc	old | no log
sparc64	3.2.1-1	BD-Uninstallable	25d 18h 34m		uncompiled	misc	old | no log
x32	3.2.1-1	BD-Uninstallable	25d 18h 26m		out-of-date	misc	old | no log

Astropy depends on Matplotlib

debian — **Debian Package Auto-Building**

Buildd status for gnuastro (sid)

PTS – Tracker – Changelog – Bugs – packages.d.o – Source

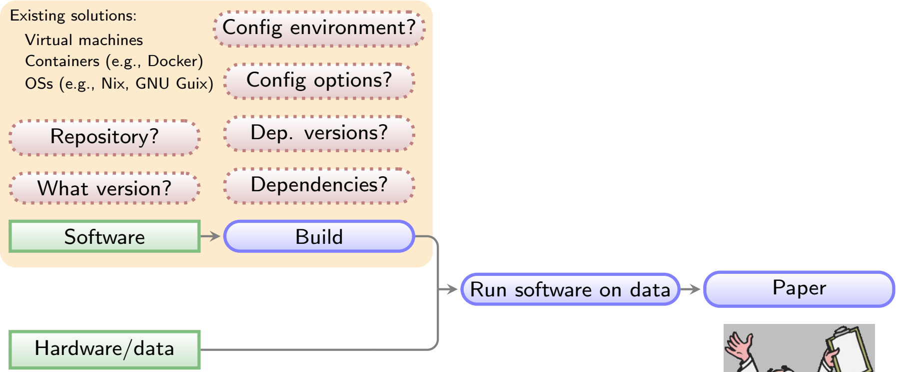
Package(s): gnuastro Suite: sid Go

Compact mode Co-maintainers

Architecture	Version	Status	For	Buildd	State	Section	Logs
all is not present in the architecture list set by the maintainer							
amd64	0.10-1	Installed	1d 2h 56m	x86-ubc-01		misc	old | all (1)
arm64	0.10-1	Installed	1d 2h 33m	arm-conova-01		misc	old | all (1)
armel	0.10-1	Installed	1d 2h 32m	arnold		misc	old | all (1)
armhf	0.10-1	Installed	1d 2h 31m	arm-ubc-06		misc	old | all (1)
i386	0.10-1	Installed	1d 2h 55m	x86-csail-01		misc	old | all (1)
mips	0.10-1	Installed	1d 2h 31m	mips-sil-01		misc	old | all (1)
mips64el	0.10-1	Installed	1d 32m	mipsel-sil-01		misc	old | all (1)
mipsel	0.10-1	Installed	1d 2h 33m	mipsel-manda-03		misc	old | all (1)
ppc64el	0.10-1	Installed	1d 2h 58m	ppc64el-osuosl-01		misc	old | all (1)
s390x	0.10-1	Installed	1d 2h 58m	zani		misc	old | all (1)
alpha	0.10-1	Installed	6h 57m	tsunami		misc	old | all (3)
hppa	0.10-1	Installed	1d 2h	phantom		misc	old | all (1)
hurd-i386	0.10-1	Installed	1d 2h 25m	ironforge		misc	old | all (1)
ia64	0.10-1	Installed	18h 3m	iridium		misc	old | all (2)
kfreebsd-amd64	0.10-1	Installed	18h 30m	kamp		misc	old | all (1)
kfreebsd-i386	0.10-1	Installed	18h 36m	kamp		misc	old | all (1)
m68k	0.10-1	Installed	18h 36m	vs92		misc	old | all (4)
powerpc	0.10-1	Installed	1d 2h 42m	kapitsa2		misc	old | all (1)
ppc64	0.10-1	Installed	18h 5m	kapitsa		misc	old | all (3)
riscv64	0.10-1	Installed	1d 2h 22m	rv-mullvad-01		misc	old | all (1)
sh4	0.10-1	Installed	17h 38m	sh4-gandi-01		misc	old | all (4)
sparc64	0.10-1	Installed	19h 2m	sompek2		misc	old | all (4)
x32	0.10-1	Installed	18h 30m	x32-do-01		misc	old | all (3)

GNU Astronomy Utilities doesn't.

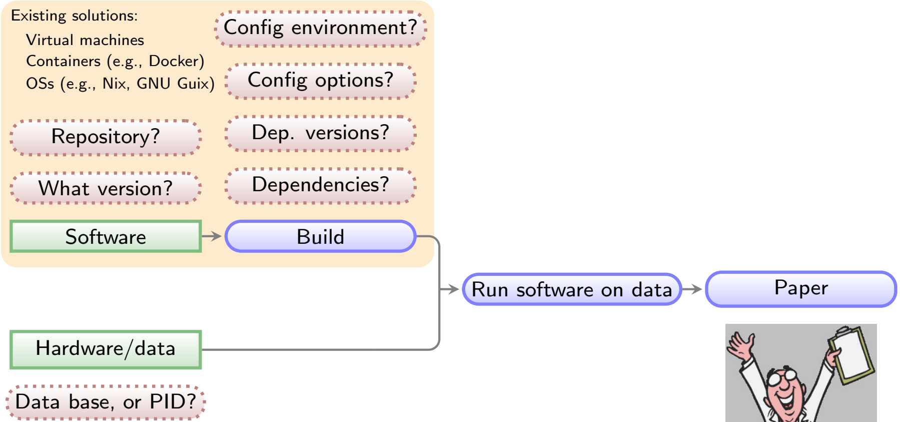
General outline of a project (after data collection)



https://heywhatwhatdidyousay.wordpress.com

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)



https://heywhatwhatdidyousay.wordpress.com

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)

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

Config environment?

Config options?

Repository?

Dep. versions?

What version?

Dependencies?

Software → Build → Run software on data → Paper

Hardware/data → Run software on data

Data base, or PID?

Calibration/version?

https://heywhatwhatdidyousay.wordpress.com

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)

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

Config environment?

Config options?

Repository?

Dep. versions?

What version?

Dependencies?

Software → Build

Run software on data → Paper

Hardware/data

Data base, or PID?

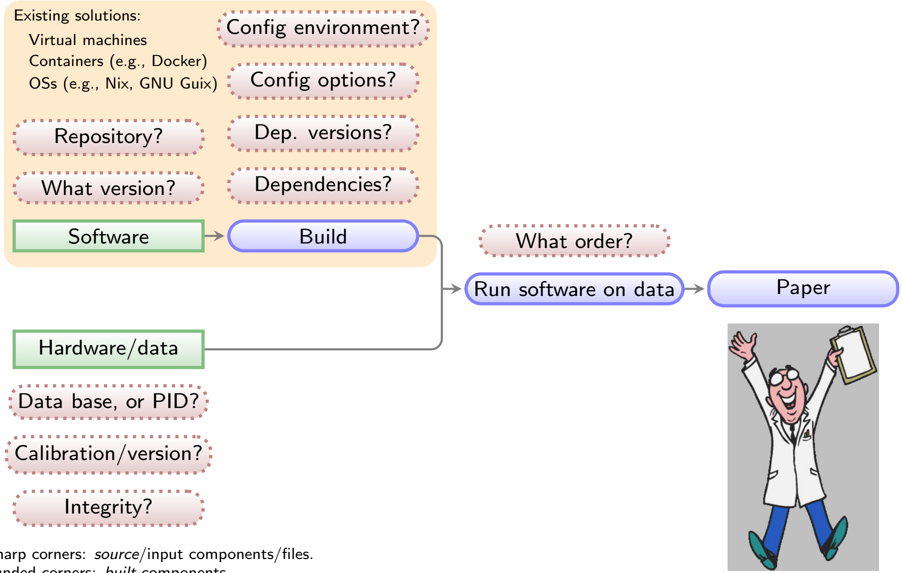
Calibration/version?

Integrity?

https://heywhatwhatdidyousay.wordpress.com

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.

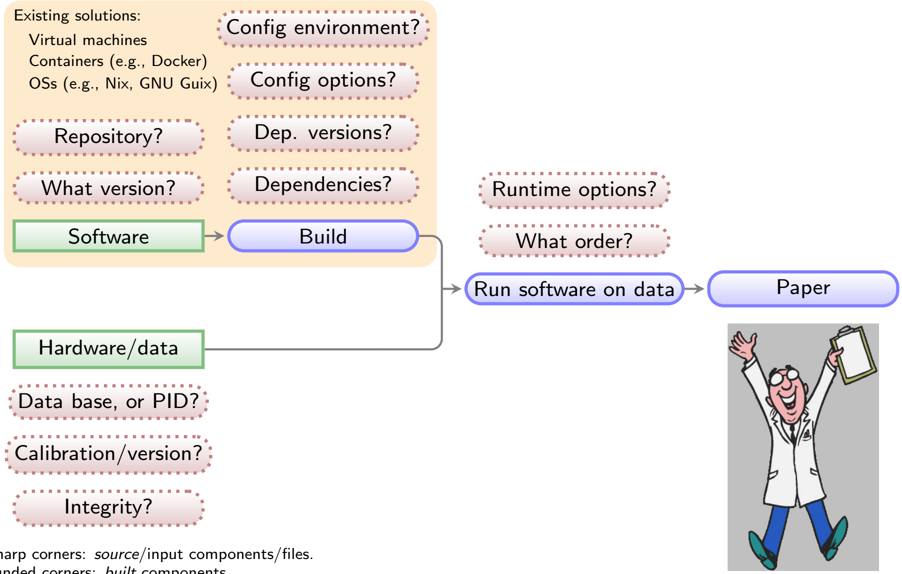
General outline of a project (after data collection)



https://heywhatwhatdidyousay.wordpress.com

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.

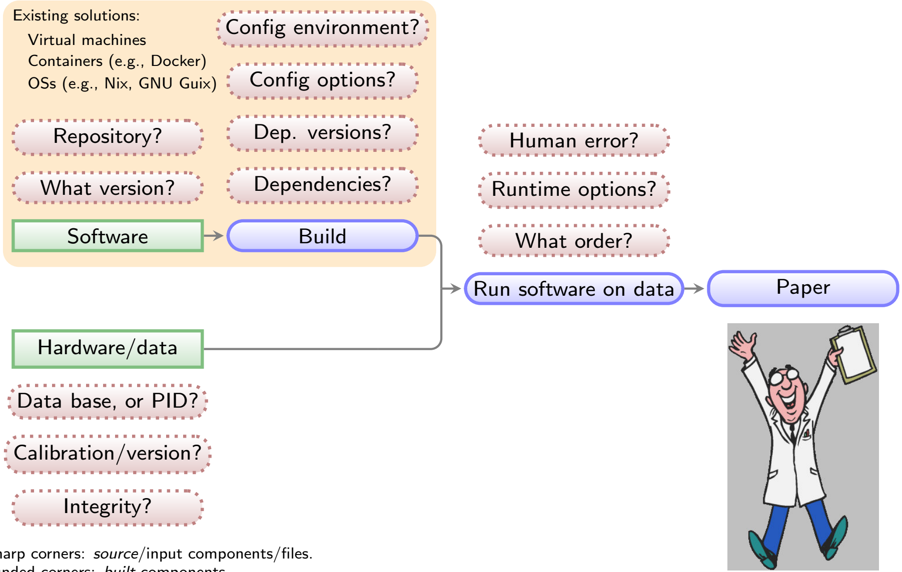
General outline of a project (after data collection)



https://heywhatwhatdidyousay.wordpress.com

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

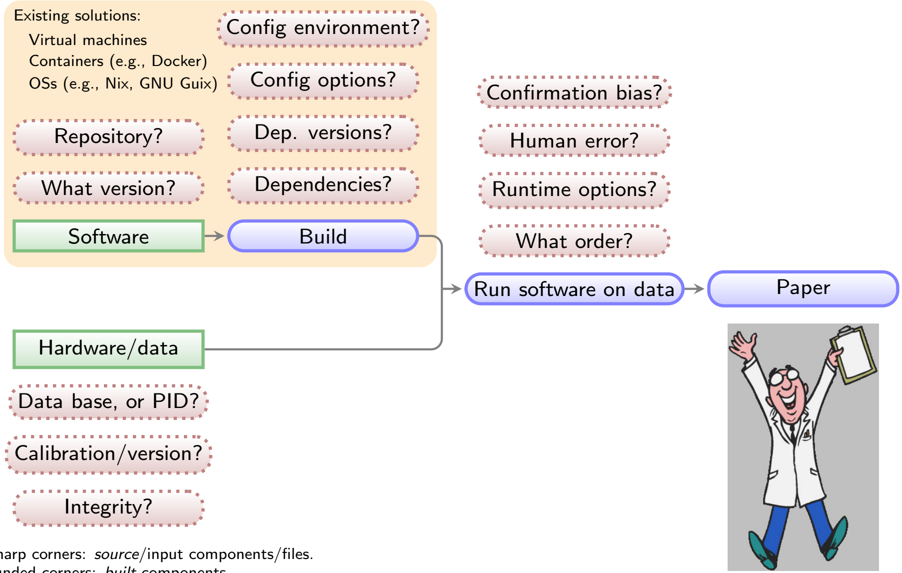
General outline of a project (after data collection)



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

https://heywhatwhatdidyousay.wordpress.com

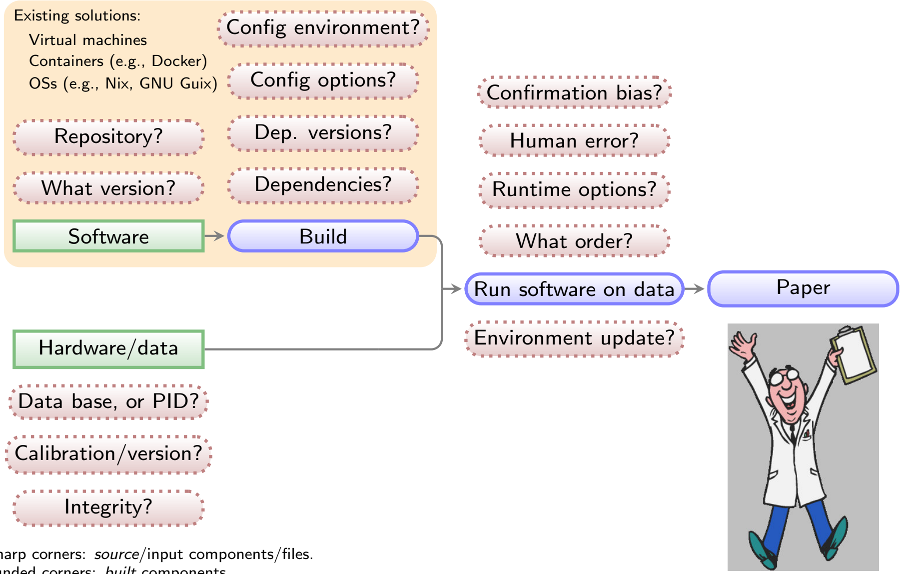
General outline of a project (after data collection)



https://heywhatwhatdidyousay.wordpress.com

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.

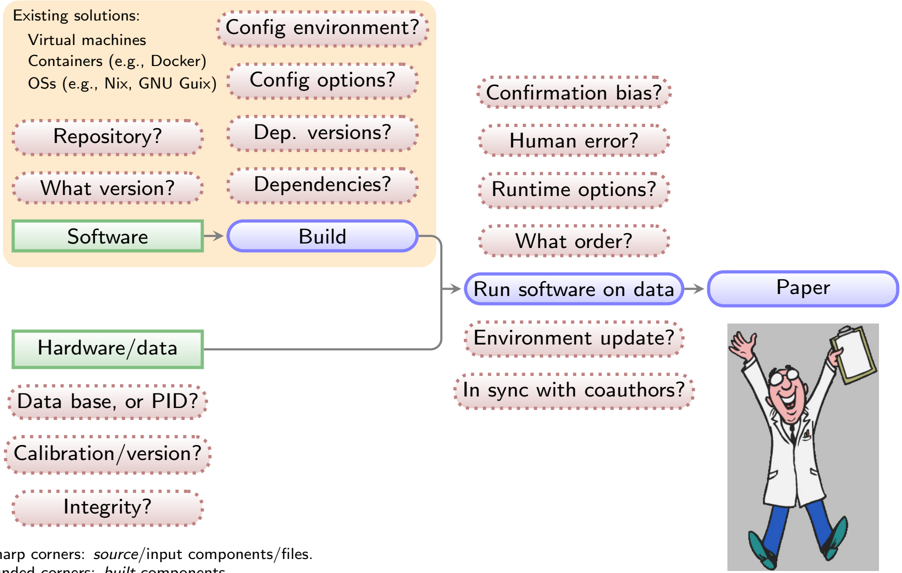
General outline of a project (after data collection)



https://heywhatwhatdidyousay.wordpress.com

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.

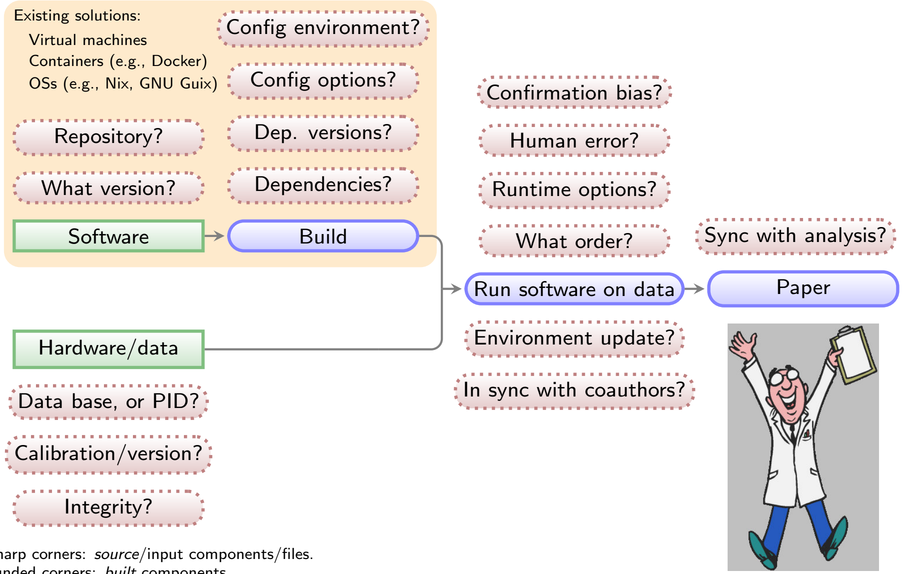
General outline of a project (after data collection)



https://heywhatwhatdidyousay.wordpress.com

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.

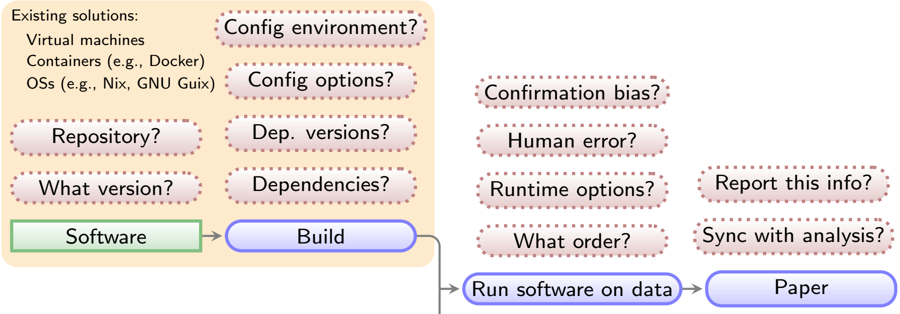
General outline of a project (after data collection)



https://heywhatwhatdidyousay.wordpress.com

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.

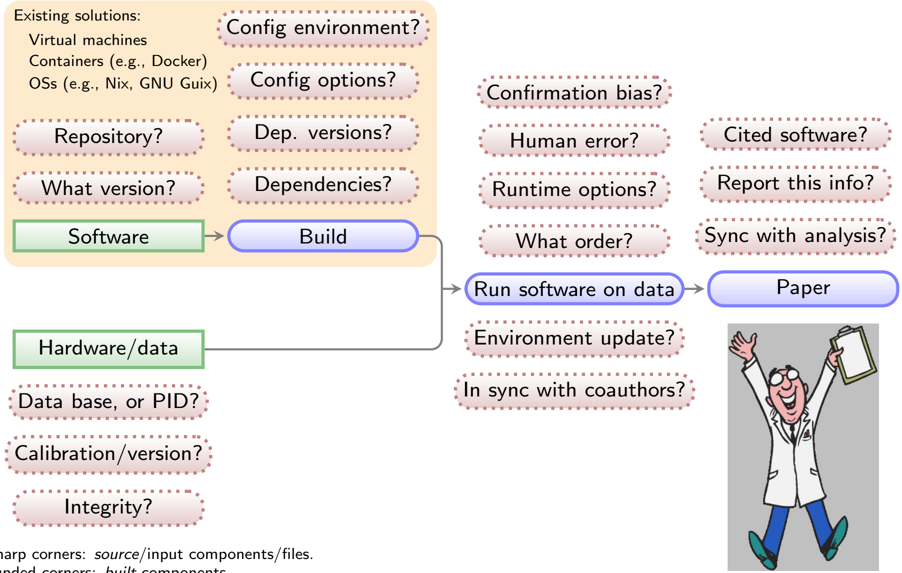
General outline of a project (after data collection)



https://heywhatwhatdidyousay.wordpress.com

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)



https://heywhatwhatdidyousay.wordpress.com

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)

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

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

Software → Build

Hardware/data

- Data base, or PID?
- Calibration/version?
- Integrity?

Build, Hardware/data → Run software on data → Paper

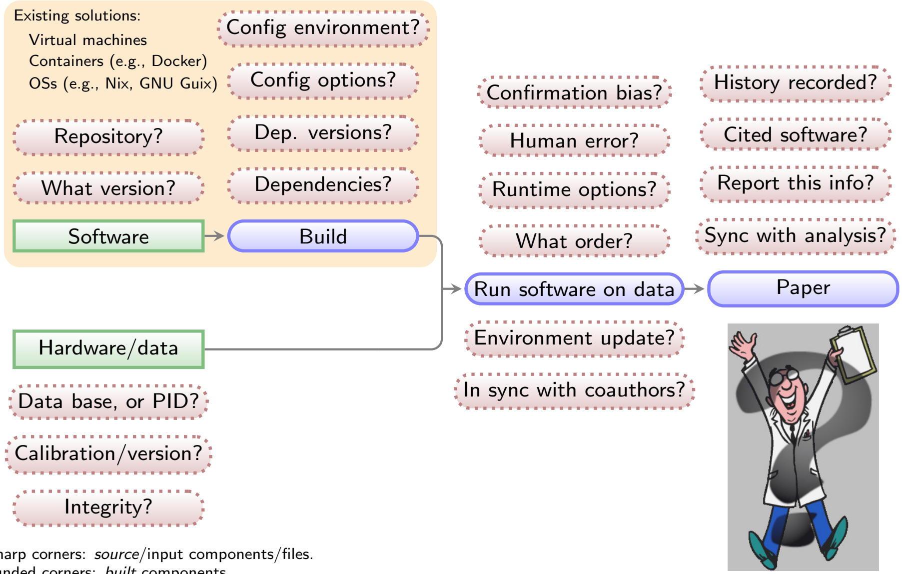
- Confirmation bias?
- Human error?
- Runtime options?
- What order?
- Environment update?
- In sync with coauthors?

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

https://heywhatwhatdidyousay.wordpress.com

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)



https://heywhatwhatdidyousay.wordpress.com
http://pngimages.net

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.

Science is a tricky business

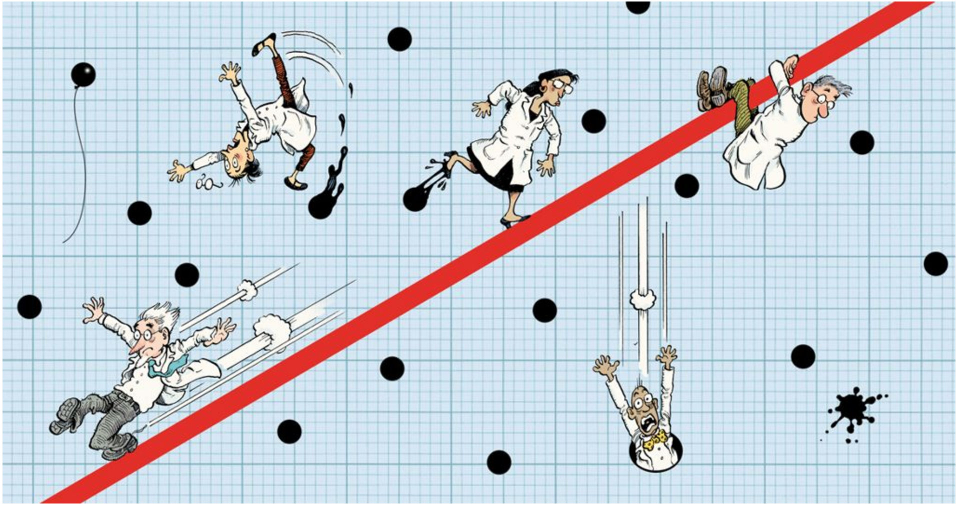


Image from nature.com ("Five ways to fix statistics", Nov 2017)

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

Five ways to fix statistics, Nature, 551, Nov 2017.

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

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

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

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

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

Principles behind proposed solution

Basic/simple principle:

Science is defined by its METHOD, not its result.

Principles behind proposed solution

Basic/simple principle:

Science is defined by its METHOD, not its result.

- **Complete/self-contained:**
  - Only dependency should be POSIX tools (discards Conda or Jupyter which need Python).

Principles behind proposed solution

Basic/simple principle:

Science is defined by its METHOD, not its result.

- **Complete/self-contained:**
  - Only dependency should be POSIX tools (discards Conda or Jupyter which need Python).
  - Must not require root permissions (discards tools like Docker or Nix/Guix).

Principles behind proposed solution

Basic/simple principle:

Science is defined by its METHOD, not its result.

- **Complete/self-contained:**
  - Only dependency should be POSIX tools (discards Conda or Jupyter which need Python).
  - Must not require root permissions (discards tools like Docker or Nix/Guix).
  - Should be non-interactive or runnable in batch (user interaction is an incompleteness).

Principles behind proposed solution

Basic/simple principle:

Science is defined by its METHOD, not its result.

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

Principles behind proposed solution

Basic/simple principle:

Science is defined by its METHOD, not its result.

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

Principles behind proposed solution

Basic/simple principle:

Science is defined by its METHOD, not its result.

- **Complete/self-contained:**
  - Only dependency should be POSIX tools (discards Conda or Jupyter which need Python).
  - Must not require root permissions (discards tools like Docker or Nix/Guix).
  - Should be non-interactive or runnable in batch (user interaction is an incompleteness).
  - Should be usable without internet connection.
- **Modularity:** Parts of the project should be re-usable in other projects.
- **Plain text:** Project's source should be in plain-text (binary formats need special software)
  - This includes high-level analysis.
  - It is easily publishable (very low volume of $\times$100KB), archivable, and parse-able.
  - Version control (e.g., with Git) can track project's history.

Principles behind proposed solution

Basic/simple principle:

Science is defined by its METHOD, not its result.

- **Complete/self-contained:**
  - Only dependency should be POSIX tools (discards Conda or Jupyter which need Python).
  - Must not require root permissions (discards tools like Docker or Nix/Guix).
  - Should be non-interactive or runnable in batch (user interaction is an incompleteness).
  - Should be usable without internet connection.
- **Modularity:** Parts of the project should be re-usable in other projects.
- **Plain text:** Project's source should be in plain-text (binary formats need special software)
  - This includes high-level analysis.
  - It is easily publishable (very low volume of $\times$100KB), archivable, and parse-able.
  - Version control (e.g., with Git) can track project's history.
- **Minimal complexity:** Occum's rasor: "Never posit pluralities without necessity".
  - Avoiding the fashionable tool of the day: tomorrow another tool will take its place!
  - Easier learning curve, also doesn't create a generational gap.
  - Is compatible and extensible.

Principles behind proposed solution

Basic/simple principle:

Science is defined by its METHOD, not its result.

- **Complete/self-contained:**
  - Only dependency should be POSIX tools (discards Conda or Jupyter which need Python).
  - Must not require root permissions (discards tools like Docker or Nix/Guix).
  - Should be non-interactive or runnable in batch (user interaction is an incompleteness).
  - Should be usable without internet connection.
- **Modularity:** Parts of the project should be re-usable in other projects.
- **Plain text:** Project's source should be in plain-text (binary formats need special software)
  - This includes high-level analysis.
  - It is easily publishable (very low volume of $\times$100KB), archivable, and parse-able.
  - Version control (e.g., with Git) can track project's history.
- **Minimal complexity:** Occum's rasor: "Never posit pluralities without necessity".
  - Avoiding the fashionable tool of the day: tomorrow another tool will take its place!
  - Easier learning curve, also doesn't create a generational gap.
  - Is compatible and extensible.
- **Verifable inputs and outputs:** Inputs and Outputs must be automatically verified.

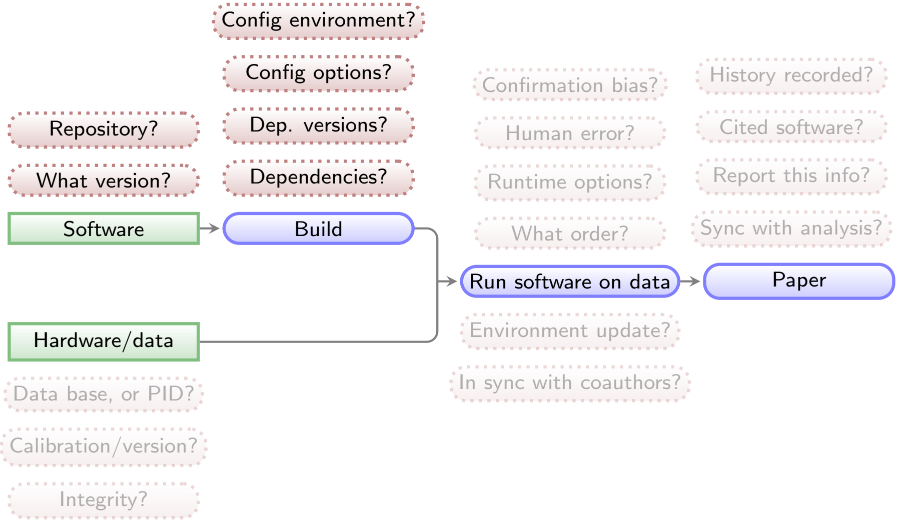
Principles behind proposed solution

Basic/simple principle:

Science is defined by its METHOD, not its result.

- **Complete/self-contained:**
  - Only dependency should be POSIX tools (discards Conda or Jupyter which need Python).
  - Must not require root permissions (discards tools like Docker or Nix/Guix).
  - Should be non-interactive or runnable in batch (user interaction is an incompleteness).
  - Should be usable without internet connection.
- **Modularity:** Parts of the project should be re-usable in other projects.
- **Plain text:** Project's source should be in plain-text (binary formats need special software)
  - This includes high-level analysis.
  - It is easily publishable (very low volume of $\times$100KB), archivable, and parse-able.
  - Version control (e.g., with Git) can track project's history.
- **Minimal complexity:** Occum's rasor: "Never posit pluralities without necessity".
  - Avoiding the fashionable tool of the day: tomorrow another tool will take its place!
  - Easier learning curve, also doesn't create a generational gap.
  - Is compatible and extensible.
- **Verifable inputs and outputs:** Inputs and Outputs must be automatically verified.
- **Free and open source software:** Free software is essential: non-free software is not configurable, not distributable, and dependent on non-free provider (which may discontinue it in N years).

General outline of a project (after data collection)



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

Predefined/exact software tools

Reproducibility & software

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

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

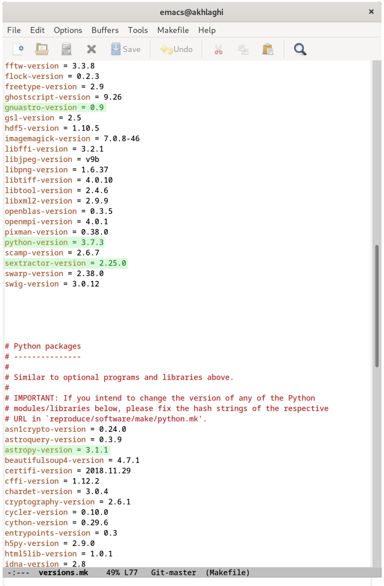
```
fftw-version = 3.3.8
flock-version = 0.2.3
freetype-version = 2.9
ghostscript-version = 9.26
gnuastro-version = 0.9
gsl-version = 2.5
hdf5-version = 1.10.5
imagemagick-version = 7.0.8-46
libffi-version = 3.2.1
libjpeg-version = v9b
libpng-version = 1.6.37
libtiff-version = 4.0.10
libtool-version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



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

Predefined/exact software tools

Reproducibility & software

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

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



Controlled environment and build instructions

emacs@akhlaghi

File Edit Options Buffers Tools Makefile Help

```
include reproduce/software/config/installation/texlive.mk
include reproduce/software/config/installation/versions.mk

lockdir = $(BDIR)/locks
tdir    = $(BDIR)/software/tarballs
ddir    = $(BDIR)/software/build-tmp
idir    = $(BDIR)/software/installed
ibdir   = $(BDIR)/software/installed/bin
ildir   = $(BDIR)/software/installed/lib
dtexdir = $(shell pwd)/reproduce/software/bibtex
itidir  = $(BDIR)/software/installed/version-info/tex
ictdir  = $(BDIR)/software/installed/version-info/cite
ipydir  = $(BDIR)/software/installed/version-info/python
ibidir  = $(BDIR)/software/installed/version-info/proglib

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

# 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 `$(ibdir)/env - NAME=VALUE $(ibdir)/bash' and set all
#       the parameters defined bellow 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.
.ONESHELL:
.SHELLFLAGS              := --noprofile --norc -ec
export CCACHE_DISABLE    := 1
export PATH              := $(ibdir)
export SHELL             := $(ibdir)/bash
export CPPFLAGS          := -I$(idir)/include
export PKG_CONFIG_PATH   := $(ildir)/pkgconfig
export PKG_CONFIG_LIBDIR := $(ildir)/pkgconfig
export LD_RUN_PATH       := $(ildir):$(il64dir)
export LD_LIBRARY_PATH   := $(ildir):$(il64dir)
export LDFLAGS           := $(rpath_command) -L$(ildir)


# 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) = xyes ]; 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 -lssl -lcrypto -lz" VERBOSE=1  \
	&& make install \
	&& cd .. \
	&& rm -rf cmake-$(cmake-version) \
	&& echo "CMake $(cmake-version)" > $@

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

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

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

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

Controlled environment and build instructions

emacs@akhlaghi

File Edit Options Buffers Tools Makefile Help

```
include reproduce/software/config/installation/texlive.mk
include reproduce/software/config/installation/versions.mk

lockdir = $(BDIR)/locks
tdir    = $(BDIR)/software/tarballs
ddir    = $(BDIR)/software/build-tmp
idir    = $(BDIR)/software/installed
ibdir   = $(BDIR)/software/installed/bin
ildir   = $(BDIR)/software/installed/lib
dtexdir = $(shell pwd)/reproduce/software/bibtex
itidir  = $(BDIR)/software/installed/version-info/tex
ictdir  = $(BDIR)/software/installed/version-info/cite
ipydir  = $(BDIR)/software/installed/version-info/python
ibidir  = $(BDIR)/software/installed/version-info/proglib

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

# 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 `$(ibdir)/env - NAME=VALUE $(ibdir)/bash' and set all
#       the parameters defined bellow 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.
.ONESHELL:
.SHELLFLAGS                 := --noprofile --norc -ec
export CCACHE_DISABLE       := 1
export PATH                 := $(ibdir)
export SHELL                := $(ibdir)/bash
export CPPFLAGS             := -I$(idir)/include
export PKG_CONFIG_PATH      := $(ildir)/pkgconfig
export PKG_CONFIG_LIBDIR    := $(ildir)/pkgconfig
export LD_RUN_PATH          := $(ildir):$(il64dir)
export LD_LIBRARY_PATH      := $(ildir):$(il64dir)
export LDFLAGS              := $(rpath_command) -L$(ildir)


# 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) = xyes ]; 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 -lssl -lcrypto -lz" VERBOSE=1  \
        && make install \
        && cd .. \
        && rm -rf cmake-$(cmake-version) \
        && echo "CMake $(cmake-version)" > $@

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

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

$(ibidir)/imagemagick: $(tdir)/imagemagick-$(imagemagick-version).tar.xz \
                       $(ibidir)/libjpeg \
                       $(ibidir)/libtiff \
                       $(ibidir)/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

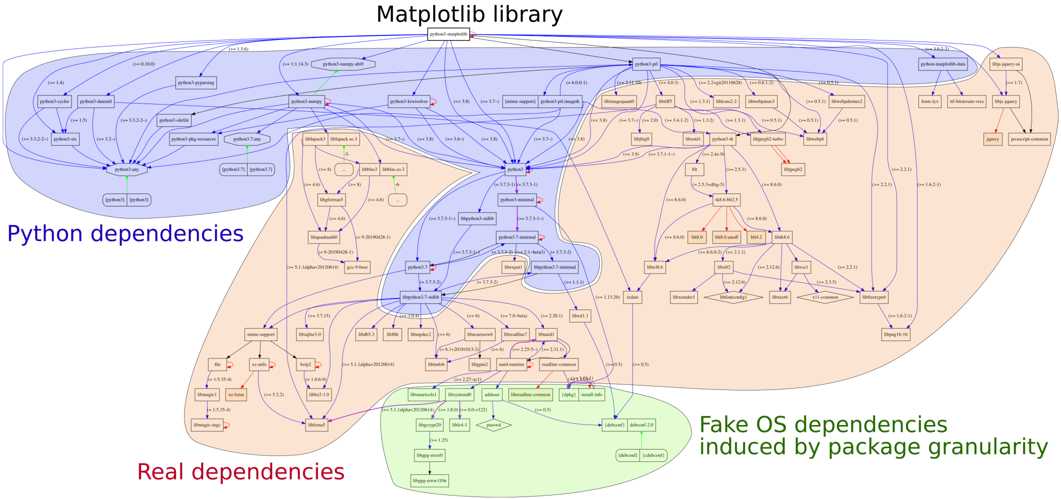


Fig. 1. Transitive dependencies of the software environment required by a simple "`import matplotlib`" command in the Python 3 interpreter.

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
```

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, LaTeX, etc.

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

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

- Dependencies of different projects will not conflict.

- Everything in plain text (human & computer readable/archivable).

https://natemowry2.wordpress.com

Software citation automatically generated in paper (including Astropy)

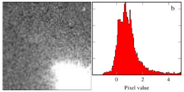


Figure 2: (a) An example image of the Wide-Field Planetary Camera 2, on board the Hubble Space Telescope from 1993 to 2009. This is one of the sample images from the FITS standard webpage, kept as examples for this file format. (b) Histogram of pixel values in (a).

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 (including Astropy)

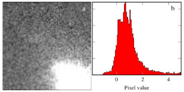


Figure 2: (a) An example image of the Wide-Field Planetary Camera 2, on board the Hubble Space Telescope from 1993 to 2009. This is one of the sample images from the FITS standard webpage, kept as examples for this file format. (b) Histogram of pixel values in (a).

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 (only GNU Astronomy Utilities)

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: Bzip2 1.0.6, CFITSIO 3.47, CMake 3.14.2, cURL 7.63.0, Discoteq flock 0.2.3, File 5.36, Git 2.22.0, GNU Astronomy Utilities 0.9.170-bffc (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, Libbsd 0.9.1, Libgit2 0.28.2, Libjpeg v9b, Libtiff 4.0.10, Lzip 1.20, Metastore (forked) 1.1.2-23-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 LaTeX source of the paper was compiled to make the PDF using the following packages: biber 2.12, biblatex 3.12, caption 2018-10-05, charter 2016-06-24, courier 2016-06-24, csquotes 5.2d, datetime 2.60, ec 1.0, environ 0.3, etoolbox 2.5f, extsizes 1.4a, fancyhdr 3.10, fmtcount 3.05, fontaxes 1.0d, footmisc 5.5b, fp 2.1d, helvetic 2016-06-24, lineno 4.41, logreq 1.0, newtx 1.554, pgf 3.1.2, pgfplots 1.16, preprint 2011, setspace 6.7a, sttools 2.0, tcolorbox 4.20, tex 3.14159265, texgyre 2.501, times 2016-06-24, titlesec 2.10.2, trimspaces 1.1, txfonts 2016-06-24, ulem 2016-06-24, xcolor 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.

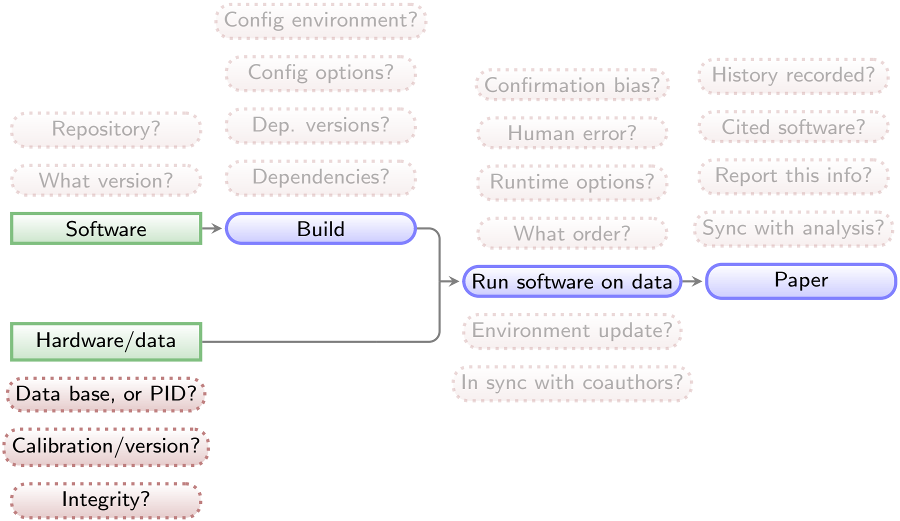
Software citation automatically generated in paper (only GNU Astronomy Utilities)

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: Bzip2 1.0.6, CFITSIO 3.47, CMake 3.14.2, cURL 7.63.0, Discoteq flock 0.2.3, File 5.36, Git 2.22.0, GNU Astronomy Utilities 0.9.170-bffc (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, Libbsd 0.9.1, Libgit2 0.28.2, Libjpeg v9b, Libtiff 4.0.10, Lzip 1.20, Metastore (forked) 1.1.2-23-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 LaTeX source of the paper was compiled to make the PDF using the following packages: biber 2.12, biblatex 3.12, caption 2018-10-05, charter 2016-06-24, courier 2016-06-24, csquotes 5.2d, datetime 2.60, ec 1.0, environ 0.3, etoolbox 2.5f, extsizes 1.4a, fancyhdr 3.10, fmtcount 3.05, fontaxes 1.0d, footmisc 5.5b, fp 2.1d, helvetic 2016-06-24, lineno 4.41, logreq 1.0, newtx 1.554, pgf 3.1.2, pgfplots 1.16, preprint 2011, setspace 6.7a, sttools 2.0, tcolorbox 4.20, tex 3.14159265, texgyre 2.501, times 2016-06-24, titlesec 2.10.2, trimspaces 1.1, txfonts 2016-06-24, ulem 2016-06-24, xcolor 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.
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.


M51SDSSRURL   = https://dr12.sdss.org/sas/dr12/boss/photoObj/frames/301/3716/6
M51SDSSRIMAGE = frame-r-003716-6-0117.fits.bz2
M51SDSSRMD5   = 965da8bd861e94a9701521a11b2d80aa
M51SDSSRSIZE  = 2.8M

XDFF755WURL   = http://archive.stsci.edu/pub/hlsp/xdf
XDFF755WIMAGE = hlsp_xdf_hst_acswfc-60mas_hudf_f775w_v1_sci.fits
XDFF755WMD5   = 81408ed0949bd3a93c4bfe7e229472e6
XDFF775WSIZE  = 106M

UVUDFSEGURL   = https://asd.gsfc.nasa.gov/UVUDF
UVUDFSEGIMAGE = segmentation_map_rafelski_2015.fits.gz
UVUDFSEGMD5   = 29d5b3e5311b77512baf27db6ad0e11b
UVUDFSEGSIZE  = 1.3M
```

Input data source and integrity is documented and checked

Stored information about each input file:

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

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

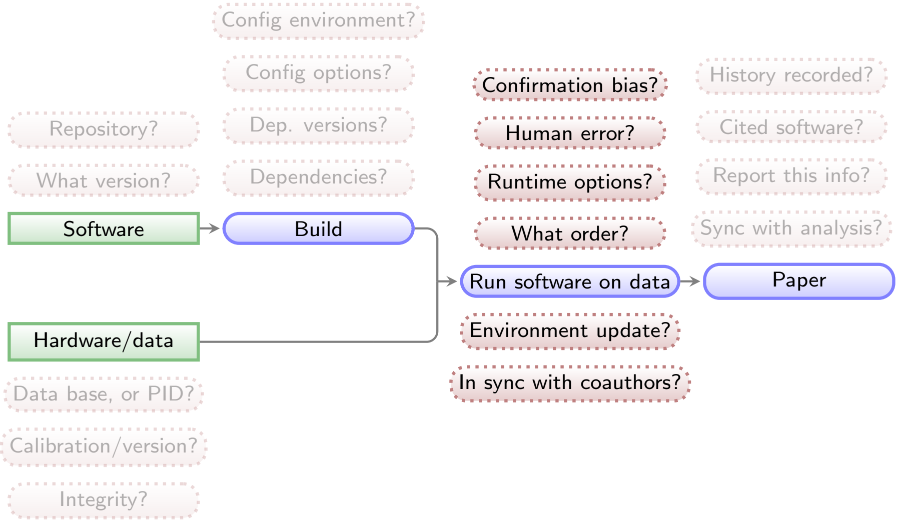
MD5-sums are checked to make sure the download was done properly or the file is the same (hasn't changed on the server/source).

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


```
# 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.


M51SDSSRURL   = https://dr12.sdss.org/sas/dr12/boss/photoObj/frames/301/3716/6
M51SDSSRIMAGE = frame-r-003716-6-0117.fits.bz2
M51SDSSRMD5   = 965da8bd861e94a9701521a11b2d80aa
M51SDSSRSIZE  = 2.8M

XDFF755WURL   = http://archive.stsci.edu/pub/hlsp/xdf
XDFF755WIMAGE = hlsp_xdf_hst_acswfc-60mas_hudf_f775w_v1_sci.fits
XDFF755WMD5   = 81408ed0949bd3a93c4bfe7e229472e6
XDFF775WSIZE  = 106M

UVUDFSEGURL   = https://asd.gsfc.nasa.gov/UVUDF
UVUDFSEGIMAGE = segmentation_map_rafelski_2015.fits.gz
UVUDFSEGMD5   = 29d5b3e5311b77512baf27db6ad0e11b
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):

- 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: Maneage is managed through a Makefile

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

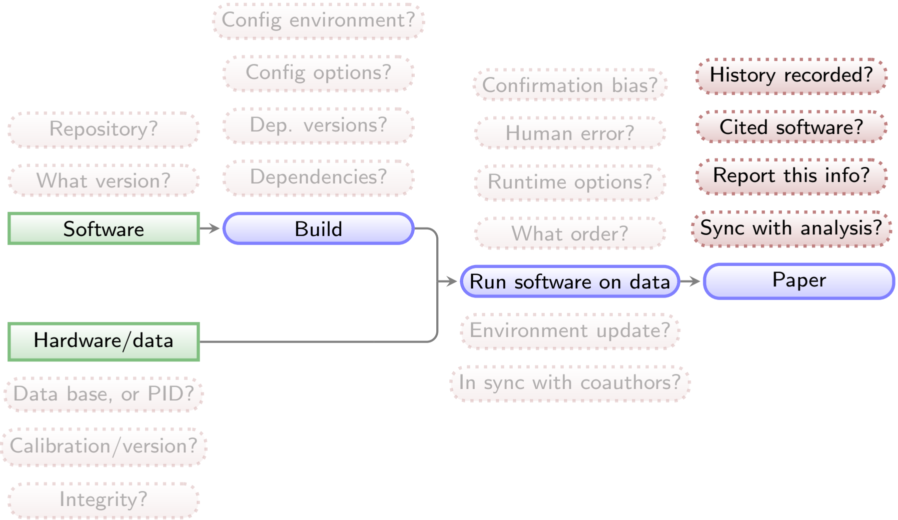
- Unlike a script which always starts from the top, a Makefile starts from the end and steps that don't change will be left untouched (not remade).
- A single *rule* can manage any number of files.
- Make can identify independent steps internally and do them in parallel.
- Make was designed for complex 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 great and free 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 great and free 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.
Shown here is a portion of the NoiseChisel paper and its LaTeX source (arXiv:1505.01664).

```
\begin{equation}
  \label{tSNeq}
  \mathrm{S/N}_T=\frac{NF-NS_a}{\sqrt{NF+N\sigma_S^2}}
  =\frac{\sqrt{N}(F-S_a)}{\sqrt{F+\sigma_S^2}}.
\end{equation}

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

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

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

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 $\mathrm{S/N}_T$ 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 $\mathrm{S/N}_T$ 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 LaTeX macros. They are thus updated automatically on any change.
Shown here is a portion of the NoiseChisel paper and its LaTeX source (arXiv:1505.01664).

```
\begin{equation}
  \label{tSNeq}
  \mathrm{S/N}_T=\frac{NF-NS_a}{\sqrt{NF+N\sigma_S^2}}
  =\frac{\sqrt{N}(F-S_a)}{\sqrt{F+\sigma_S^2}}.
\end{equation}

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

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

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

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 $\mathrm{S/N}_T$ 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 $\mathrm{S/N}_T$ distribution of detections in R_s provides a very ro-

Analysis step results/values concatenated into a single file.

All LaTeX macros come from a single file.

emacs@akhlaghi

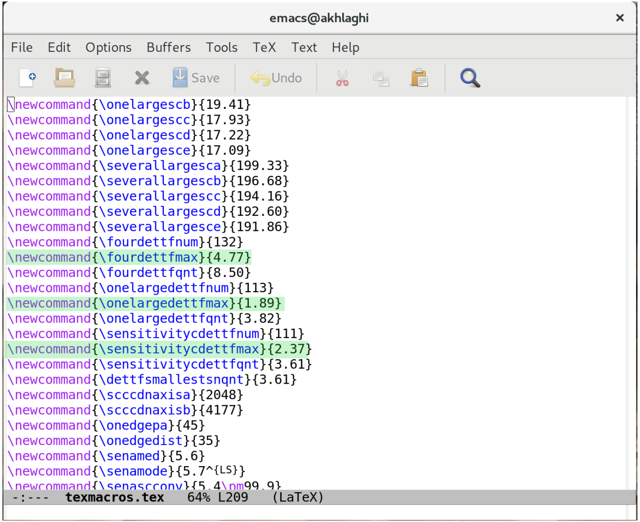
File Edit Options Buffers Tools TeX Text Help

Save Undo

```
\newcommand{\onelargescb}{19.41}
\newcommand{\onelargescc}{17.93}
\newcommand{\onelargescd}{17.22}
\newcommand{\onelargesce}{17.09}
\newcommand{\severallargesca}{199.33}
\newcommand{\severallargescb}{196.68}
\newcommand{\severallargescc}{194.16}
\newcommand{\severallargescd}{192.60}
\newcommand{\severallargesce}{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\pm99.9}
-:---  texmacros.tex    64% L209    (LaTeX)
```

Analysis step results/values concatenated into a single file.

All LaTeX macros come from a single file.



emacs@akhlaghi

File Edit Options Buffers Tools TeX Text Help

Save Undo

```
\newcommand{\onelargescb}{19.41}
\newcommand{\onelargescc}{17.93}
\newcommand{\onelargescd}{17.22}
\newcommand{\onelargesce}{17.09}
\newcommand{\severallargesca}{199.33}
\newcommand{\severallargescb}{196.68}
\newcommand{\severallargescc}{194.16}
\newcommand{\severallargescd}{192.60}
\newcommand{\severallargesce}{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\pm99.9}
-:---  texmacros.tex    64% L209    (LaTeX)
```

Analysis results stored as LaTeX macros

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

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

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

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

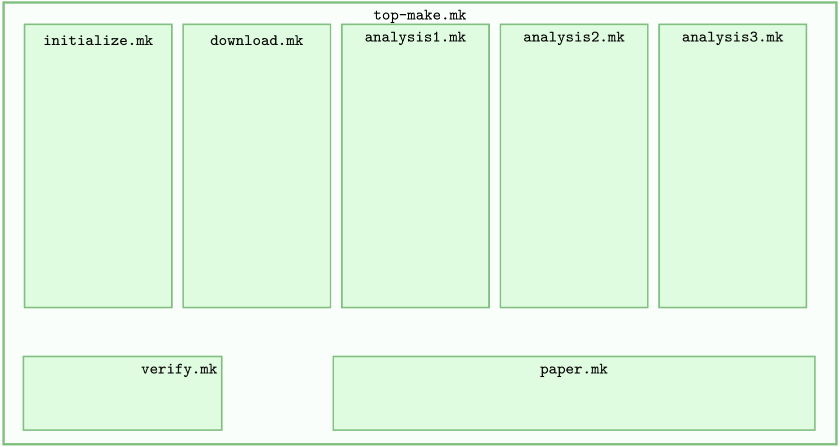
Analysis results stored as LaTeX macros

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
```

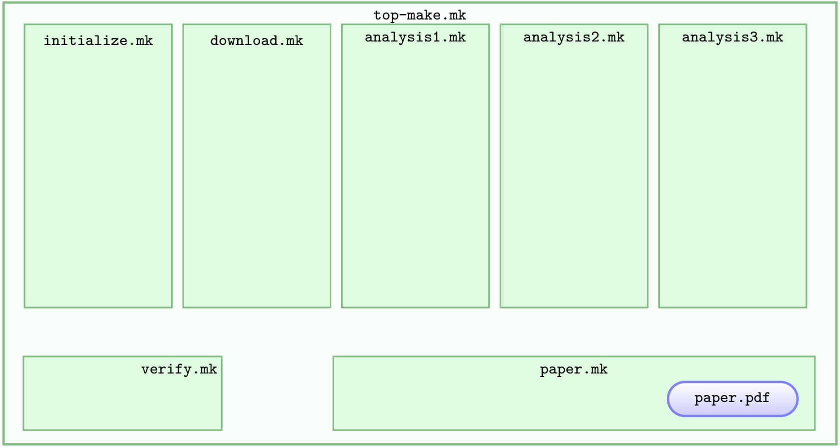
Let's see how the analysis is managed in a hypothetical project...

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



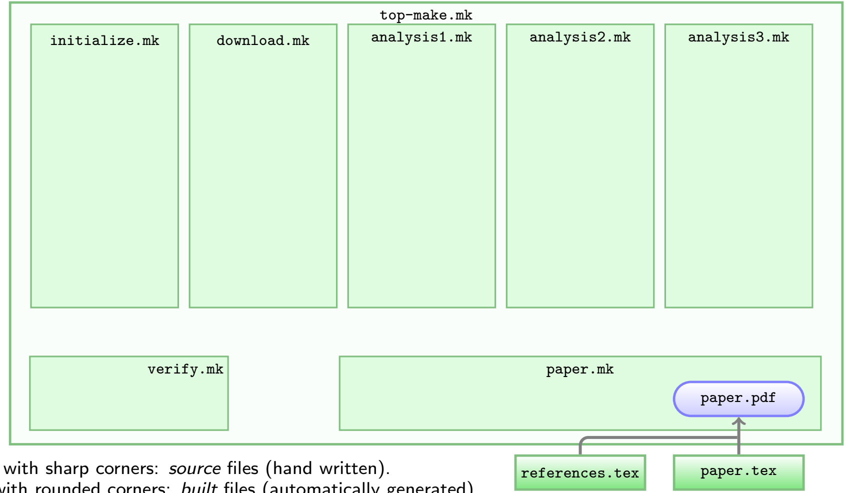
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

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



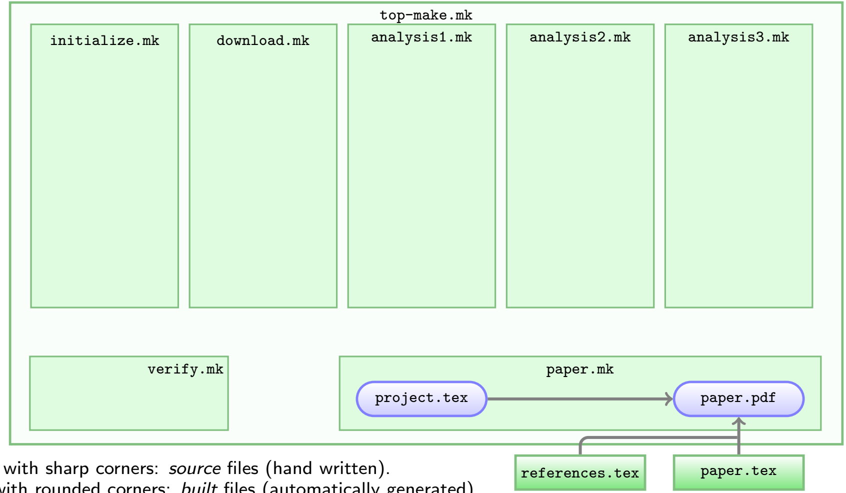
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

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



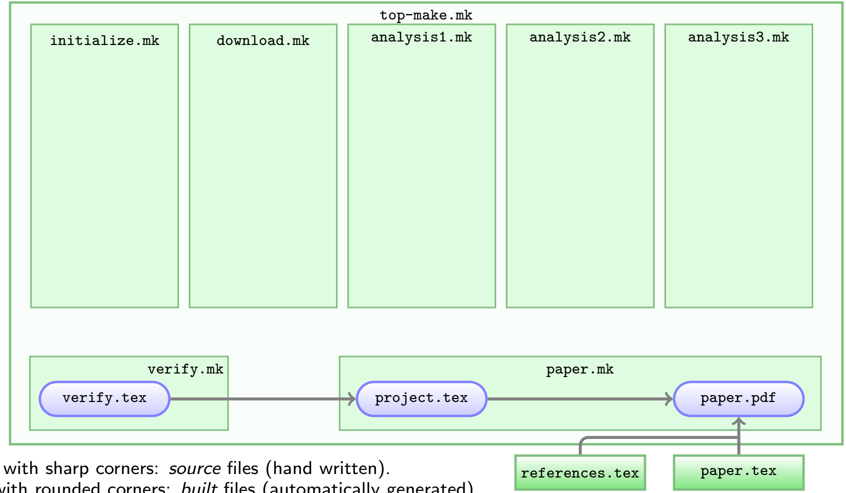
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

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



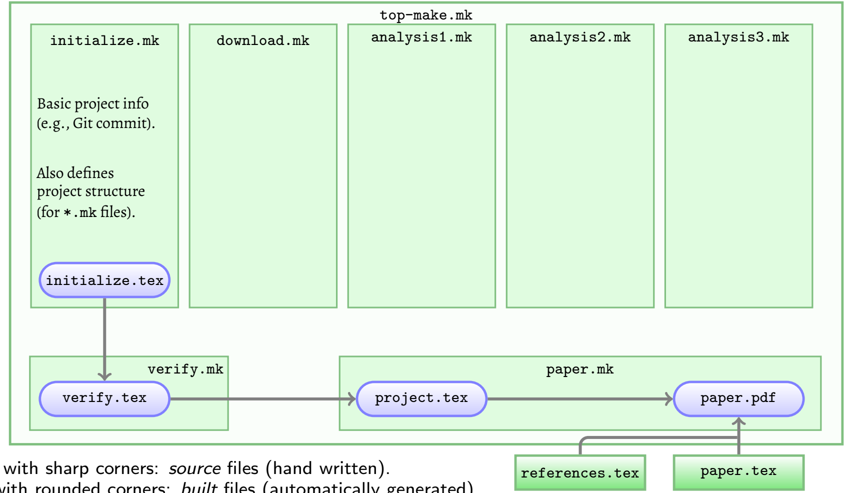
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

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



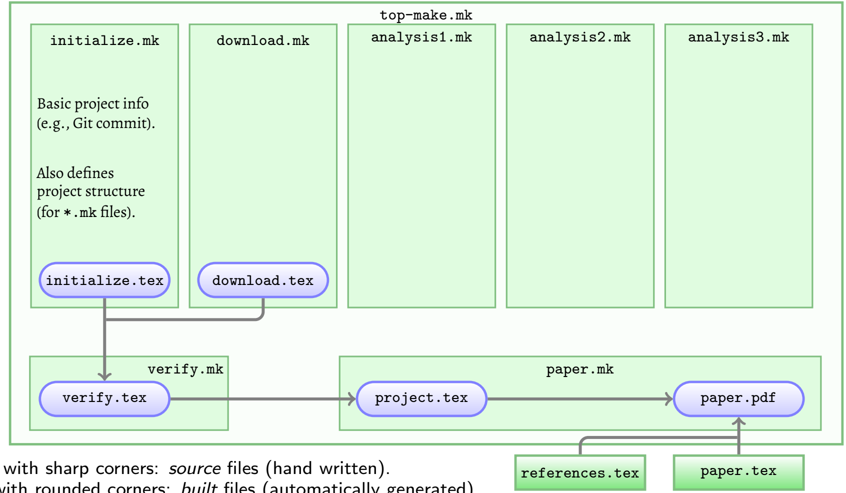
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

Basic project info comes from initialize.tex.



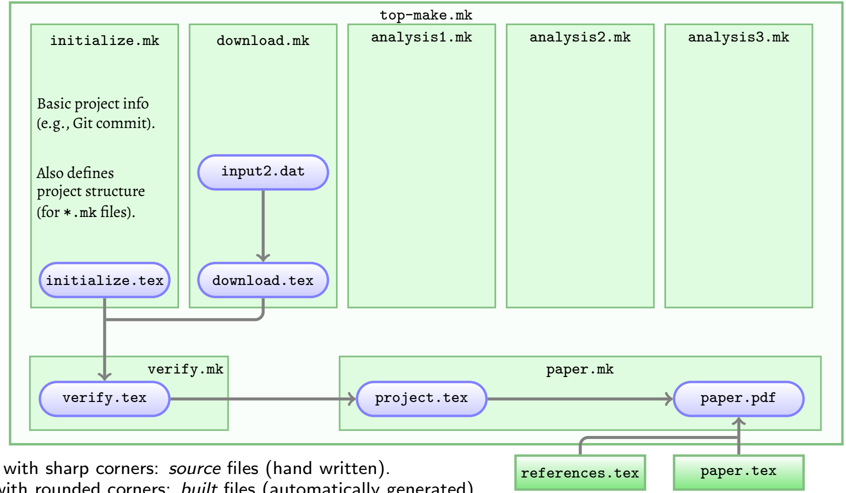
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

Reported values about the downloaded inputs come from download.tex.



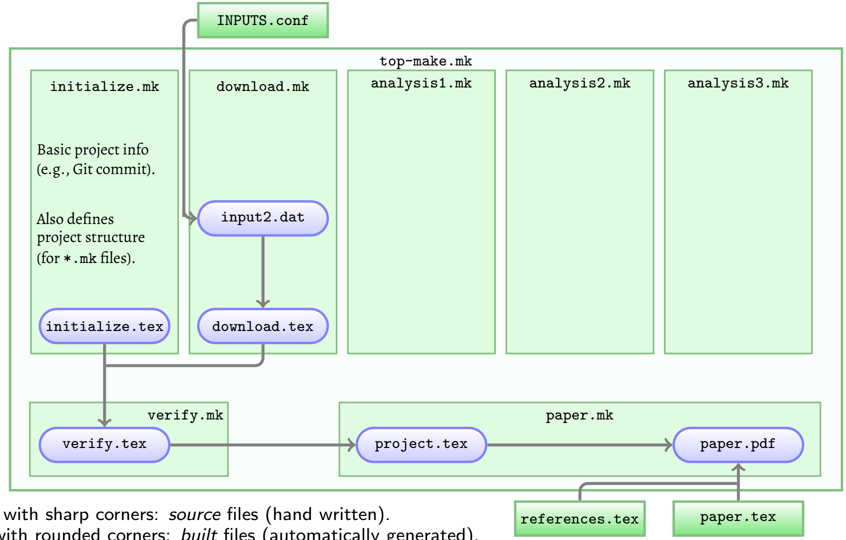
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

… for example the number of rows in the second input (a catalog) of the project.



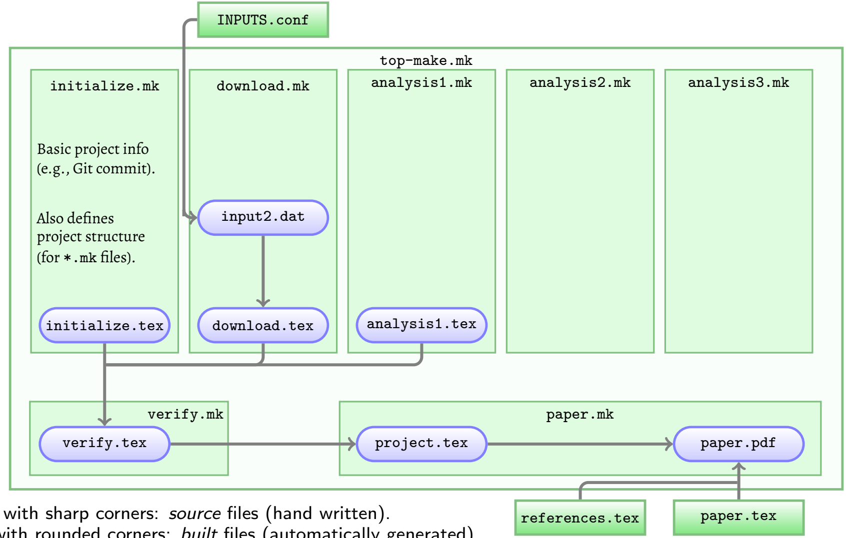
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

The URL to download input2.dat, and a checksum to validate it, are stored in INPUTS.conf.



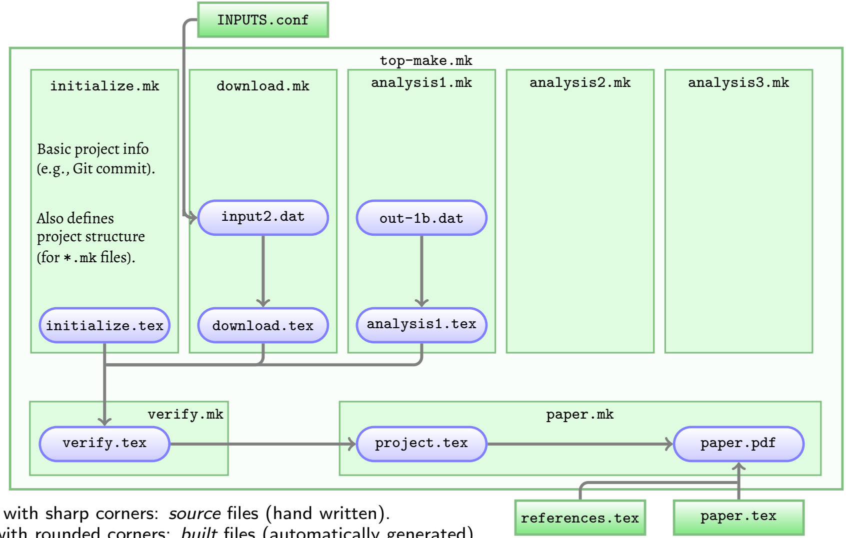
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

Reported values from first analysis steps stored in analysis1.tex.



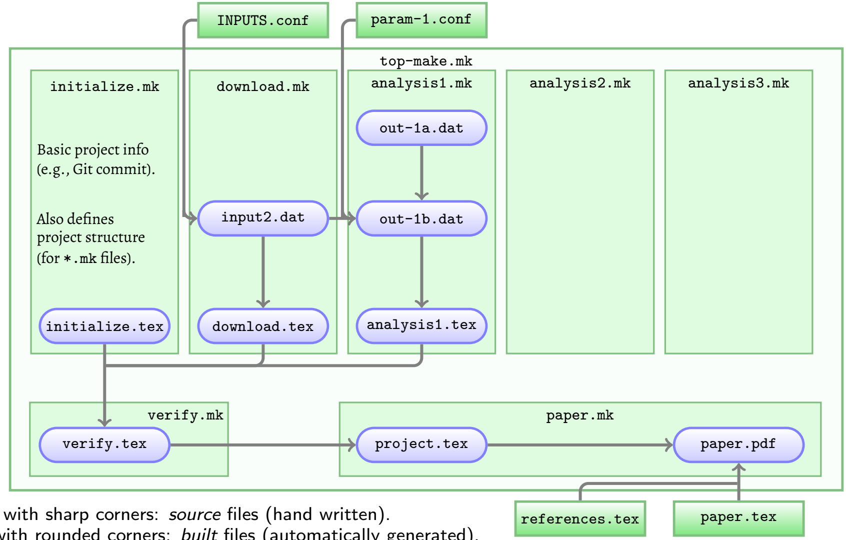
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

… for example the average of the numbers in out-1b.dat.



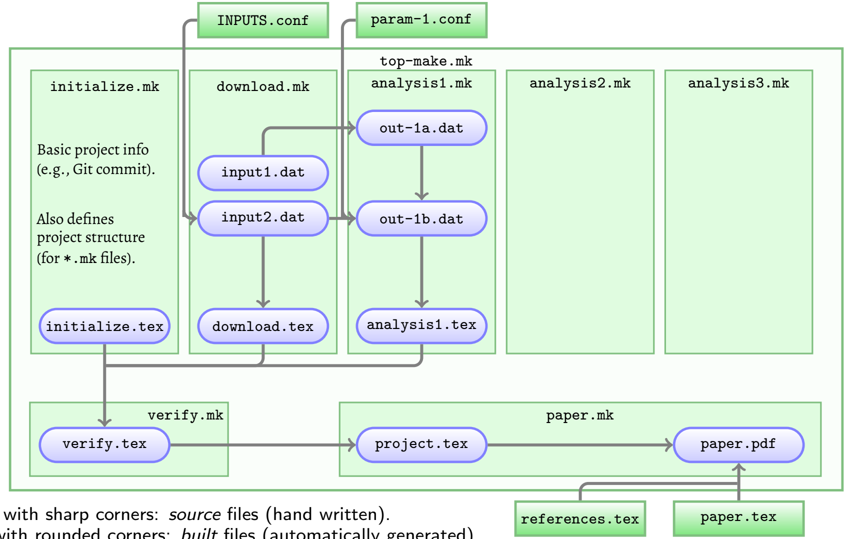
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

But out-1b.dat itself depends on other files and a paramter (for example a multiple of sigma).



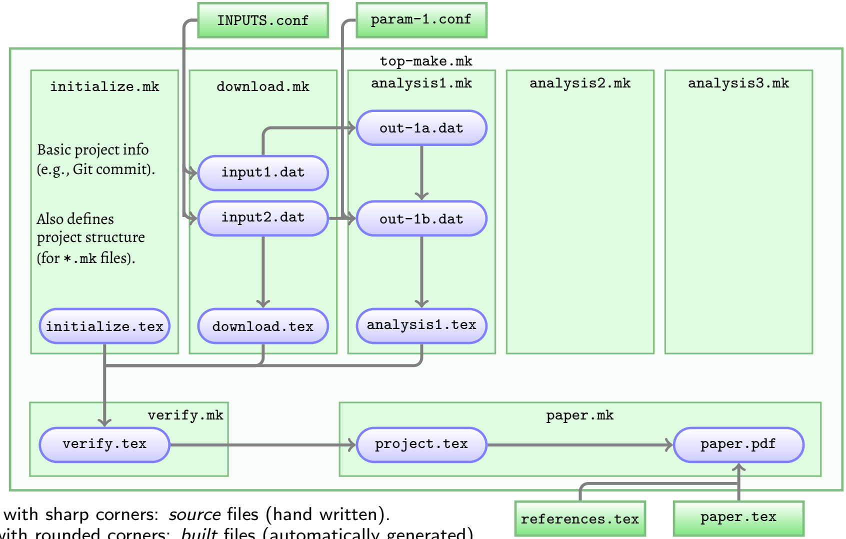
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

out-1a.dat is built from a downloaded dataset.



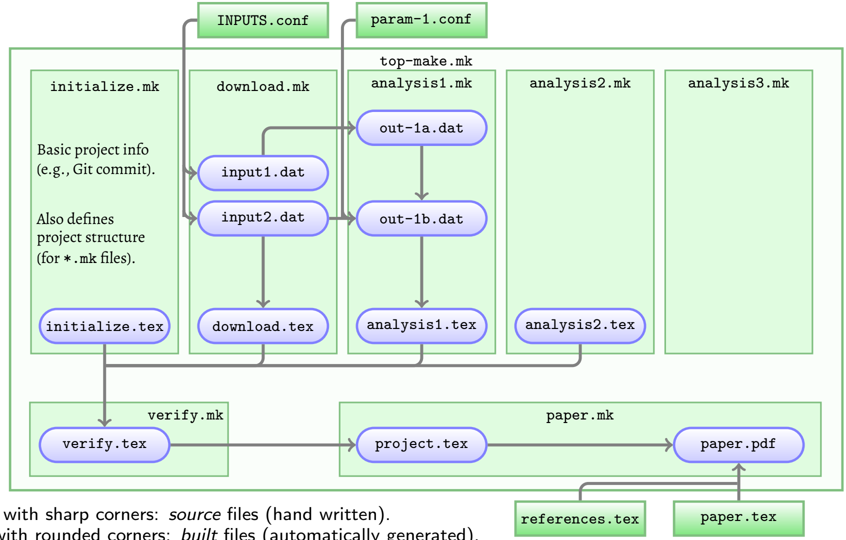
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

Download URL and checksum of input1.dat also stored in INPUTS.conf.



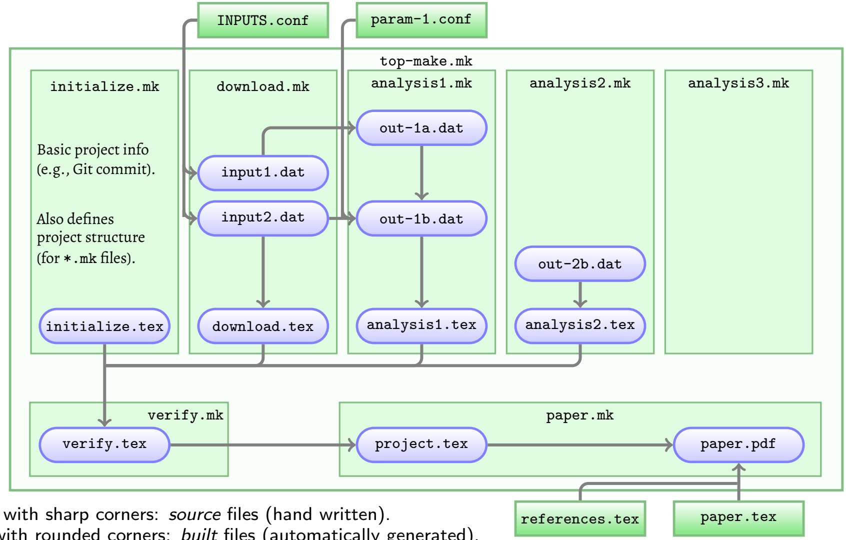
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

Reported values from second analysis steps stored in analysis2.tex.



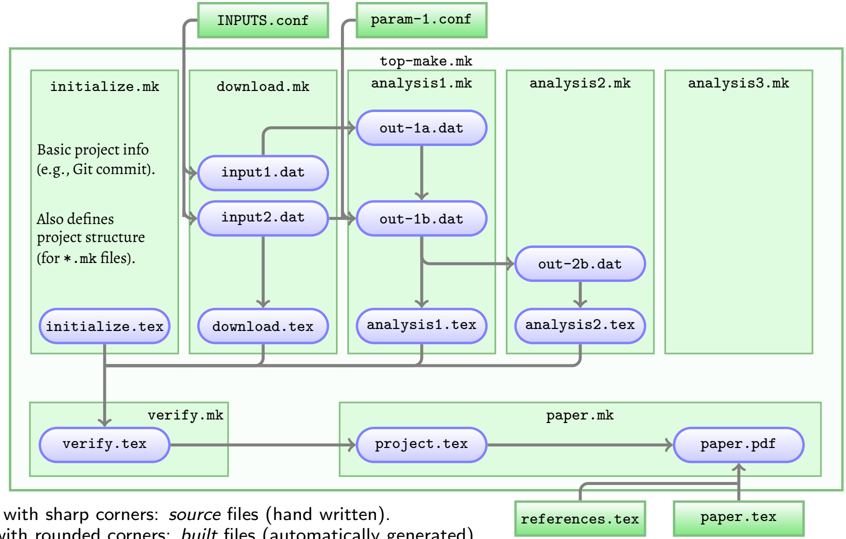
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

... for example the number of selected rows in out-2b.dat.



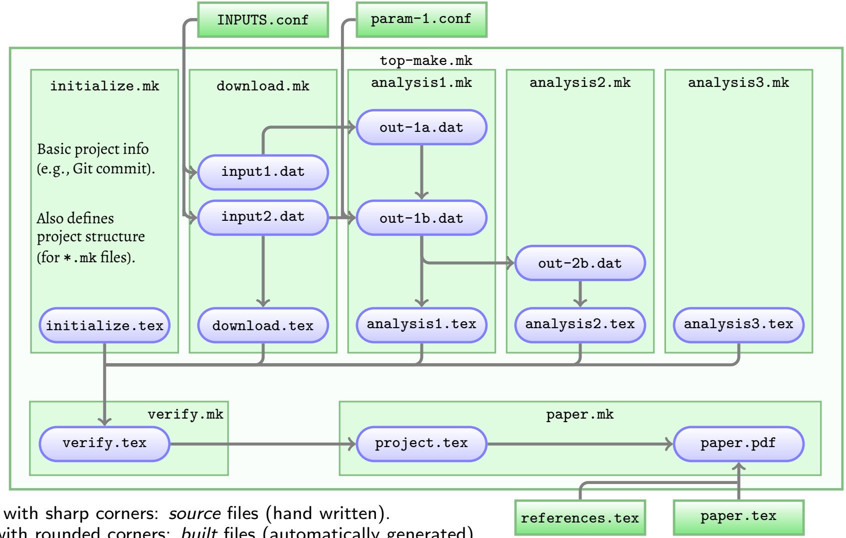
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

out−2b.dat is derived from out−1b.dat (for example, rejected some of out−1b.dat's rows).



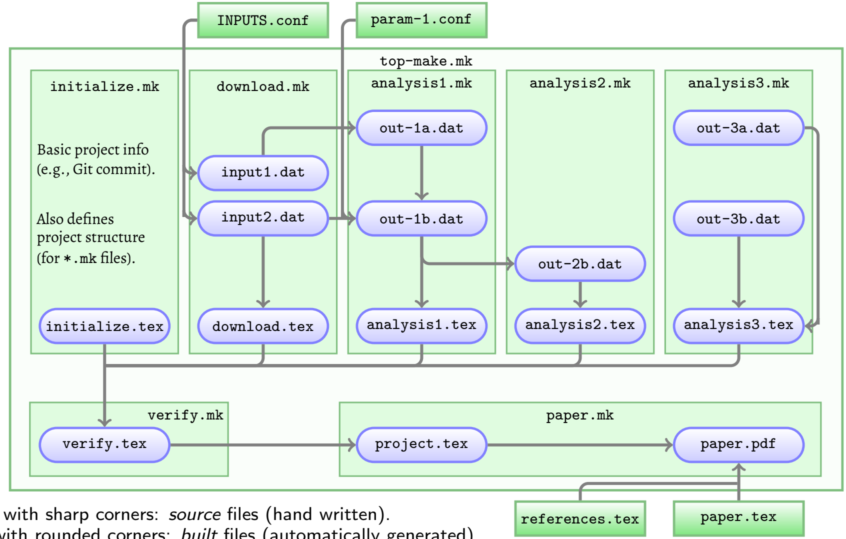
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

Reported values from third analysis steps stored in analysis3.tex.



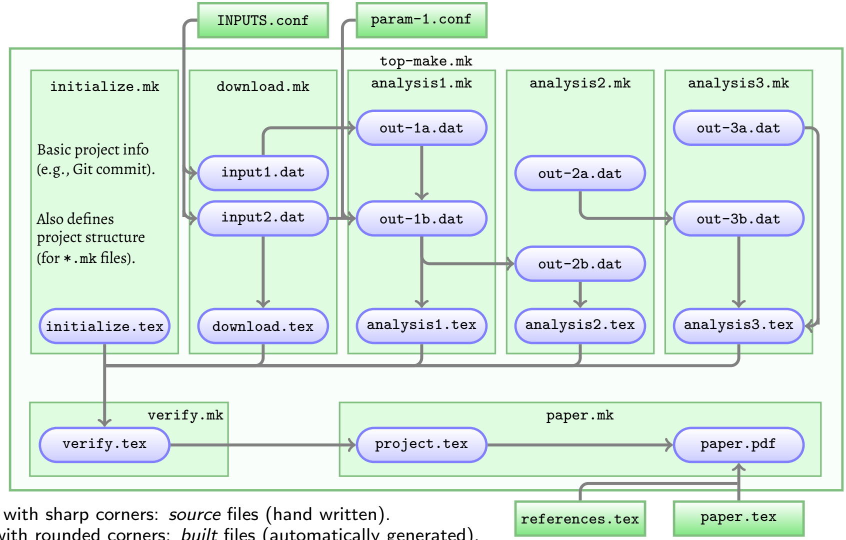
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

… for example measurements from both out-3a.dat and out-3b.dat.



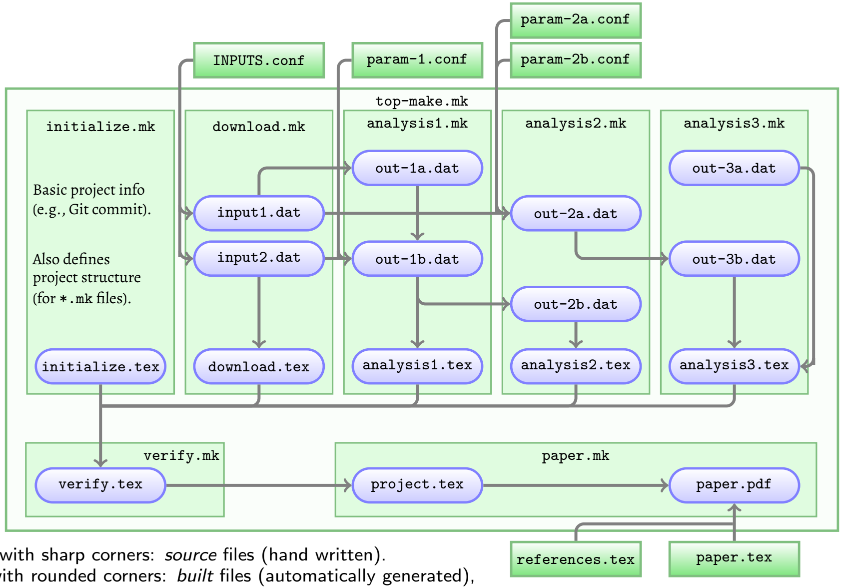
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

out-3b.dat is generated from an analysis on out-2a.dat.



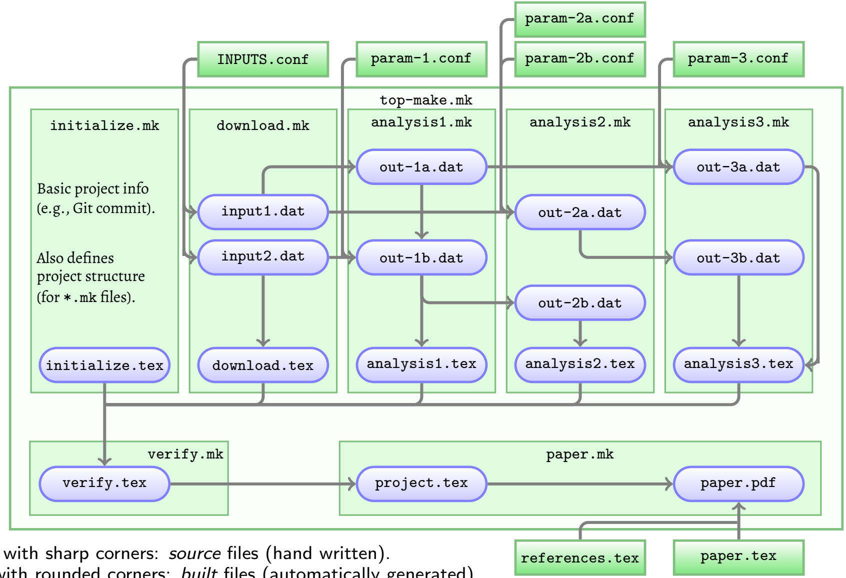
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

But out-2a.dat itself is generated from input1.dat and an analysis which has two settings.



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.

out-3a.dat also depends on out-1a.dat and an analysis with needs one parameter.



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`)

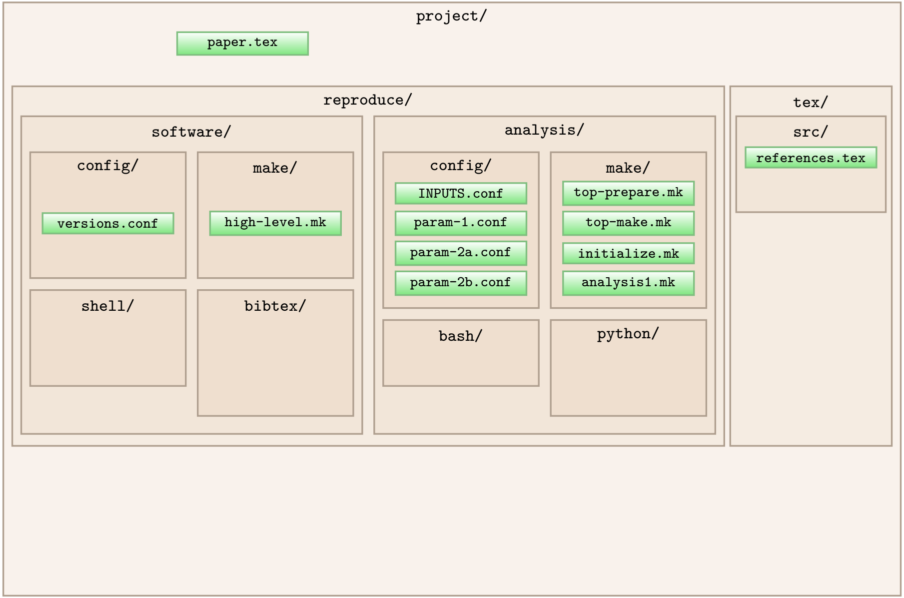
- There are two types of nodes/files:
  - Source nodes (`*.conf` and `paper.tex`) only have an outward link.
  - Built files always have inward *and* (except `paper.pdf`) outward link(s).

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

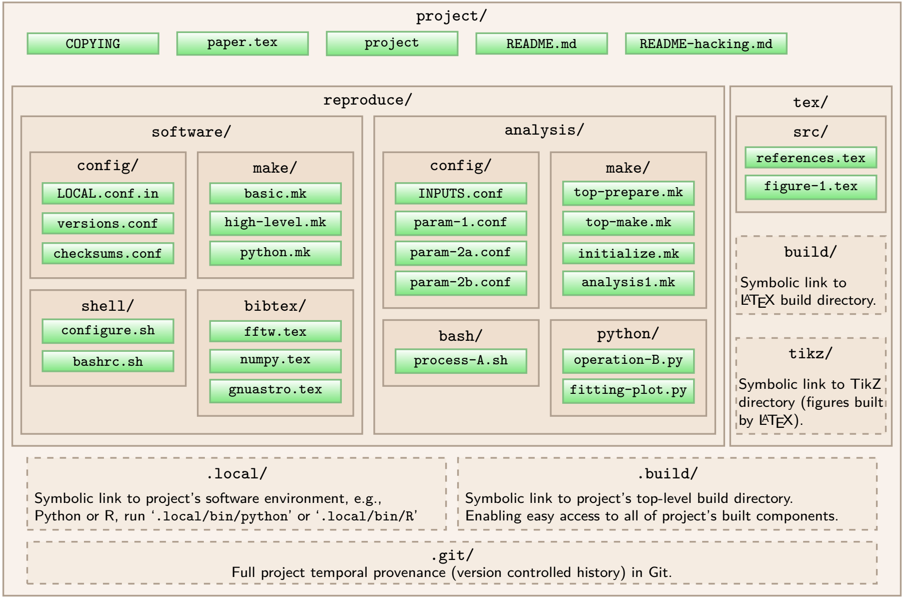
Benefits of using Make

- 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.

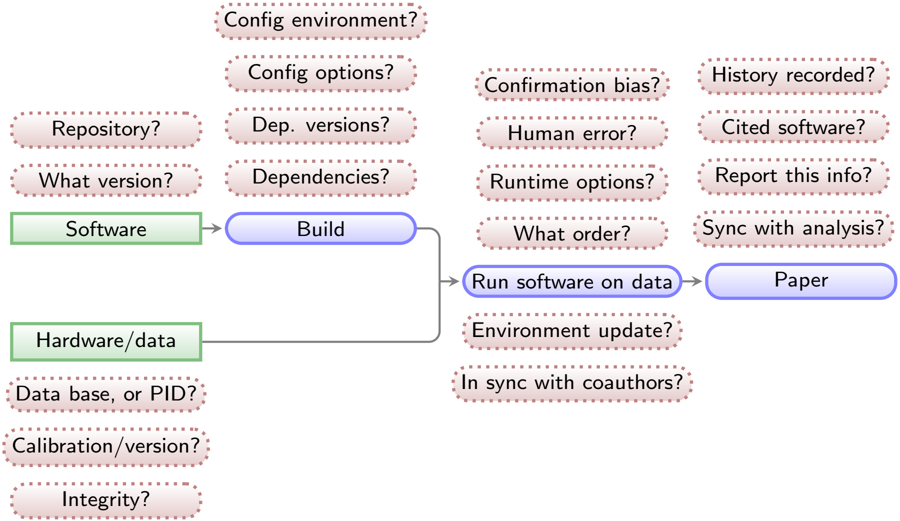
Files organized in directories by context (here are some of the files discussed before)



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

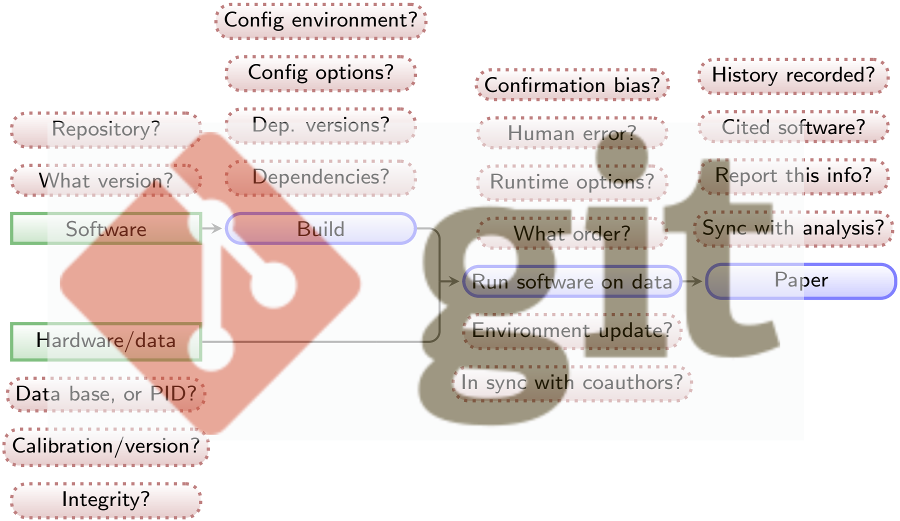


All questions have an answer now (in plain text: human & computer readable/archivable).



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.

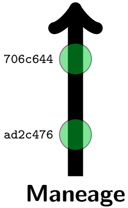
All questions have an answer now (in plain text: so we can use Git to keep its history).



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.

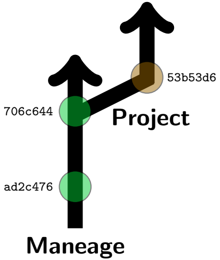
New projects branch from Maneage

- Template's history is recorded in Git.

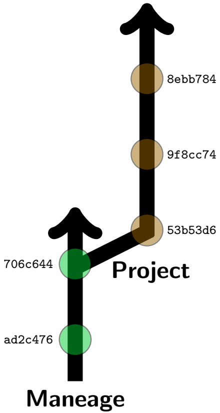


New projects branch from Maneage

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

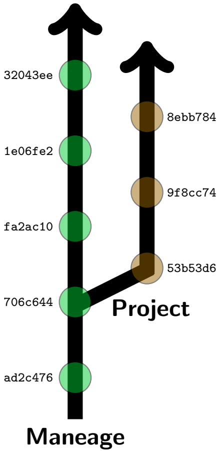


New projects branch from Maneage



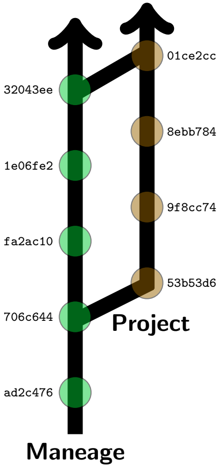
- Template's history is recorded in Git.
- New project: a branch from the template.
  Recall that every commit contains the following:
  - Instructions to download, verify and build software.
  - Instructions to download and verify input data.
  - Instructions to run software on data (do the analysis).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

New projects branch from Maneage



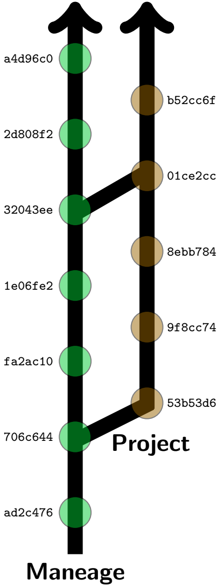
- Template's history is recorded in Git.
- New project: a branch from the template.
  Recall that every commit contains the following:
  - Instructions to download, verify and build software.
  - Instructions to download and verify input data.
  - Instructions to run software on data (do the analysis).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

New projects branch from Maneage



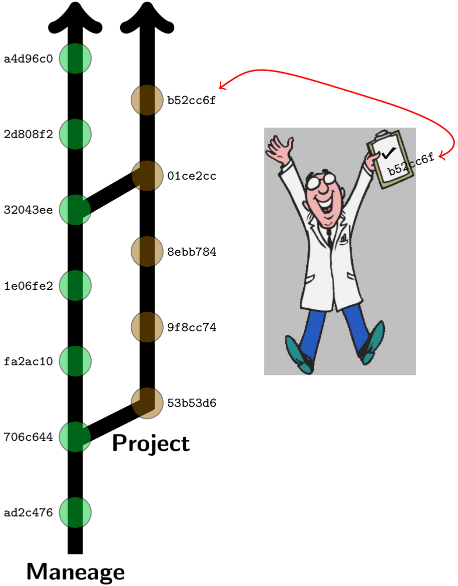
- Template's history is recorded in Git.
- New project: a branch from the template.
  Recall that every commit contains the following:
  - Instructions to download, verify and build software.
  - Instructions to download and verify input data.
  - Instructions to run software on data (do the analysis).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

New projects branch from Maneage



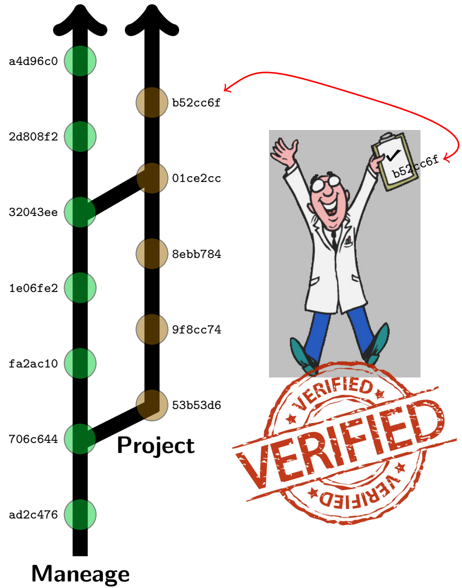
- Template's history is recorded in Git.
- New project: a branch from the template.
  Recall that every commit contains the following:
  - Instructions to download, verify and build software.
  - Instructions to download and verify input data.
  - Instructions to run software on data (do the analysis).
  - Narrative description of project's purpose/context.
- Research progresses in the project branch.
- Template will evolve (improved infrastructure).
- Template can be imported/merged back into project.
- The template and project will evolve.
- During research this encourages creative tests
  (previous research states can easily be retrieved).
- Coauthors can work on same project in parallel
  (separate project branches).

New projects branch from Maneage



- Template's history is recorded in Git.
- New project: a branch from the template.
  Recall that every commit contains the following:
  - Instructions to download, verify and build software.
  - Instructions to download and verify input data.
  - Instructions to run software on data (do the analysis).
  - Narrative description of project's purpose/context.
- Research progresses in the project branch.
- Template will evolve (improved infrastructure).
- Template can be imported/merged back into project.
- The template and project will evolve.
- During research this encourages creative tests (previous research states can easily be retrieved).
- Coauthors can work on same project in parallel (separate project branches).
- Upon publication, the Git checksum is enough to verify the integrity of the result.

New projects branch from Maneage



- Template's history is recorded in Git.
- New project: a branch from the template.
  Recall that every commit contains the following:
  - Instructions to download, verify and build software.
  - Instructions to download and verify input data.
  - Instructions to run software on data (do the analysis).
  - Narrative description of project's purpose/context.
- Research progresses in the project branch.
- Template will evolve (improved infrastructure).
- Template can be imported/merged back into project.
- The template and project will evolve.
- During research this encourages creative tests
  (previous research states can easily be retrieved).
- Coauthors can work on same project in parallel
  (separate project branches).
- Upon publication, the Git checksum is enough to
  verify the integrity of the result.

"Verified" image from vectorstock.com

Two recent examples (publishing Git checksum in abstract)

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

DOI: 00.0000/X000000000000000X

Carving out the low surface brightness universe with NoiseChisel

Mohammad Akhlaghi[1,2]

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

[2]Facultad de Física, Universidad de La Laguna, Avda. Astrofísico Fco. Sánchez s/n, 38200 La Laguna, Tenerife, Spain.

Abstract. NoiseChisel is a program to detect very low signal-to-noise ratio (S/N) features with minimal assumptions on their morphology. It was introduced in 2015 and released within a collection of data analysis programs and libraries known as GNU Astronomy Utilities (Gnuastro). Over the last ten stable releases of Gnuastro, NoiseChisel has significantly improved: detecting even fainter signal, enabling better user control over its inner workings, and many bug fixes. The most important change may be that NoiseChisel's segmentation features have been moved into a new program called Segment. Another major change is the final growth strategy of its true detections, for example NoiseChisel is able to detect the outer wings of M51 down to S/N of 0.25, or 28.27 mag/arcsec^2 on a single-exposure SDSS image (r-band). Segment is also able to detect the localized HII regions as "clumps" much more successfully. Finally, to orchestrate a controlled analysis, the concept of a "reproducible paper" is discussed: this paper itself is exactly reproducible (snapshot v4-0-g8505cfd).

Keywords. galaxies: halos, galaxies: photometry, galaxies: structure, methods: data analysis, methods: reproducible, techniques: image processing, techniques: photometric

1. Introduction

Signal from the low surface brightness universe is buried deep in the datasets noise and thus requires accurate detection methods. In Akhlaghi and Ichikawa (2015) (henceforth AI15) a new method was introduced to detect such very low signal-to-noise ratio (S/N) signal from the images in a non-parametric manner. It allows accurate detection of the diffuse outer features of galaxies (that often have a different morphology from the centers). The software implementation of this method (NoiseChisel) is released as part of a larger collection of data analysis software known as GNU Astronomy Utilities† (Gnuastro). It was the first professional astronomical software to be independently refereed by an independent panel (GNU Evaluation committee) and fully conforms with the GNU Coding Standards‡.

Since its release, NoiseChisel has been used in many studies. For example Bacon et al. (2017) used it to identify objects that were missed by Rafelski et al. (2015) (henceforth R15), who used a combination of six SExtractor (Bertin and Arnouts 1996) runs with different configurations to avoid deblending problems, but still missed many sources with significant signal, see Figure 1. Borlaff et al. (2019), Mller et al. (2019), and Trujillo et al. (2019) used it for accurate flat field and Sky subtraction to create deeper co-added images in galaxy fields for optimal detection of the low surface brightness features. Calvi et al. (2019) used it to find Lyman-α emitters in spectra. For future studies, Laine et al.

† https://www.gnu.org/s/gnuastro
‡ https://www.gnu.org/prep/standards

1

Monthly Notices of the ROYAL ASTRONOMICAL SOCIETY

MNRAS **491,** 5317–5329 (2020)   doi:10.1093/mnras/stz3111
Advance Access publication 2019 November 14

The Sloan Digital Sky Survey extended point spread functions

Raúl Infante-Sainz[1,2★] Ignacio Trujillo[1,2] and Javier Román[1,2,3]

[1]Instituto de Astrofísica de Canarias, c/ Vía Láctea s/n, E-38205 La Laguna, Tenerife, Spain
[2]Departamento de Astrofísica, Universidad de La Laguna, E-38205 La Laguna, Tenerife, Spain
[3]Instituto de Astrofísica de Andalucía (CSIC), Glorieta de la Astronomía, E-18008 Granada, Spain

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

ABSTRACT

A robust and extended characterization of the point spread function (PSF) is crucial to extract the photometric information produced by deep imaging surveys. Here, we present the extended PSFs of the Sloan Digital Sky Survey (SDSS), one of the most productive astronomical surveys of all time. By stacking ∼1000 images of individual stars with different brightness, we obtain the bidimensional SDSS PSFs extending over 8 arcmin in radius for all the SDSS filters (u, g, r, i, z). This new characterization of the SDSS PSFs is near a factor of 10 larger in extension than previous PSFs characterizations of the same survey. We found asymmetries in the shape of the PSFs caused by the drift scanning observing mode. The flux of the PSFs is larger along the drift scanning direction. Finally, we illustrate with an example how the PSF models can be used to remove the scattered light field produced by the brightest stars in the central region of the Coma cluster field. This particular example shows the huge importance of PSFs in the study of the low-surface brightness Universe, especially with the upcoming of ultradeep surveys, such as the Large Synoptic Survey Telescope (LSST). Following a reproducible science philosophy, we make all the PSF models and the scripts used to do the analysis of this paper publicly available (snapshot v0.4-0-gd966ad0).

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

1 INTRODUCTION

The point spread function (PSF) describes the response of an imaging system to the light produced by a point source. Real PSFs have complex structures as their shapes depend on the optical path that light takes as it travels through the atmosphere and multiple optical elements, mirrors, lenses, detectors, etc. For the vast majority of astronomical works, only a tiny portion of the PSF (i.e. normally a few inner arcseconds; see e.g. Trujillo et al. 2001a, b) is characterized. In practice, however, the light of both point and extended sources are spread over the entire detector due to the effect of the PSF at large radii. Therefore, it is necessary to have a good understanding of its structure along the entire detector (typically extending over arcminutes or more).

Extended PSFs have become a vital tool to obtain precise photometric information in modern astronomical surveys. For instance, Slater, Harding & Mihos (2009) modelled the extended PSF and the internal reflections produced by the stars of the Burrell Schmidt telescope and showed that virtually all the pixels of the image are dominated by the scattered light by both stars and galaxies at 29.5 mag arcsec^{-2} (V-band). Trujillo & Fliri (2016) also characterized and used the extended PSF of the 10.4 m Gran Telescopio Canarias (GTC) telescope to model and remove the scattered light in ultradeep observations of the UGC 00180 galaxy. Even more troublesome for low-surface brightness studies is the finding (see e.g. Trujillo & Bakos 2013; Sandin 2014, 2015) that the outer regions of astronomical objects are severely affected by their own scattered light produced by the convolution with the PSF. In order to correct this effect, Karabal et al. (2017) generated the PSF and models of the internal reflections from images of the Canada–France–Hawaii Telescope (CFHT) to de-convolve a sample of three galaxies and correct them from instrumental scattered light. More recently, Román, Trujillo & Montes (2019) characterized the PSFs of the Stripe 82 survey and used them to model and correct the scattered light field produced by stars to study the optical properties of the Galactic cirri. All the above works have shown that having an extended PSF is crucial when accurate photometric and structure properties of astronomical objects at low-surface brightness levels are required.

One of the most commonly used surveys for measuring photometric properties of astronomical objects is the Sloan Sky Digital Survey (SDSS; York et al. 2000), covering 14 555 deg^2 on the sky (just over 35 per cent of the full sky) in five photometric bands (u, g, r, i, and z). Although SDSS is a relatively shallow survey compared

★ E-mail: infantesainz@gmail.com

Two recent examples (publishing Git checksum in abstract)

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

DOI: 00.0000/X000000000000000X

Carving out the low surface brightness universe with NoiseChisel

Mohammad Akhlaghi[1,2]

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

[2]Facultad de Física, Universidad de La Laguna, Avda. Astrofísico Fco. Sánchez s/n, 38200 La Laguna, Tenerife, Spain.

Abstract. NoiseChisel is a program to detect very low signal-to-noise ratio (S/N) features with minimal assumptions on their morphology. It was introduced in 2015 and released within a collection of data analysis programs and libraries known as GNU Astronomy Utilities (Gnuastro). Over the last ten stable releases of Gnuastro, NoiseChisel has significantly improved: detecting even fainter signal, enabling better user control over its inner workings, and many bug fixes. The most important change may be that NoiseChisel's segmentation features have been moved into a new program called Segment. Another major change is the final growth strategy of its true detections, for example NoiseChisel is able to detect the outer wings of M51 down to S/N of 0.25, or 28.27 mag/arcsec^2 on a single-exposure SDSS image (r-band). Segment is also able to detect the localized HII regions as "clumps" much more successfully. Finally, to orchestrate a controlled analysis, the concept of a "reproducible paper" is discussed: this paper itself is exactly reproducible (snapshot v4-0-g8505cfd).

VERIFIED

Keywords. galaxies: halos, galaxies: photometry, galaxies: structure, methods: data analysis, methods: reproducible, techniques: image processing, techniques: photometric

1. Introduction

Signal from the low surface brightness universe is buried deep in the datasets noise and thus requires accurate detection methods. In Akhlaghi and Ichikawa (2015) (henceforth AI15) a new method was introduced to detect such very low signal-to-noise ratio (S/N) signal from the images in a non-parametric manner. It allows accurate detection of the diffuse outer features of galaxies (that often have a different morphology from the centers). The software implementation of this method (NoiseChisel) is released as part of a larger collection of data analysis software known as GNU Astronomy Utilities† (Gnuastro). It was the first professional astronomical software to be independently refereed by an independent panel (GNU Evaluation committee) and fully conforms with the GNU Coding Standards‡.

Since its release, NoiseChisel has been used in many studies. For example Bacon et al. (2017) used it to identify objects that were missed by Rafelski et al. (2015) (henceforth R15), who used a combination of six SExtractor (Bertin and Arnouts 1996) runs with different configurations to avoid deblending problems, but still missed many sources with significant signal, see Figure 1. Borlaff et al. (2019), Mller et al. (2019), and Trujillo et al. (2019) used it for calibrating flat field and Sky subtraction to create deeper co-added images in galaxy fields for optimal detection of the low surface brightness features. Calvi et al. (2019) used it to find Lyman-α emitters in spectra. For future studies, Laine et al.

† https://www.gnu.org/s/gnuastro
‡ https://www.gnu.org/prep/standards

Monthly Notices of the ROYAL ASTRONOMICAL SOCIETY

MNRAS **491**, 5317–5329 (2020) doi:10.1093/mnras/stz3111
Advance Access publication 2019 November 14

The Sloan Digital Sky Survey extended point spread functions

Raúl Infante-Sainz[1,2]★ Ignacio Trujillo[1,2] and Javier Román[1,2,3]
[1]*Instituto de Astrofísica de Canarias, c/ Vía Láctea s/n, E-38205 La Laguna, Tenerife, Spain*
[2]*Departamento de Astrofísica, Universidad de La Laguna, E-38205 La Laguna, Tenerife, Spain*
[3]*Instituto de Astrofísica de Andalucía (CSIC), Glorieta de la Astronomía, E-18008 Granada, Spain*

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

ABSTRACT
A robust and extended characterization of the point spread function (PSF) is crucial to extract the photometric information produced by deep imaging surveys. Here, we present the extended PSFs of the Sloan Digital Sky Survey (SDSS), one of the most productive astronomical surveys of all time. By stacking ~1000 images of individual stars with different brightness, we obtain the bidimensional SDSS PSFs extending over 8 arcmin in radius for all the SDSS filters (u, g, r, i, z). This new characterization of the SDSS PSFs is near a factor of 10 larger in extension than previous PSFs characterizations of the same survey. We found asymmetries in the shape of the PSFs caused by the drift scanning observing mode. The flux of the PSFs is larger along the drift scanning direction. Finally, we illustrate with an example how the PSF models can be used to remove the scattered light field produced by the brightest stars in the central region of the Coma cluster field. This particular example shows the huge importance of PSFs in the study of the low-surface brightness Universe, especially with the upcoming of ultradeep surveys, such as the Large Synoptic Survey Telescope (LSST). Following a reproducible science philosophy, we make all the PSF models and the scripts used to do the analysis of this paper publicly available (snapshot v0.4-0-gd966ad0).

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

VERIFIED

1 INTRODUCTION

The point spread function (PSF) describes the response of an imaging system to the light produced by a point source. Real PSFs have complex structures as their shapes depend on the optical path that light takes as it travels through the atmosphere and multiple optical elements, mirrors, lenses, detectors, etc. For the vast majority of astronomical works, only a tiny portion of the PSF (i.e. normally a few inner arcseconds; see e.g. Trujillo et al. 2001a, b) is characterized. In practice, however, the light of both point and extended sources are spread over the entire detector due to the effect of the PSF at large radii. Therefore, it is necessary to have a good understanding of its structure along the entire detector (typically extending over arcminutes or more).

Extended PSFs have become a vital tool to obtain precise photometric information in modern astronomical surveys. For instance, Slater, Harding & Mihos (2009) modelled the extended PSF and the internal reflections produced by the stars of the Burrell Schmidt telescope and showed that virtually all the pixels of the image are dominated by the scattered light by both stars and galaxies at 29.5 mag arcsec^{-2} (V-band). Trujillo & Fliri (2016) also characterized and used the extended PSF of the 10.4 m Gran Telescopio Canarias (GTC) telescope to model and remove the scattered light in ultradeep observations of the UGC 00180 galaxy. Even more troublesome for low-surface brightness studies is the finding (see e.g. Trujillo & Bakos 2013; Sandin 2014, 2015) that the outer regions of astronomical objects are severely affected by their own scattered light produced by the convolution with the PSF. In order to correct this effect, Karabal et al. (2017) generated the PSF and models of the internal reflections from images of the Canada–France–Hawaii Telescope (CFHT) to de-convolve a sample of three galaxies and correct them from instrumental scattered light. More recently, Román, Trujillo & Montes (2019) characterized the PSFs of the Stripe 82 survey and used them to model and correct the scattered light field produced by stars to study the optical properties of the Galactic cirri. All the above works have shown that having an extended PSF is crucial when accurate photometric and structure properties of astronomical objects at low-surface brightness levels are required.

One of the most commonly used surveys for measuring photometric properties of astronomical objects is the Sloan Sky Digital Survey (SDSS; York et al. 2000), covering 14 555 deg^2 on the sky (just over 35 per cent of the full sky) in five photometric bands (u, g, r, i, and z). Although SDSS is a relatively shallow survey compared

★ E-mail: infantesainz@gmail.com

Publication of the project

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

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

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

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

- arXiv: uploaded with the LaTeX source to always stay with the paper (for example arXiv:1505.01664). The file containing all macros must also be uploaded so arXiv's server can easily build the LaTeX source.
- Zenodo: Along with all the input datasets (many Gigabytes) and software (for example zenodo.3408481) and given a unique DOI.

Project source and its execution

> Programs [here: Scientific projects] must be written for people to read...
> ...and only *incidentally* for machines to *execute*.
>
> Harold Abelson, Structure and Interpretation of Computer Programs

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

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

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

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

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

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

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

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

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

Future prospects...

Adoption of reproducibility by many researchers will enable the following:

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

RDA adoption grant (2019) to IAC for Maneage

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

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

Workshop on Maneage at IAC: first week of April (March 30th to April 3rd)

We are organizing a workshop to help interested early career researchers adopt Maneage.

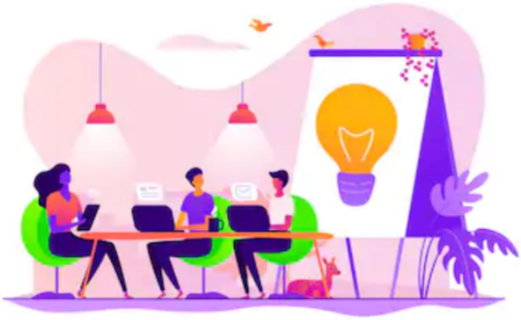


Image from shutterstock.com

Please contact akhlaghi@iac.es to join (Space is very limited: it is hands-on).

Existing technologies (Independent environment)

- **Virtual machines:**
  - Contain the full operating system, are thus very large ($\times$Gigabytes).
  - In *binary* format (decoding a built VM's environment is extremely hard and inaccurate).
- **Containers:** (For example Docker or Singularity)
  - Similar to virtual machines, but without low-level kernel (use host's kernel).
  - Will fail as soon as kernel is no longer supported
    (for example Docker currently only supports Linux kernel 3.10 and above from 2013).
  - Good solutions for software engineers (that need to *reproduce a bug's environment today*).
  - Docker is modular, needs root previlages (not available in HPCs), Dockerfiles allow incompleteness (especially in the common scenario of using the operating system's package manager, see next slide)
  - Singularity is monolithic and thus can be very large.
  - In binary format (similar to VMs, especially when OS package managers are used).

In summary, they only store a built environment (they are outputs, not good for archiving).

Existing technologies (Package managers)

- **Operating system package managers:**
  - For example apt or yum for Debian-based and RedHat-based GNU/Linux operating systems (the most common way to install software).
  - Tightly intertwined with the operating system's components (arbitrary control of software versions is not easily possible).
  - Older software (for example +5 years) is usually removed.
- **Conda/Anaconda:**
  - Conda has build instructions for software and their dependencies.
  - But it doesn't go down to the C library or the lower-level components of operating system.
  - It is written in Python (can't be used later when current Python is depreciated).
  - Authors of Uhse+2019[1] report[2] that their Conda environment breaks roughly every 3 months (Conda environments need to be updated to be used later! Breaking reproducibility).
- **Nix, or GNU Guix:**
  - Deliver perfectly reproducible builds (bit-wise reproducibility of software), needs root access.
  - Doesn't *require* documentation of dependencies.
- **Spack:** Similar to Nix/Guix but written in Python.

[1] http://dx.doi.org/10.1002/cppb.20097
[2] https://github.com/conda-forge/conda-forge.github.io/issues/787

Existing technologies (workflow tools)

- **Binder:** (https://mybinder.org) Docker+Conda.
- **Galaxy:** (https://galaxyproject.org) A web-based user interface, primarily designed for genomics. The GUI make it hard to automate, and has too many dependencies. Very similar to GenePattern (2008 to 2017): with +40,000 users and $\sim$ 4000 jobs running per week, but cut due to funding.
- **Sciunit:** (https://sciunit.run) Parses program binaries to try to infer their dependencies and copy them.
- **Popper:** (https://falsifiable.us), HCL (previously used by GitHub Actions) + Conda + Docker.
- **WholeTale:** (https://wholetale.org) Jupyter + Conda + Docker.
- **Image Processing On Line (IPOL) journal:** The best example of publishing algorithms/methods I have seen, only useful for very basic/low-level software.

Summary: except for IPOL, most solutions surveyed have far too many dependencies to be usable beyond the immediate future.

Summary:

Maneage is introduced as a customizable template that will do the following steps/instructions (all in simple plain text files).

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

For a technical description of Maneage's implementation, as well as a checklist to customize it, and tips on good practices, please see this page:
`https://gitlab.com/maneage/project/-/blob/maneage/README-hacking.md`