BIG Data, **BIG** responsibility

Maneage: *Man*aging data lin*eage* for long-term and archivable reproducibility

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

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

SoftwareHeritage 5th Anniversary
November 30th, 2021 (Inria, Paris)

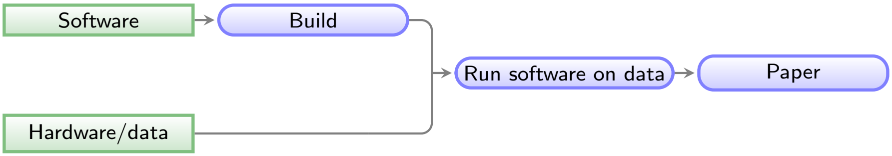
Most recent slides available in link below (this PDF is built from Git commit fc30dd7):
https://maneage.org/pdf/slides-intro-short.pdf

Financiado por la Unión Europea-NextGenerationEU

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

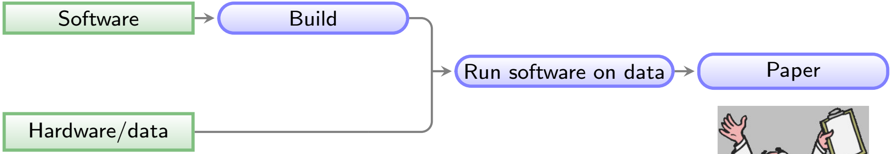
This project received financial support from the European Union's Horizon 2020 research and innovation grant agreement No. 777388 to the RDA EU 4.0 project.

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)

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?

Confirmation bias?

Human error?

Runtime options?

What order?

Run software on data → Paper

Environment update?

In sync with coauthors?

History recorded?

Cited software?

Report this info?

Sync with analysis?

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

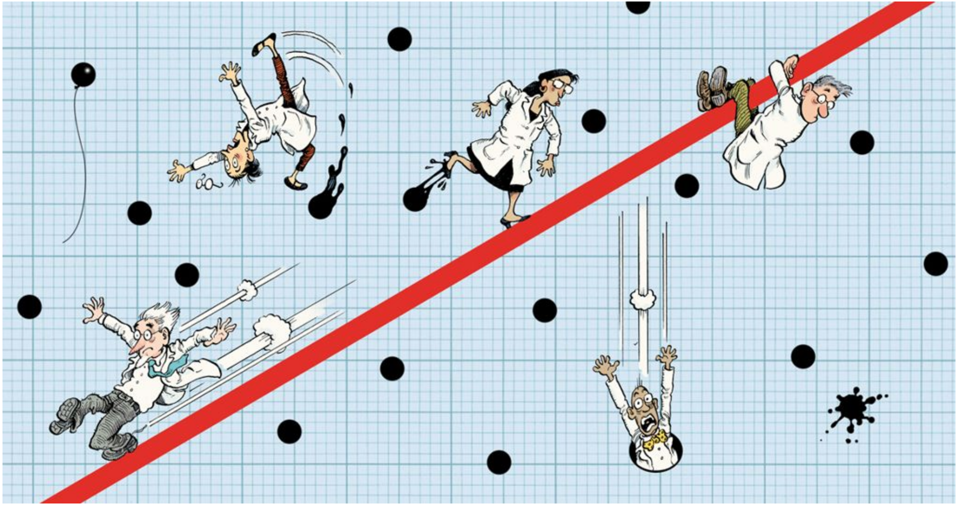
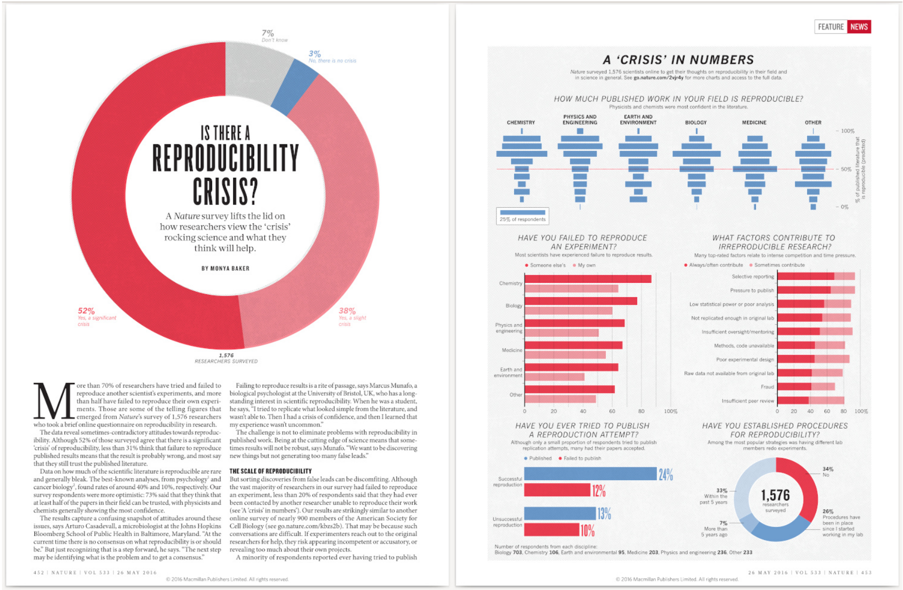


Image from nature.com ("Five ways to fix statistics", Nov 2017)

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.

“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

Founding criteria

Basic/simple principle:

Science is defined by its METHOD, not its result.

Founding criteria

Basic/simple principle:

Science is defined by its METHOD, not its result.

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

Founding criteria

Basic/simple principle:

Science is defined by its METHOD, not its result.

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

Founding criteria

Basic/simple principle:

Science is defined by its METHOD, not its result.

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

Founding criteria

Basic/simple principle:

Science is defined by its METHOD, not its result.

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

Founding criteria

Basic/simple principle:

Science is defined by its METHOD, not its result.

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

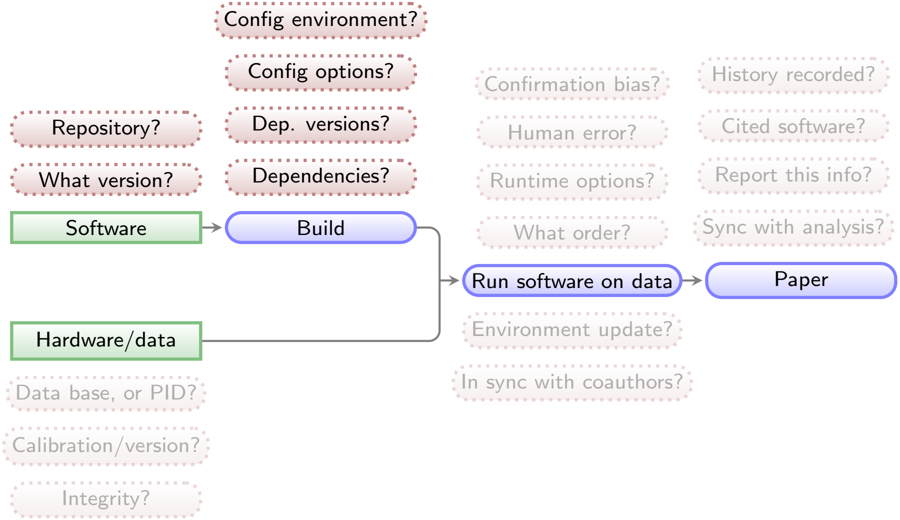
Founding criteria

Basic/simple principle:

Science is defined by its METHOD, not its result.

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

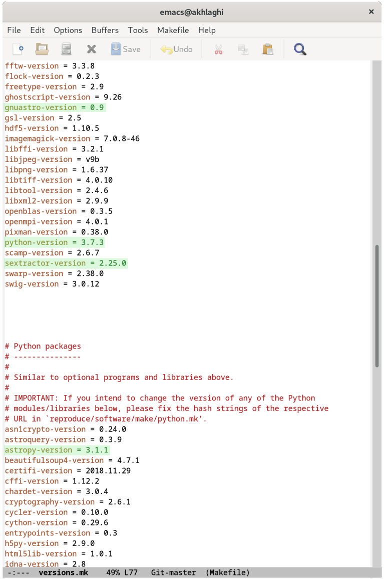
- *Containers* or *Virtual Machines* are a binary black box.
  - e.g., with '`FROM ubuntu:16.04`' (released in April 2016),
  - in a `Dockerfile`, the OS image will come from (updated monthly!): `https://partner-images.canonical.com/core/xenial`
- Maneage installs fixed versions of all necessary research software.
  - Including their dependencies.
  - All the way down to the C compiler.
- Installs similar environment on GNU/Linux, or macOS systems.
- Works like a package manager (e.g., `apt`, `brew` or Conda).
  - ... but (!), its not a third party package manager.
  - Build instructions are within same analysis project.
  - e.g., see Conda's build of Gnuastro (its gets updated behind your back): `https://anaconda.org/conda-forge/gnuastro/files`
- Source code of all software in Maneage is archived on zenodo.3883409.

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.
  - e.g., with '`FROM ubuntu:16.04`' (released in April 2016),
  - in a `Dockerfile`, the OS image will come from (updated monthly!): `https://partner-images.canonical.com/core/xenial`
- Maneage installs fixed versions of all necessary research software.
  - Including their dependencies.
  - All the way down to the C compiler.
- Installs similar environment on GNU/Linux, or macOS systems.
- Works like a package manager (e.g., `apt`, `brew` or Conda).
  - ... but (!), its not a third party package manager.
  - Build instructions are within same analysis project.
  - e.g., see Conda's build of Gnuastro (its gets updated behind your back): `https://anaconda.org/conda-forge/gnuastro/files`
- Source code of all software in Maneage is archived on zenodo.3883409.



Example: Matplotlib (a Python visualization library) build dependencies

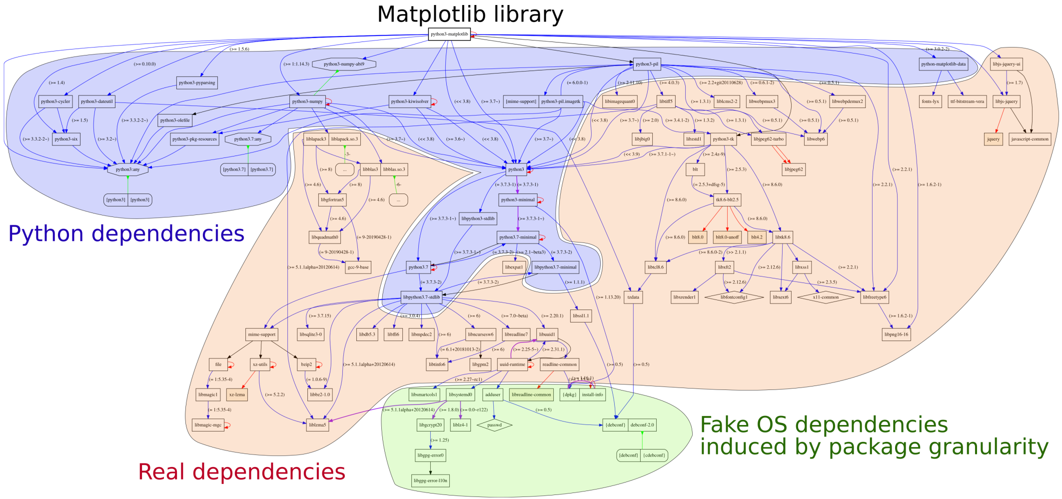


Fig. 1. Transitive dependencies of the software environment required by a simple "`import matplotlib`" command in the Python 3 interpreter.

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

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)

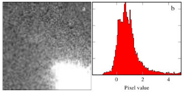


Figure 2: (a) An example image of the Wide-Field Planetary Camera 2, on board the Hubble Space Telescope from 1993 to 2009. This is one of the sample images from the FITS standard webpage, kept as examples for this file format. (b) Histogram of pixel values in (a).

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)

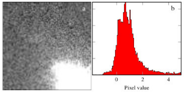


Figure 2: (a) An example image of the Wide-Field Planetary Camera 2, on board the Hubble Space Telescope from 1993 to 2009. This is one of the sample images from the FITS standard webpage, kept as examples for this file format. (b) Histogram of pixel values in (a).

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.

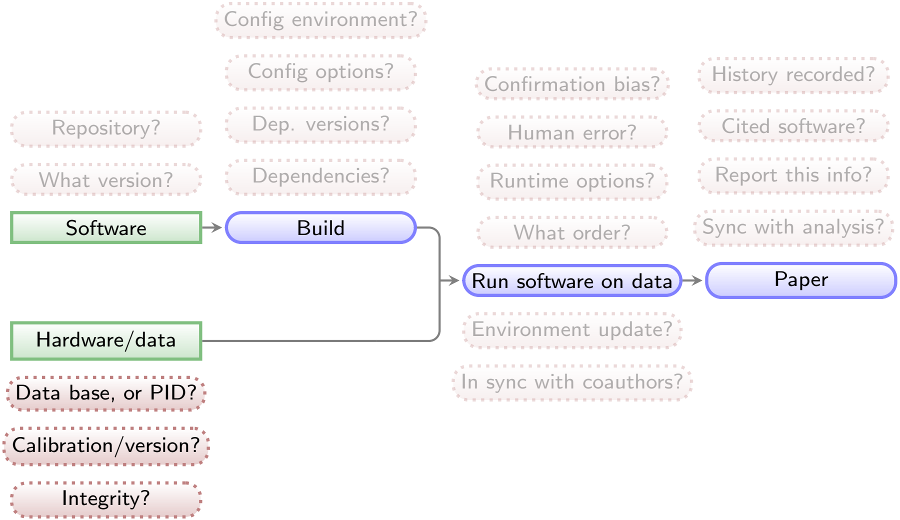
This research was done with the following free software programs and libraries: Bzip2 1.0.6, CFITSIO 3.45, CMake 3.14.2, cURL 7.63.0, Discoteq flock 0.2.3, File 5.36, FreeType 2.9, Git 2.21.0, GNU Astronomy Utilities 0.9 (Akhlaghi and Ichikawa 2015), GNU AWK 5.0.0, GNU Bash 5.0.7, GNU Binutils 2.32, 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.

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).

emacs@akhlaghi

File Edit Options Buffers Tools Makefile Help

```
# 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.

Input data source and integrity is documented and checked

Stored information about each input file:

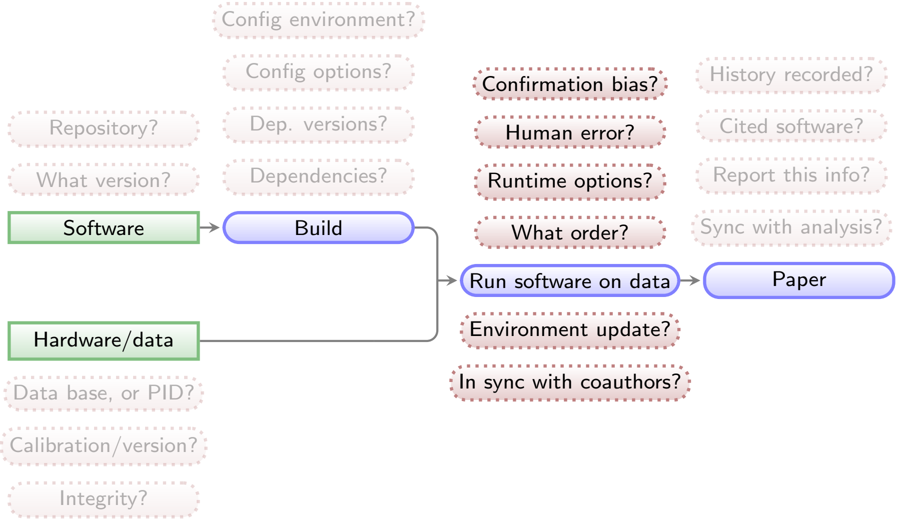
- PID (where available).
- Download URL.
- MD5-sum to check integrity.

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

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

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

General outline of a project (after data collection)



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):

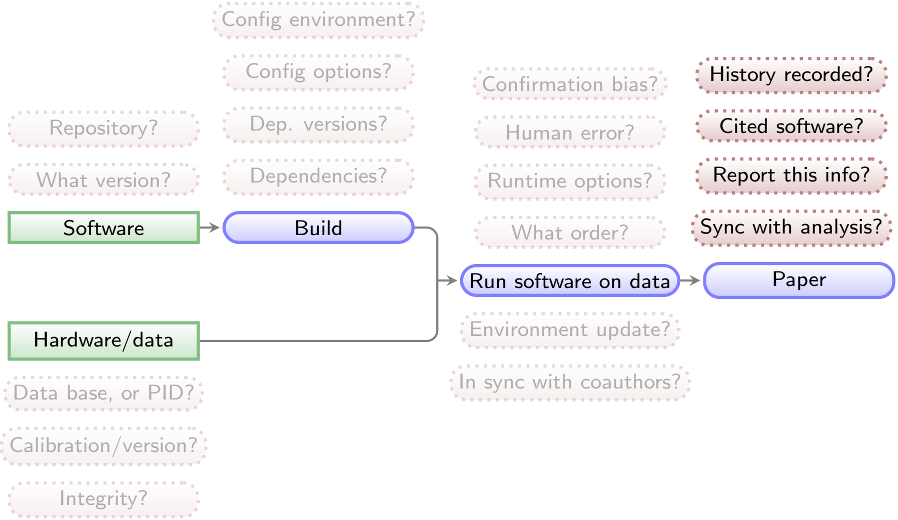
- Unlike a script which always starts from the top, a Makefile starts from the end and steps that don't change will be left untouched (not remade).
- A single *rule* can manage any number of files.
- Make can identify independent steps internally and do them in parallel.
- Make was designed for complex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

```latex
\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,

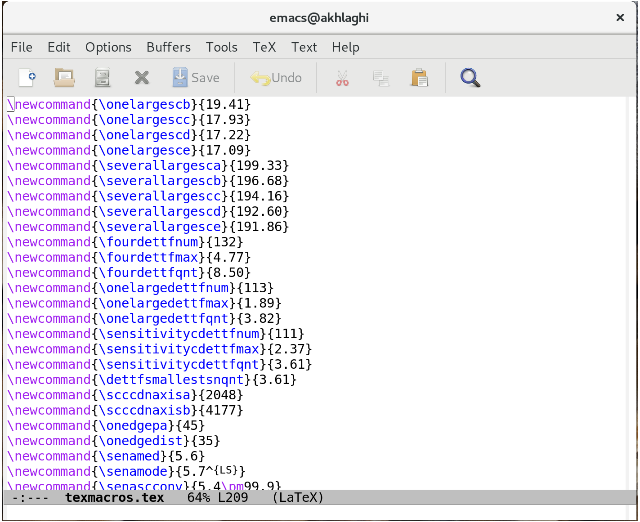
$$\mathrm{S/N}_T = \frac{NF-NS_a}{\sqrt{NF+N\sigma_S^2}} = \frac{\sqrt{N}(F-S_a)}{\sqrt{F+\sigma_S^2}}. \qquad (3)$$

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-

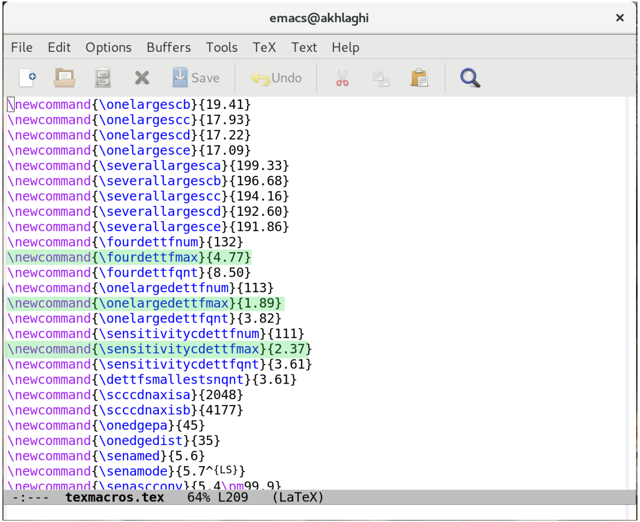
Analysis step results/values concatenated into a single file.

All LaTeX macros come from a single file.



Analysis step results/values concatenated into a single file.

All LaTeX macros come from a single file.




```
\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

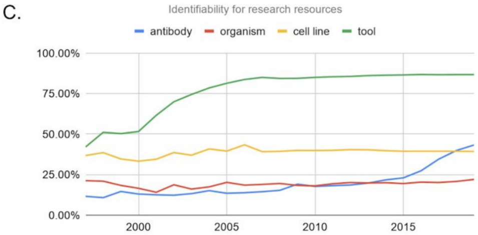
The analysis scripts write/update the $\LaTeX$ macro values automatically.

```
# 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 look at the data lineage to replicate Figure 1C (green/tool) of Menke+2020 (DOI:10.1101/2020.01.15.908111), as done in arXiv:2006.03018 for a demo.

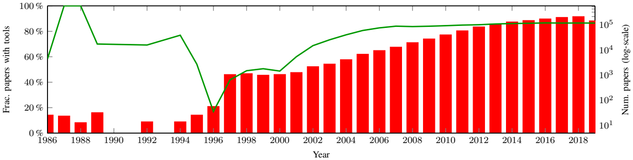
ORIGINAL PLOT

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

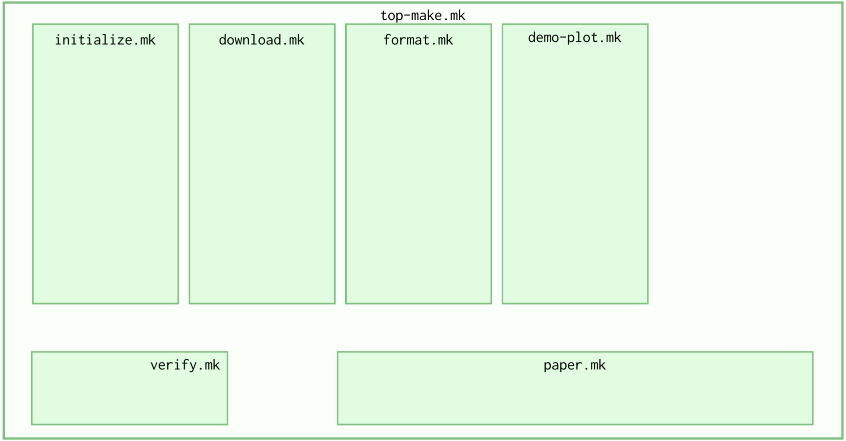


OUR enhanced REPLICATION

The green line is same as above but over their full historical range.
Red histogram is the number of papers studied in each year

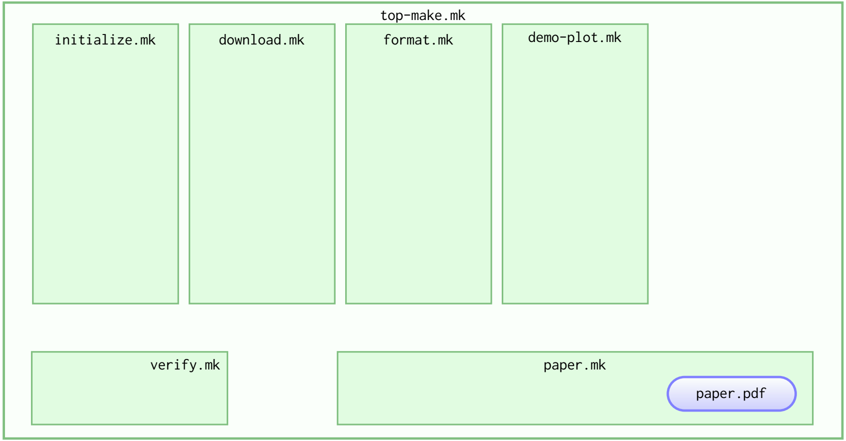


Makefiles (`.mk`) keep contextually separate parts of the project, all imported into `top-make.mk`



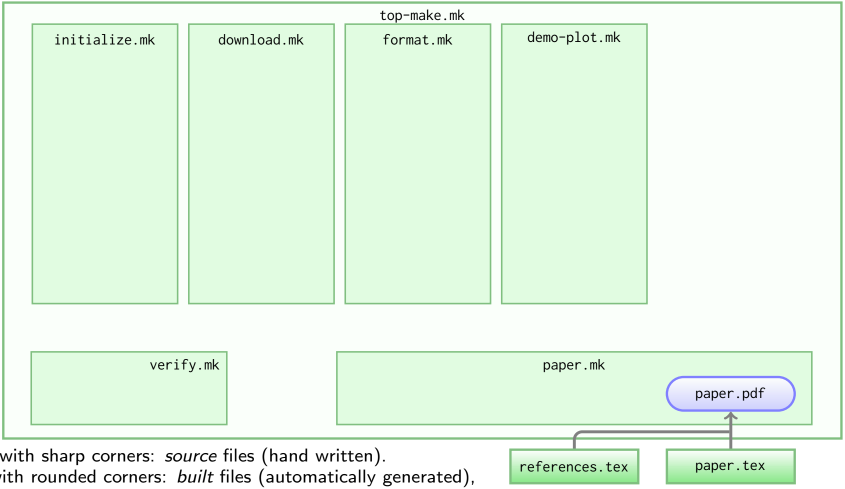
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

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



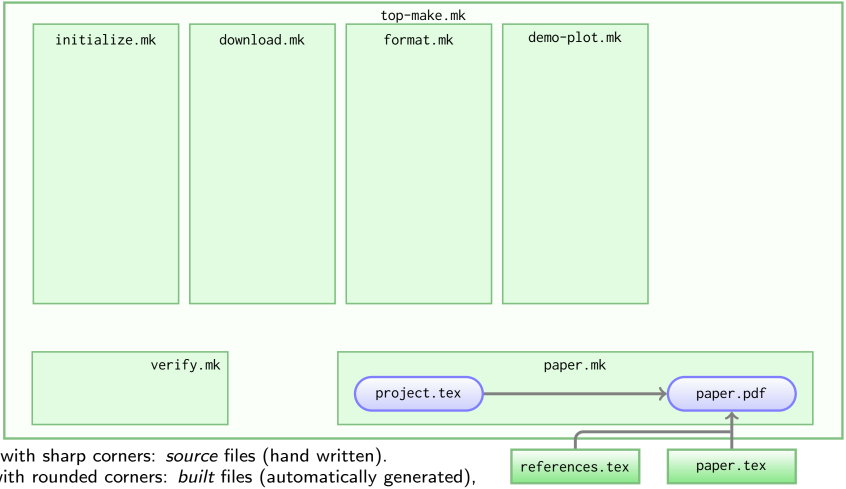
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

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



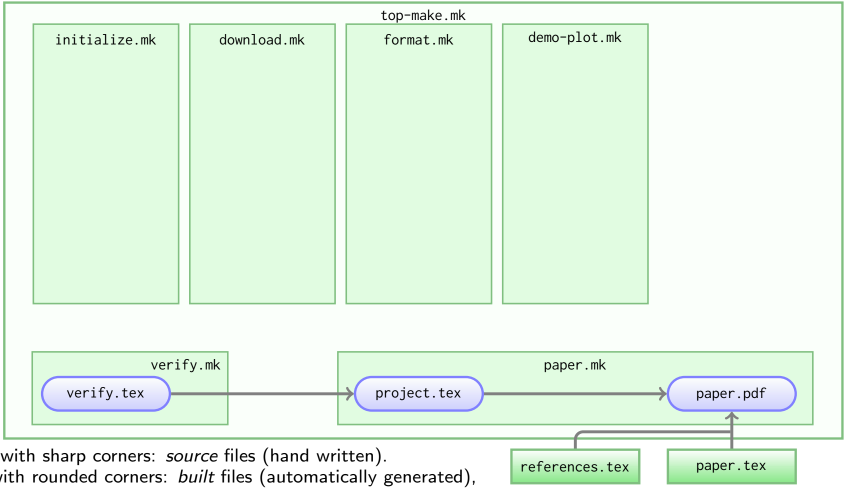
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

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



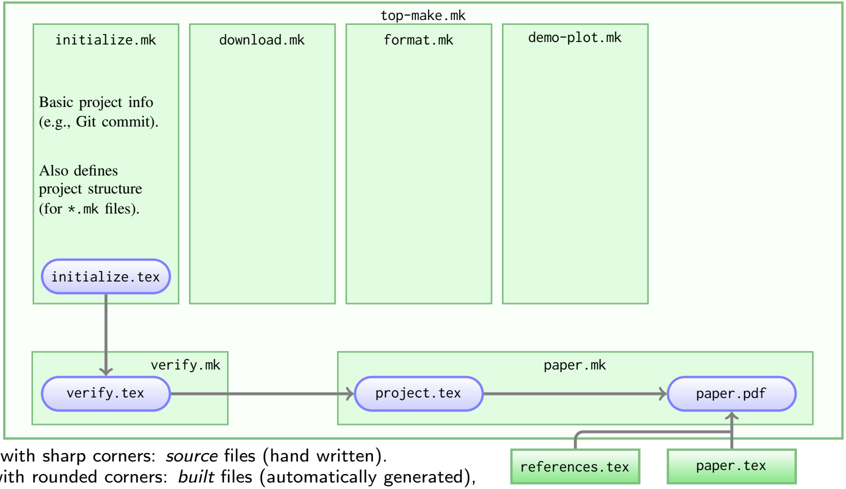
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

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



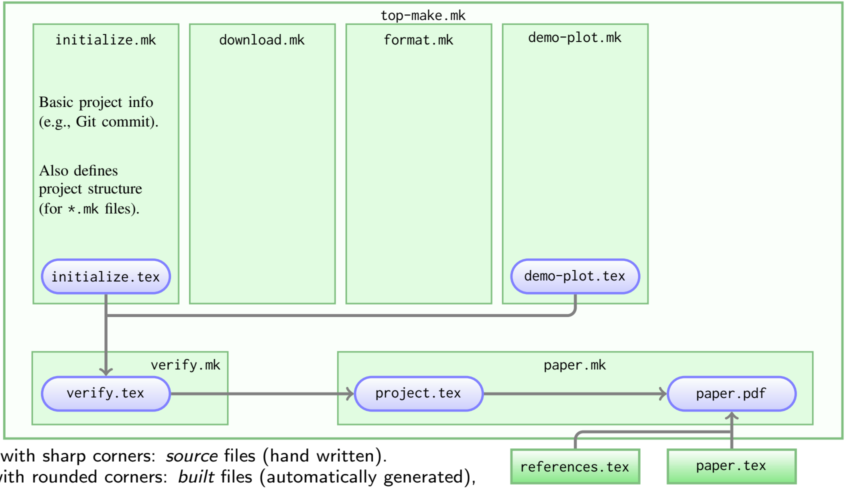
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

Basic project info comes from `initialize.tex`.



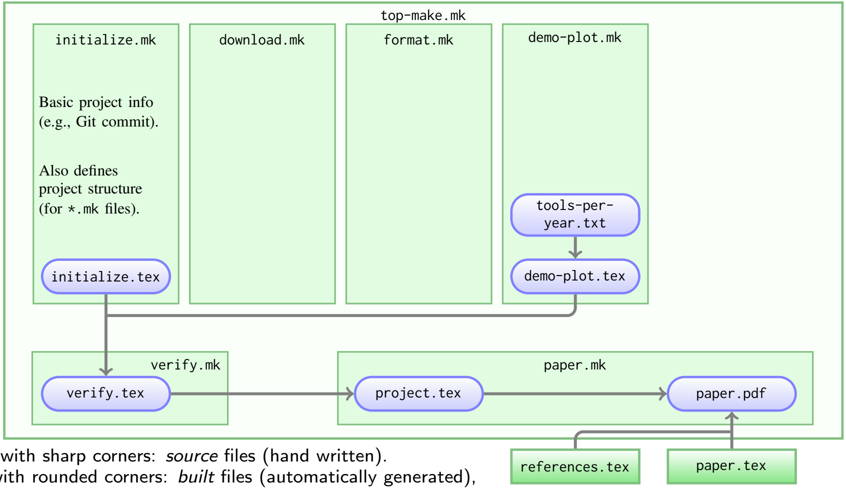
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

The paper includes some information about the plot.



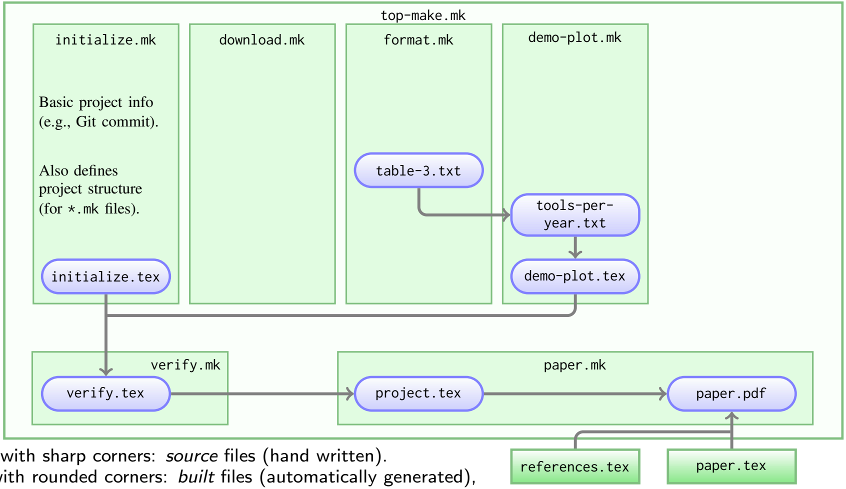
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

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



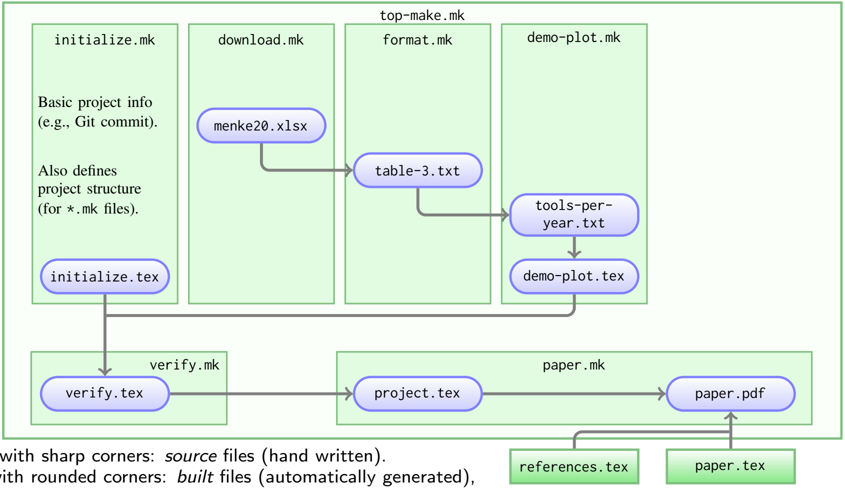
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

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



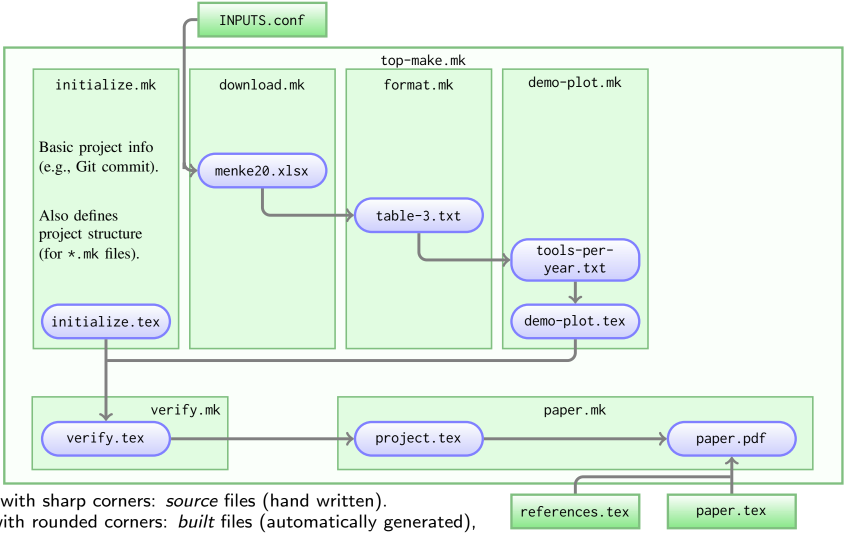
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

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



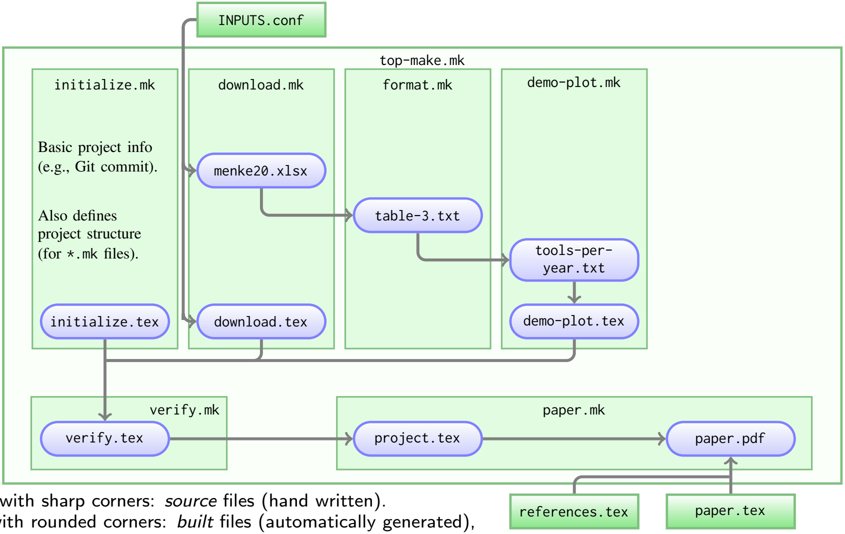
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

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



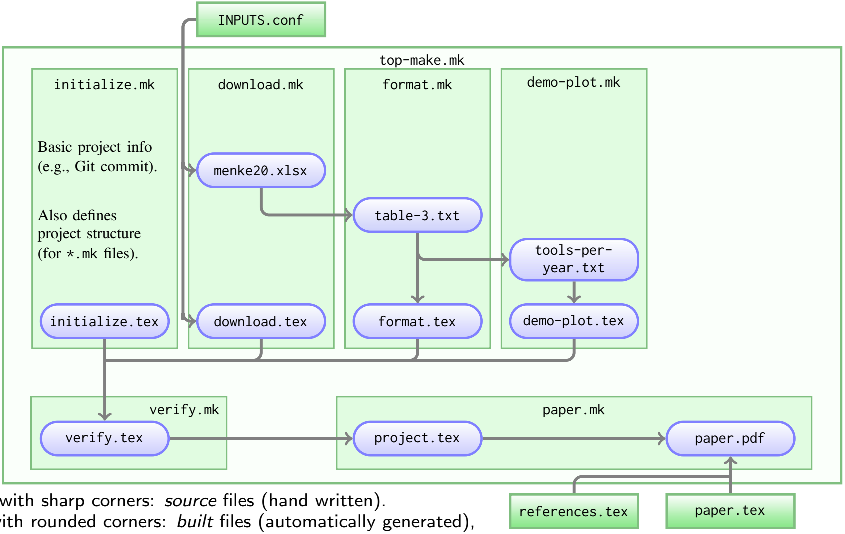
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

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



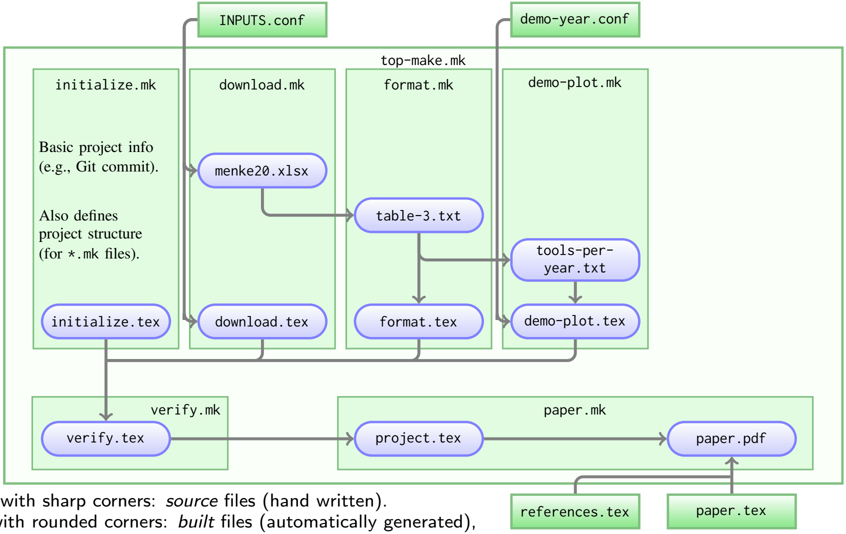
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

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



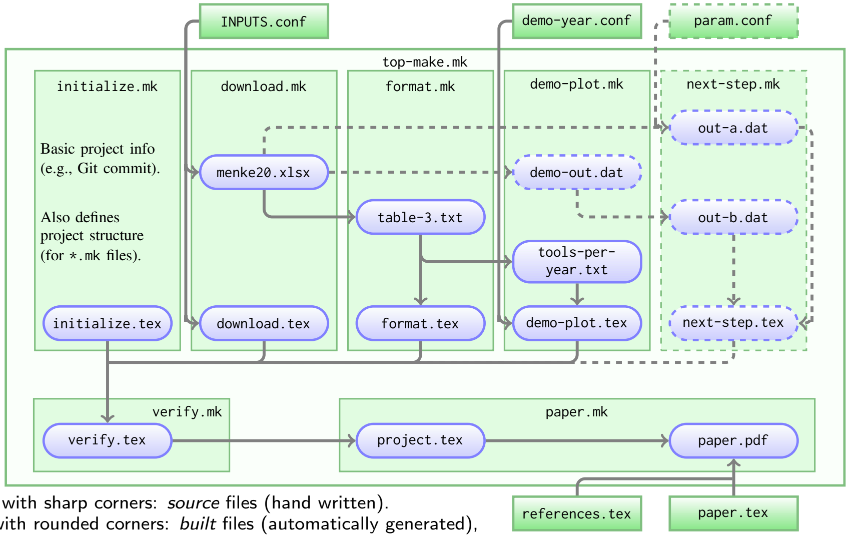
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated), built files are shown in the Makefile that contains their build instructions.

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



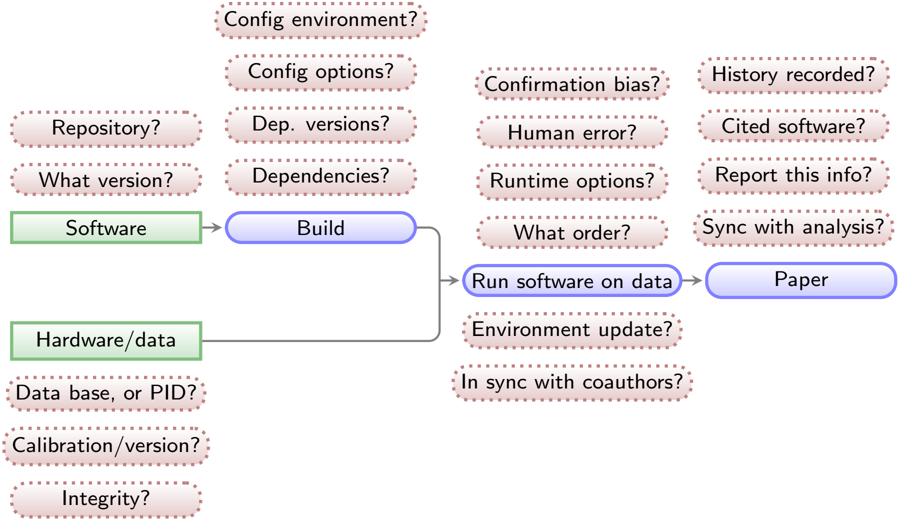
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

It is very easy to expand the project and add new analysis steps (this solution is scalable)



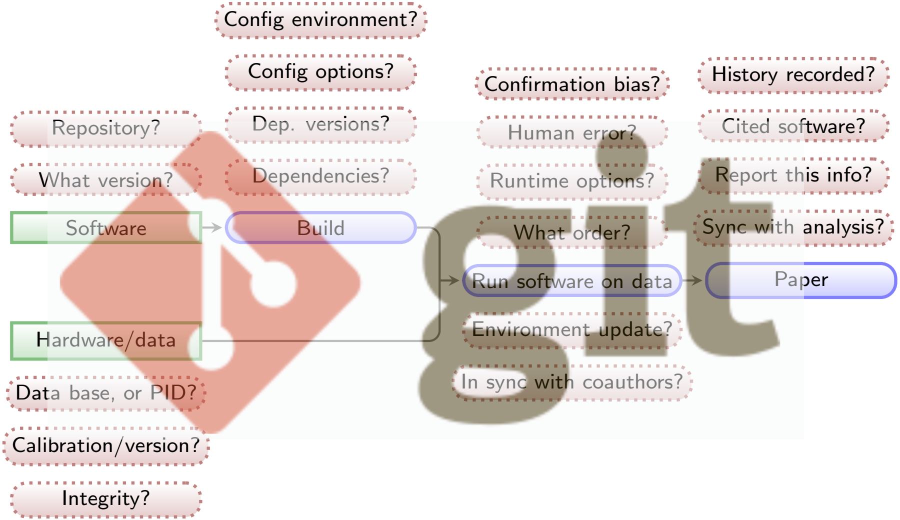
Green boxes with sharp corners: *source* files (hand written).
Blue boxes with rounded corners: *built* files (automatically generated),
built files are shown in the Makefile that contains their build instructions.

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



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

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



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

New projects branch from Maneage

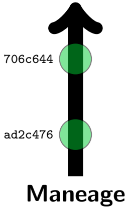
- The project (answers to questions above) will evolve.

New projects branch from Maneage

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

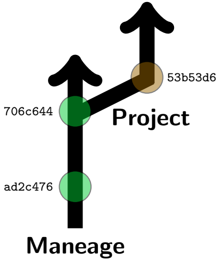
New projects branch from Maneage

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

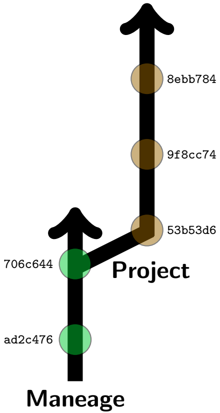


New projects branch from Maneage

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

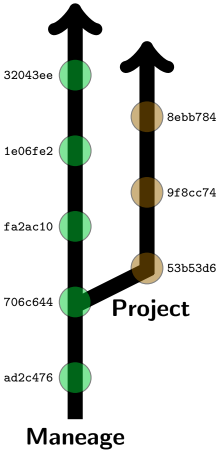


New projects branch from Maneage



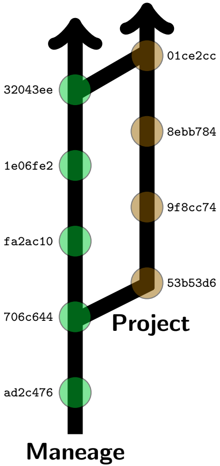
- Each point of project's history is recorded with Git.
- New project: a branch from the template.
  Recall that every commit contains the following:
  - Instructions to download, verify and build software.
  - Instructions to download and verify input data.
  - Instructions to run software on data (do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

New projects branch from Maneage



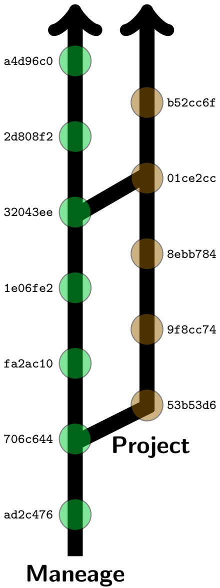
- Each point of project's history is recorded with Git.
- New project: a branch from the template.
  Recall that every commit contains the following:
  - Instructions to download, verify and build software.
  - Instructions to download and verify input data.
  - Instructions to run software on data (do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

New projects branch from Maneage



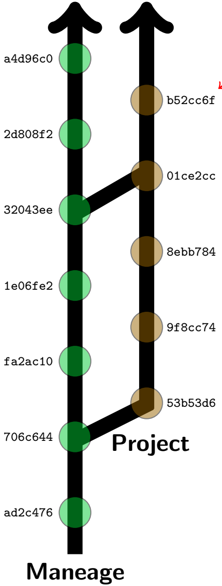
- Each point of project's history is recorded with Git.
- New project: a branch from the template.
  Recall that every commit contains the following:
  - Instructions to download, verify and build software.
  - Instructions to download and verify input data.
  - Instructions to run software on data (do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

New projects branch from Maneage



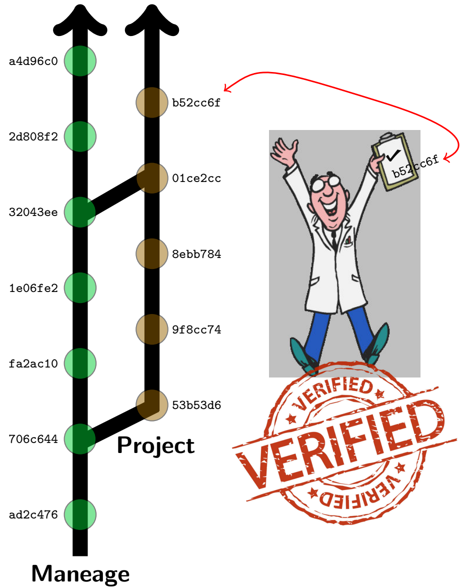
- Each point of project's history is recorded with Git.
- New project: a branch from the template.
  Recall that every commit contains the following:
  - Instructions to download, verify and build software.
  - Instructions to download and verify input data.
  - Instructions to run software on data (do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

New projects branch from Maneage



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

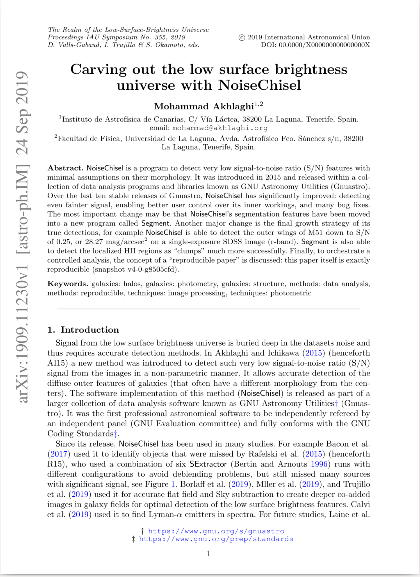
New projects branch from Maneage



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

"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`

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).

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.

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 or arXiv:2006.03018).
- Zenodo: Along with all the input datasets (many Gigabytes) and software (for example zenodo.3872248) and given a unique DOI.
- Software Heritage: to archive the full version-controlled history of the project. (for example swh:1:dir:33fea87068c1612daf011f161b97787b9a0df39fk)

Executing a Maneaged project (for example arXiv:2006.03018)

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

Executing a Maneaged project (for example arXiv:2006.03018)

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

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

Executing a Maneaged project (for example arXiv:2006.03018)

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

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

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

Future prospects...

Adoption of reproducibility by many researchers will enable the following:

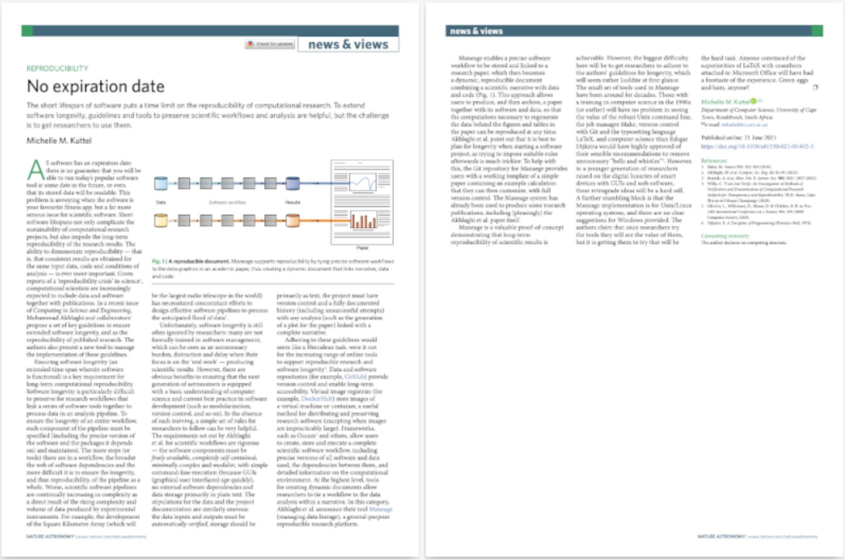
- A repository for education/training (PhD students, or researchers in other fields).
- Easy verification/understanding of other research projects (when necessary).
- Trivially test different steps of others' work (different configurations, software and etc).
- Science can progress incrementally (shorter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

Achievements: 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.

Achievements: "News and Views" in Nature Astronomy (DOI:10.1038/s41550-021-01402-3)



REPRODUCIBILITY

No expiration date

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

Michelle M. Kuttel

All software has an expiration date: there is no guarantee that you will be able to run today's popular software tool at some date in the future, or even that its stored data will be readable. This problem is annoying when the software is your favourite fitness app, but a far more serious issue for scientific software. Short software lifespans not only complicate the sustainability of computational research projects, but also impede the long-term reproducibility of the research results. The ability to demonstrate reproducibility — that is, that consistent results are obtained for the same input data, code and conditions of analysis — is ever more important. Given reports of a 'reproducibility crisis' in science[1], computational scientists are increasingly expected to include data and software together with publications. In a recent issue of *Computing in Science and Engineering*, Mohammad Akhlaghi and collaborators[2] propose a set of key guidelines to ensure extended software longevity, and so the reproducibility of published research. The authors also present a new tool to manage the implementation of these guidelines.

Ensuring software longevity (an extended time span wherein software is functional) is a key requirement for long-term computational reproducibility. Software longevity is particularly difficult to preserve for research workflows that link a series of software tools together to process data in an analysis pipeline. To ensure the longevity of an entire workflow, each component of the pipeline must be specified (including the precise version of the software and the packages it depends on) and maintained. The more steps (or tools) there are in a workflow, the broader the web of software dependencies and the more difficult it is to ensure the longevity, and thus reproducibility, of the pipeline as a whole. Worse, scientific software pipelines are continually increasing in complexity as a direct result of the rising complexity and volume of data produced by experimental instruments. For example, the development of the Square Kilometre Array (which will be the largest radio telescope in the world) has necessitated concomitant efforts to design effective software pipelines to process the anticipated flood of data[3].

Fig. 1 | A reproducible document. Maneage supports reproducibility by tying precise software workflows to the data graphics in an academic paper, thus creating a dynamic document that links narrative, data and code.

Unfortunately, software longevity is still often ignored by researchers: many are not formally trained in software management, which can be seen as an unnecessary burden, distraction and delay when their focus is on the 'real work' — producing scientific results. However, there are obvious benefits to ensuring that the next generation of astronomers is equipped with a basic understanding of computer science and current best practice in software development (such as modularization, version control, and so on). In the absence of such training, a simple set of rules for researchers to follow can be very helpful. The requirements set out by Akhlaghi et al. for scientific workflows are rigorous — the software components must be *freely available, completely self-contained, minimally complex and modular*, with simple command-line execution (because GUIs (graphical user interfaces) age quickly), no external software dependencies and data storage primarily in plain text. The stipulations for the data and the project documentation are similarly onerous: the data inputs and outputs must be *automatically verified*, storage should be primarily as text, the project must have version control and a fully documented history (including unsuccessful attempts) with any analysis (such as the generation of a plot for the paper) linked with a complete narrative.

Adhering to these guidelines would seem like a Herculean task, were it not for the increasing range of online tools to support reproducible research and software longevity[4]. Data and software repositories (for example, GitHub) provide version control and enable long-term accessibility. Virtual image registries (for example, DockerHub) store images of a virtual machine or container, a useful method for distributing and preserving research software (excepting when images are impracticably large). Frameworks, such as Occam[5] and others, allow users to create, store and execute a complete scientific software workflow, including precise versions of all software and data used, the dependencies between them, and detailed information on the computational environment. At the highest level, tools for creating *dynamic documents* allow researchers to tie a workflow to the data analysis within a narrative. In this category, Akhlaghi et al. announce their tool Maneage (managing data lineage), a general-purpose reproducible research platform.

Maneage enables a precise software workflow to be stored and linked to a research paper, which then becomes a dynamic, reproducible document combining a scientific narrative with data and code (Fig. 1). This approach allows users to produce, and then archive, a paper together with its software and data, so that the computations necessary to regenerate the data behind the figures and tables in the paper can be reproduced at any time. Akhlaghi et al. point out that it is best to plan for longevity when starting a software project, as trying to impose suitable rules afterwards is much trickier. To help with this, the Git repository for Maneage provides users with a working template of a simple paper containing an example calculation that they can then customize, with full version control. The Maneage system has already been used to produce some research publications, including (pleasingly) the Akhlaghi et al. paper itself.

Maneage is a valuable proof-of-concept demonstrating that long-term reproducibility of scientific results is achievable. However, the biggest difficulty here will be to get researchers to adhere to the authors' guidelines for longevity, which will seem rather Luddite at first glance. The small set of tools used in Maneage have been around for decades. Those with a training in computer science in the 1990s (or earlier) will have no problem in seeing the value of the robust Unix command line, the job manager Make, version control with Git and the typesetting language LaTeX, and computer science titan Edsgar Dijkstra would have highly approved of their sensible recommendations to remove unnecessary "bells and whistles"[6]. However, to a younger generation of researchers raised on the digital luxuries of smart devices with GUIs and web software, these retrograde ideas will be a hard sell. A further stumbling block is that the Maneage implementation is for Unix/Linux operating systems, and there are no clear suggestions for Windows provided. The authors claim that once researchers try the tools they will see the value of them, but it is getting them to try that will be the hard task. Anyone convinced of the superiorities of LaTeX with coauthors attached to Microsoft Office will have had a foretaste of the experience. Green eggs and ham, anyone?

Michelle M. Kuttel ✉
Department of Computer Science, University of Cape Town, Rondebosch, South Africa.
✉e-mail: mkuttel@cs.uct.ac.za

Published online: 21 June 2021
https://doi.org/10.1038/s41550-021-01402-3

References

1. Baker, M. Nature 533, 452–454 (2016).
2. Akhlaghi, M. et al. Comput. Sci. Eng. 23, 82–91 (2021).
3. Bonaldi, A. et al. Mon. Not. R. Astron. Soc. 500, 3821–3837 (2021).
4. Willis, C. *Trust, but Verify: An Investigation of Methods of Verification and Dissemination of Computational Research Artifacts for Transparency and Reproducibility*. Ph.D. thesis, Univ. Illinois at Urbana-Champaign (2020).
5. Oliveira, L., Wilkinson, D., Mossé, D. & Childers, B. R. in *Proc. 14th International Conference on e-Science* 384–395 (IEEE Computer Society, 2018).
6. Dijkstra, E. *A Discipline of Programming* (Prentice-Hall, 1976).

Competing interests

The author declares no competing interests.

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

Summary:

Maneage and its principles are described in arXiv:2006.03018. It is a customizable template that will do the following steps/instructions (all in simple plain text files).

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