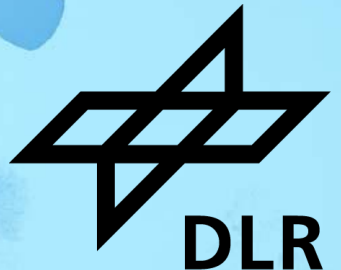


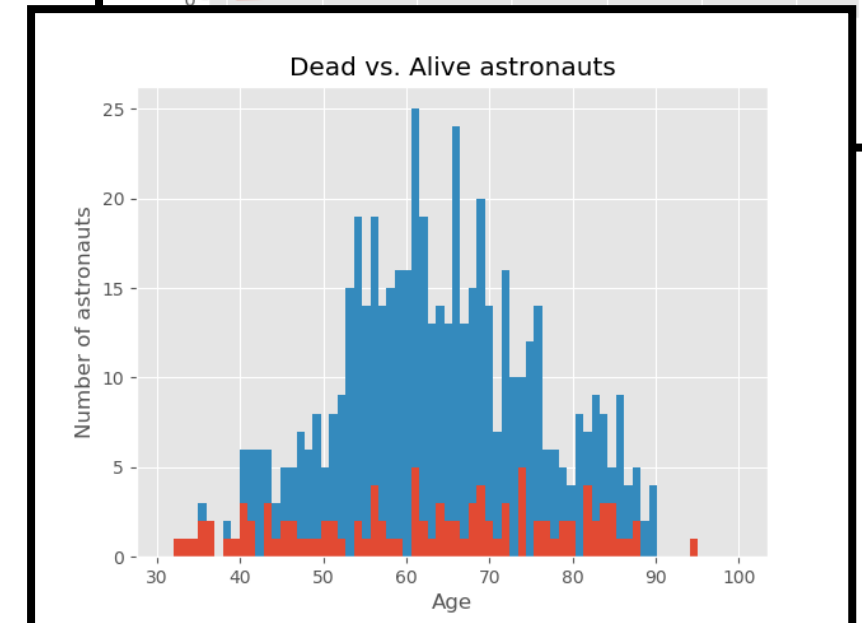
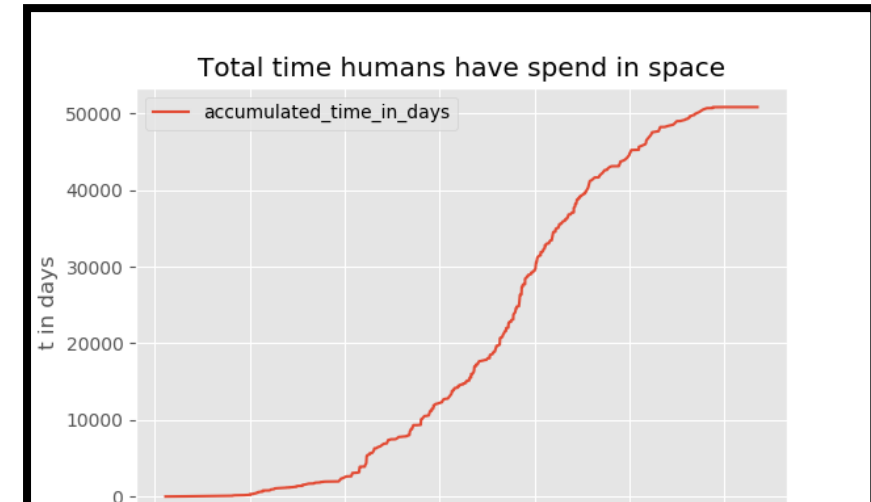
BASICS OF SOFTWARE PUBLICATION

Carina Haupt
carina.haupt@dlr.de
@caha42(@scholar.social)



Example: Astronaut Analysis

- [Astronauts Analysis](#) is a data publication consisting of:
 - Data set
 - Analysis script written in Python using [pandas](#) and [matplotlib](#)
 - Result plots
- **Scenario:**
 - I created it on my own as part of my job.
 - I want to make its reuse as easy as possible and make it available under an open source license.



Make your code reusable



- Step 1: Put your code under version control
 - Step 2: Make sure that your code is in a sharable state
 - Step 3: Add essential documentation
-
- Step 4: Add a license
 - Step 5: Release your code

Essential aspects
which you should
try to already
address for
“internal” software!

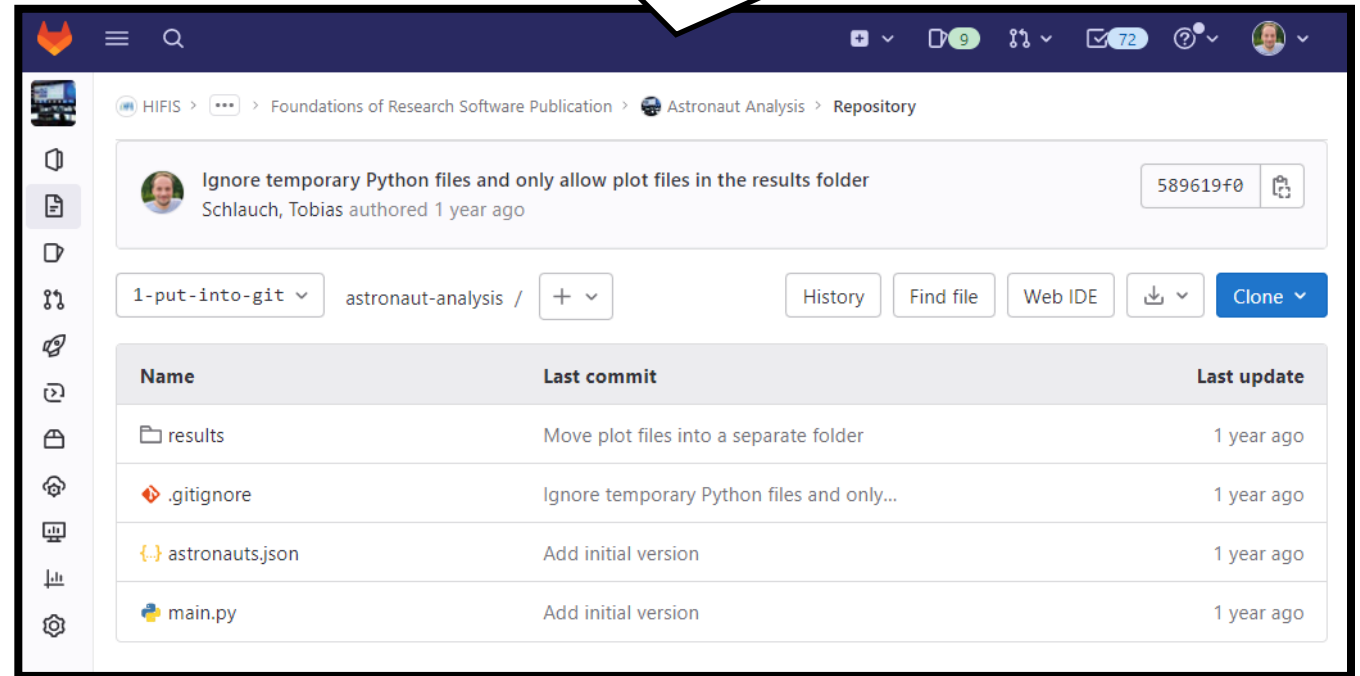
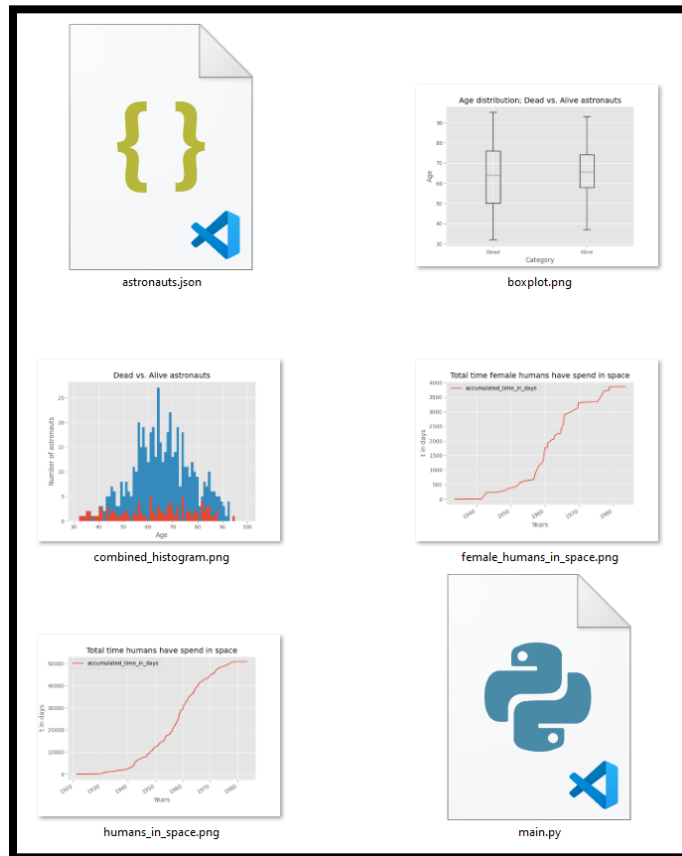
Step 1: Put Your Code Under Version Control

Where Should I Store My Code?



Minimum: Use a local Git repository + backup

Recommended: Use a code collaboration platform



Step 1: Put Your Code Under Version Control

What Belongs in the Repository?



- **Everything to make a usable version of your code** such as:
 - Source code, documentation, build scripts, test cases, configuration files, input data, ...
- **Avoid adding generated files** such as:
 - Third-party libraries, generated binaries, ...
- **How to handle large (data) files?**
 - Available could be [git-lfs](#), [git-annex](#), [Datalad](#) or your research data management publication repository
- **Please note:**
 - Details depend on the “product” that you manage in the Git repository
 - **.gitignore files** helps you to control what goes into your repository. See also <https://gitignore.io/> for templates.

Step 2: Make Sure That Your Code Is in a Sharable State

General Hints



- Make sure others can run your code:
 - No dependencies on internal resources (servers, storage, licensed software, ...)
 - No absolute paths
 - Clearly state dependencies + provide required build / installation scripts (e.g.: [pip-tools](#), [poetry](#)) => crucial aspect of reproducibility
- Organize files in a suitable directory structure (e.g.: [Python Application Layouts](#), [Good Data Practices](#))
- Do not share sensitive data such as passwords, user accounts, SSH keys, internal IP addresses, etc. (e.g.: [gitleaks](#))
- Orientate on standards of your domain / community

Step 2: Make Sure That Your Code Is in a Sharable State

Improve Your Code Style and Structure



- Strive for understandable code:
 - Apply a code style – consistency is more important than convenience (e.g.: [PEP8](#))
 - Use a consistent and light code layout
 - Structure your code in suitable "building blocks" such as functions
 - Use specific and appropriate names for all artifacts
 - Provide sufficient level of code comments
- Read code of others for inspiration
- Try to do pair programming and reviews (even if it is [with your rubber duck](#))

Step 2: Make Sure That Your Code Is in a Sharable State

Think About Testing and Automation



- Small tests are done easily but already show effect:
 - Code linters and checkers help to find poor code snippets and help to enforce coding styles (e.g.: [flake8](#), [black](#))
 - Automated tests work as an executable documentation (e.g.: [pytest](#))
- Tests offer a good starting point for your automation efforts!

Step 3: Add Documentation


General Hints



- **Mind your target groups:**
 - **Typical perspectives:** Users, contributors
 - **Users:** Installation / usage instructions, tutorials, support channels, ...
 - **Contributors:** Contribution guidelines, technical overview, ...
- **Think about adding typical documentation files** such as:
 - README (project front page), CONTRIBUTING (contributions guidelines),
CODE_OF_CONDUCT (communication rules), LICENSE (license information),
CHANGELOG (major changes), CITATION (citation metadata)
- **Please note:**
 - [Markdown](#) or another markup language is quite often used to write documentation
 - Usually, you will need additional documentation, for example, in a `docs` directory (e.g.: [Sphinx](#), [MkDocs](#))

Recommendations from the workshop

“Foundations of Research Software Publication”

- Step 1: Put your code under version control
 - Step 2: Make sure that your code is in a sharable state
 - Step 3: Add essential documentation
- A green circular icon with a white checkmark inside, indicating a positive status or completion.
- Step 4: Add a license
 - Step 5: Make your code citable
 - Step 6: Release your code

Goal – Provide a citable release



“The data set and the analysis code has been published separately [1].”

References:

[1] Schlauch, T. (2024). Astronaut Analysis (2024-03-20) [Data set]. Zenodo. <https://doi.org/10.5072/zenodo.402763>



- See also: [Software Citation Principles](#)


- ✓ **Source code repository:** We keep the repository in our own GitLab instance.
- ✓ **Release versioning style:** We use [calendar versioning](#).
- ✓ **Git tag style:** Tags have the name of the release version number.
- ✓ **Release package format:** The release package is a “snapshot” of the repository content.
- ✓ **Citation metadata:** Part of the source code repository in the Citation File Format.
- ✓ **Publication repository:** We only use Zenodo to publish/archive our releases.

Astronaut Analysis Release








**License
information
for code, data,
results
properly
annotated via
[REUSE](#)**

**Astronaut Analysis** 


















The repository contains the example code used in this workshop.

 DOI [10.5281/zenodo.10001813](https://doi.org/10.5281/zenodo.10001813)  Latest Release **1.0.0**

 **Add changelog and reference it**
Tobias Schlauch authored 2 years ago  [8a05544e](#) 

main ▾ astronaut-analysis / + ▾ History Find file Edit ▾ Code ▾

 README  LICENSE  CHANGELOG  CI/CD configuration  Add CONTRIBUTING  Add Kubernetes cluster  Configure Integrations

Name	Last commit	Last update
 LICENSES	Add license and copyright information	2 years ago
 code	Add license and copyright information	2 years ago
 data	Add license and copyright information	2 years ago
 results	Add license and copyright information	2 years ago
 .gitignore	Add license and copyright information	2 years ago
 .gitlab-ci.yml	Add license and copyright information	2 years ago
 CHANGELOG.md	Add changelog and reference it	2 years ago
 LICENSE.md	Add license and copyright information	2 years ago

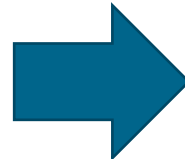
**Release
1.0.0** marked
as **Git tag** in
the repository

Astronaut Analysis Release



DOI 10.5072/zenodo.402763

```
cff-version: 1.2.0
title: Astronaut Analysis
message: >-
  If you use this analysis, please cite it using the
  metadata from this file.
type: dataset
authors:
  - given-names: Tobias
    family-names: Schlauch
    email: tobias.schlauch@dlr.de
    affiliation: German Aerospace Center (DLR)
    orcid: 'https://orcid.org/0000-0001-8760-8913'
identifiers:
  - type: doi
    value: 10.5072/zenodo.402762
    description: Conceptual DOI
```



Published March 20, 2024 | Version 2024-03-20

Astronaut Analysis

Schlauch, Tobias¹

1. German Aerospace Center

This analysis is based on publicly available astronauts data from [Wikidata](#). In this context, we investigated aspects such as time humans spent in space as well as the age distribution of the astronauts.

Files

astronaut-analysis-2024-03-20.zip

astronaut-analysis-2024-03-20.zip

astronaut-analysis-data-2024-03-20

- .gitignore 330 Bytes
- gitlab-ci.yml 271 Bytes
- CITATION.cff 962 Bytes
- LICENSE.md 659 Bytes
- LICENSES

Manage

Edit

New version

Share

0 VIEWS 0 DOWNLOADS

Show more details

Versions

Version 2024-03-20 Mar 20, 2024

10.5072/zenodo.402763

Cite all versions? You can cite all versions by using the DOI 10.5072/zenodo.402762. This DOI represents all versions, and will always resolve to the latest one. [Read more](#).

External resources

Citable release:

- Citation metadata in [Citation File Format](#)
- DOI via [Zenodo](#)

Rights

License

Apache License 2.0

Creative Commons Attribution 4.0 International

Creative Commons Zero v1.0 Universal

Copyright

Copyright © 2018 German Aerospace Center (DLR)

Citation

Schlauch, T. (2024). Astronaut Analysis (2024-03-20) [Data set]. Zenodo. <https://doi.org/10.5072/zenodo.402763>

Style APA

Export

JSON Export

Summary



- Step 1: Put your code under version control
- Step 2: Make sure that your code is in a sharable state
- Step 3: Add essential documentation
- Step 4: Add a license
- Step 5: Release your code



Thank you!

What are your Questions?

Email: carina.haupt@dlr.de

Mastodon:

<https://scholar.social/@caha42>

- Content created based on DLR/HIFIS training “Foundations of Research Software Publication” and example project “Astronaut Analysis”
 - <https://codebase.helmholtz.cloud/hifis/software/education/hifis-workshops/foundations-of-research-software-publication/workshop-materials>
 - <https://codebase.helmholtz.cloud/hifis/software/education/hifis-workshops/foundations-of-research-software-publication/astronaut-analysis>

Copyright and License Information



All content is © German Aerospace Center and licensed under [Attribution 4.0 International \(CC-BY-4.0\)](#) with the following exceptions:

- DLR logo, slide layout, © German Aerospace Center. All rights reserved.
- “Open Source vs. Closed Source”, slide 2, @ Patrick Hochstenbach. [CC0](#).
- Philae landing on comet 67 P/Churyumov-Gerasimenko, slide 16, © German Aerospace Center. [CC-BY-3.0](#).