The Journal of Open Source Software

Pooch: A friend to fetch your data files

Leonardo Uieda[1], Santiago Rubén Soler[2,3], Rémi Rampin[4], Hugo van Kemenade[5], Matthew Turk[6], Daniel Shapero[7], Anderson Banihirwe[8], and John Leeman[9]

1 Department of Earth, Ocean and Ecological Sciences, School of Environmental Sciences, University of Liverpool, UK **2** Instituto Geofísico Sismológico Volponi, Universidad Nacional de San Juan, Argentina **3** CONICET, Argentina **4** New York University, USA **5** Independent (Non-affiliated) **6** University of Illinois at Urbana-Champaign, USA **7** Polar Science Center, University of Washington Applied Physics Lab, USA **8** The US National Center for Atmospheric Research, USA **9** Leeman Geophysical, USA

DOI: 10.21105/joss.01943

Software

- Review
- Repository
- Archive

Editor: Daniel S. Katz

Reviewers:

- @hmaarrfk
- @martindurant

Submitted: 02 December 2019
Published: 17 January 2020

Summary

Scientific software is usually created to acquire, analyze, model, and visualize data. As such, many software libraries include sample datasets in their distributions for use in documentation, tests, benchmarks, and workshops. A common approach is to include smaller datasets in the GitHub repository directly and package them with the source and binary distributions (e.g., scikit-learn (Pedregosa et al., 2011) and scikit-image (Van der Walt et al., 2014) do this). As data files increase in size, it becomes unfeasible to store them in GitHub repositories. Thus, larger datasets require writing code to download the files from a remote server to the user's computer. The same problem is faced by scientists using version control to manage their research projects. While downloading a data file over HTTPS can be done easily with modern Python libraries, it is not trivial to manage a set of files, keep them updated, and check for corruption. For example, scikit-learn (Pedregosa et al., 2011), Cartopy (Met Office, n.d.), and PyVista (Sullivan & Kaszynski, 2019) all include code dedicated to this particular task. Instead of scientists and library authors recreating the same code, it would be best to have a minimalistic and easy to set up tool for fetching and maintaining data files.

Pooch is a Python library that fills this gap. It manages a data *registry* (containing file names, SHA-256 cryptographic hashes, and download URLs) by downloading files from one or more remote servers and storing them in a local data cache. Pooch is written in pure Python and has minimal dependencies. It can be easily installed from the Python Package Index (PyPI) and conda-forge on a wide range of Python versions: 2.7 (up to Pooch 0.6.0) and from 3.5 to 3.8. The integrity of downloads is verified by comparing the file's SHA-256 hash with the one stored in the data registry. This is also the mechanism used to detect if a file needs to be re-downloaded due to an update in the registry. Pooch is meant to be a drop-in replacement for the custom download code that users have already written (or are planning to write). In the ideal scenario, the end-user of a software package should not need to know that Pooch is being used. Setup is as easy as calling a single function (`pooch.create`), including setting up an environment variable for overwriting the data cache path and versioning the downloads so that multiple versions of the same package can coexist in the same machine. For example, this is the code required to set up a module `datasets.py` that uses Pooch to manage data downloads:

```
import pooch

# Get the version string from the project
```

```
from . import version

# Create a new instance of pooch.Pooch
GOODBOY = pooch.create(
    # Cache path using the default for the operating system
    path=pooch.os_cache("myproject"),
    # Base URL of the remote data server (for example, on GitHub)
    base_url="https://github.com/me/myproject/raw/{version}/data/",
    # PEP 440 compliant version number (added to path and base_url)
    version=version,
    # An environment variable that overwrites the path
    env="MYPROJECT_DATA_DIR",
)
# Load the registry from a simple text file.
# Each line has: file_name sha256 [url]
GOODBOY.load_registry("registry.txt")

def fetch_some_data():
    # Get the path to the data file in the local cache
    # If it's not there or needs updating, download it
    fname = GOODBOY.fetch("some-data.csv")
    # Load it with NumPy/pandas/xarray/etc.
    data = pandas.read_csv(fname)
    return data
```

Pooch is designed to be extended: users can plug in custom download functions and post-download processing functions. For example, a custom download function could fetch files from a password-protected FTP server (the default is HTTP/HTTPS or anonymous FTP) and a processing function could decrypt a file using a user-defined password once the download is completed. We include ready-made download functions for HTTP and FTP (including basic authentication) as well as processing functions for unpacking archives (zip or tar) and decompressing files (gzip, lzma, and bzip2).

To the best of the authors' awareness, the only other Python software with some overlapping functionality are Intake and fsspec (which is used by Intake). The fsspec library provides a unified interface for defining file systems and opening files, regardless of where the files are located (local system, HTTPS/FTP servers, Amazon S3, Google Cloud Storage, etc). fsspec implements similar download and caching functionality to the one in Pooch, but has a wider range of download methods available. In the future, fsspec could be used as a backend to expand Pooch's download capabilities beyond HTTPS and FTP. Intake manages data catalogues (with download locations and extensive metadata), data download and caching, data loading, visualization, and browsing. It has built-in capabilities for loading data into standard containers, including NumPy, pandas, and xarray. While Intake and fsspec are powerful and highly configurable tools, we argue that Pooch's strong points are its simplicity, straight-forward documentation, and focus on solving a single problem.

The Pooch API is stable and has been field-tested by other projects: MetPy (May et al., n.d.), Verde (Uieda, 2018), RockHound (Uieda & Soler, 2019), predictatops (Gosses, 2019), and icepack (Shapero, Lilien, Ham, & Hoffman, 2019). Pooch is also being implemented as the download manager for scikit-image (GitHub pull request number 3945), which will allow the project to use larger sample data while simultaneously reducing the download size of source and binary distributions.

Acknowledgements

We would like to thank all of the volunteers who have dedicated their time and energy to build the open-source ecosystem on which our work relies. The order of authors is based on number of commits to the GitHub repository. A full list of all contributors to the project can be found on the GitHub repository.

References

Gosses, J. (2019). JustinGOSSES/predictatops: V0.0.4. Zenodo. doi:10.5281/ZENODO.1450596

May, R. M., Arms, S. C., Marsh, P., Bruning, E., Leeman, J. R., Goebbert, K., Thielen, J. E., et al. (n.d.). *MetPy: A Python package for meteorological data*. Boulder, Colorado: Unidata. doi:10.5065/D6WW7G29

Met Office. (n.d.). *Cartopy: A cartographic Python library with a Matplotlib interface*. Exeter, Devon. Retrieved from https://scitools.org.uk/cartopy

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Shapero, D., Lilien, D., Ham, D. A., & Hoffman, A. (2019). Icepack/icepack: Icepack: Glacier flow modeling with the finite element method in Python. Zenodo. doi:10.5281/ZENODO.3542092

Sullivan, C. B., & Kaszynski, A. (2019). PyVista: 3D plotting and mesh analysis through a streamlined interface for the Visualization Toolkit (VTK). *Journal of Open Source Software*, *4*(37), 1450. doi:10.21105/joss.01450

Uieda, L. (2018). Verde: Processing and gridding spatial data using Green's functions. *Journal of Open Source Software*, *3*(30), 957. doi:10.21105/joss.00957

Uieda, L., & Soler, S. R. (2019). Rockhound: Download geophysical models/datasets and load them in Python. Zenodo. doi:10.5281/ZENODO.3086002

Van der Walt, S., Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., Gouillart, E., et al. (2014). scikit-image: image processing in Python. *PeerJ*, *2*, e453. doi:10.7717/peerj.453