The Journal of Open Source Software

SounderPy: An atmospheric sounding visualization and analysis tool for Python

Kyle J. Gillett[1]

1 John D. Odegard College of Aerospace Sciences, University of North Dakota, United States

DOI: DOIunavailable

Software

- Review
- Repository
- Archive

Editor: Pending Editor

Reviewers:

- @Pending Reviewers

Submitted: N/A
Published: N/A

Summary

SounderPy is an open-source Python package for retrieving, processing, and visualizing atmospheric sounding data. Designed for simplicity and reliability, SounderPy aims to provide a uniform and intuitive method for sounding analysis across various data sources.

Statement of need

Meteorological data can originate from many diverse sources and are often stored in varying file formats and structured in various ways, posing challenges for consistent and efficient data processing. This wide range of variability complicates the comprehensive analysis of atmospheric properties needed in describing the past, current, and future state of the atmosphere.

Standardizing the analysis of meteorological sounding data ensures that meaningful comparisons can be drawn across a variety of datasets. From this, reliable statistics and analogs can be developed, improving pattern recognition and situational context. For example, comparing numerical weather prediction (NWP) output to observations allows forecasters to recognize recurring patterns and establish connections between past and future events, and allows atmospheric-modelers to evaluate the accuracy and reliability of NWP products.

SounderPy addresses these challenges by providing simple, uniform access to multiple data sources, including National Weather Service radiosonde observations (RAOBs), Aircraft Communications Addressing and Reporting System (ACARS) data, NWP forecast data, and several reanalysis datasets. Each of these data types come with unique file formats and data-structures. However, after requesting data, SounderPy will process and output the data in a normalized Python dictionary, coined a "SounderPy `clean_data` dictionary." Doing so means that every data source ends up in a uniform format for analysis and visualization.

High-level API overview

Simple, intuitive classes and functions make SounderPy's API easy to understand and use. The procedure for creating sounding figures requires only a few lines of straightforward API calls that first retrieve data from a source, output the `clean_data` dictionary and plot the data on a figure. While internally this involves sophisticated data retrieval & processing methods, extensive manipulations & calculations, and intricate plotting routines, this process can be completed by the user in just two lines of code for most integrated data sources. The simplicity of these "tools" in SounderPy's "toolbox" allows quick and easy use for forecasters, researchers, students, and hobbyists alike. After importing the library, just two lines of code will create Figure 1. On top of the many integrated data sources, SounderPy also allows users to utilize their own custom data sources. So long as a user's custom data can be manually sorted into a `clean_data` dictionary, it can be used with SounderPy's analysis and plotting functionality.

The Journal of Open Source Software

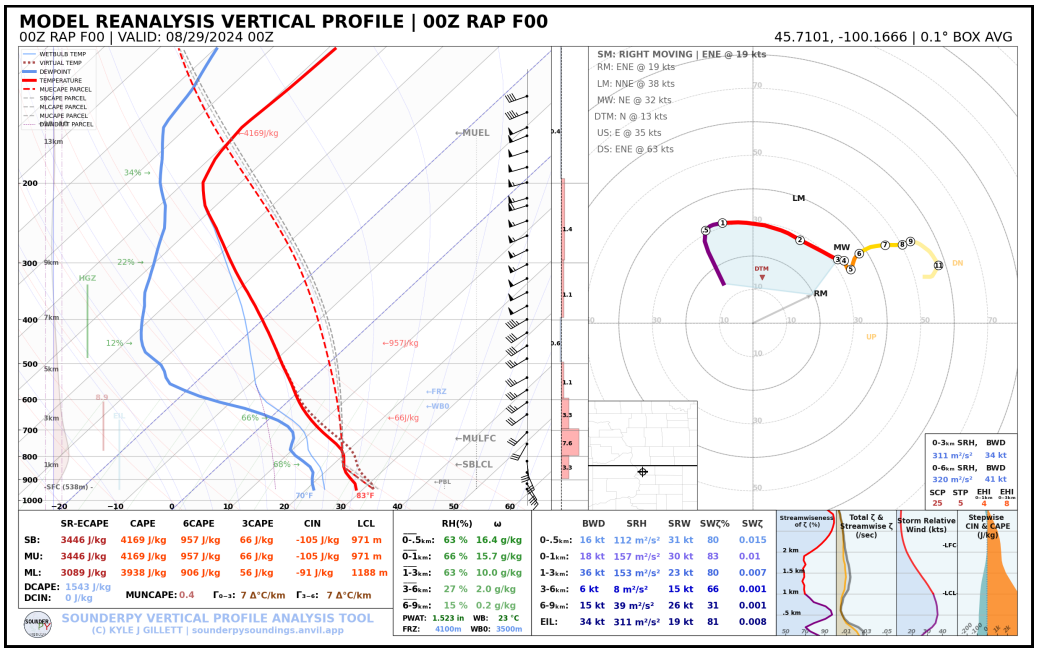

Figure 1: Figure 1: A sounding figure of NCEP RAP reanalysis data for a severe weather event in northern South Dakota on August 28th, 2024

Acknowledgements

The development of SounderPy relies on the robust functionality provided by several foundational Python libraries, including MetPy (May et al., 2022), NumPy (Harris et al., 2020), Matplotlib (Hunter, 2007), xarray (Hoyer & Hamman, 2017), Cartopy (Met Office, 2010 - 2015), SHARPpy (Blumberg et al., 2017), SciPy (Virtanen et al., 2020), and others. The author gratefully acknowledges the valuable contributions of individuals who have enhanced this project, including Scott Thomas (National Weather Service), Ethan O'Neill (University of North Carolina at Charlotte), Brian Rakoczy (Ohio State University), Daryl Herzmann (Iowa State University), Amelia R.H. Urquhart (University of Oklahoma), Ryan Vandersmith, and many others.

This project also extends its graditude to the researchers, institutions, and individuals who have utilized SounderPy in their work and publications, driving its growth and application within the meteorological community.

References

Blumberg, W. G., Halbert, K. T., Supinie, T. A., Marsh, P. T., Thompson, R. L., & Hart, J. A. (2017). SHARPpy: An open-source sounding analysis toolkit for the atmospheric sciences. *Bulletin of the American Meteorological Society*, *98*(8), 1625–1636. https://doi.org/10.1175/BAMS-D-15-00309.1

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hoyer, S., & Hamman, J. (2017). Xarray: N-d labeled arrays and datasets in python. *Journal of Open Research Software*, *5*(1), 10. https://doi.org/10.5281/zenodo.59499

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

May, R. M., Goebbert, K. H., Thielen, J. E., Leeman, J. R., Camron, M. D., Bruick, Z., Bruning, E. C., Manser, R. P., Arms, S. C., & Marsh, P. T. (2022). MetPy: A meteorological python library for data analysis and visualization. *Bulletin of the American Meteorological Society*, *103*(10), E2273–E2284. https://doi.org/10.1175/BAMS-D-21-0125.1

Met Office. (2010 - 2015). *Cartopy: A cartographic python library with a matplotlib interface*. https://scitools.org.uk/cartopy

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2