




STRAUSS: Sonification Tools & Resources for Analysis Using Sound Synthesis

James W. Trayford ^{1¶}, Samantha Youles ^{1*}, Chris Harrison ^{2*}, Rose Shepherd ^{2*}, and Nicolas Bonne ^{1*}

¹ Institute of Cosmology and Gravitation, University of Portsmouth, Dennis Sciama Building, Burnaby Road, Portsmouth PO1 3FX, UK ² School of Mathematics, Statistics and Physics, Newcastle University, NE1 7RU, UK ¶ Corresponding author * These authors contributed equally.

DOI: [10.21105/joss.07875](https://doi.org/10.21105/joss.07875)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Juanjo Bazán](#) 

Reviewers:

- [@manwithfeathers](#)
- [@johicasado](#)

Submitted: 17 February 2025

Published: 15 April 2025

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

Summary

Sonification, or conveying data using non-verbal audio, is a relatively niche but growing approach for presenting data across multiple specialist domains including astronomy, climate science, and beyond ([Lenzi et al., 2021](#); [Lindborg et al., 2023](#); [Zanella et al., 2022](#)). The *strauss* Python package aims to provide such a tool, which builds upon previous approaches to provide a powerful means to explore different ways of expressing data, with fine control over the output audio and its format. *strauss* is a free, open source (FOSS) Python package, designed to allow flexible and effective sonification to be straightforwardly integrated into data workflows, in analogy to widely used visualisation packages.

The remit of *strauss* is broad; it is intended to be able to bridge between *ad-hoc* solutions for sonifying very particular datasets, and highly technical compositional and sound-design tools that are not optimised for sonification, or may have a steep learning curve. The code offers a range of approaches to sonification for a variety of contexts (e.g. science education, science communication, technical data analysis, etc). To this end, *strauss* is packaged with a number of examples of different sonification approaches, and preset configurations to support a *low-barrier, high-ceiling* approach. *strauss* has been used to produce both educational resources ([Harrison et al., 2022](#)), and analysis tools ([James W. Trayford et al., 2023](#)).

Statement of need

Sonification has great potential as a fundamental approach for interfacing with data. This provides new perspectives on data that complement visual approaches, as well as an accessible channel to data for those who cannot access visual presentation, for example those who are blind or visually impaired (BVI). As with the dominant approach of data *visualisation*, what can constitute sonification is very broad, with different considerations for aspects such as the audience, information content and aesthetics of the sonification. In order for sonification to become more established and realise its potential as a way of interfacing with data, accessible and flexible tools are needed to make sonification intuitive and accessible to those who routinely work with data.

Unlike data *visualisation*, however, sonification of data is a far less developed methodology, with a lack of widely adopted, cross-domain tools and interfaces for those dealing with data to use. For example, in the Python programming language, a number of packages exist for visualising data, such as *matplotlib* ([Hunter, 2007](#)), *yt* ([Turk et al., 2011](#)), *seaborn* ([Waskom, 2021](#)), or *plotly* ([Technologies Inc., 2015](#)). The lack of dedicated and accessible tools for sonifying data is a barrier to exploring the approach. Most solutions are piecemeal, using *ad-hoc* tools

to parse data and map it to properties of sound as well as generating and outputting sound in different formats. What's more, many of the tools being repurposed to sonify data are proprietary and platform-dependent, requiring paid-for licenses.

A number of effective python packages for data sonification have emerged e.g. *astronify* (Brasseur et al., 2023) and *SonoUno* (Casado et al., 2024). However these are typically feature-limited in how sonification is produced, namely continuous pitch-mapped sonification.

Realising the potential of sonification may require a “crowdsourced” approach; via broad adoption of the technique and innovation in sonification approaches driven by the scientific community. In this context, we identify a need for Python package that:

- Provides a full pipeline from data to sonified audio that can be integrated into the workflows of scientists and data analysts
- Is modular and enables complex, multi-variate sonification to be produced and fine tuned, while being simple enough to produce simple sonifications, for instance for novice users or in educational contexts
- Is fully open-source and platform-independent, such that sonification is able to be integrated more broadly into scientific practice

strauss (Sonification Tools and Resources for Analysis Using Sound Synthesis) is a Python package for data sonification. The code uses an object-oriented structure to provide a clear conceptual framework for the different components of a sonification (J. Trayford & Harrison, 2023). *strauss* is designed to be used by analysts to fully sonify their data in a way analogous to a plotting pipeline, to be analysed independently or in conjunction with visuals.

By choosing the mapping between the variables being communicated and the expressive properties of sound, *strauss* is designed to be a *low-barrier, high ceiling* tool; providing the means to make diverse and highly customised sonifications with a high degree of low-level control as the user becomes more experienced, while providing a relatively simple path to produce sonifications quickly for a novice. This is intended to provide the bridge between typical analysts who know their data and the facets of it that they want to communicate, and the sound experts or musicians that understand how to express data with sound.

The *strauss* code minimises dependencies where possible, implementing built-in signal generation and audio parsing and encoding based on low-level python libraries like *numpy* (Harris et al., 2020) and *scipy* (Virtanen et al., 2020), as well as being tested on multiple platforms (*MacOS, Linux, Windows*). This also provides means to interface with commonly used audio formats, such as “*Soundfont*” files (*.sf2*) to open a range of possibilities for representing data using different instruments. This helps to ensure that the free and open source status of *strauss* is maintained and does not require difficult to install or proprietary software, that may themselves have steep learning curves.

Towards our *low barrier* aspirations for *strauss*, a Tutorial-driven development (TDD) approach is used where each major feature should have an associated tutorial example, which are packaged with the code, in both Python Notebook (*examples/.ipynb*) and Python script (*examples/*.py*) formats. *strauss* provides tools for inline playback of sound in both formats. In addition, a further collection of modified examples is provided specifically for the *Google Colab* platform (*examples/colab/*.ipynb*), allowing examples to be run fully in-browser without requiring a local install of the code.

Offering both formats for examples in *strauss* allows us to exploit the interactivity and low technical threshold needed to use notebooks, while also having the BVI accessibility of the raw-text scripts, given the difficulties of using notebooks with screen readers (J. W. Trayford et al., 2024). *strauss* is provided with extensive documentation, maintained and hosted via [Read The Docs](#).

Acknowledgements

JWT acknowledges support via the STFC Early Stage Research & Development Grant, reference ST/X004651/1, CMH acknowledge funding from an United Kingdom Research and Innovation grant (code: MR/V022830/1). RS is supported by a studentship from an STFC Centre of Doctoral Training in Data Intensive Science (code: ST/W006790/1).

References

- Brasseur, C., Fleming, S., Kotler, J., & Meredith, K. (2023). *Astronify: v0.1* (Version v0.1). Zenodo. <https://doi.org/10.5281/zenodo.7713214>
- Casado, J., Vega, G. de la, & García, B. (2024). SonoUno development: A user-centered sonification software for data analysis. *Journal of Open Source Software*, 9(93), 5819. <https://doi.org/10.21105/joss.05819>
- 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>
- Harrison, C., Zanella, A., Bonne, N., Meredith, K., & Misdariis, N. (2022). Audible universe. *Nature Astronomy*, 6, 22–23. <https://doi.org/10.1038/s41550-021-01582-y>
- 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>
- Lenzi, S., Ciuccarelli, P., Liu, H., Hua, Y., & Zizzi, N. (2021). Data sonification archive. In *Data Sonification Archive*. <https://sonification.design/>
- Lindborg, P., Lenzi, S., & Chen, M. (2023). Climate data sonification and visualization: An analysis of topics, aesthetics, and characteristics in 32 recent projects. *Frontiers in Psychology*, 13. <https://doi.org/10.3389/fpsyg.2022.1020102>
- Technologies Inc., P. &. (2015). *Collaborative data science*. Plotly Technologies Inc, from. <https://plot.ly>
- Trayford, James W., Harrison, C. M., Hinz, R. C., Kavanagh Blatt, M., Dougherty, S., & Girdhar, A. (2023). Inspecting spectra with sound: proof-of-concept and extension to datacubes. *RAS Techniques and Instruments*, 2(1), 387–392. <https://doi.org/10.1093/rasti/rzad021>
- Trayford, J. W., Youles, S., Harrison, C. M., & Bonne, N. (2024). Ear to the Sky: Astronomical Sonification for Accessible Outreach, Education and Research with STRAUSS. 57, 42–46.
- Trayford, J., & Harrison, C. (2023). *Introducing strauss: A flexible sonification python package*. 249–256. <https://doi.org/10.21785/icad2023.1978>
- Turk, M. J., Smith, B. D., Oishi, J. S., Skory, S., Skillman, S. W., Abel, T., & Norman, M. L. (2011). yt: A Multi-code Analysis Toolkit for Astrophysical Simulation Data. *The Astrophysical Journal Supplement Series*, 192, 9. <https://doi.org/10.1088/0067-0049/192/1/9>
- 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>
- Waskom, M. L. (2021). Seaborn: Statistical data visualization. *Journal of Open Source*

Software, 6(60), 3021. <https://doi.org/10.21105/joss.03021>

Zanella, A., Harrison, C. M., Lenzi, S., Cooke, J., Damsma, P., & Fleming, S. W. (2022). Sonification and sound design for astronomy research, education and public engagement. *Nature Astronomy*, 6, 1241–1248. <https://doi.org/10.1038/s41550-022-01721-z>