Retriever: Data Retrieval Tool

Henry Senyondo[1], **Benjamin D. Morris**[1], **Akash Goel**[3], **Andrew Zhang**[4], **Akshay Narasimha**[5], **Shivam Negi**[6], **David J. Harris**[4], **Deborah Gertrude Digges**[10], **Kapil Kumar**[7], **Amritanshu Jain**[5], **Kunal Pal**[8], **Kevinkumar Amipara**[9], **and Ethan P. White**[1, 2]

1 Department of Wildlife Ecology and Conservation, University of Florida **2** Informatics Institute, University of Florida **3** Delhi Technological University, Delhi **4** The University of Florida **5** Birla Institute of Technology and Science, Pilani **6** Manipal Institute of Technology, Manipal **7** National Institute of Technology, Delhi **8** RWTH Aachen University, Aachen, Germany **9** Sardar Vallabhbhai National Institute of Technology, Surat **10** PES Institute of Technology, Bengaluru

DOI: 10.21105/joss.00451

Software

- Review
- Repository
- Archive

Summary

The Data Retriever automates the first steps in the data analysis workflow by downloading, cleaning, and standardizing tabular datasets, and importing them into relational databases, flat files, or programming languages (Morris and White 2013). The automation of this process reduces the time for a user to get most large datasets up and running by hours to days.The retriever uses a plugin infrastructure for both datasets and storage backends. New datasets that are relatively well structured can be added adding a JSON file following the Frictionless Data tabular data metadata package standard (Frictionlessdata 2017). More complex datasets can be added using a Python script to handle complex data cleaning and merging tasks and then defining the metadata associated with the cleaned tables. New storage backends can be added by overloading a general class with details for storing the data in new file formats or database management systems. The retriever has both a Python API and a command line interface. An R package that wraps the command line interface and a Julia package that wraps the Python API are also available.

The 2.0 and 2.1 releases add extensive new functionality. This includes the Python API, the use of the Frictionless Data metadata standard, Python 3 support, JSON and XML backends, and autocompletion for the command line interface.

References

Frictionlessdata. 2017. "Specs: Specifications for Frictionless Data." https://github.com/frictionlessdata/specs.

Morris, Benjamin D., and Ethan P. White. 2013. "The Ecodata Retriever: Improving Access to Existing Ecological Data." *PLoS ONE*, June. Public Library of Science (PLoS). doi:10.1371/journal.pone.0065848.