Multi-language retrieval of United States hydrologic data

T. O. Hodson [1,*], L. A. DeCicco [2,*], J. A. Hariharan [3,*], L. F. Stanish [3], S. Black [4], J. S. Horsburgh [5]

[1]U.S. Geological Survey Central Midwest Water Science Center, Urbana, Illinois, USA
[2]U.S. Geological Survey Upper Midwest Water Science Center, Middleton, Wisconsin, USA
[3]U.S. Geological Survey Water Mission Area, Reston, Virginia, USA
[4]Consortium of Universities for the Advancement of Hydrologic Science, Inc. (CUAHSI)
[5]Utah State University, Logan, Utah, USA
[*]These authors contributed equally to this work.

Key Points:

- R, Python, and Julia packages for accessing U.S. hydrologic data
- Technology enabling cloud data access is critical for open and reproducible scientific workflows

Corresponding author: Timothy Hodson, `tohodson@usgs.gov`

Abstract

Much of modern science takes place in a computational environment, and, increasingly, that environment is programmed using R, Python, or Julia. Furthermore, most scientific data now live on the cloud, so the first step in many workflows is to query a cloud database and load the response into a computational environment for further analysis. Thus, tools that facilitate programmatic data retrieval represent a critical component in reproducible scientific workflows. Earth science is no different in this regard. To fulfill that basic need, we developed `dataRetrieval`, `dateretrieval`, and `DataRetrieval.jl`: R, Python, and Julia packages, respectively, that provide multi-language access to hydrologic data from the U.S. Geological Survey's National Water Information System database and the multi-agency Water Quality Portal.

1 Introduction

R, Python, and Julia are open-source languages with large communities of scientific users and developers, which have become the lingua franca—the common language—of the open science movement. Notably, all three can run within Jupyter Notebooks, a web-based interactive computing platform that scientists increasingly use to explore data and communicate their findings (Granger & Pérez, 2021), create and share reproducible workflows (Beg et al., 2021), and access data in the cloud (Abernathey et al., 2021).

Open data initiatives have pushed most scientific data to the cloud to ease accessibility, so a typical scientific workflow begins by querying a cloud database and loading the response into the computational environment for further analysis. In that paradigm, data are accessed using either some kind of graphical user interface (GUI) or by writing code to retrieve data via an application programming interface (API). Non-programmers find GUIs more intuitive, but their manual nature creates barriers to reproducibility and scalability because it can be difficult to record the exact sequence of steps within a GUI, and GUIs often change. In contrast, APIs are typically versioned, which means that code written to programmatically access an API can be executed repeatably, shared, tracked in version control, and run through automated tests, all of which are tenets of computational reproducibility and open science.

The U.S. Geological Survey (USGS) operates the largest water-monitoring network in the United States, whose data are widely used for research, as well as operationally for modeling, flood forecasting, water resources management investigations, etc. Thus, there is a great benefit to science and society in having standardized and reusable packages for programmatically accessing USGS data using widely used data science languages (i.e., R, Python, and Julia), ensuring that the first step in many workflows—loading USGS data from the cloud—is reproducible. To that end, we developed R, Python, and Julia packages providing programmatic access to data from any streamflow gage, water quality monitoring station, or groundwater well, as well as other datasets available via USGS's National Water Information System database (U.S. Geological Survey, 2023) and the multi-agency Water Quality Portal (National Water Quality Monitoring Council, 2023).

2 Sharing Scientific Knowledge as Reproducible Workflows

Given that this paper presents relatively simple utilities for retrieving data, we reflect on their role within the broader scientific enterprise. Fundamentally, these utilities facilitate the development of reusable packages and reproducible workflows. There is growing awareness of a reproducibility crisis in science (e.g., Baker, 2016): by one estimate, 95 percent of recent hydrology and water resources publications cannot be reproduced (Stagge et al., 2019). In response, many within the scientific community are advocating for greater transparency and reproducibility of research results. Journals increasingly require submissions to be accompanied by data, code, and other research artifacts that enable the reproduction of the analyses and results. But the original code and data are insufficient to ensure reproducibility; one also needs the original computational environment, or at least the means to recreate it.

A *package* is an archive of software along with metadata intended to make the software more easily shared and reused by others (Hillard, 2023). It is essentially a set of software tools that may be reused to accomplish different computational tasks, either by expanding the functionality of other packages or by performing a particular task, such as in a *workflow*.

A *workflow* is a sequence of steps that produce a particular result. A recipe for baking bread is a workflow, but in this context, we mean workflows that run in a computational environment, known as computational workflows. Often, workflows that begin as notebooks or scripts go on to be developed into packages that more formally organize and codify a set of functionality along with scientific knowledge for reuse by others. Just as in open-source software development, packages are fundamental organizational units within open science, where researchers contribute expertise to help develop packages, then use and combine those packages to create flexible and reproducible workflows. *In this regard, one might consider the development and availability of scientific code packages to be a revolution in scientific philosophy (metascience).* Recent advances in machine learning, data science, and many other domains have been accelerated through the availability of open-source packages (Nguyen et al., 2019; Langenkamp & Yue, 2022)

The principal purpose of a package is reusability: If one researcher writes a package to accomplish X, then another researcher can use that package to accomplish X without having to write the code themselves. There is also an expectation that packages evolve as code and knowledge are contributed over time. A workflow is, in essence, another type of package but its purpose and lifecycle differ. Once published, a workflow is intended as a static archive; its principal purpose is to ensure reproducibility. To achieve that, workflows adopt many of the same tools and practices used in software packaging, such as dependency resolvers to reproduce a particular computation environment, version control, automated testing, and open web-based publication platforms, etc.

A *packaged workflow* combines both concepts by using computer science tools and practices in a manner that allows it to easily migrate from one computational environment to another. Such workflows are becoming an increasingly important component of scientific communication. An example is HydroShare (Tarboton et al., 2014; Horsburgh et al., 2016), which is an online repository that supports sharing and publication of packaged workflows. Using HydroShare, a researcher can upload a Jupyter Notebook containing their workflow and then share it publicly or permanently publish it with a citable digital object identifier (DOI). Anyone can then rerun the notebook using HydroShare's linked JupyterHub environment. The importance of data retrieval packages like the three in this paper, `HyRiver` (Chegini et al., 2021), and others is that they facilitate programmatic data access, which is a key component in creating and simplifying packaged workflows.

2.1 Examples

For each language—R, Python, and Julia— we provide a brief demonstration showing how the data retrieval packages can be used to build other packages or workflows. For many more examples and tutorials see the links to the package documentation in the *Open Research Section*.

2.1.1 R

Of the three packages, the R version, `dataRetrieval` was developed first and has been downloaded over 174,000 times (as of May 2023; De Cicco et al., 2023). Along with simplifying workflows, its functionality has become integral in other packages like `EGRET` (Exploration and Graphics for RivEr Trends), which provides utilities for the analysis of long-term changes in water quality and streamflow (Hirsch et al., 2010). Several `EGRET` functions use `dataRetrieval` to retrieve data, then preprocess the output into an analysis-ready format. A typical `EGRET` workflow retrieves data, calibrates a model, and displays long-term trend calculations. Here we use it to retrieve orthophosphate data from a USGS monitoring location (01631000), then model and plot the orthophosphate load through time (Figure 1). Using `dataRetrieval`, both `EGRET` and the workflow are simpler and, therefore, easier to understand, use, and maintain.

```
library(EGRET)
site <- "01631000"
parameter <- "00660" # USGS code for orthophosphate
Sample <- readNWISSample(site, parameter)
Daily <- readNWISDaily(site,
                       startDate = min(Sample$Date))
INFO <- readNWISInfo(site, parameter,
                     interactive = FALSE)
eList <- mergeReport(INFO, Daily, Sample)
eList <- modelEstimation(eList, verbose = FALSE)
plotConcHist(eList, printTitle=FALSE)
```

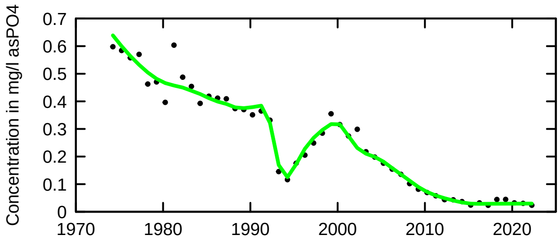

Figure 1. `EGRET` generated timeseries of flow-normalized concentration of orthophosphate (PO4) in miligrams per liter (mg/L) for the South Fork Shenanndoah River at Front Royal, Virginia. Dots depict the annual mean concentration.

2.2 Python

A strength of Jupyter is that it allows for fast prototyping of code and data exploration. Here, we demonstrate using `dataretrieval` to query sites with total phosphorus measurements in the state of Illinois, then creating an interactive webmap using `hvplot` (Figure 2).

```
from dataretrieval import nwis
import geopandas as gpd
import hvplot.pandas

parameter = '00665' # USGS code for total phosphorus
df, meta = nwis.what_sites(stateCd='IL', parameterCd=parameter)
geometry = gpd.points_from_xy(df.dec_long_va, df.dec_lat_va)
gdf = gpd.GeoDataFrame(df, geometry=geometry)

gdf.hvplot.points(geo=True, hover_cols=['site_no', 'station_nm'],
                  tiles=True, width=300, size=3)
```

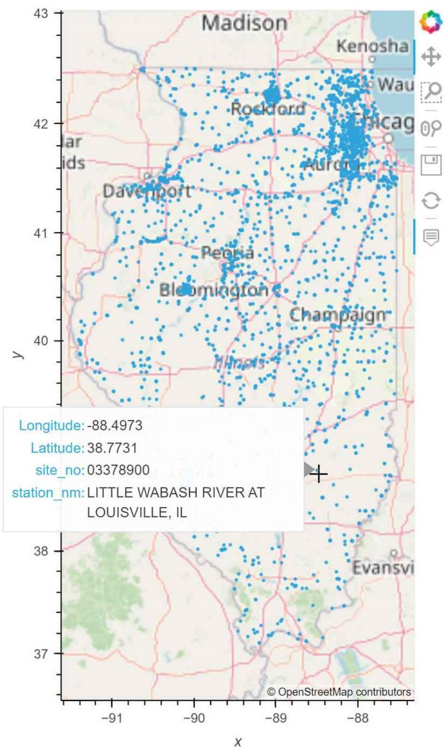

Figure 2. Interactive web map displaying locations in Illinois with phosphorus samples.

2.3 Julia

As the youngest programming language and data retrieval package, our Julia demonstration is more introductory. We use `DataRetrieval.jl` to retrieve annual groundwater levels from a single site in Delaware, then compute summary statistics on an annual basis using the `Statistics` package (JuliaStats Contributors, 2023) and format the output for publication using `Latexify` (Korsbo & other contributors, 2023) (Table 1).

```
using DataRetrieval, Dates, Statistics, DataFrames, Latexify

site = "393617075380403"
parameter = "72019" # USGS code for depth to water level
df, response = readNWISdv(site,
                          parameter,
                          startDate="1776-07-04",
                          endDate="2022-12-31",
                          format="json");
df.datetime = Dates.DateTime.(df.datetime, "yyyy-mm-ddTHH:MM:SS.SSS");
df.year = Dates.year.(df.datetime);
df2 = combine(groupby(df, :year),
              parameter => minimum => :Minimum,
              parameter => maximum => :Maximum,
              parameter => mean => :Mean;

latexify(df2, env=:table) |> print
```

Table 1. Annual (calendar year) summary statistics for groundwater levels (depth to water level in feet below land surface) at U.S. Geological Survey site 393617075380403 in Delaware.

$Year$	$Minimum$	$Maximum$	$Mean$
2012	-0.27	-0.0	-0.11
2015	-1.2	-0.03	-0.26
2016	-0.6	0.01	-0.2
2020	-0.6	0.12	-0.22
2021	-0.38	0.1	-0.16
2022	-0.44	0.15	-0.045

3 Use Cases

The data retrieval packages are basic utilities that support a range of uses. For scientific research and publishing, they automate hydrologic data retrieval in workflows such that data access can be encoded in scripts or notebooks that can be shared, re-run, and built upon by other researchers. These packages reduce the time and effort required for retrieving and loading data into performant, analysis-ready data structures, like "data frames." This lowers barriers for novice users who may struggle to identify the best data structures to use and reduces time spent learning how to parse and load data, allowing users to more quickly delve into discovery and understanding.

Beyond scientific research and academic publishing, water resources professionals use USGS hydrologic data for operational purposes including flood forecasting, operation of dams and hydraulic control structures, design of bridges and flood control projects, implementing flood warning systems, allocating irrigation water, planning for energy development, assessing water quality and pollution, and others. While our focus has been on the reproducibility of scientific work, these packages have similar utility for practitioners who need to build transparent, reliable, and repeatable modeling and analysis workflows. The availability of the tools in multiple languages provides options for use in different computational environments across platforms ranging from personal computers to on-premise or cloud-based computing clusters.

In the classroom, instructors use these packages to teach data science and hydroinformatics concepts, which are becoming increasingly important skills as scientific and engineering work becomes more data-intensive. Indeed, a growing part of hydrologic science is shifting from collecting data for testing or supporting existing conceptual models toward analyses based on models derived from observational data (Chen & Han, 2016). In a recent survey, Jones, Horsburgh, Pacheco, Flint, and Lane (2022) found that most instructors who offer a course in hydroinformatics or water data science at the college level include basics of coding/scripting; data formatting, manipulation, and wrangling; visualization and plotting; and other data science topics. Nearly all of these instructors used Python or R in their course materials, and multiple instructors reported using one of the data retrieval packages directly. Feedback from that survey was used in developing the Hydroinformatics and Water Data Science module on HydroLearn, which uses the Python `dataretrieval` (Jones, Horsburgh, & Pacheco, 2022).

4 Conclusions

R, Python, and Julia are used extensively in scientific computing and data science, and all three support Jupyter notebooks, a computing platform used for teaching and scientific discovery. The data retrieval packages provide programmatic access to USGS hydrologic data in these languages, thereby making that data accessible from notebooks or other programs, and, ultimately, making those research and analysis workflows more reproducible. The usage examples in the paper are nowhere near comprehensive of what the packages can do, especially when combined with functionality from other packages. To learn more, refer to the code repositories in the *Open Research Section*.

Open Research Section

The R version is available at `https://github.com/DOI-USGS/dataRetrieval`, as well as via CRAN. The Python version is available at `https://github.com/DOI-USGS/dataretrieval-python`, as well as via PyPI and conda-forge. The Julia version is available at `https://github.com/DOI-USGS/dataretrieval.jl` and can be installed using `Pkg`, Julia's built-in package manager.

Please cite this paper when discussing the software in an abstract sense or other ideas from the paper. When using the software, we recommend citing the specific version and its associated software release. For example, the R, Python, and Julia versions used in the paper are available as software releases (De Cicco et al., 2023; Hodson et al., 2023; Hariharan, 2023, respectively),

The Python example was created using the package versions on conda-forge:

```
conda create -n dataretrieval geopandas hvplot cartopy geoviews jupyterlab dataretrieval
conda activate dataretrieval
jupyter lab
```

and run in Jupyter on Windows Subsystem for Linux 2 (WSL2) with an Intel processor. Alternatively, the supplemental `environment.yml` contains all the necessary package metadata to reproduce our Python computational environment (`https://raw.githubusercontent.com/DOI-USGS/dataretrieval-python/paper-env/demos/webmap/environment.yml`). As with all the examples, different package managers, operating systems, and hardware may yield different results. If you are unable to reproduce the examples, please raise an issue on the relevant GitHub repository.

Acknowledgments

This material is partially based upon work supported by the National Science Foundation (NSF) under award 1931297. Any opinions, findings, conclusions, or recommendations expressed in this material do not necessarily reflect the views of the NSF. Any use of trade, firm, or product names is for descriptive purposes only and does not imply endorsement by the U.S. Government.

References

Abernathey, R. P., Augspurger, T., Banihirwe, A., Blackmon-Luca, C. C., Crone, T. J., Gentemann, C. L., . . . Signell, R. P. (2021). Cloud-native repositories for big scientific data. *Computing in Science & Engineering*, *23*(2), 26-35. Retrieved from https://doi.org/10.1109/msce.2021.3059437 doi: 10.1109/mcse.2021.3059437

Baker, M. (2016, may). 1,500 scientists lift the lid on reproducibility. *Nature*, *533*(7604), 452–454. Retrieved from https://doi.org/10.1038/533452a doi: 10.1038/533452a

Beg, M., Taka, J., Kluyver, T., Konovalov, A., Ragan-Kelley, M., Thiéry, N. M., & Fangohr, H. (2021, mar). Using Jupyter for reproducible scientific workflows. *Computing in Science & Engineering*, *23*(2), 36–46. Retrieved from https://doi.org/10.1109/mcse.2021.3052101 doi: 10.1109/mcse.2021.3052101

Chegini, T., Li, H.-Y., & Leung, L. R. (2021). Hyriver: Hydroclimate data retriever. *Journal of Open Source Software*, *6*(66), 3175. Retrieved from https://doi.org/10.21105/joss.03175 doi: 10.21105/joss.03175

Chen, Y., & Han, D. (2016, mar). Big data and hydroinformatics. *Journal of Hydroinformatics*, *18*(4), 599–614. Retrieved from https://doi.org/10.2166/hydro.2016.180 doi: 10.2166/hydro.2016.180

De Cicco, L. A., Lorenz, D., Hirsch, R. M., Watkins, W., & Johnson, M. (2023). dataretrieval: R packages for discovering and retrieving water data available from u.s. federal hydrologic web services [Computer software manual]. Reston, VA: U.S. Geological Survey. Retrieved from https://doi.org/10.5066/P9X4L3GE doi: 10.5066/P9X4L3GE

Granger, B. E., & Pérez, F. (2021). Jupyter: Thinking and storytelling with code and data. *Computing in Science & Engineering*, *23*(2), 7-14. Retrieved from https://doi.org/10.1109/msce.2021.3059263 doi: 10.1109/msce.2021.3059263

Hariharan, J. A. (2023). Dataretrieval.jl–Julia package for obtaining usgs water data directly from web services [Computer software manual]. Reston, VA: U.S. Geological Survey. Retrieved from https://doi.org/10.5066/P95XLHUH doi: 10.5066/P95XLHUH

Hillard, D. (2023). *Publishing python packages*. Manning.

Hirsch, R. M., Moyer, D. L., & Archfield, S. A. (2010, sep). Weighted regressions on time, discharge, and season (WRTDS), with an application to Chesapeake Bay river inputs. *JAWRA Journal of the American Water Resources Association*, *46*(5), 857–880. Retrieved from https://doi.org/10.1111/j.1752-1688.2010.00482.x doi: 10.1111/j.1752-1688.2010.00482.x

Hodson, T. O., Hariharan, J. A., Black, S., & Horsburgh, J. S. (2023). dataretrieval (Python): a Python package for discovering and retrieving water data available from u.s. federal hydrologic web services [Computer software manual]. Reston, VA: U.S. Geological Survey. Retrieved from https://doi.org/10.5066/P94I5TX3 doi: 10.5066/P94I5TX3

Horsburgh, J. S., Morsy, M. M., Castronova, A. M., Goodall, J. L., Gan, T., Yi, H., . . . Tarboton, D. G. (2016). HydroShare: Sharing diverse environmental data types and models as social objects with application to the hydrology domain. *JAWRA Journal of the American Water Resources Association*, *52*(4), 873–889. Retrieved from https://doi.org/10.1111/1752-1688.12363 doi: 10.1111/1752-1688.12363

Jones, A. S., Horsburgh, J. S., & Pacheco, C. J. B. (2022). *Hydroinformatics and water data science.* HydroLearn. Retrieved from https://edx.hydrolearn.org/courses/course-v1:USU+CEE6110+2022/about

Jones, A. S., Horsburgh, J. S., Pacheco, C. J. B., Flint, C. G., & Lane, B. A. (2022). Advancing hydroinformatics and water data science instruction: Community perspectives and online learning resources. *Frontiers in Water*, *4*. Retrieved from https://doi.org/10.3389/frwa.2022.901393 doi: 10.3389/frwa.2022.901393

JuliaStats Contributors. (2023). *Statistics.jl.* Retrieved from https://github.com/JuliaStats/Statistics.jl

Korsbo, N., & other contributors. (2023). *Latexify.jl.* Retrieved from https://github.com/korsbo/Latexify.jl

Langenkamp, M., & Yue, D. N. (2022, jul). How open source machine learning software shapes AI. In *Proceedings of the 2022 AAAI/ACM conference on AI, ethics, and society.* ACM. Retrieved from https://doi.org/10.1145/3514094.3534167 doi: 10.1145/3514094.3534167

National Water Quality Monitoring Council. (2023). *Water Quality Portal.* Retrieved from http://www.waterqualitydata.us

Nguyen, G., Dlugolinsky, S., Bobák, M., Tran, V., García, Á. L., Heredia, I., … Hluchý, L. (2019, jan). Machine learning and deep learning frameworks and libraries for large-scale data mining: a survey. *Artificial Intelligence Review*, *52*(1), 77–124. Retrieved from https://doi.org/10.1007/s10462-018-09679-z doi: 10.1007/s10462-018-09679-z

Stagge, J. H., Rosenberg, D. E., Abdallah, A. M., Akbar, H., Attallah, N. A., & James, R. (2019, feb). Assessing data availability and research reproducibility in hydrology and water resources. *Scientific Data*, *6*(1). Retrieved from https://doi.org/10.1038/sdata.2019.30 doi: 10.1038/sdata.2019.30

Tarboton, D. G., Horsburgh, J. S., Idaszak, R., Heard, J., Ames, D., Goodall, J. L., … Maidment, D. (2014). Hydroshare: Advancing collaboration through hydrologic data and model sharing. *Proceedings of the 7th International Congress on Environmental Modelling and Software, San Diego, CA.* Retrieved from https://doi.org/10.13140/2.1.4431.6801 doi: 10.13140/2.1.4431.6801

U.S. Geological Survey. (2023). *National Water Information System data available on the World Wide Web (USGS Water Data for the Nation).* U.S. Geological Survey. Retrieved from http://doi.org/10.5066/F7P55KJN