Jun 26th, 9:00 AM - 10:20 AM

Automated Data Discovery, Retrieval, Manipulation, and Publication using Python, Tethys, and HydroShare

Scott D. Christensen
U.S. Army Engineer and Research and Development Center, scott.d.christensen@usace.army.mil

Dharhas Pothina
U.S. Army Engineer and Research and Development Center, dharhas.pothina@usace.army.mil

Aaron Valoroso
U.S. Army Engineer and Research and Development Center, Aaron.A.Valoroso@erdc.dren.mil

Kevin Winters
U.S. Army Engineer and Research and Development Center, Kevin.D.Winters@usace.army.mil

iEMSs International Congress on Environmental Modelling and Software

9th International Congress on Environmental Modelling and Software
Fort Collins, Colorado, USA, Mazdak Arabi, Olaf David, Jack Carlson, Daniel P. Ames (Eds.)
https://scholarsarchive.byu.edu/iemssconference/2018/

Automated Data Discovery, Retrieval, Manipulation, and Publication using Python, Tethys, and HydroShare

Scott D. Christensen[a], Dharhas Pothina[a], Aaron A. Valoroso[a], Kevin D. Winters[a]
[a] US Army Engineer Research and Development Center, Information Technology Laboratory
(Scott.D.Christensen@usace.army.mil, Dharhas.Pothina@usace.army.mil,
Aaron.A.Valoroso@usace.army.mil, Kevin.D.Winters@usace.army.mil)

Abstract: Most environmental modelling efforts have significant data needs, which can present technical challenges and distract from the primary modelling objectives. These challenges include finding relevant data sources, retrieving large amounts of data, performing data transformations, and finally storing and sharing results. This presentation describes the tools being used and developed at the U.S. Army Engineer Research and Development Center (ERDC) to address these challenges. These tools include (1) Quest, an extensible Python library for searching various local and public data providers, automating downloading, performing various data filters (or transformations), and finally, publishing the data to data repositories, (2) Data Depot, an internal instance of HydroShare, a web-based data repository and catalogue, which supports data archival, discovery, and sharing, and (3) Quest Web, a map-based web interface to Quest built with Tethys Platform. These tools all work together to provide a comprehensive data solution for our scientists and engineers, and though these tools have been developed specifically to support hydrologic modelling workflows, they are generic and extensible making them applicable to a broad range of data and workflows.

***Keywords**:* Data; Python; Tethys Platform; HydroShare.

1 INTRODUCTION

Researchers at the U.S. Army Engineer Research and Development Center (ERDC) are often called upon to produce environmental modelling results with very little lead time. Efforts to improve environmental modelling have produced high-fidelity, physics-based models that have the capacity to be refined and tuned to specific modelling needs and domains. However, these improvements have come at the cost of requiring significant amounts of input data (Michener et al., 2012; Tolle et al., 2011). Data are being collected and produced now more than ever, yet formats, storage, and transfer mechanisms have not been standardized, and thus retrieving and processing all of the necessary data remain a challenge to modellers which distracts from their primary modelling objectives. Since data preparation for the cutting-edge research models remains a barrier, these modelling efforts often resort to simplistic models that are unable to leverage all available data.

Furthermore, data archival methods for the U.S. Army Corps of Engineers are not centralized or standardized and thus there often exists previous modelling efforts that could inform new queries but that go unused because the results were never published and made discoverable, becoming what Heidorn (2008) referred to as dark data. The challenge of dark data has been addressed by several efforts including the HydroShare project. HydroShare is a public data repository that allows for data publication in a standardized and discoverable way (Tarboton et al., 2014), and specifically it is well suited for the publication of environmental models (Horsburgh et al., 2016). Yet, in some cases there are organizational and project specific restrictions that prevent data from being posted to public repositories like HydroShare.

To overcome the various data barriers for ERDC modellers and allow for the latest developments in environmental models to be leveraged in projects with short turn around times, the whole data management cycle needs to be automated. The data management cycle consists of several phases that include (1) data discovery, (2) data retrieval, (3) local data management and organization, (4) data processing or manipulation, and (5) data publishing and archival. The final phase of publishing and archival should complete the cycle by permitting the data to be discoverable for future use (see Figure 1).

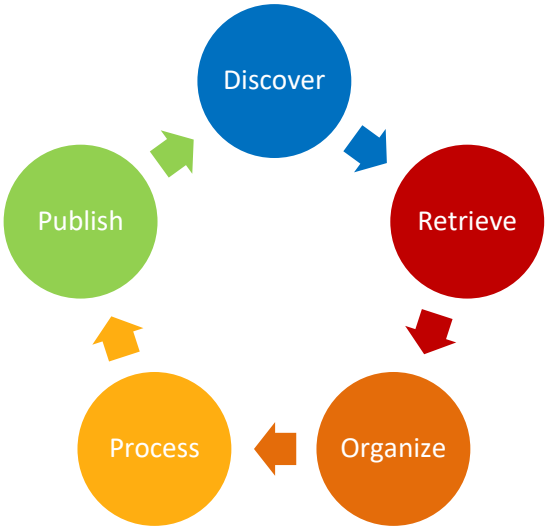

Figure 1. Data Management Cycle

Here we present a comprehensive infrastructure that leverages several open source software projects to provide automated data management for ERDC modellers. This infrastructure is composed of three components: (1) Quest, an extensible Python library for searching various local and public data providers, automating downloading, performing various data filters (or transformations), and finally, publishing the data to data repositories, (2) Data Depot, an internal instance of HydroShare, a web-based data repository and catalogue, which supports data archival, discovery, and sharing, and (3) Quest Web: a map-based web interface to Quest built with Tethys Platform.

2 METHODS

To support the data management needs of ERDC projects, we developed a Python library, Quest, to automate the data management cycle. We chose the Python language because it allows Quest to be used very flexibly in various workflows and applications. To support the ERDC's data publishing needs, we deployed an internal instance of HydroShare, which we call Data Depot to distinguish it from the public instance of HydroShare. Finally, we created a web interface for Quest, called Quest Web, to provide a widely accessible interface for data management.

2.1 Quest

Quest is an open source Python library that facilitates automating many steps in the data management process including searching, downloading, organizing, processing (filtering), and publishing. Quest accomplishes these operations by managing key metadata and through the use of three types of plugins: (1) provider plugins, (2) filter plugins, and (3) I/O plugins. The plugin architecture, diagrammed in Figure 2 makes it easy to expand the capabilities of Quest to include additional data sources and data filtering operations, and to be compatible with additional file formats. This flexibility is needed to meet the diversity of data needs for the various projects at the ERDC and to support the many different storage formats and application programming interfaces (APIs) that are offered by data providers.

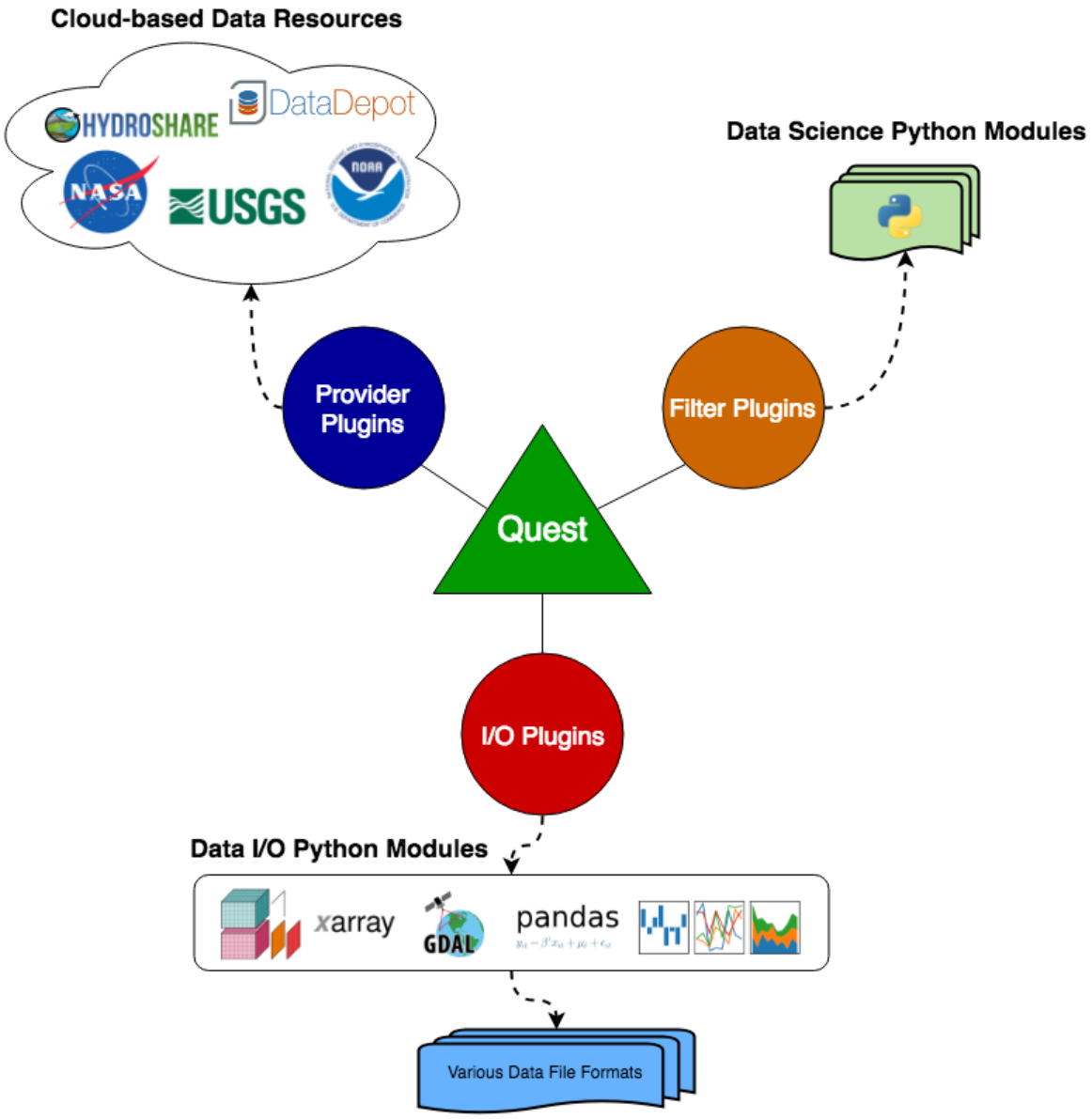

Figure 2. Plugin Architecture of Quest

2.1.1 Searching

Quest is able to search for data from various web or local data repositories, which in the Quest terminology are referred to as data providers and are described by provider plugins. A data provider is any academic, governmental, public or private organization that provides data accessible to a consumer through a web-based service such as the USGS National Water Information System (NWIS) ("USGS Water Data for the Nation", 2018). Provider plugins are written as an interface between the Quest internal API and the API or data organization of the specific data provider. In the search process Quest downloads and caches metadata from each provider to create a knowledge base of available data. It can then quickly perform queries on the metadata based on location (for geographic data), parameter, or other metadata tags.

To quickly support data produced within the ERDC or data procured from other U.S. governmental agencies that is not accessible through automated means, a special data provider, known as a user provider, can be described to interact with a simple file server (file system listing). For Quest to access these local repositories they must contain a simple metadata file, written in the YAML format, which provides a description of the data, the parameters, and the hierarchical file structure. Users can then dynamically add these data sources to Quest at runtime by simply providing the link or path to the YAML file describing the data.

2.1.2 Downloading

The provider plugins also handle retrieving data from the data providers. Each provider plugin defines a set of download options that the user must specify when downloading data. The download options are customized for each plugin and are based on what input the provider's API requires, which often includes options such as the parameter name, the start date, and the end date. Downloaded data are saved to disk in a standardized directory hierarchy created by Quest (see 2.1.3).

The download options are one of several instances of user specified options in Quest (other examples are explained in 2.1.4 and 2.1.5), In each of these cases the definition of the options is specified using the Python library Param ("Param", 2018). Param allows descriptions and validation of the options to be coded into the option definition and also provides an easy way to map the options to graphical user interface (GUI) widgets when working with Quest through browser-based interfaces.

2.1.3 Data Organization

Quest provides a simple organization structure consisting of projects and collections. A project corresponds to a directory on disk and contains a SQLite database that stores metadata about the project and its collections and data. Collections are subdirectories to the project and are containers for the actual data files. A project must have at least one collection and all data must belong to a collection. By default, the project directories are all stored in a standard location based on the operating system. This simple organization method provides a consistent and predictable structure for storing the data and metadata.

2.1.4 Data Filtering

Quest provides a mechanism for performing data transformations or manipulations through the use of filters. Filters are enabled by filter plugins and can consist of any operation or set of operations that can be done via Python. For example, the resampling filter will operate on time series data and aggregate the data based on a specified time frame (e.g. daily, weekly, monthly) using a specified method (e.g. sum, average, max, min, etc.). Similarly, the raster-merge filter takes several raster datasets and merges them to create a new dataset. A filter plugin must define a set of options, following the same pattern as download options (see 2.1.2), that serve as the input to the filter. Filters can accept one or several datasets as input, however, these input datasets are not modified directly. Rather, a new dataset or datasets are created as a result of the filter operation. The filter name and options are stored in the metadata of each new dataset so that the provenance of the data can be tracked.

The I/O plugins enable datasets of various file formats to be read in as Python data structures. Python libraries such as Xarray ("xarray: N-D labeled arrays and datasets in Python", 2018), and Pandas ("Python Data Analysis Library", 2018), provide standard data structures with the capability of reading many different file formats. Quest uses the metadata stored for each dataset to determine which I/O plugin to use to read the data. Once the data are in memory the filter plugin can use other Python libraries to operate on them.

2.1.5 Data Publishing

The last step in the data cycle is publishing. Quest enables data publishing through the provider plugins. The provider plugin, therefore, can have two purposes: (1) searching and retrieving data from a data repository, and (2) publishing data to a repository. Not all provider plugins meet both purposes. In fact, most data providers only serve as a source for retrieving data. However, some data providers, such as HydroShare, allow users both to retrieve data from and publish data to its repository. In these cases, the provider plugin can support both retrieving and publishing data. Similar to other processes in Quest, the publishing capability of a provider plugin must define options (see 2.1.2) that allow the user to specify what datasets to publish and any other required metadata that is needed for the publishing process on the specified provider.

2.2 Data Depot

As a DoD research organization, the ERDC often produces data that is restricted from being published to external repositories such as HydroShare. However, the need to archive and make data discoverable in a controlled environment still exists. We have addressed this issue by creating an internal instance of HydroShare which we call Data Depot.

The Data Depot service is hosted on the ERDC internal network. This provides researchers within the organization a way to publish or share data in a secure way. Data Depot is configured in the same way as the public HydroShare with an integrated Rule-Oriented Data System (iRODS) ("iRODS", 2018) data zone and user zone serving as the underlying data management system.

Although the Data Depot can be used directly through its web interface, Quest supports interaction with Data Depot through scripts, allowing data curation within automated workflows.

2.3 Quest Web

While Quest is designed to be scriptable and enable automated data cycle workflows, it is also designed to work with GUIs. Quest Web is a web-based interface for Quest built using the web development and hosting environment, Tethys Platform (Swain et al., 2016). Tethys was designed to lower the barrier to creating geospatial-enabled web applications. It is built on a completely free and open source software (FOSS) stack, including Django ("The Web framework for perfectionists", 2018), OpenLayers ("OpenLayers", 2018), GeoServer ("GeoServer", 2018), and PostGIS ("PostGIS", 2018). It also provides a powerful Python software development kit (SDK), which reduces much of the boiler plate code, and facilitates many common features of geospatial web apps. Since Tethys Platform is a Python-based web framework, it makes interactions with Quest very straightforward. The web-based interface allows researchers throughout the organization to easily access data in a map-based environment without needing to install anything on their local system (see Figure 3). It also serves as a way to explore data and operations in an interactive way which is often needed as a precursor to creating an automated workflow or when the objectives are not concrete enough to facilitate automation.

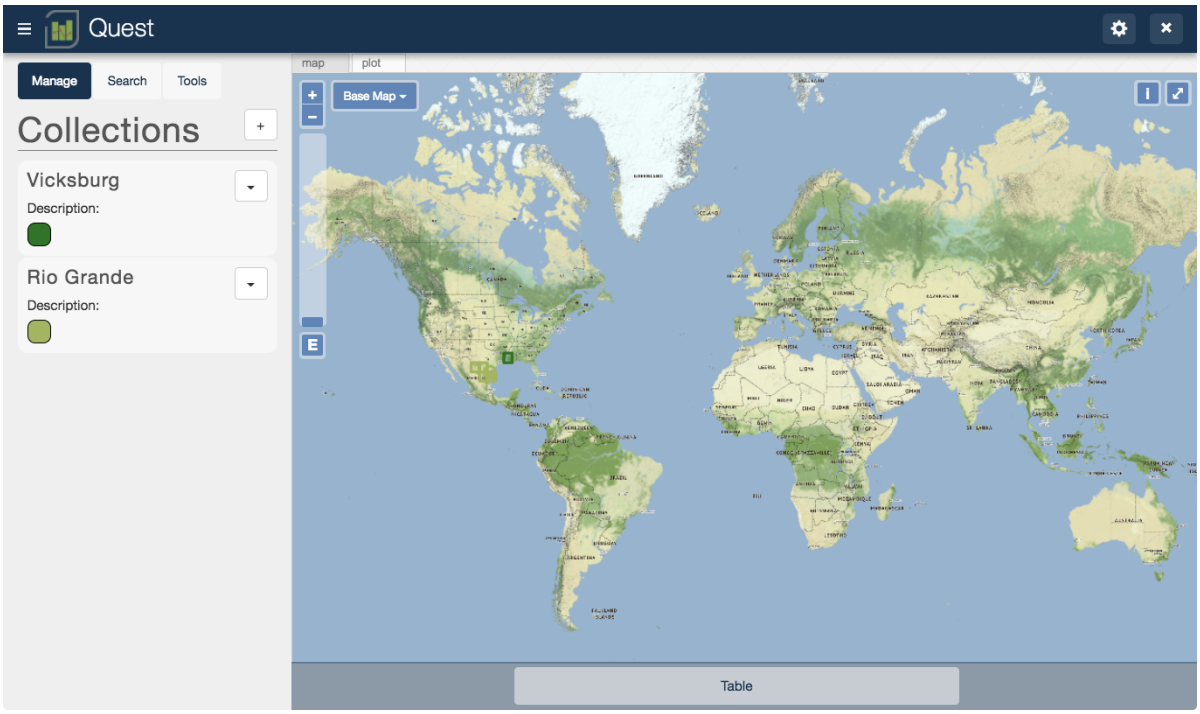

Figure 3. Web-based interface for Quest.

3 DISCUSSION

3.1 Current Limitations in Quest

Quest provides a flexible and extensible way to manage data throughout the data management cycle. It already has several provider plugins that allow it to search and download data from various sources. More provider plugins are currently being developed, but often the process of adding new data sources exposes current limitations.

Quest is currently able to perform searches on metadata harvested from its supported data sources. The metadata and the searching capabilities are currently focused on geographic information and parameters. While this is a good fit for many data sources including several data products from the USGS and from the NOAA, there are other data sources where the data are not easily searched using the location-parameter paradigm. Further work is needed to allow for geo-typical or geo-ignorant search options in Quest so that queries can be tailored to fit various types of data.

Additionally, the process of harvesting metadata from each provider only makes sense if the metadata are fairly static. Data repositories like HydroShare, where the data are potentially always in flux, present a challenge because it becomes too expensive to ensure that the metadata that Quest has cached is reasonably up-to-date. One remedy for this would be to expand HydroShare to allow for a query on any metadata that has been created or updated after a specified time. This would allow Quest to only query HydroShare for metadata that has changed since it last cached it.

Another challenge with querying across many data sources is the terminology used for parameters. Quest does not attempt to solve this problem in its entirety, but rather takes a limited approach at mapping a finite, controlled-vocabulary list of parameters to the parameters available from each provider. Improvements could be made here to better account for differences among equated parameters due to differing aggregations, units, collection methods, etc.

3.2 Current Limitations with Data Depot

Data Depot is a vital component for data management within the ERDC because of the need to share and collaborate over data while appropriately protecting sensitive data. We choose HydroShare as the application to serve as the ERDC Data Depot because it has a Python client that interacts with the REST API for querying and publishing data and because it is designed specifically for environmental data sets and models. Also, since HydroShare is built using iRODS, it has a robust and flexible data storage mechanism to allows for various data servers to be federated.

One challenge with Data Depot being a private, organizational instance of HydroShare is that the HydroShare source code, while open source, was not written with the goal of allowing multiple instantiations. As a result, HydroShare is not as customizable as it could be, and further work is needed to truly create a personalized instance of HydroShare for the ERDC.

While the ERDC has a need for a private organizational-level data repository, it also has the need to publicly publish data. This process could be facilitated if links between different instances of HydroShare could be established so that resources from one instance can be easily pushed to another.

3.3 Current Limitations with Quest Web

The Quest API provides a powerful scripting library for automating the data management cycle in workflows. Many times, however, modellers do not have a well-defined workflow that can be scripted and instead need the ability to interactively work with data. Quest Web provides a convenient way to interactively search for, retrieve, visualize, filter, and publish data without needing to install any software locally. While Quest web provides most of the functionality from the Quest API, it is currently primarily focused on the geographic data and does not fully support non-geographic data. It does provide some basic data visualizations, but these too could be expanded and made more flexible.

The use of the Param library to define options in Quest is key in allowing Quest Web to dynamically provide dialogs for the user to specify options when downloading, filtering, or publishing data. As new data providers, or filters are added to Quest, the Param-defined options can be interpreted by Quest Web dynamically, so no changes need to be made to the interface. This extensibility is an essential component for a data infrastructure that needs to constantly adapt to ever-evolving data needs.

4 CONCLUSIONS

Automation of the data management cycle has challenges due to lack of standardization of data formats and access, organizational or project restrictions on data publication, and access to software and data management tools. By building a comprehensive data infrastructure that has the ability to automate the data management cycle and provides a secure data repository and web-based interfaces, the ERDC is better equipped to focus on developing solutions through cutting-edge modelling, rather than being weighed down by data management tasks. This data infrastructure is completely composed of open source projects and thus can be reconstructed for any organization. Finally, though these tools have been developed specifically to support hydrologic modelling workflows, they are generic and extensible making them applicable to a broad range of data and workflows.

REFERENCES

GeoServer. http://geoserver.org/ (accessed 5.17.18).

Heidorn, P.B., 2008. Shedding Light on the Dark Data in the Long Tail of Science. Library Trends 57, 280–299. https://doi.org/10.1353/lib.0.0036

Horsburgh Jeffery S., Morsy Mohamed M., Castronova Anthony M., Goodall Jonathan L., Gan Tian, Yi Hong, Stealey Michael J., Tarboton David 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, 873–889. https://doi.org/10.1111/1752-1688.12363

iRODS. https://irods.org/ (accessed 3.30.18).

Michener, W.K., Allard, S., Budden, A., Cook, R.B., Douglass, K., Frame, M., Kelling, S., Koskela, R., Tenopir, C., Vieglais, D.A., 2012. Participatory design of DataONE—Enabling cyberinfrastructure for the biological and environmental sciences. Ecological Informatics, Data platforms in integrative biodiversity research 11, 5–15. https://doi.org/10.1016/j.ecoinf.2011.08.007

OpenLayers – Welcome. https://openlayers.org/ (accessed 5.17.18).

Param — Param 1.4.1-dev documentation. http://ioam.github.io/param/ (accessed 3.30.18).

PostGIS — Spatial and Geographic Objects for PostgreSQL. https://postgis.net/ (accessed 5.17.18).

Python Data Analysis Library — pandas: Python Data Analysis. https://pandas.pydata.org/ (accessed 5.17.18).

Swain, N.R., Christensen, S.D., Snow, A.D., Dolder, H., Espinoza-Dávalos, G., Goharian, E., Jones, N.L., Nelson, E.J., Ames, D.P., Burian, S.J., 2016. A new open source platform for lowering the barrier for environmental web app development. Environmental Modelling & Software 85, 11–26. https://doi.org/10.1016/j.envsoft.2016.08.003

Tarboton, D., Idaszak, R., Horsburgh, J., Heard, J., Ames, D., Goodall, J., Band, L., Merwade, V., Couch, A., Arrigo, J., Hooper, R., Valentine, D., Maidment, D., 2014. HydroShare: Advancing Collaboration through Hydrologic Data and Model Sharing. International Congress on Environmental Modelling and Software.

The Web framework for perfectionists with deadlines | Django. https://www.djangoproject.com/ (accessed 5.17.18).

Tolle, K.M., Tansley, D.S.W., Hey, A.J.G., 2011. The Fourth Paradigm: Data-Intensive Scientific Discovery [Point of View]. Proceedings of the IEEE 99, 1334–1337. https://doi.org/10.1109/JPROC.2011.2155130

USGS Water Data for the Nation. https://waterdata.usgs.gov/nwis (accessed 5.17.18).

xarray: N-D labeled arrays and datasets in Python — xarray 0.10.4 documentation. https://xarray.pydata.org/en/stable/ (accessed 5.17.18).