ABSTRACT
Web Applications for Managing Time Series Data
Matthew Michael Bayles
Department of Civil Engineering, BYU
Master of Science

The Consortium of Universities for the Advancement of Hydrologic Science, Inc. (CUAHSI) has developed a web-based client for discovering and downloading hydrologic data. Three web applications have been developed to augment the capabilities of this client and to link the discovery ability of the client with the powerful features of HydroShare, a management service for hydrologic data. These applications extend the features of these two systems by providing visualization services, an easy to use method for saving data from the CUAHSI HydroClient into HydroShare, and, finally, providing a simple approach for creating and running community created Python Jupyter Notebooks for time series data. Each of these applications was built using the Tethys Platform, a collection of tools built upon the Django framework, designed for hydrologic web applications. This paper describes the design and implementation of three web applications.

Keywords: CUAHSI, HydroShare, Web Applications, Tethys Platform
ACKNOWLEDGEMENTS

I want to thank my wife, Mackenzie, for all the support and encouragement she has given me over the last three years of working on this project. I could not have done it without her. I also want to thank Dr. Dan Ames, my committee chair, for working with and providing guidance to me during my years at BYU. I give special thanks to Shawn Crawley, Zhiyu Li, Ken Lippold, and the entire Hydroinformatics lab for helping me to create and improve these web applications. Finally, I would like to thank my parents for teaching me the importance of education and supporting me throughout my life.
# TABLE OF CONTENTS

LIST OF FIGURES .................................................................................................................. vii

1 Introduction ................................................................................................................................. 1

2 Design ........................................................................................................................................ 5

2.1 Web Application ....................................................................................................................... 5

2.2 Data Series Viewer ................................................................................................................... 6

2.2.1 Architecture ........................................................................................................................ 10

2.3 HydroShare Resource Creator ............................................................................................... 11

2.3.1 Architecture ........................................................................................................................ 12

2.4 Time Series Script Manager .................................................................................................... 13

2.4.1 Architecture ........................................................................................................................ 14

3 Results ....................................................................................................................................... 17

4 Conclusion ................................................................................................................................. 27

5 References ................................................................................................................................ 29
LIST OF FIGURES

Figure 2-1: Web application interaction diagram ................................................................. 6
Figure 2-2: Mockup diagram of Data Series Viewer ............................................................... 8
Figure 2-3: Example API using the Data Series Viewer ....................................................... 9
Figure 2-4: Sample URL query string .................................................................................. 9
Figure 2-5: Mockup diagram of the HydroShare Resource Creator ...................................... 11
Figure 2-6: Sample Reference Time Series file .................................................................. 12
Figure 2-7: Mockup diagram of the Time Series Script Manager .......................................... 14
Figure 3-1: CUAHSI HydroClient homepage ...................................................................... 18
Figure 3-2: Data Series Viewer with USGS data loaded .................................................... 19
Figure 3-3: Creating a HydroShare resource ..................................................................... 20
Figure 3-4: HydroShare resource page of the newly created resource .............................. 21
Figure 3-5: USGS and National Water Model data ............................................................... 22
Figure 3-6: Script Manager with data loaded and script selected ....................................... 23
Figure 3-7: Time Series Comparison Jupyter Notebook ...................................................... 24
Figure 3-8: Summary statistics for the USGS and National Water Model data .................. 25
Figure 3-9: Comparison of observed and predicted flow data ........................................... 26
1 INTRODUCTION

There is a vast array of useful hydrologic data available online, but these datasets can have hundreds of thousands of data points stored in a variety of formats. The problem becomes how to manage and view such vast quantities of data [Beran and Piasecki, 2009]. Offline spreadsheets have traditionally been used to manage the data; however, with the amount of data that is updated daily, spreadsheets are becoming unwieldy and less useful. In addition, there is a need to improve computer applications and performance to handle millions of data points. The next step in managing the ever-increasing amount of hydrologic data is the development of web-based tools that allow users to interact with data from any computer that is connected to the web without having to worry about transferring the data between computers or the specifications of their local machine.

The Consortium of Universities for the Advancement of Hydrologic Science, Inc. (CUAHSI) is developing tools to manage and view water resource data online [Tarboton et al., 2009]. One of those tools is a free map-based catalog, CUAHSI HydroClient, which allows users to browse and download hydrologic data from around the world. With over six million sites, the catalog is a powerful resource for scientists, engineers, and others who are searching for hydrologic data. As part of this initiative, CUAHSI and Brigham Young University (BYU) are working together to develop web-based applications to enhance the usefulness of the catalog.

The CUAHSI HydroClient references over 100 custom data servers running a suite of software applications made for storing hydrologic data. These servers host data from a variety of
agencies such as the United States Geological Survey (USGS) and the National Aeronautics and Space Administration (NASA) and are referred to as “HydroServers” [Horsburgh et al., 2010; Kadlec and Ames, 2012]. HydroServers are relatively simple to access and manage, and they provide all the tools necessary to store and publish hydrologic time series on the web. Each HydroServer can be registered with a central catalog, HIS Central, which allows a user to search through data from a variety of HydroServers. When data are requested from a HydroServer they are provided in the Open Geospatial Consortium (OGC) standard, known as CUAHSI WaterML [Zaslavsky et al., 2007], which is a custom implementation of extensible markup language (XML) built specifically for hydrologic data. Data from HydroServers can be accessed by outside applications using a suite of built in web services called WaterOneFlow, which return data in the WaterML format. However, due to the customizable nature of HydroServer, it is up to server administrators to ensure data is stored and available in proper WaterML format.

HydroServers provide some tools for visualizing data, but these tools are very limited in scope.

While HydroClient provides an effective interface for searching and retrieving time series data, it does not include tools to do anything with the data other than download it as a comma separated values file (CSV). HydroDesktop [Ames et al. 2012], the predecessor to HydroClient, was originally developed to provide a convenient means to access and visualize data stored on HydroServers. However, HydroDesktop only runs on Windows and lacks the accessibility of a web-based application, but it is one of only a few examples of a working, dynamic, WaterML viewer.

One of the goals of this project was to develop an application that that would make it possible for a user to store interesting subsets of data and then do something useful with the data. To achieve these goals it was necessary to leverage the power of another web-based application,
HydroShare [Tarboton et al., 2014]. HydroShare is a free online platform for sharing hydrologic data. Data can be stored in many different formats as resources and each resource can have multiple fields for metadata about the resource. It also features an application programming interface (API) for discovering and downloading hydrologic data.

This paper discusses three web applications that were created to extend the abilities of HydroClient by integrating HydroShare and providing a suite of tools for viewing and analyzing data. The remaining portion of this paper discusses the workflow and architecture of each of these applications and describes a test case of the kind of results that can be obtained using the applications.
2 DESIGN

2.1 Web Applications

Each of the applications was built on a custom web framework developed at BYU. This Tethys Platform [Swain, 2015] is designed specifically for building and hosting environmental and water resources web applications. Tethys is powered by Django, which allows applications to use the power of Python with a user interface written using JavaScript and HTML. It addition, Tethys also includes a Python software development kit and a suite of external software tools such as PostGIS and PostgreSQL. Tethys significantly streamlines the process of creating web applications due to the many software packages included.

Three web-based applications were created to demonstrate how HydroClient could interact with external applications such as HydroShare and Google Colaboratory (Figure 2.1). Google Colaboratory is a relatively new implementation of a Jupyter Notebook environment [Kluyver et al. 2016] provided free by Google. Jupyter Notebooks are scripts written in various languages that can be shared, edited, and hosted online. Currently, Google Colaboratory provides support for Python 2 and Python 3. Anyone with access to a Google account can run Jupyter Notebooks from their own Google Drive. This means that anyone with a Google account can run Python scripts in the cloud without having to download any files locally. An interaction diagram of how all applications are related is shown in Figure 2-1. This paper outlines the design and implementation of the Data Series Viewer, the HydroShare Resource Creator, and the Time Series Script Manager.
One of the major problems with HydroClient is the inability to visualize time series without having to download the data. Without a means of visualization, the process of discovering data trends and correlations through the client is a challenging process. Thus, the first application needed to be a simple, yet powerful, visualization tool that could download and display millions of data points quickly and efficiently. Data from HydroClient is in WaterML format which is a standard XML format; however, HydroClient does not require that all HydroServers return the correct file structure. As a result, the Data Viewer needed to have robust parsing algorithms to deal with the many possible discrepancies in the files.
While the original scope of the Data Viewer application was to parse WaterML files, the ubiquitous nature of storage formats for time series data required the addition of the ability to parse several other file formats. Other formats that were added include: ODM 2 [Horsburgh et al. 2016], a custom JSON implantation developed at BYU for storing references to time series named Reference Time Series with a file extension of refts.json which will be discussed later in this paper, and a specific NetCDF format derived from the NOAA’s National Water Model [NOAA, 2016]. The National Water Model uses data from various sources across the United States to make predictions based on a variety of variables and time ranges. The model outputs data into a NetCDF file. These files are structured so that each file contains the same timestamp for every output location in the United States. BYU has created a Tethys application that takes the NetCDF output and converts it so that each output location is stored in one NetCDF for all the timestamps. The application also allows a user to save their own NetCDF file containing their desired time range and monitor points as a HydroShare resource. These resources are compatible with Data Viewer.

The basic design of Data Viewer (Figure 2-2) features a graph that occupies the main portion of the viewing page. Below the graph, in a scrollable container, is a table displaying important metadata along with basic statistics. An important design consideration for this application was that it needed to be launchable from both HydroClient and HydroShare; the first step in connecting these two websites. The methods developed for connecting Data Viewer with HydroClient and HydroShare were used in the subsequent applications.
Figure 2-2: Mockup diagram of Data Series Viewer

One of the most important functions of the Data Series Viewer is to store the API that HydroClient uses to populate its list of approved applications. This API (Figure 2-3) includes the application name, description, URL, icon, and the range of time series required by the application. This data is displayed as JavaScript Object Notation (JSON), a well-established format for data storage and transfer across the Internet. The API is accessible from the base URL of the Data Viewer and is visible to any third-party application which allows outside applications to leverage the features of the created applications. The HydroClient hosts the code for custom launching patterns as the application requires it. In general, except for the Resource Creator, data values are passed to applications from HydroClient as URL parameters.
Figure 2-3: Example API using the Data Series Viewer

Figure 2-4 \textbf{Error! Reference source not found.} shows an example URL pattern with the URL parameters. The five parameters are: data source (Source), resource identifier (WofUri), quality control level identifier (QCLID), method identifier (MethodId), and source code identifier (SourceId). The data source indicates the source of the incoming data (HydroClient or HydroShare). This identifier is important for parsing because of the difference in data formats between HydroClient and HydroShare and simplifies the process of adding new data sources in the future. The resource identifier is the ID of the resource on the data sources server and provides the means of accessing the proper resource. The last three parameters are optional and exclusive to requests from HydroClient. These parameters allow Data Viewer to only display resources matching a certain method, quality control level, and source.

Figure 2-4: Sample URL query string
2.2.1 Architecture

The Data Viewer application uses JavaScript to retrieve the URL parameters and set up the user interface. Then, through an asynchronous JavaScript and XML (AJAX) request, the parameters are passed to a Python Controller which identifies the data source and file type, downloads the data locally and parses it, and then returns it as a JSON file to the JavaScript for display. A variety of functions and libraries are used to parse the data and display the results.

The two most critical components of the user interface (UI) are the graph and table controllers. Tethys includes several libraries for displaying graphs and tables in HTML; however, it was decided that the included libraries lacked the performance to display the potential hundreds of series and millions of data points. Third party, open source, projects were used to provide the graph and table interfaces.

The downloading and parsing of the time series data occurs in the Python controller. Python has many libraries that are simple to integrate into a project and these libraries are critical for parsing the various data formats. Without the flexible Python libraries, code would have had to be written to extract data from each of the file formats Data Viewer supports.

The Python controller uses parameters parsed by JavaScript to access the server where the specific resource is stored and downloads the requested resource locally in its native file format. Once the correct file is downloaded, the controller parses the file and extracts the values, dates, and other important metadata from the file and stores it in a Python dictionary. This dictionary is then returned to the AJAX call as a JSON string. The JavaScript then loops through the JSON, adding the metadata to the table and the actual time series to the graph. The graph only displays two different units at a time to simplify the user interface. Once all of the data has been loaded into the chart and table, they are displayed to the user.
2.3 HydroShare Resource Creator

One of the shortcomings of HydroClient is its inability to save specific time series for viewing later on a local machine, except as a comma separated values, CSV. A new Tethys application, the HydroShare Resource Creator, was created to integrate the storage ability of HydroShare with the data that can be accessed through HydroClient.

The Resource Creator user interface is designed to resemble that of Data Viewer (Figure 2-5). However, the table lists more information regarding the source of the data and how to access it, such as the URL and other parameters required to perform a WaterOneFlow request. This data is stored in the Resource Creator for easier future access. In addition to the table, there are fields for entering the resource title, an abstract, and keywords. These pieces of information are important metadata used in HydroShare to make the resource viewable to the public. Finally, the user can choose whether to create a reference time series or time series resource.

![HydroShare Resource Creator](image)

Figure 2-5: Mockup diagram of the HydroShare Resource Creator
The Resource Creator application was designed to output two different kinds of files to HydroShare. The first file type is a custom JSON format developed at BYU. This file type stores basic metadata and all parameters needed to access the data using the WaterOneFlow standard. An example file can be seen in Figure 2-6. Each file can store multiple time series. This format is designed to allow users to easily share data without having to transfer large files between computers. The second file type is an ODM 2 SQLite database which stores the actual time series and metadata. These two methods allow users flexibility in how their data is stored.

```json
...
"timeSeriesReferenceFile": { "id": "123", "name": "precipitation", "creation": "2022-01-01T00:00:00Z", "updated": "2022-01-02T00:00:00Z", "description": "Precipitation data from January 1, 2022.", "variables": [ { "variable": "precipitation", "units": "mm", "description": "Daily precipitation measurements." } ], "variablesCount": 1, "source": "https://example.com/precipitation_data.json" }
...```

**Figure 2-6: Sample Reference Time Series file**

### 2.3.1 Architecture

Data Viewer and Resource Creator have very similar structures. They share the same methods for downloading, parsing, and displaying data. The Resource Creator receives data differently than the other two applications. Due to the number of parameters required the, Resource Creator then information is sent as an HTTP POST request. The Resource Creator does not feature a graph; it offers additional functionality for uploading files to HydroShare.
There are two steps to this process. The first step is to parse the data into the correct format; the second is to upload the file to HydroShare. To correctly upload the data to an ODM 2 database, a Python script was obtained from the creators of ODM 2. Once the files are in the correct format, they are uploaded to HydroShare using a Python client written by the HydroShare team to access HydroShare’s API.

2.4 Time Series Script Manager

The final application created was designed to link HydroClient, HydroShare, and Google Colaboratory. Time Series Script Manager harnesses the power of Google’s Colaboratory, the data available on HydroClient, and the file sharing capability of HydroShare to give users a powerful and flexible new tool for working with hydrologic information.

Script Manager shares a similar layout with the other two applications (Figure 2-7). The main portion of the page is devoted to a table displaying important metadata about each time series and there is an option for associating a time series with an input to a Jupyter Notebook. The left panel contains functions for selecting a particular Notebook and displaying information about it. Finally, there are several buttons for viewing the Notebook, a button which allows the user to upload their own Notebook to HydroShare, and a button to open Google Colaboratory.
Figure 2-7: Mockup diagram of the Time Series Script Manager

Individual Jupyter Notebooks are stored on HydroShare. Script Manager is able to discover and display these Notebooks with basic information about them and can bring in data from HydroClient to be used in the script. The user can select the desired Notebook, select which time series to package with the script, and then automatically launch Google Colaboratory from their own Google Drive. Script Manager automatically populates time series data from the desired resource into the Notebook, allowing the user to run the Notebook without having to change a single line of code. In addition, to support the growth of the library of Notebooks available to users, Script Manager allows users to upload their own Notebook to HydroShare as a new resource.

2.4.1 Architecture

While the Script Manager involves more user interaction than the previous two applications, the basic architecture of data retrieval between the applications is similar. Retrieving and parsing data from HydroClient is handled in the same fashion. An important component of
this application is the ability to discover and display to the user a list of Jupyter Notebooks. The HydroShare Python client, while useful in many regards, does not provide powerful enough discovery functions to find the specific Notebooks for the application in the multitude of resources on HydroShare. To address this issue, a series of custom tags were created to aid in discovery. These tags, stored in the abstract of HydroShare resource, allow only scripts that work with a specific resource to be discovered. These tags were inspired by the templating tags Django uses on its HTML pages. The basic structure is as follows “{%Parameter Name% Description of parameter}.” There are three parameter options. The first parameter is “Application” which identifies the web application the resource was designed to function with. “Description” includes a brief description of what the Notebook calculates and returns. Any remaining tags are assumed to be time series variables needed in the Notebook. Currently, the only variables allowed are time series.

Each time series is displayed in a table with a drop down menu which allows users to assign the time series to required variables for the script. Once the user launches Google Colaboratory, the application checks the Google sign in status using an API provided by Google. This application uses the Google Drive API to save the Notebook to the user’s Google Drive and then launches Google Colaboratory. The Google API requires that an application register itself with Google to obtain a client ID and API token. These two pieces of information allow Google to track the usage of applications and enforce a quota system for specific applications. However, the quota size far exceeds the intended usage of this application. If a user is not signed in to Google, a popup window appears prompting the user to login to one of their Google accounts. Once the user has logged in, the application continues its execution.
The HydroShare resource ID of the script and the selected time series data are passed to a Python controller. This controller downloads the Jupyter Notebook from HydroShare. Each Jupyter notebook is a JSON file containing information on each cell. The information includes the actual contents of the cell and associated metadata such as if the cell is a code block or text. This information allows programs like Colaboratory to correctly format the Jupyter Notebook and display its contents to the user. The selected time series are inserted in the Notebook and the updated Notebook is returned as a JSON file to the JavaScript client. This JavaScript then executes a series of functions using the Google API to create a folder in the user’s Google Drive to contain any Jupyter Notebooks they upload. This folder will persist between sessions, allowing the user to review any Notebooks they have created. Once the folder is created, the JSON file containing the Notebook is uploaded to Google Drive and the user is automatically redirected to the Google Colaboratory instance containing their script.
3 RESULTS

The following section details a test case using all three applications. For this example, two datasets representing the same storm event were used. The first dataset comes from a USGS station along the Provo River at Provo, UT. The second dataset is a subset of the National Water Model for the same location. Both datasets will be viewed using the Data Series Viewer. Then a Time Series Resource on HydroShare will be created for the specific data range for the USGS data. Finally, the Time Series Script Manager will be used to package the data with a Jupyter Notebook that has been specially designed to compare the two datasets.

To begin, HydroClient is used to search for the desired USGS station along the Provo River. As can be seen from the Figure 3-1, there are many data services available for that location. Using the controls on the right, the user can filter the display results by organization, variable, and data range. In this case an interesting high flow event during the last two years is selected. Once the correct USGS site is identified, the application is launched with the requested time series.
USGS data will be viewed using Data Series Viewer. The time series of flow along the Provo River can be seen in Figure 3-2. In addition to viewing the data, important metadata about the series can be viewed as well and, by expanding the table basic statistics about the events are shown. An event from May 13, 2017 to May 19, 2017 is selected as it has a high flow. A one-week event will be used because the National Water Model application only allows output resources to have seven days’ worth of data at this time.
HydroShare Resource Creator application will be used to create a subset of the USGS time series for the desired date range (Figure 3-3). This subset allows us to more easily compare the National Water model with actual flow data because it narrows the time range. It also enables us to keep a personal copy of the data for future review on HydroShare. From this screen the user can specify the title, abstract, keywords, and whether to make the resource public.

Figure 3-2: Data Series Viewer with USGS data loaded
The HydroShare resource page shown in Figure 3-4 is for the new resource that was just created. This resource has a custom tag, “#{Application%Data Series Viewer%}”, added manually, which identifies that this resource can be opened by the Data Viewer application. Since Data Viewer and Script Manager can parse the same data types, this tag means that the Script Manager application can read this file as well.

Figure 3-3: Creating a HydroShare resource
Using the specified dates, BYU’s National Water Model was used to create a seven-day NetCDF file, containing the flow predictions of the Provo watershed during the specified time range, that was then stored in HydroShare. This file contains flow predictions for every stream segment located within the Provo Watershed. In addition, the ComID of the stream segment that coincides with the USGS station was recorded so that the correct stream segment is used. A ComID is an integer value that uniquely identifies the occurrence of each stream segment in the National Hydrography Dataset. The National Water Model NetCDF file can now be brought into the Data Viewer as shown in Figure 3-5. From the two graphs it can be seen that the model results match the shape of the recorded data. However, since the two-time series are in different units it is difficult to ascertain the difference in magnitude. To gain a better understanding of
how the two datasets are related, the Time Series Script Manager will be used to load the data into a Jupyter Notebook which has been written using a Python library developed at BYU for comparing time series.

![Graph showing discharge and flow data](image)

**Figure 3-5: USGS and National Water Model data**

To use Script Manager (Figure 3-6), both HydroShare resources created earlier need to be loaded by clicking “Add Time Series from HydroShare”. Data can also be loaded into this application following the same URL pattern the Data Viewer uses. The table lists any resources on HydroShare that are compatible with this application and includes both public resources and private resources of the current user.
Figure 3-6: Script Manager with data loaded and script selected

For this example, there are over 50 stream segments, each having velocity and flow data. The Script Manager interface includes searching and filtering based upon column data tools to aid the user in selecting their desired data. Using the BYU National Water Model application, the stream ComID of the USGS station was recorded. The dropdown menu on the left screen allows the user to select the desired script. Each script, when selected, displays a description and a list of the required time series data. Each available Notebook can also be viewed by clicking the “View Script” button which displays the Jupyter Notebook text. For this demonstration, the Time Series Comparison Jupyter Notebook will be used. This Notebook requires an observed and predicted time series. These variables are assigned to the individual time series using the dropdown menus next to each time series. If desired, each time series dataset can be viewed using the Data Viewer by clicking the “View Resource” link.
Once the user is ready to run the selected Jupyter Notebook, they select “Launch with Google Colaboratory”. The selected time series data is inserted into the proper variables in the Jupyter Notebook, the Notebook is uploaded to the user’s Google Drive, and the Notebook is opened in a new tab using Google’s Colaboratory software.

The Notebook display is shown in Figure 3-7. The user interface of Google Colaboratory is like that of Jupyter Notebook. The Python code is stored in cells and each cell can be run individually or as group.

![Figure 3-7: Time Series Comparison Jupyter Notebook](image)

The Python Library statistics package includes functions for merging the data to have the same time steps and date range. Figure 3-8 shows the output from all 50 available statistical
functions. These functions allow the user to compare multiple statistical functions with a single function call. The flexibility of Jupyter Notebooks allows the user to take this Notebook and add functionality such as showing a graph of the series (Figure 3-9) or providing additional statistical functions. Since the script is stored on their individual Google Drive account, the user always has access to their customized script. Once they have customized their Notebook, a user can upload their modified Notebook to HydroShare and share it with the public or keep it private.

<table>
<thead>
<tr>
<th>Metric Values</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean Error</td>
<td>-89.246458</td>
</tr>
<tr>
<td>Mean Absolute Error</td>
<td>130.399794</td>
</tr>
<tr>
<td>Mean Squared Error</td>
<td>29413.850226</td>
</tr>
<tr>
<td>Euclidean Distance</td>
<td>2222.954528</td>
</tr>
<tr>
<td>Normalized Euclidean Distance</td>
<td>1.154970</td>
</tr>
<tr>
<td>Root Mean Square Error</td>
<td>171.584665</td>
</tr>
<tr>
<td>Root Mean Squared Log Error</td>
<td>0.111500</td>
</tr>
<tr>
<td>Mean Absolute Scaled Error</td>
<td>4.312311</td>
</tr>
<tr>
<td>R^2</td>
<td>0.346984</td>
</tr>
<tr>
<td>Anomaly Correlation Coefficient</td>
<td>0.190893</td>
</tr>
<tr>
<td>Mean Absolute Percentage Error</td>
<td>7.656395</td>
</tr>
<tr>
<td>Mean Absolute Percentage Deviation</td>
<td>0.670006</td>
</tr>
<tr>
<td>Symmetric Mean Absolute Percentage Error (1)</td>
<td>4.049276</td>
</tr>
<tr>
<td>Symmetric Mean Absolute Percentage Error (2)</td>
<td>8.008733</td>
</tr>
<tr>
<td>Index of Agreement (d)</td>
<td>0.7830547</td>
</tr>
<tr>
<td>Index of Agreement (d1)</td>
<td>134.572385</td>
</tr>
<tr>
<td>Index of Agreement Refined (dr)</td>
<td>0.313660</td>
</tr>
<tr>
<td>Relative Index of Agreement</td>
<td>0.692558</td>
</tr>
<tr>
<td>Modified Index of Agreement</td>
<td>0.463484</td>
</tr>
<tr>
<td>Watterson’s M</td>
<td>0.328847</td>
</tr>
<tr>
<td>Mielke-Berry R</td>
<td>0.258767</td>
</tr>
<tr>
<td>Nash-Sutcliffe Efficiency</td>
<td>-0.536770</td>
</tr>
<tr>
<td>Modified Nash-Sutcliffe Efficiency</td>
<td>-0.372681</td>
</tr>
<tr>
<td>Relative Nash-Sutcliffe Efficiency</td>
<td>-0.595992</td>
</tr>
<tr>
<td>Legate-McCabe Efficiency Index</td>
<td>-0.372681</td>
</tr>
<tr>
<td>Spectral Angle</td>
<td>0.006006</td>
</tr>
<tr>
<td>Spectral Correlation</td>
<td>0.940940</td>
</tr>
<tr>
<td>Spectral Information Divergence</td>
<td>0.621444</td>
</tr>
<tr>
<td>Spectral Gradient Angle</td>
<td>1.520240</td>
</tr>
<tr>
<td>H1 - MME</td>
<td>-0.056660</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>H3 - RMSHE</td>
<td>0.111131</td>
</tr>
<tr>
<td>H4 - MME</td>
<td>-0.056615</td>
</tr>
<tr>
<td>H4 - AHE</td>
<td>0.001326</td>
</tr>
<tr>
<td>H4 - RMSHE</td>
<td>0.113910</td>
</tr>
</tbody>
</table>

**Figure 3-8: Summary statistics for the USGS and National Water Model data**
Figure 3-9: Comparison of observed and predicted flow data.
4 CONCLUSION

This paper has presented the design and implementation of three distinct web applications for connecting CUAHSI HydroClient with HydroShare. All three applications were designed using the Tethys Platform and are located on the HydroShare Tethys 2 portal (https://hs-apps-dev.hydroshare.org/). These applications allow users to view, save, and analyze time series data and share the results with other researchers and engineers.

All three of these applications are open source and can be deployed on any Tethys 2 server for use by other organizations. Each application can easily be modified to work with new data sources and types of data. These applications demonstrate the potential of connecting web applications to a large data catalog such as HydroClient and a file storage service like HydroShare. Future developers can use the model established by these applications and apply it to create their own applications.
5 REFERENCES


