Environmental Modelling and Software 135 (2021) 104888

Contents lists available at ScienceDirect

Environmental Modelling and Software

journal homepage: http://www.elsevier.com/locate/envsoft

Toward open and reproducible environmental modeling by integrating online data repositories, computational environments, and model Application Programming Interfaces

Young-Don Choi [a], Jonathan L. Goodall [a,*], Jeffrey M. Sadler [a], Anthony M. Castronova [b], Andrew Bennett [c], Zhiyu Li [d], Bart Nijssen [c], Shaowen Wang [d], Martyn P. Clark [e], Daniel P. Ames [f], Jeffery S. Horsburgh [g], Hong Yi [h], Christina Bandaragoda [c], Martin Seul [b], Richard Hooper [i], David G. Tarboton [g]

[a] Department of Engineering Systems & Environment, University of Virginia, Charlottesville, VA, USA
[b] Consortium of Universities for the Advancement of Hydrological Science, Inc, Cambridge, MA, USA
[c] Department of Civil and Environmental Engineering, University of Washington, Seattle, WA, USA
[d] Department of Geography & Geographic Information Science, University of Illinois at Urbana-Champaign, Illinois, USA
[e] Centre for Hydrology and Coldwater Laboratory, University of Saskatchewan, Canmore, Alberta, Canada
[f] Department of Civil and Environmental Engineering, Brigham Young University, Provo, UT, USA
[g] Department of Civil and Environmental Engineering, Utah State University, Logan, UT, USA
[h] Renaissance Computing Institute, University of North Carolina at Chapel Hill, North Carolina, USA
[i] Department of Civil and Environmental Engineering, Tufts University, Medford, MA, USA

ARTICLE INFO

Keywords:
Open hydrology
Reproducibility
Modeling frameworks
JupyterHub
Containers

ABSTRACT

Cyberinfrastructure needs to be advanced to enable open and reproducible environmental modeling research. Recent efforts toward this goal have focused on advancing online repositories for data and model sharing, online computational environments along with containerization technology and notebooks for capturing reproducible computational studies, and Application Programming Interfaces (APIs) for simulation models to foster intuitive programmatic control. The objective of this research is to show how these efforts can be integrated to support reproducible environmental modeling. We present first the high-level concept and general approach for integrating these three components. We then present one possible implementation that integrates HydroShare (an online repository), CUAHSI JupyterHub and CyberGIS-Jupyter for Water (computational environments), and pySUMMA (a model API) to support open and reproducible hydrologic modeling. We apply the example implementation for a hydrologic modeling use case to demonstrate how the approach can advance reproducible environmental modeling through the seamless integration of cyberinfrastructure services.

1. Introduction

There is a growing acknowledgment and awareness of the reproducibility challenge facing computational environmental modeling fields (Hutton et al., 2016; Stagge et al., 2019) as well as in other computational modeling disciplines (Baker, 2016; McNutt, 2014; National Academies of Sciences, 2019). According to a survey of 1576 researchers, about 70% had tried but failed to reproduce published research and 90% agreed that the problem of reproducibility is a critical problem for scientific advancement (Baker, 2016). Within the hydrology and water resources fields, Stagge et al. (2019) analyzed 360 articles in six leading journals to understand if their data were available online and if the study results were reproducible. Their analysis showed that only 5.6% of the articles had data and model code available online along with directions for use, and only 1.1% were fully reproducible while 0.6% were partially reproducible. There are many possible reasons for this outcome; however, we argue along with others that advances in the cyberinfrastructure that enable modern computational science is critical

* Corresponding author. University of Virginia, Department of Engineering System and Environment, University of Virginia, 151 Engineers Way, P.O. Box 400747, Charlottesville, VA, 22904, USA.
E-mail address: goodall@virginia.edu (J.L. Goodall).

https://doi.org/10.1016/j.envsoft.2020.104888
Accepted 29 September 2020
Available online 6 October 2020

to achieving reproducible research (Hut et al., 2017; Hutton et al., 2016).

Reviewing recent research toward this goal of improving the underlying cyberinfrastructure necessary to support reproducible computational studies, we see three distinct thrusts: 1) open sharing of data and models online, 2) encapsulating computational environments through containers and self-documented computational notebooks, and 3) creating Application Programming Interfaces (APIs) for programmatic control of complex computational models. A major effort to improve the open sharing of data and models is the FAIR (Findable, Accessible, Interoperable, Reusable) guiding principles for scientific data management and stewardship (Wilkinson et al., 2016). However, FAIR principles speak primarily to openness, which is essential but insufficient on its own for addressing reproducibility of computational software and computational environments (Bast, 2019). Ince et al. (2012) argued that, even with well-developed data and software sharing capabilities, it remains challenging to reproduce published results due to difficulties in documenting computational environments needed to repeat past studies. Moreover, they found this especially true for operating system environments and software dependencies that can cause unpredictable differences with even slight changes in model source code or configuration.

To address this need, a second thrust in recent research is aimed at overcoming the difficulties with sharing complete computational software environments. Research that has focused on improving the sharing of well documented data and software workflows for computational studies includes Stodden and Miguez (2013), for example, who proposed sharing data, algorithms, and workflows to utilize and verify published results. Similarly, Gil et al. (2016) suggested the best practices of sharing data, software, and documents in an open and transparent way using a high-level roadmap of approaches to strengthen reproducibility in the geosciences. In the meantime, the broader information technology community has introduced the concept of containers as a means for encapsulating computational environments (Kurtzer et al., 2017; Merkel, 2014). The result of this work has benefited computational modeling fields and led to efforts to improve the preservation of operating system and software dependencies, strengthening reproducibility in computational research (Boettiger, 2015; Brinckman et al., 2019). Containerization technologies such as Docker (Merkel, 2014) have been used to reproduce computational modeling environments without requiring users to install additional dependencies (Boettiger, 2015; Signell and Pothina, 2019). Software tools like Sciunit (Essawy et al., 2018; Yuan et al., 2018) ease the process of containerizing, sharing, and tracking scientific applications, lowering the barrier to entry for researchers to use containerization tools.

Containerization has also led to the ability to create new modeling environments and deploy them through interactive, online analysis environments such as JupyterHub (Kluyver et al., 2016). Jupyter notebooks are quickly growing in use and popularity in computational fields as a means to document studies as a mix of formatted text, mathematical equations, and executable code with in-line visualizations resulting from the code (Kluyver et al., 2016). JupyterHub is a cloud-based software that utilizes containerization to support the execution of multiple Jupyter notebooks simultaneously. Recent advances leveraging Jupyter for environmental modeling include work by Castronova et al. (2018) who created the CUAHSI JupyterHub to support online hydrologic modeling and analysis, Yin et al. (2017) who created a TauDEM (Tarboton, 1997) modeling environment with JupyterHub, Eynard-Bontemps et al. (2019) who created the PANGEO project that supports big data studies in the geosciences and heavily leverages JupyterHub, and Bandaragoda et al. (2019) who used JupyterHub within a larger knowledge infrastructure to support earth system modeling. Recent work has also begun to explore combining external computational environments including high performance computing (HPC) and high throughput computing (HTC) cyberinfrastructure for model execution directly through Jupyter notebooks (Lyu et al., 2019). That work also takes advantage of containerization concepts to easily port preconfigured model execution environments to available computational resources.

The third thrust we observe in recent research is efforts to create APIs for computational environmental models. While many models have Graphical User Interfaces (GUIs) for improving the usability of the models, APIs are different in that they facilitate programmatically interacting with a simulation model to configure input files, execute models, and analyze model outputs. Python (https://www.python.org) and R (https://www.r-project.org) are common programming languages used for creating model APIs. Python has examples including model APIs for the Stormwater Management Model (PySWMM, McDonnell, 2017), MODFLOW (FloPy, Bakker et al., 2016), Hydrologic Simulation Program in Fortran (PyHSPF, Lampert and Wu, 2015), and Precipitation Runoff Modeling System (PRMS-Python, Volk and Turner, 2019). R has examples including model APIs for TOPMODEL (topmodel, Buytaert, 2011), SWAT (SWATmodel, Fuka et al., 2014), and TUW model (TUWmodel, Viglione and Parajka, 2020). These model APIs help by abstracting low-level programmatic details of input file manipulation and model execution operations from end users. In this way, they are particularly useful when combined with computational notebooks for creating self-documented modeling studies that can be more easily understood and reproduced by both modelers and non-modelers alike.

While work along each of these thrusts – online data repositories, computational environments leveraging containerization and computational notebooks, and model APIs – is important individually, integrating these three thrusts offers a powerful approach for reproducible computational modeling. Recent research has started to explore this integration includes 1) the GI-RHESSys (Green Infrastructure-Regional Hydro-Ecological Simulation System) Jupyter environment created for Green Infrastructure (GI) landscape designs and modeling output using JupyterHub (Leonard et al., 2019), 2) the Landlab model (Hobley et al., 2017) with recent work to implement Landlab within JupyterHub as a knowledge infrastructure (Bandaragoda et al., 2019), and 3) the HydroTerre system (Leonard and Duffy, 2016) that links an online data repository with the Penn State Integrated Hydrologic Model (PIHM). While these examples focused on supporting individual modeling use cases, they reveal general patterns of infrastructure components necessary to implement their systems. Our aim is to build on this past work by first presenting this general pattern as a general approach that can be followed for building new modeling systems. Second, we provide an example implementation of the general approach that can be easily expanded to support any computational environmental model that is containerized and has an accompanying model API.

The objective of this research is, therefore, to put forward a general approach or framework for integrating online data repositories, computational environments, and model APIs to enable more open and reproducible environmental modeling. In the Methodology section, we first present a high-level design of the approach describing each of the three components in more detail while also discussing different options available for online repositories, notebook-based and containerized modeling environments, and model APIs. We then present an example implementation that makes use of HydroShare as an online repository, CUAHSI JupyterHub and CyberGIS-Jupyter for Water as computational environments, and pySUMMA as an example model API. In the Results and Discussion section, we present the results of applying the example implementation to reproduce a prior hydrologic modeling study (Clark et al., 2015b) and discuss the difficulty and nuance in claiming to achieve reproducibility. We also present limitations of the work that could be a focus of future research. Finally, we conclude by summarizing the findings and emphasizing their contribution to the larger goal of making past and future studies simpler to reproduce through advances in cyberinfrastructure.

2. Methodology

In this section, we describe the general approach being put forward for open and reproducible environmental modeling (Section 2.1) and then present an example implementation of this general approach for hydrologic modeling (Section 2.2).

2.1. Overview of general approach and description of system components

The general modeling system approach considered in this research consists of three primary components (Fig. 1). Component 1 is the online repository where data, models, and notebooks can be openly shared with the community. Component 2 is the JupyterHub computational environment where containerized models can be executed using notebooks. Component 3 consists of a collection of model APIs, one for each model supported within the system, that allow for programmatic configuration, execution, and visualization through computational notebooks. The three components are integrated through seamless data transfers to create a powerful framework for open and reproducible modeling analyses. In practice, we anticipate that this general approach or framework may have many different physical implementations, where different technologies may serve the needs of specific subcommunities within the broader environmental modeling field. We demonstrate one such implementation in Section 2.2 for the hydrology community. In the following subsections, we describe each of these components in more detail while also providing examples of each that are available for integration.

2.1.1. Online repository

Online repositories allow for storing, sharing, and publishing data, metadata, and other resources required to reproduce computational research findings. These online repositories often support a rich set of user-friendly features such as metadata capture, persistent digital object identifiers (DOIs), and extensive APIs for programmatically creating, updating, and deleting resources. They also often support various data types such as documents, figures, code, audio, and video with metadata tailored to each data type. Some examples of online repositories used by researchers include DataOne member nodes (https://www.dataone.org), FigShare (https://figshare.com), Harvard Dataverse (https://dataverse.harvard.edu), and HydroShare (https://www.hydroshare.org).

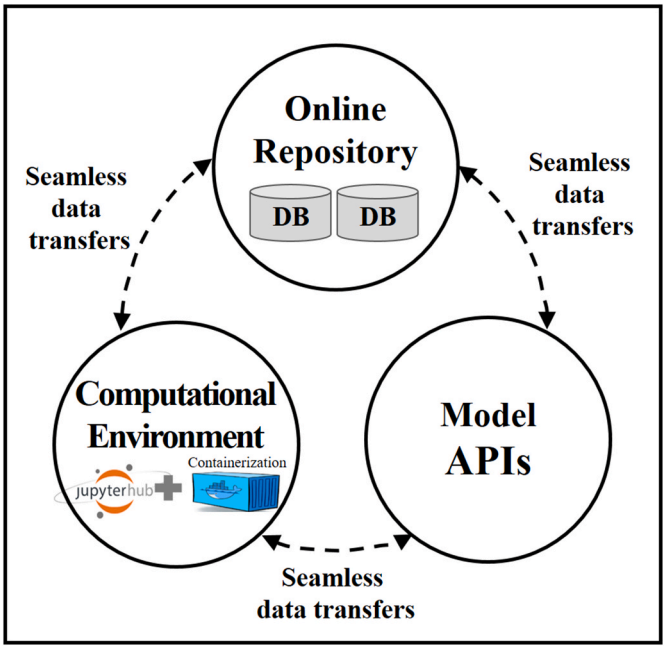

Fig. 1. A general modeling approach consisting of three primary components with seamless data transfers for open and reproducible environmental modeling.

Many online repositories serve broad scientific communities and, therefore, maintain only general and widely applicable capabilities. Others are more targeted to specific communities and, as a result, can offer more specific functionality. Environmental modeling, for example, is not a common use case for many repositories that focus on more general data sharing needs (e.g., FigShare). Environmental models, however, have their own characteristics that consist of software, input and output files, and data processing workflows. Morsy et al. (2017) described these unique needs of models being stored in data repositories and presented a data model design including metadata descriptions for key modeling objects to support flexible and applicable model sharing framework. This design is implemented within the HydroShare data repository, allowing for describing and sharing more specific model resource types.

2.1.2. Computational environment

A computational environment serves as a gateway for model configuration, execution, and post-processing. In the case of model execution, environmental modeling often includes complex simulation models along with data pre- and post-processing software, all with software dependencies that range from the operating system, to modules used within a model engine, to libraries used by data processing and analysis software (e.g., Python libraries). Without the ability to replicate a computational environment, slight inconsistencies in software dependencies can result in well-documented model studies failing when ported to a new machine. Without the use of recent innovations like containers, documenting the exact computational environment used in an analysis is difficult, time consuming, and error prone. To overcome these challenges, Docker (Merkel, 2014) and Singularity (Kurtzer et al., 2017) have emerged as containerization techniques used to encapsulate a computational modeling environment, as described further in the implementation (see Section 2.2).

Along with containers, computational gateway interfaces are also critical to lowering the barrier to entry and supporting more open and reproducible modeling in online computational environments. With the emergence of JupyterHub as a gateway innovation, there has been an increased interest in cloud-based modeling environments for creating, editing, and running computational notebooks. Markham (2019) reviewed five popular cloud services that support computational notebooks (Table 1). We reviewed two additional cloud services, 1) CUAHSI JupyterHub (hereafter CUAHSI JH) and 2) CyberGIS-Jupyter for Water (hereafter CyberGIS JW), and included them in Table 1 as well. The environments range from scientific services (e.g., the CUAHSI JH and CyberGIS JW that are used in this work) to more general services such as Binder (Jupyter Project et al., 2018). Large technology companies including Google and Microsoft have provided notebook execution environments such as Google Colab and Microsoft Azure Notebooks, demonstrating the popularity and growing interest in a variety of fields. Many cloud services have adopted the default Jupyter interface available from the Jupyter project without modification, while others have modified this interface to customize it for their own purposes (Markham, 2019). Furthermore, many cloud services support Python, R and other languages as well. Interface similarity in Table 1 considers available menus, buttons, and other visual elements that make up the user interface, and how different they are from a default Jupyter interface. All services listed in Table 1 are candidates for integration into an implementation of the modeling system described in this paper.

2.1.3. Model APIs

An API defines a set of protocols or tools to communicate with an operating system, database, network, and other lower-level aspects of a software system (Reddy, 2011). The abstraction provided by an API has benefits (Brooks, 2013) including: 1) flexibility and efficiency for data access, 2) personalization to customize the functionality that users

Table 1
Comparison of interface similarity and supported languages of cloud services for executing computational notebooks (expanded from Markham, 2019).

Cloud Services	CUAHSI JH	CyberGIS JW	Binder	Kaggle Kernels	Google Collaboratory	Microsoft Azure Notebooks (free plan)	CoCalc (free plan)
Interface similarity to Jupyter	100%	100%	100%	70%	60%	100%	95%
Supported Languages	Python 3, R	Python 3, R	Python 3, R, Julia, Many others	Python 3, R	Python 3, Swift	Python 3, R, F#	Python 3, R, Julia, Many others

access the most, and 3) reusability of code to work more productively. Examples of widely used APIs include the Google Maps API for map services and the Twitter API for social networking services. Services also widely exist for scientific systems relevant to environmental modeling including the HydroShare REST (Representational State Transfer) API for sharing and publishing water data as well as APIs for a growing number of environmental models.

In this study, we focused on Python-based model APIs and reviewed a series of model APIs including PRMS-Python (Volk and Turner, 2019) and PyHSPF (Lampert and Wu, 2015) to better understand how they are designed and structured. Doing this can help to inform the design and structure of future APIs created to support specific environmental models. We observed that model API functionalities fell into three categories: model input, model execution, and model output (Table 2). For PRMS-Python, as an example, input files often have corresponding Python modules that can be used for data manipulation. For PyHSPF, as an example, the Python modules do not have a one-to-one correspondence with the core model files and modules. Instead, the API designs include a higher-level abstraction to consider core classes needed for interacting with the model.

From this review, we suggest that communities of modelers (e.g., researchers or groups of researchers) who are considering building a model API for a specific environmental model begin with answering the following questions: 1) What configuration and input files should be exposed through the API to allow for programmatic changes and what are the logical classes for organizing these model input configuration attributes? 2) What methods and attributes should the API expose for executing the model and refining the model through, for example, calibration or sensitivity analysis? 3) What are common visualizations of the model output that many users would wish to produce? Creating a model API with this functionality in a well thought through design will serve as a solid foundation for future extensions to the software. Furthermore, the extent to which environmental model APIs can adopt conventions for the organization of their design and structure will allow users to more easily learn new model APIs by having some consistency across model APIs.

2.2. Example implementation

In this section, we present one possible physical implementation of approach described in the prior section. This example implementation uses HydroShare as the online repository, CUAHSI JH and CyberGIS JW the computational environments, and pySUMMA as one of potentially many model APIs within the system. While this example implementation targets the needs of the hydrologic modeling community, we anticipate that multiple other permutations of the technologies described in the prior section could be assembled to meet the needs of other environmental modeling communities.

2.2.1. HydroShare as the online repository

We used HydroShare as the data repository in our example implementation due to both its flexibility and tailored functionality for supporting environmental modeling use cases. HydroShare is an online repository tailored for the needs of the hydrologic community, but general enough to satisfy other environmental modeling needs (Tarboton et al., 2014). HydroShare defines a "Resource" as "the fundamental unit of digital content in HydroShare that contains data and/or model files and their corresponding metadata" (Horsburgh et al., 2016). HydroShare supports various data content types such as geographic raster (GeoTIFF), multidimensional arrays (NetCDF), geographic features (Shapefile), and time series as aggregations within a single HydroShare resource. HydroShare also supports collections as a resource type that holds a list of related HydroShare resources that can be referenced with a single unique identifier. Furthermore, realizing that data associated with models have their own characteristics, HydroShare defines specific resource types for a model program (the software) and a model instance (the input and output files for a specific model run) (Morsy et al., 2017). Resources with these two resource types are related through the "*ExecutedBy*" attribute of a model instance, which points to the specific model program resource used to execute that model instance. This design allows for a one-to-many link between a model program that is used to execute many different model instances built for different geographic locations or to address different research questions.

The methodology for sharing computational modeling resources is shown in Fig. 2. First, the user creates a model program resource for each version of a model program software used in the analysis. This resource can include the source code, executable, and container for the model program itself, or a link to one or more of these resources shared in a system external to HydroShare (e.g., in GitHub, BinderHub or DockerHub). Second, the model instance resources are created to store and

Table 2
General categories for a model API mapped to examples from PRMS-Python and PyHSPF.

General Categories	API Objective	PRMS	PRMS-Python	HSPF	PyHSPF
(a) Model Input	Generating and manipulating model input	-control file -data file -parameters file	-prms_config.txt -data.py -parameters.py	-control file -watershed data -management file	- wdmutil.py - watershed.py - hspfmodel.py . . .
(b) Model Execution	Executing and refining models	-shell script	-simulation.py -scenario.py -optimization.py	-shell script	- forecaster.py - extract.py - calibratormodel.py . . .
(c) Model Output	Visualizing and analyzing model output	-text file	-optimizer.py -utils.py	-text file	- gisplots.py - forecastplots.py - autocalibrator.py . . .

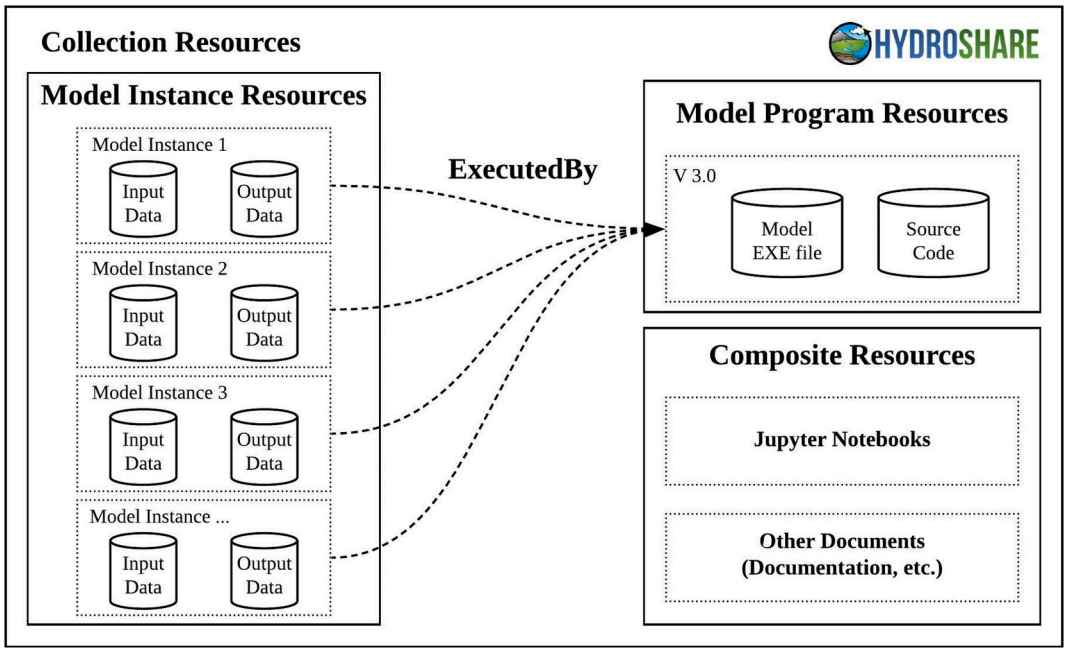

Fig. 2. A methodology for sharing resources used for a modeling analysis through HydroShare.

describe the input data required to execute the model and can optionally store the output after the model is executed. Then the model instance is linked to a specific model program resource using the "*ExecutedBy*" metadata term. A separate composite resource is used to store Jupyter notebooks that describe the overall analysis workflow. Finally, a collection resource is used to combine and conveniently share all of the resources used to complete the study.

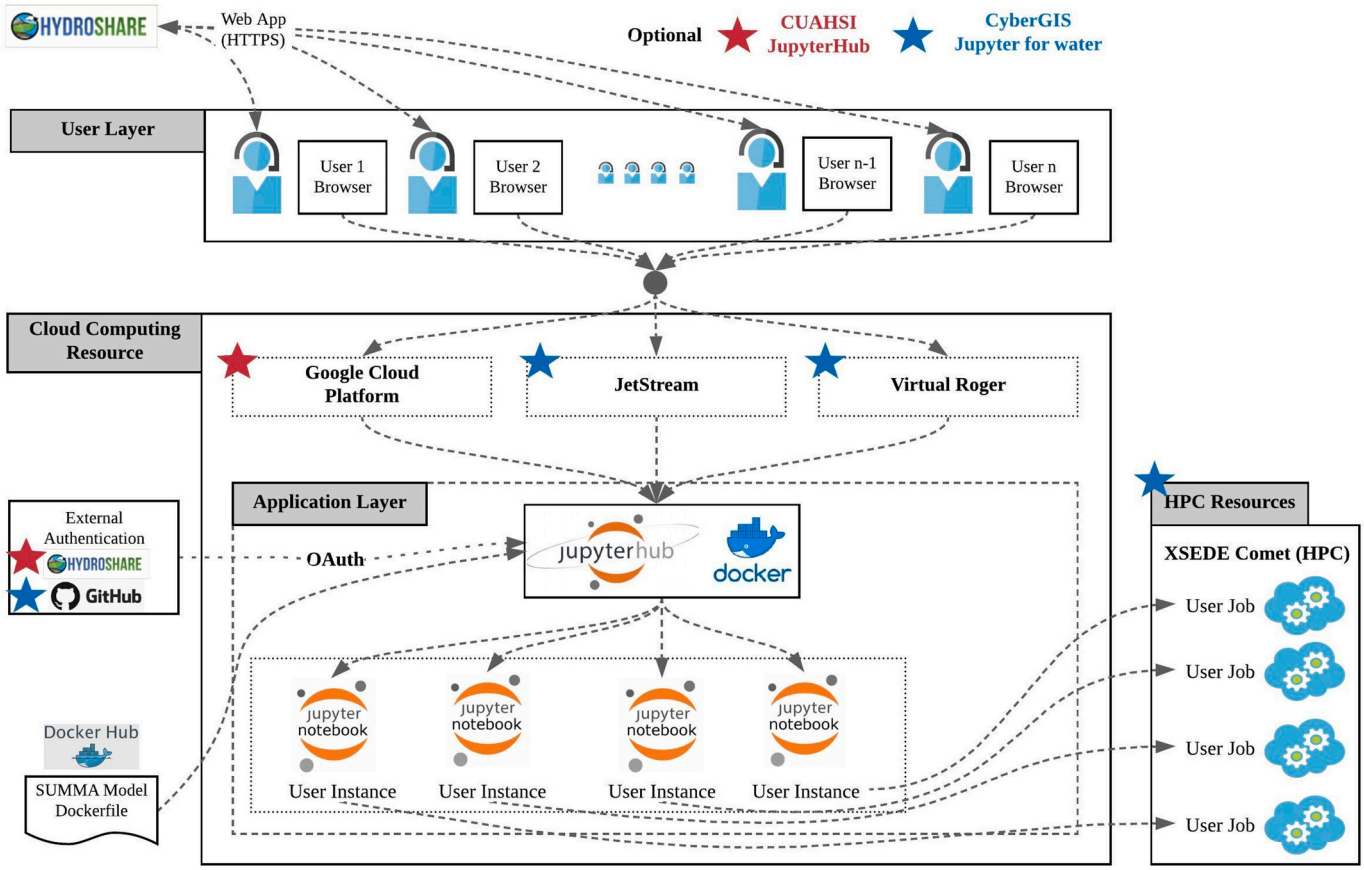

Fig. 3. The CUAHSI JH and CyberGIS JW environments with model execution environments configured as Docker images to support concurrent model execution through Jupyter notebooks.

2.2.2. JupyterHub as the computational environment

We integrated both the CUAHSI JH and CyberGIS JW as computational environments in our example implementation. We chose these environments because both are publicly available and aimed at scientific modeling in the water and environmental communities. Moreover, both systems allow for seamless data transfer with HydroShare as a data repository supporting the necessary interoperability between these two components of the general framework. This data transfer is enabled through the HydroShare REST API and the standardization of HydroShare resource data structures.

The CUAHSI JH is a cloud computing environment on the Google Cloud Platform specifically designed to support research and education in the water sciences (Fig. 3). To support a variety of applications, it leverages environment profiles that allow users to choose the ideal computing configuration for their work. Each of these profiles is a separate containerized environment that has been built with a specific set of software to support various water science use cases. Currently, the CUAHSI JH consists of seven profiles that range from scientific Python and R to HydroLearn (https://www.hydrolearn.org), educational modules and hydrologic modeling. In addition, the CUAHSI JH supports persistent data, meaning user-created content is stored between sessions and shared between profile environments. Moreover, this environment enables users to install custom software using conda virtual environments. For this study, we created a "Python 3.7 SUMMA Modeling" profile to support SUMMA 3.0 modeling environment using a Dockerfile in CUAHSI JH.

Another model execution environment interoperable with HydroShare, CyberGIS JW, is a well-tailored CyberGISX (https://cybergisxhub.cigi.illinois.edu) instance to serve the fast-emerging needs for data-intensive and reproducible research in the environmental modeling community (Fig. 3). Overall, CyberGIS JW is similar to CUAHSI JH, but CyberGIS JW also includes interoperability with advanced cyberinfrastructure resources such as Virtual ROGER (a cyberGIS supercomputer hosted by the CyberGIS Center for Advanced Digital and Spatial Studies at the University of Illinois) and XSEDE Comet (an HPC resource on the Extreme Science and Engineering Discovery Environment) for model execution support. Lyu et al. (2019) describe how to use HTC through a Jupyter notebook using SUMMA as an example case in CyberGIS-Jupyter (beta), which is the previous version of CyberGIS JW. Currently CyberGIS JW is supporting Landlab (Hobley et al., 2017) and RHESSys (Tague et al., 2004) modeling environments. For this study, we created a SUMMA modeling environment using a Dockerfile. Users can use this SUMMA modeling environment via a SUMMA kernel. For use of HPC resources, CyberGIS JW requires a Singularity image to support a computational modeling environment in XSEDE because CyberGIS JW and XSEDE are separately placed. Also, CyberGIS JW needs a particular library to connect to computational resources for submitting jobs and data exchange in XSEDE.

2.2.3. pySUMMA as the model API

The model API pySUMMA was created through this research as an example model API. pySUMMA wraps the hydrologic model Structure for Unifying Multiple Modeling Alternative (SUMMA) (Clark et al., 2015a). SUMMA was selected for this study because it is a general hydrologic modeling environment offering the ability to conduct model experiments with controlled and systematic evaluation of multiple model representations of hydrologic processes and scaling behavior. The SUMMA model simulates both the thermodynamics, the storage and flux of energy such as the heat balance of the vegetation canopy, snow, and soil affected by the radiative fluxes, as well as the hydrology, the storage and transmission of water (for example, vertical and lateral transmission of water through vegetation canopy, snow, soil, aquifer and river within a catchment system). The flexible hierarchical spatial structure of SUMMA supports different spatial configurations including the size and shape of model elements with Grouped Response Units (GRUs) (Kouwen et al., 1993) and Hydrologic Response Units (HRUs). In addition, the flexible structure enables researchers to consider the lateral flux of water across the model domain and complex topographical properties like hillslopes and riparian areas. This flexibility within SUMMA enables hydrologists to find solutions for the application of scaling behavior in relation to different physical processes.

SUMMA also enables hydrologists to select the appropriate physical process methods and model complexity. This process implements a modular structure that is supported by the conservation equations to calculate each process in a controlled and systematic way. This unified process helps users to concentrate important physical parameterizations with higher complexity and, conversely, to simplify specific processes to minimize uncertainty according to the purpose and characteristics of biophysics and hydrology. Moreover, the structure of SUMMA, which consists of a core (solver) and outer branches, enables the output of a numerical solution from SUMMA so that the user can evaluate the accuracy and efficiency of the model. Therefore, SUMMA supports flexibility to simulate different options of physical processes and numerical solutions.

We designed and implemented pySUMMA as a model API for SUMMA using the questions proposed in Section 2.1.3 for guiding the design of a new model API (Table 3). For the model input category, there are six configuration files to manipulate SUMMA input: 1) *File Manager*, 2) *Decisions*, 3) *Forcing File List*, 4) *Model Output*, 5) *Param Trial*, and 6) *Local Attribute* files. To expose the first four configuration files through the API, we created *file_manger.py, decisions.py, force_file_list.py, output_control.py* and *option.py*. For the rest of the configuration files, we created *assign_trial_params* and *assign_attributes* methods in *Simulation.py*. In the model execution category, we created *Simulation.py* to use the model execution command conveniently from the shell script format so that users do not need to edit manually every time. We also created two options to execute the SUMMA model, 'local' and 'docker', to satisfy different user requirements. Finally, the output format of SUMMA is NetCDF; therefore, we created *plotting.py* for visualization considering the output variables and their output structure in NetCDF.

The classes of pySUMMA are shown in Fig. 4. A Simulation module (*Simulation.py*) is used as the initial Python module to start a pySUMMA API and combine most pySUMMA functionalities, such as manipulating configuration files and executing SUMMA. After creating a pySUMMA simulation object, users can manipulate six configuration files. A File manager module (*file_manager.py)* reads and manipulates a *File Manager* file which controls the location of every configuration file for the SUMMA model. For example, the *File Manager* file sets the directory and

Table 3
Implementation of a model API for SUMMA.

General Categories	Questions	SUMMA	pySUMMA
(a) Model Input	(1) What configuration and input files should be exposed through the API to allow for programmatic changes?	- file manager - decision file - forcing file list file - model output file - param trial file - local attribute file	- file_manager.py - decisions.py - force_file_list.py - output_control.py - option.py
(b) Model Execution	(2) What methods and commands should the API expose for executing the model?	- shell script - SUMMA compilation (summa.exe) or Docker	- simulation.py - SUMMA compilation (summa.exe) or Docker
(c) Model Output	(3) What are common visualizations of the model output that many users would wish to produce?	- output NetCDF	- plotting.py

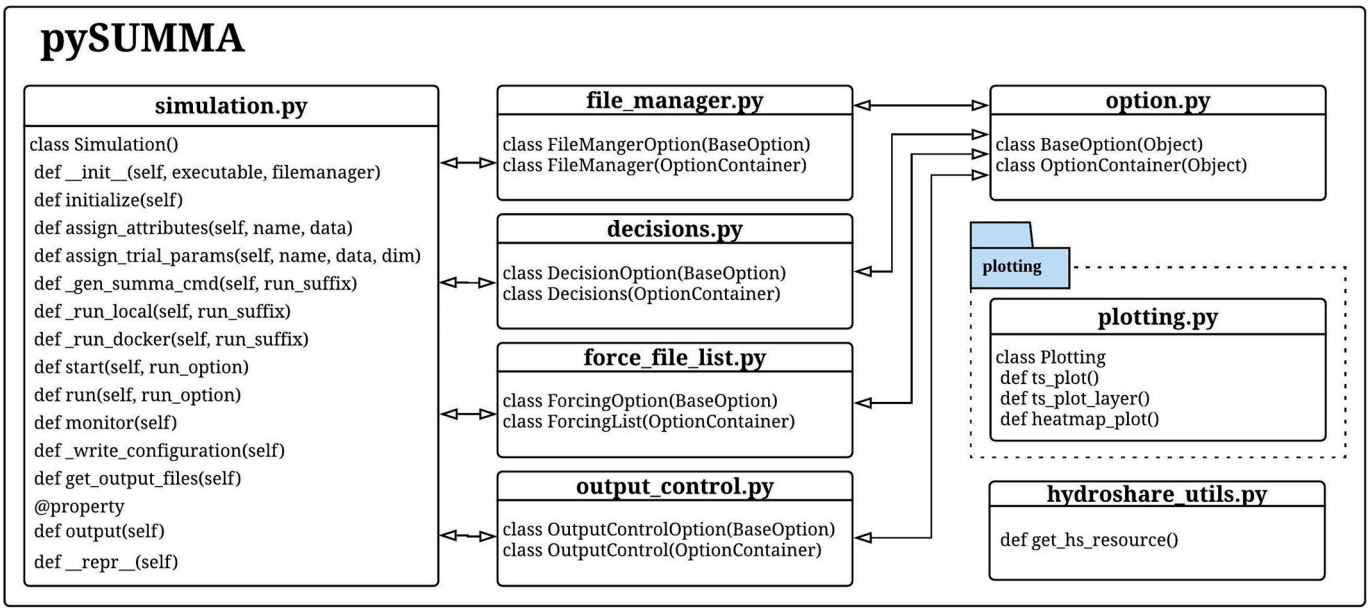

Fig. 4. pySUMMA library classes.

configuration files including the decision, forcing, parameter, and attribute files. A Decisions module (*decisions.py)* reads and manipulates a Decisions file which sets different physical process parameterizations. Through the *available_options* object in *decisions.py*, users can determine what options are available for model parameterizations and select model parameterizations from a list of options for each physical process (SUMMA Online Document, 2020). Four input configuration modules (*file_manager.py, decisions.py, force_file_list.py, and output_control.py*) have the same pattern of classes. For example, a File manager module (*file_manager.py)* is composed of *FileManagerOption* and *FileManager* classes and a Decisions module (*decisions.py)* is composed of *DecisionOption* and *Decision* classes. Each class is connected to an Option module (*option.py*) to avoid repetition of functions such as comparing, setting and writing each configuration file. After setting the SUMMA configuration, the simulation module (*Simulation.py)* is used for model execution. The *run ()* method of the *Simulation* class is used to execute the SUMMA model. This execution can be done in both "local" and "docker" computational environments. The environments are set using the *run_option* parameter for the *run()* method as discussed later in the Results and Discussion section.

Once a SUMMA model run has been completed, the plotting module (*plotting.py)* can be used to visualize the results. There are two different data output structures for SUMMA: 1) time, HRU (or GRU) number, and variable; 2) time, HRU (or GRU) number, soil (or snow) layer number, and variable. To visualize each of these output structures, the *Plotting class* consists of three methods: *ts_plot(), ts_plot_layer(),* and *heatmap_plot ()*. We used the seaborn library (statistical data visualization library) to create a 2D heat map with soil or snow layer and time as the axis for displaying a selected variable. Lastly, the model output module (*output_control.py*) is used to manipulate the output variables of SUMMA and the utilities module (*hydroshare_utils.py*) has functions to download test cases of SUMMA (model instance resources) and execution files (model program resources) from HydroShare.

3. Results and Discussion

In this section, we present a modeling case study application of the example implementation system described in Section 2.2. Then, we discuss how this approach addresses the challenge of achieving more reproducible studies summarized in the Introduction section by evaluating the approach against definitions, concepts, and metrics for reproducibility proposed by others. Lastly, we discuss the limitations of our approach that present opportunities for future research.

3.1. Case study description

Clark et al. (2015b) describe a set of thirteen modeling experiments exploring various hydrologic modeling scenarios using SUMMA. The study area for these modeling experiments is the Reynolds Mountain East Area (A = 32.7 km^2) in the Reynolds Creek Experimental Watershed in Idaho, USA (Fig. 5). In this paper, we focus on these modeling experiments as a case study with the goal of applying our approach so that each Clark et al. (2015b) experiment can be reconstructed and shared openly in a way that is easier to reproduce.

The first step toward this goal is the creation and organization of HydroShare resources to share all models and data files required for the analysis. The second step is to create Jupyter notebooks that describe the modeling experiments. These notebooks include text and equations to describe the modeling experiments while also including executable Python code using the pySUMMA API and inline visualizations that can be repeated and extended by others. We created seven Jupyter notebooks, each one documenting an experiment in the Clark et al. (2015b) study, and published them through HydroShare as a collection resource (Choi et al., 2020).

3.2. Model and data resources

Our first step in reproducing the Clark et al. (2015b) modeling experiments was to publish the specific SUMMA model version used in the analysis as a resource on HydroShare. To do this, we created a HydroShare resource using the Model Program resource type and uploaded the SUMMA 3.0.0 source code to the resource. We then published the resource through HydroShare so that it is persistent and immutable with a unique Digital Object Identifier (DOI) (Choi et al., 2020). Fig. 6 shows the landing page for this resource on HydroShare that includes detailed metadata describing: 1) the source code and compiled software engine, 2) metadata for the software, 3) a link showing the model was derived from a particular branch of a GitHub repository for SUMMA, and 4) a citation for referencing the resource. This same SUMMA 3.0.0 was also installed on the CUAHSI JH allowing users to execute the SUMMA model directly from CUAHSI JH.

We next created multiple resources in HydroShare to store the

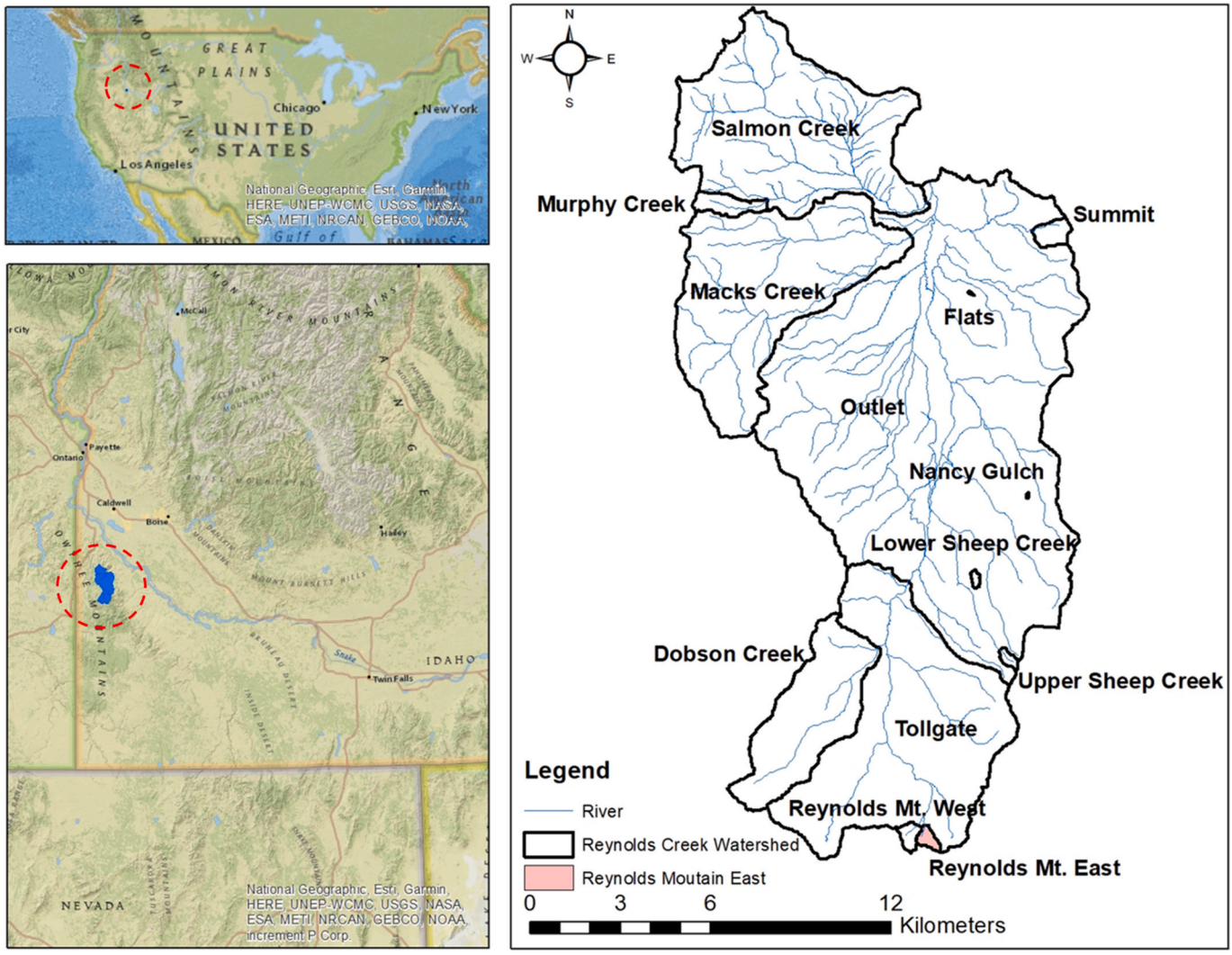

Fig. 5. Reynolds mountain east area in the Reynolds creek experimental watershed.

specific model inputs for each different SUMMA model experiment used in the Clark et al. (2015b) paper. There were four synthetic and nine field study test cases available as an online supplement to the Clark et al. (2015b) paper. From these data, we created seven unique model instance resources in HydroShare (Table 4) and grouped them into a collection resource (Choi et al., 2020). Each model instance resource includes: 1) input data for the SUMMA model, 2) a reference to the Clark et al. (2015b) paper, 3) a composite resource link that points to the Jupyter notebook used to execute the SUMMA model, and 4) a link to the model program resource used to execute the model instance.

Once this step is complete, the model and data resources required to reproduce the Clark et al. (2015b) experiments are publicly accessible in HydroShare with metadata to describe each resource and a unique URL to locate each resource. HydroShare also allows for publishing these resources in which case the resources become immutable and are assigned a Digital Object Identifier (DOI). This pattern can be adopted by other environmental modeling studies whereby both the model and data resources required to reproduce the study are uploaded into HydroShare, given metadata to describe each resource (including relationships between resources such as the "*ExecutedBy*" relationship between model program and model instance resources), and published with a DOI.

3.3. Demonstrating reproducibility

This section describes the steps that should be taken to reproduce one of the experiments described in Clark et al. (2015b). As a preparation step before starting a SUMMA simulation using Jupyter notebooks on CUAHSI JupyterHub, we recommend creating a pySUMMA conda virtual environment by running the steps described in the notebook "Creating_pySUMMA_conda_virtual_environment_in_CUAHSI_JupyterHub.ipynb" in the HydroShare composite resource for CUAHSI JH notebooks. Once this preparation step is completed, the basic algorithm to run a notebook is shown in Fig. 7. First, the pySUMMA *hydroshare_utils* module is used to download the model instance that will be used in the notebook directly from HydroShare. After downloading the SUMMA model instance, it is possible to create a pySUMMA simulation object using the *Simulation* class of pySUMMA and supplying SUMMA executable (summa.exe) and the *File Manager* file path. After creating the pySUMMA simulation object, the SUMMA model can be executed using the *run()* method, which takes a *run_option* argument as *local*. When CUAHSI JH was created by using Docker, SUMMA was automatically complied and the SUMMA executable was created in '*/usr/local/bin/summa.exe*'. Therefore, after setting the *executable* variable to the location of "summa.exe", users can set a *run_option* as *local*. By changing the *executable* variable as *"/usr/bin/summa.exe*", it is possible to execute the same notebook on CyberGIS JW.

As an example, we present here the results from running two

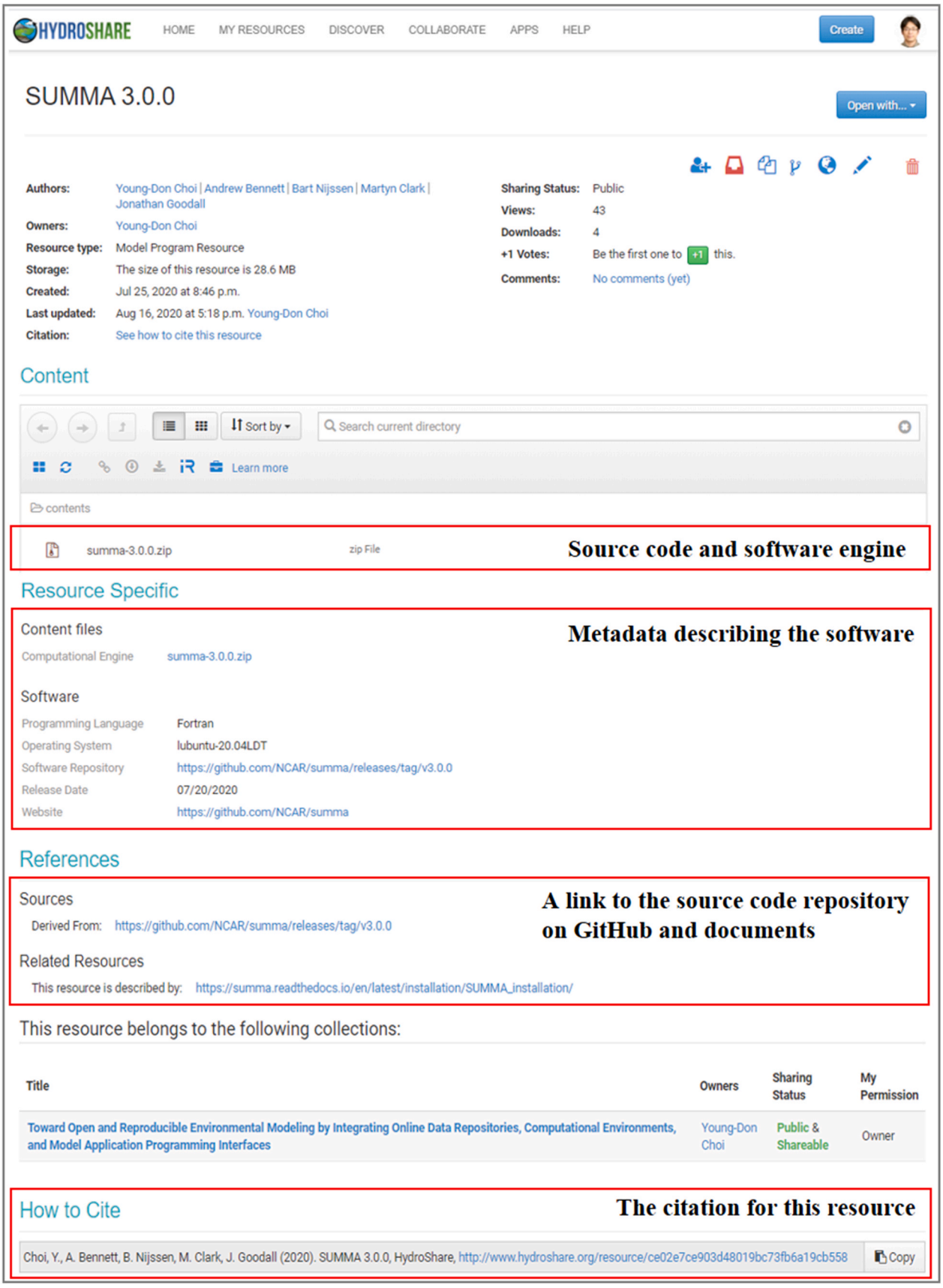

Fig. 6. The HydroShare landing page for a SUMMA model program resource used in the example analysis (Choi et al., 2020).

```
# Downloading the model instance required for the modeling scenario directly from HydroShare based on its unique Resource ID
from pysumma import hydroshare_utils
import os
resource_id = '13d6b84a9553410297a67fa366a56cb2'
instance = hydroshare_utils.get_hs_resource(resource_id, os.getcwd())

# Creating the pySUMMA simulation instance
import pysumma as ps
executable = "/usr/bin/summa.exe"
file_manager = os.path.join(os.getcwd(), instance, 'settings/summa_fileManager_riparianAspenSimpleResistance.txt')
S = ps.Simulation(executable, file_manager)

# Executing the SUMMA example model in the CUAHSI JupyterHub
S.run('local', run_suffix='_simpleResistance')
```

Fig. 7. The basic step for a SUMMA model run using Jupyter notebooks.

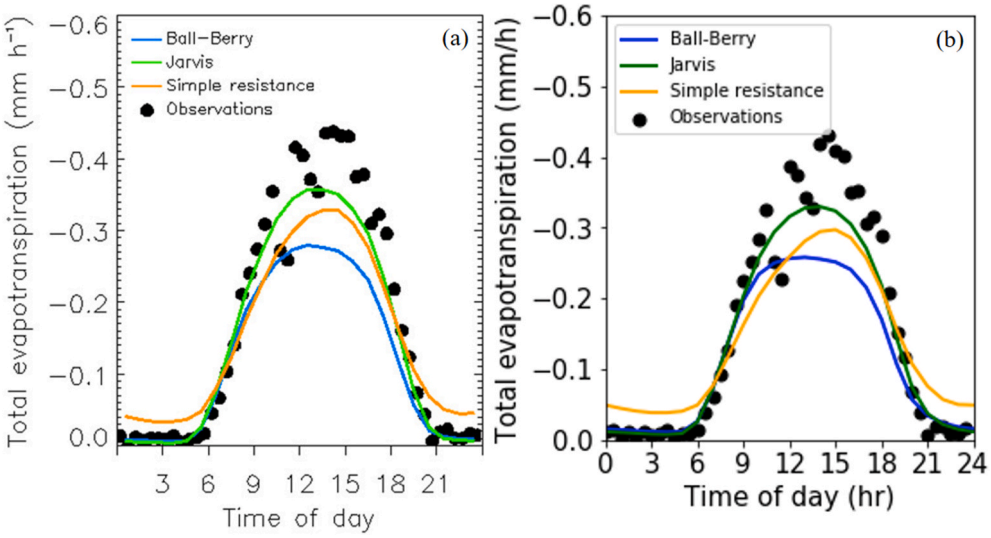

Fig. 8. Reproducibility of Fig. 7 from Clark et al. (2015b) showing the impact of the three different stomatal resistance parameterizations on total evapotranspiration (a) published result, (b) reproduced result.

different experiments included in the Clark et al. (2015b) paper. The first reproduces Fig. 7 from Clark et al. (2015b) and is published as a HydroShare resource with the title "The impact of Stomatal Resistance Parameterization on ET of SUMMA Model in Aspen stand at Reynolds Mountain East (Choi et al., 2020)." The second reproduces Fig. 8 (left) from Clark et al. (2015b) and is published as a HydroShare resource with the title "The impact of Root Distributions Parameters on ET of SUMMA Model in Aspen stand at Reynolds Mountain East."

Fig. 8 gives the results from the first experiment that explores the impact of three different stomatal resistance parameterizations on total evapotranspiration: Ball-Berry (Ball et al., 1987), Jarvis (1976), and the simple resistance method. Fig. 8a is the original result from the SUMMA paper (Clark et al., 2015b) and Fig. 8b is a reproduced figure resulting from applying this framework. These stomatal resistance parameterizations have different physical characteristics: both the Jarvis and Ball Berry methods consider photosynthesis, while the simple soil resistance method mainly considers the soil water conditions. The results show that the simple soil resistance method is higher than the other methods during the night hours. Comparing the two plots shows the complexity associated with reproducing past computational modeling studies. While the results are consistent, they are not exact. The precise reason for the differences in the model results is difficult to determine. We suspect it is due in part to upgrades to SUMMA or SUMMA dependencies between the versions of the SUMMA 2.0 used in the Clark et al. (2015b) paper and the SUMMA 3.0 used to create the newer plot. More vexing is that some of the observed data points appear to have shifted with no good explanation for why. One possible explanation could be the fact that different visualization tools were used to create each plot: Interactive Data Language (IDL) for the plot on the left and matplotlib for the plot on the right. We suspect differences like this would not be uncommon when trying to reproduce any past computational study given the difficulties in recreating the exact computational and analysis environment including data preparation routines, computational modeling software, and post-processing analysis and visualization tools. This, in fact, speaks to the difficulty of the problem and the need for innovation in cyberinfrastructure approaches that is at the heart of this study. This said, it is also important to stress that the goal of reproducibility may not be to obtain the *exact* same results, but rather *consistent* results that would produce the same conclusion. This is an idea expressed by high level reports on computational reproducibility (National Academies of Sciences, 2019) that we will discuss further in Section 3.4.

Fig. 9 shows the results from the second experiment from Clark et al. (2015b), which explores the impact of the root distribution parameters with different stomatal resistance parameterizations for total evapotranspiration. In this case, we reproduced the plot that shows the impact of root distribution parameters (Fig. 9b) and compared it to the previous result (Fig. 9a). Again, we see consistent (although not exact) results between the two model runs. Given that the modeling experiment is now implemented within the system, it is also possible to more easily extend and repurpose it for other purposes. To this point, we demonstrate reuse of past modeling studies by creating two additional plots for determining the effect of different root distribution (Fig. 9c) and stomatal resistance parameterizations (Fig. 9d) on total evapotranspiration. These plots show how higher root distribution exponents in the soil profile indicate that the roots are deeper in the soil, which makes it

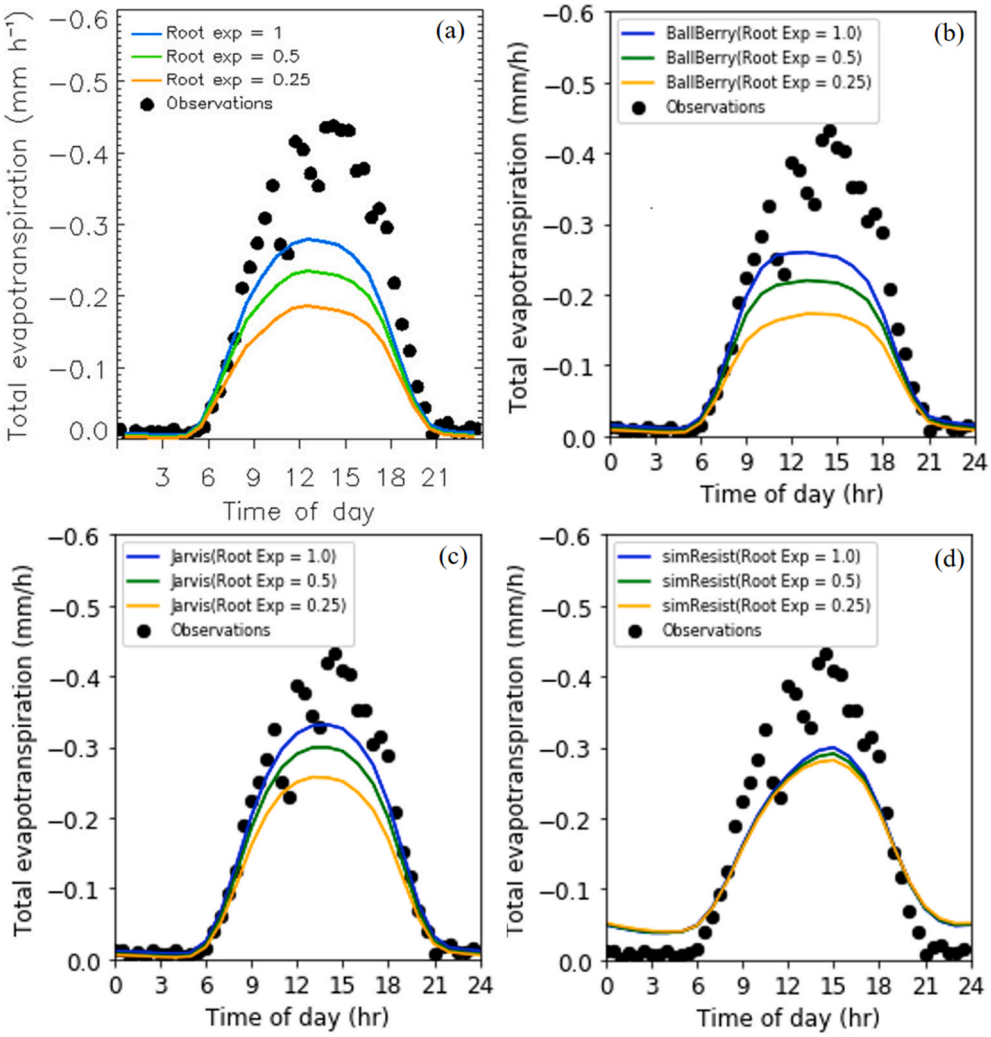

Fig. 9. Reproducibility and reusability of Fig. 8 (a) of Clark et al. (2015b) showing the impact of root distribution parameter with different stomatal resistance parameterization on total evapotranspiration (a) published output, (b) reproduced output, (c) and (d) output from reusability application extending the prior study.

Table 4
Mapping between the modeling experiments of Clark et al. (2015b) and Model Instance Resources on HydroShare used to store the input files for that model experiment.

Figures from Clark et al. (2015b)	Resource Name on HydroShare to Reproduce each Clark et al. (2015b) Figure
Fig. 1 (top)	The impact of the canopy shortwave radiation parameterizations of SUMMA Model in Aspen stand at Reynolds Mountain East
Fig. 1 (bottom)	The impact of LAI parameter on the below canopy shortwave radiation of SUMMA Model in Aspen stand at Reynolds Mountain East
Fig. 2	The impact of the canopy wind parameter for the exponential wind profile of SUMMA Model in Aspen stand at Reynolds Mountain East
Fig. 7	The impact of Stomatal Resistance Parameterization on ET of SUMMA Model in Aspen stand at Reynolds Mountain East
Fig. 8 (left)	The impact of Root Distributions Parameters on ET of SUMMA Model in Aspen stand at Reynolds Mountain East
Fig. 8 (right)	The impact of Lateral Flow Parameterizations on ET of SUMMA Model in Aspen stand at Reynolds Mountain East
Fig. 9	The impact of Lateral Flow Parameterizations on Runoff of SUMMA Model in Aspen stand at Reynolds Mountain East

easier for plants to extract soil water. As a result, during the higher evapotranspiration periods (10:00–17:00), the Jarvis method more closely matched the observation data. However, during the period when evapotranspiration is decreasing (17:00–20:00), the Ball-Berry method was more precise compared to the simple resistance method. Over the complete time period, the analysis shows that the Jarvis method had the best fit with observations.

3.4. Evaluating reproducibility

To evaluate if reproducibility was achieved, we considered definitions and concepts for evaluating reproducibility being put forward by others. For example, the National Academies of Science, Engineering, and Medicine (National Academies of Sciences, 2019) define reproducibility, focused on computational reproducibility, as "obtaining consistent results using the same input data; computational steps, methods, and code; and conditions of analysis." To guarantee reproducibility, the organization recommended delivering "clear, specific, and complete information about any computational methods and data products to repeat the previous study, and that information should include the data, methods, and computational environment." FAIR principles (Wilkinson et al., 2016) include 15 metrics that should be met as a minimum for reproducibility. These metrics are: a) Findable (4 metrics), b) Accessible (4 metrics), c) Interoperable (3 metrics), and d) Reusable (4 metrics).

In the hydrology and water resources fields, Hutton et al. (2016) recommended reproducible studies have: 1) readable and reusable code, 2) an unambiguous workflow, 3) a repository to easily find data and code with associated metadata, 4) use of unique persistent identifiers, 5) new procedures to reproduce large-scale studies using HPC. Additionally, Hut et al. (2017) suggested the use of containers and open interfaces to guarantee stronger reproducibility as a response to Hutton et al. (2016). Finally, Stagge et al. (2019) proposed a set of survey questions to assess the reproducibility of a journal article. The survey requires that eight elements be available for a study to be called reproducible: 1) directions to run or reproduce the study, 2) code/model/software files, 3) input data, 4) hardware/software requirements, 5) stated data persistence policy, 6) materials linked by unique and persistent identifiers, 7) metadata to describe the code, and 8) common file format and instructions to open these files.

With these criteria in mind, by simply using HydroShare as the data repository for all data and software used for the study, we can support many of the metrics associated with reproducibility. HydroShare supports FAIR principles (Tarboton et al., 2018) for each resource that includes model input, source code, metadata, and supplementary documents. Using JupyterHub as described in the paper provides a consistent computational environment and using Jupyter notebooks and containerized model execution environments provides a clear and easy workflow to assure users can reproduce a published study. Finally, using a model API makes it easier for a user to follow the logic and steps used to configure, run, and postprocess a modeling simulation, allowing for not only reproducibility but also reuse and extension of prior work. Therefore, if we compare these definitions and concepts for a validation of reproducibility to our approach and its example application, we can claim that it satisfies the criteria for reproducible computational modeling. Still, while the framework allows for satisfying the criteria, it is still up to the user to ensure care is taken with sharing and documenting resources with adequate metadata and instructions to achieve reproducibility.

3.5. Approach limitations and opportunities for future research

This research focuses on examples that assume model input files had already been processed and are available for use in the modeling analysis. The preprocessing steps required to generate model input files from raw geospatial and time series observational data are a necessary component of longer-term goals for creating so called "end-to-end" reproducible analysis workflows. For example, Slater et al. (2019) provided an "end-to-end" reproducible hydrology workflow using R for climate data retrieval, spatial analysis, modeling, statistical analysis, visualization, and data publishing. As another example of automated end-to-end workflows, HydroTerre (Leonard, 2015) includes: 1) data workflows (Leonard and Duffy, 2013) to create watershed models using Essential Terrestrial Variables (ETV), 2) data-model workflows to transform watershed data into model inputs, 3) model workflows (Leonard and Duffy, 2014) to execute models in HPC, especially The Penn State Integrated Hydrologic Modeling System (PIHM), and 4) visualization workflows to visualize the first three workflows to easily create and share model results for analysis.

Currently, pySUMMA has developed the functionalities of manipulating created model input, executing SUMMA, and plotting model output. To complete "end-to-end" workflows, data preprocessing is critical for improving reproducibility as the steps to create model input files are often nontrivial and require a significant time investment. Prior work to address this challenge includes the EcohydroLib Python library developed as a software framework for managing spatial data acquisition and preparation workflows for ecohydrology modeling (Miles and Band, 2015). EcohydroLib takes advantage of open source GRASS GIS libraries to automate data gathering and preparation for environmental models. It is a model agnostic approach for mapping a variety of data sources into input files required by environmental models. Alternative data processing workflows and pipelines such as HydroTerre (Leonard, 2015) could also be explored for bringing data preprocessing capabilities for environmental models into the general approach described through this work. However, just having new data processing pipelines alone will be insufficient. We also need more detailed modeling protocols and procedures to replicate (or even reproduce) a study (Ceola et al., 2015) because reproducibility is not just a technological problem, it is equally an educational problem (Grüning et al., 2018).

Post-processing for visualization and model analysis procedures is also essential to creating a powerful modeling environment, saving time when analyzing model output and strengthening reproducibility. To grow use of model APIs, many analysis methods will be necessary such as plotting, calibration, optimization, and uncertainty analysis. While pySUMMA is still being developed toward these goals, other model APIs discussed in this paper and that could be used within the example modeling system do have more robust processing capabilities already. One question that remains is the extent to which environmental model APIs can reuse underlying software to support common model post-processing routines. General libraries in Python, such as Pandas and matplotlib, are universally applicable to environmental modeling post-processing tasks. However, is a plotting or data analysis library more tailored for environmental modeling but still sufficiently general to serve many environmental models possible? If so, it could further reduce the duplication of code across environmental model APIs and, ultimately, encourage more environmental model APIs that are robust, easier to maintain, and feature rich.

The ability to include data pre- and post-processing within the framework would be an important step for moving from reproducibility to replicability within the framework. Replicability is defined by the National Academies of Science, Engineering, and Medicine (National Academies of Sciences, 2019) as "obtaining consistent results across studies aimed at answering the same scientific question, each of which has obtained its own data." Replication, therefore, can be thought of as a next step beyond reproducibility where a study is repeated using new data, potentially from a new site or different time period, but similar methods. This work has focused on a general approach to support reproducibility of computational models. The framework could be extended for replication by extending a model API, like the pySUMMA API described in this paper, to include not only functions for model configuration (e.g., settings and parameter values assuming model input files have already been generated), but also for model preprocessing where input files for the model are generated from raw data sources.

4. Conclusion

Computational irreproducibility is an important problem in many scientific fields. Recent research to improve computational reproducibility has focused on advancing the sharing of data used in studies, using computational notebooks and containers for encapsulating complete computational environments, and developing model APIs for programmatically interacting with simulation models. A contribution of this research is to present a general approach to integrate these three areas of past work into a general approach for supporting more open and reproducible environmental modeling. We present an example implementation of this approach by leveraging: 1) HydroShare as a data sharing repository, 2) JupyterHub as a notebook-based, containerized, and cloud-based computational environment, and 3) pySUMMA as an example model API able to abstract lower-level details for model configuration, execution, and visualization from end users.

Using the example implementation, we demonstrate how modeling analyses can be completed in a more open and reproducible way. Building from a prior study presenting a series of modeling experiments applying SUMMA at the Reynolds Mountain East Area in the Reynolds Creek Experimental Watershed in Idaho, USA (Clark et al., 2015b), we first created and organized HydroShare resources to share data and model files. Next, we created Jupyter notebooks that leveraged the

pySUMMA API, introduced in this paper, to reproduce and extend figures from the prior study. Each notebook a) pulled required data from HydroShare into the computational environment, b) provided a notebook using text, equations, code, and inline visualizations for documenting the experiment, and c) allowed for online execution of the notebook and sharing of modifications to the notebook through HydroShare. Finally, we discussed how we evaluated that reproducibility was achieved and future steps that could be taken to further improve the proposed framework.

From this research, we conclude that cyberinfrastructure is reaching a point where it is possible to build open and transparent environmental modeling systems. Online repositories are sufficiently mature where they can be relied upon for storing key data and software resources for studies. Computational environments able to execute containerized environmental models can be interlinked with data repositories, and the ability for these computational environments to serve as gateways to High Performance Computing (HPC) resources is improving. More models are being provided with APIs that allow for programmatic control of the model configuration, execution, and visualization. Jupyter notebooks provide an important orchestration and documentation glue across these components where users can leverage APIs to access and publish data from online repositories, submit jobs to HPC resources, and programmatically interact with state-of-the-art environmental models. Linking these capabilities in a way that can be built upon and expanded as new models become available, as demonstrated in this paper, will move environmental modeling in a direction where open, transparent, reproducible, reusable, and replicable studies become the rule rather than the exception.

Software and data availability

All software and data used in this study were published with persistent DOIs on HydroShare. A collection resource in HydroShare (Choi et al., 2020) contains each of these resources. In addition to these resources published through HydroShare, the pySUMMA source code created though this study is available on GitHub as detailed below.

Product Title: pySUMMA v3.0.0
Lead Developers: Young-Don Choi and Andrew Bennett
Contact Email: yc5ef@virginia.edu, andrbenn@uw.edu
Tested Platform:
- HydroShare CUAHSI JupyterHub
- CyberGIS-Jupyter for Water
Software Required: Python 3.5 or above
Availability: The pySUMMA source code is publicly available through GitHub
- https://github.com/UW-Hydro/pysumma/releases/tag/3.0.0
License: BSD 3-Clause License

List of relevant URLs

CUAHSI JupyterHub: https://jupyterhub.cuahsi.org/
CUAHSI JupyterHub Legacy Environment: https://jupyter.cuahsi.org
CUAHSI JupyterHub GitHub: https://github.com/hydroshare/hydroshare-jupyterhub
CyberGIS-Jupyter (beta): https://hsjupyter.cigi.illinois.edu:8000
CyberGIS-Jupyter for Water: https://go.illinois.edu//cybergis-jupyter-water
DataOne: https://www.dataone.org
Docker: https://www.docker.com
DockerHub: https://hub.docker.com
EcohydroLib: https://github.com/selimnairb/EcohydroLib
Facebook API: https://developers.facebook.com/docs/apis-and-sdks
FigShare: https://figshare.com
Google API: https://developers.google.com/apis-explorer
Harvard Dataverse: https://dataverse.harvard.edu
HydroShare REST API: https://github.com/hydroshare/hydroshare/wiki/HydroShare-REST-API
NetCDF4 GitHub: https://github.com/Unidata/netcdf4-python
Numpy: https://www.numpy.org
Pandas: https://pandas.pydata.org
Seaborn: https://seaborn.pydata.org
Singularity: https://sylabs.io
SUMMA on the UCAR: https://ral.ucar.edu/projects/summa
xarray: http://xarray.pydata.org
XSEDE: https://www.xsede.org

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was supported by the National Science Foundation under collaborative grants OAC-1664061, OAC-1664018, OAC-1664119, ICER-1928369, and ICER-1928315. We acknowledge the work of the larger HydroShare team (https://help.hydroshare.org/about-hydroshare/team) that made this research possible.

References

Baker, M., 2016. 1,500 scientists lift the lid on reproducibility. Nature 533, 452–454. https://doi.org/10.1038/533452a.
Bakker, M., Post, V., Langevin, C.D., Hughes, J.D., White, J.T., Starn, J.J., Fienen, M.N., 2016. Scripting MODFLOW model development using Python and FloPy. Groundwater 54, 733–739. https://doi.org/10.1111/gwat.12413.
Ball, J.T., Woodrow, I.E., Berry, J.A., 1987. A model predicting stomatal conductance and its contribution to the control of photosynthesis under different environmental conditions. In: Progress in Photosynthesis Research. Springer Netherlands, Dordrecht, pp. 221–224. https://doi.org/10.1007/978-94-017-0519-6_48.
Bandaragoda, C., Castronova, A., Istanbulluoglu, E., Strauch, R., Nudurupati, S.S., Phuong, J., Adams, J.M., Gasparini, N.M., Barnhart, K., Hutton, E.W.H., Hobley, D.E. J., Lyons, N.J., Tucker, G.E., Tarboton, D.G., Idaszak, R., Wang, S., 2019. Enabling collaborative numerical modeling in earth sciences using knowledge infrastructure. Environ. Model. Software 120. https://doi.org/10.1016/j.envsoft.2019.03.020.
Bast, R., 2019. A FAIRer future. Nat. Phys. https://doi.org/10.1038/s41567-019-0624-3.
Boettiger, C., 2015. An introduction to Docker for reproducible research. ACM SIGOPS - Oper. Syst. Rev. 49, 71–79. https://doi.org/10.1145/2723872.2723882.
Brinckman, A., Chard, K., Gaffney, N., Hategan, M., Jones, M.B., Kowalik, K., Kulasekaran, S., Ludäscher, B., Mecum, B.D., Nabrzyski, J., Stodden, V., Taylor, I.J., Turk, M.J., Turner, K., 2019. Computing environments for reproducibility: capturing the "whole tale. Future Generat. Comput. Syst. https://doi.org/10.1016/j.future.2017.12.029.
Brooks, G., 2013. Benefits of APIs [WWW document]. Digital.gov. https://digital.gov/2013/03/12/benefits-of-apis/. accessed 1.14.20.
Jupyter Project, Bussonnier, M., Forde, J., Freeman, J., Granger, B., Head, T., Holdgraf, C., Kelley, K., Nalvarte, G., Osheroff, A., Pacer, M., Panda, Y., Perez, F., Ragan-Kelley, B., Willing, C., 2018. Binder 2.0 - reproducible, interactive, sharable environments for science at scale. Proc. 17th Python Sci. Conf. https://doi.org/10.25080/majora-4af1f417-011.
Buytaert, W., 2011. topmodel: implementation of the hydrological model TOPMODEL in R. Global Change Biol. https://doi.org/10.1111/j.1365-2486.2006.01305.x.
Castronova, A., Seul P.D, M., 2018. A general approach for cloud-based hydrologic modeling using jupyter notebooks [WWW document]. HydroShare. http://www.hydroshare.org/resource/075664b0f0df4c58892cb4665e77e497.
Ceola, S., Arheimer, B., Baratti, E., Blöschl, G., Capell, R., Castellarin, A., Freer, J., Han, D., Hrachowitz, M., Hundecha, Y., Hutton, C., Lindström, G., Montanari, A., Nijzink, R., Parajka, J., Toth, E., Viglione, A., Wagener, T., 2015. Virtual laboratories: new opportunities for collaborative water science. Hydrol. Earth Syst. Sci. https://doi.org/10.5194/hess-19-2101-2015.
Choi, Y., Goodall, J., Sadler, J., Castronova, A.M., Bennett, A., Li, Z., Nijssen, B., Wang, S., Clark, M., Tarboton, D., 2020. Toward open and reproducible environmental modeling by integrating online data repositories, computational environments, and model application programming interfaces [WWW document]. HydroShare. https://www.hydroshare.org/resource/33cfb9622a354442b2b0a869b15ea6b0/.
Clark, M.P., Nijssen, B., Lundquist, J.D., Kavetski, D., Rupp, D.E., Woods, R.A., Freer, J. E., Gutmann, E.D., Wood, A.W., Brekke, L.D., Arnold, J.R., Gochis, D.J., Rasmussen, R.M., 2015a. A unified approach for process-based hydrologic modeling: 1. Modeling concept. Water Resour. Res. https://doi.org/10.1002/2015WR017198.

Clark, M.P., Nijssen, B., Lundquist, J.D., Kavetski, D., Rupp, D.E., Woods, R.A., Freer, J. E., Gutmann, E.D., Wood, A.W., Gochis, D.J., Rasmussen, R.M., Tarboton, D.G., Mahat, V., Flerchinger, G.N., Marks, D.G., 2015b. A unified approach for process-based hydrologic modeling: 2. Model implementation and case studies. Water Resour. Res. https://doi.org/10.1002/2015WR017200.

Essawy, B.T., Goodall, J.L., Zell, W., Voce, D., Morsy, M.M., Sadler, J., Yuan, Z., Malik, T., 2018. Integrating scientific cyberinfrastructures to improve reproducibility in computational hydrology: example for HydroShare and GeoTrust. Environ. Model. Software 105, 217–229. https://doi.org/10.1016/j.envsoft.2018.03.025.

Eynard-Bontemps, G., Abernathey, R., Hamman, J., Ponte, A., Rath, W., 2019. The PANGEO big data ecosystem and its use at CNES. In: Proc. Of the 2019 Conference on Big Data from Space (BiDS?2019). EUR 29660 EN. ARRAY(0xc86ce6c), Luxembourg, pp. 49–52. https://doi.org/10.2760/848593.

Fuka, D.R., Walter, M.T., Macalister, C., Steenhuis, T.S., Easton, Z.M., 2014. SWATmodel: a multi-operating system, multi-platform SWAT model package in R1. J. Am. Water Resour. Assoc. 50 https://doi.org/10.1111/jawr.12170.

Gil, Y., David, C.H., Demir, I., Essawy, B.T., Fulweiler, R.W., Goodall, J.L., Karlstrom, L., Lee, H., Mills, H.J., Oh, J.-H., Pierce, S.A., Pope, A., Tzeng, M.W., Villamizar, S.R., Yu, X., 2016. Toward the Geoscience Paper of the Future: best practices for documenting and sharing research from data to software to provenance. Earth Sp. Sci. 3, 388–415. https://doi.org/10.1002/2015EA000136.

Grüning, B., Chilton, J., Köster, J., Dale, R., Soranzo, N., van den Beek, M., Goecks, J., Backofen, R., Nekrutenko, A., Taylor, J., 2018. Practical computational reproducibility in the life sciences. Cell Syst. https://doi.org/10.1016/j.cels.2018.03.014.

Hobley, D.E.J., Adams, J.M., Siddhartha Nudurupati, S., Hutton, E.W.H., Gasparini, N. M., Istanbulluoglu, E., Tucker, G.E., 2017. Creative computing with Landlab: an open-source toolkit for building, coupling, and exploring two-dimensional numerical models of Earth-surface dynamics. Earth Surf. Dyn. 5, 21–46. https://doi.org/10.5194/esurf-5-21-2017.

Horsburgh, J.S., Morsy, M.M., Castronova, A.M., Goodall, J.L., Gan, T., Yi, H., Stealey, M. J., Tarboton, D.G., 2016. HydroShare: sharing diverse environmental data types and models as social objects with application to the hydrology domain. J. Am. Water Resour. Assoc. 52 https://doi.org/10.1111/1752-1688.12363.

Hut, R.W., van de Giesen, N.C., Drost, N., 2017. Comment on “Most computational hydrology is not reproducible, so is it really science?” by Christopher Hutton et al.: let hydrologists learn the latest computer science by working with Research Software Engineers (RSEs) and not reinvent the waterwheel ourselves. Water Resour. Res. https://doi.org/10.1002/2017WR020665.

Hutton, C., Wagener, T., Freer, J., Han, D., Duffy, C., Arheimer, B., 2016. Most computational hydrology is not reproducible, so is it really science? Water Resour. Res. 52, 7548–7555. https://doi.org/10.1002/2016WR019285.

Ince, D.C., Hatton, L., Graham-Cumming, J., 2012. The case for open computer programs. Nature 482, 485–488. https://doi.org/10.1038/nature10836.

Jarvis, P., 1976. The interpretation of the variations in leaf water potential and stomatal conductance found in canopies in the field. Trans. R. Soc. B 273 (927), 593–610. https://doi.org/10.1098/rstb.1976.0035.

Kluyver, T., Ragan-kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., Willing, C., 2016. Jupyter Notebooks—a publishing format for reproducible computational workflows. Position. Power Acad. Publ. https://doi.org/10.3233/978-1-61499-649-1-87.

Kouwen, N., Soulis, E.D., Pietroniro, A., Donald, J., Harrington, R.A., 1993. Grouped response units for distributed hydrologic modeling. J. Water Resour. Plann. Manag. 119, 289–305. https://doi.org/10.1061/(ASCE)0733-9496(1993)119:3(289).

Kurtzer, G.M., Sochat, V., Bauer, M.W., 2017. Singularity: scientific containers for mobility of compute. PloS One 12, e0177459. https://doi.org/10.1371/journal.pone.0177459.

Lampert, D.J., Wu, M., 2015. Development of an open-source software package for watershed modeling with the Hydrological Simulation Program in Fortran. Environ. Model. Software 68, 166–174. https://doi.org/10.1016/J.ENVSOFT.2015.02.018.

Leonard, L.N., 2015. HydroTerre: towards an expert system for scaling hydrological data and models from hill-slopes to major-river basins. ProQuest Diss. Theses Glob.

Leonard, L., Duffy, C.J., 2013. Essential terrestrial variable data workflows for distributed water resources modeling. Environ. Model. Software. https://doi.org/10.1016/j.envsoft.2013.09.003.

Leonard, L., Duffy, C.J., 2014. Automating data-model workflows at a level 12 HUC scale: watershed modeling in a distributed computing environment. Environ. Model. Software. https://doi.org/10.1016/j.envsoft.2014.07.015.

Leonard, L., Duffy, C., 2016. Visualization workflows for level-12 HUC scales: towards an expert system for watershed analysis in a distributed computing environment. Environ. Model. Software. https://doi.org/10.1016/j.envsoft.2016.01.001.

Leonard, L., Miles, B., Heidari, B., Lin, L., Castronova, A.M., Minsker, B., Lee, J., Scaife, C., Band, L.E., 2019. Development of a participatory Green Infrastructure design, visualization and evaluation system in a cloud supported jupyter notebook computing environment. Environ. Model. Software. https://doi.org/10.1016/j.envsoft.2018.10.003.

Lyu, F., Yin, D., Padmanabhan, A., Choi, Y., Goodall, J.L., Castronova, A., Tarboton, D., Wang, S., 2019. Reproducible hydrological modeling with CyberGIS-jupyter: a case study on summa. In: Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (Learning). PEARC ’19. ACM, New York, NY, USA. https://doi.org/10.1145/3332186.3333052, 21:1–21:6.

Markham, K., 2019. Six easy ways to run your Jupyter Notebook in the cloud Retrieved from [WWW Document]. https://www.dataschool.io/cloud-services-for-jupyter-notebook/ accessed 12.10.19.

McDonnell, B.E., 2017. Pyswmm 0.4. 5: Python wrapper for SWMM5 API. https://github.com/OpenWaterAnalytics/pyswmm accessed 12.10.19.

McNutt, M., 2014. Reproducibility. Science (80-.). https://doi.org/10.1126/science.1250475.

Merkel, D., 2014. Docker: lightweight Linux containers for consistent development and deployment [WWW Document]. Linux J. https://www.linuxjournal.com/content/docker-lightweight-linux-containers-consistent-development-and-deployment accessed 1.21.20.

Miles, B., Band, L.E., 2015. Ecohydrology Models without Borders? https://doi.org/10.1007/978-3-319-15994-2_31.

Morsy, M.M., Goodall, J.L., Castronova, A.M., Dash, P., Merwade, V., Sadler, J.M., Rajib, M.A., Horsburgh, J.S., Tarboton, D.G., 2017. Design of a metadata framework for environmental models with an example hydrologic application in HydroShare. Environ. Model. Software 93, 13–28. https://doi.org/10.1016/J.ENVSOFT.2017.02.028.

National Academies of Sciences, 2019. Reproducibility and Replicability in Science. https://doi.org/10.17226/25303.

Reddy, M., 2011. API Design for C++ https://doi.org/10.1016/C2010-0-65832-9.

Signell, R.P., Pothina, D., 2019. Analysis and visualization of coastal ocean model data in the cloud. J. Mar. Sci. Eng. https://doi.org/10.3390/jmse7040110.

Slater, L.J., Thirel, G., Harrigan, S., Delaigue, O., Hurley, A., Khouakhi, A., Prosdocimi, I., Vitolo, C., Smith, K., 2019. Using R in hydrology: a review of recent developments and future directions. Hydrol. Earth Syst. Sci. https://doi.org/10.5194/hess-23-2939-2019.

Stagge, J.H., Rosenberg, D.E., Abdallah, A.M., Akbar, H., Attallah, N.A., James, R., 2019. Assessing data availability and research reproducibility in hydrology and water resources. Sci. Data 6. https://doi.org/10.1038/sdata.2019.30.

Stodden, V., Miguez, S., 2013. Best practices for computational science: software infrastructure and environments for reproducible and extensible research. SSRN Electron. J. https://doi.org/10.2139/ssrn.2322276.

SUMMA Online Document, 2020. SUMMA online document. https://summa.readthedocs.io/en/latest/ accessed 2.25.20.

Tague, C.L., Band, L.E., Tague, C.L., Band, L.E., 2004. RHESSys: regional hydro-ecologic simulation system—an object-oriented approach to spatially distributed modeling of carbon, water, and nutrient cycling. Earth Interact. 8, 1–42. https://doi.org/10.1175/1087-3562(2004)8<1:RRHSSO>2.0.CO;2.

Tarboton, D.G., 1997. A new method for the determination of flow directions and upslope areas in grid digital elevation models. Water Resour. Res. 33, 309–319. https://doi.org/10.1029/96WR03137.

Tarboton, D.G., Idaszak, R., Horsburgh, J.S., Heard, J., Ames, D., Goodall, J.L., Band, L.L. E., Merwade, V., Couch, A., Arrigo, J., Hooper, R., Valentine, D., Maidment, D.R., Goodball, J.L., Merwade, V., Couch, A., Arrigo, J., Hooper, R., Valentine, D., Maidment, D.R., Goodall, J.L., Band, L.L.E., Merwade, V., Couch, A., Arrigo, J., Hooper, R., Valentine, D., Maidment, D.R., 2014. Hydro share: advancing collaboration through hydrologic data and model sharing. In: Proc. - 7th Int. Congr. Environ. Model. Softw. Bold Visions Environ. Model. iEMSs 2014, vol. 1, pp. 23–29. https://doi.org/10.13140/2.1.4431.6801.

Tarboton, David, Idaszak, Ray, Horsburgh, J., 2018. HydroShare Tools and Recommended Practices for Sharing and Publishing Data and Models in Support of Collaborative Reproducible Research. AGU 2018 Fall Meet. https://doi.org/10.1002/essoar.10500174.1.

Viglione, A., Parajka, J., 2020. TUWmodel. R Packag. Doc. https://doi.org/10.1002/hyp.6253.

Volk, J.M., Turner, M.A., 2019. PRMS-Python: a Python framework for programmatic PRMS modeling and access to its data structures. Environ. Model. Software. https://doi.org/10.1016/j.envsoft.2019.01.006.

Wilkinson, M.D., Dumontier, M., Aalbersberg, IjJ., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J.W., da Silva Santos, L.B., Bourne, P.E., Bouwman, J., Brookes, A.J., Clark, T., Crosas, M., Dillo, I., Dumon, O., Edmunds, S., Evelo, C.T., Finkers, R., Gonzalez-Beltran, A., Gray, A.J.G., Groth, P., Goble, C., Grethe, J.S., Heringa, J., t Hoen, P.A.C., Hooft, R., Kuhn, T., Kok, R., Kok, J., Lusher, S.J., Martone, M.E., Mons, A., Packer, A.L., Persson, B., Rocca-Serra, P., Roos, M., van Schaik, R., Sansone, S.A., Schultes, E., Sengstag, T., Slater, T., Strawn, G., Swertz, M. A., Thompson, M., Van Der Lei, J., Van Mulligen, E., Velterop, J., Waagmeester, A., Wittenburg, P., Wolstencroft, K., Zhao, J., Mons, B., 2016. Comment: the FAIR Guiding Principles for scientific data management and stewardship. Sci. Data. https://doi.org/10.1038/sdata.2016.18.

Yin, D., Liu, Y., Padmanabhan, A., Terstriep, J., Rush, J., Wang, S., 2017. A cybergis-jupyter framework for geospatial analytics at scale. In: ACM International Conference Proceeding Series. Association for Computing Machinery. https://doi.org/10.1145/3093338.3093378.

Yuan, Z., Ton That, D.H., Kothari, S., Fils, G., Malik, T., 2018. Utilizing provenance in reusable research objects. Informatics. https://doi.org/10.3390/informatics5010014.