Environmental Modelling and Software 132 (2020) 104797

Contents lists available at ScienceDirect

Environmental Modelling and Software

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

Towards fast prototyping of cloud-based environmental decision support systems for environmental scientists using R Shiny and Docker

Yu Li

Dept. Of Civil, Environmental, and Geomatic Engineering, Institute of Environmental Engineering, ETH Zurich, Zurich, Switzerland

ARTICLE INFO

Keywords:
Web-based EDSS
Free and open-source software (FOSS)
R shiny
Docker
Groundwater model
Crop model

ABSTRACT

Environmental decision support systems (EDSS) have drawn an increasing attention among scientists to tackle with the growing complexity of environmental problems and to support policy makers. Yet, many EDSS reported in literature are case specific, fragmented in development strategies and under-reporting server setup, thus impeding EDSS development and knowledge sharing among the scientific community. In this work we introduce an EDSS development framework mainly based on R language, which is popular among environmental scientists, and Docker related software to lower technical hurdles for deployment in a web-based context. Using two examples, we demonstrate that the framework is able to deliver a unified and cost effective solution for setting up prototypes of modern web-based EDSS without compromising usability. A public repository is created to promote access to more examples from literature, which users can adapt for their own studies.

Software availability

Product Name: Open source development framework for EDSS.
Developer: Dr. Yu Li.
Software Required: Docker, R.
License: MIT License.
Availability: https://github.com/cocomice/EDSS_dep_framework.

1. Introduction

Decision support systems (DSS) usually refer to computer aided programs that help decision makers to solve unstructured or semi-structured information using data and models (Morton, 1971). They are widely used in different fields nowadays. Among them, environmental decision support systems (EDSS) specialize in tackling environmental problems, and have attracted a growing attention from researchers (e.g., Matthies et al., 2007; Hadded et al., 2013; McDonald et al., 2019; Whateley et al., 2015; Shao et al., 2017). The popularity of EDSS, on one hand, is due to the increasing difficulties in solving environmental problems, which are often entangled with human society, turning them into complex coupled human-nature systems (Liu et al., 2007). The complexity is further amplified by global change which imposes extra uncertainties concerning the future state of the world (Pachauri et al., 2014; Walker et al., 2013; Milly et al., 2008). On the other hand, the improvement of computer and monitoring technologies have enabled researchers to produce/access vast amounts of information and to develop more advanced analytic tools. However, those trends not only increase the potential of EDSS, but also demand modern EDSS implementation to be more democratic, user-friendly and flexible (Mir and Quadri, 2009; Loucks, 1995; Hewitt and Macleod, 2017; Zulkafli et al., 2017). The democratisation implies the participation of stakeholders in data collection, development of EDSS as well as their final operation. The user-friendliness aims to lower technical barriers so that policy makers lacking specific knowledge can still apply those complex analytic tools for their decision making. The flexibility requires EDSS to be able to include new information/functionalities with ease but also to allow integration with legacy models, which were built with state-of-the-art science but not designed for the Internet context (Kumar et al., 2015). Web-based applications have become a popular solution that overcomes many of the challenges such as accessibility: Users can easily visit applications on their computers or smart devices (Swain et al., 2015). The benefit can be further leveraged via cloud computing to remove computation limits, and to provide on-demand load balancing in the face of a high numbers of simultaneous users.

Yet, most EDSS reported in literature are case specific. They are implemented in different programming languages and the programs sometimes are closed-source, making them difficult to be transferred/adapted to other study regions. This partly explains why many case studies end up with their EDSS being only experimental, small-scale and underused in practice (Matthies et al., 2007). Moreover, the

E-mail address: yu.li@ifu.baug.ethz.ch.

https://doi.org/10.1016/j.envsoft.2020.104797
Received 12 April 2020; Received in revised form 28 June 2020; Accepted 9 July 2020
Available online 24 July 2020
1364-8152/© 2020 The Author. Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

development of a web-based EDSS is not limited to the software implementation, but requires a number of indispensable steps to make the entire system operational in a web context, including testing, deployment on servers, and maintenance over time, etc. (Murugesan et al., 2001), which are often under-reported in literature. Regardless, they represent additional, if not significant, burdens on developers whose primary background is environmental modelling. To address all those steps usually requires specific knowledge of web-engineering, or outsourcing to third party services with non-trivial cost, usually unaffordable for small research projects. Last but not least, development of EDSS requires active engagement between decision makers and developers, which is a recurring theme during any project (Loucks, 1995; Hewitt and Macleod, 2017; Zulkafli et al., 2017). This requirement becomes even more prominent if a large number of stakeholders is involved, while lack of sufficient participation leads to poor acceptance of the EDSS by end-users (Volk et al., 2010). Being able to prototyping EDSS that assemble the user interface (UI) and the core functions will allow the participation of stakeholders at an early stage, so they can evaluate the systems and provide feedback from their experience (Smith, 1991). As a result, it increases the likelihood of successful use of developed EDSS. Therefore, having an effective development framework for fast prototyping EDSS, or even providing a production-ready high--quality web-based application, will be extremely valuable, particularly for research teams with limited resources, allowing them to engage less in technical exercises and spend their efforts in more crucial research tasks. Such a development framework, in particular, should favor free and open-source software (FOSS) to promote knowledge-sharing and adaptation, cover complete stages in developing web-based EDSS with minimal interventions, and, most importantly, be capable of producing state-of-the-art EDSS as seen in literature. Surprisingly, few studies reported that kind of frameworks. In a recent paper by Swain et al. (2016) the authors presented an open-source software called Tethys Platform for developing and hosting web-based models for environmental researchers. The software, implemented in Python language (Van Rossum et al., 2007), aims to lower technical barriers for researchers in shipping environmental web-apps into operational context and has sparked several applications. With new tools and software infrastructure being continuously introduced in recent years, new opportunities have also emerged for further reducing the aforementioned technical hurdles for environmental researchers who may be acquainted with programming languages other than Python.

In this paper, we aim to contribute the toolkits for developing EDSS by proposing a framework based on the R Shiny package, Docker and other open-source software. The contribution to the EDSS community is twofold: 1) We introduce a development framework that can quickly produce prototypes of modern EDSS powered by web accessibility, interactive visualization, and cloud computation; 2) since the EDSS developed with the framework are portable and can be easily reproduced on major computer platforms, we further set up a public repository collecting examples from literature to promote knowledge sharing and to encourage the adaptation of existing EDSS. Note that the framework is designed for researchers with modest experience in programming models in R or similar languages, as well as in using command line tools.

The following sections are organized as follows: Section 2 describes the detail features of the related software and the proposed development framework. Section 3 demonstrates the potential of the proposed framework in implementing modern web-based EDSS by showcasing two EDSS apps for groundwater management and crop planning, respectively. The final concluding remarks are given in Section 4.

2. Software components and development framework

2.1. R shiny

Since its birth the programming language R has quickly become one of the top open-source languages in the scientific computation community (Rank 5 according to IEEE Spectrum 2019). The advantage of using R in environmental studies has been well discussed in literature (e.g., Slater et al., 2019, Andrews et al,. 2011), and the most cited features are its ease of use, support of vast libraries, independence of operating system as well as detailed software documentation. Originally designed for statistical analysis, R is now widely applied in various fields of research and is particularly popular among environmental scientists (e.g., Lai et al., 2019, Tippmann, 2015).

Among those packages, Shiny (Chang et al., 2017) is the one providing a framework for building interactive web applications. Using that package researchers can easily turn their R-based models into EDSS

Table 1
A list of useful R packages during the Shiny app implementation. They are categorized based on the main functionalities commonly seen in modern EDSS.

Category	Package name	Description of the functions	Reference
Shiny related	**Shiny**	main package for building Shiny apps	Chang et al. (2017)
	shinydashboard	a variant of **shiny** package for implementing a dashboard layout	Chang and Borges Ribeiro (2018)
	Shinytheme	a variant of **shiny** package for modifying default UI theme of Shiny apps	Chang (2018)
	Shinyjs	package for implementing JavaScript operation in Shiny apps	Attali (2018)
	shinyWidgets	package for offering custom widgets for Shiny apps	Perrier et al. (2019)
GIS	**leaflet**	package for implementing interactive map	Cheng et al. (2019)
	leaflet.extras	package for providing additional functionality to **leaflet**	Karambelkar and Schloerke (2018)
	raster	package for manipulating gridded spatial data	Hijmans (2019)
	maptools	package for handling GIS objects	Bivand and Lewin-Koh (2019)
	Rgeos	package for topology operation on geometries	Bivand and Rundel (2019)
	rgdal	package for handling geospatial data, especially for projection	Bivand et al. (2019)
Visualization	**ggplot2**	package for declaratively creating static graphics	Wickham et al. (2019)
	rAmCharts	package for using AmCharts JavaScript API to implement interactive charts	Thieurmel et al. (2019)
	rhandsontable	package for using Handsontable JavaScript API to implement interactive table	Owen (2018)
	formattable	package for creating vectors and data frames with custom format	Ren and Russell (2016)
Database	**DBI**	package for handling relational database in R	R Special Interest Group on Databases (R-SIG-DB) et al. (2019)
	rhdf5	package for handling HDF5 format data in R	Fischer et al. (2019)
Interface to other languages	**Rjson**	package for handling JSON format data in R	Couture-Beil (2018)
	Rcpp	package for integrating R functions with C++	Eddelbuettel et al. (2020)
	reticulate	package for working with Python in R	Ushey et al. (2020)

as web-based apps targeting stakeholders. A Shiny app requires two files, namely *ui.R* and *server.R*, where the former sketches the graphical user interface (i.e., front-end), while the latter implements routines to process user inputs (i.e., backend). The package ships with a number of application programming interfaces (APIs) that encompass common UI elements, such as sliders, buttons, file upload/download, etc., for accepting user inputs and for creating output panels. In addition, thanks to the community support, other packages have been continuously introduced to embrace state-of-the-art visualization techniques and software related to geographic information systems (GIS). As an example, Table 1 lists a number of packages for implementing common functionalities in modern EDSS. Some literature already reported the use of R Shiny for EDSS development in their case studies. Whateley et al. (2015) developed a Shiny app for performing climate risk evaluations of small-scale water utilities in the northeastern U.S. Hewitt and Macleod (2017) discussed several criteria for implementing user-oriented EDSS and presented a Shiny app as a qualified example. Ye et al. (2018) proposed to use Shiny for developing cloud-based water resource analysis tools with a demonstration of a case study in China.

Despite the fact that Shiny apps are easy to implement for R users, deploying them in a web-based operational context is more cumbersome and technical, a task that is often beyond the specialties of environmental scientists. What is worse, native Shiny apps do not allow multiple users to access the same app simultaneously, which is clearly in conflict with the democratic use of EDSS among stakeholders. One solution to circumvent those issues is to use *Shinyapps.io*, a web provider specific for hosting Shiny applications with a subscription cost. Fortunately, alternatives are available and are implemented in the proposed framework, which allow the deployment of Shiny apps either on a private server or on the cloud using Docker.

2.2. Docker

Docker is an open source virtualization software, which is very popular among developers in recent years (Bernstein, 2014; Merkel, 2014). It allows applications to be encapsulated as a Docker image, a file including the application, its dependencies and fundamentals of an operating system (OS) for running apps. When launched in Docker, the image is incarnated into a running process in an isolated environment, called a container. The isolation also provides security benefit since it can protect host machine from being affected by mulfunctioning apps in containers. One notable difference between Docker and a virtual machine (VM) is that Docker does not require a virtual OS as VM does (see Fig. 1), making it more lightweight (i.e., better scalability) while still keeping apps in isolation (i.e., portability and security). As a consequence, developers can use existing hardware more efficiently. In addition, upscaling Docker-based applications is straightforward since it only needs duplication of containers, based on users' demand. All those features imply that Docker-based applications can easily fit into a production context such as an enterprise cloud or a private server. Last but not least, to deploy an application into the Docker platform only requires preparing an instruction file (named *Dockerfile*) without modifying underlying infrastructure of an application.

In addition to Shiny and Docker, another two open source software products are used to construct a complete framework, namely ShinyProxy (Reese, 2008) and Nginx (Verbeke and Michielssen, 2016). The ShinyProxy is used for communicating between Shiny apps and Docker, and it also provides additional features such as user authorization, while the Nginx is a web-serving program for managing web apps easily and securely. Both ShinyProxy and Nginx are provided as Docker images which can directly be run on the Docker platform. The following section proceeds with a detailed explanation of the proposed integrated framework.

2.3. The infrastructural design of the framework

The proposed framework is shown in Fig. 2. Specifically, for a web-based application a browser serves as front-end to present graphic interface to end-users. Applications can be accessed on the Internet via a uniform resource identifier (URI) that locates resources on a server running EDSS. On the server side, the containerized software (i.e., Shiny apps, ShinyProxy, Nginx) are deployed inside the Docker environment, where Nginx acts as a reverse proxy that directs users' requests to ShinyProxy, which then launches the required Shiny applications. The ShinyProxy organizes and manages Shiny applications in the sense that when multiple users try to access the same Shiny app, it will automatically duplicate the containerized Shiny app to match the demand, deleting them when the task is finished. Therefore, it resolves the restriction of single-user access in native Shiny apps as mentioned before. Data encryption has also been addressed automatically to provide a secured Internet traffic compatible with modern HTTPS standard. The bottom part of our infrastructure are containerized Shiny apps. It should be mentioned that a containerized program is ephemeral, meaning that intermediates (e.g., the model outputs, user inputs, etc.) generated from a running app will be destroyed after users' disconnection. Therefore, permanent information must not be packed with apps into containers. In Docker, this problem can be solved by having a database external to a containerized program as shown in Fig. 2. Besides the database, the configuration file of ShinyProxy is also located externally to the Docker environment so developers can adapt it when necessary.

2.4. Development workflow

The workflow of developing a web-based EDSS using our framework can be summarized in the following four steps:

Step 1. Developing Shiny applications;
Step 2. Preparing Docker image files for the app;
Step 3. Adapting the configuration file for ShinyProxy;
Step 4. Deploying the software on Docker platform;

Among them, Step 1 is the part which requires main effort from researchers. It involves coupling models and, optionally, databases or external databases to apps. For models that have been written in R, researchers only need to wrap them with the Shiny APIs. Existing models programmed with other languages (i.e. legacy models) can be loosely coupled by implementing in R the processing routines of input/output files to those models (see the example in Section 3.1). The framework also comes with a built-in MySQL database (DuBois and Foreword By-Widenius, 1999) and an interface for manipulating tables in it, which can be useful for cases when data should be stored permanently outside the containerized apps (see the example in Section 3.2).

Step 2 packs the developed Shiny app into a Docker image, which can then be run as Docker container(s) during deployment. The image can be shared among researchers so others can reproduce the application on their system. In Step 3, developers have to adapt two configuration files, namely the *application.yml* and *docker-compose.yml*. The first file is used to configure the ShinyProxy so it can automatically manage all Shiny apps as well as user groups, while the second file is used for setting up a server on the Internet. As shown in Fig. 3, a minimal adaptation requires adding the image name of developed Shiny apps, creating different user groups and specifying a server's domain name. Specifically, the "user specification" block defines legitimate users with their authorization details. Therefore, only the users in this list can access apps. The "Shiny app specification" defines the parameters of each Shiny app integrated within the framework so that they can be correctly displayed and executed. In the "domain specification" users have to specify a server's domain name, such as www.example.com. Detailed instructions can be found on our GitHub repository (https://github.com/cocomice/EDSS

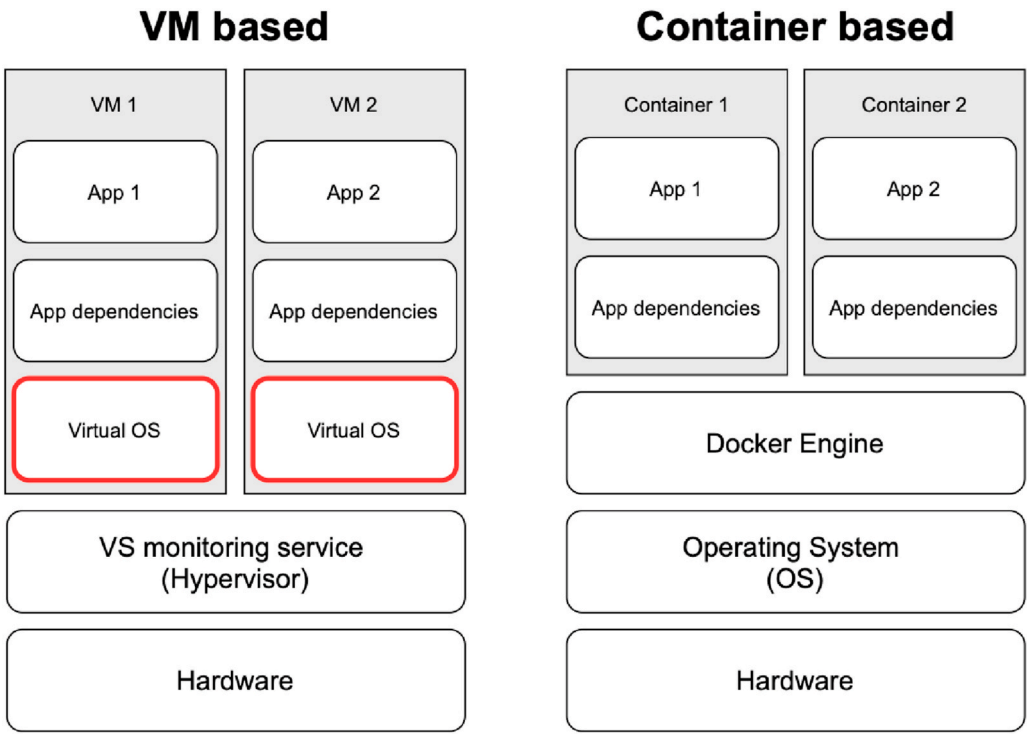

Fig. 1. Infrastructural difference between a Virtual Machine (VM) based and a container based approach. The main benefit with a container based approach is the elimination of the virtual OS, which makes it a more efficient way of exploiting hardware resources.

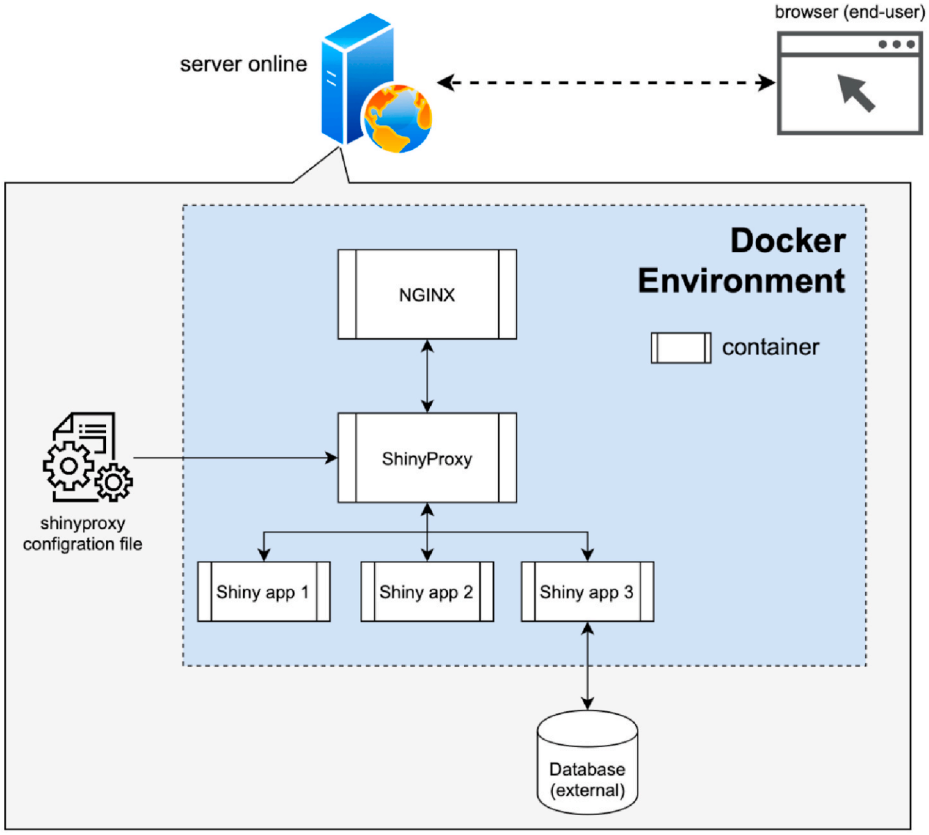

Fig. 2. The software infrastructure for a web-based EDSS using the proposed framework. Anything within the blue block lives inside the Docker environment and hence is independent from the server's OS. As a result, this framework can be used for any type of machine with Docker installed. (For interpretation of the references to colour in this figure legend, the reader is referred to the Web version of this article.)

_dep_framework.git).

In the end, the entire EDSS system can be launched by running "docker-compose run -d". End-users can visit the applications on the Internet by browsing the domain specified in the configuration file. The framework has a built-in portal to manage all Shiny apps hosted in the system, as shown in Fig. 4. The first page (Fig. 4(a)) is the login screen from which legitimate users can then enter the navigation page (Fig. 4 (b)) to access the Shiny apps available to him/her. Then each app can be opened individually for use. Behind the scenes resides the automatic server setup, which allows the renewal of HTTPS certificates, tracking application status and easy maintenance of the database. The Docker-based containerization technique allows multiple users to open the same app simultaneously. Therefore, an EDSS developed within the proposed framework can be easily fitted into operational context. In addition, by distributing working folders others can reproduce the same system on their machine by repeating Steps 2 to 4.

3. Example applications

3.1. On-line groundwater modelling platform

In this section we present an EDSS for conjunctive water management by coupling a calibrated groundwater model in a Shiny app. The Shiny package is easy-to-use and effective for building a user-friendly front-end interface, while computational routines in the back-end are essentially models that can either be transformed from researchers' own R scripts or be integrated with legacy models. The integration not only saves development effort but, more importantly, can take advantage of a program written in low-level programming languages (e.g., C/C++, Fortran), which are less human-interpretable but more computationally efficient compared to R. Thanks to the versatility of R language and package support, legacy models can be loosely integrated with Shiny applications by communicating only inputs/outputs between the Shiny apps and the models. In this example we showed an application which

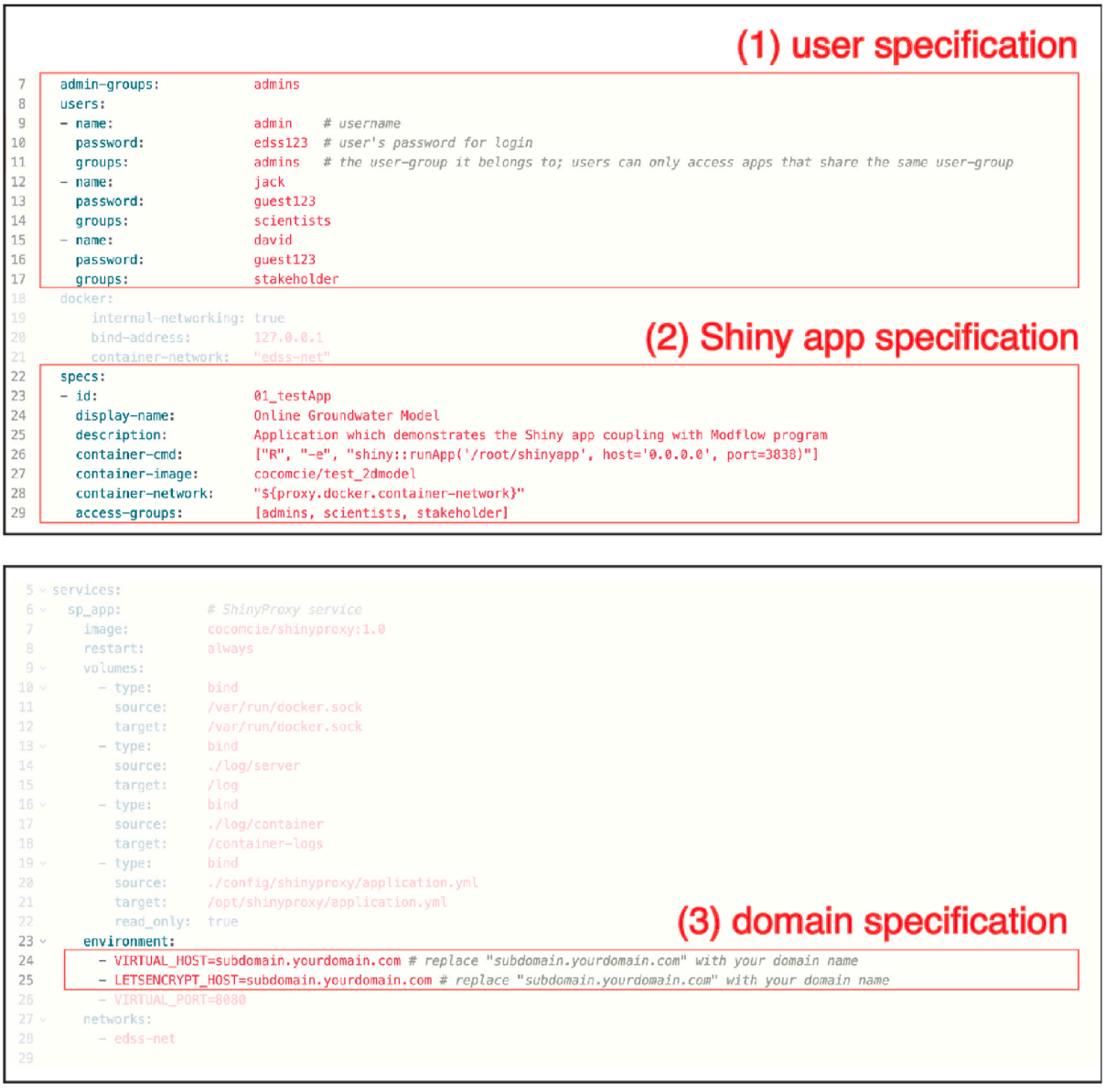

Fig. 3. The partial content in *application.yml* (top) and *docker-compose.yml* (bottom) files, respectively. The non-grayed codes are example configurations, which shall be adapted by users for their own cases.

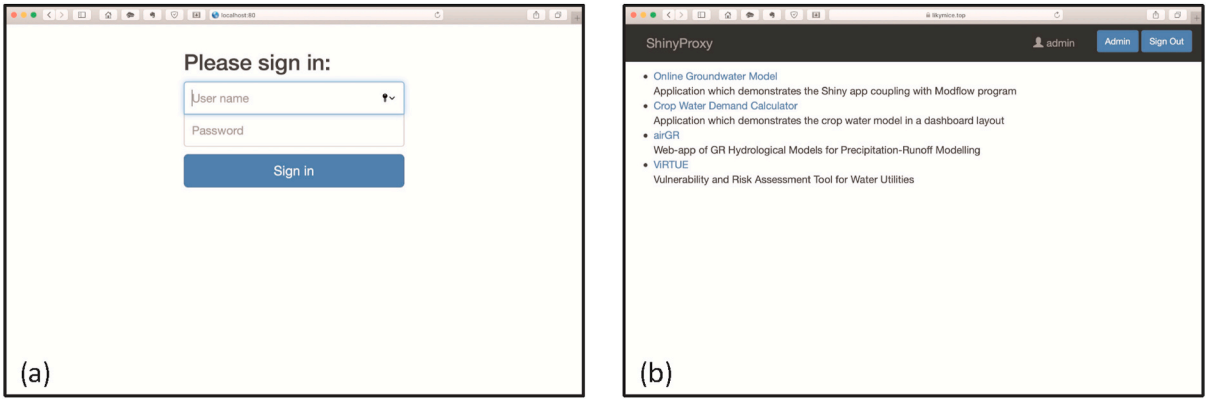

Fig. 4. The interface of a deployed EDSS on a browser. (a) The login screen prevents unauthorized users from accessing the apps, and (b) the navigation page organizes hosted Shiny apps in a list from which users can access individual app.

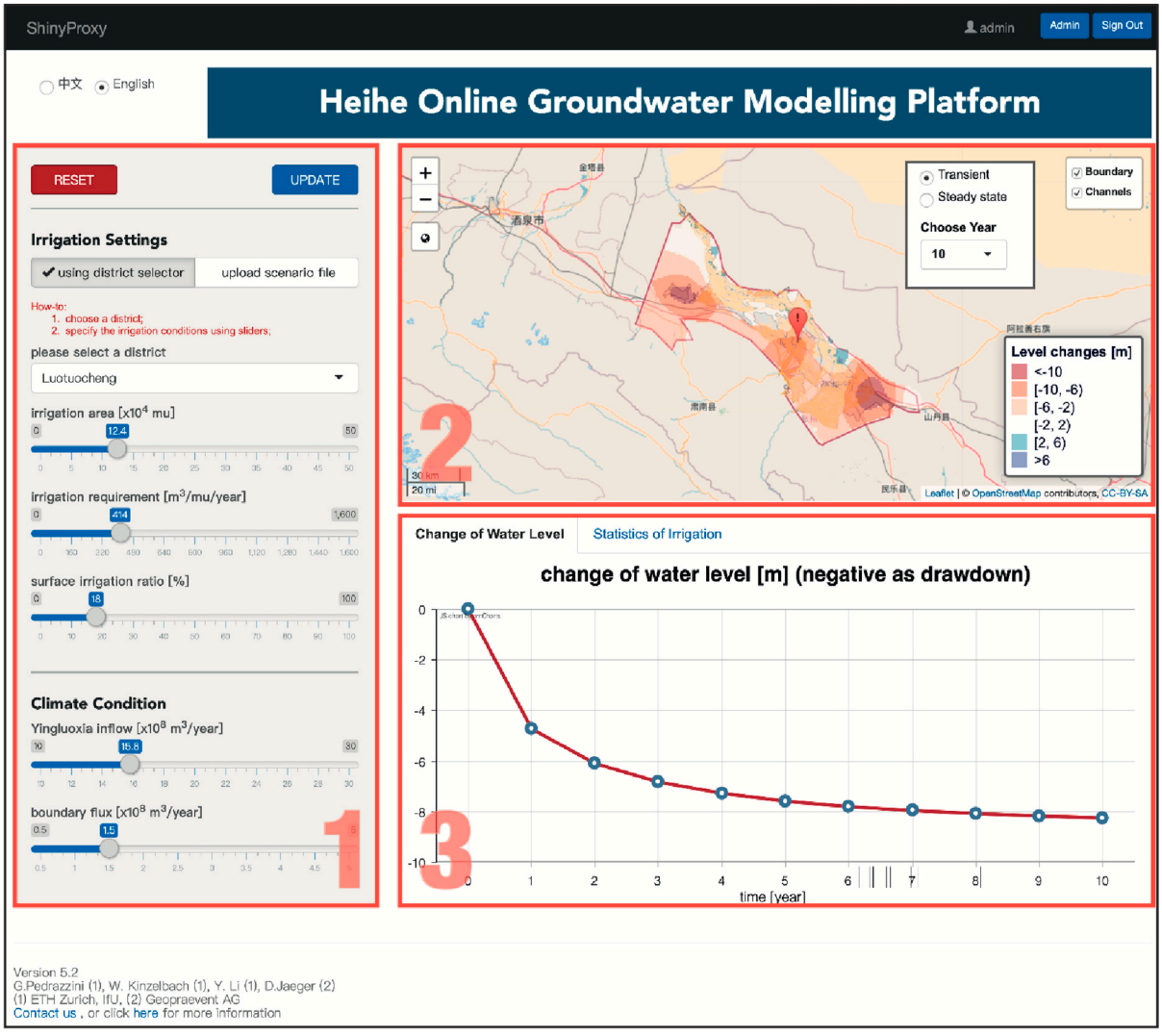

Fig. 5. The interface of Heihe online groundwater modelling platform. Part 1 is the input panel for scenario configuration from users, part 2 the output of simulated spatial head using interactive map, and part 3 the time series plot for water level change at a selected location and overall water use statistics, respectively. The UI language can switch between English and Chinese by clicking the radio button on top left corner.

directly integrates a Shiny app with Modflow (Harbaugh et al., 2000), a widely used groundwater modelling program written in Fortran. The model has been calibrated for a shallow aquifer in the middle reach of Heihe River Basin from our previous work (Li and Wolfgang Kinzelbach, 2020).

Fig. 5 shows the interface with three highlighted parts. The first one is the input panel where users can specify scenarios to drive the groundwater model, which includes the irrigation water use (i.e., planning decisions) and boundary conditions. The interaction is done via moving of sliders or uploading files. The second and third parts are output windows, where the top panel shows an interactive map of the Heihe mid-reach for displaying spatially varying hydraulic heads. The bottom panel shows a time-series plot for head change at a user-chosen location and overall statistics of water use. All figures are created as interactive plots using **leaflet** (Cheng et al., 2019) and **rAmCharts** (Thieurmel et al., 2019) packages, so users can further inspect simulated results in the spirit of a data-driven decision making process (Provost and Fawcett, 2013). The UI language can be switched between Chinese and English by clicking the radio button at the top left corner.

On the back-end the app processes users' inputs and triggers simulations of the groundwater model. It can easily become an overwhelming task if one wants to implement a Modflow model in R. Instead, in this case the Shiny app only prepares input files compatible with Modflow, calls the native Modflow executable compiled in Fortran and then retrieves outputs for producing plots. The integration between Shiny app and Modflow model is thus a loose coupling as shown in Fig. 6. Similar EDSS can follow this manner to include other legacy models for implementing real-time simulation routines. However, one should note that for large-scale complex models the latency between users' inputs and output display can be large due to longer simulation time. Therefore, instead of explicitly integrating a model in an EDSS, one can store results for a sizable number of input scenarios to build a lookup table. Then the output from a given input scenario of a user can be retrieved rapidly.

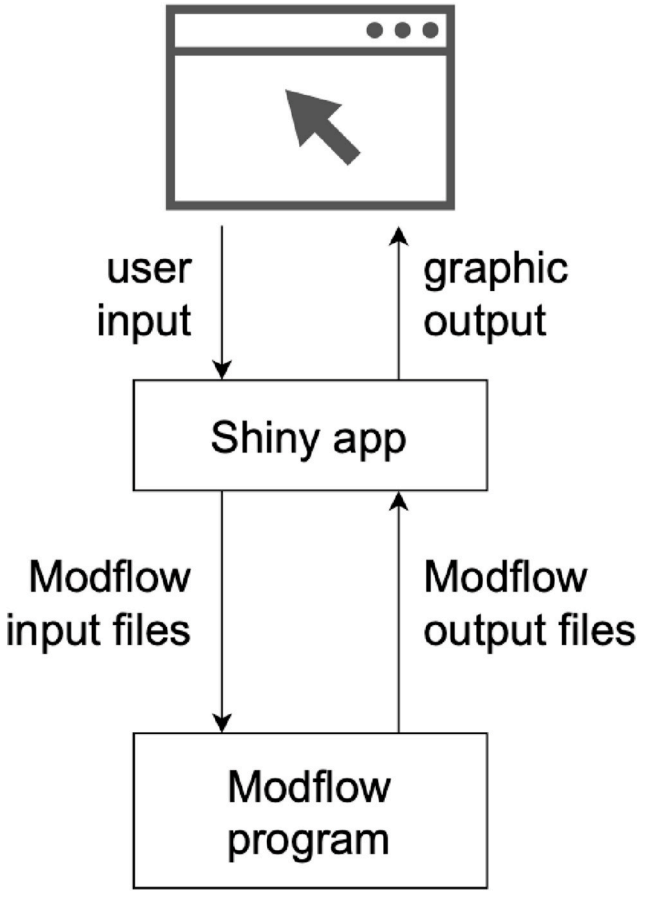

Fig. 6. The flowchart of input/output communication in the Heihe groundwater modelling platform coupled with the native Fortran-based Modflow program. The Shiny app processes users' inputs into Modflow input files and retrieve outputs after simulation for displaying results on the front-end UI.

3.2. *Dashboard for crop irrigation management*

Crop irrigation planning is another common subject for applying EDSS. In this example, we show a Shiny app which includes a crop model and a data portal for monitoring meteorological conditions. The data and materials are based on a pilot region in China, named Guantao County located in North China Plain where the subsurface groundwater resource is jeopardized by overpumping practices (e.g., Liu et al., 2011; Li et al., 2019). The UI is implemented as a dashboard consisting of three tabs (see Fig. 7 (a)-(c)). The **Home** tab provides background information about the study region. The **Irrigation Calculator** tab contains an interface to a crop model for estimating crop water deficit. The tool is implemented following FAO's AquaCrop routines (Steduto et al., 2009). With user specified inputs (i.e., crop type, soil type, planting date) on the left panel, the model calculates expected water deficit for wet/normal/dry years based on observed meteorological data from the database. In the **Data Portal** tab, users can visualize those meteorological data or download them. For demonstration purposes, in this example only synthetic precipitation data is used.

It should be mentioned that, by the design of the containerization technique, the groundwater modelling app from the first example will be reset automatically to its initial state whenever users close the application. Therefore, any data generated from the system is only available during users' access. Instead, the database used in the crop water management app is external to the containerized Shiny app, stored in a database server (i.e., MySQL; see Fig. 7 (d)), and lives permanently on a disk regardless of users' access. In an operational context with data monitoring infrastructure, this decoupling between a database and a containerized Shiny app allows the data to evolve with real-time data transmission, while tools built in a Shiny app can always access the latest data for decision support.

In summary, the two examples showed the capability of the framework to produce simple and user-friendly interfaces without compromising common functionalities (e.g., GIS, interactive visualization, file upload/download) seen in a modern EDSS. It is also possible to integrate with external database or legacy models for more practical uses. The authors also set up a repository, where other EDSS from literature were adapted according the proposed framework. The goal is to reuse materials from existing case studies and to further promote knowledge sharing.

4. Conclusions and outlook

In this work we introduced an integrated software framework for developing EDSS. The framework is based on R Shiny package, Docker and a few other open-source software products, which allows researchers to benefit from the versatility of R for developing web-based apps while lowering technical hurdles in deploying/maintaining EDSS on servers. To sum up, the following features of the framework have been identified: 1) Easy implementation of user-friendly and interactive web-based applications; 2) high flexibility for integrating legacy modelling tools; 3) low technical barrier to setup the system on a server for production-ready context.

Those features were demonstrated using two examples, one based on groundwater modelling for conjunctive water planning and the other for crop irrigation management. These and more examples are included in a public repository to promote knowledge sharing and replication of case

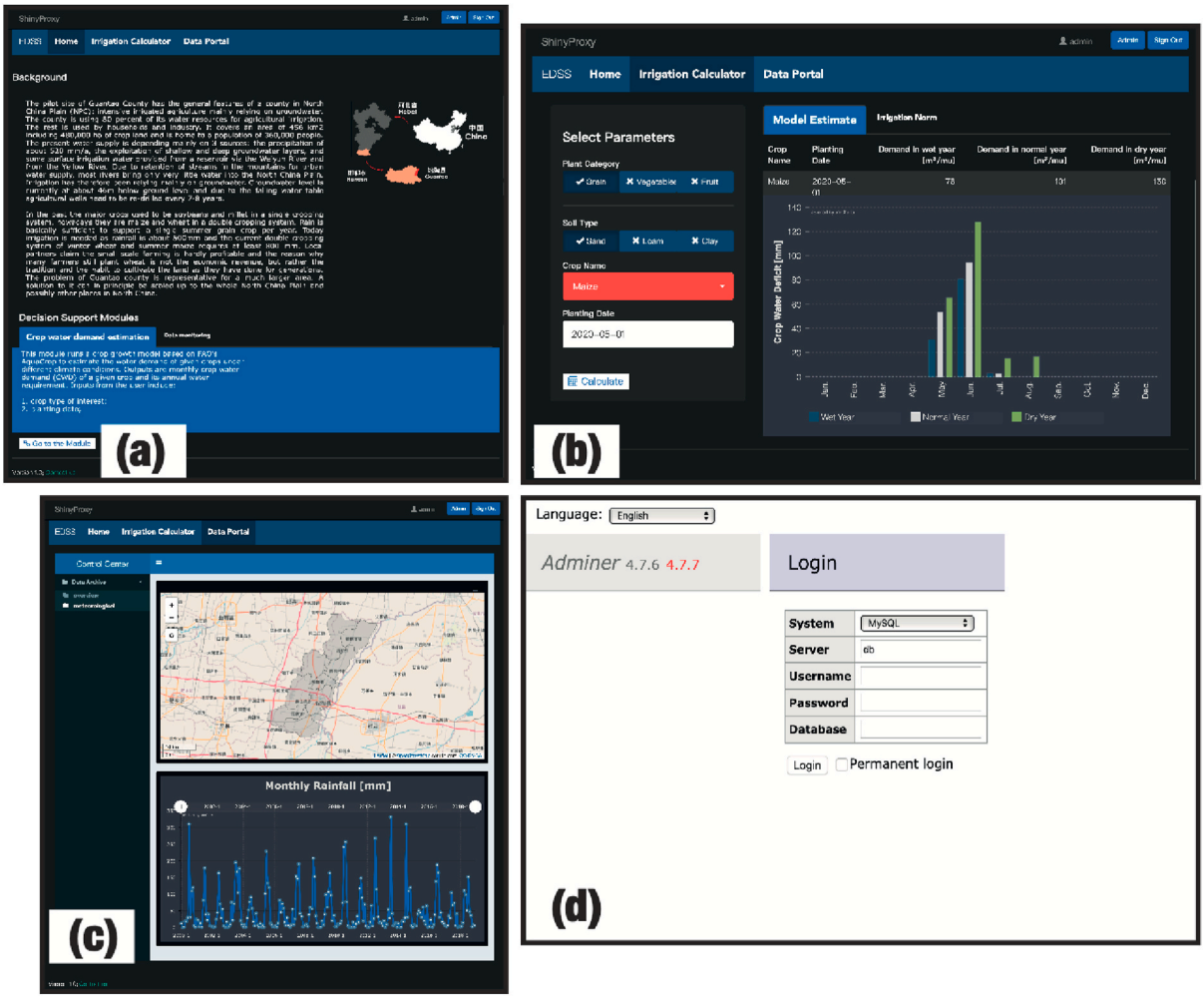

Fig. 7. The interface of the agricultural water management dashboard, where panel (a) is the Home page of the study area, panel (b) the crop water demand model (Irrigation Calculator), and panel (c) the monitoring module for displaying meteorological observation or other sensor data (Data Portal). Panel (d) is the interface for the external SQL database connected to the Shiny app.

studies from literature. Due to the popularity of R language and its wide use in the scientific community, the proposed framework can also be used in an educational context for teaching EDSS.

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.

Acknowledgements

This work is supported by the Swiss Agency for Development and Cooperation under the project "Rehabilitation and management strategy for over-pumped aquifers under a changing climate". We thank Dr. Beatrice Marti from hydrosolution Ltd. for contributing the crop model, Mr. Gianni Pedrazzini from Holinger Ltd. for contributing the groundwater model and Dr. Wolfgang Kinzelbach as the leader of the project. We also thank the developers of open-source ShinyProxy and other Docker-based software used in this work.

References

Andrews, F., Croke, B., Jakeman, A., 2011. An open software environment for hydrological model assessment and development. Environ. Model. Software 26 (10), 1171–1185. https://doi.org/10.1016/j.envsoft.2011.04.006.

Attali, D., 2018. Shinyjs: easily improve the user experience of your Shiny apps in seconds available at: https://CRAN.R-project.org/package=shinyjs. R package version 1.0.

Bernstein, D., 2014. Containers and cloud: from LXC to docker to kubernetes. IEEE Cloud Computing 1 (3), 81–84. https://doi.org/10.1109/MCC.2014.51.

Bivand, R., Lewin-Koh, N., 2019. Maptools: tools for handling spatial objects available at: https://CRAN.R-project.org/package=rgdal. Rpackage version 1.4-7.

Bivand, R., Rundel, C., 2019. Rgeos: interface to geometry engine - open source ('GEOS') available at: https://CRAN.R-project.org/package=shinythemes. Rpackage version 1.1.2.

Bivand, R., Keitt, T., Rowlingson, B., 2019. Rgdal: bindings for the 'geospatial' data abstraction library available at: https://CRAN.R-project.org/package=rgdal. Rpackage version 1.4-7.

Chang, W., 2018. Shinythemes: themes for Shiny available at: https://CRAN.R-project.org/package=shinythemes. R package version 1.1.2.

Chang, W., Borges Ribeiro, B., 2018. Shinydashboard: create dashboards with 'Shiny' available at: https://CRAN.R-project.org/package=shinydashboard. Rpackage version 0.7.1.

Chang, W., Cheng, J., Allaire, J., Xie, Y., McPherson, J., et al., 2017. Shiny: web application framework for r. R package version 1 (5).

Cheng, J., Karambelkar, B., Xie, Y., 2019. Leaflet: create interactive web maps with the JavaScript 'leaflet' library available at: https://CRAN.R-project.org/package=leaflet. Rpackage version 2.0.3.

Couture-Beil, A., 2020. rjson: JSON for R available at: https://CRAN.R-project.org/package=rjson. Rpackage version 0.2.20.

DuBois, P., Foreword By-Widenius, M., 1999. MySQL. New riders publishing.

Eddelbuettel, D., Francois, R., Allaire, J., Ushey, K., Kou, Q., Russell, N., Bates, D., Chambers, J., 2020. Rcpp: seamless R and C++ integration available at: https://CRAN.R-project.org/package=Rcpp. Rpackage version 1.0.4.6.

Fischer, B., Pau, G., Smith, M., 2019. rhdf5: R Interface to HDF5 available at: https://github.com/grimbough/rhdf5. Rpackage version 2.28.1.

Hadded, R., Nouiri, I., Alshihabi, O., Maßmann, J., Huber, M., Laghouane, A., Yahiaoui, H., Tarhouni, J., 2013. A decision support system to manage the

groundwater of the Zeuss Koutine Aquifer using the WEAP-MODFLOW framework. Water Resour. Manag. 27 (7), 1573–1650. https://doi.org/10.1007/s11269-013-0266-7.
Harbaugh, A.W., Banta, E.R., Hill, M.C., McDonald, M.G., 2000. Modflow-2000, the U. S. Geological Survey modular ground-water model-user guide to modularization concepts and the ground-water flow process. Open File Rep. U. S. Geol. Surv. (92), 134.
Hewitt, R.J., Macleod, C.J.A., 2017. What do users really need? participatory development of decision support tools for environmental management based on outcomes. Environments 4 (4), 88. https://doi.org/10.3390/environments4040088.
Hijmans, R.J., 2019. Raster: geographic data analysis and modeling available at: https://CRAN.R-project.org/package=raster. Rpackage version 3.0-7.
Karambelkar, B., Schloerke, B., 2018. leaflet.extras: extra Functionality for 'leaflet' Package available at: https://CRAN.R-project.org/package=leaflet.extras. Rpackage version 1.0.0.
Li, N., Kinzelbach, W., Li, H., Li, W., Chen, F., 2019. Decomposition technique for contributions to groundwater heads from inside and outside of an arbitrary boundary: application to Guantao County, North China Plain. Hydrol. Earth Syst. Sci. 23 (7), 2823–2840. https://doi.org/10.5194/hess-23-2823-2019.
Kumar, S., Godrej, A.N., Grizzard, T.J., 2015. A web-based environmental decision support system for legacy models. J. Hydroinf. 17 (6), 874–890. https://doi.org/10.2166/hydro.2015.007.
Lai, J., Lortie, C.J., Muenchen, R.A., Yang, J., Ma, K., 2019. Evaluating the popularity of R in ecology. Ecosphere 10 (1), 2150–8925. https://doi.org/10.1002/ecs2.2567.
Li, Y., Kinzelbach, Wolfgang, 2020. Resolving conflicts between irrigation agriculture and ecohydrology using many-objective robust decision making. J. Water Resour. Plann. Manag. 146 (9) https://doi.org/10.1061/(ASCE)WR.1943-5452.0001261.
Liu, J., Cao, G., Zheng, C., 2011. Sustainability of groundwater resources in the north China plain. In: Jones, J.A.A. (Ed.), Sustaining Groundwater Resources, Pages. Springer Netherlands, Dordrecht, pp. 69–87.
Liu, J., Dietz, T., Carpenter, S.R., Alberti, M., Folke, C., Moran, E., Pell, A.N., Deadman, P., Kratz, T., Lubchenco, J., Ostrom, E., Ouyang, Z., Provencher, W., Redman, C.L., Schneider, S.H., Taylor, W.W., 2007. Complexity of coupled human and natural systems. Science 317 (5844), 1513–1516. https://doi.org/10.1126/science.1144004.
Loucks, D.P., 1995. Developing and implementing decision support systems: a critique and a challenge. JAWRAJ. Am. Water Resour. Assoc. 31 (4), 571–582. https://doi.org/10.1111/j.1752-1688.1995.tb03384.x.
Matthies, M., Giupponi, C., Ostendorf, B., 2007. Environmental decision support systems: current issues, methods and tools. Environ. Model. Software 22 (2), 123–127. https://doi.org/10.1016/j.envsoft.2005.09.005.
McDonald, S., Mohammed, I.N., Bolten, J.D., Pulla, S., Meechaiya, C., Markert, A., Nelson, E.J., Srinivasan, R., Lakshmi, V., 2019. Web-based decision support system tools: the Soil and Water Assessment Tool Online visualization and analyses (SWATOnline) and NASA earth observation data downloading and reformatting tool (NASAaccess). Environ. Model. Software 120, 104499. https://doi.org/10.1016/j.envsoft.2019.104499.
Merkel, D., 2014. Docker: lightweight linux containers for consistent development and deployment. Linux J. 2 (239), 2014.
Milly, P.C.D., Betancourt, J., Falkenmark, M., Hirsch, R.M., Kundzewicz, Z.W., Lettenmaier, D.P., Stouffer, R.J., 2008. Stationarity is dead: whither water management? Science 319 (5863), 573–574. https://doi.org/10.1126/science.1151915.
Mir, S.A., Quadri, S.M.K., 2009. Decision support systems: concepts, progress and issues – a review. In: Lichtfouse, E. (Ed.), Climate Change, Intercropping, Pest Control and Beneficial Microorganisms. Springer Netherlands, Dordrecht, pp. 373–399.
Morton, M.S.S., 1971. Management Decision Systems: Computer-Based Support for Decision Making. Division of Research, Graduate School of Business Administration, Harvard University.
Murugesan, S., Deshpande, Y., Hansen, S., Ginige, A., 2001. Web engineering: a new discipline for development of web-based systems. In: Murugesan, S., Deshpande, Y. (Eds.), Web Engineering: Managing Diversity and Complexity of Web Application Development, Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp. 3–13.
Owen, J., 2018. Rhandsontable: interface to the 'handsontable.js' Library available at: https://CRAN.R-project.org/package=rhandsontable. R package version 0.3.7.
Pachauri, R.K., Allen, M.R., Barros, V.R., Broome, J., Cramer, W., Christ, R., Church, J.A., Clarke, L., Dahe, Q., Dasgupta, P., et al., 2014. Climate Change 2014: Synthesis Report. Contribution of Working Groups I, II and III to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change. Ipcc.
Perrier, V., Meyer, F., Granjon, D., 2019. shinyWidgets: Custom inputs widgets for Shiny available at: https://CRAN.R-project.org/package=shinyWidgets. R package version 0.4.9.
Provost, F., Fawcett, T., 2013. Data science and its relationship to big data and data-driven decision making. Big Data 1 (1), 51–59. https://doi.org/10.1089/big.2013.1508.
R Special Interest Group on Databases (R-SIG-DB), Wickham, H., Müller, K., 2019. DBI: R database interface available at: https://CRAN.R-project.org/package=DBI. R package version 1.1.0.
Reese, W., 2008. Nginx: the high-performance web server and reverse proxy. Linux J. 2 (173), 2008.
Ren, K., Russell, K., 2016. Formattable: create 'formattable' data structures available at: https://CRAN.R-project.org/package=formattable. R package version 0.2.0.1.
Shao, H., Yang, W., Lindsay, J., Liu, Y., Yu, Z., Oginskyy, A., 2017. An open source gis-based decision support system for watershed evaluation of best management practices. JAWRAJ. Am. Water Resour. Assoc. 53 (3), 521–531. https://doi.org/10.1111/1752-1688.12521.
Slater, 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. 23 (7), 2939–2963. https://doi.org/10.5194/hess-23-2939-2019.
Smith, F., 1991. Software Prototyping: Adoption, Practice, and Management. McGraw-Hill London.
Swain, N.R., Latu, K., Christensen, S.D., Jones, N.L., Nelson, E.J., Ames, D.P., Williams, G.P., 2015. A review of open source software solutions for developing water resources web applications. Environ. Model. Software 67, 108–117. https://doi.org/10.1016/j.envsoft.2015.01.014.
Steduto, P., Hsiao, T., Raes, D., Fereres, E., 2009. AquaCrop – the FAO crop model to simulate yield response to water:I. concepts and underlying principles. Agron. J. 101 (3), 426–437.
Swain, N.R., Christensen, S.D., Snow, A.D., Dolder, H., Espinoza-Dávalos, G., Goharian, E., Jones, N.L., Nelson, E.J., Ames, D.P., Burian, S.J., 2016. A new open source platform for lowering the barrier for environmental web app development. Environ. Model. Software 85, 11–26. https://doi.org/10.1016/j.envsoft.2016.08.003.
Thieurmel, B., Marcelionis, A., Petit, J., Salette, E., Robert, T., 2019. rAmCharts: JavaScript charts tool available at: https://CRAN.R-project.org/package=rAmCharts. R package version 2.1.12.
Tippmann, S., 2015. Programming tools: adventures with R. Nature 517 (7532), 109–110. https://doi.org/10.1038/517109a.
Ushey, K., Allaire, J., Tang, Y., 2020. reticulate: interface to 'Python' available at: https://CRAN.R-project.org/package=reticulate. R package version 1.16.
Van Rossum, G., et al., 2007. Python programming language. In: USENIX Annual Technical Conference, vol. 41, 36.
Verbeke, T., Michielssen, F., 2016. Shinyproxy–open Source Enterprise Deployment for Shiny. GitHub repository.
Volk, M., Lautenbach, S., van Delden, H., Newham, L.T.H., Seppelt, R., 2010. How can we make progress with decision support systems in landscape and river basin management? lessons learned from a comparative analysis of four different decision support systems. Environ. Manag. 46 (6), 834–849. https://doi.org/10.1007/s00267-009-9417-2.
Walker, W.E., Lempert, R.J., Kwakkel, J.H., 2013. Deep uncertainty. In: Gass, S.I., Fu, M. C. (Eds.), Encyclopedia of Operations Research and Management Science. Springer US, pp. 395–402.
Whateley, S., Walker, J.D., Brown, C., 2015. A web-based screening model for climate risk to water supply systems in the northeastern United States. Environ. Model. Software 73, 64–75. https://doi.org/10.1016/j.envsoft.2015.08.001.
Wickham, H., Chang, W., Henry, L., Pedersen, T.L., Takahashi, K., Wilke, C., Woo, K., Yutani, H., 2019. ggplot2: create elegant data visualisations using the grammar of graphics available at: https://CRAN.R-project.org/package=ggplot2. R package version 3.2.1.
Ye, F., Chen, Y., Huang, Q., Li, L., 2018. Developing cloudbased tools for water resources data analysis using R and Shiny. In: Barolli, L., Zhang, M., Wang, X.A. (Eds.), Advances in Internetworking, Data & Web Technologies, vol. 6. Springer International Publishing, Cham, pp. 289–297.
Zulkafli, Z., Perez, K., Vitolo, C., Buytaert, W., Karpouzoglou, T., Dewulf, A., De Bièvre, B., Clark, J., Hannah, D.M., Shaheed, S., 2017. User-driven design of decision support systems for polycentric environmental resources management. Environ. Model. Software 88, 58–73. https://doi.org/10.1016/j.envsoft.2016.10.012.