Environmental Modelling and Software 107 (2018) 59–63

Contents lists available at ScienceDirect

Environmental Modelling & Software

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

An R package for EPANET simulations

Ernesto Arandia, Bradley J. Eck*

IBM Research, Dublin, Ireland

ARTICLE INFO

Keywords:
Water networks
R
EPANET
Hydraulic modeling

ABSTRACT

The EPANET software for modeling piping networks is widely used for the design and analysis of water systems. This short communication describes epanet2toolkit, a package for accessing the EPANET simulation engine and application programming interface from the R environment for computing and graphics. Users of the package may run extended period simulations of hydraulics and water quality or create custom applications using the data exploration, visualization, and analysis capabilities of R. epanet2toolkit enables interactive use of EPANET on a variety of computing platforms, provides integrated error checking, and contributes a publicly available test suite. The package is available through the Comprehensive R Archive Network and GitHub.com.

1. Introduction

Modeling the behavior of water piping networks is an active area both in industry and academia with published studies covering a very broad range of problems, including design, optimization, and management. Several pieces of software are available for simulating piping networks including proprietary packages such as WaterGEMS(r), InfoWater(r), and KYPipe, and freely available packages such as EPANET (Rossman, 2000). Originally developed by the United States Environmental Protection Agency, EPANET has over 2000 citations on Google Scholar making it the most popular choice for academic applications.

The EPANET simulation engine is written in the C language for portability and performance. Bindings have been developed to call the simulation engine from several other languages. The Programmer's Toolkit (Rossman, 2008) includes bindings for Visual Basic and Pascal. More recently, bindings for Matlab (Eliades et al., 2016) and Python (Pathirana, 2016) have been developed. This paper describes bindings for the R environment (R Core Team, 2013) provided in the form of an add-on package called epanet2toolkit.

As an R package for water resources modeling, epanet2toolkit joins a growing group of packages for water applications. The package epanetReader (Eck, 2016) compliments the simulation capability of epanet2toolkit with support for parsing EPANET data files into R objects. In a similar vein, the package swmmr by Leutnant (2017) provides an interface for the Storm Water Management Model. The dataRetrieval package by Hirsch and De Cicco (2015) supports downloading hydrological and water quality variables from several sources. The wq package by Jassby and Cloern (2015) aims at exploring environmental monitoring data. Various aspects of water supply reservoirs can be analyzed using packages reservoir by Turner and Galelli (2016), RSSOP by Arabzadeh et al. (2016), and WRSS by Arabzadeh et al. (2017). This growing collection of software aimed at environmental and water resources topics shows that R packages are becoming a popular way to share research outputs in this field.

Two versions of source code for the EPANET simulation engine are currently available. The latest release on the US EPA website (https://www.epa.gov/water-research/epanet) is version 2.0.12 from 2008. A community group called Open Water Analytics released a version 2.1 in 2016 (https://github.com/OpenWaterAnalytics/EPANET/releases/tag/v2.1). Since version 2.1 contains some extensions and bug fixes it was chosen for inclusion in epanet2toolkit.

The remainder of this article is organized as follows. First we give a short description of epanet2tookit's design and list of new functions provided by the package according to category (Section 2). Section 3 presents usage examples ranging from toolkit installation and full network simulation to solving more complex problems such as optimization and stochastic modeling. Finally, we present future developments and conclusions (Section 4).

2. Design

The EPANET simulation engine, as distinct from the Windows(r) based graphical user interface, can be invoked from the command line or through an application programming interface (API). The EPANET API comprises 74 C functions for carrying out customized simulations. The R package epanet2toolkit is designed as a wrapper for the EPANET API. R functions provided by the package have the same name as

* Corresponding author.
E-mail address: bradley.eck@ie.ibm.com (B.J. Eck).

https://doi.org/10.1016/j.envsoft.2018.05.016
Received 1 February 2018; Received in revised form 10 April 2018; Accepted 21 May 2018
Available online 06 June 2018
1364-8152/

Software availability

epanet2toolkit is available on the world wide web at https://cran.r-project.org/package=epanet2toolkit and https://github.com/bradleyjeck/epanet2toolkit
License MIT
System Requirements R version 3.0.2 or higher
Installation using install.packages() function

functions in the API, but differ in function arguments, return values, and handling of error codes.

Functions in the EPANET API return an integer error code and provide requested values by reference. For example, the function ENgetnodeindex takes two arguments: the node ID, and a pointer to an integer variable for storing the requested value (Listing 1). The value returned by the function is not the node index, but a code indicating whether the function call finished successfully or encountered an error or warning condition. To use this function, the calling application needs to allocate an int variable for storing the requested index and corresponding pointer for passing to the function. Checking the error code returned by the function is good practice, but not strictly required.

Listing 1. C function prototype from EPANET API

```
int ENgetnodeindex(char* nodeID, int* nodeindex);
```

epanet2toolkit exposes functions from the EPANET API to the R environment through two steps. First, existing functions of the EPANET API are wrapped by new C functions with return type void or SEXP so that they can be called from R. Second, new R functions are provided to call these new C functions. The R functions check their arguments, check the error codes returned by EPANET, and return the requested value.

As an example, the R function ENgetnodeindex takes a single argument and returns the node index (Listing 2). An R script calling this function passes the node ID as an argument and stores the returned index in the desired location or prints it to the screen. In case the underlying API call returned an error or warning, the R function raises an error or warning through R's exception handling system. Checking the error code returned from EPANET happens on every function call and is transparent to the user until an error or warning condition actually occurs.

Listing 2. R function from epanet2toolkit

```
ENgetnodeindex <- function(nodeID){...}
```

New functions for the R environment provided by epanet2toolkit are summarized in Table 1. Users familiar with the EPANET API will recognize that the R functions share the name of the underlying EPANET API functions they invoke. Using consistent function names provides a familiar interface to existing users and makes it easy to port existing applications to R.

Running a full simulation of hydraulics and water quality and writing the results to a file is accomplished with the function ENepanet. The function requires an input file in EPANET's .inp format and writes output files in text and or binary format. Accessing all other functions requires opening the EPANET engine with ENopen and closing it with ENclose.

Hydraulic and water quality simulations may be carried out as extended period or stepwise simulations. Stepwise simulations allow programs to interrogate values at each step. Extended period simulations encapsulate stepwise simulations, only reporting values for the times and network elements specified in the input file. Programs can use either stepwise or extended period functions for hydraulics and water quality but must be consistent within each category. Thus, ENopenH to ENcloseH are compatible with ENsolveQ but not with ENsolveH. ENopenQ to ENcloseQ are compatible with ENopenH to ENcloseH or ENsolveH but not ENsolveQ.

Information in the simulation engine can be interrogated and changed using an appropriate function beginning with “ENget” or “ENset”. All of these have not been tabulated here, but are documented in the package manual and symbolized in the table with the wildcard character “*”. In epanet2toolkit “ENget” functions return the value of interest and “ENset” functions return NULL invisibly unless an error occurs.

3. Example usage and capability

3.1. Installation and full network simulation

In order to use epanet2toolkit in an R session it must be installed and loaded. From within an R session the install.packages function downloads packages from a repository and installs them on the local system. R selects packages from the repository according to operating system. R also keeps track of installed packages so that users do not need to know or configure the package from their working directory. This behavior contrasts with bindings for EPANET which require compiling EPANET for the relevant operating system and architecture and configuring the dll location.

The default behavior of install.packages is to download pre-compiled binary packages for Windows and Mac OS(r) systems, resulting in a quick installation for most users. Package binaries are not available for other platforms (e.g. Linux(r)) and so in these cases packages are downloaded as source code and compiled locally. Local compilation increases install time slightly but still takes only a few minutes and happens automatically. Passing the installation option –install-tests includes the package test suite with the installation so that users can run tests as described in Section 3.4.

Once a package is installed, calling library loads it into the current R session. For example, the function ENepanet carries out a full

Table 1
Functions for the R environment provided by epanet2toolkit.

Name	Description
Full Simulations	
ENepanet()	Runs a full simulation
Opening and Closing EPANET engine	
ENopen()	Opens EPANET engine
ENclose()	Closes EPANET engine
Extended Period Simulation	
ENsolveH()	Solves the network hydraulics
ENsolveQ()	Solves the network water quality
Stepwise Hydraulic Simulation	
ENopenH()	Opens the hydraulics analysis system
ENinitH()	Initializes network prior to simulation
ENrunH()	Runs a single period hydraulic analysis
ENnextH()	Length of time to next hydraulic event
ENcloseH()	Closes the hydraulics analysis system
Stepwise Water Quality (WQ) Simulation	
ENopenQ()	Sets up for WQ analysis
ENinitQ()	Initializes WQ analysis
ENrunQ()	Computes WQ results at current time
ENnextQ()	Advances WQ sim. to start of next hyd. time period
ENstepQ()	Advances WQ sim. one WQ time step
ENcloseQ()	Closes the WQ analysis
Writing Files	
ENsaveH()	Saves hydraulic results to binary file
ENsaveinpfile()	Saves current data to INP text file
ENreport()	Writes simulation report file
Retrieving and Setting Parameter Values	
ENget*()	Functions that retrieve network information
ENset*()	Functions that set network parameter values

simulation and writes results to a report file (Listing 3). The network described in the '.inp' file and the simulation results written in the '.rpt' file can be further analyzed and visualized within R using the package epanetReader (Eck, 2016).

Listing 3. Installing the package and running a full network simulation

```
> install.packages("epanet2toolkit",
    INSTALL_opts = "--install-tests")
> library(epanet2toolkit)
> ENepanet("Net1.inp", "Net1.rpt")
```

Once epanet2toolkit is available within R, functions from the package can be used to access and modify network properties and used along with functions provided by R to carry out different types of analysis. The following sections provide some examples.

3.2. Accessing and changing network properties

To access or change network properties, the first step is to open the EPANET engine with ENopen. The value of a network parameter such as the length of the pipe having index 2 can be accessed using ENgetlinkvalue. The first argument specifies the link index and the second argument specifies the parameter to be retrieved. The value of this parameter can be changed using ENsetlinkvalue. The first two arguments are the same, but the third argument specifies the new value for the parameter. A subsequent call to ENgetlinkvalue confirms that the value was changed. Finally, ENclose closes the EPANET engine.

Further function calls could be added to Listing 4 – before ENclose – and run in sequence. This workflow contrasts with using the EPANET API from C in that calling different functions requires updating a script but not recompiling an application.

Listing 4. Accessing and changing network properties

```
> ENopen("Net1.inp", "Net1.rpt")
> ENgetlinkvalue(2, "EN_LENGTH")
[1] 5280
> ENsetlinkvalue(2, "EN_LENGTH", 6789)
> ENgetlinkvalue(2, "EN_LENGTH")
[1] 6789
>ENclose()
```

3.3. Model calibration by univariate optimization

Consider a basic model calibration problem: EPANET's example network 1, which is included with the package, was operated under a high demand condition with a view to estimating the pipe roughness. Conditions were typical for the time 00:00 except that a demand of 2000 gallons per minute was induced at node 23 (index 7). Under that condition, pressure measurements of 112.11, 110.87 and 110.32 psi were collected at nodes with indices 4, 6, and 8.

Listing 5 shows one way of estimating pipe roughness assuming the same roughness value applies to all pipes in the network. First, ENopen initializes EPANET and processes network information. Then, ENsettimeparam and ENsetnodevalue update the network to the conditions of this example. Only one time period is considered and the demand at one node has been changed. Next, the optimize function built into R is used to find a value of pipe roughness that minimizes the sum of squared errors between the measured and modeled pressures. The roughness value, in this case from the Hazen-Williams formula, is constrained to fall in the interval between 50 and 150. The observed pressures and their node indices are passed through to the objective function by optimize. Finally, the EPANET engine is closed with a call to ENclose.

Listing 6 provides the user-defined functions calibObj, setAllPipeRoughness and sse used to implement the objective function for pipe roughness calibration in this example. The function calibObj calls the helper function setAllPipeRoughess to change the roughness value of all pipes in the network to the provided value. setAllPipeRoughness loops over all links in the network, only updating the roughness if the link is of type pipe as opposed to pump or valve. ENsolveH carries out a full hydraulic simulation. The function sse computes the sum of squared error between measured and modeled pressures.

Running this example yields a C-value of 131 and an objective value of 0.88. Note in an R session listing 6 should be run before listing 5 so that the objective function is available to the optimizer.

Listing 5. Pipe roughness calibration with univerate optimization

```
> ENopen("Net1.inp", "Net1.rpt")
> ENsettimeparam("EN_DURATION", 0)
> ENsetnodevalue(7, "EN_BASEDEMAND", 2000)
> optimize( calibObj,
>           interval = c(50, 150),
>           obsindex = c(4,6,8),
>           obs = c(112.11, 110.87, 110.32)  )
> ENclose()
```

Listing 6. Objective function for pipe roughness calibration

```
calibObj <- function( roughness, obsindex, obs){
  setAllPipeRoughness( roughness )
  ENsolveH()
  sse( obsindex, obs)
}

setAllPipeRoughness <- function( roughness ){
   for(i in 1:ENgetcount("EN_LINKCOUNT")){
       if( ENgetlinktype(i) < 2)
         ENsetlinkvalue(i, "EN_ROUGHNESS", roughness )
   }
}

sse <- function( nodeIndex, measuredPressure){
    N <- length(nodeIndex)
    error <- rep(NA, N)
    for( i in 1:N){
       modeledPressure <- ENgetnodevalue( nodeIndex[i],
                                          "EN_PRESSURE")
       error[i] <-  modeledPressure - measuredPressure[i]
    }
    sse <- sum( error * error )
}
```

3.4. Stochastic simulation

Stochastic simulation is often of interest in the context of modeling hydraulics under water demand forecasting scenarios. Again using example network 1, consider a problem where the water utility is interested in building a demand forecasting model and simulating the water network behavior under forecasts produced by this model.

The total water demand in the network measured at hourly intervals for a period of approximately one week is available in a comma-separated values file named data.csv. The first line of this file contains the headings Time and Measurement; the subsequent lines contain the corresponding timestamps and values of the measurements.

A seasonal ARIMA$(0,1,4) \times (0,1,1)_{24}$ model is considered appropriate (Arandia et al., 2016) and fitted to the data. Listing 7 shows how the measurement data is read into a data frame using the read.csv function provided by the base R distribution and how a demand forecast for the next 24-h period is calculated using the sarima.for function of the astsa package (Stoffer, 2017), which must be previously installed. A new demand pattern newpattern is computed by dividing the forecasts by their mean value.

In order to update the EPANET input file with the calculated new pattern, the toolkit is opened to work on network 1 with ENopen, the pattern time interval is set to 3600 s with ENsettimeparam, the existing time pattern is replaced by newpattern using ENsetpattern, the input file is saved with ENsaveinpfile, and finally the toolkit is closed with ENclose. The result is a modified file Net1-forecast.inp that can be readily used in hydraulic simulation.

Listing 7. Stochastic simulation using water demand forecasts

```
> library(astsa)
> data <- read.csv("data.csv")
> forecast <- sarima.for(data$Measurement,
                         n.ahead = 24,
                         p = 0, d = 1, q = 4,
                         P = 0, D = 1, Q = 1,
                         S = 24)
> newpattern <- as.numeric(forecast$pred /
                           mean(forecast$pred))
> ENopen("Net1.inp", "Net1.rpt")
> ENsettimeparam("EN_PATTERNSTEP", 3600)
> ENsetpattern(1, newpattern)
> ENsaveinpfile("Net1-forecast.inp")
> ENclose()
```

3.5. Test suite

As part of the development process, a collection of tests was created to verify the behavior of each R function provided by epanet2toolkit. These tests take the form of examples included with the package manual and as a suite of tests for use with a testing framework. Examples are accessed through the help system for each function (e.g. ?ENepanet, or help (ENepanet)). They are designed to run interactively and illustrate usage. In contrast, the test suite is designed to run automatically, thus making it easier to detect changes that alter package behavior and to document bugs and fixes.

Tests include rudimentary behavior checking for functions and implementations of the sample applications described in the documentation for the EPANET programmer's toolkit (Rossman, 2008). The sample applications are located in the 'tests/testthat' directory under the file names 'test_epanet_example_2.r' and 'test_epanet_example_3.r' Because the tests were written to verify the performance of epanet2toolkit roughly one third are specific to the R environment. The remaining tests represent a publicly available test suite for the EPANET API and thus they may prove useful in detecting changes or bugs in future versions of the simulation engine. Provided tests have been installed as shown in Listing 3, the test_package function from testthat (Wickham, 2011) runs the suite of tests (Listing 8). Here the test reporter has been specified to generate more verbose output.

Listing 8. Running the package test suite

```
> library(testthat)
> test_package("epanet2toolkit",
               reporter=default_reporter())
```

4. Conclusions

This short communication has described epanet2toolkit, a package for making EPANET simulations in the R environment. With the package, two commands are needed to download, compile (if required), and load the EPANET simulation engine into the R environment. These steps work on Windows, Mac OS, and Linux systems. Functions provided by the package map directly to functions in the EPANET API, enabling developers to port applications into and out of R. Standard EPANET simulations can be invoked with a single function and customized applications can be developed to leverage the visualization and analysis capabilities of R including other packages in the R ecosystem.

Advanced users of epanet2toolkit should keep in mind a few known weak points with version 0.2.1. An R object wrapping an EPANET simulation is not provided so it is only possible to run one simulation at a time within an R session. In its current version epanet2toolkit provides 55 of the 74 functions exposed by version 2.1 of EPANET's C API. Most functions not yet supported in R relate to the curves and demands sections on the inp file. Future work on the package could address these areas.

By improving accessibility of water network simulations, epanet2toolkit aims to enable more robust data-driven decision making for these critical infrastructure systems. epanet2toolkit is available under the MIT license and welcomes contributions from third party developers via GitHub github.com/bradleyjeck/epanet2toolkit. The GitHub page Issues tab is the place to raise questions about the package including bugs and proposed enhancements.

References

Arabzadeh, R., Aber, P., Panaghi, K., Araghinejad, S., Montaseri, M., 2017. WRSS: Water Resources System Simulator. R package version 1.0. https://CRAN.R-project.org/package=WRSS.

Arabzadeh, R., Aberi, P., Panaghi, K., Araghinejad, S., Montaseri, M., 2016. RSSOP: Simulation of Supply Reservoir Systems Using Standard Operation Policy. R package version 1.1. https://CRAN.R-project.org/package=RSSOP.

Arandia, E., Ba, A., Eck, B., McKenna, S., 2016. Tailoring seasonal time series models to forecast short-term water demand. J. Water Resour. Plann. Manag. 142 (3), 04015067.

Eck, B.J., 2016. An r package for reading epanet files. Environ. Model. Software 84, 149–154. http://www.sciencedirect.com/science/article/pii/S1364815216302870.

Eliades, D.G., Kyriakou, M., Vrachimis, S., Polycarpou, M.M., 2016. Epanet-matlab toolkit: an open-source software for interfacing epanet with matlab. In: Computing and Control for the Water Industry, 14th International Conference, . https://github.com/KIOS-Research/CCWI2016/blob/master/CCWI2016/Paper/Eliades2016.pdf.

Hirsch, R.M., De Cicco, L.A., 2015. User Guide to Exploration and Graphics for RivEr Trends (EGRET) and DataRetrieval: R Packages for Hydrologic Data. U.S. Geological Survey, Reston, VA Ch. A10. http://pubs.usgs.gov/tm/04/a10/.

Jassby, A.D., Cloern, J., 2015. wq: Exploring Water Quality Monitoring Data. R package version 0.4.5. http://CRAN.R-project.org/package=wq.

Leutnant, D., 2017. swmmr: R Interface for US EPA's SWMM. R package version 0.7.0. https://CRAN.R-project.org/package=swmmr.

Pathirana, A., 2016. Epanettools 0.9.2. https://pypi.python.org/pypi/EPANETTOOLS/0.9.2.
R Core Team, 2013. R: a Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. http://www.R-project.org/.
Rossman, L., 2008. Epanet 2 Programmer's Toolkit Files. https://www.epa.gov/sites/production/files/2014-06/en2toolkit.zip.
Rossman, L.A., 2000. Epanet 2 Users Manual. US EPA, Cincinnati, Ohio.
Stoffer, D., 2017. astsa: Applied Statistical Time Series Analysis. R package version 1.8. https://CRAN.R-project.org/package=astsa.
Turner, S., Galelli, S., 2016. Water supply sensitivity to climate change: an R package for implementing reservoir storage analysis in global and regional impact studies. Environ. Model. Software 76, 13–19.
Wickham, H., 2011. testthat: get started with testing. R J. 3, 5–10.