



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](https://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 epanet2toolkit'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).

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

Download English Version:

<https://daneshyari.com/en/article/6961922>

Download Persian Version:

<https://daneshyari.com/article/6961922>

[Daneshyari.com](https://daneshyari.com)