VOLUME 1

CONFLUENCE

Journal of Watershed Science and Management

VOLUME 1 | N°01 | CONFLUENCE
Journal of Watershed Science and Management

Why Watershed Analysts Should Use R for Data Processing and Analysis

R. Dan Moore & David Hutchinson

R. Dan Moore is Professor and Chair of Forest Hydrology in the Department of Geography at the University of British Columbia. Email: dan.moore@ubc.ca

David Hutchinson is a District Manager with the Water Survey of Canada Division at Environment and Climate Change Canada. Email david.hutchinson@canada.ca

Abstract

Both the science and practice associated with watershed management involve the processing, presentation and analysis of quantitative information. In this article, the use of open source programming languages by watershed analysts is advocated. The R language, in particular, provides a rich set of tools for the types of data that are commonly encountered in watershed analysis. The utility of R is illustrated through three examples: intensity-duration-frequency analysis of rainfall data, baseflow separation, and watershed delineation and mapping.

KEYWORDS: *data science; programming languages; reproducible analysis; visualization; watershed analysis*

Introduction

Both the science and practice associated with watershed management involve the processing, presentation, and analysis of quantitative information. The tools available to watershed analysts have evolved dramatically over the last few decades. When the senior author was an undergraduate student in the 1970s, science and engineering students learned to program in one of the dialects of FORTRAN on mainframe computers. Statistical analyses at that time were typically conducted using statistical functions within the International Mathematics and Statistics Library (IMSL) Numerical Libraries, called from a FORTRAN program, or by using a mainframe implementation of a statistical package such as SAS, SPSS, or BMDP.

In the 1980s, the advances in desktop personal computers changed the data analysis landscape. Programmers, including the senior author, moved from the clunky FORTRAN compilers then available for microcomputers to faster-compiling implementations of Pascal. Programs such as SYSTAT provided the capability to conduct sophisticated statistical analyses and generate complex graphs in a publication-ready quality format. Spreadsheet programs such as Lotus 123, Quattro, and later Microsoft Excel became the Swiss Army knives of data analysis, providing a broad range of functions for data processing, analysis, and graphing. Ongoing development of geographic information system (GIS) software revolutionized our ability to analyze and visualize spatial information. In particular, the ability to delineate catchment boundaries from digital elevation models (DEM) and extract land cover characteristics has been a boon to the science and practice of watershed analysis and management.

In our interactions with consultants and government scientists who are involved in watershed analysis, it appears that spreadsheets continue to dominate the state of practice, in conjunction with additional software products to perform analyses that are not supported by spreadsheets (e.g., GIS and statistical packages). Spreadsheets also remain in use in the academic community. In terms of programming languages, Matlab is popular in both the practitioner and academic communities, particularly where

Moore, R. Dan & Hutchinson, David. (2017). Why Watershed Analysts Should Use R for Data Processing and Analysis. http://confluence-jwsm.ca/index.php/jwsm/article/view/2. doi: 10.22230/jwsm2017v1n1a2.

numerical methods are required to solve equations. Matlab is a commercial product that facilitates high-level programming within a sophisticated graphical user interface (GUI), and in conjunction with add-on packages, can accommodate the full range of tasks typically performed by watershed analysts. The open-source programming language R has been adopted primarily by research-oriented watershed scientists and does not yet appear to have widespread uptake among practitioners.

The objective of this article is to argue for the expanded use of a programming language within the watershed science and management community. In particular, the use of the R language is encouraged. Although this article describes some specific functions and data types within R, it is not intended to be a hands-on introduction; however, pointers to useful resources for beginning and novice R users are provided.

Why Watershed Analysts Should Use a Programming Language

The use of a programming or scripting language facilitates automation of repetitive tasks and thus can reduce the time and cost associated with many watershed analyses. In addition, the use of programming or scripting languages is increasingly important for purposes of reproducibility of analyses. Sandve et al. (2013) noted the importance of reproducibility and transparency in the context of research by outlining 10 steps that should be followed to ensure reproducibility. For example, manual data manipulation should be avoided (e.g., infilling missing data by hand within a spreadsheet or copying and pasting data between spreadsheets). Although it is possible to edit data in a transparent manner within a spreadsheet—for example, by creating an additional column for the edited data series and leaving the original series unaltered—this approach is cumbersome and difficult to automate for repetitive application. Furthermore, it is simpler to audit code than to manually inspect multiple spreadsheets to ensure that data-infilling has been performed correctly. Reproducibility should be a high priority in practical applications as well as in research, particularly for controversial resource-related projects that may be the subject of environmental appeal processes. The recommended approach is to write a program or script that reads in the raw data, performs any required editing or manipulation, then saves the edited data in a new file for further processing and analysis, thus preserving the integrity of the original data set.

Many statistical packages, including SAS, JMP, SPSS, Minitab, and SYSTAT, have powerful scripting capabilities and thus facilitate reproducibility and ease of automation. These packages provide access to sophisticated statistical procedures and have established credibility in government, industry, and academia. Most also provide functions for statistical graphics and even mapping and spatial statistics. However, their main limitations for general purpose watershed analysis are their lack of support for numerical modelling (e.g., solution of differential equations) and GIS-related operations, such as digital terrain analysis.

An important consideration for many users is that commercial products like Matlab, ArcGIS, and SAS provide technical support and are committed to fixing bugs. However, despite these advantages of commercial software products, there has been a growing adoption of open-source software over the last two decades, especially in the data analysis and data science communities. Open-source programming languages have the advantage that a client does not need to purchase a licence for commercial software to modify code developed for them by consultants. Furthermore, open-source languages have an enthusiastic and productive community of developers and contributors, who produce a rich array of packages to accommodate a broad range of analyses. Note, however, that user-contributed packages can present potential pitfalls as discussed in the section Background Information on R.

A number of open-source languages provide access to a broad range of the analytical procedures required in watershed analysis within a single platform, which can streamline the workflow for complex projects and thus provide time and cost advantages over approaches that require data to be transferred among different software applications. Data analysts currently appear to favour the R, Python, and Julia languages. Although the authors advocate the use of R, it is acknowledged that Python, Julia, and other open-source languages can be viable alternatives and have distinct advantages over R for specific purposes. For example, Python has an easier-to-learn syntax than R, and is considered by many to be superior for development of applications and especially for server-level

Moore, R. Dan & Hutchinson, David. (2017). Why Watershed Analysts Should Use R for Data Processing and Analysis. http://confluence-jwsm.ca/index.php/jwsm/article/view/2. doi: 10.22230/jwsm2017v1n1a2.

operations (DataCamp Team 2016). Julia also has a simpler syntax than R, and stands out by virtue of its processing speed, which can rival those of Fortran and C on some benchmarks (Julia Language, n.d.). The strength of R derives from its rich capabilities in statistical analysis and visualization of data, especially for the time series and geospatial data that watershed analysts work with on a regular basis. The senior author and his research students have found that R, in conjunction with the open-source *SAGA GIS* package, provide all the functionality required for almost all of their watershed-related research needs.

Hutton et al. (2016) called for the hydrological community to move toward integrated and reproducible workflows through the use of reusable code to increase the credibility and integrity of research results. The use of an open-source programming language, like R, which allows integration of a complete workflow—from data compilation and processing to analysis or simulation, and then to reporting—can play a critical role in this process.

Background Information on R

R was developed by Robert Gentleman and Ross Ihaka of the University of Auckland. It was based on the S language developed by John Chambers at Bell Laboratories, which formed the basis of the commercially developed S-PLUS package. Much of the code written under S will run with no changes in R.

R is a Gnu's Not UNIX (GNU) project, and is available for free under the GNU General Public License. R is currently developed and maintained by the R Development Core Team. For more information, refer to the R project home page (http://www.r-project.org/).

The R language is built around a set of packages, which are collections of functions and defined data types to perform specific tasks. Packages can also contain data sets that can be used to test or demonstrate R programs without reading in data from an external file. The default installation, often called "base R," includes more than a dozen packages, including *base* (which provides data types and functions for fundamental operations such as file input/output, calculations, looping, and conditional execution), *stats* (with functions for descriptive statistics, linear and nonlinear statistical models, time series analysis), *graphics* (data visualization), and *datasets*. The *datasets* package contains a variety of data sets, such as a digital elevation model of Auckland's Maunga Whau volcano, passenger miles on commercial U.S. airlines, and speed and stopping distances of cars.

A strength of R is that a plethora of user-contributed packages is available to provide access to a rich variety of functions and data types for data processing, analysis, and visualization. However, this strength can also pose potential pitfalls. For example, in many cases, more than one package may provide similar functions, and there is no clear authority to provide guidance as to which package is superior. Also, there is no systematic process for verifying the accuracy and reliability of user-contributed packages. Even base R has known issues, as listed in the "R inferno" website.

R as a Programming Language

The R language supports conditional execution (*if* statements), loops (*for, repeat,* and *while* statements), and functions. The syntax of R is somewhat like C; for example, it uses curly braces ("{" and "}") to indicate the beginning and end of a block of statements. One limitation to the use of R as a general-purpose programming tool is that it is an interpreted language and thus will not execute as quickly as a compiled language like Fortran or C++. In particular, R code that involves loops runs slowly.

In many cases, the use of loops can be avoided by taking advantage of "vectorized" code. For example, suppose an analyst had streamflow time series measured at two locations along a stream, and wanted to compute the incremental contribution to discharge between the two locations. One approach would be to "loop" through all time intervals, compute the difference at each time interval, and then store it in the appropriate location in a vector, as in the following code fragment:

```
for (i in 1:n) Qinc[i] <- Qds[i] – Qus[i]
```


Moore, R. Dan & Hutchinson, David. (2017). Why Watershed Analysts Should Use R for Data Processing and Analysis. http://confluence-jwsm.ca/index.php/jwsm/article/view/2. doi: 10.22230/jwsm2017v1n1a2.

where "i" indexes the time interval, "n" is the number of time intervals, "Qinc" is the name of the vector holding the computed discharge increment, "Qds" and "Qus" are the names of vectors containing the downstream and upstream discharge series, and "<-" is an assignment operator. An equal sign ("=") can also be used for assignment but is not favoured by the general R community for historical reasons. An alternative, much faster approach is to code the operation as follows:

```
Qinc <- Qds – Qus
```

where the operation on the right-hand side is applied element-wise within the vectors.

Many analyses involve what data scientists call "split-apply-combine" operations (Wickham 2011). For example, a watershed analyst may have daily rainfall data for a number of years at a number of stations, and may wish to compute annual summaries (e.g., totals or maxima) for each station-year. One approach would be to write code that "loops" over each year and then over each station, computing the summary value within each iteration and storing it in a variable. A faster, more elegant and easier-to-read approach is to use the *aggregate()* or one of the *apply()* functions within base R or one of the more recently developed functions in the contributed *plyr* and *dplyr* packages. These functions apply a built-in or user-defined function to each defined subset within a data set, with the output being a data object that contains the results.

In some cases, the use of loops is unavoidable. For example, when computing a soil moisture balance through time, the calculation at a given time step involves the soil moisture at the end of the preceding time step, as in the example code:

```
for (i in 2:n) SM[i] <- SM[i-1] + P[i] – ET[i] – D[i]
```

where SM[i] is the soil moisture content at the end of time interval "i" and P[i], ET[i], and D[i] are the precipitation, evapotranspiration, and drainage during that interval. In these cases, options for speeding up R code are to use the *compiler* package, which compiles code at run-time, or to use the *inline* package, which allows the inclusion of code written in C, C++, or Fortran within the R code. The *foreach* package provides functions for looping that can be faster than standard *for* loops. If you are using a computer with multiple processors, *foreach* can significantly reduce run-time execution by supporting parallel execution of *for* loop statements.

R for Statistical Analysis and Modelling

The *stats* package in base R includes a broad range of built-in functions for both descriptive and inferential statistics. For descriptive statistics, commonly used functions include *mean( ), sd( )* (standard deviation), *fivenum( )* (minimum, lower hinge, median, upper hinge, maximum), and *quantile( )* (to compute specified percentiles). The *stats* package also has a rich set of functions for statistical modelling. For example, the *lm( )* function can fit a linear model for a numeric response variable based on one or more predictor variables, which can be numeric, factor, or a combination. Thus, *lm( )* can be used for linear regression (simple and multiple), analysis of variance, and general linear modelling. There are also functions that facilitate more advanced statistical modelling, such as nonlinear least squares regression via the *nls( )* function and generalized linear modelling via the *glm( )* function. Generalized linear models can accommodate response variables that are not normally distributed, and include logistic regression, in which the quantity to be predicted is a binary variable, such as the presence/absence of an aquatic species.

The *stats* package also contains useful functions for time-series analysis. The *acf( )* and *pacf( )* functions can generate autocorrelation and partial autocorrelation plots for a single time series, and the *ccf( )* function generates cross-correlation functions for a pair of time series. The *arima( )* function allows time series to be modelled as autoregressive integrated moving average processes. In addition to the modelling of a univariate time series, the *arima( )* function can accommodate external regressor variables, so the function can be used to fit regression models that can address issues like autocorrelated residuals, which violate the assumptions underlying ordinary least squares regression (e.g., Guenther et al. 2014).

Moore, R. Dan & Hutchinson, David. (2017). Why Watershed Analysts Should Use R for Data Processing and Analysis. http://confluence-jwsm.ca/index.php/jwsm/article/view/2. doi: 10.22230/jwsm2017v1n1a2.

In addition to packages in base R, a plethora of user-contributed packages contains functions for just about any form of analysis you could imagine. For example, the *rpart* package contains functions for fitting classification and regression tree models. This form of model is useful when the phenomenon to be predicted is a categorical variable and the predictor variables are numeric. The *lme* and *nlme* packages support the application of mixed-effects models, which can accommodate spatial and/or temporal autocorrelation in the residuals, both of which commonly occur in environmental data sets.

It is important for users to be aware that many statistical functions in R will return results that are different from those generated by a statistical package like SAS. For example, when using the *lm()* function for analysis of variance, R returns what are called Type I or incremental tests rather than Type III or marginal tests, as generated by SAS. In particular, different results will be obtained using *lm()* depending on the order in which the factors are listed in the model specification. Another example is the convention by which the *arima()* function returns the estimated mean as the estimated intercept. It is fundamental that, no matter what statistical package is being used, the onus is on the user to understand how the package performs and reports a test, and to ensure that analyses are undertaken and interpreted correctly for each specific application. When moving to R from another analytical software package, it is recommended to run parallel analyses in both to ensure consistency of results.

R for Data Visualization and Graphical Presentation

Three main packages are available for graph production within R: the default *graphics* package (commonly called base graphics), *lattice* graphics, and *ggplot2*. All can generate standard graphs such as histograms, scatterplots, and contour plots, as well as a broad range of more complex graphs, and all allow a high level of customization. However, the packages differ fundamentally in their underlying approaches.

The base graphics package applies a traditional pen-on-paper plotting strategy, in which commands are used to draw specific sets of points, lines, shapes, or text in specific locations within the plotting frame. Even for a single graph panel, several function calls are often required to create a complete graph with legends, especially when multiple data series are plotted.

In *lattice* graphics, a plot is generated using a single function call, with all customization handled by parameters within the call. A powerful feature is the ability to generate "conditioning" plots to explore how a relation between two variables or a frequency distribution varies among subsets. For example, one could generate histograms of daily rainfall or a plot of air temperature versus elevation by month of the year.

The *ggplot2* package is based on the "grammar of graphics" concept developed by Wilkinson (2005). Rather than building a graph based on individual graphic components, as in base graphics, *ggplot2* builds a graph in layers with function calls operating at a high level of abstraction. To build a plot, the user specifies a data frame (i.e., the data set to be used), aesthetic mappings (e.g., which variables correspond to the *x* and *y* axes, and the use of colour and/or size of symbols to represent additional variables), delineation into facets (for plots conditioned by subsets), geoms (points, lines, and shapes to be plotted), stats (transformations, including binning for histograms), and scales (e.g., relating colours to values of a variable). The *ggplot2* package is particularly useful for exploring multivariate relations within a data set.

R for Computation, Numerical Analysis, and Simulation

R is probably best known for its statistical and graphing routines, but it also includes a broad range of functions for general computation and numerical analysis and simulation. In particular, it provides a comprehensive set of functions for arithmetic, trigonometry, logarithms, and probability distributions, and for matrix algebra, such as matrix inversions.

Base R contains functions for random number generation, root finding, and optimization. User-contributed packages also provide functions for solution of differential equations. For example, the *deSolve* package provides solvers for initial value problems that involve ordinary differential equations, partial differential equations, differential algebraic equations, and delay differential equations, whereas *bvpSolve* can be used to solve boundary-value problems for ordinary differential

Moore, R. Dan & Hutchinson, David. (2017). Why Watershed Analysts Should Use R for Data Processing and Analysis. http://confluence-jwsm.ca/index.php/jwsm/article/view/2. doi: 10.22230/jwsm2017v1n1a2.

equations. Leach & Moore (2015) programmed a coupled hydro-thermal model for headwater catchments in R by framing the model as linked sets of differential equations. The model setup, pre-processing, and post-processing were all coded in R, with a call within the R script to a *deSolve* function that solves the model differential equations, which were coded in Fortran (but note that *deSolve* can also link with C or C++).

Data Input and Output

In base R, various functions can read in data from several formats. For data stored in text files, relevant functions include *read.table( ), read.csv( ),* and *read.fwf( )* (the latter for fixed-width formats). The result of these functions is a rectangular object called a data frame, in which each variable is a column and each row is a case. Within a column (variable), all values should be the same type, either numeric, character, or factor. Numeric variables include both integers and real numbers; character variables are character strings (such as sampling site name); and factors are categorical variables, which could be coded either as numeric (e.g., site number) or as a character (e.g., site name). Users should be aware that, unless specified otherwise, any column in which any of the entries is non-numeric will be interpreted as a factor variable when using the base R functions for reading files. In some cases, this can create problems if the user tries to treat it as a character. The solution is to coerce the variable to a character using the *as.character()* function.

Watershed analysts frequently work with time series. When reading in a data set, dates and date-time variables can be coded for input in a variety of ways. One is to code each part (e.g., year, month, day, hour, minute, second) as a separate numeric variable, then create an International Standards Organization (ISO) date-time variable using the *ISOdatetime()* function. Alternatively, one could code a date-time variable as a character string such as "2005-12-4 11:00:00" and then create an ISO date-time variable using the *strptime()* function. In addition to the date-time functions in base R, the *lubridate* package provides a consistent set of functions for easier handling and manipulation of date-time variables.

Functions are available in base R or in user-contributed packages that allow data to be read from a broad number of specialized file formats, including Excel spreadsheets (as CSV files), GIS shape files, files from statistical packages (including SAS, SPSS, and SYSTAT), and relational databases. For example, the *RMySQL* package provides functions that allow R to connect to and extract data from a MySQL database. The *XLconnect* package provides functions to allow users to read and write Microsoft Excel spreadsheets. R also supports data input from NetCDF files, which allows access to large-scale, gridded atmospheric data sets. In addition, R can output data in a number of formats that correspond to the input file formats used by a range of software packages.

Spatial Analysis and Mapping

Hydrologists rely heavily on spatial data sets and GIS analyses, and R can be an important part of the spatial analysis toolkit. For example, the *sp* package provides data classes and methods for handling, processing, and displaying spatial data; the *raster* package facilitates analysis and visualization of gridded spatial data, such as raster GIS layers, remote sensing imagery, and digital elevation models; *maptools* supports the reading, writing, manipulation, and display of shapefiles; and *gstat* provides functions for geostatistical analysis, such as fitting of variograms and spatial interpolation via inverse-distance weighting and kriging. These are just a sample of some of the more popular R packages that are available for spatial data analysis.

Packages are available for working with internet mapping applications. For example, the *ggmap* package provides a powerful interface to the OpenStreet™ and Google Maps™ application programming interface, which allows users to use geospatial data sets within R as part of their data visualizations. The *plotKML* package supports the plotting of spatial data in Google Earth by creating *.kml and *.kmz files.

Looking beyond R, packages have been developed to leverage the power of other languages and applications. For example, the *RPyGeo* package provides access to (virtually any) ArcGIS geoprocessing tool from within R by running Python geoprocessing scripts without writing Python code or touching ArcGIS (although both Python and ArcGIS must be installed on the user's system). The *spgrass6* package provides a similar way to run the GRASS GIS from within R. Many watershed scientists use

Moore, R. Dan & Hutchinson, David. (2017). Why Watershed Analysts Should Use R for Data Processing and Analysis. http://confluence-jwsm.ca/index.php/jwsm/article/view/2. doi: 10.22230/jwsm2017v1n1a2.

the *SAGA GIS* application, which is an open-source raster-based GIS written by geoscientists (Conrad et al. 2015). It is fast and powerful, and includes robust routines for watershed delineation. Use of the *RSAGA* package allows *SAGA* commands to be run from within an R script, thereby facilitating automation. A script for using R and RSAGA to delineate watershed boundaries is available at http://ibis.geog.ubc.ca/~rdmoore/Rcode.htm.

Hydrologic Analysis and Modelling

Table 1 provides brief descriptions of a sample of packages that are useful for conducting hydrologic analyses. Rigon (n.d.) provides links to and descriptions of many of these packages.

Table 1. Brief descriptions of a sample of contributed R packages of particular interest to hydrologists.

Package Name	Examples of functions provided
EcoHydRology	Hydrograph separation; functions to assist in setting up and calibrating the SWAT2005 hydrologic model; calculation of solar and longwave radiation, evapotranspiration, and surface energy balance; a model to simulate snowpack water equivalent at a point; a lumped model to simulate streamflow based on the saturation-excess runoff generation mechanism
hydroTSM	Management, analysis, interpolation, and plotting of time series used in hydrology and related environmental sciences; this package is highly oriented to hydrological modelling tasks.
TUWmodel	Lumped catchment hydrology model based on the Swedish HBV model
nsRFA	Tools to perform unsupervised regional frequency analysis of hydrologic data using the index-value method
seas	Tools for analyzing and visualizing seasonal variations in environmental time series data; functions for parsing Canadian Climate Centre data files
Lmoments	Estimates L-moments and trimmed L-moments from data
evd	Functions for analysis of extreme value distributions
lfstat	Functions for low-flow analysis based on World Meteorological Organization Operational Hydrology Report No. 50 (1999); functions for low-flow frequency analysis, recession analysis, flow duration curves, and baseflow separation
spei	Calculation of standardized drought indices; functions for computing potential evaporation based on Hargreaves, Penman, and Thornthwaite methods
waterData	Access to U.S. Geological Survey daily flow data and anomaly calculation
extRemes	Tools for extreme value analysis

Although it is possible to create time-stepping hydrologic models within R (e.g., Moore et al. 2012; Leach & Moore 2015), the R programming language is not an ideal model-coding platform, except perhaps for testing prototype code. A better approach would be to work with a flexible modelling platform such as the Raven hydrological modelling framework (Craig et al. 2015) or Cold Regions

Moore, R. Dan & Hutchinson, David. (2017). Why Watershed Analysts Should Use R for Data Processing and Analysis. http://confluence-jwsm.ca/index.php/jwsm/article/view/2. doi: 10.22230/jwsm2017v1n1a2.

Hydrological Model (CRHM) (Pomeroy et al. 2007), using R to pre-process spatial and temporal data for model input. For example, the *WATCHr* package generates meteorological forcing data files for input to CRHM based on gridded re-analysis data products (Shook 2015).

R can also be used as a "wrapper" to automate model setup and execution. It can be especially useful for model calibration within a generalized likelihood uncertainty estimation (GLUE) framework, in which a model is run multiple times with different parameter sets, with all sets that achieve a specified performance standard being adopted as part of a "behavioural" family. For example, Jost et al. (2012) used R as a wrapper to conduct a GLUE-type calibration of the HBV-EC hydrological model for the Columbia River catchment upstream of the Mica Dam.

R for Reporting and Presentation

For users of the LaTeX document preparation application, the sweave function within the base installation allows users to embed R code within a LaTeX source file. The R code is run each time the LaTeX document is compiled. Thus, for example, rather than inserting a figure as a graphic file or a table as a text object, one inserts the R code for creating the figure or table, which allows for automated updating of documents if data or analyses change. *Sweave* can also be used within OpenOffice documents and HTML files but not Word documents.

Another valuable reporting tool is R Markdown, which is a markup language in which content, formatting codes, and R code are incorporated into text files. Unlike markup languages like HTML, in which formatting is controlled by tags inserted into the text, R Markdown uses less obtrusive formatting codes that are more intuitive and easier to learn. Like *sweave,* R Markdown allows the creation of dynamic documents, in which graphs and tables are updated automatically when the document is processed. R Markdown allows the generation not only of pdf and OpenOffice documents, but also Word documents, HTML files, slide presentations, and other formats.

How to Get Started

The R software can be downloaded via the Comprehensive R Archive Network (CRAN) (https://cran.r-project.org/). In addition to the software, CRAN provides access to manuals, FAQs, and the R Journal.

Many resources that are suitable for beginner and novice R users are available online and through textbooks. A free online interactive learning tutorial for R is available at O'Reilly's Code School website called Try R (http://tryr.codeschool.com/). There is also an offline course for learning R programming and data science using the swirl R package (http://swirlstats.com/). The R Core Team also maintains an introductory manual to discuss the basic syntax, graphics, and architecture of R (Venables et al. 2016). Online sites, such as R-bloggers (https://www.r-bloggers.com), keep users current with news and tutorials from more than 600 contributing authors.

Perhaps the most important thing for novice users to familiarize themselves with is the R code style guide and conventions. The inherent elegance and flexibility of R can also be its main weakness. Novice users can develop poor habits that can make it difficult to interpret program logic or operations and share code with others. The authors recommend that users review Hadley Wickham's Advanced R style guide (http://adv-r.had.co.nz/Style.html) and Google's R style guide (https://google.github.io/styleguide/Rguide.xml) to familiarize themselves with typical R programming conventions and to establish a consistent programming style.

The most popular way to work with R is within R Studio (www.rstudio.org), which is an integrated development environment for R that is available for most operating systems. There are other solutions for working with R using text editors (such as Notepad++, emacs, sublime, or vi), but unless users have a proficiency and love for text editors, the authors recommend using R Studio over other solutions. R Studio supports the use of R Markdown.

Inevitably, users will encounter a bug or error for which no immediate solution can be found. Sites such as Stack Overflow (http://stackoverflow.com/questions/tagged/r), R Seek (http://rseek.org/), and Google can be used to search the World Wide Web for solutions. The R community supports mailing lists (https://www.r-project.org/mail.html), where users can post questions and receive help

Moore, R. Dan & Hutchinson, David. (2017). Why Watershed Analysts Should Use R for Data Processing and Analysis. http://confluence-jwsm.ca/index.php/jwsm/article/view/2. doi: 10.22230/jwsm2017v1n1a2.

from a very supportive user community. Stack Overflow provides similar functionality. With all posts to an online community for help, it is important to be concise, include reproducible examples, and use an informative subject line regarding the question at hand.

Examples

Three examples are provided to illustrate applications of R for watershed analysis. The associated Figures 1, 2, and 4 were generated using *base* graphics; Figure 3 was generated using *ggplot2*. The code for these examples can be accessed at http://www.geog.ubc.ca/~rdmoore/Rcode.htm

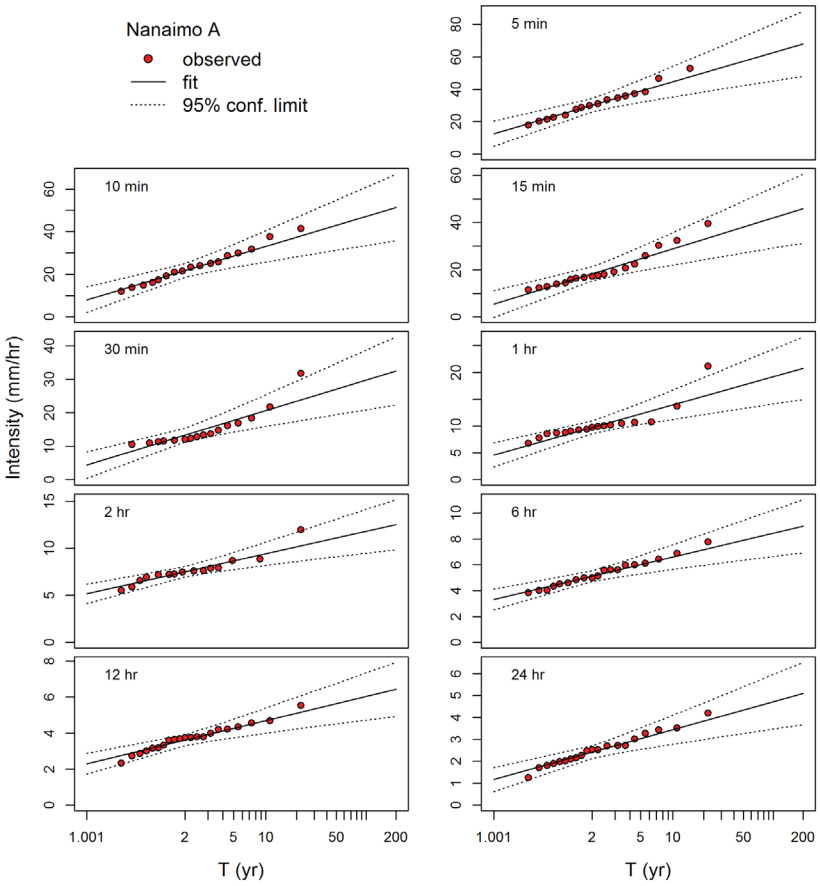

Figure 1. Frequency analyses of annual extreme rainfall intensities at Nanaimo Airport weather station (Nanaimo A) for durations ranging from 5 minutes to 24 hours. T is the return period. Fitted lines and confidence limits were computed using the method of moments.

Intensity-Duration-Frequency Analysis of Rainfall Data

Hydrologists are often called upon to analyze extreme rainfall events. The standard approach is to compute annual extreme rainfall intensity series for a range of durations, from as low as 5 or 10 minutes up to 24, 48, or 72 hours. For each duration, a Gumbel Extreme Value Type I distribution is fitted to the annual extreme series to estimate the intensities associated with specific return periods, typically 2, 5, 10, 20, 50, 100, and 200 years. The distributions for each return period are typically plotted on special "Gumbel paper" on which the abscissa is scaled so that data drawn from a Gumbel distribution should fall approximately along a straight line. It is relatively straightforward to generate such a plot within R. It is also common to summarize the full intensity-duration-frequency (IDF) analysis on a single graph with double-logarithmic axes; however, the abscissa, which represents the duration, has a change in units, from minutes for shorter durations to hours for longer durations. Again, it is relatively straightforward in R to format such an axis. Figures 1 and 2 illustrate the results of an IDF analysis for the Nanaimo Airport weather station on Vancouver Island.

Moore, R. Dan & Hutchinson, David. (2017). Why Watershed Analysts Should Use R for Data Processing and Analysis. http://confluence-jwsm.ca/index.php/jwsm/article/view/2. doi: 10.22230/jwsm2017v1n1a2.

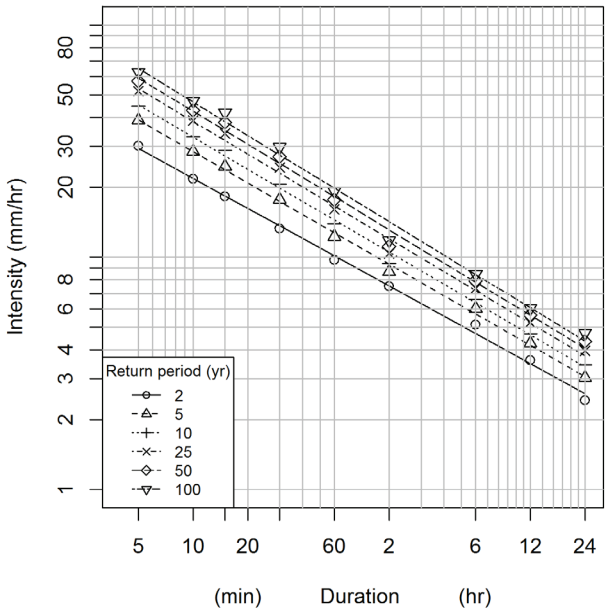

Figure 2. Intensity-duration-frequency analysis of annual extreme rainfall intensities at Nanaimo Airport.

Baseflow Separation

In both research and practical applications, hydrologists often need to separate a streamflow hydrograph into baseflow (slow-response) and stormflow (fast-response) components. Several methods are available to accomplish baseflow separation. Nathan & McMahon (1990) developed a robust hydrograph separation method based on the application of a high-pass filter, which has been implemented in the *BaseflowSeparation()* function in the *EcoHydRology* package. For the example shown in Figure 3, daily streamflow data for Newhalem River near Foss, Oregon (station identification number 14301000) were extracted from the U.S. Geological Survey National Water Information System (NWIS) web services. The *waterData* package provides easy-to-use routines to extract and clean streamflow data from NWIS.

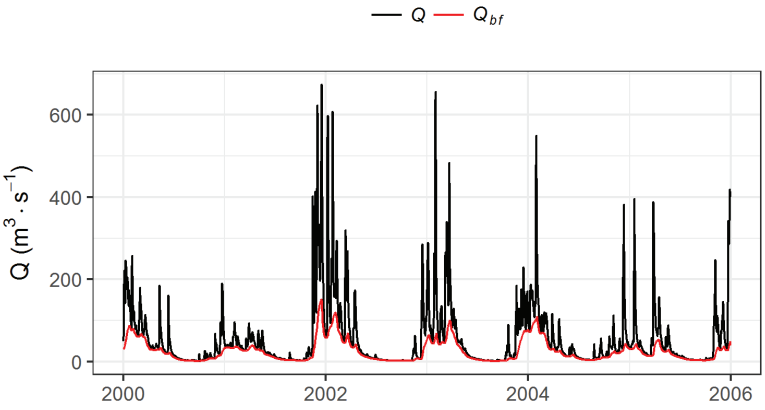

Figure 3. Baseflow separation of a daily streamflow record. The black line is streamflow; the red line is the baseflow.

Watershed Delineation and Mapping

Watershed delineation is one of the few functions that does not have a robust implementation within R. However, it is possible to use R and *RSAGA* to process a digital elevation model and generate catchment boundaries. In the example in Figure 4, the catchment boundaries were generated using *RSAGA,* and all other processing and mapping was performed using the *raster* and *sp* packages. The DEM and shapefiles were downloaded from Natural Resources Canada's GeoGratis site (http://geogratis.gc.ca/site/eng/extraction).

Moore, R. Dan & Hutchinson, David. (2017). Why Watershed Analysts Should Use R for Data Processing and Analysis. http://confluence-jwsm.ca/index.php/jwsm/article/view/2. doi: 10.22230/jwsm2017v1n1a2.

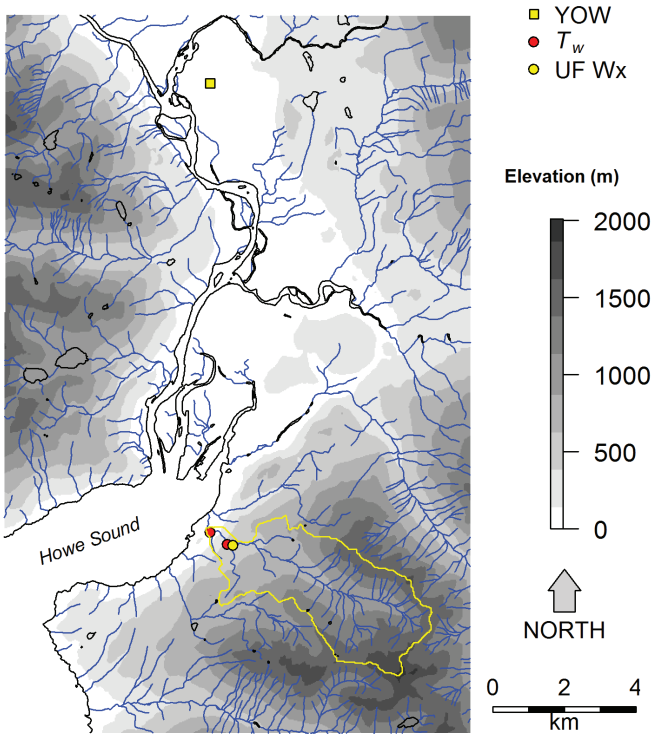

Figure 4. Map of the area around Squamish, B.C., including the catchment boundary for a stream temperature monitoring site below Shannon Falls (yellow line), as well as locations of temperature monitoring sites above and below the falls (T_w) (red circles), location of a weather station located above the falls (UF Wx) (yellow circle), and Squamish Airport (YOW) (yellow square).

Summary: Pros and Cons of R for Watershed Analysts

In summary, R has many compelling features that make it a valuable tool for research and application in watershed analysis. In fact, many analysts find that they can manage their entire workflow within the R environment, from data import and processing to analysis, graphing, and mapping, and finally to generation of reports and presentations. Specific features of value in watershed analyses include the following:

- access to a rich variety of statistical procedures, from the common to the exotic;
- ability to generate almost any kind of graph or map with a high level of control over axes, plotting symbols, text, and placement;
- flexibility in the representation of time for both calculation and graphing;
- access to a rich set of packages for numerical analysis;
- implementation of standard programming concepts (loops, conditional execution, and functions) to simplify the automation of repetitive tasks;
- ability to import and export data in a broad variety of formats;
- availability of packages that are customized for watershed analysis applications;
- availability of relatively sophisticated GUIs to assist with code development, maintenance, and debugging;
- ability to write your own custom packages and contribute to the R community through CRAN;
- ability to integrate foreign language code for faster processing (e.g., C, Fortran); and
- ability to build web-based visualizations and data analyses.

In addition, R is freely available, including all the packages, and there is no need to pay for extra add-ons.

The major downside to R is the learning curve and the required investment of time to learn how to use it. After more than 20 years of open-source development worldwide and thousands of

Moore, R. Dan & Hutchinson, David. (2017). Why Watershed Analysts Should Use R for Data Processing and Analysis. http://confluence-jwsm.ca/index.php/jwsm/article/view/2. doi: 10.22230/jwsm2017v1n1a2.

contributed packages, R can prove challenging to the novice user. Users of R must invest time to learn how to, and continue to, use it effectively.

With the increasing availability of "big data" sources such as remote sensing products and gridded atmospheric data sets, and the growing use of spatially distributed hydrologic simulation models, the watershed analysis community needs powerful and adaptive tools to manage, manipulate, and model the vast quantities of data with complex interrelationships that have become increasingly common in the geosciences. The authors argue that structured programming-based data analysis ensures an auditable workflow from data extraction to publication, and that the R language currently provides the functionality required to achieve this goal.

Acknowledgements

The authors gratefully acknowledge the comments provided by four anonymous reviewers, which resulted in significant improvement to the manuscript, as well as the copy editing by Tracey Hooper, which improved the English expression.

References

Conrad, O., B. Bechtel, M. Bock, H. Dietrich, E. Fischer, L. Gerlitz, J. Wehberg, V. Wichmann, & J. Böhner. 2015. System for automated geoscientific analyses (SAGA) v. 2.1.4. *Geoscientific Model Development 8:*1991–2007. DOI:10.5194/gmd-8-1991-2015

Craig, J.R., S. Huang, A. Khedr, S. Pearson, S. Spraakman, G. Stonebridge, C. Werstuck, & C. Zhang. 2015. *Raven: user's and developer's manual. Raven Version 2.1.* http://www.civil.uwaterloo.ca/jrcraig/Raven/Main.html (Accessed August 2016)

DataCamp Team. 2016. *Choosing R or Python for data analysis? An infographic.* http://www.datacamp.com/community/tutorials/r-or-python-for-data-analysis (Accessed August 2016)

Guenther, S.M., T. Gomi, & R.D. Moore. 2014. Stream and bed temperature variability in a coastal headwater catchment: influences of surface-subsurface interactions and partial-retention forest harvesting. *Hydrological Processes 28:*1238–1249. DOI:10.1002/hyp.9673

Hutton, C., T. Wagener, J. Freer, D. Han, C. Duffy, & B. Arheimer. 2016. Most computational hydrology is not reproducible, so is it really science? *Water Resources Research 52:*7548–7555. DOI:10.1002/2016WR019285

Jost, G., R.D. Moore, B. Menounos, & R. Wheate. 2012. Quantifying the contribution of glacier runoff to streamflow in the upper Columbia River basin, Canada. *Hydrology and Earth Systems Science 16:*849–860. DOI:10.5194/hess-16-849-2012

Julia Language. n.d. *Julia.* http://julialang.org/ (Accessed August 2016)

Leach, J.A., & R.D. Moore. 2015. Observations and modeling of hillslope throughflow temperatures in a coastal forested catchment. *Water Resources Research 51:*3770–3785. DOI:10.1002/2014WR016763

Moore, R.D., J.W. Trubilowicz,. & J.M. Buttle, 2012. Prediction of streamflow regime and annual runoff for ungauged basins using a distributed monthly water balance model. *Journal of the American Water Resources Association 48:*32–42. DOI:10.1111 / j.1752-1688.2011.00595.x

Nathan, R.J., & T.A. McMahon. 1990. Evaluation of automated techniques for base flow and recession analysis. *Water Resources Research 26:*1465–1473. DOI:10.1029/WR026i007p01465

Pomeroy, J.W., D.M. Gray, T. Brown, N.R. Hedstrom, W.L. Quinton, R.J. Granger, & S.K. Carey. 2007. The cold regions hydrological model: a platform for basing process representation and model structure on physical evidence. *Hydrological Processes 21:*2650–2667. DOI: 10.1002/hyp.6787

Rigon, R. n.d. *R resources for hydrologists.* [blog]. http://abouthydrology.blogspot.ca/2012/08/r-resources-for-hydrologists.html (Accessed August 2016)

Sandve, G.K., A. Nekrutenko, J. Taylor, & E. Hovig. 2013. Ten simple rules for reproducible computational research. *PLoS Computational Biology 9*(10):e1003285. DOI:10.1371/journal.pcbi.1003285

Shook, K. 2015. *Package 'WATCHr'.* www.usask.ca/hydrology/RPkgs/WATCHr.pdf (Accessed December 6, 2016).

Venables, W.N., D.M. Smith, & R Core Team. 2016. *An introduction to R. Version 3.3.2.* https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf (Accessed December 9, 2016).

Wickham, H. 2011. The split-apply-combine strategy for data analysis. *Journal of Statistical Software 40(1):1–29.* https://www.jstatsoft.org/article/view/v040i01/v40i01.pdf (Accessed August 2016).

Wilkinson, L. 2005. *A grammar of graphics*. Second edition. Springer-Verlag, New York.