RSPARROW: An R System for SPARROW Modeling

Richard B. Alexander and Lillian Gorman Sanisaca, U.S. Geological Survey

23 July, 2020(Version 1.1.0)

Contents

1 Introduction

1.1 Overview of SPARROW

SPARROW (SPAtially Referenced Regressions on Watershed attributes; Smith et al., 1997; Schwarz, et al., 2006) is a spatially explicit, hybrid (statistical and mechanistic) water-quality model developed by the USGS. The model has been used to quantify the sources and transport of contaminants in watersheds of widely varying sizes, from catchment to continental scales (e.g., Preston et al.,2011a,b; Alexander et al., 2000).

SPARROW applications have included studies of the sources of nutrients and sediment in watersheds and their delivery to inland lakes and coastal water bodies (Alexander et al., 2001; Alexander et al., 2002; Moore et al., 2004; Elliott et al., 2005; Hoos and McMahon, 2009; Alexander et al., 2008, Robertson et al., 2009; Brakebill et al, 2010; Preston et al., 2011; Robertson and Saad, 2011; Hoos et al., 2019). The studies have evaluated the water-quality effects of natural and anthropogenic factors, including the effects of denitrification (e.g., Alexander et al., 2000, 2007), impounded waters (Schmadel et al, 2018, 2019), and farm conservation practices (Garcia et al., 2016) on stream nutrients. SPARROW has also been used in studies of total streamflow across continental scales (Alexander et al., 2019a,b) and baseflow in western basins (Miller et al., 2016). SPARROW predictions have informed regional carbon budgets for coastal ecosystems (Najjar et al., 2018) and regional assessments of the effects of coastal fluxes of nutrients and organic carbon on estuarine ecosystem production (Herrmann et al., 2015). The model has also been used to simulate potential stream water-quality effects of hypothetical changes in land use and managed reductions in nutrient sources (Miller et al., 2019). Forecasting studies have used SPARROW to assess the effects of future climate and land-use change on nutrient and sediment fluxes and carbon storage in U.S. coastal waters (Bergamaschi et al., 2014). Internationally, SPARROW has been applied in New Zealand (Alexander et al., 2002; Elliott et al., 2005; Harris et al., 2009), Japan (Duan et al., 2015); and Canada (Robertson et al., 2019; Benoy et al., 2016; Wellen et al., 2014, 2012), with models currently under development for selected river basins in Brazil through collaborative projects between the USGS and the National Water Agency of Brazil (ANA) and Geological Survey of Brazil (CPRM).

SPARROW conceptually includes three major process components that explain spatial variability in stream water quality: contaminant source generation, land-to-water delivery, and stream and reservoir transport and decay. The non-linear and mechanistic structure of the model includes mass balance constraints and non-conservative transport components. This includes factors that control the attenuation and delivery of contaminants to streams via surficial and subsurface pathways and the removal of contaminants in streams and reservoirs, according to first-order decay kinetics. SPARROW is structured as a network of one-dimensional stream segments and their contributing drainage areas.

SPARROW models are developed by using NonLinear Least Squares (NLLS) methods to statistically estimate how stream water quality varies spatially in relation to explanatory variables associated with the source, land-to-water delivery, and aquatic components of the model. The NLLS methods and supporting diagnostics (Schwarz et al., 2006) are used to evaluate the statistical importance of candidate explanatory variables and the accuracy of the model fit to stream water-quality measurements. These evaluations have the objective of identifying the level of model complexity (i.e., the number and types of explanatory variables) that provides the most accurate fit to the observed stream monitoring data.

To ensure the accuracy of the model estimation, SPARROW is typically applied over large spatial domains to improve the data *quantity* (number of stream monitoring sites) and *quality* (i.e., variability in the water-quality response and explanatory variables) for calibrating the model. SPARROW model calibration requires estimates of the long-term mean annual load (mass per time; product of concentration and streamflow), the model response variable, which are obtained for stream water-quality monitoring stations on selected reach segments (typically, more than 30 stations are preferred). Estimates of the mean load are calculated in a step prior to SPARROW modeling using statistical load estimation methods (Lee et al., 2016, 2019; Hirsch, 2014; e.g., *LOADEST*, Runkel, 2004; *Fluxmaster*, Schwarz et al., 2006, Saad et al., 2019; *loadflexBatch*, Appling et al., 2015; *WRTDS* and *EGRET*, Hirsch et al., 2010, Hirsch and De Cicco, 2015). These methods statistically combine periodically collected stream concentration measurements (preferrably available for five or more years) with high frequency (e.g., daily) discharge, collected at the same or nearby locations, to estimate the

mean annual load; use of the more frequently collected discharge measurements improves the accuracy of the mean estimates of load (Lee et al., 2016, 2019).

SPARROW models also require geospatial data that describe natural and anthropogenic features the river network and watersheds to support the estimation of contaminant sources and transport in the model. These include stream network attributes (e.g., reach node topology, catchment drainage area, reservoir properties, mean annual velocity and streamflow; Brakebill and Terziotti, 2011), land cover, and other data on environmental conditions that are known to affect the supply of water and contaminants and their transport in watersheds, such as agricultural activities and chemical inputs, water use, soil properties, and climate. Pre-modeling analysis is necessary to spatially reference the geospatial data to stream reaches and catchments to facilitate their use in SPARROW models (see Chapter sub-section 3.1).

For complete information on the model theory, including the development, application, and interpretation of SPARROW models, the reader is referred to Schwarz et al. (2006), which also includes the SAS propriety version of the SPARROW software.

1.2 RSPARROW features

RSPARROW is a system of R scripts and functions for executing and evaluating SPARROW models that generates graphical, map, and tabular output. Users operate the system within RStudio from a single control script (*sparrow_control.R*) that accesses the supporting input files and functions.

This documentation describes the steps necessary to estimate the static version of the model. The static model provides reach-level predictions (and uncertainties) of the long-term mean annual water-quality loads, yields, and flow-weighted concentrations. The predictions also include the shares of the load attributable to individual upstream sources and predictions of the mass quantities of the total load and individual sources that are delivered to downstream water bodies.

RSPARROW offers the following enhancements to the SAS SPARROW (Schwarz et al., 2006) version:

- R functions and scripts, coded according to standardized open-source R protocols, promote work environment efficiencies that encourage the collaborative development of sharable SPARROW methods and functions. Use of the USGS GitLab and GitHub repositories ensures standardized methods of storage, maintenance, documentation, and retrieval of R functions and control scripts. A description of the RSPARROW design elements (Chapter 7) provides an informative guide for experienced R users and developers. Custom string search tools are also provided to developers to ensure that functional dependencies and variable attributes can be readily identified in the RSPARROW library to support the collaborative development of the software.
- Diagnostic metrics and interactive plots and maps, with user-controlled options, enable efficient evaluations of model performance and production of map output for explanatory and response data and model prediction metrics by monitoring sites, stream reaches and catchments. These capabilities are enhanced by use of a data dictionary as a master repository of system and user-defined variable names and attributes. Users also have the option to sequester monitoring (validation) sites to independently assess model performance.
- An R Shiny interactive Decision Support System (DSS) allows users to map SPARROW model predictions and geographic data by stream reach, catchment, and monitoring site location and to evaluate the effects of hypothetical changes in contaminant source inputs on downstream water-quality loads and concentrations. The mapper operates within a user-specified Web browser, with options to display standardized, globally available base maps (e.g., topography, land use, street maps). A batch-mode option supports production of multiple maps in a single execution for model applications to large watersheds that require longer execution times. The DSS executes source-change scenarios for user-defined watershed and reach locations, source types, and change magnitudes, with the water-quality effects reported in tabular and mapped output. Scenarios are applicable to modeled sources with inputs expressed in units of mass (e.g., fertilizers) or area (land use/cover).

RSPARROW includes the same analytical capabilities as the SAS SPARROW version (Schwarz et al., 2006), with the exception of the full bootstrap resampling method for prediction uncertainties, one of two methods in SAS SPARROW for estimating uncertainties. The full bootstrap method is computationally demanding and has been only rarely employed by USGS users. To estimate prediction uncertainties, RSPARROW executes the parametric Monte Carlo resampling methods that are also used in SAS SPARROW; this method provides sufficiently robust estimates with fewer computational demands.

1.3 RSPARROW software installation

1.3.1 System requirements and software preferences

RSPARROW requires a 64-bit processor on a Windows platform. A minimum of 4GB of RAM or larger is highly recommended. Users may find that 8GB or more of RAM is necessary to obtain acceptable performance when using the R Shiny interactive mapper and for model execution with large watersheds (e.g., more than 300,000 reaches). An installation of RStudio is the required interface for using RSPARROW. The software has not been tested on MAC and Unix platforms (note that two features of the current software version have Windows dependencies: the automated pop-up of CSV control files and the batch-mode operation). Our evaluations indicated that the R Shiny interactive mapper performed more reliably using Chrome, Microsoft Edge, and Firefox Web browsers as compared with that for Internet Explorer.

1.3.2 RSPARROW download and directory contents

Download RSPARROW from the USGS GitLab repository: https://code.usgs.gov/water/stats/RSPARROW. This includes scripts, functions, and supporting information (e.g., documentation, tutorials, R libraries). Please consider reporting bugs and asking questions on the GitHub Issues page: https://github.com/USGS-R/RSPARROW/issues.

Two directories contain the contents of RSPARROW (See Figure 1 for the contents of the RSPARROW directories as installed on a user's computer):

1. **RSPARROW_master**: Includes functions, sourced files (DLLs), meta-data, and documentation vignettes. The directory contents should not be modified by users. The documentation, with clickable links in the PDF table of contents, is located in the *"RSPARROW_master/inst/doc"* sub-directory.
2. **UserTutorial**: Contains the SPARROW total nitrogen models that are used for the tutorials presented in Chapter 6.
    - The *"results"* sub-directory contains a control script, input control files to execute the SPARROW model, and sub-directories for the tutorial models. The control script executes a predetermined sequence of function calls, governed by user settings in the control script (see the *executionTree* section of chapter 7 for a list of functions). Separate execution of RSPARROW library functions by users (e.g., to estimate models or generate predictions and maps) is not supported; however, the R Shiny interactive mapper can be independently enabled for any previously executed model within an RStudio session (see Chapter sub-section 4.4.8.5.2).
    - The directory structure and contents for the tutorial model, including the control scipt and input control files, can be used as a template to guide the setup and execution of user developed models.

1.3.3 Download of the required R software

RSPARROW 1.1.0 was tested for compatibility with versions of R, RStudio, and the required R library functions available at the time of public release (Sept. 2020): These include R-4.0.2, RStudio 1.3.959, and the required versions of R library dependencies that are listed in the DESCRIPTION file. RSPARROW may operate with more current versions of the software, but full compatibility of RSPARROW with newer versions is not guaranteed.

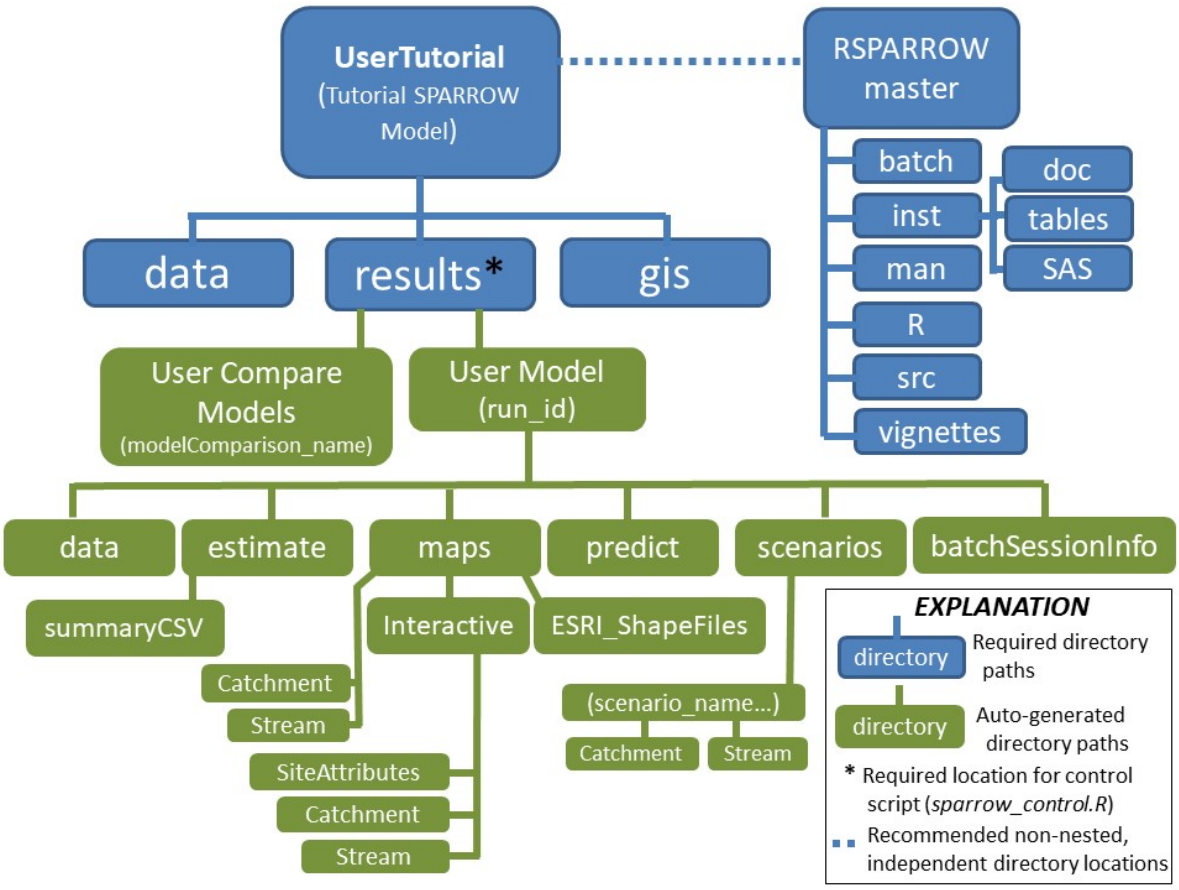


Figure 1: **RSPARROW directory structure.** Estimation of a new SPARROW model automatically creates a User Model directory, with the user-defined name "run_id", where copies of the control files are saved; model output is saved in functionally named sub-directories (green color). Results of previously estimated models can be accessed for predictions, mapping, or source-change management scenario evaluations, without having to re-estimate models. The "data", "gis", "results", and scenario name directories have user-assigned names as specified in the control settings. The "RSPARROW master" directory and its sub-directories should not be modified by user; no other sub-directories should be created in this directory.

The required software can be obtained as follows:

- *R:* The current and past versions are available from the Cran server at *https://cran.r-project.org/bin/windows/base/old/*. We recommend installation of the R directory in a location on your PC that does not require Administrator rights.
- *RStudio:* The most current version is available at *https://rstudio.com/products/rstudio/download/*, with older versions available at *https://rstudio.com/products/rstudio/older-versions/*.
- *R library dependencies:* Execution of the RSPARROW control script automatically installs the most recent versions of the required R library packages and functions in the user's default R library as specified in RStudio (see the next sub-section for instructions on setting RStudio defaults).

1.3.4 RStudio defaults

Users should be attentive to the RStudio default settings for the R version and the working directory. These default settings can be viewed and changed by selecting *"Tools » Global Options..."* from the RStudio drop-down menu. In the pop-up dialog box under the *"General"* and *"Basic"* tabs, the *"R Sessions"* settings can be specified:

- For the *"R version"*, select the *"Change..."* button and choose the appropriate 64-bit R version or browse to the directory where the R version is located and select it.
- For the *"Default working directory"*, select the *"Browse..."* button to identify the current default working directory. The user should either ensure that the *"RSPARROW_master"* directory is located in the current default working directory or change the default working directory to the directory where the *"RSPARROW_master"* directory resides.

After making these selections, *"Apply"* the change(s) and restart RStudio. The selections will then be applied to all future RStudio sessions. To verify that the selected R version is used as the default in RStudio, users can run the R command *.libPaths()* in the Console window; the first displayed library is the default.

1.3.5 R package installation during execution of the RSPARROW control script

When opening and executing the RSPARROW control script in RStudio, users should be attentive to messages and prompts associated with the R package installations:

- *Initial installation of R packages:* RStudio will print the following message in a yellow banner when the RSPARROW control script is opened in RStudio for the first time with a new R version: *"Packages devtools and rstudioapi required but are not installed."* Users should ignore this message and instead execute (*"Source"*) the control script, which will install these and the other required R packages (**User caution:** Package installation using the yellow banner prompt may exclude certain required packages such as *"backports"*).
- *R package installation from source:* Users should select *"No"* as the response to the following question, if it appears in a pop-up box in RStudio during the installation of R library dependencies: *"Do you want to install from sources the packages which need compilation?"*. A *"No"* response installs R packages from binary versions; these satisfy the RSPARROW required dependencies at the time of public release but may not include the most recent versions of the packages. The selection of *"Yes"* requires that users have installed *Rtools* (an R developer package) on the Windows operating system; this enables the compilation and installation of R packages from their source.
- *R package updating:* When the control script is executed, users are notified of the availability of more recent versions of the required R packages, based on comparisons of the installed versions with those available on the Cran server. If a newer version is available, then the user is asked whether the installed R package should be updated, as illustrated below for the *plyr* R package. Users are not required to update the R packages to versions that are more recent than those listed for the RSPARROW compatibility versions in the DESCRIPTION file; RSPARROW compatibility with updated R package

versions is not guaranteed. If all R packages are up to date, RSPARROW will run without any installation of R packages.

```
# AUTOMATED R PACKAGE UPDATING DURING EXECUTION OF THE RSPARROW CONTROL SCRIPT
#
# Illustration of the RSPARROW control script checks of R package versions
#
# The following statements are printed to the RStudio Console window because a more
# recent version of the required R package "plyr" is available on the Cran server:
#
These packages have more recent versions available.
It is recommended to update all of them.
Which would you like to update?

1: All
2: CRAN packages only
3: None
4: plyr (1.8.5 -> 1.8.6) [CRAN]

Enter one or more numbers, or an empty line to skip updates:
4

plyr (1.8.5 -> 1.8.6) [CRAN]  # RSPARROW was tested for compatibility with version 1.8.6
Installing 1 packages: plyr
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.0/plyr_1.8.6.zip'
Content type 'application/zip' length 1499854 bytes (1.4 MB)
downloaded 1.4 MB

package ‘plyr’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
    C:\Users\...\AppData\Local\Temp\RtmpeAHyOz\downloaded_packages

Loading RSPARROW
```

2 Directories, datafiles, and general operation

2.1 Required directory structure

The RSPARROW installation should include the two upper-level directories as shown in Figure 1, including the User Tutorial Directory and Master directory. The directories may be located parallel to one another or placed in other directory locations, provided that the directories are not nested within each other. Additionally, no changes should be made to the Master directory structure and contents. Advanced users and developers should consult chapter 7 to ensure that any updates to the Master files will not interfere with the operation of the RSPARROW system.

The User Directory must contain the three parallel sub-directories *"data"*, *"gis"*, and *"results"*, although the names of these sub-directories may be defined by users in the control script (see section 11 of the script). This directory structure and the required input datafiles are provided with the SPARROW tutorial model (Chapter 6), which automatically installs with the RSPARROW.zip file.

The *"results"* sub-directory contains the control script and supporting input control files that are required to operate the RSPARROW program. We recommend that users operate RSPARROW in RStudio by opening and executing the control script in the *"results"* sub-directory of any User Directory. Execution of the control script in an RStudio session makes the *"results"* sub-directory active by storing its path name internally in R for use in managing file input/output. The names of the control script and the associated input control files are fixed and should not be changed by users.

When modeling different water-quality constituents, users have the option to create and maintain multiple *"results"* directories, with names that correspond to the modeled chemical, physical, or biological constituents. This approach assumes that the reach network data, including monitoring site loads for all of the constitutents, are stored in the data input and geographic files in the *"data"* and *"gis"* directories located parallel to the *"results"* directory.

2.2 Model execution and required input files

RSPARROW can only be executed in RStudio via user settings in the control script, *sparrow_control.R*, and supporting input control files located in the *"results"* sub-directory (Table 1). The script consists of system variable settings, functionally organized in 12 sections (see Chapter 4, sub-section 4.1). Details are given in sub-section 4.2 on how to get started with executing the control script, including a checklist with an ordered sequence of steps to guide the setup and testing of system settings and new models (see sub-section 4.2.4). An overview of some of the key features of the control script operation is provided in this section.

In RStudio, users must navigate (*File » Open File...*) and open the copy of the control script in the user's *"results"* directory or if RStudio is the default program for opening R scripts simply double click on the script in Windows explorer. The execution of the control script automatically determines the location (path name) of the User Directory, which is internally stored in R. The user is only required to specify the relative or full path name for the master directory `path_master` (i.e., "RSPARROW_master") in a control setting (section 11 of the control script).

To execute the control script, users must always select and execute all statements in the script. One approach is to select the "Source" button in the upper right of the control script window in RStudio (located next to the "Run" button), which will execute the entire script. Execution of the script automatically loads the RSPARROW functions and implements the user control settings.

Model execution with the control script requires the input control files listed in Table 1 (and described in detail in Chapter 3). The *data1.csv* file (with user-specified name optional) provides the core geospatial data for navigating the reach network, estimating the model, and performing model applications (e.g., predictions, mapping); this file must reside in the *"data"* sub-directory. This file can be used with control script settings to automatically generate the other required CSV input control files (although additional editing of these files will be required prior to model execution; see checklist in chapter sub-section 4.2.4). These files (*dataDictionary.csv*, *parameters.csv*, *design_matrix.csv*), which reside in the *"results"* directory, control the model parameter settings and variable definitions (note that the CSV file names are fixed and

should not be changed by users). An optional control function (*userModifyData.R*), which must reside in the *"results"* directory, allows users to define new variables, perform R calculations (e.g., stream time of travel, land use classes), and execute conditional statements in the R environment (note that the function name is fixed and should not be changed by users).

Users have the option to display and edit the CSV input control files in a "pop up" window during the execution of the control script (implemented via shell commands–see section 11 of the script). Alternatively, users may open and edit these files from the system (e.g., Windows Explorer) environment (and from RStudio for *userModifyData.R*). All changes to these files must be saved before execution can continue (users can optionally choose to close the files).

Geographic data files (Table 1), which reside in the *"gis"* sub-directory, are not required to execute a SPARROW model, but are highly recommended to support the mapping of diagnostic metrics, model predictions, and explanatory variables by monitoring site, reach, or catchment. This is also a requirement for use of the R Shiny interactive DSS mapper. Note that all mapping functions in RSPARROW require R binary object versions of the geographic data (e.g, shape files); the binary objects are created by using control settings to convert the geographic shape files (section 8 of the control script).

2.3 User model subdirectories and datafiles

Execution of an RSPARROW model creates a *"User Model"* subdirectory in the *"results"* directory with the user-assigned model name, *"run_id"* (Fig. 1). The control and input files are automatically copied to this subdirectory (as an archive of model settings) and model output is directed to a series of functionally named subdirectories (e.g., *"maps"*, *"predict"*; Fig. 1).

The *"run_id"* functional subdirectories (Fig. 1) include text, PDF, HTML, binary R objects, and CSV output files (see Table 2 for a file listing). Saved R objects for estimation results and predictions allow users to conveniently access results of a previously estimated model to make predictions, maps, or evaluate management source-change scenarios, without having to re-estimate the model. The binary R object *subdata*, an extension of the *data1.csv*, is used to perform model estimation and prediction. All model estimation summary metrics in the text file *(run_id)_summary.txt* are additionally saved as separate CSV files in the *"summaryCSV"* subdirectory to provide users with convenient access to the results for post processing.

Users may overwrite these files and model subdirectories with each subsequent model execution by using the same *"run_id"* (e.g., during exploratory model development). Alternatively, users may define a new *"run_id"* to save the results of preferred models. The model subdirectory *"run_id"* provides a convenient archive of meta-data for publishing models. Users may also compare the results of the currently executed model to multiple prior models using the control setting `compare_models<-c("run_id1","run_id2",...)`; the control setting `modelComparison_name` creates a user-named subdirectory (Fig. 1) with CSV and text files containing summary metrics for the specified models.

Stream and catchment maps of model predictions, or other variables listed in the *dataDictionary.csv*, may be created by users with the `master_map_list` setting. The maps are stored as PDF files (with non-interactive features) or HTML files (with interactive features) in the *"maps"* subdirectory. Alternatively, the map setting `enable_shinyMaps<-"yes"` (section 9 of the control script) can be used to enable the R Shiny mapper, allowing users to interactively select and view predictions and optionally save maps of prediction metrics or other variables in the *"maps/Interactive"* subdirectory.

To enhance the computational speeds for model development and mapping output, binary R objects are created and stored in the *data*, *results*, and *gis* directories (see Tables 1, 3). Users are required to use the control file settings to create GIS R binary objects to support all mapping functions (see Chapter sub-section 4.4.8.1).

Results of the execution of user-defined management scenarios are output to the *"scenarios"* directory. The results are saved as CSV files and PDF or HTML maps in a user-assigned subdirectory that is identified by the setting `scenario_name<-"user named scenario directory"`, allowing users to archive the results of multiple scenarios, provided the model is not re-estimated (NOTE: if model estimation is executed, then all prior scenario results are deleted). The R Shiny interactive mapper can also be used, and is encouraged, to

execute management scenarios and create CSV and PDF/HTML output; the mapped results can be displayed interactively in the R Shiny mapper and optionally output to PDF/HTML files or batch processed from the menu and output as PDF/HTML files. The R Shiny batch mode is recommended for modeled spatial domains with many reaches (> ~300-500K) due to longer processing times.

RSPARROW models and procedures can also be executed in batch mode using the setting `batch_mode<-"yes"`. This saves a log file of the batch processing, with an archive of the R session stored as a binary file (*(run_id).RData*) in the *"batchSessionInfo"* directory (Table 2). The *(run_id).RData* file includes the dataframes, objects, and other data associated with the R modeling session; this file can be opened in RStudio to support further analysis of the model results or to track the source of possible execution errors. The RStudio session and the Rscript.exe (blackbox window) must remain open during batch mode execution.

Table 1. RSPARROW upper level directories and input control datafiles. R binary objects are shown in **bold** and contain R dataframes or list objects used to control the execution of functions in the R environment (see Table 3 explanation).

Directory Name	Datafile/**R Object** Name	Description/Purpose
data	*data1.csv*, **data1**	Reach-level attribute data required for RSPARROW model calibration and prediction.
gis	GIS shape files, **GeoLines**, **lineShape**, **polyShape**	Geographic data to support the mapping of metrics by monitoring site, reach, or catchment. Users are required to create the binary files to execute mapping functions.
results	*sparrow_control.R*	R script to control the specification and execution of RSPARROW models and all other operations, with user settings organized by 12 topical categories.
results	*userModifyData.R*	R script supports user-supplied R data calculations and conditional statements with existing and newly-created variables prior to model execution.
results	*dataDictionary.csv*	Repository for the variable definitions of all RSPARROW input data. These are required to support variable calculations, model execution, and user-controlled mapping of station and reach metrics. The file provides a crosswalk between user-defined variable names in the *data1.csv* and the required RSPARROW system variable names.
results	*master_dataDictionary.csv*	A continuously updated version of the *dataDictionary.csv* that is revised with each model execution to record any user edits to the *dataDictionary.csv* file.

Directory Name	Datafile/**R Object** Name	Description/Purpose
results	*parameters.csv*	Repository of the candidate explanatory variables for specifying RSPARROW models. Columns include initial parameter values, their minimum and maximum bounds, and a flag to identify variables to include in bivariate correlations of explanatory variables. User settings for the minimum and maximum bounds control the selection of parameters for model calibration.
results	*design_matrix.csv*	Interaction matrix for the source and land-to-water delivery variables.
results	**data1_priorImport**	Updated *data1.csv* input file with *dataDictionary.csv* variable names and user-selected network attribute updates. Created with the execution of the control script, following input of the *data1.csv* file (facilitates the speed up of data input in subsequent executions of the control script for different model specifications).

Table 2. Listing of the RSPARROW input and output datafiles and R objects in the User Model subdirectories. File names shown in parentheses are user defined model names (i.e., "run_id"), scenario and variable names, or labels assigned during execution ("time"). R objects (shown in **bold**) are stored as binary files; the objects contain dataframes or list objects within the R environment. Creation of output files requires that all settings are turned on; otherwise only selected files are output (see Table 6 in Chapter 4.3 for a listing of the crosswalk between the control settings and output of specific datafiles and objects).

User Model `run_id` Directory and Sub-Directory Names	Datafile/R Object Name
(run_id)	(run_id)_sparrow_control.R
	(run_id)_userModifyData.R
	(run_id)_dataDictionary.csv
	(run_id)_parameters.csv
	(run_id)_design_matrix.csv
	(run_id)_userSettings.csv
(run_id)/data	**subdata**
	sitedata
	vsitedata
(run_id)/estimate	(run_id)_summary.txt
	(run_id)_log.txt
	(run_id)_summary_predictions.csv
	(run_id)_diagnostic_plots.html
	(run_id)_diagnostic_sensitivity.html
	(run_id)_diagnostics_spatialautocor.html
	(run_id)_diagnostics_spatialautocor.txt
	(run_id)_explvars_correlations.pdf
	(run_id)_explvars_correlations.txt
	(run_id)_diagnostic_darea_mismatches.csv
	(run_id)_diagnostic_darea_mismatches.html
	(run_id)_residuals.csv
	(run_id)_bootbetaest.csv
	(run_id)_weights.pdf
	(run_id)_validation_plots.html
	(run_id)_DataMatrix.list
	(run_id)_SelParmValues
	(run_id)_sparrowEsts
	(run_id)_JacobResults
	(run_id)_HessianResults
	(run_id)_Mdiagnostics.list
	(run_id)_ANOVA.list
	(run_id)_vMdiagnostics.list
	(run_id)_vANOVA.list
	(run_id)_sensitivities.list
	(run_id)_Cor.ExplanVars.list
	(run_id)_BootBetaest
(run_id)/maps. . . (/Stream). . . (/Catchment)	(run_id)_(variable_name).pdf (.html)
(run_id)/maps/Interactive. . . (/Stream)	(run_id)_(variable_name).pdf (.html)
. . . (/Catchment). . . (/SiteAttributes)	**batch_(time).RData**
(run_id)/maps/ESRI_ShapeFiles/prediction	lineShape.(dbf,prj,shp,shx)
	polyShape.(dbf,prj,shp,shx)
(run_id)/maps/ESRI_ShapeFiles/residuals	residShape.(dbf,prj,shp,shx)
(run_id)/maps/ESRI_ShapeFiles/siteAttributes. . .	siteAttrshape.(dbf,prj,shp,shx)

User Model `run_id` Directory and Sub-Directory Names	Datafile/R Object Name
(run_id)/predict	(run_id)_predicts_load.csv (run_id)_predicts_load_units.csv (run_id)_predicts_yield.csv (run_id)_predicts_yield_units.csv (run_id)_predicts_load_boots.csv (run_id)_predicts_yield_boots.csv **(run_id)_predict.list** **(run_id)_BootUncertainties** **(run_id)_predictBoots.list**
(run_id)/scenarios/(scenario_name)	(scenario_name)_(run_id)_predicts_load_scenario.csv (scenario_name)_(run_id)_predicts_load_scenario_units.csv (scenario_name)_(run_id)_predicts_yield_scenario.csv (scenario_name)_(run_id)_predicts_yield_scenario_units.csv (scenario_name)_(run_id)_predicts_loadchg_scenario.csv (scenario_name)_(run_id)_predicts_yieldchg_scenario.csv (scenario_name)_(run_id)_scenario_metainfo.txt **(scenario_name)_(run_id)_DataMatrixScenarios.list** **(scenario_name)_(run_id)_predictScenarios.list** **batch_(time).RData**
(run_id)/scenarios/(scenario_name) . . . (/Stream). . . (/Catchment)	(scenario_name)_(run_id)_(variable_name).pdf (.html)
(run_id)/batchSessionInfo	**(run_id).RData** (run_id)_log.txt

Table 3. Binary files in upper level directories that enhance computational speed during model development and mapping. R binary object files are shown in **bold**. Control settings appear in the *sparrow_control.R* script for the identified section number.

Binary file name	Source	Updates/Changes	Control setting to enable use
data1	*data1.csv*	none	`input_data_fileName` (section 1)
data1_priorImport	*data1.csv* *dataDictionary.csv*	-*data1UserNames* replaced with *sparrowNames* from *dataDictionary.csv* in data1 object -calculated network attributes based on control setting (section 2): `calculate_reach_attribute_list <- c("hydseq","headflag", "demtarea")`	`load_previousDataImport` (section 1)
GeoLines, **lineShape**, **polyShape**	GIS shape files	none	`if_create_binary_maps` (section 8)

3 Input control files

The four input files that are required to execute an RSPARROW model are described in the following sections. The *data1.csv* file (section 3.1 below) provides the core geospatial data for the model (the name of the file is optional and can be set using the `input_data_fileName` setting; section 1 of the control script). This file can be used to automatically generate the other input files (*dataDictionary.csv*, *parameters.csv*, and *design_matrix.csv*; ***these file names are required and may not be changed***) using settings in section 1 of the control script. An optional R script *userModifyData.R* (***also a required file name***) supports user data calculations and is described in the final section 3.5 below.

3.1 *data1.csv*: Master data file of reach and monitoring station attributes for model input

This file contains the river network geographic data (i.e., reach topology) and the response (mean annual load) and explanatory variables required for RSPARROW model calibration and prediction. Each record in the file corresponds to a river reach segment and its contributing (incremental) drainage area. The river network data provides the core spatial infrastructure required to execute an RSPARROW model (Fig. 2; e.g., Brakebill and Terziotti, 2011). The stream reach network explicitly defines the surface-water flow paths that spatially connect contaminant sources and landscape features with observations of water quality at downstream monitoring stations. During the execution of RSPARROW, the sorting of the *data1.csv* records in hydrologic order from upstream to downstream (using the hydrological sequence number *"hydseq"*) ensures the accurate accumulation of area and contaminant mass and tracking of the downstream delivery of mass from upstream sources.

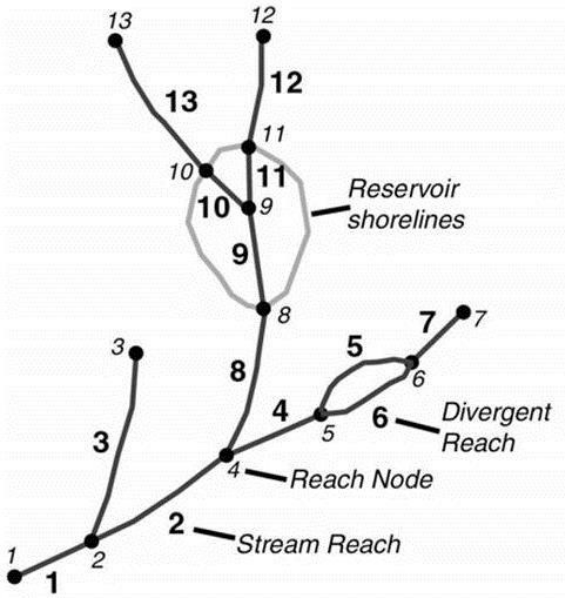
For a vector-based reach topology, a stream reach represents the length of stream channel that extends from one tributary junction to another. Reach nodes are point features that are associated with the location of tributary junctions at the ends of a reach. Reach nodes are preferred at locations where reaches overlay with the shorelines of impoundments (reservoirs, lakes). Thalweg "transport" reaches and their associated nodes are also preferred to simplify the transport of mass through large impoundments with shoreline segments. The reach-type indicator (Fig. 2) distinguishes between river reaches and reaches associated with impoundments (i.e., shorlines, interior thalweg segments), and allows the application of separate reach and impoundment decay functions. RSPARROW can also support raster-based network data provided that the node and data structure is identical to that described for the *data1.csv*.

In the vector-based topology, reach nodes are also preferred at the location of stream water-quality monitoring stations to enhance model prediction accuracy; in cases where reaches cannot be digitally "split", stations should be geographically referenced to reaches associated with the nearest upstream or downstream reach (headwater reaches may pose some limitations that require special attention).

The model structure supports the presence of distributary reaches or reach diversions in the stream network (Fig. 2); these may include braided channels or reaches where water is diverted to canals, other waterbodies, or for water supply. For distributary reaches, the model assumes that contaminants are diverted along flow paths in proportion to water flow. Therefore, an estimate is required for each reach of the "diversion fraction"-a measure of the fraction of the stream flow that is diverted in distributary reaches.

Preparation of the geographic data in the *data1.csv* file requires two processing tasks prior to RSPARROW modeling. These include using: (1) digital elevation model (DEM) data to derive the reach network, node topology, and catchment polygons; and (2) "spatial referencing" methods to associate explanatory data (e.g., land use, contaminant sources) with the hydrologic network features. Spatial referencing employs geographic information system (GIS) techniques (e.g., point-arc or polygon-polygon intersections) to digitally establish the geographic relation between stream reaches (and their associated catchments) and various watershed explanatory attributes that are used to estimate a model. For additional details, users should consult the SPARROW documentation (Schwarz et al., 2006, sections 1.3.2 and 1.3.3, p. 34-39).

RSPARROW provides optional methods for users to verify the accuracy of the hydrologic connectivity of the stream reaches as given in the *data1.csv* file; this ensures accurate water and contaminant routing and accumulation of mass in the model. The verification methods (section 2 of the control script) accumulate the



Reach Node Attribute Table

Reach Number	Up-stream Node	Down-stream Node	Diversion Fraction	Reach-Type Indicator
13	13	10	1.0	0
10	10	9	1.0	1
12	12	11	1.0	0
11	11	9	1.0	0
9	9	8	1.0	2
8	8	4	1.0	0
7	7	6	1.0	0
5	6	5	0.7	0
6	6	5	0.3	0
4	5	4	1.0	0
2	4	2	1.0	0
3	3	2	1.0	0
1	2	1	1.0	0

Figure 2: **Illustration of a vector-based RSPARROW reach network with node topology and water/mass routing table.** The reach-type indicator has possible values of "0" (stream reach), "1"(impoundment interior reach), and "2" outlet reach for impoundment. (from Schwarz et al., 2006)

reach incremental drainage area and compare the resulting totals with an independently derived measure of the total drainage area provided by the user (e.g., from GIS tools such as Arc Info). This option can also be used to identify erroneous breaks in the reach network, caused by disconnections in nodes or improperly oriented arcs.

In the initial execution of the control script, a binary version of the *data1.csv* file, which includes the results of data checks and any user-specified updates from section 2 of the control script, is automatically created in the "User Directory/results" sub-directory and called "(input_data_fileName)*_priorImport*". This allows users to speed up data input in subsequent executions of the control script (see control setting details described in section 1 of the control script or Chapter sub-section 4.4.1).

During the execution of an RSPARROW model in an RStudio session, a modified version of the *data1.csv*, the R object *subdata*, is created and serves as the data input for model estimation, prediction, and mapping during the RStudio session; the records are sorted in hydrological order (from upstream to downstream)–i.e., ascending order by the system variable *hydseq* (See Table 4 in section 3.2 for explanation). The object is modified from the *data1.csv* based on user-defined filters, using the control setting `filter_data1_conditions` (section 2 of the control script) or R commands in the *userModifyData.R* script (see sub-section 3.5). All variables in the *subdata* object are also contained in the *sitedata* object, with the number of records equal to the number of calibration sites and sorted in hydrological (from upstream to downstream reach) order using the *hydseq* system variable. The *subdata* and *sitedata* objects are used to manage model estimation and prediction as well as the processing of model residuals and diagnostics plots and maps and are stored in the "User Directory/results/(run_id)/data" directory. See Chapter sub-section 5.1 for additional details on the contents of the R objects.

3.2 *dataDictionary.csv*: Active respository of the system and user-defined variable names

This file provides a repository for defining the variable names (Table 4) that are input to all RSPARROW functions. These include variable calculations (*userModifyData.R* script), model execution and prediction, and user-controlled mapping of station and reach metrics. The dictionary also provides a crosswalk/linkage between

user-defined variable names in the *data1.csv* and the RSPARROW system variable names (*sparrowNames*) required to support RSPARROW functions and operations.

The *dataDictionary.csv* ensures global system-wide access to the *sparrowNames* variables (Table 4) during an RStudio session. By executing the control script within an RStudio session, variables listed in the *dataDictionary.csv*, including those modified in *userModifyData* script, are stored internally and are availabe for RSPARROW operations, including estimation, prediction, mapping, and source-change scenario evaluations (see sub-section 3.5).

There are eight types of variables that can be defined in the *dataDictionary.csv* (see Table 4). Two of the types, REQUIRED and FIXED, define system variables with unique reserved *sparrowNames* that cannot be changed by users. These variable types are also necessary for river network navigation and the estimation of models (although the FIXED variable *weight*, required for executing weighted least squares, is an optional variable).

REQUIRED variables indicate the minimum set of system variables needed to execute an RSPARROW model in simulation mode (`if_estimate_simulation<-"yes"`), using the initial parameter values as fixed model coefficients. This may be useful to test the model accumulation of sources and generation of reach predictions or to execute an exploratory land-cover/use model based on literature export coeficients.

All REQUIRED variables and their values must be defined in the *data1.csv* file (and not created in the *userModifyData* script) to execute network connectivity verifications (section 2 of the control script) with the following two exceptions. One is the *hydseq* variable (see Table 4), which can be optionally calculated using the control setting `calculate_reach_attribute_list` (this setting can also be used to compute the FIXED variables *demtarea* and *headflag*). The second is the *calsites* variable, which can be optionally created in the *userModifyData* script, as illustrated for the tutorial model.

FIXED variables are additional reach network variables needed to support the execution an RSPARROW model (with aquatic decay variables) using nonlinear least squares methods (`if_estimate<-"yes"`). These include variables necessary to generate diagnostics (metrics, plots, maps) to evaluate the statistical adequacy of the model fit to the observed data. Users have the option to assign "NAs" to the FIXED variables *length*, *rchtot*, *meanq*, and *hload* if aquatic decay is not included in the model specification.

Table 4 also identifies five types of explanatory variables that can be used to create the parameter files (*parameters.csv*, *design_matrix.csv*) for an initial SPARROW model (see sub-section 4.4.1). These variables include the SOURCE (contaminant sources or surrogates, such as land use), DELIVF (land-to-water delivery factors, such as climatic variables), STRM (stream decay), RESV (reservoir decay), and OTHER (additional variables for use in the stream or reservoir decay). At least one SOURCE variable is required to estimate an RSPARROW model. A DELIVF variable must also be defined in the data dictionary and parameter control files for RSPARROW to operate properly, although the DELIVF variable does not have to be included in the model estimation.

Note that the user is not restricted to the variable type (*varType*) designation for these explanatory variables when estimating RSPARROW models; this is controlled by the *parmType* designation in the *parameters.csv* file (*varType* is only used to create initial parameter files). Any *varType* in the *dataDictionary.csv* that does not match the *parmType* in the *parameters.csv* file is edited by RSPARROW during execution so that the *dataDictionary.csv* maintains an accurate record of the metadata for all variables. In this case, the following message will appear during execution of the control script and the changed variable types will be printed:

```
THE FOLLOWING ROWS OF THE dataDictionary.csv FILE HAVE BEEN UPDATED
 WITH varTypes MATCHING THE parmTypes IN THE parameters.csv FILE
```

All other variables in the *dataDictionary.csv* are classified as OPEN and are optional (Table 4). These include any variables that are present in the *data1.csv* file or created in the *userModifyData* script for which global access is required for model diagnostics, mapping, or evaluating management scenarios. For example, this may include land-use and physiographic classification variables or stream morphological variables, such as water velocity.

Users are encouraged to complete as many of the data dictionary entries as possible by assigning attributes

(including units and explanations) to all of the *data1.csv* variables and newly-created or modified variables in the *userModifyData.R* script (`if_userModifyData<-"yes"`) where user-defined calculations are specified. Selected output (tables, plots, maps) is annotated with the variable units and explanations; this information also provides documentation for RSPARROW model developers and for future users in archived model files.

Note that a master archive of the *dataDictionary.csv* file, called the *master_dataDictionary.csv*, is maintained in the *results* directory to track the evolution of the file contents. The master version of the file is continuously updated with each model execution to record any user edits to the data dictionary. New variable records are appended to the file if updates are detected to any of the dataDictionary columns. Two additional columns also appear in the file with the labels *run_id* and *duplicate_sparrowName*. The *run_id* records the original model ID for unchanged variables, and records the most recent model *run_id* for variables where updates are detected to any of the *dataDictionary.csv* columns. The *duplicate_sparrowName* column defaults to a "0" value, but will display a "1" for any variables with columns that have been edited.

Table 4. The dataDictionary.csv file contents and explanation. The user-defined names are the column headers from the *data1.csv* file. The column headings in *italics* are the column labels in the *dataDictionary.csv* file. Variables in **bold** can be optionally calculated with the user control setting `calculate_reach_attribute_list`. The REQUIRED and FIXED *sparrowNames* shown in *italics* or **bold** have unique reserved names that cannot be changed by the user; all other *sparrowNames* are user-defined names, with the exception of the OPEN variable *valsites*. Variables that are missing in the *data1.csv* file and created in the *userModifyData.R* script should display a "NA" for their *data1UserNames*. REQUIRED variables indicate the minimum set of variables needed to execute an RSPARROW model in simulation mode, using the initial parameter values as fixed model coefficients. FIXED variables are additional river network variables needed to execute an RSPARROW model (with aquatic decay) using nonlinear least squares methods (with the exception of the variable *weight*).

varType (variable type)	*sparrowNames* (R internal name)	*data1UserNames* (user defined names)	*explanation* (variable description)
REQUIRED	*waterid*	user_id	Unique reach identification (ID) number
REQUIRED	*fnode*	user_fnode	Reach from (upstream) node
REQUIRED	*tnode*	user_tnode	Reach to (downstream) node
REQUIRED	*frac*	user_frac	Reach transport fraction, ranging from 0 to 1 (1=no diversion of water/mass)
REQUIRED	*iftran*	user_iftran	"if transport" indicator (0=no; 1=yes); nontransport reaches, such as coastal and lake shoreline segments, should be set to zero
REQUIRED	*demiarea*	user_area	Reach incremental drainage area
REQUIRED	**hydseq**	user_hydseq	Unique hydrological sequence number; used to sort the *data1.csv* reach file (and *subdata* object file) in hydrological order (from upstream to downstream) to enable the summation of area and constituent mass
REQUIRED	*termflag*	user_termflag	Reach terminal flag indicator (1=stream reach; 3=coastal shoreline reach)
REQUIRED	*rchtype*	user_rchtype	Reach type indicator (0=reach; 1=reservoir internal reach; 2=reservoir outlet reach; 3=coastal segment)
REQUIRED	*calsites*	user_calsites	Calibration site flag that identifies user selected sites (0=not selected; 1=selected) for model calibration (`if_estimate`<-"yes") and model simulation (`if_estimate_simulation`<-"yes"). Selected sites must have positive load values (*depvar*>0). One site per reach is allowed.

varType (variable type)	*sparrowNames* (R internal name)	*data1UserNames* (user defined names)	*explanation* (variable description)
FIXED	*target*	user_target	Terminal target reach (1=target; 0=non target) for computing load delivery from each upstream reach to the nearest downstream target reach (e.g., input to estuary or reservoir)
FIXED	*lat*	user_lat	Monitoring station latitude (decimal degrees)
FIXED	*lon*	user_lon	Monitoring station longitude (decimal degrees)
FIXED	*rchname*	user_reach_name	Reach name
FIXED	**demtarea**	user_demtarea	Reach total drainage area
FIXED	**headflag**	user_headflag	Reach headwater flag (1=headwater reach; 0=other reach)
FIXED	*length*	user_length	Reach length
FIXED	*meanq*	user_mean_flow	Mean annual streamflow; needed for estimating concentration and stream decay
FIXED	*rchtot*	user_rchtot	Reach time of travel; needed for estimating stream decay
FIXED	*hload*	user_hload	Areal hydraulic load for impoundments; needed for estimating aquatic decay
FIXED	*staid*	user_staid	Unique station ID sequence number; an updated hydrologically ordered sequence number is automatically assigned based on user-selected calibration sites (*calsites*=1)
FIXED	*station_id*	user_station_id	Unique alphanumeric station ID number
FIXED	*station_name*	user_station_name	Monitoring station name
FIXED	*depvar*	user_depvar	Mean annual load (response variable)
FIXED	*depvar_se*	user_depvar_se	Mean annual load standard error
FIXED	*weight*	user_weight	Optional residual weights used to execute a weighted nonlinear RSPARROW model (default=1.0)
SOURCE	source_variables	user_source_vars	Source-related explanatory variables
DELIVF	delivery_variables	user_delivery_vars	Land-to-water delivery explanatory variables
STRM	stream_variables	user_stream_vars	Stream decay explanatory variables
RESV	reservoir_variables	user_reservoir_vars	Reservoir decay explanatory variables
OTHER	other_variables	user_other_vars	User-specified variables for inclusion in the stream or reservoir decay specifications
OPEN	LULC_variables	user_LULC_vars	Variables for the drainage areas of land cover and land use classes of interest; used for diagnostic metrics and plots (see section 5 of the control script; `class_landuse` setting)
OPEN	CLASS_variables	user_class_vars	A unique (e.g., hierarchical) classification code for spatially contiguous drainage areas, such as hydrologic regions (e.g., HUC in USA), political units, or other ecological or physiographic regions; used for diagnostic metrics and plots (see section 5 of the control script; `classvar` setting)

varType (variable type)	*sparrowNames* (R internal name)	*data1UserNames* (user defined names)	*explanation* (variable description)
OPEN	S_source_variables	user_source_vars	User-defined source variables (listed in the `scenario_sources` setting; see section 9 of control script); contains the source-change factors for evaluating hypothetical management scenarios; the prefix "S_" is required
OPEN	user_variables	user_variables	any variables present in the *data1.csv* file or created in the *userModifyData* script for which global access is required for mapping
OPEN	*valsites*	user_valsites	Validation site flag to support one of two methods for the selection of sites for model validation (0=not selected; 1=selected); selected sites must also have positive load values (*depvar*>0). See explanation for the `if_validate<-"yes"` control setting in subsection 4.4.6; the *valsites* name (i.e., *sparrowNames*) is a unique reserved system variable name that must be used with one of the validation site methods.

3.3 *parameters.csv*: Controls selection of explanatory variables in the model

This file controls the selection of explanatory variables for inclusion in the models and controls the parameter settings for model estimation (Table 5). The columns of the file include parameter names and descriptors and settings for initial values and the minimum and maximum bounds of model parameters. Users are encouraged to use the *parameters.csv* file as a repository of the candidate explanatory variables for RSPARROW models from which subsets of variables can be systematically evaluated.

See the discussion of the `if_estimate` control setting in Chapter sub-section 4.4.4 (section 4 of the control script) for details on recommended parameter settings for the *parameters.csv* file.

The following are guidelines on the setup and specification of the RSPARROW model parameters:

- At least one SOURCE variable is required to estimate/simulate an RSPARROW model. Thus, at least one SOURCE parameter must be specified in the *parameters.csv* file for inclusion in the model estimation/simulation.
- At least one land-to-water delivery (DELIVF) variable must be defined in the *dataDictionary.csv* and *parameters.csv* control files for RSPARROW to operate properly; however, a DELIVF variable does not have to be included in the model estimation/simulation.
- Any of the variables listed in the *dataDictionary.csv* may be added to the *parameters.csv* file for possible inclusion in RSPARROW models.
- There is no requirement for the *parmType* in the *parameters.csv* to be consistent with the *varType* designation for the variable in the *dataDictionary.csv*. Any *varType* in the *dataDictionary.csv* that does not match the *parmType* in the *parameters.csv* file is edited by RSPARROW during execution so that the *dataDictionary.csv* maintains an accurate record of the metadata for all variables.
- The *varType* designation of explanatory variables in the *dataDictionary.csv* can be used with settings in section 1 of the control script to create an initial version of the *parameters.csv* and *design_matrix.csv* files.

- The SOURCE and DELIVF variables in the *parameters.csv* are required to be identical to the variables listed in the *design_matrix.csv* file.

Table 5. The model parameters.csv file contents and explanation. User selection of the minimum and maximum bounds on the parameters control the selection of variables for model estimation. Explanatory variables to be included and estimated in the model require that users set *parmMin* to be less than *parmMax*. Parameters to be excluded from the model estimation or simulation (using a fixed *parmInit* value) must have a value of zero or missing (NA) specified for both *parmMin* and *parmMax*. An individual parameter can be fixed to a constant value (*parmInit*) by setting the initial value and the lower and upper bounds to the identical non-zero value (i.e., *parmInit* = *parmMin* = *parmMax*). All parameters in a model can be set to a constant value (*parmInit*) using the control setting `if_estimate_simulation<-"yes"` (section 4 of control script).

Column Header Label	Description / Explanation
sparrowNames	Internal R system variable name for the model explanatory variable
description	Character description for the variable for output files
parmUnits	Unit definitions for the model parameter for output files
parmInit	Initial value for the nonlinear least squares (NLLS) estimation or the fixed value for predictions in simulation mode. For SOURCE and aquatic decay (STRM, RESV) variables, an initial value of zero or a small positive value is recommended. For the delivery factors (DELIVF), an initial value of zero is generally recommended, given that the estimated parameters may be either negative or positive
parmMin	Minimum bound on the estimated or fixed parameter. Typically zero for SOURCE, STRM, and RESV parameter types and a zero or small negative value (e.g., -10000) for DELIVF and OTHER parameter types. A zero or NA value for both the *parmMin* and *parmMax* will exclude the parameter from the model.
parmMax	Maximum bound on the estimated or fixed parameter. Typically a large positive value for all parameter types (e.g., 10000), but a zero may be specified for DELIVF or OTHER parameter types. A zero or NA value for both the *parmMin* and *parmMax* will exclude the parameter from the model.
parmType	Parameter type (SOURCE, DELIVF, STRM, RESV, OTHER)
parmCorrGroup	Parameter selection setting for inclusion or exclusion from diagnostic plots and matrices of all-possible explanatory variable Spearman's rank correlations (enter "1" to be included or "0" to be excluded from the correlations; see explanation of the control setting `if_corrExplanVars` in section 5 of the control script)

3.4 *design_matrix.csv*: Controls source and land-to-water delivery interactions

This file defines an interaction matrix that controls which land-to-water delivery variables are allowed to interact with and mediate individual source variables. The rows list the sources (SOURCE) and the columns display the land-to-water delivery variables (DELIVF). These variables and their variable type (*varType*)

should be identical to those listed in the *parameters.csv* file. The matrix values are either "1" (source-delivery interaction allowed) or "0" (no interaction effect is allowed). All variable names should be specified to be consistent with the *sparrowNames* listed in the *dataDictionary.csv* files.

3.5 *userModifyData.R* script: Supports user-defined R data calculations prior to model execution

The function supports user-defined R data calculations and conditional R statements with variables defined in the *dataDictionary.csv* file or newly-created variables that are defined and only temporarily used in the function. These include R statements to perform pre-modeling calculations of explanatory variables and reach attributes, including attributes of the calibration sites that are associated with selected reaches. Examples include calculations of the reach water time-of-travel, land-use class definitions, or monitoring site metrics (e.g., mean concentration or yield). Examples also include the use of conditional statements to identify target reaches (for tracking the downstream delivery of loads to user-preferred watershed outlets or receiving waterbodies) or to select calibration and validation sites (e.g., including conditions to eliminate sites from consideration that have loads deemed unacceptable for use in the model calibration). Example R statements are shown below in sub-section 3.5.2.

The *userModifyData* script is executed by the control script setting `if_userModifyData<-"yes"` (see section 1 of the control script). Users should only edit the copy of the function located in the active *"results"* sub-directory–i.e., the *"results"* sub-directory associated with the user's current active control script in the RStudio session.

3.5.1 Guidelines for calculations with *dataDictionary.csv* and newly-created variables

- **Global access to the variables** - Global access requires that the variables are defined in the *dataDictionary.csv* file. This ensures availability of the variables for RSPARROW operations during an RStudio session, including estimation, prediction, mapping, and source-change management scenario evaluations. This is achieved via execution of the control script, which saves variables listed in the *dataDictionary.csv*, including *dataDictionary* variables that are defined or modified in the *userModifyData* script, into the binary R object *subdata*. The object is a modified dataframe of the *data1.csv* file. Variables in the *subdata* object are also subsequently stored in the *sitedata* binary object, which has a record length equal to the number of calibration sites (the object file controls processing of model residuals and diagnostics plots and maps).
- **Record length of the variables** - R statements in the *userModifyData.R* script should not alter the number of reach records associated with the *dataDictionary* variables. The length of the *dataDictionary* variables is equal to the number of reaches in the *data1.csv* file or the number of "user-selected" reaches in cases where a user chooses to reduce the number of reaches (i.e., using the control setting `filter_data1_conditions` in section 2 of the control script).
  - Record lengths equal to these reach lengths are also required for variables that only have data available for selected reaches, such as the variables associated with reservoirs or calibration sites (see the example below using the required system variable *calsites* for calibration sites).
  - The `filter_data1_conditions` control setting should be used if a user wants to reduce the number of reach records; for example, to select a subset of reaches to apply the SPARROW model to a smaller spatial domain (note that this setting is implemented prior to execution of the *userModifyData* script).
  - Newly-created variables that are only used temporarily in the function (i.e., variables that are not listed in the *dataDictionary.csv*) can have lengths that differ from the number of user-selected reaches (e.g., length = number of calibration sites), provided that the variables are properly match-merged to execute computations or associations with reach-length *dataDictionary* variables (see examples in the next sub-section).

- **Record order of the variables** - R statements in the *userModifyData* script should not alter the hydrological order (from upstream to downstream) of the reach-length records associated with the *dataDictionary* variables. These records are contained in the *subdata* object and sorted in ascending order by the system variable *hydseq* (Table 4 in section 3.2) prior to execution of the *userModifyData.R* script.
  - Newly-created variables that are read from external files into the *userModifyData* script and have a different reach order or number of records (e.g., calibration site attributes) from that of the ordered *hydseq* variable, or *dataDictionary* variables in which the *userModifyData* calculations alter their reach ordering, should be match-merged with a static reach identification variable (e.g., *waterid*, *station_id*) and sorted by the *hydseq* variable (i.e., from upstream to downstream). This ensures that these variables are consistent with the reach length and order of other system variables and reach attributes that are stored in the *subdata* object upon completion of the *userModifyData* script. See the examples in the next sub-section for illustrations of the match-merge operations.
  - Note that the order of the reach records in the *subdata* object and the associated sequence ordering of the calibration and validation stations (*sitedata* and *vsitedata* objects) are sensitive to the method used to assign the *hydseq* variable. For example, the record order for these files is potentially altered by using the internal RSPARROW function enabled by the control setting option `calculate_reach_attribute_list<- c("hydseq")`.

3.5.2 Examples of user-defined R statements for calculations

The *userModifyData* script includes the following standard code:

- The longitude is set to be consistent with the map limits for North and South American model applications where a negative longitude is required:

```
if (!is.na(lon_limit)){
  if(lon_limit[1] < 0 & lon_limit[2] < 0) {
    for (i in 1:length(lon)) {  # ensure that longitude is negative
      if(!is.na(lon[i])) {
        lon[i] <- -(abs(lon[i]))
      }
    }
  }
}
```

The following examples of R code illustrate various calculations and settings that users can execute in the userModifyData.R function. For examples of the use of the R code in RSPARROW models, users can also examine the *userModifyData.R* scripts associated with the tutorial models as described in Chapter 6.

- Identify calibration stations for model estimation (1=station selected for use in calibration; 0=station not selected for use:

```
# set default for stations with positive mean annual load values
calsites <- ifelse(depvar>0,1,0)

# Exclude monitoring station loads with standard error greater than 50%,
#    where the depvar_se is defined by the same units as depvar
calsites <- ifelse(depvar > 0 & (depvar_se/depvar)*100 > 50,0,calsites)

# Exclude a monitoring station by identification number
calsites <- ifelse(station_id=="XYZ02220",0,calsites)
```

- Define the land use and land cover variables:

```
# Check land use variables (in units of area) for NAs
agric <- cultiv + pasture
shrubgrass <- shrub + grass
names <- named.list(agric,forest,grass,shrub,barren,trans,
                    wetland,urban,shrubgrass)
replaceNAs(names)  # function to replace the NAs with zeros
```

- Hydrological / morphological calculations:

```
# Jobson (1996) equation 14 calculation of reach velocity
g <- 9.80665   # gravitational acceleration (meters per second squared)
# demtarea - total drainage area (units=square kilometers)
# meanq - Mean annual streamflow (units=cubic meters per second)
DAdimless <- (demtarea**1.25 * sqrt(g)) / meanq    # equation 10 from Jobson
# Mean annual velocity (units=meters per second)
velocity <- 0.02 + 0.051 * DAdimless**0.821 * 1**-0.465 * meanq / demtarea

# rchtot - reach time of travel (days)
rchtot <- (length*1000) / (velocity * 86400)

# depth - Mean annual stream depth (units=meters) based on Leopold and Maddock
#  relation (1953) (see also Alexander et al. 2000)
# meanq - Mean annual streamflow (cubic feet per second)
# 35.31467 converts streamflow from ft3/s to m3/s
depth <- 0.0635 * (meanq * 35.31467) ** 0.3966
```

- Discrete stream decay, with reaction rate constants for coefficients estimated separately for stream flow classes:

```
# meanq - Mean annual streamflow (units=cubic feet per second)
# rchtot - Mean annual reach time of travel (units=days)
# rchtype - Reach type indicator (0=reach)
# rchdecay1, rchdecay2, rchdecay2 are the explanatory variables associated
# with the estimated RSPARROW reaction rate constants expressed as the loss
# per day of mean water travel time (see equation 1.30 in Schwarz et al. 2006)
rchdecay1 <- ifelse(meanq <= 500 & rchtype == 0,rchtot,0.0)
rchdecay2 <- ifelse(meanq > 500 & meanq <= 10000 & rchtype == 0,rchtot,0.0)
rchdecay3 <- ifelse(meanq > 10000 & rchtype == 0,rchtot,0.0)
```

- Continuous stream decay, expressed as a mass-transfer rate constant (units=length per time):

```
# rchtot - Mean annual reach time of travel (units=days)
# depth - Mean annual stream depth (units=meters)
# strmdecay is the time of travel to depth ratio explanatory variables associated
#   with the estimated RSPARROW mass-transfer rate (units=meters per day)
#   (see equation 1.32 in Schwarz et al. 2006)
strmdecay <- rchtot / depth

# only apply continuous decay to reaches with mean annual flow greater than
#   2.8 cubic meters per second
strmload <- ifelse(meanq > 2.8,0,strmload)
```

- Reservoir decay, expressed as a mass-transfer rate constant, assuming that the water body is uniformly mixed (see Schwarz et al., 2006 section 1.4.5 for details):

```
#  hload - the areal hydraulic load (units=meters per day
#  resdecay - the reciprocal areal hydraulic load, the explanatory variable associated
#    with the apparent settling velocity or mass-transfer coefficient for impoundments
#    (see equation 1.34 in Schwarz et al. 2006)
resdecay <- ifelse(hload > 0,1.0/hload,0.0)

#  Assignment of the reservoir hydraulic load to the impoundment outlet
#    reach (rchtype=2) based on the assumption that the water body is uniformly mixed
resdecay <- ifelse(rchtype == 2,1.0/hload,0.0)
```

- Define the source-related variables for the source-change scenarios that are targeted to specific reaches:

```
# Apply load reduction factors to the scenario matrix for selected reaches
#  in cases where the control setting: if_predict_scenarios<-"selected reaches".
#  The "S_" prefix is required in these cases.
#  The example applies scenario conditions to all reaches in HUC2 = 5 (Ohio basin)
S_point <- ifelse(huc2 == 5,1,1)      # point sources
S_atmdep <- ifelse(huc2 == 5,1,1)     # deposition
S_fertilizer <- ifelse(huc2 == 5,0.25,1)     # farm fertilizer
```

- Define monitoring station attributes to display in the diagnostics output PDF (setting `map_siteAttributes.list` in the control script section 8):

```
# demtarea (units=square kilometers)
# meanq (units=cubic meters per second)
# ConcFactor (units=3.170979e-05) - the concentration conversion factor for
#     the product of load (kg/yr) and discharge (m3/s)
meanload <- depvar                  # Mean annual load (kg/yr)
meanyield <- depvar / demtarea      # Mean annual yield (kg/km2/yr)
meanconc <- depvar/meanq*ConcFactor # Mean annual flow-weighted concentation (mg/L)
meanloadSE <- depvar_se             # Standard error of the mean annual load (%)
```

- Designate a reach as a target reach to allow the model to compute for upstream non-target reaches the fraction of the reach load that is delivered to the nearest target reach, a downstream receiving water body (e.g., estuary, reservoir):

```
# termflag gives the type of terminal reach (1=reach; 3=coastal or
#   reservoir shoreline segment)
target <- ifelse(termflag == 1 | termflag == 3, 1, 0)
```

- Specify the "if transport" condition for transfer of load from upstream node to downstream node:

```
# transport requires a positive mean annual streamflow
iftran <- ifelse(meanq > 0,1,0)
# transport of mass no allowed beyond the terminal reaches or along
#   coastal/reservoir shorelines
iftran <- ifelse(termflag == 3 | termflag == 1,0,iftran)
```

- Read an attribute of the calibration sites (with length equal to the number of sites) from an external file and associate the attribute with the system variable *station_id* (with length equal to the number of reaches and hydrological ordering by the system variable *hydseq*):

```
# Read a calibration site attribute from a user's external CSV file
#   'Indata' object contents, with length equal to the number of calibration sites:
#      station_id - unique alphanumeric station ID (type=character)
#      attribute_site - a calibration site attribute (type=numeric)
Indata <- read.csv(file="CSV path and filename", header=TRUE, sep=",")
```

```
  # create a data frame with a common ID, the "station_id" system variable,
  #    with length equal to the number of reaches
  sdata <- data.frame(station_id,hydseq)   # length = number of reaches
  # merge with the site attribute in the Indata object
  sdata <- merge(sdata,Indata,by="station_id",all.y=FALSE,all.x=TRUE)
  # resort by 'hydseq' order to ensure consistency with the system variables
  #   in the "userModifyData.R" function
  sdata <- sdata[with(sdata,order(sdata$hydseq)), ]
  # create a reach length variable, also defined in the dataDictionary.csv file, to
  #  store the site attribute for reaches associated with calibration sites
  attribute1_reach <- numeric(length=waterid)
  attribute1_reach <- sdata$attribute_site
```

- Read a reach attribute (with length equal to the number of reaches) from an external file and associate the attribute with the system variable *waterid* (with length equal to the number of reaches and hydrological ordering by the system variable *hydseq*):

```
# Read a reach attribute from a user's external CSV file
#   'Indata' object contents, with length equal to the number of calibration sites:
#      waterid - unique reach identifier number
#      attribute_reach - a reach attribute (type=numeric)
Indata <- read.csv(file="CSV path and filename", header=TRUE, sep=",")

  # create a data frame with a common ID, the "waterid" system variable
  sdata <- data.frame(waterid,hydseq)   # length = number of reaches
  # merge with the reach attribute in the Indata object
  sdata <- merge(sdata,Indata,by="waterid",all.y=FALSE,all.x=TRUE)
  # resort by 'hydseq' order to ensure consistency with the system variables
  #   in the "userModifyData.R" function
  sdata <- sdata[with(sdata,order(sdata$hydseq)), ]
  # create a reach length variable, also defined in the dataDictionary.csv file, to
  #  store the site attribute for reaches associated with calibration sites
  attribute1_reach <- numeric(length=waterid)
  attribute1_reach <- sdata$attribute_reach
```

4 RSPARROW execution and control script settings

4.1 Overview of the *sparrow_control.R* script settings

The *sparrow_control.R* script controls the specification and execution of RSPARROW models and all other modeling operations in RStudio. It includes user settings organized functionally by the following 12 topical categories:

1. Data import settings and parameter control file setup options
2. Stream network attributes, verification, and reach filtering
3. Monitoring site filtering options
4. Model estimation
5. Model spatial diagnostics
6. Selection of validation sites
7. Model predictions
8. Diagnostic plots and maps
9. RShiny Decision Support System (DSS) mapper
10. Model prediction uncertainties
11. Directory and model identification and control script operations:
    - Set path and directory names
    - Set model identification *run_id* name
    - Load previous model settings into the active control script
    - Model comparison summary
    - File editing and batch execution options
    - Customized error handling options
12. Installation and updating of the R libraries

4.2 Executing the control script in RStudio

4.2.1 Finding initial copies of the control script and control input files

To setup a model for the first time, a version of the control script (*sparrow_control.R*) and a verson of the *userModifyData* script in the *UserTutorial/results* should be copied to the users *results* directory. The settings should be reviewed and updated in RStudio to conform with user preferences for a new model.

The associated parameter control input files *parameters.csv*, *design_matrix.csv*, and *dataDictionary.csv*) can be created using the control settings in section 1 of the control script and the user's new *data1.csv* file. Consult the comments in the control script and the control script settings described in Chapter 4 sub-sections 4.2.4 and 4.4 below; these provide assistance with the setup of the parameter control input files (as described in Chapter 3 above) and the execution of a model.

4.2.2 RSPARROW execution in six steps

1. Open the control script (*sparrow_control.R*), located in the user's *"results"* directory. In RStudio, users should navigate using *File » Open File...* and open the copy of the control script in the user's *"results"* directory. The name of the control script and its placement in the *"results"* directory are required to execute all models. Execution of the control script in an RStudio session automatically determines the location (path name) of the User Directory, which is internally stored in R. Users are only required to specify the path name of the RSPARROW master directory `path_master` in a control setting (section 11 of the control script). *Users should not open and attempt to execute in RStudio any archived copies of the control script in the model "run_id" subdirectories; this will cause termination of the model execution. Execution of archived control files should only be done using the* `copyPriorModelFiles` *setting.*

2. Edit the control settings and save the control script. In addition to editing the control script settings, the following control input files are required to execute the control script: *parameters.csv*, *design_matrix.csv*, and *dataDictionary.csv*. The *userModifyData* script is optional but is commonly needed to

define or modify variables prior to model execution. See instructions in section 4.2.4 on the setup of these files.

3. Execute all statements in the control script. One approach is to select the "Source" button in the upper right of the control script window in RStudio (located next to the "Run" button), which will execute the entire script.

4. Install or update the R library dependencies. Execution of the control script automatically installs the most recent versions of the required R library packages and functions in the user's default R library as specified in RStudio. However, users should be attentive to messages and prompts associated with the R package installations during execution of the control script. See Chapter sub-section 1.3 for details on the recommended user responses to prompts.

5. If CSV model input files pop up, make any required edits, then save and close the files. Execution will be paused until the user indicates that all control files have been saved. The CSV files control the variable definitions (*dataDictionary.csv*) and explanatory variables that are specified in the models (*parameters.csv*, *design_matrix.csv*). The CSV files can be set to automatically open using the settings described in section 11 of the control script. For example, using the "pop up" setting to open the *parameters.csv* file (see sub-section 3.3) allows users to control the explanatory variables that are included in a model estimation; this avoids having to navigate the Window's directory structure to manually open the CSV file each time a model is estimated.

6. Enter "1" for "yes" in the Console window (and a carriage return) in response to the message "Did you save the active control file. . . ".. This will start the execution of data preparation and modeling tasks.

4.2.3 Tips for executing the control script in RStudio

When selecting and executing the control script, users should take care to execute cursor and keyboard selections in the correct sequence. This will avoid possible complications when R executes the control script.

We highly recommend the following sequence of cursor and keyboard steps in RStudio to execute the control script (with care to avoid the cursor placement problems described in step 5):

1. Once the control script has been opened in RStudio and edits of user settings completed, save the edits by clicking the "save" icon tab located on the bar above the control script text.
2. Users have two options to execute all statements in the script. One is to select the "Source" button in the upper right of the control script window in RStudio (located next to the "Run" button), which will execute the entire script.
3. If CSV input files pop up, make any required edits, then save and close the files.
4. The message "*Did you save the active control file. . .* " will appear in the Console window (lower left). Place the cursor in the Console window, and enter "1" (and a carriage return) to continue the model run (if you have saved the control script). This will prompt the execution of data preparation and modeling tasks. Alternatively, enter "2" (and a carriage return) to cancel the current run if additional edits are needed; then return to step 1 above.
5. Improper sequencing of the cursor placement in step 2 above will prompt the following steps:
    - If the cursor is inadvertently placed in the Console window, or another script is opened, after the control script text is executed, then a red warning message will appear in the Console: "*Please select current control.R file. Browser window may appear behind Rstudio.*" This message will occur because RSPARROW automatically finds the path to the user directory using the path associated with the active script in RSTUDIO (i.e., the script associated with the cursor location). However, if the user prematurely locates the cursor in the Console window, then the active script (*sparrow_control.R*) is not found. Thus, a backup process is enabled in which the user manually selects the active control script from a Windows browser pop-up.

- After some processing delay, use the pop-up browser window to locate and open the *sparrow_control.R* script (*note that the pop-up may appear behind the RStudio window*).
- After this, the user should continue with step 4. If the active control script is successfully selected in the pop-up window, then the user can proceed with the run normally by entering "1" in the Console window; otherwise, enter "2" in the Console window to cancel the current run, and begin again with step 1.

4.2.4 Checklist for setup and testing of system settings, control files, and new models

The checklist provides an ordered sequence of recommended steps for users to follow when initiating RSPARROW with new data input files and new RSPARROW models. Many of the steps can be implemented in a single execution of the control script (please note the exceptions in step 3). References are provided to the relevant section numbers of the control script, with specific settings given in selected cases.

To execute any of the steps in the checklist, users must always select and execute all statements in the control script. The easiest approach is to click on the "Source" button in the upper right of the control script window in RStudio (located next to the "Run" button), which will execute the entire script.

1. Execute at least one of the tutorial models (Chapter 6) to ensure that the required RSPARROW software (R, RStudio, R library dependencies) has been properly installed. See Chapter sub-section 1.3 for details on the software installation.
2. For a new model control script, set the path name for the RSPARROW Master directory (i.e., "RSPARROW_master") and User sub-directory names (e.g., "data", "gis", "results") or use a relative path name for the RStudio default working directory (section 11 of the control script). Execution of the control script in an RStudio session will automatically determine the location (path name) of the upper level user directory (e.g., "UserTutorial") and the sub-directory paths (e.g., "results", "gis", "data").
3. Setup the required control input files and data import options (section 1 of the control script; also see Chapter 3 for guidance on the required content of the control input files). The control settings in the control script section 3 define CSV file delimiters, CSV decimal symbols, and the name of the data1 input file. Optional settings are also available to create the initial versions of the CSV control input files for modeling; these use the column header variable names in the *data1.csv* file to setup columns in the files.

   3a. Users are required to create the *data1.csv* input file as a pre-processing step and to define its name in the control script (`input_data_fileName <- "data1.csv"`). Users should consult Table 4 in Chapter 3 for a listing of the system and parameter variables required for the setup and execution of RSPARROW models. This includes reach network navigation variables, including reach node variables and reach attribues such as the 'termflag' variable, which is necessary to define the primary terminal reaches at the outlets of all hydrologically independent drainages.

   3b. The setting `create_initial_dataDictionary<-"yes"` (see sub-section 4.4.1 for details) allows users to automatically create the *dataDictionary.csv* file, using column header names in the *data1.csv* file and internal RSPARROW system variable names. The file will be saved to the "User Directory/results" sub-directory where it must remain to be located by the control script. The file name is also fixed and should not be changed by users.

   - Execution of the control script fills-in the column information for the REQUIRED and FIXED system variable types (*varType*) in cases where the *data1.csv* header names match the pre-defined *sparrowNames* for these variable types (as shown in Table 4). Where no match is found for these variable types, an `NA` will appear in the *data1UserNames* column. All other variables in the *data1.csv* file will be assigned to the *data1UserNames* column with `NA` values for all other columns.
   - Additional editing of the *dataDictionary.csv* is required to fill-in the `NA` values by completing the variable types (*varType*) and internal R variable names (*sparrowNames*).

3c. The setting `create_initial_parameterControlFiles<-"yes"` (see sub-section 4.4.1 for details) allows users to automatically create the *parameters.csv* and *design_matrix.csv* control input files from the *dataDictionary.csv* file. The control files will be saved to the "User Directory/results" sub-directory where they must remain to be located by the control script. The file names are also fixed and should not be changed by users.

- Prior to executing this setting with the control script, users should edit the *dataDictionary.csv* to fill-in the *varType* for all variables. A minimum of one SOURCE *varType* is required to execute the control setting. In addition, one DELIVF *varType* is required to subsequently execute steps 5 and 6 below.
- Execution of the control script will create the parameter and design matrix files and fill-in in the *sparrowNames*, *parmType*, units, and description columns using the *dataDictionary.csv* definitions.
- Additional editing of the columns and cells in the newly-created parameter control files (*parameters.csv*, *design_matrix.csv*) will not be necessary to complete the setup and verification in steps 5 and 6 below. However, additional editing will be required to the columns in these files to execute user-specified models (see sub-section 4.4.4.3). The *parmUnits* and *description* columns in the *parameters.csv* file are optional but will be automatically filled according to values in the *dataDictionary.csv*.

3d. Once the *dataDictionary.csv* is finalized and the *data1.csv* file has been read into an RStudio session during the execution of a model (step 7 below), users can speed up the data input process when executing subsequent models. This is done by using the setting `load_previousDataImport<-"yes"` (section 1 of the control script), which accesses a saved binary version of the *data1.csv* file. The binary file contains *sparrowNames* from the *dataDictionary.csv* file. Thus, any edits to either file requires the recreation of the binary file uisng the `load_previousDataImport<-"no"` setting.

4. Check that the model parameters are properly defined in the parameter control files (*parameters.csv*, *design_matrix.csv*) and *userModifyData.R* script. Ensure that newly-created variables in the *userModifyData.R* script are present in the *dataDictionary.csv* if these variables are to be saved in the *subdata* object for subsequent use in functions or mapping. Also ensure that all "NA" values are converted to zeros (or "filled in" with appropriate estimates) for the explanatory variables associated with the user-selected model parameters (e.g., use the `replaceNAs(named.list(names))` RSPARROW function; see Chapter sub-section 3.5). Once the parameter control files are created, note that the following error messages may occur with the subsequent execution of the control script in cases where the parameters or variables have not been properly defined in the data dictionary or parameter files.

- If only a SOURCE *varType* but no DELIVF *varType* is defined in the *parameters.csv* and *design_matrix.csv* files, the following message appears. Note that a DELIVF variable only needs to be listed in the *parameters.csv* and *design_matrix.csv* files but does not have to be selected with a parameter for estimation in a model.

```
Reading parameters and design matrix...

ERROR: INVALID design_matrix.csv FILE '...results/design_matrix.csv'
CHECK NUMBER OF COLUMNS
design_matrix.csv FILE SHOULD HAVE THE FOLLOWING COLUMNS:
```

- If only a SOURCE *varType* but no DELIVF *varType* is defined in the *parameters.csv* file, the following message appears.

```
Reading parameters and design matrix...
AN ERROR OCCURRED IN PROCESSING selectDesignMatrix.R
Error in matrix(unlist(as.data.frame(dmatrixin)), ncol = adel, nrow = asrc) :
'data' must be of a vector type, was 'NULL'
RUN EXECUTION TERMINATED.
```

- If the *parmInit*, *parmMax*, and *parmMin* are set to 0, then the following message is printed:

```
Reading parameters and design matrix...
NO PARAMETERS FOUND FOR ESTIMATION IN PARAMETERS FILE.
ALL PARAMETERS FOUND HAVE parmMAX==0
EDIT PARAMETERS FILE TO RUN ESTIMATION
RUN EXECUTION TERMINATED.
```

- If no parameters are selected for estimation in RSPARROW by defining the minimum and maximum parameter values in the *parameters.csv* file, then the following message appears.

```
Reading parameters and design matrix...
AN ERROR OCCURRED IN PROCESSING selectParmValues.R
Error in if (abs(x[k]) >= b[i] & abs(x[k]) < b[i + 1]) { :
  missing value where TRUE/FALSE needed
RUN EXECUTION TERMINATED.
```

- In this example, the land-use variable 'forest' was not defined in the *dataDictionary.csv* file, which was required to use the variable in the *named.list* function in the *userModifyData.R* script. The following message appears and the CSV input files will popup for editing.

```
ERROR: forest PARAMETER NOT FOUND IN dataDictonary.csv
 forest HAS BEEN ADDED TO dataDictionary.csv
 USER MUST EDIT dataDictionary.csv, userModifyData.R, and design_matrix.R

TO ALLOW FOR NEW PARAMETER
DATA IMPORT MUST BE RE_RUN
SET run_dataImport<-'yes' AND load_previousDataImport<-'no'

USER MUST EDIT CONTROL FILES WITH MISSING PARAMETER INFORMATION
design_matrix.csv, dataDictionary.csv, and userModifyData.R ARE OPEN FOR EDIT
RUN EXECUTION TERMINATED
```

- In this example, the watershed classification variable 'huc2' did not appear in the *dataDictionary.csv*, which was required to use the variable as a classification variable with the control setting `classvar` in section 7 of the control script. The following error message appears.

```
Creating and Modifying subdata...
Testing for missing variables in subdata...
INVALID classvar huc2 NOT FOUND IN dataDictionary.csv
RUN EXECUTION TERMINATED
```

5. Users are required to convert the geographic shape files to binary R object files (section 8 of the control script). The R object files are required for mapping, including the mapping output in step 6. Two control settings govern the conversion: `if_create_binary_maps<-"yes"` (set to "no" in subsequent runs) and `convertShapeToBinary.list`.

6. Check the network reach connectivity and create new network variables (section 2 of the control script). The reach connectivity can be verified in cases where users provide an independent measure of the total drainage area (e.g., from a geographic information system), which is compared with the accumulated incremental drainage area for reaches using the RSPARROW algorithms. Users can also specify as many as three different reach network attributes for automated computation in RSPARROW (i.e., `calculate_reach_attribute_list <- c("hydseq","headflag","demtarea")`.

7. Determine whether the number of reaches needs to be reduced (section 2 of the control script) using the control setting `filter_data1_conditions`. For example, variables and conditional R operators can be specified with this setting if a user wants to apply the model to a smaller spatial domain than described by the reaches included in the *data1.csv* file.

8. Determine whether the number of calibration sites needs to be filtered (section 3 of the control script)

according to the headwater reach drainage area or the size of the incremental area or number of reaches separating calibration sites.

9. Setup the RSPARROW model specification and execute the model (section 4 of the control script). The model estimation sub-section 4.4.4 should be consulted for guidance on model specifications and variable selection options, model convergence and performance attributes, estimation methods, model assumptions, and diagnostics and their interpretation. Users should also consult the tutorial in Chapter 6, which illustrates the interpretation of a series of models that build in complexity. Additionally, users should consult the description of control script settings in sections 5 (diagnostics), 6 (validation sites), 7 (prediction), and 8 (prediction and diagnostic mapping). Users can also enable the R Shiny interactive DSS mapper during model runs (`enable_shinyMaps<-"yes"`; section 9) to assist with the visualization of spatial patterns in explanatory and response data and model predictions by monitoring sites, stream reaches and catchments.
10. Once exploratory RSPARROW models are evaluated and a final model is selected, users should assess the uncertainties of the parameters and predictions of the final model (section 10 of the control script).
11. RSPARROW models can be used to evaluate management decision-support scenarios (section 9 of the control script), such as the effects of hypothetical source reductions on stream loads. The decision-support scenarios can be evaluated for any model specification, using the R Shiny interactive mapper via the control setting `enable_shinyMaps<-"yes"` (section 9).

4.2.5 Loading a previous model's control settings and output files into RStudio

Users can perform additional model applications, such as predictions, mapping (interactive R Shiny or batch mode), or evaluations of management scenarios, without re-estimation of the model, by accessing the control settings and output files of a previous model in an RStudio session. Users can also explore new model specifications, using a prior model's control script settings and/or control input file settings (e.g., *parameters.csv*) as a starting point.

Two options exist to access a previous model's control settings and input files in RStudio:

1. Execute the `copy_PriorModelFiles<-"run_id"` control setting in the existing control script (*sparrow_control.R*) in an RStudio session. This setting is in section 11 of the active control script (also see Chapter sub-section 4.4.11).
    - Execution of the control script will overwrite the active control script (and all input control files) in the *"results"* directory with the versions of these files from the specified prior model. The control script from the prior model is then opened in the RStudio session and ready for further editing or execution.
    - Note that the path name specified for the `path_master` setting (section 11 of the control script) in the active control script will be retained and is not overwritten. Also, the *master_dataDictionary.csv* file and *input_data_fileName_priorImport* binary file (e.g., *data1_priorImport*) are not overwritten.
    - If the user is prompted to close the active control script (*sparrow_control.R*) while executing the `copy_PriorModelFiles` setting, note that this is required so that the active control script can be overwritten with the prior control script settings. Failing to close the active control script and all input CSV control files will cause an error in the `copy_PriorModelFiles` functionality, which could result in a mixture of old and new control files in the results directory. Note that only the active control script should be closed; the RStudio session should be kept open.
2. Using the Window's environment, manually copy the R control script and control input files from the previously executed model *"run_id"* sub-directory into the upper-level *"results"* directory.
    - Users must delete the existing control script and input files in the *"results"* directory, and remove the *"run_id"* prefix from the newly copied files. Do not delete the *master_dataDictionary.csv* file

and *input_data_fileName_priorImport* binary file (e.g., *data1_priorImport*). When the newly copied control script is opened in an RStudio session, it becomes the active control script.

- Users also have the option to selectively copy individual control files to access specific control settings of a prior model in an RStudio session, such as the previous parameter selections (*parameters.csv*) or user's R modification statements (*userModifyData.R*).

USER CAUTION: In the subsequent execution of the new, active control script file (with `copy_PriorModelFiles<-NA`), users should be careful in selecting an option for the `if_estimate` and `if_estimate_simulation` settings (section 4 of the control script):

- Set `if_estimate<-"no"` (and `if_estimate_simulation<-"no"`) to use the prior estimated (or simulated) model results to generate new predictions, maps, and management scenarios, or to enable the R Shiny interactive mapper. Thus, no model re-estimation is necessary in this case.
- Set `if_estimate<-"yes"` to re-estimate the prior model (or `if_estimate_simulation<-"yes"` to re-simulate the prior model). Note that this will automatically delete all files in the *"estimate"*, *"maps"*, *"predict"*, and *"scenarios"* directories.

4.3 Crosswalk linking the control settings with RSPARROW output

As a general guide, the crosswalk in Table 6 links the control script settings (and topical section numbers) with RSPARROW output files (datafiles and R binary objects). Users must specify supplementary settings in many of these sections to obtain complete functionality and to output all of the text and graphics.

Table 6. Listing of the datafiles and R objects output to RSPARROW directories by the control script settings. R objects (shown in **bold**) are stored as binary files; the objects contain dataframes or list objects within the R environment. File names shown in parentheses are user defined model names (i.e., "run_id"), scenario and variable names, or labels assigned during execution ("time").

Directory `Control Setting` (control script topical section)	Name of Datafile or R Binary Object
Directory: results	
`create_initial_dataDictionary` (1)	dataDictionary.csv
`create_initial_parameterControlFiles` (1)	parameters.csv
`create_initial_parameterControlFiles` (1)	design_matrix.csv
.	
Directory: results/(run_id)	
(`run_id`) (11)	(run_id)_dataDictionary.csv
(`run_id`) (11)	(run_id)_parameters.csv
(`run_id`) (11)	(run_id)_design_matrix.csv
(`run_id`) (11)	(run_id)_userModifyData.R
(`run_id`) (11)	(run_id)_userSettings.csv
.	
Directory: results/(run_id)/data	
(no condition)	**subdata**
`if_estimate` (4)	**sitedata**
`if_estimate` and `if_validate` (6)	**vsitedata**
.	
Directory: results/(run_id)/estimate	
`if_corrExplanVars` (5)	(run_id)_explvars_correlations.txt
`if_corrExplanVars` (5)	(run_id)_explvars_correlations.pdf
`if_corrExplanVars` (5)	**(run_id)_Cor.ExplanVars.list**
`if_verify_demtarea` (2) new drainage area differs	(run_id)_diagnostic_darea_mismatches.csv
from *demtarea* by >1%; `enable_plotlyMaps` (8)	(run_id)_diagnostic_darea_mismatches.html
`if_estimate` (4)	(run_id)_log.txt
`if_estimate` or `if_estimate_simulation` (4)	(run_id)_summary.txt

Directory `Control Setting` (control script topical section)	Name of Datafile or R Binary Object
`if_estimate` or `if_estimate_simulation` (4)	(run_id)_diagnostic_plots.html
`if_estimate` or `if_estimate_simulation` (4)	(run_id)_diagnostic_sensitivity.html
`if_estimate` or `if_estimate_simulation` (4)	(run_id)_residuals.csv
`if_estimate` or `if_estimate_simulation` (4)	**(run_id)_DataMatrix.list**
`if_estimate` or `if_estimate_simulation` (4)	**(run_id)_SelParmValues**
`if_estimate` or `if_estimate_simulation` (4)	**(run_id)_sparrowEsts**
`if_estimate` or `if_estimate_simulation` (4)	**(run_id)_Mdiagnostics.list**
`if_estimate` or `if_estimate_simulation` (4)	**(run_id)_ANOVA.list**
`if_estimate` or `if_estimate_simulation` (4)	**(run_id)_sensitivities.list**
`if_estimate` or `if_estimate_simulation` (4)	**(run_id)_JacobResults**
`if_estimate`, `ifHess` (4)	**(run_id)_HessianResults**
`if_estimate` or `if_estimate_simulation` (4), and `if_predict` (7)	(run_id)_summary_predictions.csv
`if_estimate` or `if_estimate_simulation` (4), and `if_spatialAutoCorr` (5)	(run_id)_diagnostics_spatialautocor.html
`if_estimate` or `if_estimate_simulation` (4), and `if_spatialAutoCorr` (5)	(run_id)_diagnostics_spatialautocor.txt
`if_estimate`, `if_boot_estimate`, `ifHess` (4)	(run_id)_bootbetaest.csv
`if_estimate`, `if_boot_estimate`, `ifHess` (4)	**(run_id)_BootBetaest**
`if_estimate` and `if_validate` (6)	(run_id)_validation_plots.html
`if_estimate` and `if_validate` (6)	**(run_id)_vMdiagnostics.list**
`if_estimate` and `if_validate` (6)	**(run_id)_vANOVA.list**
`NLLS_weights` for `lnload` or `user` options (4)	(run_id)_weights.pdf
. Directory: results/(run_id)/maps . . . (/Stream). . . (/Catchment)	
`master_map_list`, `output_map_type`, and `enable_plotlyMaps` (8)	(run_id)_(variable_name).html (.pdf)
. Directory: results/(run_id)/maps/Interactive . . . (/Stream). . . (/Catchment). . . (/SiteAttributes)	
`enable_shinyMaps` (9)	(run_id)_(variable_name).html (.pdf)
. Directory: results/(run_id)/maps/ESRI_ShapeFiles/ prediction; `outputESRImaps` (8)	lineShape.(dbf,prj,shp,shx), polyShape.(dbf,prj,shp,shx)
Directory: results/(run_id)/maps/ESRI_ShapeFiles/ residuals; `outputESRImaps` (8)	residShape.(dbf,prj,shp,shx)
. Directory: results/(run_id)/maps/ESRI_ShapeFiles/ siteAttributes; `outputESRImaps` (8)	siteAttrshape.(dbf,prj,shp,shx)
. Directory: results/(run_id)/predict	
`if_predict` (7)	(run_id)_predicts_load.csv
`if_predict` (7)	(run_id)_predicts_load_units.csv
`if_predict` (7)	(run_id)_predicts_yield.csv
`if_predict` (7)	(run_id)_predicts_yield_units.csv
`if_predict` (7)	**(run_id)_predict.list**
`if_predict` (7), `if_boot_predict` (10)	(run_id)_predicts_load_boots.csv

Directory `Control Setting` (control script topical section)	Name of Datafile or R Binary Object
`if_predict` (7), `if_boot_predict` (10)	(run_id)_predicts_yield_boots.csv
`if_predict` (7), `if_boot_predict` (10)	**(run_id)_predictBoots.list**
`if_predict` (7), `if_boot_predict` (10)	**(run_id)_BootUncertainties**
.	
Directory: results/(run_id)/scenarios/(scenario_name)	
`if_predict_scenarios`, `scenario_name` (9)	(scenario_name)_(run_id)_predicts_load_scenario.csv
`if_predict_scenarios`, `scenario_name` (9)	(scenario_name)_(run_id)_predicts_load_scenario_units.csv
`if_predict_scenarios`, `scenario_name` (9)	(scenario_name)_(run_id)_predicts_yield_scenario.csv
`if_predict_scenarios`, `scenario_name` (9)	(scenario_name)_(run_id)_predicts_yield_scenario_units.csv
`if_predict_scenarios`, `scenario_name` (9)	(scenario_name)_(run_id)_predicts_loadchg_scenario.csv
`if_predict_scenarios`, `scenario_name` (9)	(scenario_name)_(run_id)_predicts_yieldchg_scenario.csv
`if_predict_scenarios`, `scenario_name` (9)	(scenario_name)_(run_id)_scenario_metainfo.txt
`if_predict_scenarios`, `scenario_name` (9)	**batch_(time).RData**
`if_predict_scenarios`, `scenario_name` (9)	**(scenario_name)_(run_id)_DataMatrixScenarios.list**
`if_predict_scenarios`, `scenario_name` (9)	**(scenario_name)_(run_id)_predictScenarios.list**
`if_predict_scenarios`, `scenario_name` (9)	**(scenario_name)_(run_id)_predictScenarios.list**
.	
Directory: results/(run_id)/scenarios/(scenario_name) . . . (/Stream). . . (/Catchment)	
`if_predict_scenarios` (9), `scenario_name` (9), `enable_plotlyMaps` (8), and `enable_shinyMaps` (9)	(scenario_name)_(run_id)_(variable_name).pdf(.html)
.	
Directory: results/(run_id)/batchSessionInfo	
`batch_mode` (11)	**(run_id).RData**
`batch_mode` (11)	(run_id)_log.txt

4.4 Explanation of control settings by topical category

Most of the control settings are self explanatory using the comments provided in the *sparrow_control.R* script. Additional explanations are provided below for each of the topical categories and settings, including discussion of the statistical methods and reference to the supporting documentation for the methods. This expands on the information provided in the control script. The following narrative also highlights cases where combinations of various settings can be used to ensure that user preferences are executed when using RSPARROW.

4.4.1 Data import and parameter control file setup options (section 1 of control script)

```
#-------------------------------------------------------------------------------
   #Set CSV read/write options
   csv_decimalSeparator <- "."
   csv_columnSeparator <- ","
```

The column and decimal separators for the CSV (Comma Separated Values) files are flexible to accommodate differences in standards that vary internationally. User settings for the separators are implemented in RSPARROW for both reading and writing CSV files.

```
#-------------------------------------------------------------------------------
   #Create an initial Data Dictionary file from the data1 column names
   #This file will have to be edited prior to executing RSPARROW
   create_initial_dataDictionary<-"yes"
```

The setting allows users to automatically create the *dataDictionary.csv* file, using column header names in the *data1.csv* file and internal RSPARROW system variable names. The *dataDictionary.csv* will be saved to the "User Directory/results" sub-directory where it must remain to be located by the control script. The file name is also fixed and should not be changed by users.

Execution of the control script fills-in the column information for the REQUIRED and FIXED system variable types (*varType*) in cases where the *data1.csv* header names match the pre-defined *sparrowNames* for these variable types (as shown in Table 4). Where no match is found for these variable types, an `NA` will appear in the *data1UserNames* column. All other variables in the *data1.csv* file will be assigned to the *data1UserNames* column with `NA` values for all other columns.

Additional editing of the *dataDictionary.csv* is required to fill-in the `NA` values by completing the variable types (*varType*) and internal R variable names (*sparrowNames*). Users should assign the *data1UserNames* variable name to missing entries of the *sparrowNames* for other than REQUIRED and FIXED *varType* variables. It is recommended that users identify the *varType* for the largest possible set of model explanatory variables (i.e., SOURCE, DELIVF, STRM, and RESV) before attempting to create the parameter control files (at a minimum, one SOURCE and DELIVF *varType* must be added to the file; see description in step 3c).

We recommend that users add variable names to the *sparrowNames* column that are not present in the *data1.csv* file but created in the *userModifyData* script; a `NA` should be assigned to the *data1UserNames* for these variables. This recommendation includes adding the definition of land-use variables, which are used in the model diagnostics. The variable units (*varunits*) and descriptive information (*explanation*) are optional but recommended.

Note that execution of the setting `create_initial_dataDictionary<-"yes"` will print the message `NO PARAMETERS FILE FOUND IN RESULTS DIRECTORY.`, and will request that you save the active control file (i.e., `Did you save the active control file...`). After the control script is saved and a `1` entered, the message will list the `MISSING REQUIRED sparrowNames :` and the `MISSING FIXED sparrowNames :`, before printing `RUN EXECUTION TERMINATED`. The *dataDictionary.csv* file will also pop up to allow user editing.

```
#-------------------------------------------------------------------------------
   #Create an initial parameter and design_matrix files from names in the Data Dictionary
```

```
   #file. The parameter names must be listed for both the sparrowNames and data1UserNames
   #and the varType should be defined as SOURCE, DELIVF, STRM, or RESV to populate
   #parmType in the (run_id)_parameters.CSV
   #The initial file will have to be edited prior to executing RSPARROW
   create_initial_parameterControlFiles<-"yes"
```

The setting allows users to automatically create the *parameters.csv* and *design_matrix.csv* control input files from the *dataDictionary.csv* file. The *parameters.csv* and the *design_matrix.csv* will be saved to the "User Directory/results" sub-directory where they must remain to be located by the control script. The file names are also fixed and should not be changed by users.

Execution of the control script will create the parameter and design matrix files and fill-in in the *sparrowNames*, *parmType*, units, and description columns using the *dataDictionary.csv* definitions.

Prior to executing this setting with the control script, users should edit the *dataDictionary.csv* to fill-in the *varType* for all variables. A minimum of one SOURCE *varType* is required to execute the control setting. In addition, one DELIVF *varType* is required to subsequently execute steps 5 and 6 in the checklist in sub-section 4.2.4. Users should assign the *data1UserNames* variable name to missing entries of the *sparrowNames* for all variables other than those with a *varType* designation of REQUIRED or FIXED.

In cases where the parameters file is missing, execution of the setting will print the message `NO PARAMETERS FILE FOUND IN RESULTS DIRECTORY.`, and will request that you save the active control file (i.e., `Did you save the active control file...`). After the control script is saved and a `1` entered, the message will indicate that the control parameter files are available for editing by printing: `INITIAL DESIGN_MATRIX FILE : ...design_matrix.csv AVAILABLE FOR EDIT` and `INITIAL PARAMETERS FILE : ...parameters.csv AVAILABLE FOR EDIT`, before printing `RUN EXECUTION TERMINATED`. Both of the parameter input control files will pop-up, ready for user editing.

In cases where the parameters file already exists, execution of the setting will not overwrite the file. A message will appear stating `(parameters.csv filepath) ALREADY EXISTS. NEW PARAMETERS FILE NOT CREATED. SET create_initial_parameterControlFiles<-'no' to RUN RSPARROW WITH CURRENT PARAMETERS FILE. RUN EXECUTION TERMINATED`. To create a new parameters file, the old one must be deleted first.

Additional editing of the columns and cells in the newly-created parameter control files (*parameters.csv*, *design_matrix.csv*) will not be necessary to complete the setup and verification in steps 5 and 6 in the checklist in sub-section 4.2.4. However, additional editing will be required to the columns in these files to execute user-specified models (see sub-section 4.4.4.3). The *parmUnits* and *description* columns in the *parameters.csv* file are optional but will be automatically filled according to values in the *dataDictionary.csv*.

```
#-------------------------------------------------------------------------------
   #Select input data file (accepted file types ".csv" or binary file
   #  with no extension created in previous RSPARROW run).
   #Binary file will be automatically created if file type is not binary
   #  for fast import in subsequent runs.
   input_data_fileName <- "data1.csv"
```

The `input_data_fileName` file provides the core geospatial data for navigating the reach network, estimating the model, and performing model applications (e.g., predictions, mapping). This file must reside in the "User Directory/data" sub-directory. The file can be used with the control script settings `create_initial_dataDictionary` and `create_initial_parameterControlFiles` to automatically generate the other required control input CSV files.

In the initial execution of the control script, users should use the `input_data_fileName` file setting with the *csv* extension. To speed up data input in subsequent executions of the control script, users have the option to read the binary version of this file, which is automatically created in the "User Directory/results" sub-directory when a *csv* extension is used in the control script. The binary version is saved in the "User Directory/data" sub-directory and can be loaded by removing the *.csv* form the `input_data_fileName`

setting. Any updates to the CSV version of the `input_data_fileName` file will require an execution of the control script with the CSV extension to recreate the binary version of the file.

```
#-------------------------------------------------------------------------------
   # Loads previous binary data input file "(input_data_fileName)_priorImport"
   #  from results directory. This setting will override run_dataImport.
   # NOTE: The OPTIONS FOR NETWORK ATTRIBUTES AND AREA VERIFICATION (section 2)
   #     will ONLY be executed if load_previousDataImport<-"no"
   load_previousDataImport<-"yes"
```

The setting `load_previousDataImport<-"yes"` is designed to speed up data input during the model execution and development phase by allowing access to a saved binary version of the `input_data_fileName`.*csv* input file (e.g., *data1.csv*) in the "User Data/results" sub-directory; the binary file is called `input_data_fileName`_*priorImport*.

The setting `load_previousDataImport<-"no"` imports the `input_data_fileName`.*csv* input file (e.g., *data1.csv*) or the binary version stored in the data sub-directory if no *.csv* extension is specified, which includes the reach network data with the variable definitions listed in the *dataDictionary.csv* file. This setting is required to allow the creation of user-requested network variables as specified by the setting `calculate_reach_attribute_list` in section 2 of the control script. The binary file `input_data_fileName`_*priorImport* is then automatically saved, overwritting any prior versions of this file that may exist. The binary file will include the renumbered *waterid* variable in cases where the number of reaches exceeds one million.

The setting `load_previousDataImport<-"no"` should be used until the *dataDictionary.csv* and `calculate_reach_attribute_list` settings are finalized and the `input_data_fileName`.*csv* input file (e.g., *data1.csv*) has been read into an RStudio session by executing the model control script.

Any changes in the `input_data_fileName`.*csv*, the `calculate_reach_attribute_list` settings, or the *dataDictionary.csv* file require use of the control setting `load_previousDataImport<-"no"` to recreated the binary input file.

```
#-------------------------------------------------------------------------------
   #Indicate whether or not to run _userModifyData.R
   if_userModifyData<-"yes"
```

This setting executes the function *userModifyData.R* (located in the "User Directory/results" sub-directory). The function supports user-defined data calculations and conditional statements with existing or newly-created variables in the `input_data_fileName` file (e.g., *data1.csv*). Users may also use R statements in the function to filter calibration sites (e.g., eliminate sites or their associated loads that are deemed unacceptable for use in the model calibration). See Chapter sub-section 3.5 for details on the setup and use of this function, including examples of R statements that can be used with RSPARROW models.

4.4.2 Stream network attributes, verification, and reach filtering (section 2 of control script)

```
#-------------------------------------------------------------------------------
  #   NOTE: This section is only executed if data import of the CSV file is run.
  #         To run data import, set load_previousDataImport<-"no"

  # Verify drainage area accumlation in the network
  #   NOTE: This requires that precalculated values of 'demtarea' are present in DATA1
  #         Area totals will be affected by 'frac'
  if_verify_demtarea <- "yes"

   #Indicate whether maps are to be generated for if_verify_demtarea<-"yes"
   #   NOTE: Generating maps can significantly slow processing time for larger models
   if_verify_demtarea_maps<-"yes"
```

Checks on reach network connectivity and area accumulation are performed with these control settings. The checks are an important step prior to developing an RSPARROW model to ensure that the node topology is correctly specified. For these settings to be operable, the input data file specified by the control setting `input_data_fileName` must be imported to the RStudio session using the setting `load_previousDataImport<-"no"`.

In cases where the control setting `if_verify_demtarea<-"yes"` and the user supplies an independently calculated reach total drainage area (*demtarea*), RSPARROW performs a verification of the total drainage area (*demtarea*) and reach connectivity by comparing the user-supplied total drainage area values with RSPARROW determined estimates of the total drainage area. The RSPARROW newly-derived estimates are determined by summing the incremental drainage area (*demiarea*) for reaches using the RSPARROW area/load-accumulation function with the to-from node structure in the user's `input_data_fileName` (e.g., *data1.csv*).

For reaches where the RSPARROW newly-derived total drainage area differs from the user-supplied *demtarea* by more than 1 percent, a CSV file is output with selected reach attribute information. The control setting `if_verify_demtarea_maps<-"yes"` will also produce a PDF output file of at least four maps with selected reach attribute metrics; this is optional to allow user control over the output of maps for larger watersheds that will require additional processing time.

The functions enabled by this setting will operate on a renumbered *waterid* variable in cases where the number of reaches exceeds one million.

For the control setting `if_verify_demtarea<-"yes"`, the messages below are written to the RStudio Console window. This includes a listing of the the number of reaches with differences in drainage areas, based on a comparison of the newly-computed areas to the user-provided total drainage area.

```
Reporting checks of total drainage area...
Number of reaches with different (>1%) total drainage areas
       (see records in DAreaFailCheckObj): 99

Writing results from drainage area comparisons (CSV, PDF maps) in estimate directory...
```

.

```
#-------------------------------------------------------------------------------
  # Request the calculation of selected reach attributes:
  #  Identify the attributes for calculation and placement in DATA1
  #  Select from following: 'hydseq', 'headflag', and/or 'demtarea'

  #calculate_reach_attribute_list <- c("hydseq","headflag","demtarea")  # calculate for
  #                                                                     these variables
  #calculate_reach_attribute_list <- c("hydseq")  # calculate for these variables
  calculate_reach_attribute_list <- NA    # no calculation is requested
```

Users can optionally use RSPARROW to determine values for as many as three reach attributes, *hydseq*, *headflag*, and *demtarea*. The values are determined using the RSPARROW "hydseq" function (*hydseq.R*), an R translation of a SAS subroutine available with the SAS SPARROW software (G.E. Schwarz, 2017, personnal communication). These attributes are defined as: *demtarea* is the RSPARROW total drainage area, *hydseq* is the unique hydrological sequence number, and *headflag* is the headwater identifier value (1=headwater reach; 0=non-headwater reach).

Note that the reach order associated with the RSPARROW calculated *hydseq* variable can potentially differ from the sequential reach order associated with a user-supplied *hydseq* variable. Despite potential differences in the sequential order, both *hydseq* variables can produce an accurate hydrological ordering of the reaches, which will yield identical summations of the loads at reach nodes and model estimation results. However, the numerical sequencing of calibration and validation monitoring sites is hydrologically ordered (from upstream to downstream) and sensitive to the numerical sequencing of the reaches. Therefore, the order of the monitoring

sites can potentially differ for models that are based on different methods of obtaining the *hydseq* values.

```
#-------------------------------------------------------------------------------
  # Specify any additional DATA1 variables (and conditions) to be used to filter reaches:
  #  Default conditions are FNODE > 0 and TNODE > 0
  #  The filter is used to create 'subdata' object from 'data1'
  filter_data1_conditions <- c("data1$drainden > 0 & !is.na(data1$drainden)")
  filter_data1_conditions <- NA
```

This setting allows users to subselect reaches from the set of reaches provided in the `input_data_fileName` (e.g., *data1.csv*) for use in model calibration and prediction. The filtering conditions are applied when the *subdata* R object is created (in the function *createSubdataSorted.R*; this occurs prior to execution of user-specified R statements in the *userModifyData.R* script). The object *subdata* is used for model calibration and other RSPARROW model applications (e.g., prediction, management scenario evaluations). By default, RSPARROW requires that only reaches with node values (*tnode*, *fnode*) greater than one are used when creating the *subdata* R object. Other reach attributes may be specified in the control setting to ensure that the spatial domain of the model includes valid reach attributes, or if a user wants to estimate the model for a smaller spatial domain than described by the reaches included in the *data1.csv* file. See the example R statement syntax given above in the comments section.

```
#-------------------------------------------------------------------------------
  # Indicate whether hydseq in the DATA1 file needs to be reversed
  if_reverse_hydseq <- "no"
```

Users have the option to reverse the order of the *hydseq* values to account for numerical sequences with negative signs or reverse ordering (e.g., NHD reach network).

4.4.3 Monitoring and site filtering options (section 3 of control script)

```
#-------------------------------------------------------------------------------
  # The index 'calsites' requires setting in the 'userModifyData.R' script
  #   to exclude unwanted sites.
  # (calsites:  1=site selected for calibration; 0 or NA = site not used)
  # Default setting = 1 for sites with positive loads 'depvar>0'

  # Minimum drainage area size for monitoring sites located on headwater reaches
  minimum_headwater_site_area <- 0

  # Minimum number of reaches separating monitoring sites
  minimum_reaches_separating_sites <- 1

  # Minimum allowable incremental drainage area between monitoring sites
  minimum_site_incremental_area <- 0
```

RSPARROW provides a REQUIRED calibration site indicator, *calsites* (see Table 4), that controls the monitoring site selection for use in the model calibration. R statements (i.e., "ifelse") can be added to the *userModifyData.R* script to filter (eliminate) sites that are used in model calibration (see examples in chapter sub-section 3.5).

Three additional (optional) automated site-selection filters can also be executed using the above control settings; explanations are given in the associated comments. The filters based on the incremental reach drainage area (`minimum_site_incremental_area`) and the number of reaches separating monitoring sites (`minimum_reaches_separating_sites`) are especially useful to reduce the number of model calibration monitoring sites in cases where the monitoring station network may be overly dense. The load estimates of closely spaced monitoring sites may be highly correlated and can contribute to larger spatial correlation in the model errors than is statistically desirable. These options also allow users to test the sensitivity of model

outcomes to these site density properties.

For the `minimum_reaches_separating_sites` control setting, a value of "N" for the setting (for N>0) will allow a minimum of N-1 reaches to exist between the reaches with calibration monitoring sites. For example, N=1 (and also N=0) will allow calibration monitoring sites to be located on an adjacent upstream or downstream reach; N=2 or a larger value will ensure that monitoring sites in adjacent reaches are not selected as calibration sites and at least one reach separates nearby monitoring sites (the most downstream monitoring site is selected in cases where the number of reaches separating sites falls below the minimum threshold).

The `minimum_headwater_site_area` control setting defines the minimum allowable drainage area for monitoring sites on headwater reaches. This can be used to eliminate monitoring sites on reaches with very small drainage areas, which are potentially associated with large model errors. Because the model assumes that monitoring sites are located at the downstream end of the reach, model prediction errors are potentially sensitive to headwater monitoring sites with actual drainage areas that differ appreciably from the total drainage area of the reach. As a precaution, users have the option to eliminate such sites from a model calibration or test the sensitivity of model outcomes to this property.

These three control setting filters are stored in the *min.sites.list* object for import into all necessary RSPARROW functions.

4.4.4 Model estimation (section 4 of control script)

4.4.4.1 Model specification settings

Conceptually, SPARROW predicts the contaminant load exported from an outlet of a stream reach as a function of the sum of two components: (1) the load generated within upstream reaches and transported through the stream network to the reach outlet, and (2) the load that originates within the incremental catchment of the reach and is delivered to the reach outlet.

The generation and transport of contaminant loads according to this conceptual framework is controlled by three major process components (see equation 1.27; Schwarz et al., 2006): contaminant source generation, land-to-water delivery, and stream and reservoir transport and decay. These components are contained within a non-linear and mechanistic model structure that includes mass balance constraints and non-conservative transport features. This includes factors that control the attenuation and delivery of contaminants to streams via surficial and subsurface pathways and the removal of contaminants in streams and reservoirs, according to first-order decay kinetics.

The explanatory variables for the source, land-to-water delivery, and aquatic components of the model are associated with rate coefficients that are statistically estimated during model calibration.

4.4.4.1.1 Land-to-water delivery variables

The land-to-water delivery component of SPARROW is a source-specific function of the delivery variables and an estimated coefficient associated with each delivery variable (see equation 1.28 in Schwarz et al., 2006). Land-to-water delivery variables (DELIVF *parmType* in the *parameters.csv* file) typically reflect climatic or landscape characteristics of catchments that potentially attenuate or accentuate the delivery of contaminants to streams via surficial and subsurface transport pathways. The variables may include, for example, precipitation, temperature, soil properties (e.g., permeability), topography (e.g., slope), physiography, or hydrology (baseflow index, saturation excess overland flow). The interaction of the delivery variable with specific sources is controlled by the user-specification of the interaction in the *design_matrix.csv* file (Chapter sub-section 3.4).

It is generally preferable to express the land-to-water variables according to a measure of intensity or density (e.g., percent of area, mass quantity per unit area) to provide a standardized and symmetrical expression of the variable in the commonly used exponential functional form (Schwarz et al., 2006; equation 1.28). For certain variables, a logarithmic transform of the delivery variable may be required to reduce the influence of more extreme values and produce the desired symmetrical distribution of the values. For a log-transformation, the land-to-water coefficient can be interpreted as the percent change in load delivered to streams, associated

with all sources interacting with the land-to-water variable, that results from a one-percent change in the land-to-water delivery variable (Schwarz et al., 2006).

```
#-------------------------------------------------------------------------------
  # Specify the land-to-water delivery function.
  #  The exponential computation yields an NxS matrix, where N is the number of
  #  reaches and S is the number of sources (see Chapter 4.4.4.1 for details).

  incr_delivery_specification <- "exp(ddliv1 %*% t(dlvdsgn))"
```

This setting allows users to specify the mathematical expression for the land-to-water delivery function. The default function as shown above computes an exponential of the product of the delivery variable matrix (`ddliv1`) and the transpose of the source-delivery interaction matrix (`dlvdsgn`); this yields a NxS matrix, where N is the number of reaches and S is the number of model sources. The delivery variable matrix `ddliv1` is computed as the product of the reach delivery variables and their associated estimated model coefficients, and has the dimensions of NxD, where D is the number of land-to-water delivery variables. The `dlvdsgn` matrix is a SxD matrix.

The exponential land-to-water delivery expression is the most commonly used mathematical form for this specification by USGS modelers. For experienced R users interested in executing an alternative specification, the RSPARROW model estimation function (*estimateFeval.R*) should be examined to view the syntax of the R statements used to execute the land-to-water delivery specification.

```
#-------------------------------------------------------------------------------
  # Specify if the delivery variables are to be mean adjusted (recommended). This
  #  improves the interpretability of the source coefficients */
  if_mean_adjust_delivery_vars <- "yes"
```

This setting performs a mean-adjustment to the land-to-water delivery variables by subtracting the mean of the variable over the full spatial domain for each reach-level value of the variable. Mean-adjustment is recommended to ensure that the reported values of the mean coefficients for the sources can be more readily compared and interpreted for a single model or across different models. The adjustment standardizes each source coefficient to the mean of the land-to-water delivery values for the full spatial domain. The execution of a model without the mean adjustment allows the source coefficients to be influenced by the particular spatial distribution of the land-to-water delivery values across the domain, which can complicate inter- and intra-model comparisons of source coefficients.

Subsequent to model execution, the mean-adjusted land-to-water delivery variables (and all other variables required to execute a model) are stored in the *estimate* subdirectory in the R binary object *DataMatrix.list* (see Chapter sub-section 5.2.3.1 for details on the content of the object).

4.4.4.1.2 Aquatic decay variables

```
#-------------------------------------------------------------------------------
  # Specify the R reach decay function code.
  reach_decay_specification <- "exp(-data[,jdecvar[i]] * beta1[,jbdecvar[i]])"
```

This setting defines the reach decay equation for calculating the net mean annual in-stream attenuation of contaminant load as a function of the water time-of-travel (`jdecvar`) and an estimated loss rate coefficient(s) (`jbdecvar`). The first-order rate coefficient can be expressed as a volumetrically weighted reaction rate constant (equation 1.30, Schwarz et al., 2006) or as a mass-transfer rate coefficient (equation 1.32, Schwarz et al., 2006). Both rates can be estimated using the same reach decay specification setting shown above. Section 3.5 provides examples of the R computations and functional expressions necessary to obtain the properly specified state variables for these two rate expressions. R code in the *userModifyData.R* script for the tutorial models can also be consulted for examples. For experienced R users, the RSPARROW model estimation function (*estimateFeval.R*) can be examined to view details of the R statements used to execute the reach decay specification.

```
#------------------------------------------------------------------------------
  # Specify the R reservoir decay function code.
  #reservoir_decay_specification <- "exp(-data[,jresvar[i]] * beta1[,jbresvar[i]])"
  reservoir_decay_specification <- "(1 / (1 + data[,jresvar[i]] * beta1[,jbresvar[i]]))"
```

This setting provides a reservoir decay equation for the net mean annual attenuation of contaminant load in reservoirs. The contaminant loss is modeled assuming steady-state and uniformly mixed conditions in reservoirs. The estimated first-order loss rate coefficient is expressed as a mass-transfer coefficient (unit=length per time; `jbresvar`), according to equation 1.34 in Schwarz et al. (2006). The state condition (explanatory variable) for the reservoir is the areal hydraulic load (i.e., water flushing rate; `jresvar`), expressed as the ratio of the outflow discharge to the water body surface area). Section 3.5 provides R code for computing the areal hydraulic load. R code in the *userModifyData.R* script for the tutorial models can also be consulted for examples. For experienced R users, the RSPARROW model estimation function (*estimateFeval.R*) can be examined to view details of the R statements used to execute the reservoir decay specification.

```
#------------------------------------------------------------------------------
  # An alternative simplified Arrhenius temperature dependent function can also be used
  #  as described in equation 1.35 in the SPARROW documentation (Schwarz et al. 2006).
  #  This requires the specification of an OTHER class variable for temperature.
  # OTHER variable (e.g., temp)
  reservoir_decay_specification <- "(1 / (1 + data[,jresvar[i]] * beta1[,jbresvar[i]])) *
      (beta1[,jbothervar[1]]+1)**data[,jothervar[1]]"
```

This setting provides a modified version of the reservoir decay specification. This illustrates how the equation can be modified with additional explanatory variables (and associated estimated coefficients) to account for interactions of the hydraulic properties (areal hydraulic load, water time-of-travel) with other physical or chemical properties, such as temperature. This modification equally applies to the stream decay specification.

One example as illustrated above is a simplified Arrhenius temperature-dependent function, specified as described in equation 1.35 in the SPARROW documentation (Schwarz et al., 2006). This requires the specification of an OTHER type (*parmType*) variable (see Table 4) for temperature in the *dataDictionary.csv* and other input control files.

The above equation for the decay specification setting defines an additional expression in which an estimated Arrhenius coefficient is raised to a power value expressed by the temperature (typically standardized relative to a fixed value; e.g., 20 degress C.). The OTHER class variables (e.g., temperature and other properties) are stored in the vector `jothervar`; the corresponding estimated coeffficients are stored in the `jbothervar` vector, with the index referencing the parameter vector sequence, corresponding to the order of the OTHER variables listed in the *parameters.csv* file.

The numerical value of 1.0 is added to equation (1.35) so that the t-statistic accurately evaluates deviations of the coefficient from zero; the Arrhenius coefficient is typically evaluated in relation to unity rather than zero. Estimated values of the Arrhenius coefficient above zero indicate a positive relation between the loss rate and temperature (the expected sign of most temperature-dependent reactions); values below zero indicate a negative relation.

4.4.4.2 Model estimation method and execution

```
#------------------------------------------------------------------------------
  # Specify if estimation is to be performed
  #  ("no" obtains coefficient estimates from a previous estimation;
  #    if no previous estimates available, then the initial coefficient values
  #    in the beta file are used)
  # "yes" indicates that all estimation, prediction, maps, and scenario files
      #from the subdirectory with the name = run_id will be deleted
      #and only regenerated if settings are turned on
```

```
  if_estimate <- "yes"
```

`if_estimate<-"no"` - This setting automatically loads the prior model estimation results from binary files if a previously estimated model is specified by the `run_id` setting. This allows users to efficiently execute model predictions and uncertainties or to map predictions and evaluate source-change scenarios, without any re-estimation of the model.

`if_estimate<-"yes"` - This setting estimates the user-selected model parameters using NonLinear Least Squares (NLLS) optimization methods. The use of this setting with a model `run_id` name for a previously executed model will delete all previous files and directories in the estimate, prediction, mapping, and scenario folders prior to model execution.

Estimation of the model parameters employs NLLS optimization methods using the function *nlfb* from the *nlmrt* R package (Nash, 2016). The function uses Marquardt-Levenburg methods (as used in SAS SPARROW). NLLS estimation methods are iterative and require user-specified initial values (*parmInit*) for the model parameters. The *nlfb* function allows constrained optimization of parameters, with user-specified lower and upper bounds (*parmMin*, *parmMax*) on the parameters.

Model estimation uses the conditioned model predictions, which are calculated by substituting the observed station load for the model predicted load on reaches with monitoring stations (Schwarz et al., 2006). Model performance (see sub-section 4.4.4.5) is reported for both the conditioned and unconditioned (simulated) model predictions. Model prediction output also includes both conditioned and unconditioned predictions (for details on the prediction types, see sub-section 4.4.7).

For each model iteration during execution, the *nlfb* function prints a progress report to the RStudio Console window as shown in the example below (the report is also saved to the *(run_id)_log.txt* file). This includes the Sum of Squares of Error (`SS`), the estimates of the parameters (`at = ...`), and a gradient measure. Parameter estimation proceeds until a negligible change is detected in all parameter values, as reported by the *nlfb* "`No parameter change`" message. Iteration 38 in the example applies to the evaluation of the model residuals (`SS`), whereas iteration 23 applies to the evaluation of the Jacobian gradients (the first-order partial derivatives of the model residuals with respect to the model coefficients).

```
lamda: 175.9219  SS= 115.7059  at  = 7.895277  = 5.143434  = 3.025148  = 1.189804
                                   = 7.39913  = 1.421906  = 0.1565517  = -3.554747
                                   = 1.114159  = 1.334169  = 0.4323695  = 0.2364273
                                   = 6.768724                                  38 / 23
roff = 1.822244e-05   converged =  FALSE
gradient projection =  -0.0009375305  g-delta-angle= 123.139
Stepsize= 1
No parameter change
```

```
#-------------------------------------------------------------------------------
  # NLMRT optimization shift setting that tests floating-point equality
  # Initially select values between 1.0e+12 and 1.0e+14
  # Lower magnitude offset values will execute more NLLS iterations
  s_offset <- 1.0e+14
```

This setting controls the floating-point equality offset threshold (*nlfb* 'offset'), which allows users to indirectly control the magnitude of the change in parameter values that terminates model execution. Lower values of `s_offset` allow model execution to proceed with more iterations, with evaluations of smaller parameter changes that potentially yield smaller `SS`. Values of `s_offset` between 1.0e+12 and 1.0e+14 are recommended. With the exception of this setting, all other *nlfb* defaults are used. Note that the *nlfb* function reports model convergence results for a test of the relative offset orthogonality type (`roff`). This typically indicates non-convergence (`converged = FALSE`) for RSPARROW models (see example above), based on the *nlfb* default setting. However, the test does not control termination of model estimation.

.

4.4.4.3 Selecting explanatory variables and parameter settings for model estimation

The inclusion of explanatory variables in a model is controlled through user settings of parameter attributes in the *parameters.csv* file. The contents of the file include initial parameter values (*parmInit*), minimum (*parmMin*) and maximum (*parmMax*) values (lower/upper bounds), parameter names (*sparrowNames*), and other parameter attributes (see Table 5 in sub-section 3.3 for details).

The following conditions control the inclusion/exclusion of parameters (i.e., explanatory variables) in a model:

- Parameters are included and estimated in the model in cases where *parmMin* is less than *parmMax*.
- Parameters are excluded from the model in cases where both *parmMin* and *parmMax* are equal to a value of zero or missing (NA).
- Parameters are included in a model with a "fixed" (i.e., constant and unestimated) value in cases where *parmInit*, *parmMin*, and *parmMax* equal the identical non-zero value. This will exclude the parameter from the model estimation but include the parameter when making model predictions.

The recommended parameter settings for explanatory variables to be estimated in an RSPARROW model are listed below in Table 7.

Table 7. Recommended parameter settings in the *parameters.csv* file. Note that at least one of the initial values for the model must be non-zero for the *nlfb* function to execute without error. *Fixed Parameters* are set to a constant non-zero value and included when making model predictions.

Parameter Type (*parmType*)	Initial Value (*parmInit*)	Minimum/Lower Bound (*parmMin*)	Maximum/Upper Bound (*parmMax*)
Estimated Parameters			
DELIVF	0	large negative value (e.g., -10000)	large positive value (e.g., 10000)
SOURCE	small positive value	0	large positive value (e.g., 10000)
STRM	small positive value	0	large positive value (e.g., 1000)
RESV	small positive value	0	large positive value (e.g, 1000)
Excluded Parameters	0 or NA	0 or NA	0 or NA
Fixed Parameters	*parmInit=parmMin=parmMax*		

The following guidelines are applicable to the parameter settings:

- *Minimum and maximum parameter bounds* - The SOURCE and aquatic decay (STRM, RESV) parameters should have a lower bound of zero to ensure that these parameters are positive. Thus, sources are only allowed to contribute and not remove constaminant mass. A positive aquatic decay parameter is also consistent with the removal/loss of contaminant mass during transport in streams and reservoirs. For the maximum/upper bounds, large positive values that are well above the expected final estimated values are recommended for all estimated parameters.
- *Initial parameter values* - The initial parameter values (*parmInit*) of RSPARROW models should preferably be neutral in their magnitude, either zero or close to zero (note that at least one of the initial values for the model must be non-zero for the *nlfb* function to execute without error). For the delivery factors (DELIVF), an initial value of zero is generally recommended, given that the estimated parameters may be either negative or positive; thus, the selection of a zero value provides a neutral starting value for estimating the final mean coefficient value. The initial values can also be set to fall

within the same order of magnitude or close to the expected coefficient values if these are known in advance. The initial model runs will allow the general range of the expected mean coefficient values to be determined. In subsequent models, users have the option to update the initial parameter values with these prior estimates to improve computational speeds or enhance the convergence of the model during the model development phase. However, models should be evaluated using alternative initial values that differ by at least an order of magnitude to ensure that the model with the minimum error is identified and stable estimates are obtained for the model coefficients (see discussion below on "Model convergence"; also, note the sensitivity of model outcomes to the use of different initial coefficient values for the less statistically significant coefficients).

4.4.4.4 Model convergence and guidelines for model development

The convergence of a nonlinear model to a global minimum model error (i.e., smallest Sum of Squares of Error) is never assured. The model convergence and outcomes (i.e., estimates of the coefficient means, standard errors, t-statistics, and residuals) are potentially influenced by the the presence of multi-collinear or statistically insignificant explanatory variables and the initial values of the parameters.

RSPARROW model performance (e.g., Root Mean Sum of Squares of Error; RMSE) and coefficient metrics (mean estimates, standard errors, t-statistics) closely match those reported by SAS SPARROW, based on comparisons with previously published national (Smith et al., 1997; Alexander et al., 2019a) and selected regional (Preston et al., 2011) SPARROW models. However, RSPARROW estimation with the *nlfb* function, as compared with SAS SPARROW, can be somewhat less robust when estimating the coefficients of explanatory variables with levels of statistical significance that are more marginal ($0.01<p<0.20$) or relatively weak ($p>0.20$) as well as when estimating with variables that exhibit collinearities. In these cases, RSPARROW coefficient metrics may exhibit sensitivities to the initial values of the less statistically significant and collinear variables. Users should be especially attentive to these tendencies during the early stages of model development when many explanatory variables with marginal or weak signficance levels and collinearities are typically evaluated.

The following guidelines are important to consider during model development to ensure that stable estimates are obtained for the model coefficients and to increase the likelihood that the model with the minimum error is identified:

- Users should initially develop and evaluate alternative models using initial coefficient values (*parmInit*) that are more neutral in magnitude (i.e., zero for DELIVF *parmType* and positive but close to zero for SOURCE, STRM, and RESV *parmType*) as described in the previous section and Table 7.
- Users should be attentive to the potential sensitivites of the coefficient metrics (mean, standard error, t-statistics) to the use of different initial coefficient values for explanatory variables with marginal (e.g., $0.01<p<0.10$) or weak (e.g., $p>0.10$) levels of statistical significance or strong collinearities:
  - Models with these characteristics should be evaluated using alternative initial coefficient values that differ by an order of magnitude or more. See the results for the tutorial model 3a in Chapter 6.2.3 and model 7 in Chapter 6.2.7.
  - Users should evaluate models in which the explanatory variables with marginal or weak statistical significance are removed (e.g., see tutorial models 3a-3c in Chapter 6.2.3) or included in different combinations with one another using different initial coefficient values. The evaluations may yield different coefficient metrics, related to their sensitivities to the initial values of the less statistically significant coefficients.
  - The evaluations should include models in which collinear variables with statistically insignificant ($p>0.10$) t-statistics are removed or evaluate models that include new variables that are combinations of the collinear variables.
  - The use of different initial coefficient values for weakly significant (e.g., $p>0.10$) variables can potentially cause unusually large model residuals (e.g., standardized residuals), which can contribute to instances of excessively large values of the log-retransformation bias correction factor (`Mean Exp`

`Weighted Error`; see sub-section 4.4.4.6). In these cases, unusually large standardized residuals are typically associated with highly *leveraged* observations for one or more variables that are statistically insignificant. Highly leveraged observations are associated with values of explanatory variables that exert a disproportionate influence on the model fit. See tutorial model 3a in Chapter 6.2.3.

- To obtain model performance and coefficient metrics in RSPARROW that are as close as possible to those in SAS SPARROW, users should enter the SAS mean coefficient estimates (with four or more significant figures) as the initial coefficient values for the RSPARROW model estimation (e.g., see tutorial model 8 in Chapter 6.2.8).
- Note that the rate of model convergence, expressed by the elapsed time for the optimization to run to completion, may depend on the choice of the initial parameter values and their distance from the final estimates of the parameters.

4.4.4.5 Measures of model performance

Performance measures are reported in the *(run_id)_summary.txt* file (sub-section 5.2.4.2) and saved as CSV files in "summaryCSV" sub-directory. These measures describe the general acceptability of the overall fit of the model to the observations. Users should also conduct more detailed assessments of whether the model assumptions are satisfied as described in sub-sections of Chapter 4.4 (sub-sections 4.4.4.6 to 4.4.4.8 and 4.4.5.1).

The performance metrics include several average measures of model error. A measure of the total model error, the Sum of Squares of Error (`SSE`), is calculated as the sum of the squares of the weighted residuals. The Mean Sum of Squares of Error (`MSE`) adjusts the total model error for the model complexity, and is expressed as the quotient of the `SSE` and the degrees of freedom (i.e., expressed as the difference between the number of observations and number of model parameters). The Root Mean Sum of Squares of Error (`RMSE`, the square root of the `MSE`), gives a measure of the average model error; for values greater than 0.6, the product of the `RMSE` and 100 approximates the percent error of the reach-level prediction associated with one standard deviation of model error. An average measure of the model prediction bias, the `PERCENT BIAS`, is computed as the ratio of the sum of the model residuals (observed load - predicted load) to the sum of the observed load across all calibration sites, and multipled by 100 (see Moriasi et al., 2007). Positive values indicate an average prediction underestimation bias, whereas negative values indicate an average prediction overestimation bias.

Model fit statistics are reported for load and yield based on the R-squared metric. The model R-squared for load observations, `RSQ`, is computed as the ratio of the sum of the model residuals to the total variance in the log transformed observed loads. An R-squared metric (`RSQ-ADJUST`), adjusted for the number of degrees of freedom, is also reported. The R-squared metric for yield observations, `RSQ-YIELD`, adjust the R-squared load metric for the mean log drainage area. `RSQ-YIELD` provides a scale-independent measure of model fit that provides an improved measure of the process interpretability of the model. A limitation of the `RSQ` load metric is that it does not necessarily indicate the explanatory strength of the model because much of the variation in load is associated with drainage area size rather than model processes.

The above performance measures are reported for *conditioned* (monitoring-station adjusted) and *unconditioned* (simulated) predictions (also see discussion of model predictions in sub-section 4.4.7).

- *Conditioned (monitoring-station adjusted) predictions* are calculated by substituting the observed station load for the model predicted load on reaches with monitoring stations. Conditioned predictions provide the most accurate reach predictions for quantifying river loads and estimating model coefficients that quantify the effects of contaminant sources and hydrological and biogeochemical processes on stream quality (Schwarz et al., 2006). Use of the conditioned predictions also reduces the downstream propagation of model errors and their correlation across watersheds (Schwarz et al., 2006). Thus, conditioned predictions are preferred for NLLS model estimation/calibration to obtain the most accurate and physically interpretable model and to evaluate the adequacy of the model fit to observations.

- *Unconditioned predictions* are calculated by executing the model in simulation mode, using mean coefficients from the NLLS model estimated with monitoring-adjusted (conditioned) predictions; thus, the simulated predictions are not adjusted for the observed loads on monitored reaches. Unconditioned predictions preserve mass balance in the calculations of mass transport in stream reaches and mass delivery to downstream waters, including calculations of the contributions of mass from contaminant sources to reaches (i.e., "source shares"). Unconditioned predictions also provide the best representation of the predictive skill of the estimated model at monitored locations (calibration and validation sites). Performance metrics based on the unconditioned predictions are well suited for comparing the prediction accuracy of different models because conditioning effects are removed (i.e., the effects of substituting the observed station loads for the model predicted loads). Model performance metrics based on the unconditioned predictions also give a generally preferred estimate of the expected average accuracy of the model when the model is applied to unmonitored stream reaches.

4.4.4.6 Model assumptions and diagnostic metrics

The assumptions of the nonlinear SPARROW model require that the residuals are identically distributed (homoscedastic), independent across observations, and uncorrelated with the explanatory variables. In evaluating whether an estimated model satisfies these assumptions, users can consult various RSPARROW diagnostics on the model errors, including residual values (observed values minus model predictions), ratios of the observed values to the model predictions, and statistical measures of outliers (as described below). The diagnostics are reported in tabular and graphical output. These include the model summary metrics (*(run_id)_summary.txt*; sub-section 5.2.4.2), the station-by-station listing of diagnostic metrics (*(run_id)_residuals.csv*; sub-section 5.2.4.3), and various diagnostic plots and maps of the station metrics (*(run_id)_diagnostic_plots.html*; sub-section 5.2.4.9). Selected examples of the diagnostics are discussed below.

These diagnostics enable users to evaluate key features of the model accuracy, including the model variance and prediction biases, to improve understanding of whether the model gives an acceptable fit to observed loads and reasonable estimates of the model errors. The diagnostics are reported for conditioned (monitoring-station adjusted) and unconditioned (simulated) predictions in the tabular and graphical output.

Coefficient *sensitivities* are also reported for the model as a measure of the relative importance of each coefficient (and its associated explanatory variable) on model predictions of load. Sensitivity is computed as the absolute value of the percentage change in the predicted load in response to a unit one-percent change in each explanatory variable, with all other explanatory variables (and coefficients) held constant. Plots of the model sensitivities are output to the *(run_id)_diagnostic_sensitivity.html* file (see sub-section 5.2.4.10).

Users can also evaluate regional variations in model performance and coefficient sensitivites by enabling diagnostic control settings for user-specified regional classification variables that define physiographic regions, political jurisdictions, or land use types (see the discussion in sub-section 4.4.5 and the `classvar` and `class_landuse` settings in section 5 of the control script). By default, variations in sensitivities and in the ratio of the observed values to the model predictions are plotted in the *(run_id)_diagnostic_sensitivity.html* and *(run_id)_diagnostic_plots.html* files, respectively, for different watershed sizes, based on total drainage area deciles.

Measures of model outliers

The diagnostic metrics described below and reported by RSPARROW are especially sensitive to unusual outliers. These can inform assessments of the acceptability of the model fit and model errors.

Stations with unusual outliers are automatically identified during model estimation, with selected station attributes output to the model summary metrics file (*(run_id)_summary.txt*; sub-section 5.2.4.2) for observations with the following conditions: a high leverage value greater than an internally computed acceptance criterion, a large standardized residual with an absolute value greater than 3, or a high Cook's D measure of influence (p-value<0.10).

- *Leverage* is used to identify observations, associated with one or more of the explanatory variables, that have a disproportionate influence on the estimated values of the model coefficients. Leverage is computed according to equation 1.62 (Schwarz et al., 2006) and reported by RSPARROW in the

model residuals output file for each station observation *(run_id)_residuals.csv*; see sub-section 5.2.4.3). Leverage values are always positive and sum to the number of model parameters across all observations, with an average value over all observations equal to the number of model parameters (K) divided by the number of observations (N). In general, a value of unusually high leverage exceeds 3K/N (Schwarz et al., 2006). Stations with leverage values (`leverage`) that exceed this high-leverage criterion (`leverageCrit`) are output with selected attributes to the summary metrics file *(run_id)_summary.txt*; sub-section 5.2.4.2). These leverage metrics (`leverage`, `leverageCrit`) are also reported in the station-by-station listing of diagnostic metrics in the *(run_id)_residuals.csv* file.

- *Standardized residuals* provide a normalized measure of the model errors that is useful for identifying outlying observations that are poorly fit by the model. The residuals are standardized in relation to the overall leverage-weighted model variance (equation 1.102, Schwarz et al., 2006). Standardized residuals have unit variance, which allows these measures to be interpreted in absolute terms in relation to a standard normal distribution. For normally distributed residuals, it is expected that on average no more than 3 in 1000 residuals should have an absolute value of the standardized residual that is larger than 3. Standardized residuals are reported in the station-by-station listing of diagnostic metrics (`standardResids` in *(run_id)_residuals.csv*; sub-section 5.2.4.3).
- *Cook's D* (distance) provides a measure of the influence of observations on the model fit, as indicated by the combined effect of leverage and the model error as measured by the prediction residual (prediction error sum of squares). Observations with a statistically large Cook's D metric have an especially strong influence on the estimates of the model coefficients. The metric is computed as the ratio of the square of the leverage-weighted prediction residual to the product of the number of model parameters and the mean square error (Helsel and Hirsch, 2002). Highly influential observations would be expected to exceed the critical test statistic value associated with a small p-value (e.g., <0.10). The p-value is computed from an F distribution with degress of freedom equal to the number of model parameters plus 1 and the the number of observations minus the number of model parameters (Helsel and Hirsch, 2002). The Cooks' D statistic and its p-value are reported in the station-by-station listing of diagnostic metrics (`CooksD` and `CooksDpvalue` in *(run_id)_residuals.csv*; sub-section 5.2.4.3). Observations with statistically significant Cook's D values (e.g., `CooksDpvalue`<0.10) can potentially occur in models with one or more highly insignificant coefficients and can co-occur with unusually high leverage values; the removal of these insignificant explanatory variables from the model may eliminate the unusual influence and high leverage of these observations.

Bias-correction log-retransformation factor

The bias log-retransformation correction factor (or "bias-correction" factor) is estimated for each SPARROW model and used to obtain unbiased estimates of the model predictions of the mean annual load in the original load units (see sub-section 4.4.7). As an additional model performance diagnostic, larger values of the bias-correction factor are also generally indicative of a less precise model.

The bias-correction factor is needed because SPARROW models are estimated using log-transformed values of the mean annual load. Exponentiation of the model predictions of the mean annual log-transformed load results in biased estimates of the mean load in the original units (i.e., typically an underestimation of the mean). Multiplication of the exponentiated loads by the bias-correction factor adjusts for this intrinsic bias. Values of the bias-correction factor are positive and commonly larger than one; for example, values for regional SPARROW total nitrogen models typically range from 1.05 to about 1.4 (Preston et al., 2011), reflecting about a 5% to 40% increase in the mean annual load.

The bias-correction factor is computed as the mean of the exponentiated leverage-weighted log residuals according to equation 1.124 in Schwarz et al. (2006), using the "Smearing Estimator" method (Duan, 1983). For NLLS estimates, the bias-correction factor is unbiased for large samples regardless of the underlying error distribution, but is potentially biased in small samples even with normal residuals (Schwarz et al., 2006).

The bias-correction factor is reported as the `Mean Exp Weighted Error` in the model summary *(run_id)_summary.txt* file (see sub-section 5.2.4.2).

In selected models, the bias-correction factor is potentially sensitive to large outlying values of the model

residuals and leverage; in rare cases, this can cause an excessively large value of the bias-correction factor. In these cases, a warning message is printed to the RStudio Console window as shown below. To assess possible causes, users should check for data errors (load estimates, explanatory data) and re-estimate the model with different initial parameter values after eliminating variables with small and statistically insignificant estimated coefficients.

In cases where the bias-correction factor (`Mean Exp Weighted Error`) exceeds a value of 1000, the following message appears. This may be indicative of unusually high leverage and log residual values.

```
WARNING: THE Mean Exp Weighted Error PRINTED IN THE SUMMARY TEXT FILE
IS EXCESSIVELY LARGE. THIS IS CAUSED BY A LARGE LEVERAGE AND MODEL RESIDUAL
FOR A STATION. CHECK THE DATA FOR THE OUTLYING STATION. ALSO CONSIDER
RE-ESTIMATING THE MODEL USING DIFFERENT INITIAL PARAMETER VALUES OR AFTER
ELIMINATING VARIABLES WITH SMALL AND STATISTICALLY INSIGNIFICANT
ESTIMATED COEFFICIENTS.
```

In cases where the bias-correction factor (`Mean Exp Weighted Error`) is undefined (i.e., infinity), the following message appears. This is caused by a calculated value of one for the leverage for one of the station observations, which is indicative of errors in the data or model estimation.

```
WARNING: THE Mean Exp Weighted Error IS UNDEFINED, CAUSED BY A LEVERAGE VALUE OF ONE.
A PARAMETER MAY HAVE BEEN IMPROPERLY ESTIMATED.
EVALUATE DIFFERENT INITIAL VALUES FOR THE PARAMETERS, INCLUDING INITIAL VALUES
CLOSER TO THE ESTIMATED COEFFICIENT VALUES, OR ELIMINATE VARIABLES WITH SMALL
AND STATISTICALLY INSIGNIFICANT ESTIMATED COEFFICIENTS.
DIAGNOSTIC PLOTS WERE NOT OUTPUT.
```

4.4.4.7 Interpretation of diagnostic plots and maps

RSPARROW provides various diagnostic graphics for evaluating model fit and whether the model assumptions are satisfied. These include two sets of four panel plots (Figure 3 and 6) and maps of model errors (Figure 4). Diagnostic plots and maps can be viewed in the *(run_id)_diagnostic_plots.html* file [sub-section 5.2.4.9; PDF plots include both conditioned (monitoring-station adjusted) and unconditioned (simulated) predictions.

One important model assumption is that the residuals are identically distributed (homoscedastic) across observations. The four panel plots in Figure 3 can be used to identify non-constant systematic patterns in the variance of the residuals–i.e., *heteroscedasticity*. Heteroscedastic residuals can cause inaccurate estimates of the model error (i.e., variance), which may bias the interpretations of coefficient test statistics. The SPARROW model residuals are not required to be normally distributed because of the large-sample properties of the t-statistics, but deviations of the residuals from normality are potentially indicative of heteroscedasticity.

Figure 3a displays the observed log loads vs. the predicted log loads. In this plot, the observations should be evenly distributed about the one-to-one correspondence line with no large outliers. Systematic differences in the spread of the observations–e.g., a larger scatter among observations with either small or large loads–could be indicative of heteroscedasticity in the residuals. The plot of residuals vs. the predicted log loads shown in Figure 3c can also be used to identify whether the scatter of the model errors is uniformly distributed about zero througout the range of of the predicted loads. In these plots (Figure 3a,c), there is a general heteroscedastic pattern visible, with the tendency of larger errors to occur in watersheds with smaller predicted loads. This pattern is often seen for SPARROW models; this may be an expression of the intrinsically larger heterogeneity in accuracy of small catchment loads compared to those for large rivers, where contaminant sources (and model errors) are potentially more integrated and averaged. If such heteroscedasticity can be confidently identified as a structural property of the model, then weighted NLLS can potentially be used to improve the estimates of the model variance and coefficient errors (see discussion in sub-section 4.4.4.11).

Similar observed vs. predicted and residuals plots are also provided for the units of yield, mass per unit area per time (Figure 3b,d). Yield offers the advantage of removing spatial scale effects that are associated with drainage area. Thus, these plots provide scale-independent displays of the model fit that provide an improved perspective on the process interpretability of the model.

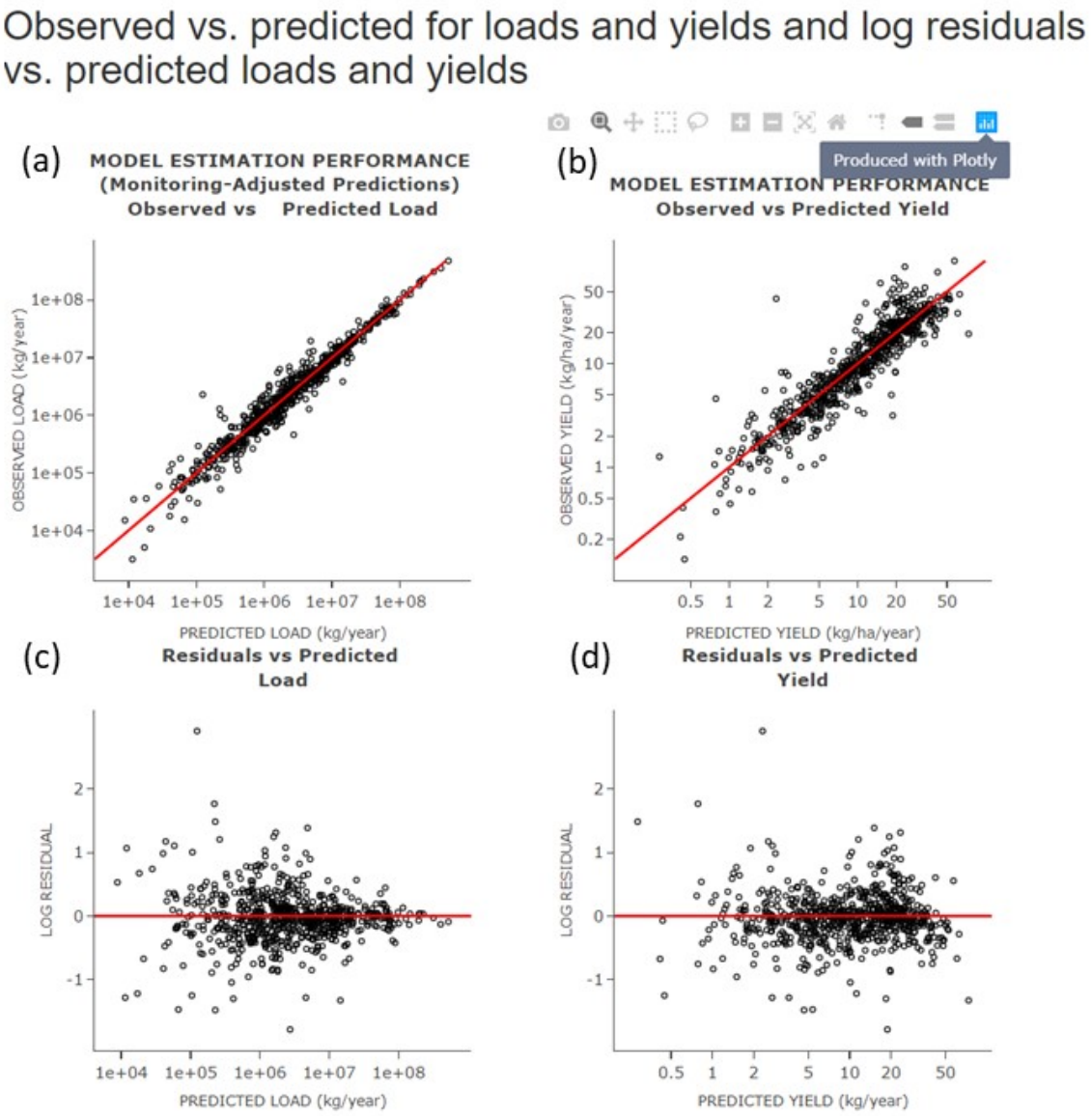


Figure 3: **Example diagnostic plots for the RSPARROW tutorial model 6.** Model performance plots for the monitoring-adjusted (conditioned) predictions are shown for: (a) observed load vs. predicted load; (b) observed yield vs. predicted yield; (c) log residuals vs. predicted load; and (d) log residuals vs. predicted yield.

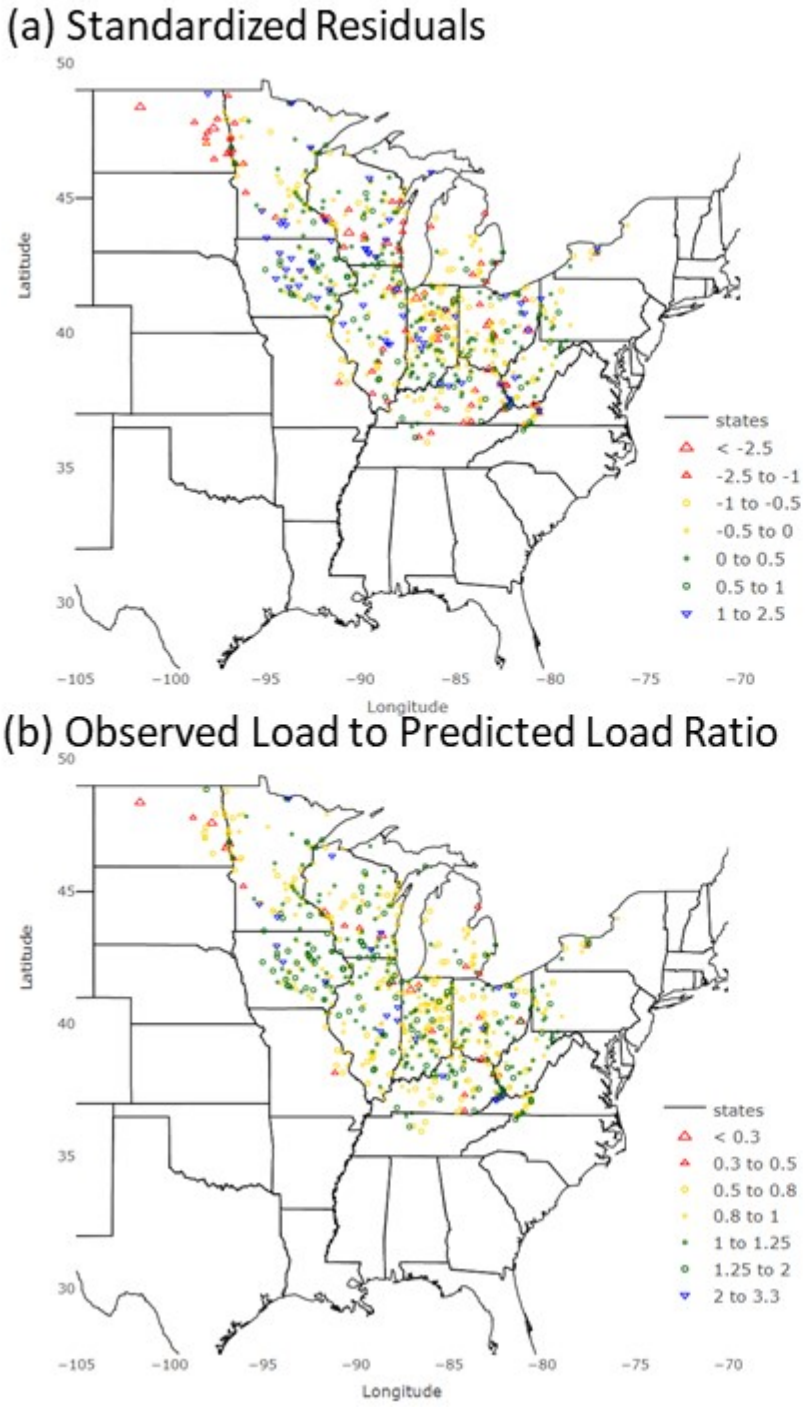


Figure 4: **Example diagnostic maps for RSPARROW tutorial model 6 for the monitoring-adjusted (conditioned) predictions.** (a) standardized residuals; (b) ratios of the observed load to predicted load.

Model prediction biases–i.e., occurrences of over- and under-prediction–can also be identified by using the four panel plots (Fig. 3) and inspecting maps of the standardized residuals (Fig. 4a) and the ratios of the predicted to observed loads (Fig. 4b). Maps are especially useful to identify prediction biases in specific regions and watersheds. Ideally, a SPARROW model should provide generally similar predictive capability over different regions (Fig. 4) and across the range of predicted loads (Fig. 3). As discussed below, some spatial variability in the predictive capability of the model is observed.

Of particular concern are large systematic deviations from the one-to-one correspondence line in the observed vs. predicted plots (Fig. 3a) or the spatial clustering of large model errors with a similar sign across sub-regions or large river basins (Fig. 4). Spatial clustering would be indicative of spatially correlated model errors, which could potentially violate the assumption that the residuals are spatially independent of one another. A statistical test for the presence of spatial autocorrelation in the residuals (Moran's I) can be executed using the control setting `if_spatialAutoCorr <- "yes"` (section 5 of the control script); selected results are illustrated in Fig. 5. The test measures the extent of clustering, dispersion, or random organization in the model error (i.e., over- or under-predictions); the test statistic ranges from +1 to -1, with the level of statistical significance (p-value) reported. A positive test statistic would be associated with a pattern of spatial clustering of the model errors, whereas a negative test statistic would suggest a systematic spatially dispersed pattern of model error. Separate tests are executed for residuals, weighted according to Euclidean and hydrologic distances between the calibration stations (Fig. 5). Example results are shown for tests executed with hydrologic distance weights for hydrologically independent river basins with five or more sites, with the results reported for the most downstream site in each river basin (Fig. 5a,b). Results are also reported for tests with Euclidean distance weights executed for the full spatial domain and for sub-regions (see *(run_id)_diagnostic_spatialautocor.txt*), based on the user-specified contiguous drainage areas (Fig. 5c,d). For these cases, statistically significant ($p<0.10$) results are only observed for two river basins, with both showing negatively correlated residuals.

Distinctive patterns of spatial clusters of large model errors of a similar sign (i.e., positively correlated residuals) may indicate model mis-specification–i.e., evidence of a structurally biased model, which can cause biased coefficient estimates. This may indicate that the model errors are correlated with the explanatory variables (a violation of a model assumption). Likely solutions include identifying one or more important explanatory variables that have been excluded from the model. Adding explanatory variables that scale with basin area may help to correct systematic deviations from the one-to-one line. Region-specific biases may be potentially corrected by the regionalization of one or more of the model coefficients, provided sufficient station observations of load are available to inform the estimation of additional coefficients. Widespread patterns of negatively correlated residuals could potentially occur for watersheds with hydrologically dependent monitoring sites in cases where load observation errors are a dominant component of the model error.

Additional plots shown in Figure 6 can be used to assess model prediction biases, precision, and heteroscedasticity in the model errors. First, the boxplots in Figures 6a,b are useful for evaluating both the overall bias and precision of the predictions (and the plotly tool bar can be used to enlarge details of the plot graphics). The position of the median relative to zero (Figures 6a) or one (Figures 6b) gives a measure of bias, whereas precision is indicated by the symmetry (interquartile range, the edges of the box) of the model errors in relation to zero or the observed to predicted ratios in relation to one.

Second, Figure 6c provides a "Normal Q-Q Plot" of the standardized residuals vs. the standard deviates from a normal distribution. Although the residuals are not required to be normally distributed, deviations of the residuals from normality, visible in the plot as diversions from the one-to-one lines, are potentially indicative of heteroscedasticity. Supporting Q-Q plot statistics are provided in the model output file *(run_id)_summary.txt* (see sub-section 5.2.4.2), including the Normal Probability Plot Correlation Coefficient (`Normal PPCC`). This provides a measure of the linear correlation between the ordered, standardized weighted residuals from the estimated model and the quantiles of the standard normal distribution. A value of the correlation coefficient near 1.0 is evidence that the residuals are from a normal distribution, whereas a value below 0.98 is generally indicative of non-normal residuals (Schwarz et al., 2006). This is accompanied by a reported Shapiro-Wilk's test statistic (`SWilks W`), which provides a statistical test of the normality assumption for the model residuals, with a reported value of statistical significance (`P-Value`). The Q-Q plot is also useful to identify unusually large residuals–i.e., those that diverge appreciably from a standard normal distribution. Unusually large

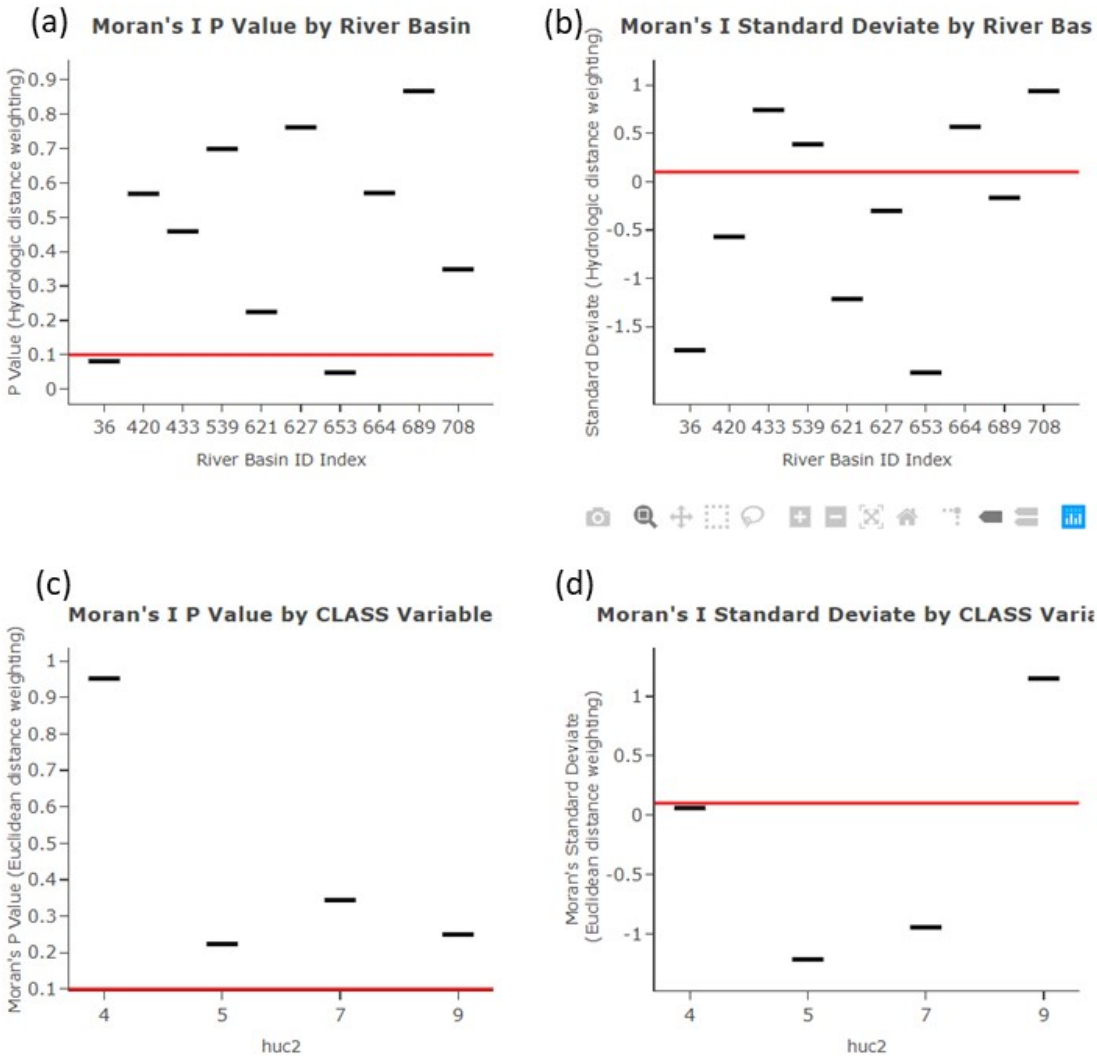


Figure 5: **Results for the Moran's I test for spatial autocorrelation for the RSPARROW tutorial model 6.** (a) P-value for a hydrologic-distance weighted test in river basins; (b) Standard deviate for a hydrologic-distance weighted test in river basins; (c) P-value for a Euclidean-distance weighted test for user-specified spatial classes; and (d) Standard deviate for Euclidean-distance weighted test for user-specified spatial classes. A red line is shown for a p value of 0.10 in plots (a) and (c). The class variable is associated with hydrologic regions in the Midwest USA: 4=Great Lakes, 5=Ohio, 7=Upper Mississippi, and 9=Red-Rainy.

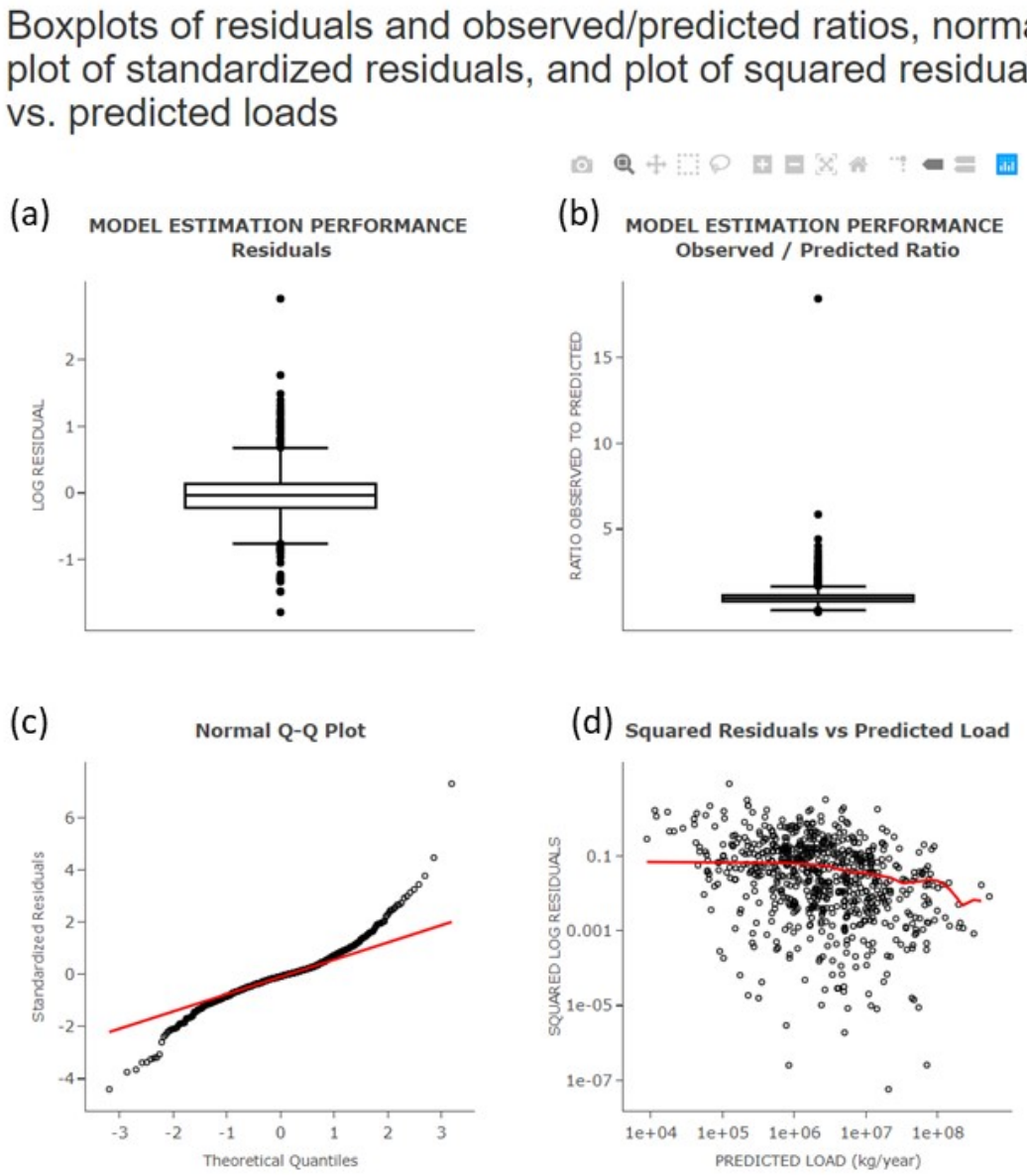


Figure 6: **Example diagnostic plots for the RSPARROW tutorial model 6.** Model performance plots for the monitoring-adjusted (conditioned) predictions are shown for: (a) boxplot of the logged residuals; (b) boxplot of the ratios of the observed to predicted loads; (c) Normal Quantile-Quantile Plot; and (d) plot of the squared log residuals vs. log predicted load.

values of the standardized residual can also be located in the model output file *(run_id)_summary.txt* (see sub-section 5.2.4.2), which displays a listing of station attributes for stations with absolute values of the standardized residuals greater than 3.

Finally, Figure 6d can be used to evaluate whether the model residuals, expressed as the squared logged residuals, are correlated with the model predictions (a violation of one of the model assumptions). A Loess smooth line is fit to the observed residuals. Evidence of a systematic linear or monotonic relation may be indicative of heteroscedasticity that may be potentially addresssed through the development of a weighted NLLS (see the discussion below of the control setting `NLLS_weights`).

Addtional diagnostics are reported for the model coefficient sensitivities of estimated models in the *(run_id)_diagnostic_sensitivity.html* file (see sub-section 5.2.4.10). Selected results are illustrated in Figures 7 and 8. Sensitivity measures the relative importance of each explanatory variable on model predictions of load, reported as the absolute value of the percentage change in the predicted values in response to a unit one-percent change in each explanatory variable, with all other explanatory variables held constant. The parameters are ordered from smallest to largest and plotted using arithmetic and log scales (Figure 7). Plots of the spatial variability in the sensitivity of the coefficients for user-specified regional classes are also reported (Figure 8) in cases where users have defined a regional classification variable (see the discussion in sub-section 4.4.5 and the `classvar` control setting).

Sensitivity measures are generally positively correlated with the coefficient t-statistics but provide a measure of the relative importance of the coefficients on stream load that integrate statistical significance with the magnitude of the predicted load response. An advantage of the sensitivity measures, as compared to measures of the statistical variance explained by the coefficients, is that they capture nonlinearities in the load response to individual explanatory variables. In SPARROW models, load predictions will typically show the highest sensitivities to the source variables, with a smaller load response observed to the land-to-water delivery factors and aquatic decay variables.

4.4.4.8 Multi-collinearity effects

Model convergence to a global minimum and the estimation of accurate standard errors for the model coefficients are dependent on explanatory variables that are statistically independent of one another. Multi-collinearity is indicated by the presence of high levels of correlation between two or more explanatory variables that cause all of the correlated variables to have statistically insignificant coefficients. In such cases, the coefficients associated with collinear variables will be imprecisely estimated, owing to an inflated estimate of the variance of the coefficient; this will also produce inaccurate t-statistics and measures of statistical significance (p-values).

The Variance Inflation Factor (VIF) metrics, which are reported in the RSPARROW output (see sub-section 5.2.4.2), measure the importance of multi-collinearity in the explanatory variables. The square root of the VIF represents the proportion by which the t-statistic could be increased if multicollinearity were eliminated (Schwarz et al., 2006). Note that confirmation of multicollinearity additionally requires evidence of statistically insignificant (e.g., p value > 0.10) parameters with large VIF values. Confirmation of multicollinearity should also be accompanied by a value of the eigenvalue spread (computed from the eigenvalues of the X'X matrix of normalized gradients) that are generally greater than 100 (Schwarz et al 2006; see sub-section 5.2.4.2).

To assist with identifying collinear variables, users can also inspect the bivariate correlations for the explanatory variables in the *(run_id)_explvars_correlations.txt* file in the *estimate* sub-directory (and also the bivariate plots in the *(run_id)_explvars_correlations.pdf* file). These files are created using the setting `if_corrExplanVars<-"yes"` (section 5 of the control script).

One solution for multicollinearity is to remove one of the correlated variables from the model or to develop a new explanatory variable that results from the combination of the two collinear variables.

In cases of severe multi-collinearity that results in a singular (ill-conditioned) Jacobian (or Hessian) matrix, RSPARROW terminates the model execution and prints the warning message as shown below to the RStudio Console or in the batch log file.

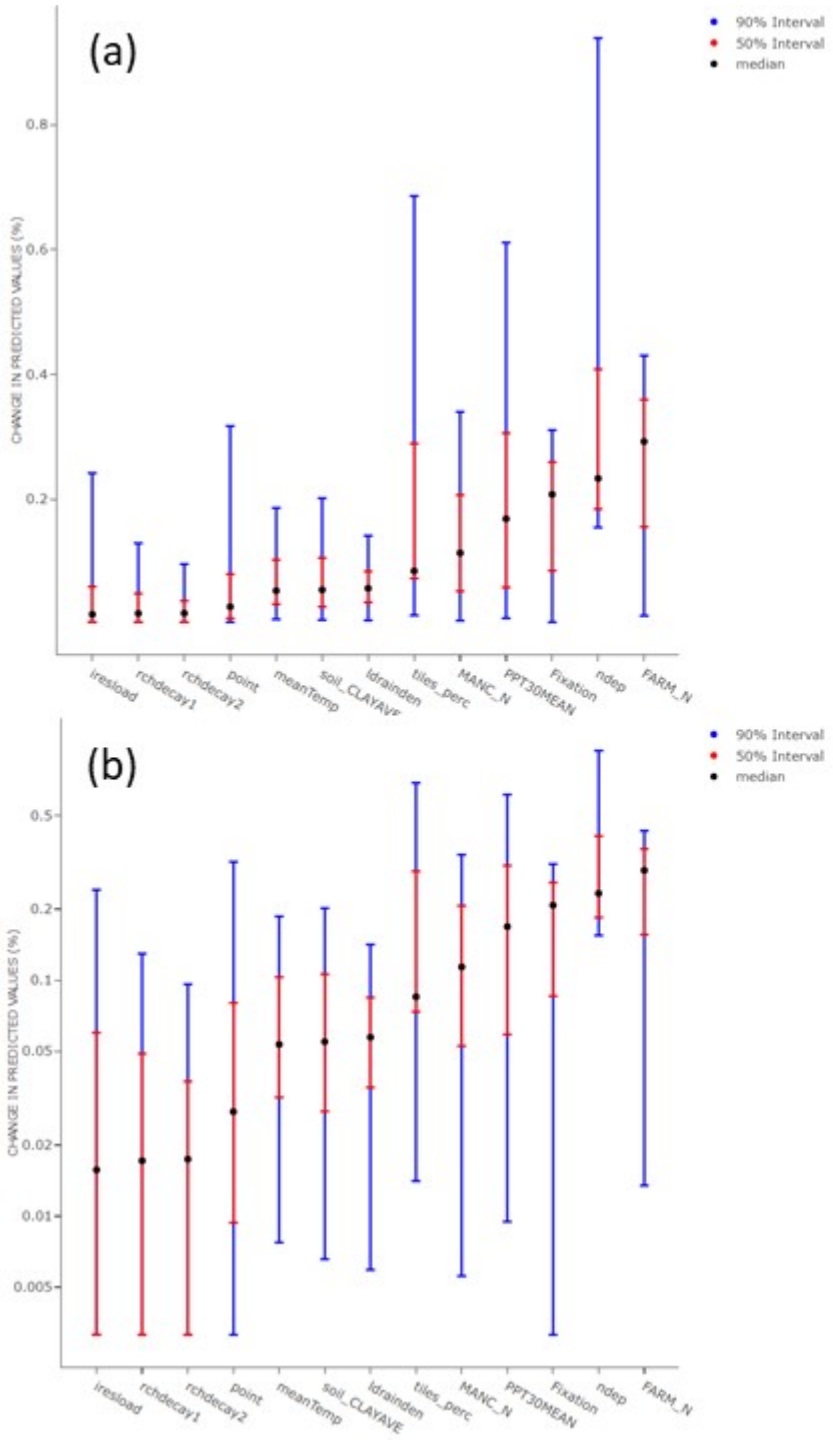


Figure 7: **Example sensitivity plots for model parameters for the RSPARROW tutorial model 6.** The sensitivity is computed as the absolute value of the percentage change in the predicted values in response to a unit one-percent change in each explanatory variable, with all other explanatory variables held constant. The percentage change is plotted against the name of the explanatory variables, ordered from the smallest to the largest change percentage. Sensitivities are reported for (a) percentages shown on a linear scale; and (b) percentages shown for a log scale. The median sensitivity is plotted as a dot; the interquartile range (50 percent interval) is shown by the inner red bars and the 90 percent interval is shown by the blue outer bars.

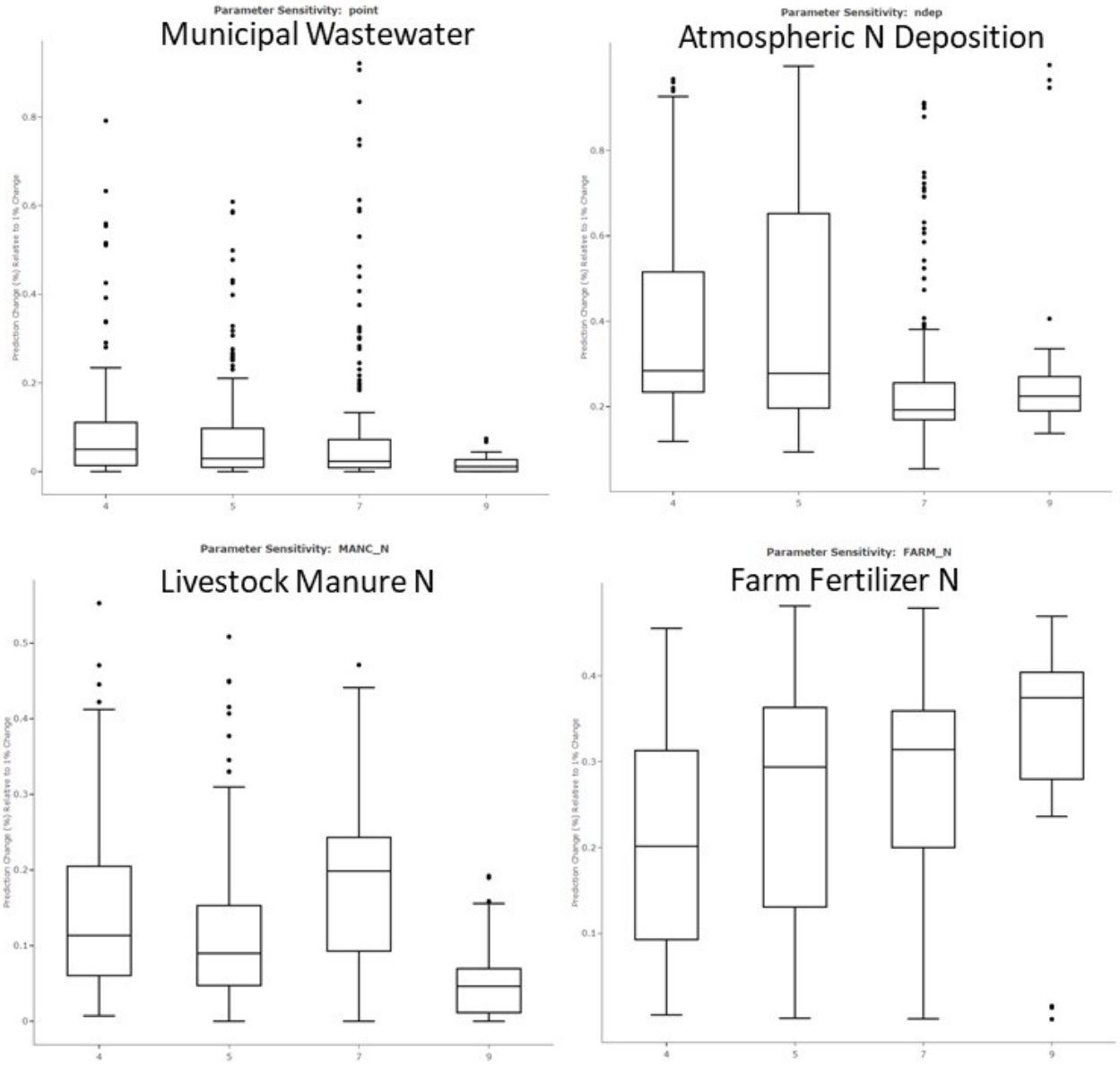


Figure 8: **Example sensitivity plots for model parameters in sub-regions of the modeled domain for the RSPARROW tutorial model 6.** The sensitivity is computed as the absolute value of the percentage change in the predicted values in response to a unit one-percent change in each explanatory variable, with all other explanatory variables held constant. The distribution of percentage change is shown for four model parameters across four hydrologic unit sub-regions (huc2) of the Midwest (4=Great Lakes, 5=Ohio, 7=Upper Mississippi, and 9=Red Rainy).

The Jacobian matrix contains the first-order partial derivatives of the vector functions of the explanatory variables (the Hessian matrix contains the second derivatives). The Jacobian (or Hessian) matrix will be singular (with a determinant equal zero) if at least one of the explanatory variables can be expressed as a linear combination of one or more of the other explanatory variables. In such a case, the *nlfb* optimizer may indicate that the model has "converged" (`No parameter change"` and `converged = TRUE` messages) and will print coefficient estimates in the Console window (the Console text is also output to the execution log file, *(run_id)_log.txt*). However, RSPARROW will detect a singularity condition during the subsequent evaluation of the Jacobian (or Hessian) matrix when the standard errors of the model coefficients are estimated.

```
SINGULAR (ILL-CONDITIONED) JACOBIAN (OR HESSIAN) MATRIX FOUND.
  MODEL ESTIMATION SUMMARY METRICS ARE NOT OUTPUT.
  RUN EXECUTION TERMINATED.

  A SINGULARITY CONDITION INDICATES THAT AT LEAST ONE OF THE EXPLANATORY VARIABLES IS A
  LINEAR COMBINATION OF ONE OR MORE OF THE OTHER EXPLANATORY VARIABLES.

  THESE CONDITIONS CAN BE CAUSED BY:
  (1) A SMALL ESTIMATED COEFFICIENT OR INITIAL PARAMETER VALUE;
  (2) PROBLEMS WITH THE EXPLANATORY DATA, INCLUDING A CONSTANT EXPLANATORY VARIABLE
      OR TWO OR MORE EXPLANATORY VARIABLES THAT SUM TO A CONSTANT OR ARE IDENTICAL

  USERS SHOULD CHECK THE EXPLANATORY VARIABLES FOR CONSTANT OR IDENTICAL VALUES.
  RE-ESTIMATE THE MODEL AFTER ADJUSTING SMALL INITIAL PARAMETER VALUES OR
  AFTER ELIMINATING VARIABLES WITH SMALL NEAR-ZERO COEFFICIENTS.
```

A singularity warning can be caused by:

- *A small estimated coefficient or initial parameter value* - A small, near-zero estimated coefficient can potentially contribute to a singular cross product of the Jacobian matrix (implying that at least one row of the matrix is a scalar multiple of another; e.g., a row of zero values). To confirm that a coefficient is responsible for the singularity warning, users can re-estimate the model after eliminating the predictor variable associated with a near-zero coefficient. A near-zero estimated coefficient may be influenced by an initial parameter value that is very close to zero (e.g., <0.001) and less sensitive to parameter adjustments during optimization. These effects are potentially accentuated by a predictor variable that has small variability or small numerical values; the former may require eliminating the variable from the model, whereas the latter may necessitate rescaling the data by a constant to enhance the sensitivity of the optimization procedure. Users may also attempt to re-estimate the model using a larger initial value for the coefficient.
- *Errors in the explanatory data* - Two or more explanatory variables that sum to a constant or are identical to one another can cause a singularity condition. A singularity can also be caused by a constant explanatory variable, including cases where virtually all reaches are constant (e.g., near zero values). Users should check the explanatory data for these instances. Land-use explanatory variables are also potentially subject to cases where the variables sum to a constant; one or more of the minor land-use classes may need to be eliminated from the model to avoid this situation.

4.4.4.9 Model execution in simulation mode

```
#------------------------------------------------------------------------------
  #Specify if simulation is to be performed using the initial parameter values
  # "yes" indicates that all estimation, prediction, maps, and scenario files
  #   from the subdirectory with the name = run_id will be deleted
  #   and only regenerated if settings are turned on.
  # A "yes" setting will over-ride a selection of if_estimate<-"yes".
if_estimate_simulation<-"yes"
```

This setting executes the model in simulation mode, without formal NLLS parameter estimation, using

the user-defined initial values (*parmInit*) in the *parameters.csv* file as fixed parameter values during model execution. This setting can be used to test the model calculations and predictions (e.g., summation of mass loads in the river network) prior to estimating the model (also specifiy the control setting `if_predict<-"yes"`). The setting can also be used, for example, to execute a simple land-use model with literature export coefficient values, which may be informative to establish baseline information about constituent loads in streams. Note that simulation performance diagnostics are output (see sections 5.2.4.2 and 5.2.4.9) for monitoring sites with positive load values (*depvar*>0).

Note that this setting will over-ride selections of `if_estimate<-"yes"` and `if_validate<-"yes"`, and all files and directories in the estimate, prediction, mapping, and scenario folders for a given *"run_id"* will be deleted.

User settings for the lower (*parmMin*) and upper (*parmMax*) bounds in the *parameters.csv* file control the explanatory variables that are included in the simulated model execution. To include an explanatory variable in the model execution, *parmMin* should be set to a numerical value that is less than *parmMax*. See Table 5 in Chapter 3.3 for details on these settings.

4.4.4.10 Estimation of the standard error of the coefficients

```
#-------------------------------------------------------------------------------
  # Specify if more accurate Hessian coefficient SEs are to be estimated
  #  Note: set to "no" for Jacobian estimates and to reduce run times
  ifHess <- "yes"
```

This computes the second derivative Hessian standard errors for the estimated model coefficients. These estimates are more accurate than the conventional Jacobian estimates, obtained with the setting `ifHess<-"no"` (the *nlfb* function computes the Jacobian estimates by default). Note that the Hessian estimates are more computationally demanding, typically requiring as much as 50% or more of the time required for the Jacobian estimates. During the early stages of developing more computationally demanding models, computing only the Jacobian standard error estimates offers some time savings. However, Hessian standard errors (and the associated t statistics and p values) should be used to make final decisions about variable selection because these provide the most accurate statistical information. Hessian estimates are also more consistent with the SAS SPARROW estimates of standard errors, which are derived using Hessian methods.

In cases of severe multi-collinearity that results in a singular (ill-conditioned) Hessian matrix, RSPARROW terminates the model execution. The Hessian matrix will be singular (with a determinant equal zero) if at least one of the explanatory variables can be expressed as a linear combination of one or more of the other explanatory variables. See sub-section 4.4.4.8 for details.

4.4.4.11 Weighted nonlinear least squares

```
#-------------------------------------------------------------------------------
  # Select regression weights:
  #  "default": weights = 1.0
  #  "lnload":  weights expressed as reciprocal of variance proportional to
  #               log of predicted load
  #  "user":    weights assigned by user in userModifyData.R script, expressed as the
  #               reciprocal of the variance proportional to user-selected variables
  NLLS_weights <- "default"
```

The "default" setting executes a conventional unweighted nonlinear least squares (NLLS) estimation. This method assigns equal weights with a value of "1" to the observations.

The two other settings allow users to execute a weighted NLLS estimation of an RSPARROW model, according to the general methods described in the SPARROW documentation (Schwarz et al., 2006; see section 1.5.1.2).

4.4.4.11.1 Background

The weighted NLLS methods are generally meant to address cases where the SPARROW model residuals are not identically distributed (non-uniform)–i.e., heteroscedastic, where the variance of the model errors is systematically related in some manner to the predictor variables. In SPARROW, the residuals are not required to be normally distributed, but certain deviations of the residuals from normality may be associated with heteroscedastic patterns. Although the coefficient estimates are unbiased in the presence of heteroscedasticity, the test statistics are potentially less accurate for models with heteroscedastic residuals because the true variance may be underestimated.

In weighted NLLS methods, the optimal values of the weights are proportional to the reciprocal of the variance of the errors of the observations (Schwarz et al., 2006). In a weighted NLLS SPARROW estimation (and in RSPARROW), the residuals are multiplied by the square root of the weights (Schwarz et al., 2006; equation 1.51). The objective of weighted NLLS is to ensure that the variance of the model errors is approximately the same for all observations. Thus, observations with a larger estimated error variance receive less relative weight in the NLLS estimation. Unless the weights of the observations are known, the general approach is to estimate the weights using a functional relation between the unweighted NLLS model residuals and one or more explanatory variables (e.g., area, predicted load, etc.).

4.4.4.11.2 General procedures

The use of these weighting methods should be guided by sufficient evidence of heteroscedasticity in diagnostic residual plots for unweighted NLLS SPARROW models. Users should also conduct evaluations of functional variables that might be used to model the heteroscedasticity (i.e., variables that correlate with the squared values of the log residuals from the unweighted SPARROW model). To assist with this, users can access the model residuals of an unweighted SPARROW model in the *(run_id)_residuals.csv* file (located in the "(run_id)/estimate" sub-directory) or from the *Mdiagnostics.list* object in RStudio.

Following execution of a SPARROW weighted NLLS, users can access the estimated weights in the *(run_id)_residuals.csv* file or in the R object *Csites.weights.list* in RStudio. A message is also written to the *(run_id)_summary.txt* file after the `PARAMETER SUMMARY` information indicating that a weighted NLLS has been executed according to the method specified by the user in the `NLLS_weights` setting.

For the two weighting options described below, the uniquely named system variable *"weight"* must be created, with the estimates of the residual weights assigned to the variable. This requires adding the system variable *"weight"* in the column *sparrowNames* in the *dataDictionary.csv* as a FIXED *varType* (see Table 4 in subsection 3.2). This ensures that the variable is added to the *subdata* and *sitedata* R objects, which are needed to support the modeling functions. Additional instructions are given below for each of the options to ensure that the values for the *"weight"* variable are properly defined.

Both the unweighted and weighted NLLS models must employ the identical model specification and river network node structure and reach order. The identical control settings should be used for the `calculate_reach_attribute_list` setting (section 2 of the control script). Note that the use of the RSPARROW calculated *hydseq* variable can potentially produce a different order and sequence number for the calibration sites than that based on a user-supplied *hydseq* variable in the *data1.csv* file. Weighted NLLS models with different calibration site sequencing numbers than those of the unweighted model will produce inaccurate results.

4.4.4.11.3 The `"lnload"` option

For this option, the weights are unknown but are estimated as the reciprocal of variance of the model errors as a function of the log transformed predicted load. The weights are estimated and implemented in a stepwise procedure (see Schwarz et al., 2006, p.63-64), with the residuals from a SPARROW unweighted NLLS, executed in the first step, used to estimate the weights and execute a SPARROW weighted NLLS in a second step. In estimating the SPARROW weighted NLLS model, the residuals are multiplied by the square root of the weights (Schwarz et al., 2006; equation 1.51).

The weighting procedure can potentially be used to account for heteroscedasticity in the error variance that is positively or negatively correlated with the predicted load. For SPARROW, the error variance is often observed to be negatively correlated with the predicted load (i.e, smaller variance for larger loads), although the severity of the heteroscedasticity varies considerably by model. In such cases, the nonlinear regression relation between the squared residuals and the log predicted load (used to estimate the weights) will typically have a negative exponent coefficient; thus, greater weight would be given to the smaller error variance associated with higher loads that typically occur in larger rivers and watersheds.

Procedures for the "lnload" option. Execution requires users to implement the following steps:

- Add the system variable *weight* in the column *sparrowNames* in the *dataDictionary.csv* as a FIXED *varType*. A "NA" should also be entered into the *data1UserNames* column in the *dataDictionary.csv* file associated with the variable *weight* in the column *sparrowNames*.
- Execute an initial SPARROW unweighted NLLS (using the `NLLS_weights<-"default"` setting), which saves the model residuals to a CSV file.
- Using a new `run_id`, execute a SPARROW weighted NLLS (using the `NLLS_weights<-"lnload"` setting), with the R statements (shown below) added to the *userModifyData.R* script, including the name of the unweighted model (*pre_run_id*).

```
# Required R statements in the "userModifyData.R" function to execute the
#  NLLS_weights <- "lnload" option.

 # Enter the run_id of the initial SPARROW unweighted SPARROW model:
 pre_run_id <- "the run_id for the previous unweighted SPARROW model"

 # Estimate the weights for the SPARROW model
 nreaches <- length(fnode)
 weight <- estimateWeightedErrors(file.output.list,run_id,pre_run_id,
                                  nreaches,calsites)
```

During the RStudio execution of the SPARROW weighted NLLS model, the RSPARROW procedure first reads the residuals from the prior unweighted SPARROW model (*pre_(run_id)_residuals.csv*) and executes the RSPARROW function *estimateWeightedErrors.R*; this performs a nonlinear regression (power function using the *nls* R routine) between the unweighted model residuals (squared of the log residuals) and the log of the predicted load. The weights, returned by the function, are computed as the reciprocal of the squared residuals (variance) predicted by the power function, normalized by the mean of the weights (with normalization according to equation 1.50, Schwarz et al., 2006; this maintains the average variance of the error in the weighting procedure). The SPARROW weighted NLLS then applies the square root of the reciprocal weight to the model residuals during model estimation (equation 1.51, Schwarz et al., 2006). If the system variable *weight* is not present in the *dataDictionary.csv* file or has missing ("NA") values, RSPARROW will terminate model execution and print a warning message.

After execution of the SPARROW weighted NLLS (before exiting the RStudio session), users can issue the following command in the Console window to view the nonlinear regression results for the residual weight relation: *summary(Csites.weights.lnload.list$regnls)*. A plot of the functional relation (squared residuals vs. log predicted load) is also output to the file *(run_id)_weights.pdf* to allow users to evaluate the acceptability of the weights. The values of the weights are written to the *(run_id)_residuals.csv* file and are available during the RStudio session in the R object *Csites.weights.lnload.list*.

4.4.4.11.4 The "user" option

For this option, the weights are defined by the user and expressed as the reciprocal of the variance of the model errors proportional to user-defined functions and variables. It is recommended that users normalize the weights by the mean of the weights as shown below (with normalization according to equation 1.50, Schwarz et al., 2006) to maintain the average variance of the error in the weighting procedure. In estimating the

SPARROW weighted NLLS model, the residuals are multiplied by the square root of the weights (Schwarz et al., 2006; equation 1.51).

One optional weighting method is to apply the general approach illustrated by the `"lnload"` method described above but to account for additional or alternative predictor variables that are suggested by user evaluations of heteroscedastic patterns in the unweighted residuals. Users can develop the weights using programming languages other than R (and use R statements to read a file with the weights into the *userModifyData.R* script), or alternatively, more advanced R users can edit the *estimateWeightedErrors.R* function to read additional user-supplied predictor variables from the *pre_(run_id)_residuals.csv* and execute a multi-variate power function. This new user-modified function should be called from the *userModifyData.R* script using similar R statements as shown above for the `"lnload"` option; the new function can be located in the *userModifyData.R* script or sourced from an external location.

Another optional weighting method is one that accounts for uncertainties in the estimates of the long-term mean annual load at the monitoring sites. The weights are intended to account for variations in the residuals that are related to uncertainties in the estimates of the mean annual loads that are associated with the measurement of water quality and flow and the statistical modeling techniques used to estimate station loads (Lee et al., 2016). A previous application of this weighting method used the following two-step procedure (see supporting information of Alexander et al., 2008): (1) estimate the functional relation (regression) between the squared residuals from an unweighted SPARROW model and the station mean-adjusted mean square error of the mean annual load estimate (i.e., computed as the ratio of the variance of the monitoring station mean annual load estimate to the square of the mean annual load estimate); (2) use the reciprocals of the predicted squared residuals from the regression as the weights in a re-estimated weighted SPARROW model.

Methods have also been applied that assume the residual weights are proportional to the incremental drainage size between nested calibration monitoring sites to account for heteroscedasticity in the variance of the SPARROW model errors associated with the influence of conditioned predictions on model errors. In one SPARROW model (Anning and Flynn, 2014), the weights were calculated as a function of the log-transformed intervening area between stations, adjusted for the average of the log-transformed intervening area for all stations; the weighting expression was formulated to give greater weight to stations with larger intervening drainage areas. More recently, weights have been calculated in regional SPARROW models as a function of the percentage of the total drainage above monitoring sites that is associated with the incremental area between monitoring sites (G. Schwarz, USGS, written communication, 2019). This function is designed to reduce heteroscedasticity in model errors associated with the size of the nested areas of monitoring sites, which has been observed to be related to the underestimation of model errors at downstream nested sites. To support the application of incremental area-based weighting functions, the variable *tiarea* (sum of the incremental drainage area of reaches located between monitoring sites) is computed by the *setNLLSWeights.R* function and output to the *(run_id)_residuals.csv* file and the *Csites.weights.list* R object.

Procedures for the `"user"` option. The weights are estimated and implemented in a stepwise procedure. In the first step, an initial SPARROW unweighted NLLS model is executed (the residuals are automatically saved into the *(run_id)_residuals.csv* file in this step). In the second step, the user should evaluate the model residuals from step 1 for heteroscedasticity and develop estimates of the residual weights as a function of one or more appropriate predictor variables (e.g., predicted load, incremental nested area, mean load measurement errors). In a final step, a SPARROW weighted NLLS model is estimated based on the user-developed weights, using the same specification and calibration sites as that of the model executed in the first step.

The user's weights should be input to the SPARROW weighted NLLS model according to the following steps:

- Add the required *sparrowNames* system variable *weight* to the *dataDictionary.csv* file as a FIXED *varType*. The name of this variable is reserved as a unique name (see Table 4 in section 3.2 of the documentation), with the variable length equal to the number of reaches. If the system variable *weight* is not present in the *dataDictionary.csv* file or has missing ("NA") values, RSPARROW will terminate model execution and print a warning message.
- Transfer the user-defined weights to the system variable *weight* using one of two methods:
  - Option 1. If the user-defined weights are included as a variable in the *DATA1.csv* file, enter the

name of the user-defined weights variable to the *data1UserNames* column in the *dataDictionary.csv* file, associated with the *sparrowNames* system variable *weight* (see option 1 example below).

– Option 2. If the user-defined weights are not present in the *DATA1.csv* file, then calculate the weights in the *userModifyData.R* script of the SPARROW weighted NLLS model via user-supplied R statements. A "NA" should be entered into the *data1UserNames* column in the *dataDictionary.csv* file, associated with the *sparrowNames* system variable *weight* (see option 2 example below). Example R coding statements are given below to illustrate the steps that can be used in the *userModifyData* script to define the values of the *weight* variable.

```
# dataDictionary.csv (optional entries for the "weight" system variable):
#  Option 1:
#        varType   sparrowNames      data1UserNames        varunits    explanation
#         FIXED       weight      "user-defined name"

#  Option 2:
#        varType   sparrowNames      data1UserNames        varunits    explanation
#         FIXED       weight             NA
```

Example R statements in the *userModifyData.R* script applicable to option 2 of the `NLLS_weights <- "user"` option.

```
# userModifyData.R function

# Option 2:  Example R statements to obtain final "weight" values for executing a
#            SPARROW weighted NLLS estimation.

# Read the calibration site weights from a user's external CSV file
#   'Indata' object contents, with length equal to the number of calibration sites:
#      station_id - unique alphanumeric station ID (type=character)
#      rvarWeight - weights, expressed as the reciprocal variance (type=numeric)
Indata <- read.csv(file="CSV path and filename", header=TRUE, sep=",")

# Normalize the weights relative to the mean (equation 1.5, Schwarz et al. 2006)
Indata$rvarWeight <- Indata$rvarWeight * mean(1/Indata$rvarWeight)

# Transfer the weights for the calibration site model errors (length equal to the
#   number of calibration sites) to the system variable "weight" (length equal to
#   the number of reaches):
  # create a data frame with a common ID, the "station_id"
  sdata <- data.frame(station_id,demiarea,hydseq)   # length = number of reaches
  # merge with the weights in the Indata object
  sdata <- merge(sdata,Indata,by="station_id",all.y=FALSE,all.x=TRUE)
  # resort by 'hydseq' order to ensure consistency with the system variables
  #   in the "userModifyData.R" function
  sdata <- sdata[with(sdata,order(sdata$hydseq)), ]
  # transfer weights to the system "weight" variable
  weight <- sdata$rvarWeight
```

4.4.5 Model spatial diagnostics (section 5 of control script)

4.4.5.1 Spatial autocorrelation tests

```
#-------------------------------------------------------------------------------
  # Specifiy if the spatial autocorrelation diagnostic graphics for Moran's I
  #    test are to be output
  if_spatialAutoCorr <- "yes"
```

This setting computes a Moran's I statistical test for spatial autocorrelation in the model residuals. This allows users to test for the presence of a spatial correlation in the residuals, which would invalidate the model assumption that the model errors are independently and identically distributed over the spatial domain.

The Moran's I test provides a statistical measure of extent of clustering, dispersion, or random organization (see Fig. 9) associated with the SPARROW model residuals (predicted minus observed load values) at the monitoring calibration stations. A positive value for the Moran's standard deviate test statistic I indicates a tendency for *clustering* of the model residuals, whereas a negative value indicates a tendency for the residuals to be *dispersed*. In RSPARROW, the distance weights are not row standardized, and thus, the values of the standard deviate test statistic can fall outside of the range -1 to +1.

The null hypothesis for the Moran's I test is that there is no spatial pattern of clustering or dispersion in the residuals (i.e., no spatial correlation). A statistically significant *positive* Moran's I test statistic would indicate that the spatial pattern of residual values (over- or under-predictions) in specific regions and watersheds are more spatially clustered than expected for a random process. Such a pattern of spatial clustering of the model errors may indicate model mis-specification. A statistically significant *negative* Moran's I test statistic would indicate that the spatial pattern of residual values are more spatially dispersed than expected for a random process. Such a pattern could potentially occur within watersheds with hydrologically connected monitoring sites, for example, if load observation errors were a dominant component of the model error.

RSPARROW executes the Moran's I test using the *moran.test* function in the *spdep* R library. The number of observations in the test is adjusted for no-neighbor observations (`adjust.n=TRUE`) in cases where zero values (`zero.policy=TRUE`) exist for the distance weights. The p-values are reported for a two-sided null hypothesis test.

Test results are reported in the *(run_id)_diagnostic_spatialautocor.html* and *(run_id)_diagnostic_spatialautocor.txt* files in the "(run_id)/estimate" sub-directory. Separate test results are reported using distance weights specified according to Euclidean and hydrologic distances between the calibration stations (Fig. 10). The results are reported for the following three spatial domains:

- Hydrologically independent river basins with five or more sites, with the results reported for the most downstream site in each river basin; separate tests are reported for both Euclidean and hydrologic (Fig. 10a,b) distance weights.
- The full modeled domain, using hydrologic distance weights.
- The full modeled domain and regions, based on the user-specified contiguous drainage areas (Fig. 10c,d) as defined by the the first classification variable listed in the `classvar` setting (described below in subsection 4.4.5.2), using Euclidean distance weights.

```
#-------------------------------------------------------------------------------
  # Specify the R statement for the Moran's I distance weighting function:
  MoranDistanceWeightFunc <- "1/sqrt(distance)"    # inverse square root distance
  MoranDistanceWeightFunc <- "1/(distance)^2"    # inverse squared distance
  MoranDistanceWeightFunc <- "1/distance"        # inverse distance
```

The `MoranDistanceWeightFunc` setting allows users to specify the functional form of the distance weights. Weights are typically expressed in the Moran's I test as an inverse of the distance, distance squared, or square root of the distance between site locations. The variable name `distance` is required in the user's R functional expression for the `MoranDistanceWeightFunc` setting. Among these weighting functions, the inverse squared

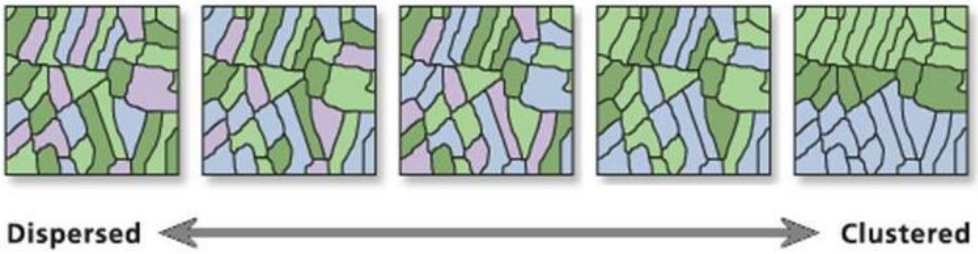


Figure 9: **Illustration of a continuum of spatially correlated patterns ranging from highly dispersed (negatively correlated) to highly clustered (positively correlated).** (from ESRI, Spatial Autocorrelation summary; http://pro.arcgis.com/en/pro-app/tool-reference/spatial-statistics/spatial-autocorrelation.htm)

distance function gives the least amount of weight to the residuals at distant sites, whereas the inverse square root distance function gives comparatively higher weights to the residuals at distant sites.

Other functional expressions may be used; however, users should verify that the functions produce positive values within an acceptable range for the distance weight variable. The presence of negative values of the distance, for example if an inverse log function is used, will produce the following error message.

```
AN ERROR OCCURRED IN PROCESSING diagnosticSpatialAutoCorr.R
Error in mat2listw(xmoran.dists.inv) : values in x cannot be negative
RUN EXECUTION TERMINATED.
```

4.4.5.2 Discrete spatial classification variables

```
#-------------------------------------------------------------------------------
  # Specify spatially contiguous discrete classification variables (e.g., HUC-x)
  #  for producing site diagnostics
  # Diagnostics include: (1) observed-predicted plots
  #                      (2) spatial autocorrelation (only the first variable is used)
  #                      (3) sensitivities (only the first variable is used)
  classvar<-NA  # for NA, total drainage area is used as the default classification
  classvar <- c("huc2","huc4")
```

This optional setting defines spatially contiguous discrete classifications for the monitoring stations. This allows the model diagnostic plots (e.g., observed vs. predicted loads, model residuals vs. prediction loads, and ratios of the observed to predicted loads) to be displayed and evaluated by users for different regions of the modeled spatial domain. These can be used to assess regional spatial variations in the bias and precision of the model predictions.

The `classvar` variable can include one or more discrete classification variables that define spatially contiguous drainage areas. Note that the required variable type is numeric. The variables may include hydrologically connected regions, such as the 18 "huc2" hydrologic regions for the conterminous USA or non-hydrologically connected areas having geographic importance to water-quality processes (e.g., ecological zones, physiographic regions) or relevance to governmental management and reporting (e.g., states, counties). The length of `classvar` should be equivalent to the total number of reaches.

Only the first entry for the `classvar` is used to report results for contiguous drainage areas for selected diagnostic plots and analyses (e.g., spatial autocorrelation analysis, model parameter sensitivities, 4-panel plots for loads and yields). If the control setting is missing (`classvar<-NA`), then the total drainage area is used as the default classification variable, with the deciles of drainage area used to define the discrete class intervals.

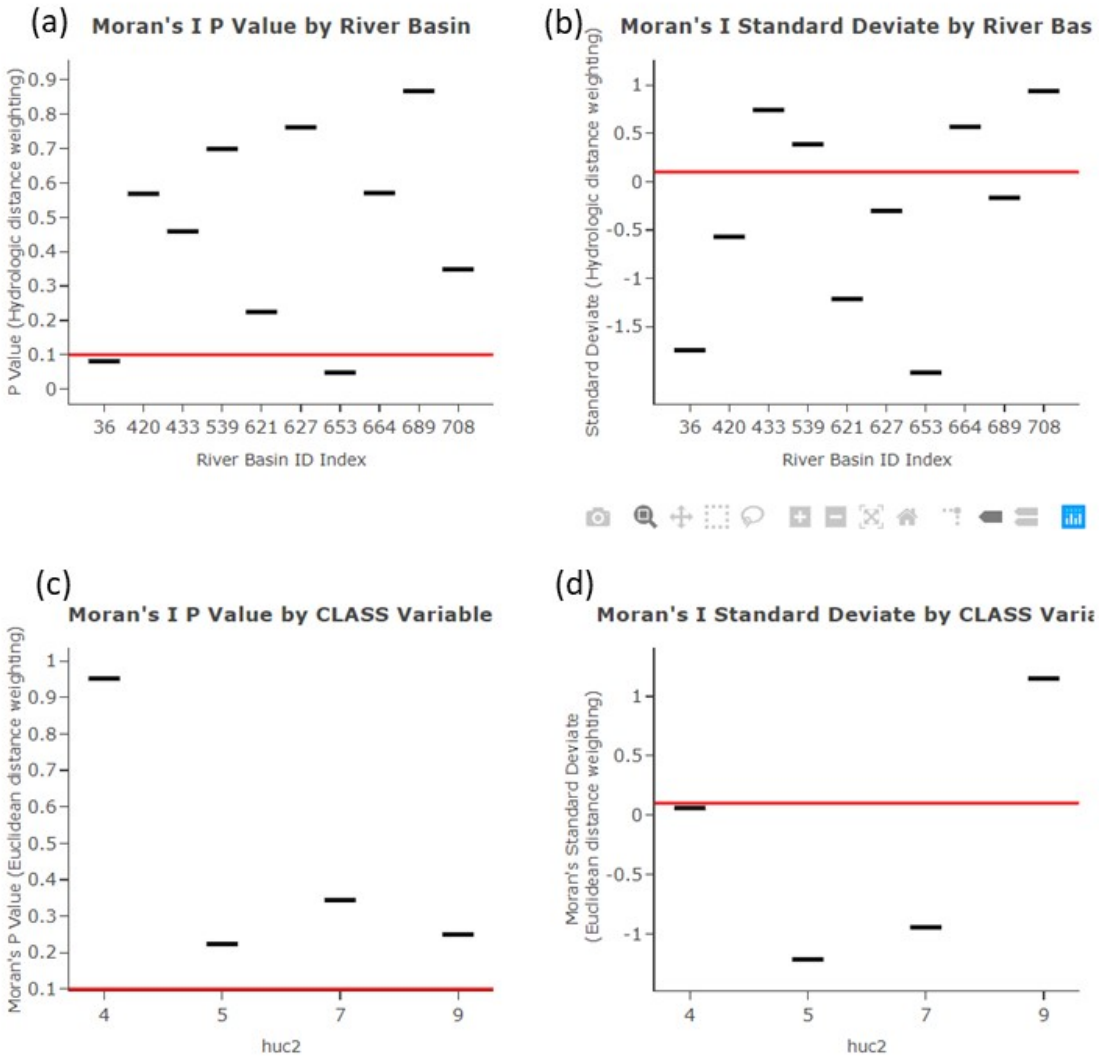


Figure 10: **Results for the Moran's I test for spatial autocorrelation.** (a) P-value for a hydrologic-distance weighted test in river basins; (b) Standard deviate for a hydrologic-distance weighted test in river basins; (c) P-value for a Euclidean-distance weighted test for user-specified spatial classes and total area; and (d) Standard deviate for Euclidean-distance weighted test for user-specified spatial classes and total area. A red line is shown for a p value of 0.10 in plots (a) and (c). The class variable is associated with hydrologic regions in the Midwest USA: 4=Great Lakes, 5=Ohio, 7=Upper Mississippi, and 9=Red-Rainy.

Note that the `classvar` classification of the monitoring station diagnostic metric is based on the class value for the stream reach segment associated with the monitoring station. More precise classifications that account for areas beyond the reach catchment require user pre-processing of the classification variable.

4.4.5.3 Land-use classification variables

```
#-------------------------------------------------------------------------------
  # Specify non-contiguous land use classification variables for boxplots of
  #   observed-predicted ratios by decile class
  # Note that the land use variables listed for "class_landuse" should be defined
  #   in areal units (e.g., sq. km) in the data1.csv file or by defining the
  #   variables in the userModifyData.R function.
  #  In the RSPARROW diagnostic output plots, land use is expressed as a
  #   percentage of the incremental area between monitoring sites.

  class_landuse<-NA   # for NA, total drainage area is used as the default classification
  class_landuse <- c("forest","agric","urban","shrubgrass")
```

This optional setting specifies discrete land-use classification variables that are used for the display of selected model diagnostic plots. The plots can be used to assess model prediction bias and precision. Boxplots are displayed separately for the ratio of the observed to predicted loads for the deciles of each specified land-use class, expressed as the land-use percentage of the summed incremental drainage area between monitoring stations. For land-use classes where fewer than 10 unique categories of monitoring stations can be calculated, a scatterplot is reported for the ratios of the observed to predicted loads vs. the values of the land-use percentages.

The land use variables listed for `class_landuse` should be defined in areal units (e.g., square kilometers) in the *data1.csv* file or by defining the variables in the *userModifyData.R* script.

The values of the classification variable are computed for the incremental drainage area between monitoring sites, based on the summation of the land-use areas for the individual reaches located in the incremental areas between sites. The length of the classification variable should be equivalent to the total number of reaches and is identical to the length of the `classvar` variable.

```
#-------------------------------------------------------------------------------
 # Produces a summary table of reach predictions of the total yield for watersheds
 #  with relatively uniform land use.
 # Specify the minimum land-use percentages of the total drainage area above a
 #  reach to select reaches with uniform land use for the land uses listed in the
 #  "class_landuse" setting.
  class_landuse_percent<-c(90,50,80,10)
```

This setting, used in combination with the `class_landuse` setting, produces a summary of reach predictions of the total yield (*yield_total*; see sub-section 4.4.7) for watersheds with relatively uniform land use, as measured by the minimum land-use percentages of the total drainage area above a reach as specified by the `class_landuse_percent` setting.

The reach predictions are reported in *(run_id)_summary_predictions.csv* file (see sub-section 5.1.4.5). Because contaminant yields have been reported to vary in a generally systematic manner with land use based on small catchment data (e.g., for nutrients, Alexander et al., 2008), the summary of the RSPARROW total yields for relatively uniform land uses can be informative to use to assess the general physical interpretability of the model predictions. Note that SAS SPARROW computes the minimum land-use percentages for the reach incremental drainage area and reports the incremental yield for watersheds with uniform land uses.

Example output of selected metrics from the *(run_id)_summary_predictions.csv* file is shown below for the tutorial model 6. The model predictions of yield are shown for catchments dominated by four land use types. The number of watersheds satisfying the `class_landuse_percent` criteria for "dominant" land use is

reported along with the mean, standard deviation (SD), and percentiles of the distributions of yields (more percentiles are reported in the RSPARROW output table).

```
     Land Use Yield and Units              Number  Mean     SD    10th  50th   90th
                                         Watersheds
90%  forest_Yield Total (kg/ha/yr)            292   4.5    1.5    2.7   4.2    6.7
50%  agric_Yield Total (kg/ha/yr)            4813  17.7   10.5    6.5  15.9   30.1
80%  urban_Yield Total (kg/ha/yr)              32  87.4  171.0    2.2  28.1  196.6
10%  shrubgrass_Yield Total (kg/ha/yr)        877   5.7   11.8    1.3   3.4   13.2
```

Note that the `classvar`, `class_landuse` and `class_landuse_percent` control settings are included in the global input object list *class.input.list*.

4.4.5.4 Bivariate correlations for explanatory variables

```
#-------------------------------------------------------------------------------
  # Specify whether bivariate correlations among explanatory variables are to be executed
  #   by specifying 'parmCorrGroup' in the parameters.csv file as "1" (yes) or "0" (no)
  if_corrExplanVars <- "yes"
```

This setting executes bivariate Spearman's Rho correlations for all combinations of the user-selected explanatory variables in the model, where a value of "1" is entered in the column labeled *parmCorrGroup* in the *parameters.csv* file. A value of "0" should be entered to exclude the variable from the correlation analyses. The setting outputs results to the *(run_id)_explvars_correlations.txt* and *(run_id)_explvars_correlations.pdf* files in the "(run_id)/estimate" sub-directory.

If more than 10 monitoring sites are identified for use in model estimation, then bivariate correlation results are reported for the area-weighted mean of the explanatory variables for the incremental drainage area between monitoring sites. By default, correlation results are also reported for all reaches; these include correlation matrices for all observations (with N equal the number of reaches), a subsample of observations (N=500), and the logged values for the subsampled observations. The subsample of n=500 and the logged transformed observations are reported to assist with the viewing and interpretation of the bivariate plots (see sub-section 5.1.1.2), in cases where the number of reaches are large and nonlinearities occur in the relations.

Summary metrics (mean and quartiles) and boxplots are also reported for the explanatory variables in the *(run_id)_explvars_correlations.txt* and the *(run_id)_explvars_correlations.pdf* files, respectively (see Chapter 5.2.1 for details). The raw data and summary metrics are also output to the R binary object *(run_id)_Cor.ExplanVars.list* (see Chapter sub-section 5.2.1.3 for details).

As a diagnostic for estimated SPARROW models, the area-weighted mean values of the user-selected explanatory variables are also plotted against the observed to predicted ratio for the calibration stations. The plots provide visual information about the accuracy of the model specification for each of the explanatory variables; in general, no correlation should be visible between the ratios and the mean values of the explanatory variable if the model is properly specified. The plots are output only if the control setting `if_corrExplanVars<-"yes"` (section 5 of the control script) is selected and only for the modeled variables where a value of "1" is entered in the *parmCorrGroup* labeled column in the *parameters.csv* file.

4.4.6 Selection of validation sites (section 6 of control script)

```
#-------------------------------------------------------------------------------
  # Split the monitoring sites into calibration and validation set
  if_validate <- "yes"

  # Indicate the decimal fraction of the monitoring sites as validation sites
  #  Two methods are avilable for selecting validation sites (see documentation)
  pvalidate <- 0.25
```

These control settings allow users to withhold a subset of the possible calibration monitoring stations as validation sites for independently evaluating the model performance. The model performance at the validation sites is intended to be used to assess the general robustness of the calibrated model at an independent set of stream locations; there are inherent limitations to obtaining a realistic validation of a model, related to the station sample size and other unique differences in the characteristics of the validation and calibration sites.

Model performance metrics and diagnostic graphics are output for the validation sites, based on simulated model predictions (i.e., no monitoring substitution is peformed and unconditioned predictions are used in the simulation). These results are output to the the model summary file *(run_id)_summary.txt* and the diagnostics PDF file *(run_id)_validation_plots.html* in the "/estimate" sub-directory. These performance results should be compared with those for the simulated (unconditioned) predictions that are reported for the calibration monitoring sites in the *(run_id)_summary.txt*.

Two methods are available to select validation sites in cases where `if_validate<-"yes"`. The method selection is controlled by the value of the `pvalidate` setting. The first method executes a random selection of validation sites, whereas the second allows users to manually identify the validation sites. Both methods select the validation sites from the user-defined set of possible calibration monitoring sites, as designated by the REQUIRED *calsites* variable (with length equal to the number of reaches). The set of calibration sites should have valid mean annual load values and satisfy other user quality assurance requirements. The selection of validation sites triggers the creation of an R binary object *vsitedata*, containing attributes for the validation sites based on a subsetting of the *subdata* R object–see Chapter sub-section 5.1.3 for details.

- Method 1 (`pvalidate` > 0): The validation sites are randomly selected from the user-designated set of possible calibration monitoring sites (as indicated by the *calsites* variable). The fraction of the sites used for validation is specified according to the fraction `pvalidate`. The fraction defined by 1-`pvalidate` is used for model calibration. The *iseed* setting in section 4.4.10 ("Prediction uncertainties") provides a user-selected initial integer seed value to populate the argument of the R random number generator function "set.seed" for the random selection of calibration sites and validation sites. Use of this *iseed* value in subsequent RStudio runs of the same model (with the same data filtering criteria) will reproduce the identical selection of validation and calibration sites.
- Method 2 (`pvalidate` $= 0$): The validation sites are identified by the indicator values of the user-defined variable *valsites* (0=no selected; 1=selected). Users are required to use this variable name. The variable must be either placed in the original *data1.csv* file or defined using a statement in the *userModifyData.R* script (with length equal to the number of reaches). The *valsites* variable should also be specified as an OPEN variable in the *dataDictionary.csv* file. Sites identified as validation sites (i.e., *valsites*=1) should be a subset of the user-defined set of possible calibration monitoring sites, such that a value of `1` should be denoted in both the *calsites* and *valsites* variables for reaches associated with the validation stations.

4.4.7 Model predictions (section 7 of control script)

4.4.7.1 Prediction types and variables

```
#-------------------------------------------------------------------------------
  # Specify if standard predictions are to be produced.
  #  Note: Bias retransformation correction is applied to all predictions,
  #    except "deliv_frac", based on the Smearing estimator method

  if_predict <- "yes"
```

This setting outputs the standard set of RSPARROW predictions of load and yield shown in the R comment section below to two files in the model *"predict"* subdirectory: *(run_id)_predicts_load.csv* and *(run_id)_predicts_yield.csv* (see Table 3). The predictions are also required for automated and interactive mapping of the predictions, using either the setting `master_map_list` and `enable_shinyMaps<-"yes"`, respectively.

```
  # Load prediction names and explanations
  #   pload_total                Total load (fully decayed)
  #   pload_(sources)            Total source load (fully decayed)
  #   mpload_total               Monitoring-adjusted (conditional) total load
  #                                 (fully decayed)
  #   mpload_(sources)           Monitoring-adjusted (conditional) total source load
  #                                 (fully decayed)
  #   pload_nd_total             Total load delivered to streams (no stream decay)
  #   pload_nd_(sources)         Total source load delivered to streams (no stream decay)
  #   pload_inc                  Total incremental load delivered to reach
  #                                 (with 1/2 of reach decay)
  #   pload_inc_(sources)        Source incremental load delivered to reach
  #                                 (with 1/2 of reach decay)
  #   deliv_frac                 Fraction of total load delivered to terminal reach
  #   pload_inc_deliv            Total incremental load delivered to terminal reach
  #   pload_inc_(sources)_deliv  Total incremental source load delivered to terminal reach
  #   share_total_(sources)      Source shares for total load (percent)
  #   share_inc_(sources)        Source shares for incremental load (percent)

  # Yield prediction names and explanations
  #   concentration              Flow-weighted concentration based on decayed total load
  #                                 and mean discharge
  #   yield_total                Total yield (fully decayed)
  #   yield_(sources)            Total source yield (fully decayed)
  #   myield_total               Monitoring-adjusted (conditional) total yield
  #                                 (fully decayed)
  #   myield_(sources)           Monitoring-adjusted (conditional) total source yield
  #                                 (fully decayed)
  #   yield_inc                  Total incremental yield delivered to reach
  #                                 (with 1/2 of reach decay)
  #   yield_inc_(sources)        Total incremental source yield delivered to reach
  #                                 (with 1/2 of reach decay)
  #   yield_inc_deliv            Total incremental yield delivered to terminal reach
  #   yield_inc_(sources)_deliv  Total incremental source yield delivered to
  #                                 terminal reach
```

RSPARROW outputs three major types of load predictions: incremental, delivered incremental, and total load (Fig. 11a). A non-decayed version of the load predictions with the *_nd_* label (i.e., no aquatic decay applied) is also output to the *(run_id)_predicts_load.csv* file.

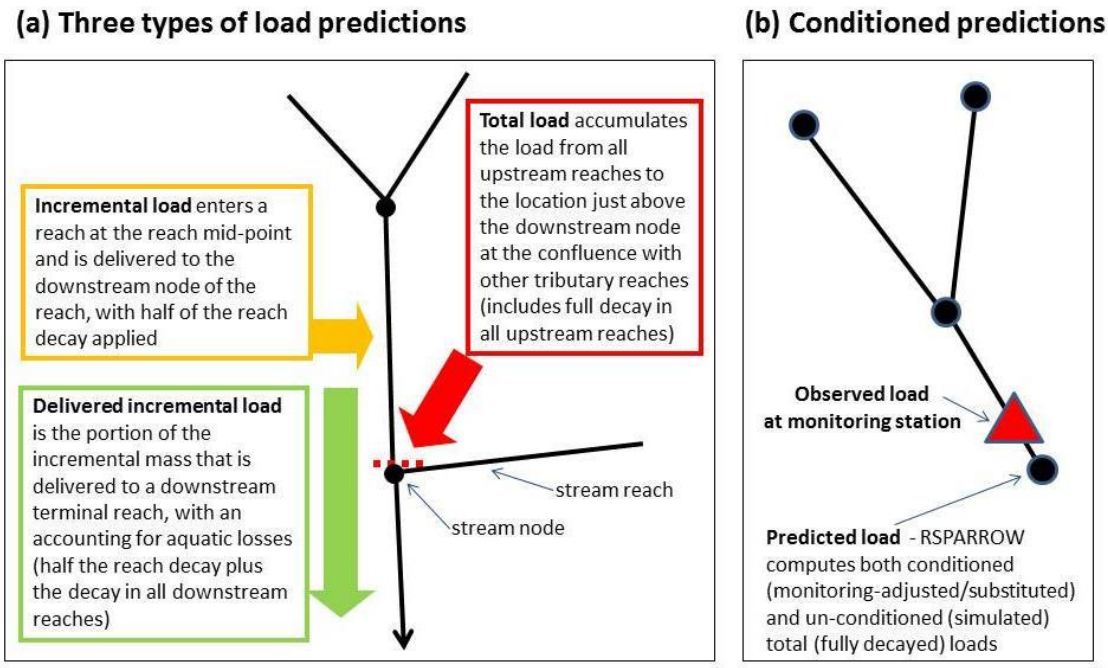


Figure 11: **Illustration of RSPARROW load predictions for reaches in a simple stream network: (a) the three major prediction types, with applied aquatic decay; (b) conditioned predictions, which are obtained by substituting the observed load on monitored reaches for the model predicted load.**

RSPARROW also automatically outputs both *"conditioned"* (monitoring-station adjusted) and *"unconditioned"* (simulated) loads, yields, and source shares. The conditioned loads and yields are labeled as *mpload_* and *myield_* in the output files as shown above in the listing of prediction variables. Note that this differs from SAS SPARROW, which requires separate model runs with the *%if_adjust = "yes/no"* setting to output these two prediction types.

Conditioned (monitoring-station adjusted) predictions are calculated by substituting the observed load on reaches with monitoring stations for the model predicted load (Fig. 11b), and have the following attributes:

- They provide the most accurate reach predictions for quantifying river loads and estimating the model coefficients that quantify the effects of contaminant sources and hydrological and biogeochemical processes on stream quality (Schwarz et al., 2006). Use of the conditioned predictions reduces the downstream propagation of model errors and their correlation across watersheds (Schwarz et al., 2006). Thus, conditioned predictions are preferred for NLLS model estimation/calibration to obtain the most accurate and physically interpretable model and to evaluate the adequacy of the model fit to observations. The predictions are also preferred to provide the most accurate estimates of the stream load inputs to managed or regulated water bodies (e.g., estuary, reservoir, stream reach).
- They adhere to the following assumptions and methods: (1) conditioning affects the estimated load in all reaches downstream of the monitored reach; (2) monitoring stations are assumed to be located at the downstream node of a monitored reach; and (3) source shares for conditioned reach predictions are apportioned according to the source shares for the unconditioned loads which conserve mass.

Unconditioned predictions are calculated by executing the model in simulation mode, using mean coefficients from the NLLS model estimated with monitoring-adjusted (conditioned) predictions; thus, the simulated predictions are not adjusted for the observed loads on monitored reaches. Unconditioned predictions have the following attributes:

- They preserve mass balance in the calculations of contaminant transport in stream reaches and mass delivery to downstream waters, including calculations of the contributions of mass from contaminant sources to reaches (i.e., "source shares"). Thus, unconditioned predictions are preferred to support investigations of contaminant sources and process-based studies of stream transport processes, which depend on the conservation of mass.
- They provide the best representation of the predictive skill of the estimated model at monitored locations (calibration and validation sites). Performance metrics based on the unconditioned predictions are well suited for comparing the prediction accuracy of different models because conditioning effects are removed (i.e., the effects of substituting the observed station loads for the model predicted loads). Model performance metrics based on the unconditioned predictions also give a generally preferred estimate of the expected average accuracy of the model when the model is applied to unmonitored stream reaches.

The standard RSPARROW predictions of the mean load and yield, reported in the output files *(run_id)_predicts_load.csv* and *(run_id)_predicts_yield.csv*, are corrected for log-retransformation bias (with the exception of the non-mass type predictions of the delivery factor, *deliv_frac*). SPARROW model predictions require adjustment for log-retransformation bias because the model is estimated using log transformed values of the observed and predicted loads; simple exponentiation of the log-transformed mean predicted load, without bias correction, would yield only an estimate of the median rather than the mean load.

The bias correction is determined using "Smearing Estimator" methods (Duan, 1983). The Smearing bias-correction factor is computed according to equation 1.124 in Schwarz et al. (2006), which is expressed as the mean of the exponentiated leverage-weighted log model residuals. The bias correction is applied as a multiplicative factor to the mean predictions of loads and yields in all stream reaches. The value of the bias-correction factor is reported in the *(run_id)_summary.txt* file for each model as the `Mean Exp Weighted Error`. Using the Smearing factor with NLLS methods gives predictions that are unbiased for large samples regardless of the underlying error distribution, but the predictions are potentially biased in small samples even with normal residuals; although weighting of the residuals may reduce bias in small samples (Schwarz et al., 2006).

Note that the delivery-related prediction metrics (e.g., *deliv_frac*, *pload_inc_deliv*, *yield_inc_deliv*) quantify the load, yield, and the fraction of the load that is delivered to user-defined terminal reaches. The terminal reaches are specified by the system variable *target* (see Table 4), as defined in the *data1.csv* file and/or modified in the *userModifyData* script. The terminal reaches are typically defined by users as the most downstream terminus of the stream networks, such as reaches associated with coastal fall-line or international boundaries, inland lakes, or the downstream margins of the modeled spatial domain.

4.4.7.2 Prediction units

```
#-------------------------------------------------------------------------------
  # Specify the load units for predictions and for diagnostic plots
  loadUnits <- "kg/year"
```

This setting specifies the units of the load predictions that appear in various *.csv* and *.txt* files and in the diagnostic plots and their axis labels.

```
#-------------------------------------------------------------------------------
  # Specify the concentration conversion factor, computed as
  #  Concentration = load / discharge * ConcFactor
  #  (for predictions and diagnostic plots)
  ConcFactor <- 3.170979e-05   # kg/yr and m3/s to mg/L
  ConcFactor <- 0.001143648    # kg/yr and ft3/s to mg/L

  ConcUnits <- "mg/L"
```

These two settings allow the proper units to be calculated and labeling assigned, respectively, for the constituent concentration value (e.g., milligrams per liter) that is output to the prediction *.csv* yield file. Two examples are shown above for factors in cases where the mean annual load is expressed as kilograms per year and the mean annual discharge is expressed either as cubic meters per second or cubic feet per second.

```
#-------------------------------------------------------------------------------
  # Specify the yield conversion factor, computed as Yield = load / demtarea * yieldFactor
  #  (for predictions and diagnostic plots)
  yieldFactor <- 1/100    # convert kg/km2/yr to kg/ha/year use 1/100 factor
  yieldFactor <- 0.001    # convert m3/year to mm/year

  yieldUnits <- "mm/year"
```

These two settings allow the proper units to be calculated and labeling assigned, respectively, for the constituent yield values (mass per unit area per time) that are output to the prediction *.csv* yield file and the model residuals file. The control settings also allow users to control the units and labeling that appear in the yield-related diagnostic plots.

4.4.7.3 Output of additional variables to the prediction files

```
#-------------------------------------------------------------------------------
  #Specify additional variables to include in prediction, yield, residuals CSV files
  add_vars<-NA
  add_vars<-c("huc2")
```

The setting allows users to specify any of the variables defined in the *dataDictionary.csv* file as additional variables for output to the standard prediction, yield, and residuals CSV files. These variables will appear as columns entered to the right of the standard set of output variables for these files.

4.4.8 Diagnostic plots and maps (section 8 of control script)

The settings in this section allow user control over the management of the input/output of geographic shape files for streams, catchments, and base (line) reference mapping and display settings for diagnostic plots and maps and prediction maps.

4.4.8.1 Shape file input/output and geographic coordinates

```
#-------------------------------------------------------------------------------
# Shape file settings

# Identify the stream reach shape file and 'waterid' common variable in the shape file
lineShapeName <- "ccLinesMRB3"
lineWaterid <- "MRB_ID"

# Identify the stream catchment polygon shape file and 'waterid' common variable
#   in the shape file
polyShapeName <- "mrb3_cats_usonly_withdata_tn"
polyWaterid <- "MRB_ID"

# Identify optional geospatial shape file for overlay of lines on stream/catchment maps
LineShapeGeo <- NA
LineShapeGeo <- "states"

# Identify the desired Coordinate Reference System (CRS) mapping transform for
#   geographic coordinates (latitude-longitude)
CRStext <- "+proj=longlat +datum=NAD83"

# Indicate whether shape files are to be converted to binary
if_create_binary_maps<-"no"

# Convert shape files to binary using the RSPARROW setting names as listed:
convertShapeToBinary.list <- c("lineShapeName","polyShapeName","LineShapeGeo")
convertShapeToBinary.list <- c("lineShapeName")
```

These settings allow user specification of the shape file names and unique identification tags for the streams and catchments that allow match-merge operations between the shape files and R internal objects that contain reach and catchment attributes from the user's `input_data_fileName` (e.g., *data1.csv* file). The user's shape projection file is required to define the geographic coordinates for internal processing of the shape file.

The setting `if_create_binary_maps<-"yes"` enables the conversion of shape files to *sf* binary files; this command must be executed at least once to use any of the mapping capabilities in RSPARROW, which operate exclusively from the binary files. Use of the binary shape files enhances map display and PDF/HTML output processing speeds. This setting is used in combination with the `convertShapeToBinary.list` setting. Once the binary files are produced, the setting `if_create_binary_maps<-"no"` should be used, unless the files need to be replaced with new or altered shape files. **USER CAUTION:** RSPARROW version 1.1.0 creates *sf* binary files, which provides reduced processing times compared with those for the version 1.0 *sp* binary files. Version 1.0 users must re-create the binary files from their shape files.

Note that only geographic (latitude-longitude) projections are currently supported for RSPARROW map output to HTMLS, PDFs, and in R Shiny map displays (i.e., `CRStext <- "+proj=longlat +datum=NAD83"`). The maps render using a Mercator projection. Note that the *leaflet* option in the R Shiny mapper provides a Google Mercator projection, with a map display CRS=EPSG:3857 and a data CRS=EPSG:4326. No edits should be made to the `CRStext` setting in the current RSPARROW version.

```
#------------------------------------------------------------------------------

# Select ERSI shape file output for streams, catchments, residuals, site attributes
outputESRImaps <- c("no","no","no","no")   # c("yes","yes","yes","yes")
```

This setting controls the production and output of ESRI shape files, which can be optionally specified for four types of map output (see Chapter sub-section 5.3.4 for details on the directory structure and output file names). The shape files may include:

- Streams and catchments: This controls the stream and catchment maps of the variables specified in the `master_map_list` setting. These variables may include any of the standard SPARROW prediction variables (listed above in Chapter sub-section 4.4.7), the prediction uncertainties (as described in Chapter sub-section 5.2.2.3), or any of the *sparrowNames* variables listed in the *dataDictionary.csv* file.
- Residuals: This controls maps of the model residuals that are produced when an RSPARROW model is estimated, which is enabled by the setting `if_estimate<-"yes"`.
- Site attributes: This controls the mapped output of monitoring site variables specified in the `map_siteAttributes.list` setting shown below in section 8 of the control script. This may include any of the *sparrowNames* variables listed in the *dataDictionary.csv* file.

```
#------------------------------------------------------------------------------
# Specify the geographic units minimum/maximum limits for mapping
#   and prediction maps.
# If set to NA, limits will be automatically determined from the
#   monitoring site values.
lat_limit <- c(35,50)
lon_limit <- c(-105,-70)
```

These settings allow users to specify the latitude and longitude boundaries for the production of maps of site attributes, streams, and catchments. Specification of a `NA` value will automatically determine the map boundaries based on the longitude-latitude of the calibration sites.

4.4.8.2 Maps of model predictions and *dataDictionary* variables

```
#------------------------------------------------------------------------------
#Prediction and station attribute maps for streams/catchments

# Identify list of load and yield predictions for mapping to output files
#  (enter NA for none)
# Any variables listed in the data-dictionary are available for mapping by
#   streams or catchments
# Note: To map model predictions, then 'if_predict' must = "yes" or predictions
#   must have been previouly executed
master_map_list <- NA
master_map_list <- c("pload_total","pload_inc")

#Identify type of map(s) to output to PDF file from "stream","catchment", or "both"
output_map_type<-"stream"
```

These settings allow model predictions (listed above in Chapter sub-section 4.4.7) or any of the numeric *sparrowNames* variables listed in the *dataDictionary.csv* file to be specified (`master_map_list`) and mapped to streams and/or catchments as selected in the `output_map_type` setting. The maps are output to PDF or HTML files, with non-interactive or interactive features, according to the control settings described in Chapter sub-section 4.4.8.4 below.

The execution of these statements requires that user's have produced the model predictions

(`if_predict<-"yes"`) either in the current or a past RStudio session.

```
#-------------------------------------------------------------------------------
# map display settings for model predictions or dataDictionary variables
predictionTitleSize<-1
predictionLegendSize<-0.6
predictionLegendBackground<-"grey"
#length sets number of breakpoints
predictionMapColors<-c("blue","dark green","gold","red","dark red")
predictionClassRounding<-1
predictionMapBackground<-"white"
lineWidth<-0.8 #for stream maps #0.8
```

These settings allow user customization of a variety of features of the maps of model predictions or dataDictionary variables that are produced for reaches and catchments. The settings are self-explanatory based on the control variable name and comments. Optional R colors can be found here: http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf. Mapping is executed using base plotting syntax.

4.4.8.3 Model diagnostics: Station attribute maps, model plots, residual maps

4.4.8.3.1 Station attribute maps

```
#-------------------------------------------------------------------------------
# SiteAttribute maps - Identify site attributes to map
#  Note that any variables in the dataDictionary.csv can be mapped
map_siteAttributes.list<-c("meanload","meanyield","meanconc","meanloadSE")
# Example site attribute R statements in the 'userModifyData.R' function:
  # meanload <- depvar
  # meanyield <- depvar / demtarea
  # meanconc <- depvar/meanq*ConcFactor
  # meanloadSE <- depvar_se/depvar*100
```

This setting enables the output of monitoring site maps for user-specified site attributes, using any of the *sparrowNames* listed in the *dataDictionary.csv* file. If the variables are not present in the *data1* input file, then users can define these variables in the *userModifyData.R* script, as is currently illustrated above for the SPARROW tutorial model. The maps are output to the beginning pages of the *(run_id)_diagnostic_plots.html* file when a model is executed.

```
#-------------------------------------------------------------------------------
siteAttrTitleSize<-1
siteAttrLegendSize<-1
#siteAttrColors length sets number of breakpoints
#siteAttrColors<-c("blue","green4","yellow","orange","red","darkred")
siteAttrColors<-c("blue","green4","yellow","orange","red")
siteAttrClassRounding<-2
siteAttr_mapPointStyle<-16 #pch=16
siteAttr_mapPointSize<-2  # sets point size scaling
siteAttrMapBackground<-"grey"
```

These settings allow user customization of a variety of features of the site attribute maps that are produced for the monitoring sites for variables specified in the `map_siteAttributes.list` setting. This may include any of the *sparrowNames* listed in the *dataDictionary.csv* file. The settings are self-explanatory based on the control variable name and comments. For further details on "pch", see https://www.rdocumentation.org/packages/graphics/versions/3.4.3/topics/points".

4.4.8.3.2 Diagnostic plots and residual maps from the model estimation

The following control settings allow users to manage features of the diagnostic plots and residuals maps that are output to the *(run_id)_diagnostic_plots.html* file in the "(run_id)/estimate" directory following the estimation of an RSPARROW model, which is enabled by the setting `if_estimate<-"yes"`.

```
#-------------------------------------------------------------------------------
# Diagnostic output from model estimation

# Diagnostic plot settings
diagnosticPlotPointSize <- 0.6
diagnosticPlotPointStyle <- 1
```

The setting `diagnosticPlotPointSize` controls the overall size of the diagnostic plots, including the text, numerical labels for the tic mark indices, and the plotted symbols. The setting `diagnosticPlotPointStyle` controls the style of the plotted points associated with the numerical values for the *pch* command (e.g., 0=open square; 1=open circle; 15=solid square; 16=solid circle). For further details on "pch", see https://www.rdocumentation.org/packages/graphics/versions/3.4.3/topics/points".

```
#-------------------------------------------------------------------------------
#Residual maps
#specify breakpoints for mapping of residuals
#if residual_map_breakpoints set to NA, then breakpoint defaults will be applied
#   breakpoint defaults are c(-2.5,-0.75,-0.25,0,0.25,0.75,2.5)
#   breakpoints must have a length of 7 and be centered around 0
residual_map_breakpoints<-c(-2.5,-0.75,-0.25,0,0.25,0.75,2.5)
# for obs/pred ratio maps must have length=7
ratio_map_breakpoints<-c(0.3,0.5,0.8,1,1.25,2,3.3)
residualTitleSize<-1
residualLegendSize<-1
#residualColors must be length=8 corresponding to residual_map_breakpoints
residualColors<-c("red","red","gold","gold","dark green","dark green","blue","blue")

#residualPointStyle must be length=8 corresponding to residual_map_breakpoints
                                        # 2 = open upward triangle
                                        # 6 = open downward triangle
                                        # 1 = open circle
residualPointStyle<- c(2,2,1,1,1,1,6,6)
#must be length=8 corresponding to breakpoints
residualPointSize_breakpoints<-c(0.75,0.5,0.4,0.25,0.25,0.4,0.5,0.75)
#residualPointSize_factor causes symbol size to increase or decrease
#  This scales all point sizes in residualPointSize_breakpoints
residualPointSize_factor<-2
residualMapBackground<-"grey"
```

These settings allow user customization of a variety of features of the residual maps that are produced for the calibration monitoring sites. The settings are self-explanatory based on the control variable name and comments.

Note that all of the above mapping customization settings have default values that are assigned in the *setMapdefaults.R* function; therefore, specifying an `NA` for the above settings will trigger the default values.

4.4.8.4 Output of Web browser-accessible *plotly* interactive plots and maps

```
#------------------------------------------------------------------------------
#Enable plotly interactive displays for maps (interactive plots are automatic)
enable_plotlyMaps<-"yes"
add_plotlyVars<-c("waterid","rchname","staid")
showPlotGrid<-"no"
```

The `enable_plotlyMaps<-"yes"` setting outputs interactive *plotly* maps (i.e., browser-accessible HTML files) for the station attributes and residuals (section 4.4.8.3) and the stream/catchment predictions and *dataDictionary* variables that are listed in the `master_map_list` control setting (section 4.4.8.2). Note that *plotly* interactive plots are automatically output in the model diagnostics files (see Table 2 in section 2.3 and Table 6 in section 4.3).

Interactive *plotly* plots and maps are output to HTML files. **The setting `enable_plotlyMaps<-"yes"` produces HTML files that are very large in size; users should expect some delays when opening these files in a browser.** The setting `enable_plotlyMaps<-"no"` outputs non-interactive maps in the HTML diagnostics files but plots remain interactive; this setting also outputs separate PDF files for each of the maps of stream and catchment predictions and *dataDictionary* variables listed in the `master_map_list` control setting (section 4.4.8.2).

Each interactive plot and map in the HTML file displays an R *plotly* tool box banner, located in the upper right portion of each plot and map (e.g., Fig. 12). The tool box allows users to interactively control various features of each image, including the ability to pan/zoom, box selected areas of the image for display, rescale plot axes, sub-select features from the map legend for display, download 'png' files, and display the data associated with plot and map features by hovering the mouse cursor over these features (i.e., "hover-text"). The "hover-text" feature controls the data displayed for individual data points in plots and the monitoring sites, stream segments, and catchment polygons for maps.

The `add_plotlyVars` setting controls which *dataDictionary* variables are displayed in the interactive "hover-text" feature. The latitude and longitude are automatically included in the displayed variable list for site attribute maps. Examples of "hover-text" displays are illustrated for diagnostic plot and map output (Figure 12) and for model predictions of total nitrogen load in midwestern USA streams (Figure 13).

The `showPlotGrid` setting enables the display of grid lines for the bivariate plots and boxplot diagrams.

To reduce the wait-time when opening the HTML files, stream and catchment maps are output to separate HTML files for each prediction type and *dataDictionary* variable that is specified by a user in the `master_map_list` control setting (section 4.4.8.2).

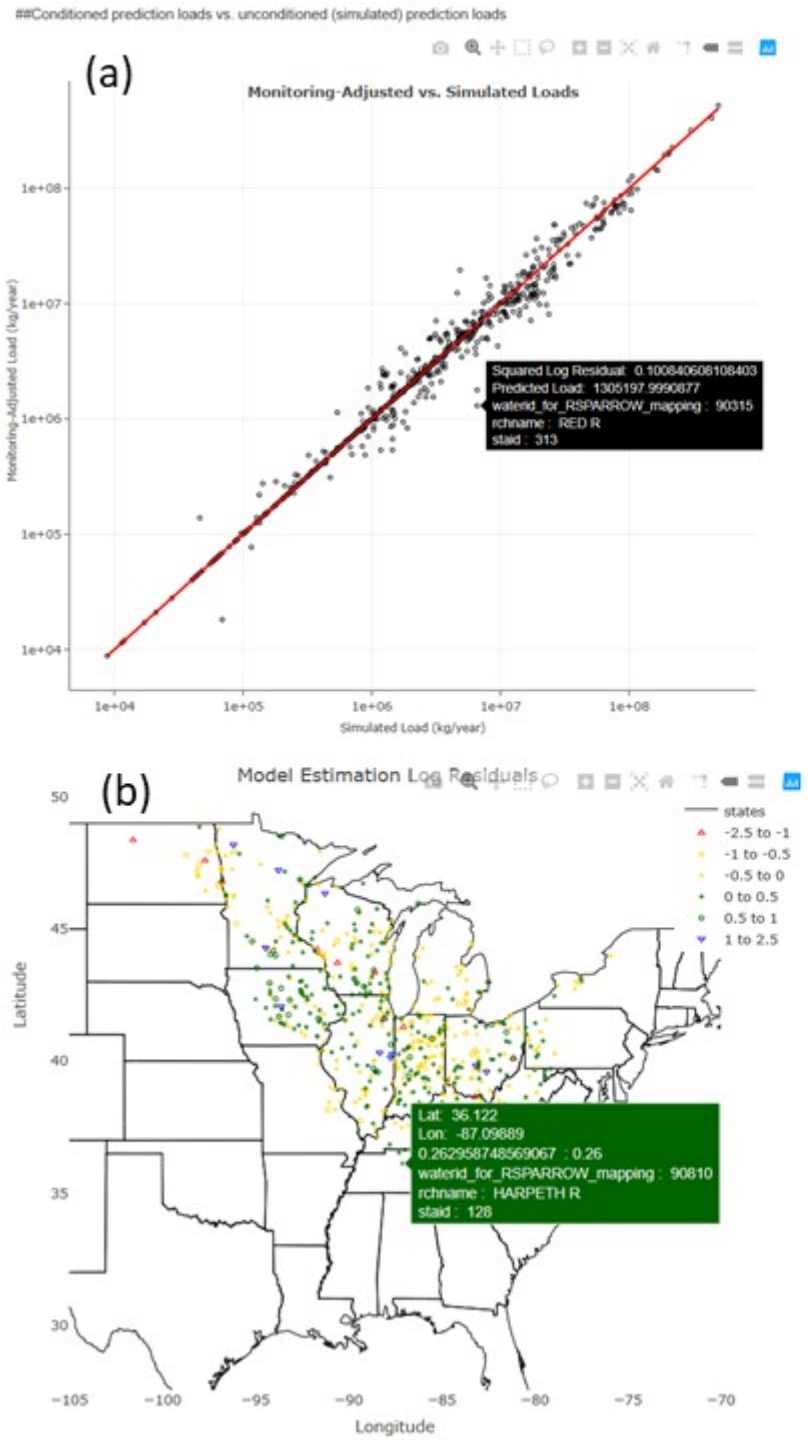


Figure 12: **Example interactive diagnostic displays with hover-text:** (a) Plot of monitoring-adjusted loads vs. simulated loads; (b) Map of the model estimation log residuals at the calibration monitoring sites. The displays are for results from the tutorial model 6. Map output is from the "Model6_diagnostic_plots.html" file in the "estimate" sub-directory.

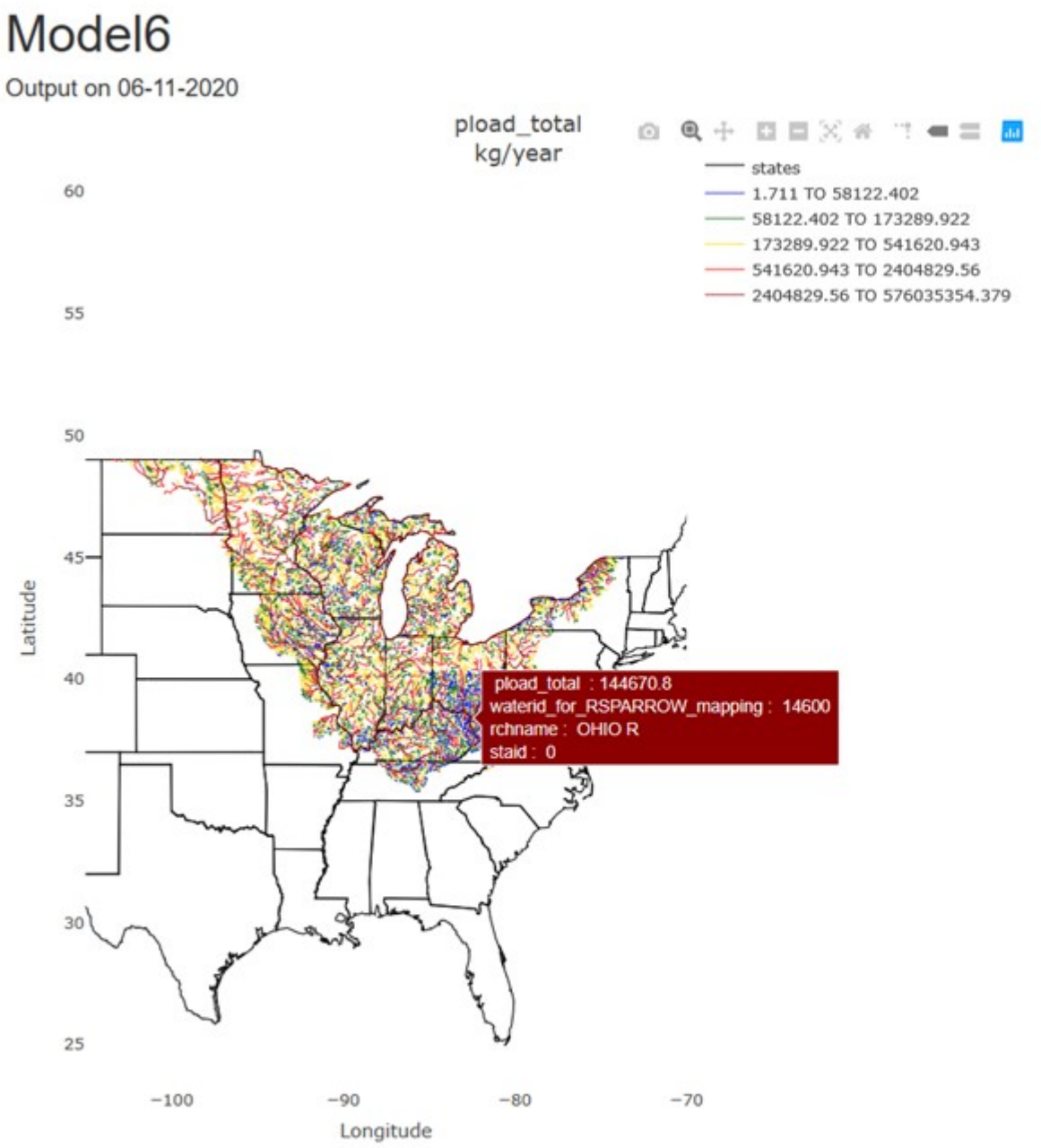


Figure 13: **Example interactive display for model predictions with hover-text:** The display is for results from the tutorial model 6. Map output is from the "Model6_pload_total.html file" in the "maps" sub-directory.

4.4.9 RShiny Decision Support System (DSS) mapper (section 9 of control script)

4.4.9.1 Enabling the interactive RShiny mapper in a Web browser

```
#-------------------------------------------------------------------------------
#Enable the interactive RShiny mapper
enable_shinyMaps<-"yes"

#Specify preferred web browser for the RShiny display
path_shinyBrowser<-"C:/Program Files/Mozilla Firefox/firefox.exe"
path_shinyBrowser<-"C:/Program Files (x86)/Microsoft/Edge/Application/msedge.exe"
path_shinyBrowser<-"C:/Windows/SystemApps/Microsoft.MicrosoftEdge_.../MicrosoftEdge.exe"
path_shinyBrowser<-"C:/Program Files (x86)/Google/Chrome/Application/chrome.exe"
path_shinyBrowser <- NA      # user's default browser
```

The `enable_shinyMaps<-"yes"` setting enables an R Shiny interactive mapping interface within a Web browser window. The browser type is specified in the `path_shinyBrowser` control setting. From R Shiny, users can map RSPARROW predictions and variables (*sparrowNames*) in the data dictionary by stream reach, catchment, and monitoring site location and interactively define and execute hypothetical source-change management scenarios (see Chapter sub-section 4.4.9.2). A separate *Rscript.exe* window also opens in a black box during the R Shiny session, where message output is written including warning messages. If execution errors occur, the black box will close and the application window will become inoperable and grey in color; an error message will be written to the *error.log* file.

The R Shiny mapper is not enabled if a user executes a model in batch mode (i.e., `batch_mode<-"yes"`; section 11 of the control script). Note that the R Shiny mapper contains a batch-mode option to support map production for larger watersheds that require longer execution times. This option allows users to output maps of multiple variables (i.e., prediction, uncertainty, and *dataDictionary*) in a single execution.

4.4.9.1.1 R Shiny mapper menu options

The R Shiny menu options will change with the user's selections, from top to bottom, with different input boxes appearing based on each upper level selection by the user. An example of the R Shiny mapper interface is shown in Figure 14 for the map of total nitrogen yield for streams in the Midwest.

The interface provides the following options:

(1) **Ouput Mode: *"Interactive"* or *"Batch"*** - Interactive mode allows users to map one variable at a time, with output directed to CSV files and optionally to PDF or HTML files. Batch mode is recommended for large model domains (e.g., >300K stream reaches), which may require considerable time to fully display. Batch mode allows users to select multiple variables and to execute these in the background with the map output directed to PDF or HTML files (and optionally to ESRI shape files in batch mode). In batch mode, the following windows must remain open: (a) the RStudio session, which is unavailable for use during map production; (b) the *Rscript.exe* window running the shinyApp; and (c) the *Rscript.exe* window running the batch execution.

(2) **Output Map Format: *"static"*, *"plotly"*, or *"leaflet"*** - The *"static"* option displays non-interactive maps, whereas the *"plotly"* and *"leaflet"* options display interactive maps with capabilities for pan/zoom, hover-text, and other interactive features. Selection of either interactive option will add the button, "Add Hover Variable(s)", which contains a drop down display of the *dataDictionary* variables; this button appears below the *"Mapping Variable"* control. A summary of the features and drawbacks of the static and interactive display options is presented in Figure 15.

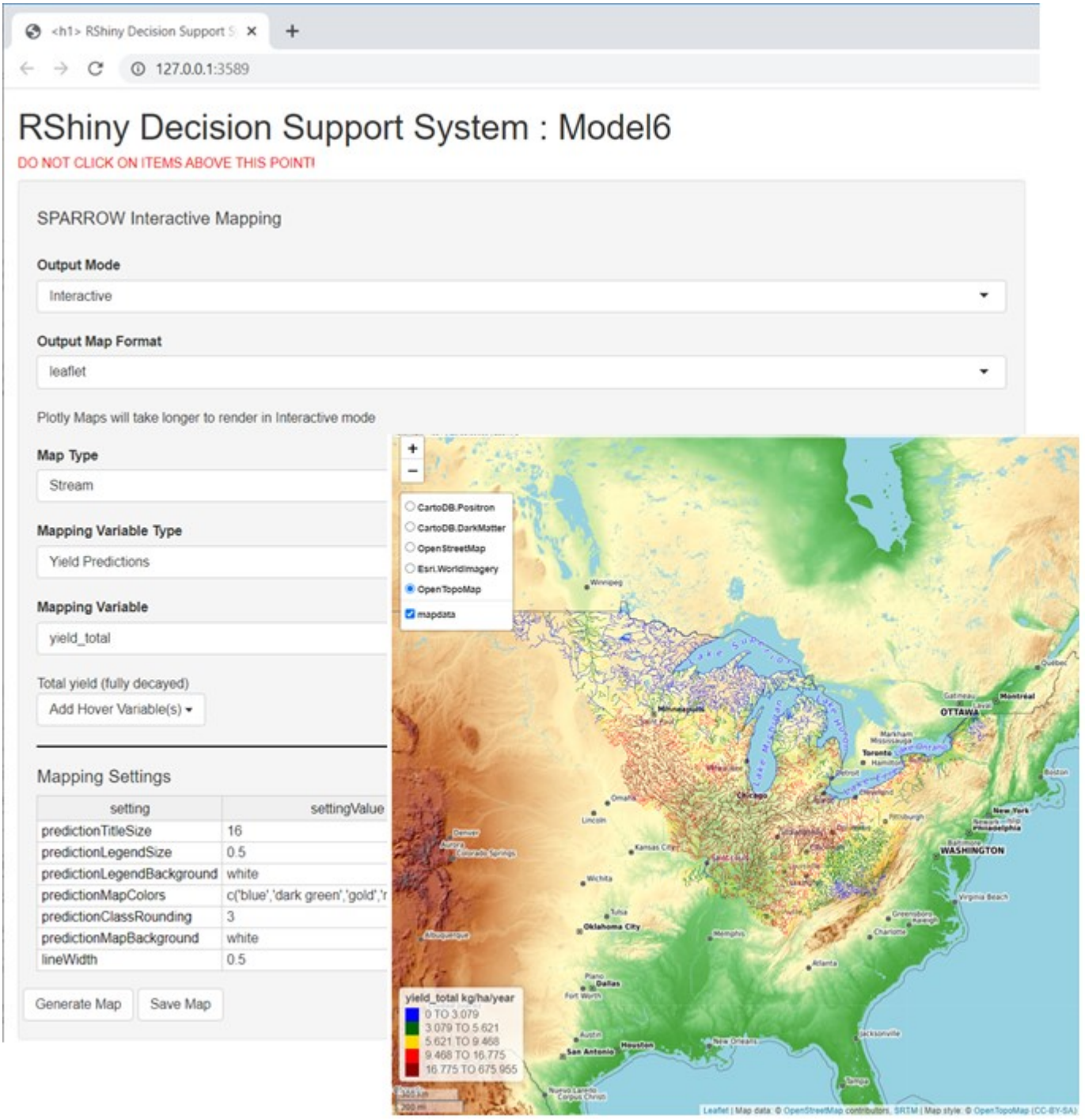


Figure 14: **R Shiny interactive mapper interface and example map for total nitrogen yield for streams in the Midwest.** The leaflet map includes five optional base maps, and is shown for the tutorial model 6. The menu and map panels appear side-by-side in the actual browser display.

Output Options	*Static (ggplot)*	*plotly*	*leaflet*
Features	1. Fast/efficient map generation 2. Customizable via all *Mapping Settings* 3. Includes custom base layer input as shapefile 4. Custom projection	1. Multiple zooming tools 2. Excellent hover performance including a compare hover option (i.e., display of nearest neighbors) 3. Custom base layer input as shapefile 4. Custom projection 5. Legend elements can be easily turned on/off 6. Save output of a customized display to a PNG file (i.e., outputs the user view exactly; e.g., zoomed in and with one color turned off) 7. Save output as interactive HTML, with all *plotly* features intact 8. Customizable using most *Mapping Settings*	1. Zooms faster than *plotly* 2. Includes options to display five standard base maps (e.g., topography, street map, world imagery) 3. Save output as interactive HTML, which retains all interactivity 4. Customizable with some *Mapping Settings* 5. Catchment maps substantially faster to render than in *plotly*
Drawbacks	1. Save output only as PDF 2. Not interactive (zoomable) 3. Only one background layer displayed	1. Time consuming to render 2. Hover text performs better in the saved HTML output file than within R Shiny 3. Catchment maps VERY slow to display; recommend their execution in batch mode and output to HTML 4. Only one background layer displayed	1. Hover text difficult to display, unless the display is zoomed in 2. Legend cannot be subset 3. ONLY available projection is Mercator (map display CRS=EPSG:3857 and data CRS=EPSG:4326)

Figure 15: **Summary of the features and drawbacks for the three "Output Map Format" options in the R Shiny interactive browser interface.** Note that only geographic (latitude-longitude) map projections are currently supported in RSPARROW.

(3) **Map Type: *"Stream"*, *"Catchment"*, *"Site Attributes"*, or *"Source-Change Scenarios"***

- For *"Stream"* or *"Catchment"* map types, users can select from among the following four options for *"Mapping Variable Type"*: *"Load Predictions"*, *"Yield Predictions"*, *"Prediction Uncertainties"*, *"Data Dictionary Variables"*.
  - For the first three prediction-related options for *"Mapping Variable Type"*, users may select a *"Mapping Variable"* for any of the prediction metrics using the names shown in sub-section 4.4.7.1 above; these can only be mapped for a stream or catchment *"Map Type"*.
  - For the data dictionary option for *"Mapping Variable Type"*, users can map any of the numeric *sparrowNames* variables listed in the *dataDictionary.csv* file for streams or catchments.
- For *"Site Attributes"* map type, users can select and map any of the monitoring site attributes that are listed as numeric *sparrowNames* variables in the *dataDictionary.csv* file.
- For the *"Source-Change Scenarios"* map type, users can display and save the results of management source-change scenarios for streams or catchments based on choices for the *"Select output map type"* option (see examples of the menu options and output in Chapter sub-section 6.2.9). Users have control over the selection of sources, source-change factors, reach locations, and a name for each scenario (see Chapter sub-section 4.4.9.2 below for details on the decision support methods).
  - A name can be specified for the scenario, using the *"Enter Scenario Name"* option, with the choice to *"Overwrite"* an existing named scenario using a check box on the menu.
  - The setting *"Select Target Reach Watersheds"* allows users to identify one or more outlet reaches for watersheds where the change scenarios are applied. Users may use the terminal reaches for the network as default outlets (if the scenario is to be applied in all watersheds), or may enter a single or multiple outlet reach(es) or flag reaches in a CSV file *run_id/scenarios/flag_TargetReachWatersheds.csv*) in cases where many reaches are to be designated. RSPARROW identifies all hydrologically connected reach segments upstream of the outlet reaches (using the *hydseqTerm.R* function) where the scenarios are applied.
  - The setting *"Select reaches for applying the scenario (within targeted watersheds)"* allows users to select *"all reaches"* or *"selected reaches"* to apply the scenario. This enables tabular entry of the details of the source-change scenario attributes, including variable names and conditions for the sources (*Percent Change (+/-) Factors*), and reach locations in the targeted watersheds where the scenarios apply (a right click allows the addition of rows to the table to select additional sources or reach locations–see examples in Chapter sub-section 6.2.9). The choice to apply scenarios to *"selected reaches"* provides users with two options for entering the scenario attributes via the setting *"Apply same reach selection criteria to all selected sources (yes/no)"*. A *"yes"* response applies the same reach selection to all sources using two separate tables (one for sources and one for the reach selection criteria), whereas a *"no"* response allows users to apply different reach selection criteria and source-change percentages to each source in the same table as well as to change the same source in different ways in different areas.
  - Execution of a scenario produces a map that displays the user choice of a relative measure of the change in stream load or an absolute measure of the stream load, yield, or concentration associated with the changed source(s). The relative measure is expressed as the ratio of the changed model prediction of load (with source changes applied) to the baseline (unchanged) model load prediction (or optionally expressed as a percentage). Scenario results are output to a sub-directory with the user-specified name (*"Enter scenario name"* option), located within the upper-level *scenarios* directory.

(4) **Customization settings for map displays:** - A variety of functional settings (*"Mapping Settings"*) are provided (see Fig. 14) to allow users to customize the map and legend background color, binning breakpoints and intervals, symbol colors, numerical rounding, and text and symbol/line sizes. The customized settings for these controls, shown in the previous sections for the non-interactive production of maps, are loaded as the default settings in the R Shiny mapper.

(5) **Generate and save map output:** - The *"Generate Map"* button displays the map according to the user-defined settings. The *"Save Map"* button outputs the map according to the user-defined settings to appropriate sub-directories in the "maps" or "scenarios" directories (Figure 1). The output format is controlled by the *"Output Map Format"* setting, with a PDF file output for the *"static"* option (with non-interactive features) and a HTML file output (with interactive features) for the *"plotly"* and *"leaflet"* options. The *"plotly"* map option is not available for the display of catchment metrics because of the lengthy rendering time. Note that a map does not have to be generated (*"Generate Map"*) prior to saving (*"Save Map"*).

4.4.9.1.2 Enabling the RShiny mapper for an existing model

To enable the R Shiny mapper for a previously executed model within an existing or new RStudio session, users should run the following statements in the RStudio Console window. The `runBatchShiny` control setting will enable the mapper for the results of a prior model specified according to the full pathname that is listed in the function argument, as illustrated below.

```
path_master<-'./RSPARROW_master'
suppressWarnings(remove(list="runRsparrow"))
devtools::load_all(path_master,recompile = FALSE) # read the RSPARROW library into memory

runBatchShiny("C:/UserTutorial/results/Model6")    # enable mapper for tutorial model 6
```

A user can also enable R Shiny in a specified browser for a previously executed model by specifying the browser pathname (`path_ShinyBrowser`) for the second argument of the `runBatchShiny` control setting, as illustrated below.

```
runBatchShiny("C:/UserTutorial/results/Model6",
path_shinyBrowser ="C:/Program Files (x86)/Google/Chrome/Application/chrome.exe" )
```

4.4.9.2 Simulation of source-change management scenarios in the DSS mapper

```
#-------------------------------------------------------------------------------
  # Simulation of source-change management scenarios in the RShiny mapper

  #  NOTE: Requires prior execution of the model estimation ('if_estimate')
  #        and standard predictions ('if_predict').

  #  Scenarios can be executed using the R Shiny interactive mapper using the
  #    control setting 'enable_ShinyApp <-"yes"'
```

The control settings in this section provide a decision-support tool for executing user-defined source-change management scenarios with tabular and mapped output. The tool allows users to evaluate the effects of hypothetical changes (increases or decreases) in source inputs, expressed in units of *mass* (e.g., fertilizer use) or *area* (e.g., land use/cover), on the water-quality loads and concentrations in downstream reaches. For scenarios with area-based land-use sources, users have the option to convert the land area from one land-use type to another, or users can change the land-use source loading per unit area through changes in the mean estimates of the land-use model coefficients (this latter option can only be executed in the R Shiny interactive mapping interface; see description in sub-section 4.4.9.2).

The scenario settings are only applicable to RSPARROW models for which model estimation and predictions have been previously executed (see `if_estimate` and `if_predict` control settings).

Users may execute scenarios in two modes. The first mode uses the control script settings in this section to output load predictions for the the source-change scenarios to tables (CSV files) and maps (PDF, HTML). The second mode allows users to execute scenarios in the R Shiny interactive mapping interface, activated by the `enable_shinyMaps<-"yes"` control setting (see description above), which also supports the output of results to tables and maps.

Use of the R Shiny interactive mapper is recommended because it provides considerable flexibility and is easier to use than the control script settings that are described below. Also, the *"leaflet"* mapping option (with standardized, globally available base maps) is only supported in the R Shiny mapper. Nevertheless, users of the R Shiny mapper should consult the sections below to obtain a general understanding of the methods for identifying watersheds and reaches for scenarios, the limitations of scenarios with land-use sources, and the output produced for scenario results. Users should also consult Chapter 6.2.9 for a scenario tutorial that illustrates use of the R Shiny mapper.

4.4.9.2.1 Identify the locations for applying scenarios

```
#-------------------------------------------------------------------------------
  # Identify the locations for applying scenarios
  #    "none", "all reaches", "selected reaches"
  select_scenarioReachAreas <- "all reaches"
  select_scenarioReachAreas <- "selected reaches"
  select_scenarioReachAreas <- "none"     # do not execute scenarios

  # Indicate the watershed locations where the scenarios will be applied
  #   to either "all reaches" or "selected reaches".
  select_targetReachWatersheds <- NA      # Execute the scenarios for all reaches in the
                                  # modeled spatial domain (i.e., above the user-defined
                                  # terminal reaches)
  select_targetReachWatersheds <- 15531   # Execute for a single watershed inclusive of
                                  # this watershed outlet reach ('waterid' system
                                  # variable) and all upstream reaches
  select_targetReachWatersheds <- c(15531,14899,1332)   # Execute for multiple watersheds
                                  # inclusive of these watershed outlet reaches
                                  # ('waterid' system variable) and all upstreams
                                  # reaches
```

These two settings enable users to identify the watershed locations within the modeled spatial domain where the management scenarios will be applied. The first setting, `select_scenarioReachAreas`, identifies whether the scenario feature is to be disabled ("none") or whether scenarios are to be applied to "all reaches" or "selected reaches" above user-defined watershed outlets. The second setting, `select_targetReachWatersheds`, allows users to define the watershed outlets (i.e., targeted reach locations) and the hydrologically-connected upstream reaches in the watersheds where the source-change scenarios are to be applied.

The `select_scenarioReachAreas` setting controls the execution of several additional settings that specify scenario conditions for the sources (types, magnitude of change) and reach locations, as described below for the "all reaches" and "selected reaches" options.

The `select_targetReachWatersheds` setting provides two options for defining the watershed outlets and their upstream reaches where the scenarios are applied:

- The `NA` option defaults to watershed outlets associated with the user-defined terminal reaches. The terminal reaches are specified by the system variable *target* (see Table 4), as defined in the *data1.csv* file and/or modified in the *userModifyData* script. The terminal reaches are typically defined by users as the most downstream terminus of stream networks (e.g., coastal fall-line or international boundaries, inland lakes, edge of modeled spatial domain). Therefore, the selection of the default terminal reaches will typically allow all or most reaches in the spatial domain to be subject to the user's source-change scenario.
- Alternatively, users may apply the source-change scenarios to a single or multiple watersheds by listing the watershed outlet reach identifier (i.e., value of the *waterid* system variable). RSPARROW automatically defines the watershed reaches, inclusive of the outlet reach and all upstream hydrologically connected reaches.

- As a resource for users, the *flag_targetReachWatersheds.csv* file, located in the "(run_id)_scenarios" directory, can be searched by river name (*rchname*) and drainage area (*demtarea*) to assist in locating *waterid* reach values. The file contains all reaches in the modeled watersheds, and includes additional columns for any variables specified by the user in the `add_vars` setting (the file is created when predictions are executed using the `if_predict` setting).
- The R Shiny interactive mode allows users to optionally identify watershed outlet reaches in the *flag_targetReachWatersheds.csv* file and input these reaches to scenarios. Users enter a "1" in the column named *Flag* either directly from the R Shiny session or prior to enabling an R Shiny session. This method is preferred in cases where users have a large number of outlet reach locations. A copy of the edited file is saved to the user's `scenario_name` sub-directory.

4.4.9.2.2 Set scenario source conditions for "all reaches" in user-defined watersheds

```
#-------------------------------------------------------------------------------
  # Settings applicable to select_scenarioReachAreas<-"all reaches" option.
  # Source changes are applied to "all reaches" in the user-defined watersheds

  # List the source variables evaluated in the change scenarios.
  scenario_sources <- NA
  scenario_sources <- c("point","ndep","crops")

  # For land-use 'scenario_sources' with areal units, specify a land-use source in the
  # model to which a complimentary area conversion is applied that is equal to the
  # change in area of the `scenario_source` variable. Note that the converted source
  # variable must differ from those that appear in 'scenario_sources' setting.
  landuseConversion<-c(NA, NA,"forest") # convert crop area to forested area
  landuseConversion<-NA    # option if no land-use variables appear in
                           # 'scenario_source' setting

  # Source-adjustment factors (increase or decrease) applied to "all reaches"
  #  in the user-specified watersheds. Enter a factor of 0.1 or 1.1 to obtain a 10%
  #  reduction or increase in a source, respectively.
  scenario_factors <- NA
  scenario_factors <- c(0.20,1.1,0.25)   # order consistent with order of
                                         #  the 'scenario_sources'
```

Source-change scenarios applied to all reaches (`select_scenarioReachAreas<-"all reaches"`) in user-specified watersheds require control settings to define the sources and magnitude of change in the sources for the scenario.

The `scenario_sources` setting identifies one or more source variables in the SPARROW model that are to be evaluated in the management source-change scenario. The sources can be expressed in mass units (e.g., fertilizer use) or areal units (e.g., land use/cover). Note that the `scenario_sources` setting is also applicable for the "selected reaches" option described in the next sub-section 4.4.9.3.

The `landuseConversion` setting should specify a `NA` for each of the `scenario_sources` that is expressed in *mass* units, as illustrated above. If all `scenario_sources` are expressed in mass units, then a single `NA` entry can be specified instead.

For `scenario_sources` expressed in *areal* units, the `landuseConversion` setting should specify a land-use source in the SPARROW model to which a complimentary area conversion is applied that is equal to the change in area of the `scenario_sources` variable. For example, a scenario that reduces cropland area might be offset by increases in pasture or forest land area, or a scenario that increases urban land area might be offset by decreases in forested land area. This setting ensures that the hypothetical changes in land-use sources are suitably paired with the equivalent conversion of the area of other modeled land-use sources, such that the affected total drainage area associated with the land-use sources in the SPARROW model remains

the same in the simulation. Note that land-use types listed in the `landuseConversion` setting must exist as sources in the SPARROW model (RSPARROW prints a user warning if this condition is violated, and the change scenario is not executed).

Users should note the following features for scenarios with land-use sources:

- Land-use area conversions using the `landuseConversion` setting are most accurately applied in SPARROW models with land-use sources that represent the full range of natural and cultural land uses observed in the modeled watersheds. By contrast, models with mixed sources (mass and area) may include only one land-use source or a limited number of land uses as sources. This may complicate the ability of users to specify realistic conversions of drainage areas among the available land-use sources in a change scenario; in these cases, users should limit the change scenarios to the mass-based sources.
- Change scenarios are not executed (and a user warning printed to the *Console* window in RStudio) in cases where a land-use source listed in the `landuseConversion` setting is selected as a managed source in the `scenario_sources` setting. The termination of execution in these cases avoids the possibility of cascading conversions of land areas that might lead to errors, such as crop converted to pasture and pasture converted to forest.
- A warning message is printed to the *Console* window in RStudio in cases where the magnitude of the increase in the incremental reach area of a land-use `scenario_source` (e.g., urban lands) is larger than the incremental area associated with a `landuseConversion` source (e.g., forested lands). In these cases, the decrease in area of the `landuseConversion` source (e.g., forest), corresponding with the increase in area of the land-use `scenario_source` (e.g., urban), would lead to erroneous negative values. When these cases arise, the scenario is executed for the user's targeted watersheds; however, the scenario adjustments in area are not applied to the individual reaches where these conditions (i.e., erroneous negative area values) are expected to occur. Additionally, a CSV file is output to the *scenario_name* subdirectory with the contents of the *subdata* object reported for reaches where the scenario is not applied. This allows users to determine the number and location of reaches that are affected by this condition. The CSV file is assigned the name (scenario_name)*_NegativeLanduseFound.csv*.
- *Available only with the R Shiny interactive mapping interface* - For scenarios with land-use sources, users have the option to change the land-use source loading per unit area. This adjustment is applied to the SPARROW model coefficient for a land-use source, which is expressed in units of mass per unit area per time; the scenario requires no change in the area (or area conversion) of land-use sources. This adjustment in the unit-area loading can be used to mimic the effects of management practices that would be expected to change the contaminant loadings from a given land type, rather than alter the land area of the source. For example, this feature might be used to evaluate downstream water-quality effects of reductions in contaminant loadings to reaches associated with the addition of human wastewater treatment in urban areas or the implementation of best management practices on agricultural lands.

The `scenario_factors` setting specifies the factors (i.e., multipliers) by which each source is either reduced or increased according to the user's source-change scenario. For example, a factor of 0.1 or 1.1 would apply a 10% reduction or increase in a source, respectively. The order of the factors is identical to that as the sources listed in the `scenario_sources` control setting. By comparison, in the R Shiny interactive mapper, the user is prompted to enter *Percent Change (+/-) Factors*, which are internally converted to the appropriate `scenario_factors`, such that -10 and 10 imply a 10% reduction or increase, respectively (see the tutorial illustration in Chapter 6.2.9).

Note that users have the choice to assign `NA` to all of the above settings in this section, except for the `scenario_sources` setting, in cases where the `select_scenarioReachAreas<-"selected reaches"` option is executed as described below; otherwise, all settings in the above section, except for the `scenario_sources` setting, will be ignored for the `"selected reaches"` option.

4.4.9.2.3 Set scenario source conditions for "selected reaches" in user-defined watersheds

```
#-------------------------------------------------------------------------------
# Settings applicable to select_scenarioReachAreas<-"selected reaches" option.
```

```
# Source changes applied to "selected reaches" in the user-defined watersheds.

    # (A) Specify the source-adjustment factors in the 'userModifyData.R'
    #      script for each of the "scenario_source" variables.
    #      The variable names for the factors are defined by adding the
    #      prefix "S_" to the *sparrowNames* variable name for each source.
    #      In the example, point sources are reduced by 20% and atmospheric
    #      deposition is increased by 10% in Ohio Basin (huc2=5) and cropland
    #      area is reduced by 25% in the Upper Mississippi Basin:
            S_point <- ifelse(huc2 == 5,0.2,NA)
            S_atmdep <- ifelse(huc2 == 5,1.1,NA)
            S_crops <- ifelse(huc2 == 7,0.25,NA)

    # (B) Specify the land-use types used for land conversion in the 'userModifyData.R'
    #      script, in cases where the scenario sources are land use/cover variables,
    #      expressed in areal units. The variable names for the conversion types are
    #      defined by adding the suffix "_LC" to the *sparrowNames* variable name for
    #      each land-use area source.
    #      In the example, cropland area is converted to pasture area in reaches of
    #      the Upper Mississippi River Basin (huc2=7), and "NA" is assumed for the
    #      land conversion for the mass-based point sources and atmospheric sources:
            S_crops_LC <- ifelse(huc2==7,"pasture",NA)

    # (C) Add the variable names for the source-adjustment factors (e.g., "S_crops")
    #      and the conversion land-use type (e.g., "S_crops_LC") as 'sparrowNames'
    #      in the dataDictionary.csv file, with an OPEN 'varType'.

    #   dataDictionary.csv file:
    #        varType    sparrowNames      data1UserNames        varunits    explanation
    #         OPEN      S_crops                NA
    #         OPEN      S_crops_LC             NA
```

Source-change scenarios applied to selected reaches (`select_scenarioReachAreas<-"selected reaches"`) in user-specified watersheds require users to add conditional R statements to the *userModifyData.R* script to define the sources, magnitude of change in the sources, and the reach locations for applying the scenario.

Users are required to define new variables for the source-change factors and land-use conversion variables in the *dataDictionary.csv* file (with OPEN *varType*) and the *userModifyData.R* script. The variable names for the change factors are defined by adding the prefix "S_" to the *sparrowNames* variable name for each source. In cases where the scenario sources are land use/cover variables, expressed in areal units, the variable names for the conversion types are defined by adding the suffix "_LC" to the *sparrowNames* variable name for each land-use area source. The required syntax for the R statements in the *userModifyData.R* script is illustrated in the example above.

For the `select_scenarioReachAreas<-"selected reaches"` setting, the `landuseConversion` control setting is not required, but can be used in combination with the `scenario_sources` setting as a default condition if no "_LC" variables are defined in the *userModifyData.R* script.

The example R statement above illustrate a scenario with the Midwest SPARROW tutorial model (Chapter 6) in which a 20% reduction (factor=0.2) is applied to municipal/industrial point sources and a 10% increase is applied to atmospheric deposition all reaches in the Ohio River basin (i.e., huc2=5), and a 25% reduction in area is applied to the cropland source in the Upper Mississippi River basin (huc2=7). A factor of 1 (i.e., no change) is applied to all other reaches outside of the Ohio and Upper Mississippi River basins. Reductions in the area of the cropland source is offset by increases in pasture land by an equivalent area.

Users are free to devise more elaborate R statements to create scenarios that include a more diverse range of

sources, factors, and land conversion types. These could, for example, allow different source-change factors to be applied across different reaches. Alternatively, execution of source-change scenarios with the R Shiny option provides considerable flexibility and ease of use.

4.4.9.2.4 Specify the scenario output settings

```
#-------------------------------------------------------------------------------
  #Set scenario name; this becomes the directory and file name for all scenario output
  #  NOTE: only one scenario can be run at a time; avoid using "/" or "\" for name
  scenario_name<-"scenario1"

  # specify the colors for six classes for mapped predictions
  scenarioMapColors<-c("light blue","blue","dark green","gold","red","dark red")
```

The `scenario_name` setting allows users to specify a name for each evaluated scenario. This creates a separate parallel sub-directory in the "(run_id)/scenarios" directory for each named scenario (see Fig. 1 in Chapter 1).

The setting `scenarioMapColors` allows users to specify the colors for six classes of mapped predictions; the first class argument (e.g., "light blue") will be used to display reaches where no change occurs in the load because the reaches are located in watersheds that are unaffected by the user's selected targeted reaches, as specifed by the `select_targetReachWatersheds` setting.

```
#-------------------------------------------------------------------------------
  # Identify prediction variables to display a stream map of the effects of the
  #  scenario on water-quality loads. Options include:
  #
  # RELATIVE METRICS:
  # Ratio of the changed load (resulting from the scenario) to the baseline load
  # associated with the original (unchanged) mass or area of the model sources.
  #  Metric names and explanations:
  #   ratio_total                Ratio for the total load (a measure of the watershed-
  #                               scale effect of the change scenario)
  #   ratio_inc                  Ratio for the total incremental load delivered to
  #                               the reach (a measure of the "local" effect of the
  #                               change scenario)

  # ABSOLUTE METRICS:
  # Load prediction names and explanations
  #   pload_total                Total load (fully decayed)
  #   pload_(sources)            Total source load (fully decayed)
  #   pload_nd_total             Total load delivered to streams (no stream decay)
  #   pload_nd_(sources)         Total source load delivered to streams (no stream decay)
  #   pload_inc                  Total incremental load delivered to reach
  #                                 (with 1/2 of reach decay)
  #   pload_inc_(sources)        Source incremental load delivered to reach
  #                                 (with 1/2 of reach decay)
  #   pload_inc_deliv            Total incremental load delivered to terminal reach
  #   pload_inc_(sources)_deliv  Total incremental source load delivered to terminal reach
  #   share_total_(sources)      Source shares for total load (percent)
  #   share_inc_(sources)        Source shares for incremental load (percent)

  # Yield prediction names and explanations
  #   Concentration              Flow-weighted concentration based on decayed total load
  #                                 and mean discharge
  #   yield_total                Total yield (fully decayed)
  #   yield_(sources)            Total source yield (fully decayed)
```

```
#   yield_inc                   Total incremental yield delivered to reach
#                                  (with 1/2 of reach decay)
#   yield_inc_(sources)         Total incremental source yield delivered to reach
#                                  (with 1/2 of reach decay)
#   yield_inc_deliv             Total incremental yield delivered to terminal reach
#   yield_inc_(sources)_deliv   Total incremental source yield delivered to
#                                  terminal reach
#
scenario_map_list <- c("ratio_total","ratio_inc","pload_total","concentration")
```

The `scenario_map_list` setting allows users to select the type of predictions to display for streams, catchments, or both according to the `output_map_type` setting described in Chapter sub-section 4.4.8.2. The maps are output to either a PDF or HTML file, separately for each prediction type, according to the `enable_plotlyMaps` setting as described in Chapter subsection 4.4.8.4, which controls the generation of maps with interactive (HTML) or non-interactive features (PDF):

- *Relative metrics* - These metrics map the ratio of the changed load (resulting from the scenario) to the baseline load associated with the original (unchanged) mass or area of the model source, thereby providing a relative measure of the magnitude of the effect (change in load) of the user-specified management scenario. Users can select to display the ratio for two metrics:
  - Total load ratio (`ratio_total`) - This provides a relative measure of the watershed-scale effect of the scenario on the total load, inclusive of the mass contributed from all upstream sources and reaches. For example, this metric is relevant to measuring the effect on stream concentrations that are influenced by the total streamflow and contaminant sources contributed from all upstream catchments. The metric is also useful to evaluate the potentially distant downstream effect of management scenarios on reaches and waterbodies that fall outside of the catchments where the management scenarios are applied.
  - Incremental load ratio (`ratio_inc`) - This provides a relative measure of the "local" effect of the scenario on the load delivered to an individual reach from all contaminant sources in the incremental drainage area of that reach. This metric isolates the effect of the change scenario on an individual reach from the effects of contaminant sources and the change scenario in the adjacent and upstream reaches.
- *Absolute metrics* - Users may select these metrics by specifying the name of one or more of the standard unconditioned predictions as listed above (also see section 7 of the control script for a listing). The predictions are expressed in mass units and will, therefore, reflect the absolute effect of the change scenario on the model predictions of load and yield. The available metrics include the mean load and yield for total and incremental drainage areas and for the individual modeled sources; these predictions are corrected for log-retransformation bias as is done for the standard predictions (without changed sources). Note that the conditioned (monitoring-adjusted) predictions are excluded as options because the scenarios can only be evaluated using simulated (i.e., unconditioned) model predictions that preserve a spatial mass balance in the loads. The model prediction of the delivery fraction is unaffected by a change scenario and is not reported.

Tabular output is automatically output to CSV files for loads and yields (also concentration) for all of the standard metrics (e.g., "pload_total", etc., including loads and yields for each source) for the unconditioned model predictions. These include a separate set of CSV files for loads and yields (also concentration) that express the changed metric (resulting from the scenario) as a fraction of the baseline metric associated with the unchanged source input. See above in this section for a listing of output metrics and Chapter 5.5 for details on the CSV and PDF output files (also see Table 6 in sub-section 4.3 for a listing of the output file names).

Note that the downstream effects of a user management scenario are automatically simulated for all hydrologically-connnected reaches located downstream of the user's targeted watersheds, with the most downstream reach location defined by the terminal reaches (e.g., coastal fall-line or international boundaries,

inland lakes). The terminal reaches are specified by the user in the system variable *target* (see Table 4); this variable is defined in the *data1.csv* file and/or modified in the *userModifyData* script. For example, for the Midwest SPARROW tutorial model (Chapter 6), the *target* reaches are set equal to the terminal reaches, defined as reaches with *termflag* values of 1 (stream reach) or 3 (coastal shoreline reach). Therefore, the scenario prediction results (e.g., "ratio", "pload_total") output to the stream map (and CSV files) will include results for the effects of the scenario on all reaches downstream of the watershed reaches that are specifed by the `select_targetReachWatersheds` setting. In cases where a user applies the management scenarios to only selected watersheds within the modeled spatial domain, the simulation of changed predictions to the most downstream reaches allows uses to evaluate and visualize the potentially distant downstream effects of source-change scenarios on loads and concentrations.

4.4.10 Model prediction uncertainties (section 10 of control script)

4.4.10.1 Background

The settings in this section enable the bootstrap calculation of bias-retransformation corrected mean estimates and uncertainties of the model predictions for individual river reaches. The methods are intended for use with the final specifications of RSPARROW models.

Bias correction of the model predictions of mean load is necessary to adjust for the effects of log-retransformation bias that is present in exponentiated SPARROW predictions of the log-transformed load (logged values of load are used as the response variable in SPARROW models). The bias correction is achieved by multiplying the SPARROW exponentiated predictions of log load by a bias-retransformation correction factor (also referenced as the "bias-correction" factor in this section), computed as the mean of the exponentiated leverage-weighted log residuals from the model, according to equation 1.124 in Schwarz et al. (2006) using the "Smearing Estimator" method (Duan, 1983).

RSPARROW uses a parametric bootstrapping method to account for two sources of uncertainties in model predictions: *sampling error* and *model error*. The *sampling error* is associated with uncertainties in the estimation of the model coefficients from a finite sample, whereas the *model error* is associated with limitations in the ability of the model to account for all of the processes that control spatial variability in the observed water-quality loads. The sampling error would be expected to approach zero with increasingly large sample sizes, whereas reductions in model error would be associated with increased model complexity that improves prediction accuracy. The parametric bootstrapping method is used to obtain random samples of the model coefficient distributions and model errors (exponentiated leverage-weighted log residuals). The method employs a Monte Carlo resampling of the assumed standard normal distribution of the model parameters from which a sampling distribution of the model errors is obtained.

Measurement errors in explanatory variables and the response variable are additional sources of uncertainties in model coefficients and model predictions. These errors cannot be removed by adding more observations or explanatory variables to a SPARROW model. Measurement errors in explanatory variables can cause biased coefficient estimates but may not necessarily cause biased model predictions (Schwarz et al., 2006). Measurement errors in the response variable are potentially caused by errors in estimating water-quality loads, associated with field sampling and laboratory analytical methods as well as uncertainties intrinsic to statistical estimation of mean annual loads (Lee et al., 2016). Uncertainties related to these measurement errors are a latent component of the SPARROW model residuals and prediction uncertainties. Methods are not currently available in SAS SPARROW or RSPARROW to separately account for the effects of measurement errors in the response variables, when estimating model coefficients or prediction uncertainties. Hierarchical Bayesian methods exist that explicitly account for monitored load uncertainties as part of the SPARROW model estimation and prediction of reach-level loads and uncertainties; these methods have been previously applied to SPARROW (e.g., Qian et al., 2005; Wellen et al., 2012; Alexander, 2015; Alexander et al., 2019b). A hierarchical Bayesian version of RSPARROW is currently under development that will account for measurement-related errors in loads.

The parametric bootstrapping method executed in RSPARROW is one of the two bootstrapping methods available in SAS SPARROW. The parametric method provides sufficiently robust estimates of the coefficient uncertainties with fewer computational demands than that of the full bootstrap resampling method in SAS

SPARROW (the full method re-estimates the model using repeated resampling of the observations with replacement; this method is not currently available in RSPARROW). Schwarz et al. (2006) report that the parametric Monte Carlo estimates of the coefficient uncertainties, with an assumed standard normal distribution, are generally of the same order of accuracy as the full bootstrap estimates (however, note that the full bootstrap methods can be useful to assess small sample bias in the NLLS estimated coefficients). USGS SPARROW modelers have also commonly employed the parametric bootstrapping methods (e.g., Preston et al., 2011) as provided in RSPARROW.

Note that the standard normal samples of the model coefficients, generated by the Monte Carlo resampling as part of the parametric bootrapping method in RSPARROW, are not intended for use in hypothesis testing and reporting of the statistical significance of the coefficients of estimated models (Schwarz et al., 2006). The estimated NLLS coefficients and their associated standard errors and t-statistics (generated by settings in section 4 of the control script) are preferred for this purpose. The parametric coefficient estimates and their statistical significance are sufficient to make statistically informed decisions about the final model specifications (i.e., selection of the final set of explanatory variables and functional forms); this is justified based on the statistical robustness of the model estimation under assumed asymptotic behavior in large samples (Schwarz et al., 2006).

4.4.10.2 Control settings for iterations, confidence itervals, seed values

```
#-------------------------------------------------------------------------------
  # Number of parametric bootstrap iterations
  biters <- 200
```

This setting specifies the number of Monte Carlo samples for estimation and/or prediction. Based on prior USGS experiences with SPARROW modeling, a sample of 200 is recommended and is typically sufficient to obtain robust estimates of the model and prediction uncertainties.

```
#-------------------------------------------------------------------------------
  # Confidence interval setting
  confInterval <- 0.90
```

This setting specifies the two-sided confidence interval for calculating the lower and upper bounds for the prediction uncertainties. For example, at setting of `confInterval<-0.90` will calculate the bounds such that five percent of the probability density is contained in each of the tails of the distribution of prediction metrics.

```
#-------------------------------------------------------------------------------
  # Specify the initial seed for the boot_estimate, boot_predict,
  #   and if_validate<-"yes" and pvalidate>0
  iseed <- 139933493
```

This setting provides a user-selected initial integer seed value to populate the argument of the R random number generator function (`set.seed`). Use of the identical *iseed* value in subsequent execution of the control script (with the same settings for other control variables) will reproduce identical random number selections and associated model outcomes.

Three RSPARROW control settings use the *iseed* value to initiate the following random selection methods:

- `if_boot_estimate <- "yes"` - Enables the random selection of standard normal deviates (R function `rnorm`) in a Monte Carlo parametric sampling that generates a feasible set of model outcomes (i.e., coefficient distributions and their associated model errors and bias-retransformation correction factors) to support estimation of the bias-corrected mean load predictions and their uncertainties (standard errors, confidence intervals).
- `if_boot_predict <- "yes"` - Enables the random selection of the model errors (i.e., exponentiated leverage-weighted log residuals) in the bootstrapping prediction function to support estimation of the confidence intervals for the bias-corrected mean load. This employs the R random sample function `sample`, using the bootstrapping sampling option with the sample replacement option (`replace==TRUE`).

- `if_validate <- "yes"` and `pvalidate` > 0 - Enables the random selection of calibration and validation sites, using the R random sample function `sample`, with the no sample replacement option (`replace==FALSE`). See sub-section 4.4.6 for details on the selection of validation sites.

4.4.10.3 Coefficient and model uncertainties

```
#-------------------------------------------------------------------------------
  # Specify if parametric bootstrap estimation (Monte Carlo) is to be executed
  if_boot_estimate <- "yes"
```

This setting calculates and outputs the *biters* Monte Carlo samples of the model coefficients and the corresponding bias-retransformation correction factors (column labeled *bootmean_exp_weighted_error* in the output files) associated with the model residuals for each iteration of the sampled model coefficients with an assumed standard normal distribution. The bias-retransformation correction factors are derived using Smearing estimate methods (Duan, 1983). The results are output to the *(run_id)_bootbetaest.csv* file and the *(run_id)_BootBetaest* R binary file. Selected metrics are also stored in the *BootResults* object internally in the RStudio session.

RSPARROW Monte Carlo parametric methods assume a standard normal distribution for each of the model coefficients. The methods account for the variance-covariance of the model coefficient estimates according to methods described in Schwarz et al. (2006; equation 1.86 in section 1.5.3). Random samples of the model coefficients are obtained by multiplying the product of the randomly selected standard normal deviates and the square root of the variance-covariance matrix (based on Hessian methods) by the NLLS mean estimates of the model coefficients. The randomly generated coefficient values are adjusted, if necessary, to ensure consistency with the user-specified parameter constraints (lower and upper bounds in the *parameters.csv* file), which are reflected in the NLLS estimates of the coefficient covariance matrix. Thus, lower and upper bounds are substituted for coefficient values that are less than or greater than the user-specified lower or upper bounds, respectively.

4.4.10.4 Model predictions: Bias-corrected means and uncertainties

```
#-------------------------------------------------------------------------------
  # Specify if bootstrap predictions (mean, SE, confidence intervals) are to be executed
  #  Note: Bias retransformation correction based on parametric bootstrap estimation
  #        Requires completion of bootstrap estimation
  if_boot_predict <- "yes"
```

This setting calculates and outputs reach-level bias-corrected mean predictions and their associated uncertainties (standard error, confidence intervals) for the mass-related metrics (e.g., total load, incremental load), including the source-share related mass metrics. No bias correction is applied to the delivery factor (Schwarz et al., 2006). The prediction metrics are output to the *(run_id)_predicts_load_boots.csv* and *(run_id)_predicts_yield_boots.csv* files. Prediction metrics are also stored in the *predictBoots.list* object internally in the RStudio session, and selected uncertainties are also output to the binary file *(run_id)_BootUncertainties*.

Execution of the `if_boot_predict` setting requires the execution of the bootstrap model parameters (`if_boot_estimate <- "yes"`) during the current RStudio session or during a prior session.

4.4.10.4.1 Unconditioned predictions

The following parametric bootstrap methods are used to calculate the bias-corrected mean and uncertainties for the unconditioned (simulated) model predictions of loads and yields for reaches:

- Bias-corrected mean - The bootstrap proportional bias-transformation corrected mean is computed as the ratio of the squared NLLS mean prediction (inclusive of the mean exponentiated error correction) and the average of the *biters* mean bootstrap predictions (inclusive of the mean exponentiated error correction for each of the *biters* iterations), as described in equation 1.138 (Schwarz et al., 2006).

Note that the estimates of the bootstrap proportional mean are bias corrected individually for each mass-based prediction metric and reach location, without any restrictions on the network-wide mass balance. Thus, the mean estimates do not retain a spatial mass balance within the river network.

  - For analyses of model predictions that require mass balance restrictions, the standard or non-bootstrap estimates for the unconditioned predictions should be used (Chapter sub-section 4.4.7), based on use of the control setting `if_predict` as described in section 7 of the control script (output is directed to the *(run_id)_predicts_load.csv* and *(run_id)_predicts_yield.csv* files). The non-bootstrap unconditioned predictions of load and yield receive a single, network-wide estimate of the bias-correction factor (mean exponentiated model error), thereby conforming with the mass balance constraints.

- Standard error of the mean - This is computed as the product of the bootstrap proportional bias-corrected mean and the square root of the sum of the model error variance term and sample error variance term, according to equation 1.144 (Schwarz et al., 2006). The model error variance is computed as the ratio of the variance of the exponentiated NLLS errors to the square of the bootstrap mean exponentiated error correction. The sample error variance is computed as the variance of the bootstrap predictions (inclusive of the mean exponentiated error for each of the *biters* iterations) to the squared NLLS mean prediction (inclusive of the mean NLLS exponentiated error).
- Confidence intervals - The intervals are derived from the distribution of *biters* mean predictions that are computed as the ratio of the squared NLLS mean predictions (with mean error correction removed) to the bootstrap predictions from the *biters* sampled model coefficients (inclusive of a randomly selected reach-level value of the *biters* NLLS exponentiated errors), as specified by equation 1.150 (Schwarz et al., 2006). For example, the 90 percent intervals are computed as the 5th and 95th quantiles of the distribution of the mean predictions.

4.4.10.4.2 Conditioned predictions

SPARROW conditional loads reflect the substitution of the observed (i.e., monitored) station load for the model simulated load on reaches with monitoring stations, as described in sub-section 4.4.7. The methods for calculating the uncertainties for conditioned (monitoring-adjusted) model predictions of loads and yields are consistent with those described in Schwarz et al. (2006) for unconditioned predictions, but the methods additionally assume that the model error only applies to the proportion of the load that is simulated by the model. The proportion of the load that is contributed from upstream monitoring stations is assumed to be estimated without error.

The methods for conditional load uncertainties employ similar equations as those described for the unconditional loads (Schwarz et al., 2006), but additional adjustments are made to the standard error and confidence interval equations to account for the proportion of the reach load contributed by upstream monitored loads. These proportions are determined for each reach in the river network and used to appropriately quantify the proportion of mass that receives the randomly selected model error. These methods are documented in the SAS SPARROW code and supplementary information from G.E. Schwarz (USGS, written communication, 2018).

In RSPARROW, the conditional methods are used to calculate the prediction variables for load (*mean_mpload_total*, *mean_mpload_(sources)*) and yield (*mean_myield_total*, *mean_myield_(sources)*) as described in sub-section 5.2.2.1.

Note that the uncertainties of the conditional loads are likely to be underestimated for reaches with upstream contributions from monitored loads. This is because the monitored loads are assumed to have zero error, whereas statistical errors are known to be associated with the estimation of the monitored loads, based on rating curve and other statistical methods (e.g., Schwarz et al., 2006; Robertson and Saad, 2011; Lee et al., 2016). Methods are not currently available in SAS SPARROW or RSPARROW to separately account for the effects of these errors when estimating model coefficients or prediction uncertainties. As noted previously, hierarchical Bayesian methods exist that explicitly account for monitored load uncertainties as part of the SPARROW model estimation and prediction of reach-level loads and uncertainties; these techniques have

been previously applied to SPARROW (e.g., Qian et al., 2005; Wellen et al., 2012; Alexander, 2015; Alexander et al., 2019b). A hierarchical Bayesian version of RSPARROW is currently under development that will account for measurement-related errors in loads.

4.4.10.4.3 Output files for the bias-corrected means and uncertainties

The model predictions described below are output for loads and yields, respectively, in the *(run_id)_predicts_load_boots.csv* and *(run_id)_predicts_load_boots.csv* files (see sub-section 5.2.2 for additional details). Unless specifically identified as a conditional load or yield with the naming labels *mpload* or *myield*, respectively, all load and yield predictions are computed using the unconditioned loads. These names are used for the predictions available for mapping, including the `master_map_list` control setting in sub-section 4.4.8 and the R Shiny interactive mapper.

The mean load predictions include the variables and naming syntax as shown below. The uncertainties are reported for each variable according to the standard error and the lower and upper bounds for the user-specified percent confidence intervals, using the prefixes of `se_`, `ci_lo_`, and `ci_hi_`, respectively.

```
# Bootstrap load prediction names and explanations
#   mean_pload_total                Bias-adjusted total load (fully decayed)
#   mean_pload_(sources)            Bias-adjusted total source load (fully decayed)
#   mean_mpload_total               Bias-adjusted conditional (monitoring-adjusted)
#                                      total load (fully decayed)
#   mean_mpload_(sources)           Bias-adjusted conditional (monitoring-adjusted)
#                                      toal source load (fully decayed)
#   mean_pload_nd_total             Bias-adjusted total load delivered to streams
#                                      (no stream decay)
#   mean_pload_nd_(sources)         Bias-adjusted total source load delivered to streams
#                                      (no stream decay)
#   mean_pload_inc                  Bias-adjusted total incremental load delivered
#                                      to reach (with 1/2 of reach decay)
#   mean_pload_inc_(sources)        Bias-adjusted total source incremental load
#                                      delivered to reach (with 1/2 of reach decay)
#   mean_deliv_frac                 Fraction of total load delivered to terminal reach
#   mean_pload_inc_deliv            Bias-adjusted total incremental load delivered to
#                                      terminal reach
#   mean_pload_inc_(sources)_deliv  Bias-adjusted total incremental source load
#                                      delivered to terminal reach
#   mean_share_total_(sources)      Bias-adjusted percent source shares for total load
#   mean_share_inc_(sources)        Bias-adjusted percent source shares for
#                                      incremental load
```

The mean yield predictions include the variables and naming syntax as shown below. The uncertainties are reported for each variable according to the standard error and the lower and upper bounds for the user-specified percent confidence intervals, using the prefixes of `se_`, `ci_lo_`, and `ci_hi_`, respectively.

```
# Bootstrap yield prediction names and explanations
#   mean_conc_total                 Bias-adjusted concentration based on decayed total
#                                      load and mean discharge
#   mean_yield_total                Bias-adjusted total yield (fully decayed)
#   mean_yield_(sources)            Bias-adjusted total source yield (fully decayed)
#   mean_myield_total               Bias-adjusted conditional (monitoring-adjusted)
#                                      total lyield (fully decayed)
#   mean_myield_(sources)           Bias-adjusted conditional (monitoring-adjusted)
#                                      total source yield (fully decayed)
#   mean_yield_inc                  Bias-adjusted total incremental yield delivered to
#                                      reach (with 1/2 of reach decay)
```

```
#   mean_yield_inc_(sources)        Bias-adjusted total incremental source yield
#                                      delivered to reach (with 1/2 of reach decay)
#   mean_yield_inc_deliv            Bias-adjusted total incremental yield delivered to
#                                      terminal reach
#   mean_yield_inc_(sources)_deliv  Bias-adjusted total incremental source yield
#                                      delivered to terminal reach
```

4.4.11 Directory and model identification and control script operations (section 11 of the control script)

4.4.11.1 Set path and directory names

```
#------------------------------------------------------------------------------
  path_master <- "~/RSPARROW_master"

  #results, data, and gis directories should be in Users Directory
  #  results_directoryName<-"results_1234"
  results_directoryName<-"results"
  data_directoryName<-"data"
  gis_directoryName<-"gis"
```

Users should specify the full path name for the Master directory, using forward slashes.

Users should specify names for the three required sub-directories in the User Directory; the names shown above are examples and are not required. The subdirectories must be located parallel to one another (see Figure 1). Multiple "results" directories and/or other non-RSPARROW directories can also exist parallel to the required three directories.

Executing the *sparrow_control.R* script in RStudio automatically determines the location (path name) of the User Directory, which is internally stored in R, and will create the sub-directories for use in saving file output.

4.4.11.2 Set model identification number

```
#------------------------------------------------------------------------------
  #current run_id for the model
  run_id<-"TN_rev1"
```

The name of the current model (*"run_id"*) will be the name of the directory (see Figure 1) created to store model results and archived copies of control script and control input files.

4.4.11.3 Load previous model settings into active control script

```
#------------------------------------------------------------------------------
  # Load previous model settings into active control script
  #  Specify the name (run_id) of a previously executed model to copy the control files
  #    into the results directory for editing and execution
  #  Set copy_PriorModelFiles<-NA to run control files currently located in results directory
  #    copy_PriorModelFiles<-run_id
  copy_PriorModelFiles<-NA
```

The `copy_PriorModelFiles` setting allows users to load a previous model's control settings into the RStudio active control script to perform additional model applications, such as predictions, mapping, or the evaluation of management scenarios.

The control setting also provides an efficient way for users to explore new model specifications, using control script settings and/or control input file settings (e.g., *parameters.csv*) from a previously executed model as a starting point; users have the choice to then change only selected settings in these files.

Execution of the control script with a "run_id" specified for this setting will do the following:

- Overwrite the active control script (and all input control files) in the "User Directory/results" sub-directory with the prior versions of these files. Before the files are overwritten, the user will be asked to close all control files in the results directory, including the sparrow_control.R script (but not RStudio). This will allow these files to be overwritten. Failing to close the *sparrow_control.R* file will cause an error in the `copy_PriorModelFiles` functionality, which could result in a mixture of old and new control files in the results directory. Note that the user's pathname specified for the `path_master` setting in the active control script will be retained and is not overwritten.
- The control script from the specified prior model will be opened in the RStudio session. The control script and control input files will be ready for execution or further editing.

In the subsequent execution of the new active control script settings (with `copy_PriorModelFiles<-NA`), users are advised to be careful in selecting an option for the `if_estimate` and `if_estimate_simulation` settings (section 4 of the control script):

- Users should set `if_estimate<-"no"` (and `if_estimate_simulation<-"no"`) to use prior estimated (or simulated) model results to generate new predictions, maps, and management scenarios, or to enable the R Shiny interactive mapper. Thus, no model re-estimation is necessary in this case.
- Users should set `if_estimate<-"yes"` to re-estimate a prior model (or `if_estimate_simulation<-"yes"` to re-simulate a prior model). Note that this will automatically delete all files in the *"estimate"*, *"maps"*, *"predict"*, and *"scenarios"* directories.

4.4.11.4 Model comparison summary

```
#------------------------------------------------------------------------------
  # Run model comparison summary
  #  Select models to compare, use ONLY previous run_ids found in the active
  #    results directory; select NA for no comparision
  #compare_models<-c("TN_rev1","TN_rev2","TN_rev3")
  compare_models<-NA

  #Specify model comparison name, subdirectory name for the comparision results
  modelComparison_name<-"Compare1"
  modelComparison_name<-NA
```

These settings allow the results of the current executed model from the (run_id)*_summary.txt* file (saved in the "estimate/summaryCSV" directory) to be compared with the results of any prior models that are included as a list in the `compare_models` setting; multiple prior models can be selected. The comparison output files are assigned the sub-directory name given by the setting `modelComparison_name`, which is located under the "/User Directory/results" directory. Several output files are generated as described in Chapter sub-section 5.7.

4.4.11.5 File editing and batch execution options

```
#------------------------------------------------------------------------------
  # Option to open CSV control files from R-session for editing
  edit_Parameters<-"yes"
  edit_DesignMatrix<-"no"
  edit_dataDictionary<-"no"

  batch_mode<-"no"
```

These settings allow the option for the dataDictionary and parameter control input files to pop-up for viewing and/or editing once the control script is executed. All control files should be saved before the user indicates that execution of the control script should proceed in the RStudio session. The files will pop up in the users default program for reading CSV files, such as Excel.

A batch-mode operation of the control script is also available to users. This will enable a shell command that will execute all of the model operations in Rscript.exe. Rstudio and the Rscript.exe windows must both remain open until execution is complete. Multiple batch models can be run simultaneously.

4.4.11.6 Customized error handling options

```
#------------------------------------------------------------------------------
  #Enable RSPARROW error handling
  #If error occurs, type options(backupOptions) in the console to restore user settings
  RSPARROW_errorOption<-"yes"
```

The RSPARROW system contains customized error handling both for R system errors (i.e. code syntax errors) and for SPARROW specific errors that are unique to the SPARROW model being executed. SPARROW specific errors will always be captured and a custom message displayed in the console or saved in the batchSessionInfo>(run_id)_log.txt file.

R system errors can be customized by RSPARROW using the control file setting `RSPARROW_errorOption<-"yes"`. When errors are detected, this control setting option allows RSPARROW to output the full `traceback(2)` to the console automatically with any syntax error running in normal mode and will save the full `traceback(2)` to an error.log.txt file in the upper level of model subdirectory when running in `batch_mode`. Line numbers where the error occurs are also output, but it should noted that the line numbers start with `from #1` being the function call (i.e. `estimateFeval<-function(beta0,` is line 1) and do not take into account lines above the function call–i.e., the code meta data.

RSPARROW error handling is applied by customizing the user's `options()` for `error`, `show.error.locations`, and `keep.source`, while saving user's original `options()` and applying them when the model execution is complete. Without applying `RSPARROW_errorOption<-"yes"`, the user can still view the full traceback by typing `traceback(2)` in the console, but errors that occur in `batch_mode` will not be saved or output. Therefore, it is recommended that when running in `batch_mode` the `RSPARROW_errorOption` should be always be set equal to `yes`. It is important to note that the RSPARROW custom `options()` will apply until they are reset, even in a new R session, and if a syntax error is found, the user's options cannot be automatically reset. In the event of a syntax error, the user must manually reset their settings using the saved backup by typing `options(backupOptions)` in the console window.

The custom `options()` settings are shown below.

```
if (RSPARROW_errorOption=="yes"){

  #save user's current options
  backupOptions<-list(error = options()$error,
                      show.error.locations = options()$show.error.locations,
                      keep.source = options()$keep.source)

  #set custom RSPARROW options
  if (batch_mode=="no"){
    options(error=quote({
      cat('\nTraceback:\n');
      # Print full traceback of function calls.
      #The '2' omits the outermost two function calls in the traceback.
      traceback(2);
      #print custom error message
      message("\nRSPARROW SYSTEM ERROR OCCURRED");
      #instruct the user to reset their options
      message('To reset user options in R use options(backupOptions)')}),
      #show line numbers in traceback
      show.error.locations = TRUE,keep.source = TRUE)
```

```
  }else{#batch_mode=="yes"
    options(error = quote({
      #print custom message to console
      message("\nRSPARROW SYSTEM ERROR OCCURRED");
      #instruct the user to reset their options
      message('To reset user options in R use options(backupOptions)');
      #First dump error stack to file; not accessible by the R session.
      dump.frames("errorDump", to.file=TRUE, include.GlobalEnv=TRUE);
      #sink to file
      sink(file=paste0(path_results,"error.log"));
      #print custom error message to file
      cat("RSPARROW SYSTEM ERROR OCCURRED\n");
      #instruct the user to reset their options
      cat('To reset user options in R use options(backupOptions)\n \n');
      #Dump again to get error message and write it to error log;
      #accessible by the R session.
      dump.frames();
      #Print simple error message to file
      cat(attr(last.dump,"error.message"));
      cat('\nTraceback:');
      cat('\n'); #line space
      # Print full traceback of function calls.
      #The '2' omits the outermost two function calls in the traceback.
      traceback(2);
      shell.exec(paste0(path_results,"error.log"));
      sink() #end sink
    }),
    #show line numbers in traceback (shown as 'from #4')
    #line numbers count from the function call (i.e. `nestedFunc<-function(){` is line 1)
    show.error.locations = TRUE,keep.source = TRUE)
  }
}
```

4.4.12 Installation and updating of the R libraries (section 12 of control script)

```
#-------------------------------------------------------------------------------
  # Install required packages
  #   This is a one time process unless a new version of R is installed or more recent
  #   packages are found on Cran. Packages previously installed by user will be skipped.
  if(!"devtools" %in% installed.packages()){install.packages("devtools")}
  suppressWarnings(devtools::install_deps(path_master, upgrade = "ask", type="binary"))

  # Load RSPARROW functions (These 2 lines should ALWAYS be run together)
  suppressWarnings(remove(list="runRsparrow"))
  devtools::load_all(path_master,recompile = FALSE)
```

Execution of the RSPARROW control script (`devtools::install_deps` function) automatically installs the most recent versions of the required R library packages and functions in the user's default R library as specified in RStudio. See Chapter sub-section 1.3 for details on RSPARROW library dependencies and their installation.

RSPARROW 1.1.0 was tested for compatibility with versions of the required R library functions available at the time of public release that are listed in the DESCRIPTION file. RSPARROW may operate with more current versions of the libraries, but full compatibility of RSPARROW with newer versions is not guaranteed.

When opening and executing the RSPARROW control script in RStudio, users should be attentive to messages and prompts associated with the R package installations. See Chapter sub-section 1.3.5 for details on messages and prompts associated with the installation and updating of the RSPARROW library dependencies.

4.5 Guide for executing SAS SPARROW models in RSPARROW

A SAS SPARROW model is most easily executed in RSPARROW by exporting the SAS *INDATA* file from a SAS SPARROW session and saving it as the *data1.csv* file (or other user-specified name in CSV format) that is imported to RSPARROW. The *INDATA* file is the internal SAS file used for SPARROW model calibration and predictions. *INDATA* includes the user's data modifications to stream network variables and model explanatory variables. Importing the *INDATA* file to R avoids having to recode SAS data modifications statements into R statements in the RSPARROW *userModifyData.R* script, a task that is subject to many errors, especially if the SAS statements are numerous and complex or the user is new to R.

The user also has the option to code the SAS data modification statements into R by adding R statements to the *userModifyData.R* script. Considerable care will need to be exercised to ensure that the R statements give results that are identical to the results of SAS statements.

The following steps are necessary to ouput the SAS *INDATA* file and use this file as input to execute the SAS SPARROW model in RSPARROW.

1. Execute the SAS SPARROW control program with the following control settings:

- Set *%let if_mean_adjust_delivery_vars* = no [note that this allows the option of applying the mean adjustment in R]
- Set *%let if_estimate* = no [execution of the model is unnecessary]
- Set *%let if_predict* = no [model predictions are unnecessary]
- Set *%let optional_reach_information* to any SAS variables that are not specified in the model or listed as SAS control variables (e.g., *%let tot_area*) that may be of use in R as diagnostic variables or for other user reference, such as station attributes, physiographic variables (e.g., HUC-2, state), or land use. SAS SPARROW will add these to the *INDATA* file.
- Define any land-use variables in areal units in the data modifications section, and add these variables to the list in the *%let optional_reach_information* statement. Note that areal units (e.g., square kilometers) are required for the land use variables for certain RSPARROW settings.
- Ensure that missing data are set to zero in SAS; alternatively, add R statements in the RSPARROW *userModifyData.R* script using the RSPARROW function `replaceNAs(named.list(land use variable names separated by a comma))`. These functions (*named.list* and *replaceNAs*) are the only ones that the user should execute aside from the execution of the RSPARROW control script.
- From the SAS session, export the *INDATA* file from the working directory and save as a CSV file in the RSPARROW *"User Directory/data"* directory.

2. Execute the optional SAS program that exports the labels of the SAS *data1* variables to a CSV file for use in RSPARROW.

- The SAS program *output_SAS_labels.sas* (G. Schwarz, USGS, written communication, 2018) is provided in the "*RSPARROW_master/inst/SAS*" sub-directory. The program outputs a CSV file with the SAS data1 variable header labels. This CSV file can be used to populate the variable explanations in the RSPARROW data dictionary file (*dataDictionary.csv*). Users are required to declare all the paths/filenames in the header of the program and allow the SAS data1 file to reside either in an external directory or the internal SAS WORK library.
- A second program, *import_csv_data1.sas* (G. Schwarz, USGS, written communication, 2018), is also provided for users who would like to import the variable labels in the RSPARROW data dictionary into the SAS data1 file to support modeling in SAS.

3. Use the instructions in Chapter 4.2 to guide the setup of the control script, data Dictionary, and parameter control input files for the model. The instructions include a detailed checklist in sub-section 4.2.4 for the setup and testing of the system settings and new models. This includes instructions for using the control script in an RStudio session to set pathnames, verify the RSPARROW library installation, set data import options, setup mapping shape files, verify the river reach connectivity, and establish control settings for model execution, prediction, and mapping.

4. To execute the model in RSPARROW in an RStudio session (step 6 in the checklist in sub-section 4.2.4), the following settings and data statements should be applied in the *sparrow_control.R* script:

- Set `if_mean_adjust_delivery_vars <- "yes"` if mean adjustment is required.
- Ensure that the land use variables in the `class_landuse` setting (section 7 of the control script) are defined in areal units; use the *userModifyData.R* script to calculate these variables in areal units if they do not already exist in the *data1.csv* file.
- Ensure that any missing data for model variables are set equal to zero. This is implemented by using the function `replaceNAs(named.list(landusevariable1, landusevariable2, landusevariable3))` in the *userModifyData.R* script.
- Users can select `if_modifySubdata<-"no"` if no data modifications are necessary and all model data are present with proper units in the *data1.csv* file.
- To obtain RSPARROW model coefficient estimates with values similar to those for the SAS estimated coefficients for a given model specification, enter the SAS estimated coefficients (with four or more significant figures) as the initial values (*parmInit*) in the *parameters.csv* file when executing the model in RSPARROW. An illustration of the use of this approach is presented for the tutorial model 8 in Chapter sub-section 6.2.8.
- To evaluate alternative model specifications with different explanatory variables, consult sub-section 4.4.4 for guidance on model estimation in RSPARROW.

5 Model output and explanations

5.1 Model data directory: (run_id)/data

Three R binary object files are output to the model *data* directory. In an RStudio session, these objects ensure global system-wide access to the *sparrowNames* variables in the *dataDictionary.csv* file for reach and monitoring site attributes. The exported binary files serve as an archive of the data outcomes for the user's control settings for the evaluated SPARROW model. These and other binary files with model results can be accessed by RSPARROW in subsequent RStudio sessions to support prediction and mapping without having to re-estimate a model.

5.1.1 subdata (R binary object)

This binary file is a modified version of the *data1.csv*, with the contents defined by the *sparrowNames* variables in the *dataDictionary.csv* file (see Chapter sub-section 3.2 for a detailed description).

Modification of the number of reach records of the *data1.csv* is on based on user-defined filters, using the control setting `filter_data1_conditions` (section 2 of the control script). The contents of the *sparrowNames* variables is subject to modification by user-defined R statements in the *userModifyData.R* script (see sub-section 3.5).

The records of the binary file are sorted in hydrological order (from upstream to downstream)–i.e., ascending order by the system variable *hydseq* (See Table 4 in section 3.2 for explanation).

In addition to the *dataDictionary sparrowNames* variables, the following system variable names, associated with the calibration and validation sites, are stored in the *subdata* object:

staidseq = unique station sequence number assigned to each reach that identifies the *staid* (unique station ID sequence number) for the nearest downstream calibration monitoring station (length=number of reaches). This has utility to identify the collection of reaches associated with the incremental drainage area between calibration monitoring sites.

vdepvar = The mean annual load associated with the validation monitoring sites.

vstaid = The unique station ID sequence number associated with the validation monitoring sites. The number is hydrologically ordered from upstream to downstream.

vstaidseq = unique station sequence number assigned to each reach that identifies the *vstaid* for the nearest downstream validation monitoring station (length=number of reaches).

sitedata.demtarea.class = a vector with the reach value for the upper boundary of the decile classes of the total drainage area system variable, *demtarea* (length=number of reaches). The decile class boundary value is used as a default classification variable (where `classvar<-NA`; section 5 of the control script) in diagnostic plotting functions to support evaluations of model performance and sensitivities.

5.1.2 sitedata (R binary object)

All variables in the *subdata* object are contained in the *sitedata* object, with the number of records equal to the number of calibration sites. The records are sorted in hydrological (from upstream to downstream reach) order using the *hydseq* system variable.

In addition to the *dataDictionary sparrowNames* variables, the following system variable names, associated with the calibration sites, are stored in the *subdata* object:

staidseq = unique station sequence number that identifies the *staid* (unique station ID sequence number) for the nearest downstream calibration monitoring station (length=number of reaches). This has utility to identify the collection of reaches associated with the incremental drainage area between calibration monitoring sites.

sitedata.demtarea.class = a vector with the monitored reach value for the upper boundary of the decile classes of the total drainage area system variable, *demtarea* (length=number of calibration sites). The decile

class boundary value is used as a default classification variable (where `classvar<-NA`; section 5 of the control script) in diagnostic plotting functions to support evaluations of model performance and sensitivities.

5.1.3 vsitedata (R binary object)

All variables in the *subdata* object are contained in the *vsitedata* object, with the number of records equal to the number of validation sites. The records are sorted in hydrological (from upstream to downstream reach) order using the *hydseq* system variable. The *vsitedata* object is created by user specification of the `if_validate <- "yes"` setting (section 6 of the control script).

In addition to the *dataDictionary sparrowNames* variables, the following system variable names, associated with the validation sites, are stored in the *subdata* object:

vdepvar = The mean annual load associated with the validation monitoring sites.

vstaid = The unique station ID sequence number associated with the validation monitoring sites. The number is hydrologically ordered from upstream to downstream.

vstaidseq = unique station sequence number assigned to each reach that identifies the *vstaid* for the nearest downstream validation monitoring station (length=number of reaches).

sitedata.demtarea.class = a vector with the monitored reach value for the upper boundary of the decile classes of the total drainage area system variable, *demtarea* (length=number of validation sites). The decile class boundary value is used as a default classification variable (where `classvar<-NA`; section 5 of the control script) in diagnostic plotting functions to support evaluations of model performance and sensitivities.

5.2 Model estimation directory: (run_id)/estimate

5.2.1 Bivariate correlations among explanatory variables

The control setting `if_corrExplanVars<-"yes"` (section 5 of the control script) outputs three files (described in the following three sub-sections) with information on the bivariate Spearman Rho correlations between all user-selected explanatory variables for the model in the *parameters.csv* file where the user also enters a *parmCorrGroup* value of "1".

5.2.1.1 (run_id)_explvars_correlations.txt

The file displays multiple correlation matrices of all possible bivariate Spearman Rho correlations among the user-selected explanatory variables in the SPARROW model, based on different spatial domains (reaches, incremental area between monitoring sites), data subsets, and log transformed observations.

By default, correlation results are reported for all reaches, including correlation matrices for all observations (with N equal the number of reaches), a subsample of observations (N=500), and the logged values for the subsampled observations. The subsample of n=500 and the logged transformed observations are reported to assist with the viewing and interpretation of the bivariate plots, in cases where the number of reaches are large and nonlinearities occur in the relations.

If more than 10 monitoring sites are identified for use in model estimation, then correlation results are reported for the area-weighted mean of the explanatory variables for the incremental drainage area between monitoring sites.

```
#-------------------------------------------------------------------------------
  CORRELATION MATRICES FOR EXPLANATORY VARIABLES (Site Incremental Areas)

  SPEARMAN CORRELATIONS FOR ALL OBSERVATIONS
                    ndep      MANC_N      FARM_N  PPT30MEAN    meanTemp soil_CLAYAVE
ndep          1.00000000  0.59673095  0.67800713 -0.2361533 -0.20508125  -0.03047798
MANC_N        0.59673095  1.00000000  0.74916808 -0.3062842 -0.25619694   0.01388232
FARM_N        0.67800713  0.74916808  1.00000000 -0.2313667 -0.06351606   0.21104555
```

```
PPT30MEAN    -0.23615334 -0.30628417 -0.23136673  1.0000000  0.86486660   0.25390557
meanTemp     -0.20508125 -0.25619694 -0.06351606  0.8648666  1.00000000   0.46058929
soil_CLAYAVE -0.03047798  0.01388232  0.21104555  0.2539056  0.46058929   1.00000000


  CORRELATION MATRICES FOR EXPLANATORY VARIABLES (Reaches)

  SPEARMAN CORRELATIONS FOR ALL OBSERVATIONS
                    ndep      MANC_N      FARM_N  PPT30MEAN    meanTemp soil_CLAYAVE
ndep          1.00000000  0.69326532  0.69577192 -0.1229751 -0.12995894   0.04957898
MANC_N        0.69326532  1.00000000  0.88494077 -0.1837570 -0.08265758   0.15122712
FARM_N        0.69577192  0.88494077  1.00000000 -0.1649129  0.02349329   0.27971030
PPT30MEAN    -0.12297511 -0.18375704 -0.16491288  1.0000000  0.84408036   0.34682528
meanTemp     -0.12995894 -0.08265758  0.02349329  0.8440804  1.00000000   0.51798819
soil_CLAYAVE  0.04957898  0.15122712  0.27971030  0.3468253  0.51798819   1.00000000

  SPEARMAN CORRELATIONS FOR SUBSAMPLE OF OBSERVATIONS (n=500)
                    ndep      MANC_N       FARM_N  PPT30MEAN     meanTemp soil_CLAYAVE
ndep          1.0000000  0.68403655  0.693196445 -0.1043255 -0.123752463  -0.01447000
MANC_N        0.6840365  1.00000000  0.901946013 -0.2135330 -0.111816857   0.09209832
FARM_N        0.6931964  0.90194601  1.000000000 -0.1776974  0.003298761   0.19964039
PPT30MEAN    -0.1043255 -0.21353298 -0.177697358  1.0000000  0.827506315   0.35221865
meanTemp     -0.1237525 -0.11181686  0.003298761  0.8275063  1.000000000   0.51387064
soil_CLAYAVE -0.0144700  0.09209832  0.199640391  0.3522187  0.513870644   1.00000000

  SPEARMAN CORRELATIONS FOR SUBSAMPLED LOGGED OBSERVATIONS (zero values are converted
                                                    to minimum of non-zero values)
                    ndep      MANC_N      FARM_N  PPT30MEAN    meanTemp soil_CLAYAVE
ndep          1.00000000  0.63771781  0.63547330 -0.1036732 -0.14392781   0.02789517
MANC_N        0.63771781  1.00000000  0.87440446 -0.1609421 -0.06671169   0.19872728
FARM_N        0.63547330  0.87440446  1.00000000 -0.1555643  0.03292049   0.31274454
PPT30MEAN    -0.10367323 -0.16094209 -0.15556434  1.0000000  0.82129105   0.34441928
meanTemp     -0.14392781 -0.06671169  0.03292049  0.8212910  1.00000000   0.49464411
soil_CLAYAVE  0.02789517  0.19872728  0.31274454  0.3444193  0.49464411   1.00000000
```

Summary metrics as shown below are also reported for the user-selected explanatory variables for the monitoring site incremental areas (where more than 10 monitoring sites have been identified for use in model estimation) and by default for all stream reaches.

```
#------------------------------------------------------------------------------
  SUMMARY METRICS FOR EXPLANATORY VARIABLES (Site Incremental Areas)
      ndep              MANC_N            FARM_N          PPT30MEAN         meanTemp
 Min.   :   405.9   Min.   :      0   Min.   :      0   Min.   : 427.5   Min.   : 2.469
 1st Qu.: 77172.3   1st Qu.:  19126   1st Qu.: 115221   1st Qu.: 853.8   1st Qu.: 7.765
 Median :131158.7   Median :  75546   Median : 488558   Median : 958.1   Median : 9.829
 Mean   :149644.0   Mean   : 153446   Mean   : 798332   Mean   : 958.7   Mean   : 9.358
 3rd Qu.:193442.5   3rd Qu.: 208438   3rd Qu.:1159268   3rd Qu.:1064.0   3rd Qu.:11.080
 Max.   :842636.7   Max.   :3216720   Max.   :6799758   Max.   :1508.5   Max.   :14.857

  FILTERED SUMMARY METRICS FOR EXPLANATORY VARIABLES (zero values converted to minimum
                                                    of non-zero values)
      ndep              MANC_N            FARM_N          PPT30MEAN         meanTemp
 Min.   :   405.9   Min.   :      2   Min.   :     12   Min.   : 427.5   Min.   : 2.469
 1st Qu.: 77172.3   1st Qu.:  19126   1st Qu.: 115221   1st Qu.: 853.8   1st Qu.: 7.765
```

```
Median :131158.7   Median :  75546   Median : 488558   Median : 958.1   Median : 9.829
Mean   :149644.0   Mean   : 153446   Mean   : 798332   Mean   : 958.7   Mean   : 9.358
3rd Qu.:193442.5   3rd Qu.: 208438   3rd Qu.:1159268   3rd Qu.:1064.0   3rd Qu.:11.080
Max.   :842636.7   Max.   :3216720   Max.   :6799758   Max.   :1508.5   Max.   :14.857



  SUMMARY METRICS FOR EXPLANATORY VARIABLES (Reaches)
     ndep               MANC_N            FARM_N           PPT30MEAN        meanTemp
Min.   :      3.8   Min.   :      0   Min.   :       0   Min.   : 380.2   Min.   : 2.323
1st Qu.:  16019.1   1st Qu.:    883   1st Qu.:    4699   1st Qu.: 835.5   1st Qu.: 7.124
Median :  45593.2   Median :  11610   Median :   57004   Median : 963.4   Median : 9.855
Mean   :  73340.8   Mean   :  64569   Mean   :  321581   Mean   : 974.5   Mean   : 9.366
3rd Qu.:  96174.4   3rd Qu.:  61877   3rd Qu.:  354068   3rd Qu.:1128.0   3rd Qu.:11.821
Max.   :1075322.0   Max.   :4341221   Max.   :13396730   Max.   :1703.6   Max.   :14.923

 FILTERED SUMMARY METRICS FOR EXPLANATORY VARIABLES (zero values converted to minimum
                                                     of non-zero values)
     ndep               MANC_N            FARM_N           PPT30MEAN        meanTemp
Min.   :      3.8   Min.   :      0   Min.   :       1   Min.   : 380.2   Min.   : 2.323
1st Qu.:  16019.1   1st Qu.:    883   1st Qu.:    4699   1st Qu.: 835.5   1st Qu.: 7.124
Median :  45593.2   Median :  11610   Median :   57004   Median : 963.4   Median : 9.855
Mean   :  73340.8   Mean   :  64569   Mean   :  321581   Mean   : 974.5   Mean   : 9.366
3rd Qu.:  96174.4   3rd Qu.:  61877   3rd Qu.:  354068   3rd Qu.:1128.0   3rd Qu.:11.821
Max.   :1075322.0   Max.   :4341221   Max.   :13396730   Max.   :1703.6   Max.   :14.923
```

5.2.1.2 (run_id)_explvars_correlations.pdf

The file includes a scatterplot matrix for all possible bivariate Spearman Rho correlations among the user-selected explanatory variables in the model and includes boxplots of the explanatory variables.

In cases where more than 10 monitoring sites are identified for use in model estimation, two graphical displays are presented for the explanatory variables, which are calculated as area-weighted means for the incremental drainage area between monitoring sites:

- Scatterplot matrix with Lowess smooth for the raw data (see https://en.wikipedia.org/wiki/Local_regression for information on the Lowess technique)
- Boxplots of the logged raw values of the explanatory variables

By default, the following graphical displays are presented for all stream reaches:

- Scatterplot matrix with Lowess smooths for the raw data
- Boxplots of the raw values of the explanatory variables
- Scatterplot matrix with Lowess smooths for the log-transformed data
- Boxplots of the log-transformed values of the explanatory variables

Note that zero values are converted to minimum of non-zero values for the log-transformed data)

5.2.1.3 (run_id)_Cor.ExplanVars.list (R binary object)

The R list file contains the data and the correlation results associated with the execution of all possible bivariate Spearman Rho correlations among the user-selected explanatory variables in the model, as described in the previous first two sections of 5.2.1.

names = The *sparrowNames* associated with the matrix elements stored in the R list.

The following three matrices are included in the object and are associated with the area-weighted mean values of the explanatory variables for incremental areas between nested monitoring sites. The parameter names associated with the columns of the matrices are listed in the *names* vector in the object.

cmatrixM_all = a data matrix of the values (rows=number of calibration sites, columns=number of variables). The records are sorted by the station identification number, corresponding to the hydrologically ordered *staidseq* station sequence number in the *sitedata* R object.

cmatrixM_filter = a data matrix of the values (rows=number of calibration sites, columns=number of variables) with zeros converted to the minimum value to allow log transformation of positive values. The records are sorted by the station identification number, corresponding to the *staidseq* station sequence number in the *sitedata* R object.

cor.allValuesM = a matrix of the Spearman's Rho correlation values (rows and column equal to the number of variables) associated with the **cmatrixM_all** data matrix.

The following matrices are included in the object and are associated with the raw values of explanatory variables associated with individual reaches. The parameter names associated with the columns of the matrices are listed in the *names* vector in the object.

cmatrix_all = a data matrix of the reach values (rows=number of reaches, columns=number of variables). The records are sorted by the reach identification number *waterid*.

cmatrix_filter = a data matrix of the reach values (rows=number of reaches, columns=number of variables) with zeros converted to the minimum value to allow log transformation of positive values. The records are sorted by the reach identification number *waterid*.

cor.allValues = a matrix of the Spearman's Rho correlation values (rows and column equal to the number of variables) associated with the **cmatrix_all** data matrix.

cor.sampleValues = a matrix of the Spearman's Rho correlation values (rows and column equal to the number of variables) associated with 500 or fewer randomly sampled values from the **cmatrix_all** data matrix.

cor.sampleLogValues = a matrix of the Spearman's Rho correlation values (rows and column equal to the number of variables) associated with the **cmatrix_filter** data matrix.

nsamples = the number of randomly samples values used for calculation of the **cor.sampleValues** Spearman's Rho correlations.

5.2.2 Verification of reach network connectivity

In cases where the control setting `if_verify_demtarea<-"yes"` (section 2 of the control script) and the user supplies an independently calculated reach total drainage area (*demtarea*), RSPARROW performs a verification of the total drainage area (*demtarea*) and reach connectivity by comparing the user-supplied total drainage area values with RSPARROW determined estimates of the total drainage area. The RSPARROW newly-derived estimates are determined by summing the incremental drainage area (*demiarea*) for reaches using the RSPARROW area/load-accumulation function with the to-from node structure in the user's `input_data_fileName` (e.g., *data1.csv*). The *demtarea* variable, user-supplied or newly-derived, is a REQUIRED variable used in the SPARROW model execution.

For reaches where the RSPARROW newly-derived total drainage area differs from the user-supplied *demtarea* by more than 1 percent, a CSV file is output with selected reach attribute information; the output of a PDF file with mapped reach attributes is optional.

5.2.2.1 (run_id)_diagnostic_darea_mismatches.csv

For cases where the control setting `if_verify_demtarea<-"yes"` and the user supplies an independently calculated reach total drainage area (*demtarea*), the CSV file is created and contains the following reach

attributes (this file will be created only if there are reaches where the RSPARROW newly-derived total drainage area differs from the user-supplied *demtarea* by more than 1 percent):

waterid = Unique reach identification (ID) number.

originalWaterid = The value of the user's original *waterid* variable in the *dataDictionary.csv* in cases where the reach IDs and nodes are renumbered (i.e., where the integer magnitude is greater than 1.0e+06). This variable may be useful as a common identification attribute to digitally link to other user data files (e.g., ESRI).

fnode_pre = Reach from (upstream) node from the user's *data1.csv*.

tnode_pre = Reach from (downstream) node from the user's *data1.csv*.

frac_pre = Reach transport fraction, ranging from 0 to 1 (1=no diversion of water/mass), from the user's *data1.csv*.

demtarea_pre = Reach total drainage area from the user's *data1.csv*.

demtarea_post = RSPARROW total drainage area, post processing based on RSPARROW area/load-accumulation function.

hydseq_new = Unique hydrological sequence number determined by the RSPARROW hydseq function.

AreaRatio_NewOld = Ratio of the RSPARROW newly-derived total drainage area to the prior value of the total drainage area from the user's *data1.csv*.

headflag_new = The newly-determined headwater identifier value (1=headwater reach; 0=non-headwater reach) using the RSPARROW hydseq function.

headflag_check = The outcome of a comparison of the newly-determined headflag value with the pre-existing value from the user's *data1.csv*; no difference is coded as a blank, whereas a difference is coded as "DIFFER".

5.2.2.2 (run_id)_diagnostic_darea_mismatches.html

The control setting `if_verify_demtarea_maps<-"yes"` produces the plots and maps for the following reach attributes (this file will be created only if there are reaches where the RSPARROW newly-derived total drainage area differs from the user-supplied *demtarea* by more than 1 percent):

- A plot of the newly-calculated total drainage area (*demtarea_post*) vs. the pre-calculated (*demtarea*) total drainage area for the reaches where the two metrics differ by more than 1 percent.
- A reach map of the pre-calculated (*demtarea*) total drainage area.
- A reach map of the existing hydrological sequence number (*hydseq*). In cases where the user request a newly-generated *hydseq* (i.e., `calculate_reach_attribute_list`), this map will not appear.
- A reach map of the newly-determined hydrological sequence number (*hydseq*).
- A reach map of the *AreaRatio_NewOld* variable. This expresses the ratio of the RSPARROW newly-derived total drainage area to the prior value of the total drainage area from the user's *data1.csv*. In cases where the areas match to within 1 percent, a grey color code is used for the reach.

Note that a plotly interactive plot is automatically output in the browser-accessible HTML file. The `enable_plotlyMaps<-"yes"` setting (see section 8 of the *sparrow_control.R* script and Chapter sub-section 4.4.8.4) produces interactive maps. **Note that the setting `enable_plotlyMaps<-"yes"` produces HTML files that are very large in size; users should expect some delays when opening the HTML file content in a browser.** The setting `enable_plotlyMaps<-"no"` outputs non-interactive maps in the HTML file.

Each interactive plot and map in the HTML file displays an R *"plotly"* tool box banner, located in the upper right portion of each plot or map (e.g., Fig. 12), that allows users to interactively control various features of

each image. The interactive tool box provides users with a standard set of control options for the display image, including the ability to pan/zoom, box select areas of the image, rescale plot axes, sub-select features from the map legend for display, download 'png' files, and to display the data associated with plot and map features by hovering the mouse cursor over these features (i.e., "hover-text"). The "hover-text" feature allows users to display the data associated with individual data points in plots and the monitoring sites and stream segments in maps.

5.2.3 Model and parameter setup

5.2.3.1 (run_id)_DataMatrix.list (R binary object)

The list contains five elements with the geospatial data that are necessary to estimate a SPARROW model and generate model predictions. The object is created by the *createDataMatrix.R* function. The list elements include:

dataNames = the names of the FIXED and REQUIRED system variables and explanatory variables associated with the user-specified model.

betaNames = the names associated with the user-specified model parameters.

data = a data matrix of reach values (rows=number of reaches, columns=number of variables), with the names of the column variables given in the **dataNames** variable. In cases where the control setting `if_mean_adjust_delivery_vars <- "yes"` (section 4 of the control script) is applied, the data values for the land-to-water delivery variables are mean adjusted by subtracting the mean of all values in the spatial domain from each reach value.

beta = a data matrix containing the initial values for the user-specified model parameters (rows=number of reaches, columns=number of model parameters), with the names of the column variables given in the **betaNames** variable.

data.index.list = contains 22 index variables referencing the variable locations in the *data* and *beta* matrices (defined in the *createDataMatrix.R* function). The index variables are:

jwaterid = index for the SPARROW Reach Identifier in the *data* matrix

jstaid = index for the SPARROW Monitoring Station Identifier in the *data* matrix

jfnode = index for the upstream Reach Node Identifier in the *data* matrix

jtnode = index for the downstream Reach Node Identifier in the *data* matrix

jfrac = index for the fraction Upstream Flux Diverted to Reach in the *data* matrix

jiftran = index for the if reach transmits flux (1=yes, 0=no) in the *data* matrix

jtarget = index for the downstream target reach in the *data* matrix

jtotarea = index for the total upstream drainage area in the *data* matrix

jiarea = index for the incremental reach drainage area in the *data* matrix

jdepvar = index for the dependent variable mean annual streamflow in the *data* matrix

jhydseq = index for the SPARROW Reach Hydrologic Sequencing Code in the *data* matrix

jmean_flow = index for the mean flow in the *data* matrix

jcalsites = index for identifying the sites selected for model calibration or simulation in the *data* matrix

jsrcvar = index for the SOURCE parameter types in the *data* matrix

jdlvvar = index for the DELIVF parameter types in the *data* matrix

jdecvar = index for the STRM parameter types in the *data* matrix

jresvar = index for the RESV parameter types in the *data* matrix

jothervar = index for the OTHER parameter types in the *data* matrix

jbsrcvar = index for the SOURCE parameter types in the *beta* matrix

jbdlvvar = index for the DELIVF parameter types in the *beta* matrix

jbdecvar = index for the STRM parameter types in the *beta* matrix

jbresvar = index for the RESV parameter types in the *beta* matrix

jbothervar = index for the OTHER parameter types in the *beta* matrix

5.2.3.2 (run_id)_SelParmValues (R binary object)

The object contains variables associated with the user-defined selection of parameters for their SPARROW model. The object is created by the *selectParmValues.R* function. The source of the information is the *parameters.csv* file (see Chapter sub-section 3.3 and Table 5 for details on the variables in the CSV file).

The object contents include:

sparrowNames = an internal R system variable name for the model explanatory variable.

bcols = the number of variables selected for use in the model.

beta0 = initial value for the nonlinear least squares (NLLS) estimation or the fixed value for predictions in simulation mode.

betamin = minimum bound on the estimated or fixed parameter value.

betamax = maximum bound on the estimated or fixed parameter value.

betatype = the parameter type (SOURCE, DELIVF, STRM, RESV, OTHER).

pselect = a flag designating the user-selected parameters (0=not selected; 1=selected).

betaconstant = a flag designating the user-selected parameters that are set to a constant value and not estimated (0=not constant; 1=selected as a constant).

bsrcconstant = a flag designating the user-selected SOURCE type parameters that are set to a constant value and not estimated (0=not constant; 1=selected as a constant).

bCorrGroup = Parameter selection setting for inclusion or exclusion from diagnostic plots and matrices of all-possible explanatory variable Spearman's rank correlations (1=included; 0=excluded from the correlations).

srcvar = the *sparrowNames* of the SOURCE parameter types.

dlvvar = the *sparrowNames* of the DELIVF (land-water delivery) parameter types.

decvar = the *sparrowNames* of the STRM (stream decay) parameter types.

resvar = the *sparrowNames* of the RESV (reservoir decay) parameter types.

othervar = the *sparrowNames* of the OTHER parameter types.

5.2.3.3 (run_id)_weights.pdf

The file provides diagnostic information for the `NLLS_weights` control setting for the `"lnload"` option (section 4 of the control script). Two sets of values are plotted against the log transform of the predicted load: one is for the square of the logged model residuals (plotted as open circles) and the second is for the normalized weights. A red line shows the nonlinear least squares fit to the square of the logged residuals as a function of the log transform of the predicted load. A blue line shows the normalized weights, computed as the reciprocal of the predicted variance, normalized for the mean of the reciprocal weights.

5.2.4 Model summary metrics and diagnostic output

The files described below in eight sub-sections are output when a model is estimated (control setting `if_estimate<-"yes"`) or executed in simulation model (`if_estimate_simulation<-"yes"`).

- Section 5.2.4 describes a log record of the model execution iterations and related metrics (5.2.4.1), a model performance summary file (5.2.4.2), a station-by-station listing of model residual and performance metrics (5.2.4.3), a file of diagnostic plots and maps (5.2.4.4), and model sensitivity results (5.2.4.5).
- Sub-section 5.2.2.6 describes a prediction summary that additionally requires use of the setting `if_predict<-"yes"`.
- Sub-section 5.2.2.7 describes graphical output for validation sites when the `if_validate<-"yes"` control setting is used.
- Sub-section 5.2.2.8 summarizes the R binary output files that archive the model estimation results and diagnostic metrics, and allow subsequent use of the estimation results for prediction and mapping without having to re-estimate the model. These also have utility for experienced R users and developers.

5.2.4.1 (run_id)_log.txt

The log file contents, output by the *nlmrt* function *nlfb*, appears in the RStudio console window during model execution and is recorded in this text file. An example of the output for the SPARROW tutorial model is shown below.

```
#-----------------------------------------------------------------------------------------
lower: [1]     0     0     0     0     0 -10000 -10000 -10000 -10000 -10000    0     0    0
upper: [1] 10000 10000 10000 10000 10000  10000  10000  10000  10000  10000 1000 1000  100
ssminval = 3.085193e-52
```

Above, the initial display of lower and upper parameter bounds is shown.

```
#-----------------------------------------------------------------------------------------
Start:lamda: 1e-04  SS= 1269.175  at  = 1  = 1  = 1  = 1  = 1  = 1  = 1  = 1  = 1  = 1
                                      = 0.1  = 0.1  = 1                              1 / 0
roff = 0.01974369   converged =  FALSE
gradient projection =  -1133.895  g-delta-angle= 108.1665
Stepsize= 1
```

Results for iteration 1 displays the Sum of Squares (SS) for the starting values of the parameters.

```
#-----------------------------------------------------------------------------------------
<<lamda: 4e-05  SS= 161.2216  at  = 3.462128  = 2.789422  = 2.101426  = 1.698957
                                  = 1.571683  = 1.032549  = 0.1324208  = 0.02170433
                                  = 1.028502  = 0.9916986  = 0.3563034  = 0.2460158
                                  = 1.294346                                    2 / 1
roff = 0.004139408   converged =  FALSE
gradient projection =  -44.66054  g-delta-angle= 103.8326
Stepsize= 1
```

Results for iteration 2 displays the SS for the first adjustment to the initial parameter values.

```
#-----------------------------------------------------------------------------------------
lamda: 175.9219  SS= 115.7059  at  = 7.895277  = 5.143434  = 3.025148  = 1.189804
                                   = 7.39913  = 1.421906  = 0.1565517  = -3.554747
                                   = 1.114159  = 1.334169  = 0.4323695  = 0.2364273
                                   = 6.768724                                38 / 23
roff = 1.822244e-05   converged =  FALSE
gradient projection =  -0.0009375305  g-delta-angle= 123.139
Stepsize= 1
```

```
No parameter change

Time elapsed in optimization
    user  system elapsed
  44.68    0.39   45.12
```

The model is declared to be converged, based on a measure of negligle change in the parameters–i.e., `No parameter change`. The output indicates that 38 and 23 iterations (shown as `38 / 23` above) were executed to evaluate the model residuals (SS) and Jacobian first-order partial differential matrix, respectively. Results for the final iteration 38 displays the SS for the final optimized parameter values, followed by elapsed time in seconds.

Note that the above *nlfb* progress report for each iteration displays the results of a termination test of the relative offset orthogonality type (`roff`). This will typically indicate non-convergence (`converged = FALSE`) for SPARROW models as shown above, based on the use of an internal default *nlfb* setting for the floating-point equality offset shift ("offset=100"); however, this setting does not control the termination point for model estimation (see sub-section 4.4.4.2). Instead, the *nlfb* measure of the parameter change (and the final message `No parameter change`) should be used as the indicator of model convergence.

5.2.4.2 (run_id)_summary.txt

```
#-------------------------------------------------------------------------------
 MODEL ESTIMATION PERFORMANCE (Monitoring-Adjusted Predictions)
 MOBS NPARM  DF      SSE       MSE      RMSE       RSQ RSQ-ADJUST RSQ-YIELD PERCENT BIAS
  708    13 695 115.7058 0.1664831 0.4080235 0.9533492  0.9525437 0.8488054    -1.856245

 MODEL SIMULATION PERFORMANCE (Simulated Predictions)
 MOBS NPARM  DF      SSE       MSE    RMSE       RSQ RSQ-ADJUST RSQ-YIELD PERCENT BIAS
  708    13 695 155.0515 0.2230956 0.47233 0.9374856  0.9364062 0.7973918    -4.911171

   Simulated predictions are computed using mean coefficients from the NLLS model
     that was estimated with monitoring-adjusted (conditioned) predictions
```

The model performance metrics as shown above are reported separately for the conditioned (monitoring-adjusted) and unconditioned (simulated) predictions (See Chapter sub-section 4.4.7.1 for a discussion of the attributes of conditioned and unconditioned predictions, and see sub-section 4.4.4.5 for a discussion of the performance metrics).

`MODEL ESTIMATION` performance metrics provide average measures of the accuracy of the NLLS model estimation/calibration, based on the use of conditioned predictions in which the observed loads on monitored reaches are substituted for the model predicted loads. The conditioned predictions describe the most accurate reach predictions for the purpose of quantifying river loads and the model coefficients that quantify the effects of contaminant sources and hydrological and biogeochemical processes on stream quality; thus, conditioned predictions are preferred for estimating the model. The associated `ESTIMATION` performance metrics are used to assess the adequacy of the overall model fit to the observations.

`MODEL SIMULATION` metrics provide a measure of the predictive skill of the estimated model in simulation mode at the monitored locations and are well suited for comparing the prediction accuracy of different models because conditioning effects are removed (i.e., the effects of substituting the observed station loads for the model predicted loads). The `SIMULATION` performance metrics also give a generally preferred estimate of the expected average accuracy of the model when applied to unmonitored stream reaches.

The performance metrics are defined as follows (see section 1.5.4.1 in Schwarz et al., 2006, for formal definitions and equations):

MOBS = Number of monitoring stations used in the model calibration.

NPARM = Number of model parameters.

DF = degrees of freedom, computed as MOBS - NPARM.

SSE = Sum of Squares of Error, calculated as the sum of the squares of the weighted residuals.

MSE = Mean Sum of Squares of Error, computed as the quotient of the SSE and DF.

RMSE = Root Mean Sum of Squares of Error, computed as the square root of the MSE. For RMSE<0.6, the product of the RMSE and 100 approximates the percent error of the reach-level prediction associated with one standard deviation error.

RSQ = R-Squared, computed as the ratio of the sum of the model residuals to the total variance in the log transformed observed loads.

RSQ-ADJUST = Adjusted R-Squared, expressed as the RSQ adjusted for the number of degrees of freedom.

RSQ-YIELD = Yield R-Squared, expressed as the RSQ adjusted for the mean log drainage area. Much of the variation in load is associated with drainage area size, which is typically highly correlated with the contaminant sources variables. Thus, high RSQ values do not necessarily indicate the explanatory strength of the model. RSQ-YIELD provides a scale-independent measure of model fit that provides an improved measure of the process interpretability of the model.

PERCENT BIAS = Percent Bias, computed as the ratio of the sum of the model residuals (observed load - predicted load) to the sum of the observed load across all calibration sites, and multipled by 100 (see Moriasi et al., 2007). Positive values indicate prediction underestimation bias, whereas negative values indicate prediction overestimation bias.

```
#-------------------------------------------------------------------------------
 MODEL VALIDATION PERFORMANCE (Simulated Predictions)
 MOBS       SSE       MSE      RMSE       RSQ RSQ-ADJUST RSQ-YIELD PERCENT BIAS
  177  32.49526 0.1835890 0.4284729 0.9469203  0.9430365 0.8046648    -3.725237
```

Use of the control setting `if_validate <- "yes"` (section 8 of the control script) identifies a user-specified fraction (e.g., `pvalidate <- 0.25`) of the monitoring locations that are randomly selected as validation sites, and outputs performance metrics for the validation sites to the summary text file as illustrated above. When validation sites are selected, the fraction of the monitoring locations used for model calibration is equal to "1-`pvalidate`". The validation performance metrics are based on the unconditioned (simulated) model predictions.

The mean sum of squares of error, MSE, for validation sites is computed as the quotient of the sum of squares of error, SSE, and the number of validation observations, MOBS (G. Schwarz, USGS, written communication, 2019); this is an unbiased approximation of the MSE, under the assumptions that the number of validation observations is reasonably large in relation to the number of estimated model parameters and that both the validation and model estimation observations are randonly selected from the same population. The adjusted R-Squared, RSQ-ADJUST, is also computed using an adjustment based on the number of validation observations.

```
#-------------------------------------------------------------------------------
  PARAMETER SUMMARY
PARAMETER    ESTIMATE PARM TYPE    EST TYPE  INITIALVALUE MIN MAX  DESCRIPTION  UNITS
1  point      0.78950    SOURCE   Estimated     0.10       0 1000  Municipal point
2  ndep       0.51435    SOURCE   Estimated     0.10       0 1000  Atmospheric dep N
3  shrubgrass 0.00000    SOURCE   Fixed         1e-05      0 50000 Shrub/grass lands

   A 'Fixed' parameter estimation type (EST TYPE) indicates a user choice of a constant
     coefficient value or a coefficient estimate equal to zero, the minimum or maximum
     boundary value (this may indicate a statistically insignificant coefficient, a
     coefficient with a value outside of the bounds, or an unusually small initial
     parameter value).
```

PARAMETER = The parameter name as defined by *sparrowNames* in *parameters.csv*.

ESTIMATE = The mean estimate of the parameter.

PARM TYPE = The parameter type as defined by *parmType* in *parameters.csv*.

EST TYPE = The estimation type. The term "`Estimated`" indicates that the parameter values are determined by NLLS optimization. The term "`Fixed`" indicates that either (1) the user has selected to fix the parameter to a constant value (*parmInit*), with *parmMin* = *parmMax* and *parmMax* set to a non-zero value in *parameters.csv*; or (2) the estimated parameter is equal to zero, the minimum or maximum boundary value set by the user; this may indicate a statistically insignificant coefficient, a coefficient with a value outside of the parameter bounds, or an unusually small initial parameter value).

INITIAL VALUES = *parmInit* in *parameters.csv*.

MIN = *parmMin* in *parameters.csv*.

MAX = *parmMax* in *parameters.csv*.

DESCRIPTION = The parameter description as defined by *description* in *parameters.csv*.

PARAMETER UNITS = The parameter units as defined by *parmUnits* in *parameters.csv*.

```
#-------------------------------------------------------------------------------
  PARAMETER SUMMARY
PARAMETER    ESTIMATE PARM TYPE    EST TYPE  INITIALVALUE MIN MAX  DESCRIPTION  UNITS
1  point      0.78950    SOURCE   Estimated     0.10       0 1000  Municipal point
2  ndep       0.51435    SOURCE   Estimated     0.10       0 1000  Atmospheric dep N

   The model was estimated with a weighted error variance. The weights are proportional
     to the log predicted load to account for heteroscedasticity.

   NLLS_weights control setting = lnload
```

If the user chooses to execute a weighted NLLS model, by specifying one of the three optional methods in the `NLLS_weights` control setting, then the above message with confirmation of the user's selected method is printed in the summary file. No message is printed for the default setting (`NLLS_weights<-"default"`).

```
#-------------------------------------------------------------------------------

  PARAMETER ESTIMATES
   PARAMETER    PARM TYPE ESTIMATE SE     T      P-VALUE VIF  DESCRIPTION  PARAMETER UNITS
1  point        SOURCE     0.78950 0.1106  7.142 0.00000  1.10587 Municipal point source
2  ndep         SOURCE     0.51435 0.0374 13.766 0.00000  2.59608 Atmospheric deposition N
```

The above table includes statistical results from the nonlinear least squares estimation. See section 1.5.4.1 in Schwarz et al. (2006) for details on the computation and interpretation of the metrics.

PARAMETER = The parameter name as defined by *sparrowNames* in *parameters.csv*.

PARM TYPE = The parameter type as defined by *parmType* in *parameters.csv*.

ESTIMATE = The mean estimate of the parameter.

SE = The standard error of the mean estimate of the parameter.

T = The t-statistic for the parameter mean, computed as the ratio of the mean parameter estimate to its standard error.

P-VALUE = The statistical significance of the t-statistic, equivalent to a two-sided partial F test. The test evaluates the statistical significance of adding this single explanatory variable to a complex model that has all of the other variables present. In cases where the lower bound of the parameter is set to zero, as recommended for source and aquatic decay variables, users have the option of evaluating a one-sided test by multiplying the reported p-value by one-half.

VIF = The Variance Inflation Factor, a measure of the importance of multicollinearity in the explanatory variables, related to the effect that collinearity has on the coefficient variance, t-statistics, and confidence intervals. The square root of the VIF represents the proportion by which the t-statistic could be increased if multicollinearity were eliminated (Schwarz et al., 2006). Note that confirmation of multicollinearity requires evidence of statistically insignificant (e.g., P-VALUE>0.10) parameters with large VIF values. Confirmation of multicollinearity should also be accompanied by a value of the *eigenvalue spread* generally greater than 100 (see below; Schwarz et al., 2006).

DESCRIPTION = The parameter description as defined by *description* in *parameters.csv*.

PARAMETER UNITS = The parameter units as defined by *parmUnits* in *parameters.csv*.

```
#-------------------------------------------------------------------------------
  EigenValue Spread  Normal PPCC  SWilks W      P-Value   Mean Exp Weighted Error
           87.16093    0.9558289 0.9282152 6.709323e-18                  1.081097
```

EigenValue Spread - Computed from the eigenvalues of the X'X matrix of normalized gradients. Values greater than 100 are indicative of multicollinearity (Schwarz et al., 2006), and are typically associated with two or more parameters with high VIF values.

Normal PPCC - The Normal Probability Plot Correlation Coefficient provides a measure of the linear correlation between the ordered, standardized weighted residuals from the estimated parametric model and the quantiles of the standard normal distribution. A value of the correlation coefficient near one is evidence that the residuals are from a normal distribution, whereas a value below 0.98 is generally indicative of non-normal residuals (Schwarz et al., 2006).

SWilks W - The Shapiro-Wilk's test statistic provides a formal test of the normality assumption for the model residuals, with statistical significance reported by the corresponding **P-Value**.

Mean Exp Weighted Error - This is the bias correction log retransformation factor computed according to equation 1.124 in Schwarz et al. (2006). It is computed as a simple mean of the exponentiated leverage-weighted log residuals (see section 4.4.4). This estimate is called a Smearing estimator (Duan, 1983). For NLLS estimates, it is unbiased for large samples regardless of the underlying error distribution, but is potentially biased in small samples even with normal residuals; weighting of the residuals may reduce bias in small samples (Schwarz et al., 2006). The standard RSPARROW predictions of the mean load and yield are corrected for log-retransformation bias (no corrections is applied to predictions of the delivery factor).

```
#-------------------------------------------------------------------------------
[1] LOG RESIDUALS, Station quantiles
    2.5%      10%      20%      30%      50%      70%      80%      90%      97%
-0.82255 -0.43930 -0.27660 -0.17440 -0.03400  0.09190  0.21020  0.41230  0.79543

[1] STANDARDIZED RESIDUALS, Station quantiles
    2.5%      16%      20%      30%      50%      70%      84%      90%      97%
-2.06190 -0.84076 -0.68000 -0.42950 -0.08350  0.22670  0.71076  1.02560  1.97213

[1] RATIO OF OBSERVED TO PREDICTED LOAD, Station quantiles
   2.5%     10%     20%     30%     50%     70%     80%     90%     97%
0.43905 0.64470 0.75840 0.83950 0.96650 1.09600 1.23380 1.50990 2.21602

[1] OBSERVED YIELD (kg/km2/yr), percentiles
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  12.82  462.73 1032.71 1463.38 2160.89 9944.97

[1] PREDICTED YIELD (kg/km2/yr), percentiles
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   27.6   508.8  1094.0  1404.0  2048.9  7278.0
```

Summary statistics are reported for the above metrics for the calibration sites. Model predictions are conditioned on the monitored loads. The residual metrics are based on conditioned predictions, and include the LOG RESIDUALS, expressed as the difference in the log observed load and log predicted load, and the STANDARDIZED RESIDUALS, computed as the leverage-weighted model residuals (see definitions below).

```
  MODEL VALIDATION (simulated predictions)
[1] LOG RESIDUALS, Station quantiles
    2.5%      10%      20%      30%      50%      70%      80%      90%      97%
-0.82240 -0.51420 -0.35900 -0.24160 -0.04200  0.12360  0.24100  0.44560  0.73772

  MODEL VALIDATION (simulated predictions)
[1] RATIO OF OBSERVED TO PREDICTED LOAD, Station quantiles
  2.5%    10%    20%    30%    50%    70%    80%    90%    97%
0.4398 0.5980 0.6982 0.7852 0.9590 1.1316 1.2724 1.5608 2.0910
```

Use of the control setting `if_validate <- "yes"` (section 8 of the control script) outputs the above summary metrics for the validation sites to the *(run_id)_summary.txt* file.

```
#--------------------------------------------------------------------------------
[1] LARGEST STANDARDIZED RESIDUALS (>3)
      waterid demtarea          rchname      station_id         station_name    staid
11209   91203   812.92        S SKUNK R        10850003    South Skunk River      181
```

The above listing is output for calibration sites with unusual outliers in the model estimation that satisfy the following conditions: a high leverage value greater than an internally computed acceptance criterion, a large standardized residual with an absolute value greater than 3, or a high Cook's D measure of influence (p-value<0.10). The list includes the above required *sparrowNames* system variable names from the *dataDictionary.csv* file (see Table 4).

The following variables are also output in the listing, which are not shown above:

classvar = the first variable for this control script setting, a spatially contiguous classification for the monitoring stations (see section 4.6.1).

standardResids = the standardized residuals. This measure of the model residual, standardized in relation to the overall leverage-weighted model variance (equation 1.102; Schwarz et al., 2006), is useful for identifying outlying observations that are poorly fit by the model. For normally distributed residuals, it is expected that on average no more than 3 in 1000 residuals should have an absolute value of the standardized residual that is larger than 3.

Resids = the model log residuals expressed as the difference in the log observed load and log predicted load.

leverage = The leverage statistic, defined according to equation 1.62 (Schwarz et al., 2006). High leverage values typically are associated with observations that exert a disproportionate influence on the estimated values of the model coefficients, related to outlying values associated with one or more explanatory variables. High leverage values are typically larger than three times the ratio of the number of model parameters to the number of observations (Helsel and Hirsch, 2002). The leverage values are defined such that their sum across all observations should equal the number of model parameters.

leverageCrit = The high leverage critical value equivalent to three times the ratio of the number of model parameters to the number of observations (Helsel and Hirsch, 2002).

CooksD = the Cook's D statistic, which provides a measure of influence of an observation on the model fit, as indicated by the combined effect of leverage and the model error as measured by the prediction residual (prediction error sum of squares). Observations with a statistically large Cook's D metric have an especially strong influence on the estimates of the model coefficients. The statistic is computed as the ratio of the square of the leverage-weighted prediction residual to the product of the number of model parameters and the mean square error (Helsel and Hirsch, 2002).

CooksDpvalue = the p-value associated with the Cook's D statistic, computed from an F distribution with

degress of freedom of the number of model parameters plus 1 and the the number of observations minus the number of model parameters (Helsel and Hirsch, 2002).

weight = The optional residual weight applied by a user to estimate the model using weighted Nonlinear Least Squares; a value of one is used as a default.

tiarea = The incremental area of the drainage located upstream of the monitoring site extending to the next upstream monitoring locations.

residCheck = the absolute value of **standardResids**.

```
#-------------------------------------------------------------------------------
  REGIONAL MODEL PERFORMANCE (Monitoring-Adjusted Predictions)
  REGION NUMBER OF SITES      SSE
1      4             121 13.97444
2      5             295 44.67647
3      7             252 42.06906
4      9              40 14.98582

  REGIONAL MODEL PERFORMANCE (Simulated Predictions)
  REGION      SSE
1      4 19.66945
2      5 53.71159
3      7 51.47973
4      9 30.19069
```

The above summary metrics provide a regional-level reporting of the number of monitoring sites and Sum of Squares of Error (**SSE**) for the user-define spatial classification variable (**REGION**) for the monitoring sites. The classification variable shown is the first variable defined for the **classvar** control setting (see section 4.6.1). The metrics are reported separately for predictions that are conditioned (monitoring-adjusted predictions) and unconditioned (simulated predictions with no monitoring adjustment).

```
#-------------------------------------------------------------------------------
 X'X EIGENVALUES AND EIGENVECTORS

               column 1    column 2   column 3      column 4       (last column)
EigenValues   3.52877982  1.79104329  1.24915172  0.935422425...... 0.181495983
point        -0.17762457  0.08707083  0.39221725 -0.317506456...... 0.004048305
ndep         -0.34241362  0.11753849  0.49451951  0.174124968...... 0.170035148
...
```

At the end of the *(run_id)_summary.txt* file, three matrices are output for the parameter covariances, parameter correlations, and eigensystem components (eignvalues and eigenvectors as shown above) for the X'X matrix. The parameter covariances and correlations are bivariate calculations.

In additional to using the VIF and eigenvalue spread to identify multicollinearity, the eigensystem components (as illustrated above for a subset of model parameters) can be used to identify specific parameters that are correlated with one another. The first row of output gives the eigenvalues, of length equal to the number of model parameters. The column beneath each eigenvalue displays the associated eigenvector. Identifying collinear parameters in the model is accomplished by selecting the column with an eigenvalue near zero (the last column of the ordered columns), and locating the largest absolute value elements in the column. These elements correspond to the predictor variables with collinear gradients. See section 1.5.4.3 in Schwarz et al. (2006) for further discussion of multicollinearity and use of the eigensystem components.

```
#-------------------------------------------------------------------------------
  CORRELATION MATRICES FOR EXPLANATORY VARIABLES (Site Incremental Areas)

  SPEARMAN CORRELATIONS FOR ALL OBSERVATIONS
```

```
                    ndep      MANC_N      FARM_N  PPT30MEAN    meanTemp soil_CLAYAVE
ndep          1.00000000  0.59673095  0.67800713 -0.2361533 -0.20508125  -0.03047798
MANC_N        0.59673095  1.00000000  0.74916808 -0.3062842 -0.25619694   0.01388232
FARM_N        0.67800713  0.74916808  1.00000000 -0.2313667 -0.06351606   0.21104555
PPT30MEAN    -0.23615334 -0.30628417 -0.23136673  1.0000000  0.86486660   0.25390557
meanTemp     -0.20508125 -0.25619694 -0.06351606  0.8648666  1.00000000   0.46058929
soil_CLAYAVE -0.03047798  0.01388232  0.21104555  0.2539056  0.46058929   1.00000000


  CORRELATION MATRICES FOR EXPLANATORY VARIABLES (Reaches)

  SPEARMAN CORRELATIONS FOR ALL OBSERVATIONS
                    ndep      MANC_N      FARM_N  PPT30MEAN    meanTemp soil_CLAYAVE
ndep          1.00000000  0.69326532  0.69577192 -0.1229751 -0.12995894   0.04957898
MANC_N        0.69326532  1.00000000  0.88494077 -0.1837570 -0.08265758   0.15122712
FARM_N        0.69577192  0.88494077  1.00000000 -0.1649129  0.02349329   0.27971030
PPT30MEAN    -0.12297511 -0.18375704 -0.16491288  1.0000000  0.84408036   0.34682528
meanTemp     -0.12995894 -0.08265758  0.02349329  0.8440804  1.00000000   0.51798819
soil_CLAYAVE  0.04957898  0.15122712  0.27971030  0.3468253  0.51798819   1.00000000

  SPEARMAN CORRELATIONS FOR SUBSAMPLE OF OBSERVATIONS (n=500)
                   ndep      MANC_N       FARM_N  PPT30MEAN     meanTemp soil_CLAYAVE
ndep          1.0000000  0.68403655  0.693196445 -0.1043255 -0.123752463  -0.01447000
MANC_N        0.6840365  1.00000000  0.901946013 -0.2135330 -0.111816857   0.09209832
FARM_N        0.6931964  0.90194601  1.000000000 -0.1776974  0.003298761   0.19964039
PPT30MEAN    -0.1043255 -0.21353298 -0.177697358  1.0000000  0.827506315   0.35221865
meanTemp     -0.1237525 -0.11181686  0.003298761  0.8275063  1.000000000   0.51387064
soil_CLAYAVE -0.0144700  0.09209832  0.199640391  0.3522187  0.513870644   1.00000000

  SPEARMAN CORRELATIONS FOR SUBSAMPLED LOGGED OBSERVATIONS (zero values are converted
                                                      to minimum of non-zero values)
                    ndep      MANC_N      FARM_N  PPT30MEAN    meanTemp soil_CLAYAVE
ndep          1.00000000  0.63771781  0.63547330 -0.1036732 -0.14392781   0.02789517
MANC_N        0.63771781  1.00000000  0.87440446 -0.1609421 -0.06671169   0.19872728
FARM_N        0.63547330  0.87440446  1.00000000 -0.1555643  0.03292049   0.31274454
PPT30MEAN    -0.10367323 -0.16094209 -0.15556434  1.0000000  0.82129105   0.34441928
meanTemp     -0.14392781 -0.06671169  0.03292049  0.8212910  1.00000000   0.49464411
soil_CLAYAVE  0.02789517  0.19872728  0.31274454  0.3444193  0.49464411   1.00000000
```

The control setting `if_corrExplanVars<-"yes"` (section 5 of the control script) outputs the above results for all possible bivariate Spearman Rho correlations between user-selected explanatory variables that are identified in the *parameters.csv* file with a *parmCorrGroup* value of one.

If more than 10 monitoring sites are identified for use in model estimation, then correlation results are reported for the area-weighted mean of the explanatory variables for the incremental drainage area between monitoring sites.

By default, correlation results are reported for all reaches, including correlation matrices for all observations (with N equal the number of reaches), a subsample of observations (N=500), and the logged values for the subsampled observations. The subsample of N=500 and the logged transformed observations are reported to assist with the viewing and interpretation of the bivariate plots (see sub-section 5.1.1.2), in cases where the number of reaches are large and nonlinearities occur in the relations.

Summary metrics as shown below are also reported for the user-selected explanatory variables for the monitoring site incremental areas (where more than 10 monitoring sites are identified for use in model

estimation) and by default for all stream reaches.

```
 SUMMARY METRICS FOR EXPLANATORY VARIABLES (Site Incremental Areas)
     ndep              MANC_N            FARM_N          PPT30MEAN         meanTemp
Min.   :   405.9   Min.   :      0   Min.   :      0   Min.   : 427.5   Min.   : 2.469
1st Qu.: 77172.3   1st Qu.:  19126   1st Qu.: 115221   1st Qu.: 853.8   1st Qu.: 7.765
Median :131158.7   Median :  75546   Median : 488558   Median : 958.1   Median : 9.829
Mean   :149644.0   Mean   : 153446   Mean   : 798332   Mean   : 958.7   Mean   : 9.358
3rd Qu.:193442.5   3rd Qu.: 208438   3rd Qu.:1159268   3rd Qu.:1064.0   3rd Qu.:11.080
Max.   :842636.7   Max.   :3216720   Max.   :6799758   Max.   :1508.5   Max.   :14.857

 FILTERED SUMMARY METRICS FOR EXPLANATORY VARIABLES (zero values converted to minimum
                                                    of non-zero values)
     ndep              MANC_N            FARM_N          PPT30MEAN         meanTemp
Min.   :   405.9   Min.   :      2   Min.   :     12   Min.   : 427.5   Min.   : 2.469
1st Qu.: 77172.3   1st Qu.:  19126   1st Qu.: 115221   1st Qu.: 853.8   1st Qu.: 7.765
Median :131158.7   Median :  75546   Median : 488558   Median : 958.1   Median : 9.829
Mean   :149644.0   Mean   : 153446   Mean   : 798332   Mean   : 958.7   Mean   : 9.358
3rd Qu.:193442.5   3rd Qu.: 208438   3rd Qu.:1159268   3rd Qu.:1064.0   3rd Qu.:11.080
Max.   :842636.7   Max.   :3216720   Max.   :6799758   Max.   :1508.5   Max.   :14.857



  SUMMARY METRICS FOR EXPLANATORY VARIABLES (Reaches)
     ndep              MANC_N            FARM_N          PPT30MEAN         meanTemp
Min.   :      3.8   Min.   :      0   Min.   :       0   Min.   : 380.2   Min.   : 2.323
1st Qu.:  16019.1   1st Qu.:    883   1st Qu.:    4699   1st Qu.: 835.5   1st Qu.: 7.124
Median :  45593.2   Median :  11610   Median :   57004   Median : 963.4   Median : 9.855
Mean   :  73340.8   Mean   :  64569   Mean   :  321581   Mean   : 974.5   Mean   : 9.366
3rd Qu.:  96174.4   3rd Qu.:  61877   3rd Qu.:  354068   3rd Qu.:1128.0   3rd Qu.:11.821
Max.   :1075322.0   Max.   :4341221   Max.   :13396730   Max.   :1703.6   Max.   :14.923

 FILTERED SUMMARY METRICS FOR EXPLANATORY VARIABLES (zero values converted to minimum
                                                    of non-zero values)
     ndep              MANC_N            FARM_N          PPT30MEAN         meanTemp
Min.   :      3.8   Min.   :      0   Min.   :       1   Min.   : 380.2   Min.   : 2.323
1st Qu.:  16019.1   1st Qu.:    883   1st Qu.:    4699   1st Qu.: 835.5   1st Qu.: 7.124
Median :  45593.2   Median :  11610   Median :   57004   Median : 963.4   Median : 9.855
Mean   :  73340.8   Mean   :  64569   Mean   :  321581   Mean   : 974.5   Mean   : 9.366
3rd Qu.:  96174.4   3rd Qu.:  61877   3rd Qu.:  354068   3rd Qu.:1128.0   3rd Qu.:11.821
Max.   :1075322.0   Max.   :4341221   Max.   :13396730   Max.   :1703.6   Max.   :14.923
```

5.2.4.3 (run_id)_residuals.csv

The CSV file is output for an estimated model (`if_estimate <- "yes"`), and includes the following required *sparrowNames* system variable names from the *dataDictionary.csv* file (see Table 4): **waterid**, **originalWaterid**, **demtarea**, **rchname**, **station_id**, **station_name**, **staid**.

The **originalWaterid** is the value of the user's original *waterid* variable in the *dataDictionary.csv* in cases where the reach IDs and nodes are renumbered (i.e., where the integer magnitude is greater than 1.0e+06). This variable may be useful as a common identification attribute to digitally link to other user data files (e.g., ESRI).

Additionally, the following variables are output to the file:

`classvar[1]` = the first element of this control script setting, a spatially contiguous classification for the

monitoring stations (see section 4.6.1).

Obs = the monitoring site observed load value.

predict = the model predicted load value, based on the use of prediction conditioning (monitoring adjustment) in reaches with monitored load estimates. See section 4.6.3 for details on conditioned and unconditioned model predictions.

ObsYield = the model observed yield value, calculated as the **Obs** value divided by the total drainage area, *demtarea*. Note that excessively high or low observed yield estimates are potentially indicative of errors associated with the estimation of the long-term mean load from the monitoring site water-quality record. Relatively high yields could also indicate the influence of a contaminant source (e.g., wastewater treatment discharge) on the observed load; if associated with a large model residual, this could indicate model mis-specification related to an incomplete accounting of a source.

predictYield = The model predicted yield value, calculated as the **predict** value divided by the total drainage area, *demtarea*. Note that excessively high or low model predicted yield estimates are potentially indicative of prediction outliers related to model mis-specification.

Resids = the model log residuals expressed as the difference in the log observed load and log (conditioned) predicted load.

standardResids = the standardized residuals. This measure of the model residual, standardized in relation to the overall leverage-weighted model variance (equation 1.102; Schwarz et al., 2006), is useful for identifying outlying observations that are poorly fit by the model. For normally distributed residuals, it is expected that on average no more than 3 in 1000 residuals should have an absolute value of the standardized residual that is larger than 3.

leverage = the leverage statistic, defined according to equation 1.62 (Schwarz et al., 2006). High leverage values typically are associated with observations that exert a disproportionate influence on the estimated values of the model coefficients, related to outlying values associated with one or more explanatory variables. High leverage values are typically larger than three times the ratio of the number of model parameters to the number of observations (Helsel and Hirsch, 2002). The leverage values are defined such that their sum across all observations should equal the number of model parameters.

leverageCrit = The high leverage critical value equivalent to three times the ratio of the number of model parameters to the number of observations (Helsel and Hirsch, 2002).

CooksD = the Cook's D (distance) statistic, which provides a measure of influence of an observation on the model fit, as indicated by the combined effect of leverage and the model error as measured by the prediction residual (prediction error sum of squares). Observations with a statistically large Cook's D metric have an especially strong influence on the estimates of the model coefficients. The statistic is computed as the ratio of the square of the leverage-weighted prediction residual to the product of the number of model parameters and the mean square error (Helsel and Hirsch, 2002).

CooksDpvalue = the p-value associated with the Cook's D statistic, computed from an F distribution (with numerator degrees of freedom equal to the number of model parameters plus 1, and denominator degrees of freedom equal to the number of observations minus the number of model parameters; Helsel and Hirsch, 2002).

boot_resid = the model log residuals, expressed as the difference in the log observed load and log predicted load, weighted by the leverage (computed as the reciprocal square root of one minus the leverage).

weight = The optional residual weight applied by a user to estimate the model using weighted NLLS; a value of one is used as a default.

tiarea = The incremental area of the drainage located upstream of the monitoring site extending to the next upstream monitoring locations.

pResids = The model residuals expressed as the difference in the log observed load and log unconditioned (simulated) predicted load. See section 4.6.3 for details on conditioned and unconditioned model predictions.

ratio.obs.pred = The ratio of the observed to the (conditioned) predicted loads.

pratio.obs.pred = The ratio of the observed to the (unconditioned) predicted loads.

xlat = The latitude of the monitoring station.

xlon = The longitude of the monitoring station.

(model coefficient name)_Jgradient = A column of the Jacobian gradients is output for each of the estimated model coefficients, with the *sparrowNames* shown for the coefficient name. The gradients are the first-order partial derivatives of the model residuals with respect to the model coefficients. The column is of length N where N is number of residuals associated with the observations (sites).

Users may also specify additional *sparrowNames* variables from the *dataDictionary.csv* to include in the residuals file by listing the variables in the control setting (section 7 of the control script) `add_vars<-c("sparrowName1","sparrowName2",...)`. These appear as additional columns after the Jacobian gradient columns for the estimated coefficients.

The contents of the residuals CSV file is also output as a shape file format if the third element of the `outputESRImaps` control setting is "yes".

5.2.4.4 (run_id)_sparrowEsts (R binary object)

The object is output for an estimated model (`if_estimate <- "yes"`) and contains the following summary variables:

resid = the model log residuals expressed as the difference in the log observed load and log (conditioned) predicted load.

jacobian = The first-order partial derivatives of the model residuals with respect to the model coefficients, which produces a N x K matrix, where N is number of residuals associated with the observations (calibration sites) and K is the number of estimated coefficients.

feval = the number of residual evaluations (sum of squares computations) used to estimate the model.

jeval = the number of Jacobian evaluations used to estimate the model.

coefficients = the estimated parameter values for the model associated with the minimum sum of square of error among the evaluated model iterations.

ssquares = Sum of Squares of Error, calculated as the sum of the squares of the weighted residuals.

lower = minimum bound on the estimated parameter value.

upper = maximum bound on the estimated parameter value.

maskidx = a vector of indices of the parameters to be masked. A value of NULL appears for standard SPARROW models.

betamn = minimum bound on the estimated parameter value.

betamx = maximum bound on the estimated parameter value.

if_mean_adjust_delivery_vars = A character value of "yes" or "no", indicating whether the land-to-water delivery variables are mean adjusted to improve the interpretability of the source coefficients.

NLLS_weights = the user-specified residuals weighting option for the NLLS model estimation.

incr_delivery_specification = the R expression used for the land-to-water delivery specification.

reach_decay_specification = the R expression used for the stream decay specification.

reservoir_decay_specification = the R expression used for the reservoir decay specification.

dlvdsgn = the source and land-to-water delivery interaction matrix, a SxD matrix, where S is the number of sources and D is the number of land-to-water delivery variables.

5.2.4.5 (run_id)_JacobResults (R binary object)

The object is output for an estimated model (`if_estimate <- "yes"`) and contains the following summary variables:

Parmnames = the names of the user-selected model parameters as defined by *sparrowNames* in *parameters.csv*.

Beta.initial = initial value for the nonlinear least squares (NLLS) estimation or the fixed value for predictions in simulation mode, as defined by *parmInit* in *parameters.csv*.

bmin = minimum bound on the estimated or fixed parameter value, as defined by *parmMin* in *parameters.csv*.

bmax =maximum bound on the estimated or fixed parameter value, as defined by *parmMax* in *parameters.csv*.

esttype = the estimation type. The term "`Estimated`" indicates that the parameter values are determined by NLLS optimization. The term "`Fixed`" indicates that either (1) the user has selected to fix the parameter to a constant value (*parmInit*), with *parmMin* = *parmMax* and *parmMax* set to a non-zero value in *parameters.csv*; or (2) the estimated parameter is equal to zero, the minimum or maximum boundary value set by the user; this may indicate a statistically insignificant coefficient, a coefficient with a value outside of the parameter bounds, or an unusually small initial parameter value).

btype = the parameter type (SOURCE, DELIVF, STRM, RESV, OTHER), as defined by *parmType* in *parameters.csv*.

oEstimate = a vector of the estimated and fixed (unestimated or constant) parameter values for the model.

oSEj = the standard error of the mean estimate of the parameter, based on the first-order Jacobian estimates.

oTj = the t-statistic for the parameter mean, computed as the ratio of the mean parameter estimate to its Jacobian standard error.

opTj = the statistical significance of the t-statistic (oTj), equivalent to a two-sided partial F test.

oVIF = the Variance Inflation Factor, a measure of the importance of multicollinearity in the explanatory variables, related to the effect that collinearity has on the coefficient variance, t-statistics, and confidence intervals.

odesign = the source and land-to-water delivery interaction matrix, a SxD matrix, where S is the number of sources and D is the number of land-to-water delivery variables.

oNames = the names of the user-selected parameters as defined by *sparrowNames* in *parameters.csv*.

e_val_spread = Computed from the eigenvalues of the X'X matrix of normalized gradients. Values greater than 100 are indicative of multicollinearity (Schwarz et al., 2006), and are typically associated with two or more parameters with high VIF values.

ppcc = The Normal Probability Plot Correlation Coefficient provides a measure of the linear correlation between the ordered, standardized weighted residuals from the estimated parametric model and the quantiles of the standard normal distribution. A value of the correlation coefficient near one is evidence that the residuals are from a normal distribution, whereas a value below 0.98 is generally indicative of non-normal residuals (Schwarz et al., 2006).

shap.test = The Shapiro-Wilk's test statistic provides a formal test of the normality assumption for the model residuals, with statistical significance reported by the corresponding value for **shap.p**.

mean_exp_weighted_error = the bias correction log retransformation factor computed according to equation 1.124 in Schwarz et al. (2006). It is computed as a simple mean of the exponentiated leverage-weighted log residuals (see section 4.4.4). This estimate is called a Smearing estimator (Duan, 1983).

boot_resid = the model log residuals, expressed as the difference in the log observed load and log predicted load, weighted by the leverage (computed as the reciprocal square root of one minus the leverage).

e_vec = a matrix of the eigensystem components (eignvalues and eigenvectors) associated with the X'X matrix. The matrix is printed in the *(run_id)_summary.txt* file. The first row of output gives the eigenvalues, of length equal to the number of estimated model parameters. The column beneath each eigenvalue displays the associated eigenvector for the parameters.

leverage = the leverage statistic, defined according to equation 1.62 (Schwarz et al., 2006).

jacobian = a matrix of the Jacobian gradients, with dimensions NxP, where N is the number of residuals associated with the observations (sites) and P is the number of estimated model coefficients. The gradients are the first-order partial derivatives of the model residuals with respect to the model coefficients.

5.2.4.6 (run_id)_HessianResults(R binary object)

The object is output for an estimated model (`if_estimate <- "yes"`) when the control setting `ifHess <- "yes"` is used to obtain the more accurate second-order Hessian standard errors. The object contains the following summary variables:

Parmnames = the names of the user-selected model parameters as defined by *sparrowNames* in *parameters.csv*.

Hesnames = the names of the user-selected model parameters (estimated only; SelParmValues$betaconstant==0) as defined by *sparrowNames* in *parameters.csv*.

oEstimate = a vector of the estimated and fixed (unestimated or constant) parameter values for the model.

oSEh = the standard error of the mean estimate of the parameter, based on the second-order Hessian estimates.

oTh = the t-statistic for the parameter mean, computed as the ratio of the mean parameter estimate to its Hessian standard error.

optH = the statistical significance of the t-statistic (oTh), equivalent to a two-sided partial F test.

cov2 = the covariance matrix (with PxP dimensions) for the estimated model parameters, where P is the number of estimated model coefficients.

cor2 = the correlation matrix (with PxP dimensions) for the estimated model parameters, where P is the number of estimated model coefficients.

HesRunTime = the execution time required for the Hessian calculations.

5.2.4.7 (run_id)_Mdiagnostics.list (R binary object)

The object is output for an estimated model (`if_estimate <- "yes"`), and contains the following summary variables for the calibration sites:

Obs = the monitoring site observed load value.

predict = the model predicted load value, based on the use of prediction conditioning (monitoring adjustment) in reaches with monitored load estimates. See section 4.6.3 for details on conditioned and unconditioned model predictions.

yldobs = the model observed yield value, calculated as the **Obs** value divided by the total drainage area, *demtarea*.

yldpredict = The model conditioned predicted yield value, calculated as the **predict** value divided by the total drainage area, *demtarea*.

xstaid = hydrologically ordered (upstream to downstream) unique calibration station sequence number (length=number of reaches).

tarea = the reach total drainage area.

ratio.obs.pred = The ratio of the observed to the (conditioned) predicted loads.

xlat = The latitude of the monitoring station.

xlon = The longitude of the monitoring station.

ppredict = the model unconditioned predicted load value. See section 4.6.3 for details on conditioned and unconditioned model predictions.

pyldobs = the model observed yield value, calculated as the **Obs** value divided by the total drainage area, *demtarea* (equivalent to *yldobs*).

pyldpredict = The model unconditioned predicted yield value, calculated as the **ppredict** value divided by the total drainage area, *demtarea*.

pratio.obs.pred = The ratio of the observed to the (unconditioned) predicted loads.

classgrp = a vector with the classes for the site attribute variable, listed as the first element of the `classvar` control script setting (section 5 of the script). The attribute variable provides a spatially contiguous classification for the calibration monitoring stations (see section 4.6.1) for evaluating model performance diagnostics.

RMSEnn = the number of calibration sites located within each of the *classgrp* classes.

SSEclass = the Sum of Squares of Error for each of the *classgrp* classes, calculated as the sum of the squares of the weighted residuals based on the conditioned predictions.

pSSEclass = the Sum of Squares of Error for each of the *classgrp* classes, calculated as the sum of the squares of the weighted residuals based on the unconditioned predictions.

Resids = the model log residuals expressed as the difference in the log observed load and log (conditioned) predicted load.

pResids = The model residuals expressed as the difference in the log observed load and log unconditioned (simulated) predicted load. See section 4.6.3 for details on conditioned and unconditioned model predictions.

standardResids = the standardized residuals. This measure of the model residual, standardized in relation to the overall leverage-weighted model variance (equation 1.102; Schwarz et al., 2006), is useful for identifying outlying observations that are poorly fit by the model. For normally distributed residuals, it is expected that on average no more than 3 in 1000 residuals should have an absolute value of the standardized residual that is larger than 3.

CooksD = the Cook's D (distance) statistic, which provides a measure of influence of an observation on the model fit, as indicated by the combined effect of leverage and the model error as measured by the prediction residual (prediction error sum of squares). Observations with a statistically large Cook's D metric have an especially strong influence on the estimates of the model coefficients. The statistic is computed as the ratio of the square of the leverage-weighted prediction residual to the product of the number of model parameters and the mean square error (Helsel and Hirsch, 2002).

CooksDpvalue = the p-value associated with the Cook's D statistic, computed from an F distribution (with numerator degrees of freedom equal to the number of model parameters plus 1, and denominator degrees of freedom equal to the number of observations minus the number of model parameters; Helsel and Hirsch, 2002).

leverage = the leverage statistic, defined according to equation 1.62 (Schwarz et al., 2006). High leverage values typically are associated with observations that exert a disproportionate influence on the estimated values of the model coefficients, related to outlying values associated with one or more explanatory variables. High leverage values are typically larger than three times the ratio of the number of model parameters to the number of observations (Helsel and Hirsch, 2002). The leverage values are defined such that their sum across all observations should equal the number of model parameters.

leverageCrit = The high leverage critical value equivalent to three times the ratio of the number of model parameters to the number of observations (Helsel and Hirsch, 2002).

5.2.4.8 (run_id)_ANOVA.list (R binary object)

The object is output for an estimated model (`if_estimate <- "yes"`), and contains the following model performance-related variables for the calibration sites:

mobs = Number of monitoring stations used in the model calibration.

npar = Number of model parameters.

DF = degrees of freedom, computed as *mobs* - *npar*.

SSE = Sum of Squares of Error, calculated as the sum of the squares of the weighted residuals.

MSE = Mean Sum of Squares of Error, computed as the quotient of the *SSE* and *DF*.

RMSE = Root Mean Sum of Squares of Error, computed as the square root of the *MSE*. For RMSE<0.6, the product of the RMSE and 100 approximates the percent error of the reach-level prediction associated with one standard deviation error.

RSQ = R-Squared, computed as the ratio of the sum of the model residuals to the total variance in the log transformed observed loads.

RSQ_ADJ = Adjusted R-Squared, expressed as the RSQ adjusted for the number of degrees of freedom.

RSQ_YLD = Yield R-Squared, expressed as the RSQ adjusted for the mean log drainage area. Much of the variation in load is associated with drainage area size, which is typically highly correlated with the contaminant sources variables. Thus, high RSQ values do not necessarily indicate the explanatory strength of the model. RSQ-YIELD provides a scale-independent measure of model fit that provides an improved measure of the process interpretability of the model.

NSeff = Nash-Sutcliffe model efficiency coefficient. Values can range from negative infinity to one, with a value of one indicating a perfect match between the observed and model predicted load.

PBias = Percent Bias, computed as the ratio of the sum of the model residuals (observed load - predicted load) to the sum of the observed load across all calibration sites, and multipled by 100 (see Moriasi et al., 2007). Positive values indicate prediction underestimation bias, whereas negative values indicate prediction overestimation bias.

pSSE = Sum of Squares of Error, calculated as the sum of the squares of the weighted residuals, based on use of the unconditioned predictions.

pMSE = Mean Sum of Squares of Error, computed as the quotient of the *pSSE* and *DF*.

pRMSE = Root Mean Sum of Squares of Error, computed as the square root of the *pMSE*.

pRSQ = R-Squared, computed as the ratio of the sum of the model residuals (based on the unconditioned predictions) to the total variance in the log transformed observed loads.

pRSQ_ADJ = Adjusted R-Squared (based on *pRSQ*), expressed as the *pRSQ* adjusted for the number of degrees of freedom.

pRSQ_YLD = Yield R-Squared, expressed as the *pRSQ* adjusted for the mean log drainage area.

pNSeff = Nash-Sutcliffe model efficiency coefficient, based on the use of the unconditioned predictions.

pPBisas = Percent Bias, computed as the ratio of the sum of the model residuals (observed load - unconditioned predicted load) to the sum of the observed load across all calibration sites, and multipled by 100 (see Moriasi et al., 2007).

5.2.4.9 (run_id)_diagnostic_plots.html

There are four major sub-sections of the output file, including an initial section with user-specified site attribute maps and three sections of model performance diagnostics (plots, maps) for the calibration sites. The model performance diagnostics are reported separately based on the use of conditioned and unconditioned

model predictions (see Chapter sub-section 4.4.7.1 for full explanation). These provide two informative perspectives on model performance for *estimation* and *simulation*.

Users may optionally output diagnostic plots to evaluate regional variations in model performance. The plots are enabled via control settings for user-specified regional classification variables that define physiographic regions, political jurisdictions, or land use types (see the `classvar` and `class_landuse` settings in section 5 of the control script and their description in Chapter sub-section 4.4.5).

Note that plotly interactive plots are automatically output in the model diagnostics file. The `enable_plotlyMaps<-"yes"` setting (see section 8 of the *sparrow_control.R* script and Chapter subsection 4.4.8.4 for details) outputs interactive plotly maps (i.e., browser-accessible HTML files) for the station attributes and residuals (section 4.4.8.3). **Note that the setting `enable_plotlyMaps<-"yes"` produces HTML files that are very large in size; users should expect some delays when opening the HTML file content in a browser.** The setting `enable_plotlyMaps<-"no"` outputs non-interactive maps in the HTML diagnostics files.

Each interactive plot and map in the HTML file displays an R *"plotly"* tool box banner, located in the upper right portion of each plot or map (e.g., Fig. 12), that allows users to interactively control various features of each image. The interactive tool box provides users with a standard set of control options for the display image, including the ability to pan/zoom, box select areas of the image, rescale plot axes, sub-select features from the map legend for display, download 'png' files, and to display the data associated with plot and map features by hovering the mouse cursor over these features (i.e., "hover-text"). The "hover-text" feature allows users to display the data associated with individual data points in plots and the monitoring sites and stream segments in maps.

5.2.4.9.1 User-specified site attribute maps

The initial pages of the HTML display optional maps for user-specified site attributes, as shown in the `map_siteAttributes.list` control setting (see section 8 of the control script; section 4.4.8.3.1 above).

5.2.4.9.2 Model estimation diagnostics

The model *estimation* diagnostic plots are output for the *conditioned* predictions, which are used to estimate the NLLS model coefficients. The conditioned predictions are monitoring adjusted–i.e., the observed load on reaches with calibration monitoring sites is substituted for the model predicted load. These predictions provide the most accurate reach predictions for use in estimating the model. The associated residuals and observed to predicted ratios provide the most relevant information about the accuracy of the model fit to observed loads.

The following plots are output to the file:

- Four-plot panel: observed vs. predicted loads, observed vs. predicted yields, log residuals vs. predicted loads, and log residuals vs. predicted yields. These plots can be useful to identify outliers, prediction biases, and cases where the model residuals are not identically distributed (i.e., heteroscedastic).
- Four-plot panel: boxplot of log residuals, boxplot of observed to predicted ratios, normal quantile plot of the standardized residuals, and plot of squared residuals vs. predicted loads. These plots can be used to identify cases where the model residuals are not identically distributed (i.e., heteroscedastic).
- Plot of conditioned (monitoring-adjusted) load predictions vs. unconditioned (simulated) load predictions. The plot provides complementary information about the model accuracy, related to the performance of the model in *simulation* mode when monitoring conditioning effects are removed (see Chapter sub-section 4.4.4.5 on measures of model performance). The plot can be used to identify differences in conditioned and unconditioned predictions along the prediction gradient but especially in larger streams where monitoring adjustment may improve prediction accuracy. Cases where the unconditioned predictions display systematic differences from the conditioned predictions may indicate prediction biases and limitations in the accuracy of the model when applied in simulation mode independently of monitoring load adjustments. Evidence of general agreement in the conditioned and unconditioned

predictions, with relatively uniform scatter along the line of equivalence, indicates that the simulated predictions are generally unbiased in relation to the conditioned predictions.

- Plots of the observed to predicted ratio vs. the area-weighted mean values of the user-selected explanatory variables for the incremental areas between calibration sites. The plots provide visual information about the accuracy of the model specification for each of the explanatory variables; in general, no correlation should be visible between the ratios and the mean values of the explanatory variable if the model is properly specified. The plots are output only if the control setting `if_corrExplanVars<-"yes"` (section 5 of the control script) is selected and only for the modeled variables where a value of "1" is entered in the *parmCorrGroup* labeled column in the *parameters.csv* file.
- Boxplots of the ratio of the observed to predicted loads vs. the decile classes of the total drainage area for the sites. The boxplots provide a measure of the general bias (symmetry of the interquartile range about a value of one) and precision (length of the interquartile range) of the model predictions across different sized drainages.
- Boxplots of the ratio of the observed to predicted loads vs. the contiguous spatial classes specified by users in the control setting `classvar` (e.g., HUC-4). The boxplots provide a measure of the general bias (symmetry of the interquartile range about a value of one) and precision (length of the interquartile range) of the model predictions across sub-regions of the modeled spatial domain. The number of plots (pages) is equal to number of user-specified variables.
- Boxplots of the ratio of the observed to predicted loads vs. the deciles of the land-use class variable specified by users in the control setting `class_landuse`, with the land-use classes expressed as a percentage of the incremental drainage area extending from the calibration site to the nearest upstream site locations. The boxplots provide a measure of the general bias (symmetry of the interquartile range about a value of one) and precision (length of the interquartile range) of the model predictions across different land-cover/use types.
- Four-plot panels are reported separately for each of the contiguous spatial classes specified for the first element of the control setting `classvar`: observed vs. predicted loads, observed vs. predicted yields, log residuals vs. predicted loads, and log residuals vs. predicted yields. The panels provide a more detailed look at the model residuals in relation to loads and yields across sub-regions of the modeled spatial domain, according to the user-specified contiguous spatial classes.

5.2.4.9.3 Model simulation diagnostics

The model *simulation* diagnostic plots are output for the *unconditioned* predictions. The unconditioned predictions are generated when the model is executed in simulation mode. In this mode, the predictions are computed using mean coefficients from the NLLS model estimated with monitoring-adjusted (conditioned) predictions; thus, the unconditioned predictions are not directly conditioned on the observed loads in monitored reaches. These predictions (and the associated residuals and observed to predicted ratios) provide the best measure of the predictive skill of the estimated model in simulation mode at the monitored locations.

Using the unconditioned predictions, the plots described in the above sub-section (5.2.4.9.2) are replicated (except for the plot of conditioned vs. unconditioned loads) and output to the file.

5.2.4.9.4 Maps of the residuals and the observed to predicted ratios for the calibration sites

The following maps are output to the HTML for both the conditioned and unconditioned predictions:

- Log residuals, based on monitoring conditioned predictions (labeled as "Model estimation residuals")
- Log residuals, based on the unconditioned predictions (labeled as "Model simulation residuals")
- Standardized residuals based on the conditioned predictions
- Ratio of observed to predicted loads for the conditioned predictions (labeled as "Model estimation Obs/Pred ratio")

- Ratio of observed to predicted load for the unconditioned predictions (labeled as "Model simulation Obs/Pred ratio")

5.2.4.10 (run_id)_diagnostic_sensitivity.html

The reported model coefficient sensitivities provide a measure of the relative importance of each explanatory variable on model predictions of load over the entire spatial domain and for sub-regions. The sensitivity measure is computed as the percentage change in the predicted values in response to a unit one-percent change in each explanatory variable, with all other explanatory variables held constant. The sensitivity measures are positively correlated with the coefficient t-statistics but provide an additional measure of the relative importance of the explanatory variables, which integrates the statistical significance with the magnitude of the predicted load response. The sensitivities are computed for all reaches but are reported for only the calibration sites to enhance the plotting efficiency.

Sensitivities are computed and output for estimated models (`if_estimate <- "yes"`) or simulated models (`if_estimate_simulation<-"yes"` and calibration site loads are available).

The last two pages of the file includes a statistical summary of the parameter sensitivities for the calibration sites for the full spatial domain. Plots are presented with the sensitivity summary metrics for each model coefficient, ordered from lowest to highest median value. The displayed summary metrics include the median (black point), the interquartile range (red brackets; 25th and 75th quantiles), and the 10th and 90th quantiles (blue brackets). Separate plots are shown with arithmetic and logarithmic scales for the sensitivity metrics.

As a measure with the regional specificity of the parameter sensitivities, the initial pages of the PDF (the plots for four model coefficients per page) show separate boxplot diagrams for each model coefficient with the spatial distributions of the sensitities for the spatial classes (e.g., huc2) specified for the first element of the `classvar` control setting. These plots provide information about the spatial distribution of the relative response of stream loads to changes in the explanatory variables.

Note that plotly interactive plots are automatically output to the HTML file (see section 8 of the *sparrow_control.R* script and Chapter subsection 4.4.8.4 for details). Each interactive plot displays an R *"plotly"* tool box banner, located in the upper right portion of each plot, that allows users to interactively control various features of each image. The interactive tool box provides users with a standard set of control options for the display image, including the ability to pan/zoom, box select areas of the image, rescale plot axes, download 'png' files, and to display the data associated with plot features by hovering the mouse cursor over these features (i.e., "hover-text"). The "hover-text" feature allows users to display the data associated with individual data points in plots.

5.2.4.11 (run_id)_sensitivities.list (R binary object)

The object contains sensitivity measures, computed as the percentage change in the predicted values in response to a unit one-percent change in each explanatory variable, with all other explanatory variables held constant. The sensitivity measures are computed for all reaches in the model domain, with the summary metrics as described below reported for only the calibration sites. Sensitivities are computed and output to the list object for estimated models (`if_estimate <- "yes"`) or simulated models (`if_estimate_simulation<-"yes"` and calibration site loads are available).

The variables in the object list are:

xparm = the names of the user-selected model parameters as defined by *sparrowNames* in *parameters.csv*.

xmed = median sensitivity for the modeled spatial domain for each of the user-selected model parameters (vector length is the number of model parameters, with the order identical to that of *xparm*). A value of 1.0 is stored for a fixed parameter.

xiqr = quantiles (10th, 25th, 75th, and 90th) for the sensitivity for each model parameter for the modeled spatial domain (QxP matrix, where Q is the quantile value from 10th to 90th and P is the number of model parameters, with the order identical to that of *xparm*). A value of 1.0 is stored for a fixed parameter.

xsens = sensitivity values for each of the calibration site and model parameters (NxP matrix, where N is the number of calibration sites and P is the number of model parameters, with the order identical to that of *xparm*). A value of 1.0 is stored for a fixed parameter.

5.2.4.12 (run_id)_summary_predictions.csv

The file is created when both the `if_estimate<-"yes"` and `if_predict<-"yes"` control settings are specified. The file contains selected model predictions that are summarized for all reaches in the full spatial domain, with reported mean, standard deviation, and percentiles. The predictions include the total load, total and incremental yields, flow-weighted concentration, delivery fraction, and the source shares expressed as a percentage of the incremental loads.

A summary of the model predictions of yield for catchments with relatively uniform land use types is also provided. The land use classes are defined by the control setting `class_landuse` (see Chapter sub-section 4.4.5), with the corresponding minimum land use area percentages of total drainage area defined by the `class_landuse_percent` setting. The number of watersheds satisfying the criteria is reported along with the mean, standard deviation, and percentiles of the distributions of yields.

5.2.4.13 (run_id)_validation_plots.html

In cases where the control setting `if_validate <- "yes"` is specified, the model performance diagnostics for the validation sites are reported in two major sections.

(a) Model simulation diagnostics are reported for the unconditioned (simulated) predictions, and include:

- Four-plot panel: observed vs. predicted loads, observed vs. predicted yields, log residuals vs. predicted loads, and log residuals vs. predicted yields
- Four-plot panel: boxplot of log residuals, boxplot of observed to predicted ratios, normal quantile plot of the standardized residuals, and plot of squared residuals vs. predicted loads
- Boxplots of the observed to predicted loads vs. the decile classes of the total drainage area for the sites
- Boxplots of the observed to predicted loads vs. the contiguous spatial variables specified by users in the control setting `classvar` (e.g., "huc2")
- Boxplots of the observed to predicted loads vs. the deciles of the land-use class variable specified by users in the control setting `class_landuse`, with the land-use classes expressed as a percentage of the incremental drainage area extending from the calibration site to the nearest upstream site locations.
- Four-plot panels are reported separately for each of the contiguous spatial classes specified for the first entry for the control setting `classvar`: observed vs. predicted loads, observed vs. predicted yields, log residuals vs. predicted loads, and log residuals vs. predicted yields

(b) Maps of the residuals and the observed to predicted ratios for validation sites:

- Log residuals, based on the unconditioned predictions
- Ratio of observed to predicted load for the unconditioned predictions

Note that plotly interactive plots are automatically output in the model diagnostics file. The `enable_plotlyMaps<-"yes"` setting (see section 8 of the *sparrow_control.R* script and Chapter subsection 4.4.8.4 for details) outputs interactive plotly maps (i.e., browser-accessible HTML files) for the station attributes and residuals (section 4.4.8.3). **Note that the setting `enable_plotlyMaps<-"yes"` produces HTML files that are very large in size; users should expect some delays when opening the HTML file content in a browser.** The setting `enable_plotlyMaps<-"no"` outputs non-interactive maps in the HTML diagnostics files.

Each interactive plot and map in the HTML file displays an R *"plotly"* tool box banner, located in the upper right portion of each plot or map, that allows users to interactively control various features of each image. The interactive tool box provides users with a standard set of control options for the display image, including

the ability to pan/zoom, box select areas of the image, rescale plot axes, sub-select features from the map legend for display, download 'png' files, and to display the data associated with plot and map features by hovering the mouse cursor over these features (i.e., "hover-text"). The "hover-text" feature allows users to display the data associated with individual data points in plots and the monitoring sites and stream segments in maps.

5.2.4.14 (run_id)_vMdiagnostics.list (R binary object)

The object is output for an estimated model (`if_estimate <- "yes"`) when validation sites are selected (`if_validate <- "yes"`), and contains the following summary variables for the validation sites:

Obs = the monitoring site observed load value.

xstaid = hydrologically ordered (upstream to downstream) unique validation station sequence number (length=number of reaches).

tarea = the reach total drainage area.

xlat = The latitude of the monitoring station.

xlon = The longitude of the monitoring station.

ppredict = the model unconditioned predicted load value. See section 4.6.3 for details on conditioned and unconditioned model predictions.

pyldobs = the model observed yield value, calculated as the **Obs** value divided by the total drainage area, *demtarea* (equivalent to *yldobs*).

pyldpredict = The model unconditioned predicted yield value, calculated as the **ppredict** value divided by the total drainage area, *demtarea*.

pratio.obs.pred = The ratio of the observed to the (unconditioned) predicted loads.

classgrp = a vector with the classes for the site attribute variable, listed as the first element of the `classvar` control script setting (section 5 of the script). The attribute variable provides a spatially contiguous classification for the calibration monitoring stations (see section 4.6.1) for evaluating model performance diagnostics.

pSSEclass = the Sum of Squares of Error for each of the *classgrp* classes, calculated as the sum of the squares of the weighted residuals based on the unconditioned predictions.

pResids = The model residuals expressed as the difference in the log observed load and log unconditioned (simulated) predicted load. See section 4.6.3 for details on conditioned and unconditioned model predictions.

5.2.4.15 (run_id)_vANOVA.list (R binary object)

The object is output for an estimated model (`if_estimate <- "yes"`) when validation sites are selected (`if_validate <- "yes"`), and contains the following model performance-related variables for the validation sites:

mobs = Number of monitoring stations used in the model calibration.

pSSE = Sum of Squares of Error, calculated as the sum of the squares of the weighted residuals, based on use of the unconditioned predictions.

pMSE = Mean Sum of Squares of Error, computed as the quotient of the *pSSE* and *mobs*.

pRMSE = Root Mean Sum of Squares of Error, computed as the square root of the *pMSE*.

pRSQ = R-Squared, computed as the ratio of the sum of the model residuals (based on the unconditioned predictions) to the total variance in the log transformed observed loads.

pRSQ_ADJ = Adjusted R-Squared (based on *pRSQ*), expressed as the *pRSQ* adjusted for the number of degrees of freedom.

pRSQ_YLD = Yield R-Squared, expressed as the *pRSQ* adjusted for the mean log drainage area.

pNSeff = Nash-Sutcliffe model efficiency coefficient, based on the use of the unconditioned predictions.

pPBisas = Percent Bias, computed as the ratio of the sum of the model residuals (observed load - unconditioned predicted load) to the sum of the observed load across all calibration sites, and multipled by 100 (see Moriasi et al., 2007).

5.2.5 Moran's I test for spatial autocorrelation

The Moran's I test provides a statistical measure of extent of clustering, dispersion, or random organization (see Fig. 9) associated with the SPARROW model residuals (predicted minus observed load values) at the monitoring calibration stations. Test results are reported for residuals weighted according to Euclidian and hydrological distances between the calibration stations.

Results from the application of the Moran's I test (executed by the `if_spatialAutoCorr<-"yes"` setting) are reported in the two files described below (see Chapter sub-section 4.4.5.1 for additional details).

5.2.5.1 (run_id)_diagnostics_spatialautocor.html

The following plots are displayed in the file:

- CDF (cumulative distribution function) of station stream distances (units of 'length' variable)
- CDF of station Euclidean distances (kilometers)
- Four panel plot with Moran's I results by river basin, reported for the most downstream site in the basin, including the p-value for Euclidean-distance weights, standard deviate for Euclidean-distance weights, the p-value for hydrologic-distance weights, and the standard deviate for hydrologic-distance weights.
- Two panel plot with Moran's I results reported for the full domain and for the spatial classes of the variable specified for the first element of the control setting `classvar`; the reported metrics include the for Euclidean-distance weights, and the standard deviate with Euclidean-distance weights.

Note that plotly interactive plots are automatically output to the HTML file (see section 8 of the *sparrow_control.R* script and Chapter subsection 4.4.8.4 for details). Each interactive plot displays an R *"plotly"* tool box banner, located in the upper right portion of each plot, that allows users to interactively control various features of each image. The interactive tool box provides users with a standard set of control options for the display image, including the ability to pan/zoom, box select areas of the image, rescale plot axes, download 'png' files, and to display the data associated with plot features by hovering the mouse cursor over these features (i.e., "hover-text"). The "hover-text" feature allows users to display the data associated with individual data points in plots.

5.2.5.2 (run_id)_diagnostics_spatialautocor.txt

The text file reports numerical results associated with the graphical Moran's I output to the *(run_id)_diagnostics_spatialautocor.html* file. This includes Moran's I results for the following:

- Results by river basin, reported for the most downstream site in each basin with five or more sites. The p-value and standard deviate metrics are reported separately for tests using Euclidean and hydrologic-distance weights.
- Results for the full domain, using hydrologic-distance weights
- Results for the regions and full domain, using Euclidean-distance weights. Regions are based on the user-specified contiguous areas as for the variable specifed for the first element of the `classvar` setting in section 5 of the control script.

5.2.6 Parameter estimation uncertainties:

5.2.6.1 (run_id)_bootbetaest.csv

This file is output by the execution of the parametric bootstrap uncertainty estimation (see Chapter sub-section 4.4.10) using the control setting `if_boot_estimate <- "yes"` (section 10 of the control script). The file contains the following elements:

biters = the number of Monte Carlo iterations, corresponding to the number of records in the file.

(sparrowNames) = a sequence of columns with the mean coefficient value for the explanatory variables for each of the *biters* iterations. The names of the columns correspond to their *sparrowNames* for the parameters selected from the *parameters.csv* file.

bootmean_exp_weighted_error = the mean exponentiated error (bias correction log retransformation factor) for each iteration.

5.2.6.2 (run_id)_BootResults (R binary object)

The file is created as a data archive using the control setting `if_boot_estimate <- "yes"` (section 10 of the control script) for use in subsequent execution of parametric bootstrap predictions. The contents include:

bEstimate = an IxP matrix consisting of the mean coefficient value for each of P columns associated with the model coefficients and I rows associated with the *biters* iterations. The names of the columns correspond to their *sparrowNames* for the parameters selected from the *parameters.csv* file, and listed in the *Parmnames* variable in the *JacobResults* object.

bootmean_exp_weighted_error = the mean exponentiated error (bias correction log retransformation factor) for each iteration, stored as a vector of length *biters*.

boot_resids = an IxN matrix consisting of the model log residual, expressed as the difference in the log observed load and log predicted load, weighted by the leverage (computed as the reciprocal square root of one minus the leverage), where I rows are associated with the *biters* iterations and N columns are associated with the calibration monitoring sites.

boot_lev = an IxN matrix consisting of the leverage statistic, defined according to equation 1.62 (Schwarz et al., 2006), where I rows are associated with the *biters* iterations and N columns are associated with the calibration monitoring sites.

5.3 Model predictions directory: (run_id)/predict

5.3.1 Standard predictions

The standard predictions as described in Chapter sub-section 4.4.7 are output to the two CSV files described below. Each file contains the following standard list of *sparrowNames* from the *dataDictionary.csv* file (see Table 4): **waterid**, **originalWaterid**, **rchname**, **rchtype**, **headflag**, **termflag**, **demtarea**, **demiarea**, **meanq**, **fnode**, **tnode**, **hydseq**, **frac**, **iftran**, **staid**.

The **originalWaterid** is the value of the user's original *waterid* variable in the *dataDictionary.csv* in cases where the reach IDs and nodes are renumbered (i.e., where the integer magnitude is greater than 1.0e+06). This variable may be useful as a common identification attribute to digitally link to other user data files (e.g., ESRI).

Users may also specify additional *sparrowNames* variables from the *dataDictionary.csv* to include in these two CSV output files by listing the variables in the control setting (section 7 of the control script) `add_vars<-c("sparrowName1","sparrowName2",...)`.

5.3.1.1 (run_id)_predicts_load_csv

The load predictions include the following variables, with the naming syntax as shown in the excerpt from the control script section 7 below. Note that the RSPARROW predictions of the mean load are adjusted for

log-retransformation bias by multiplying the predictions by the log-retransformation correction bias factor (with the exception of the non-mass type predictions of the delivery factor, *deliv_frac*). The bias-correction factor is reported in the *(run_id)_summary.txt* file for each model as the `Mean Exp Weighted Error`.

```
# Load prediction names and explanations
#   pload_total                Total load (fully decayed)
#   pload_(sources)            Total source load (fully decayed)
#   mpload_total               Monitoring-adjusted (conditional) total load
#                                 (fully decayed)
#   mpload_(sources)           Monitoring-adjusted (conditional) total source load
#                                 (fully decayed)
#   pload_nd_total             Total load delivered to streams (no stream decay)
#   pload_nd_(sources)         Total source load delivered to streams (no stream decay)
#   pload_inc                  Total incremental load delivered to reach
#                                 (with 1/2 of reach decay)
#   pload_inc_(sources)        Source incremental load delivered to reach
#                                 (with 1/2 of reach decay)
#   deliv_frac                 Fraction of total load delivered to terminal reach
#   pload_inc_deliv            Total incremental load delivered to terminal reach
#   pload_inc_(sources)_deliv  Total incremental source load delivered to terminal reach
#   share_total_(sources)      Source shares for total load (percent)
#   share_inc_(sources)        Source shares for incremental load (percent)
```

5.3.1.2 (run_id)_predicts_load_units.csv

The names and units of the load predictions are output with the following three column labels:

Prediction Metric Name = the load prediction variable short name, shown in the previous section 5.3.1.1, and stored in the *(run_id)_predict.list* binary object variable *oparmlist* as described in section 5.3.1.5.

Units = the load prediction variable units as stored in the *(run_id)_predict.list* binary object variable *loadunits* as described in section 5.3.1.5.

Metric Explanation = the load prediction variable long explanatory name, shown in the previous section 5.3.1.1, and stored in the *(run_id)_predict.list* binary object variable *oparmlistExpl* as described in section 5.3.1.5.

5.3.1.3 (run_id)_predicts_yield.csv

The yield predictions include the following variables, with the naming syntax as shown in the excerpt from the control script section 7 below. Note that the RSPARROW predictions of the mean yield are adjusted for log-retransformation bias by multiplying the predictions by the log-retransformation correction bias factor. The bias-correction factor is reported in the *(run_id)_summary.txt* file for each model as the `Mean Exp Weighted Error`.

```
# Yield prediction names and explanations
#   concentration              Flow-weighted concentration based on decayed total load
#                                 and mean discharge
#   yield_total                Total yield (fully decayed)
#   yield_(sources)            Total source yield (fully decayed)
#   myield_total               Monitoring-adjusted (conditional) total yield
#                                 (fully decayed)
#   myield_(sources)           Monitoring-adjusted (conditional) total source yield
#                                 (fully decayed)
#   yield_inc                  Total incremental yield delivered to reach
#                                 (with 1/2 of reach decay)
#   yield_inc_(sources)        Total incremental source yield delivered to reach
```

```
#                                    (with 1/2 of reach decay)
#   yield_inc_deliv              Total incremental yield delivered to terminal reach
#   yield_inc_(sources)_deliv    Total incremental source yield delivered to
#                                    terminal reach
```

5.3.1.4 (run_id)_predicts_yield_units.csv

The names and units of the yield predictions are output with the following three column labels:

Prediction Metric Name = the yield prediction variable short name, shown in the previous section 5.3.1.3, and stored in the *(run_id)_predict.list* binary object variable *oyieldlist* as described in section 5.3.1.5.

Units = the yield prediction variable units as stored in the *(run_id)_predict.list* binary object variable *yieldunits* as described in section 5.3.1.5.

Metric Explanation = the yield prediction variable long explanatory name, shown in the previous section 5.3.1.3, and stored in the *(run_id)_predict.list* binary object variable *oyieldlistExpl* as described in section 5.3.1.5.

5.3.1.5 (run_id)_predict.list (R binary object)

The file is created as an archive with all load and yield prediction variables (from Chapter sub-sections 5.2.1.1 to 5.2.1.4) to provide for the efficient access and use of predictions in subsequent execution of the parametric bootstrap predictions and uncertainties, mapping, and scenario evaluations. Note that the RSPARROW predictions of the mean yield are adjusted for log-retransformation bias by multiplying the predictions by the log-retransformation correction bias factor. The bias-correction factor is reported in the *(run_id)_summary.txt* file for each model as the `Mean Exp Weighted Error`.

The object contents include:

oparmlist = a character vector with the names of the variables associated with the model load predictions.

loadunits = a character vector with the units of the load predictions corresponding to the variable names given in the *oparmlist* vector.

predmatrix = a NxR matrix with the values of the load predictions, where N is the number of reaches and R is the number of the load prediction variables corresponding to the names given in the *oparmlist* vector.

oyieldlist = a character vector with the names of the variables associated with the model yield predictions.

yieldunits = a character vector with the units of the yield predictions corresponding to the variable names given in the *oyieldlist* vector.

yldmatrix = a NxR matrix with the values of the yield predictions, where N is the number of reaches and R is the number of the yield prediction variables corresponding to the names given in the *oyieldlist* vector.

predict.source.list = a list containing 9 vector elements, consisting of the variable names associated with the loads and yields for sources.

oparmlistExpl = a character vector with the explanation of the variable names for the load predictions.

oyieldlistExpl = a character vector with the explanation of the variable names for the yield predictions.

mpload_decay = a vector (length=number of reaches) with values of the monitoring site total load subjected to aquatic decay.

mpload_fraction = a vector (length=number of reaches) of the ratio of the decayed monitored load (*mpload_decay*) to the conditional (monitoring-adjusted) model predictions of the total load that are fully decayed (*mpload_total*).

5.3.2 Bootstrap bias-corrected mean model predictions and uncertainties

The prediction mean estimates and uncertainties as described in Chapter sub-section 4.4.10 are output to the two CSV files described below. Each file contains the following standard list of *sparrowNames* from the *dataDictionary.csv* file (see Table 4): **waterid**, **originalWaterid**, **rchname**, **rchtype**, **headflag**, **termflag**, **demtarea**, **demiarea**, **meanq**, **fnode**, **tnode**, **hydseq**, **frac**, **iftran**, **staid**.

Users may also specify additional *sparrowNames* variables from the *dataDictionary.csv* to include in these two CSV output files by naming the variables in the following control setting: `add_vars<-c("sparrowName1","sparrowName2",...)`.

5.3.2.1 (run_id)_predicts_load_boots.csv

The mean load predictions include the variables and naming syntax as shown in the excerpt from the control script section 10 below. The uncertainties are reported for each variable according to the standard error and the lower and upper bounds for the 90 percent confidence intervals, using the prefixes of **se_**, **ci_lo_**, and **ci_hi_**, respectively.

Note that the estimates of the bootstrap mean are bias corrected individually for each mass-based prediction metric and reach location, without any restrictions on the network-wide mass balance. Thus, the mean estimates do not retain a spatial mass balance within the river network (see Chapter sub-section 4.4.10.4 for details).

```
# Bootstrap load prediction names and explanations
#   mean_pload_total                Bias-adjusted total load (fully decayed)
#   mean_pload_(sources)            Bias-adjusted total source load (fully decayed)
#   mean_mpload_total               Bias-adjusted conditional (monitoring-adjusted)
#                                      total load (fully decayed)
#   mean_mpload_(sources)           Bias-adjusted conditional (monitoring-adjusted)
#                                      toal source load (fully decayed)
#   mean_pload_nd_total             Bias-adjusted total load delivered to streams
#                                      (no stream decay)
#   mean_pload_nd_(sources)         Bias-adjusted total source load delivered to streams
#                                      (no stream decay)
#   mean_pload_inc                  Bias-adjusted total incremental load delivered
#                                      to reach (with 1/2 of reach decay)
#   mean_pload_inc_(sources)        Bias-adjusted total source incremental load
#                                      delivered to reach (with 1/2 of reach decay)
#   mean_deliv_frac                 Fraction of total load delivered to terminal reach
#   mean_pload_inc_deliv            Bias-adjusted total incremental load delivered to
#                                      terminal reach
#   mean_pload_inc_(sources)_deliv  Bias-adjusted total incremental source load
#                                      delivered to terminal reach
#   mean_share_total_(sources)      Bias-adjusted percent source shares for total load
#   mean_share_inc_(sources)        Bias-adjusted percent source shares for
#                                      incremental load
```

5.3.2.2 (run_id)_predicts_yield_boots.csv

The mean yield predictions include the variables and naming syntax as shown in the excerpt from the control script section 10 below. The uncertainties are reported for each variable according to the standard error and the lower and upper bounds for the 90 percent confidence intervals, using the prefixes of **se_**, **ci_lo_**, and **ci_hi_**, respectively.

Note that the estimates of the bootstrap mean are bias corrected individually for each mass-based prediction metric and reach location, without any restrictions on the network-wide mass balance. Thus, the mean

estimates do not retain a spatial mass balance within the river network (see Chapter sub-section 4.4.10.4 for details).

```
# Bootstrap yield prediction names and explanations
#   mean_conc_total                 Bias-adjusted concentration based on decayed total
#                                      load and mean discharge
#   mean_yield_total                Bias-adjusted total yield (fully decayed)
#   mean_yield_(sources)            Bias-adjusted total source yield (fully decayed)
#   mean_myield_total               Bias-adjusted conditional (monitoring-adjusted)
#                                      total lyield (fully decayed)
#   mean_myield_(sources)           Bias-adjusted conditional (monitoring-adjusted)
#                                      total source yield (fully decayed)
#   mean_yield_inc                  Bias-adjusted total incremental yield delivered to
#                                      reach (with 1/2 of reach decay)
#   mean_yield_inc_(sources)        Bias-adjusted total incremental source yield
#                                      delivered to reach (with 1/2 of reach decay)
#   mean_yield_inc_deliv            Bias-adjusted total incremental yield delivered to
#                                      terminal reach
#   mean_yield_inc_(sources)_deliv  Bias-adjusted total incremental source yield
#                                      delivered to terminal reach
```

5.3.2.3 (run_id)_predictBoots.list (R binary object)

The file is created using the control setting `if_boot_predict <- "yes"` (section 10 of the control script) and contains parametric bootstrap predictions for load and yield (see Chapter sub-section 4.4.10 for details on the methods and prediction metrics). Note that the estimates of the bootstrap mean are bias corrected individually for each mass-based prediction metric and reach location, without any restrictions on the network-wide mass balance. Thus, the mean estimates do not retain a spatial mass balance within the river network.

The object contents include:

boparmlist = a character vector with the names of the variables associated with the model load predictions. The base variable names are shown above in Chapter sub-section 5.3.2.1 and include for each load prediction metric the standard error and the lower and upper bounds for the 90 percent confidence intervals, using the prefixes of **se_**, **ci_lo_**, and **ci_hi_**, respectively.

bootmatrix = a NxR matrix with the values of the load prediction bias-corrected mean values and related uncertainties, where N is the number of reaches and R is the number of the load prediction variables corresponding to the names given in the *boparmlist* vector.

byldoparmlist = a character vector with the names of the variables associated with the model yield predictions. The base variable names are shown above in Chapter sub-section 5.3.2.2 and include for each yield prediction metric the standard error and the lower and upper bounds for the 90 percent confidence intervals, using the prefixes of **se_**, **ci_lo_**, and **ci_hi_**, respectively.

bootyldmatrix = a NxR matrix with the values of the yield prediction bias-corrected mean values and related uncertainties, where N is the number of reaches and R is the number of the yield prediction variables corresponding to the names given in the *byldoparmlist* vector.

error.sample = a N x *biter* matrix containing the randomly sampled exponentiated NLLS leverage-weighted logged model errors (*JacobResults$boot_resid*), where N is the number of reaches and *biter* is the number of bootstrap iterations. The residual are used to compute the standard error sample error variance and the confidence interval error variation.

5.3.2.4 (run_id)_BootUncertainties (R binary object)

The file is created using the control setting `if_boot_predict <- "yes"` (section 10 of the control script) and contains uncertainty estimates, expressed as a percentage of the mean load. The file serves as an archive for use in subsequent execution of the mapping functions. See section 4.4.10 for a discussion of the uncertainty estimation methods.

The contents includes:

se_pload_total = Standard error of the unconditioned total load (percentage of mean)

ci_pload_total = 95% prediction interval of the unconditioned total load (percentage of mean)

se_mpload_total = Standard error of the conditioned (monitoring-adjusted) total load (percentage of mean)

ci_mpload_total = 95% prediction interval of the conditioned (monitoring-adjusted) total load (percentage of mean)

model.error.var = the model error variance, computed as the ratio of the variance of the model log residual (*JacobResults$boot_resid*) to the squared mean exponentiated log residual from the NLLS model (per equation 1.144; Schwarz et al., 2006).

sample.error.var.boots = a vector of the model error variance for total load for each reach (length=number of reaches)

5.4 Mapping directory: (run_id)/maps

The maps in the following files are controlled by user settings in sections 8 and 9 of the control script (also see Chapter sub-section 4.4.8 and 4.4.9 for details).

The `enable_plotlyMaps<-"yes"` setting (see section 8 of the *sparrow_control.R* script and Chapter sub-section 4.4.8.4 for details) outputs *plotly* maps with interactive features (i.e., browser-accessible HTML files) for the stream and catchment predictions and *dataDictionary* variables (section 4.4.8.2). **Note that the setting `enable_plotlyMaps<-"yes"` produces HTML files that are very large in size; users should expect some delays when opening the HTML file content in a browser.** The setting `enable_plotlyMaps<-"no"` outputs non-interactive maps in PDF files.

Users should note that if a large number of variables are specified for output, especially for a model with a large spatial domain (e.g., more than 300,000 reaches), the execution time for map production will be lengthy. We advise using the R Shiny interactive mapper (`enable_shinyMaps<-"yes"` in section 9 of the control script) to output multiple variables in batch mode. This will produce one map per file, which will allow more efficient opening of the output file. Additionally, note that stream maps require much less time to produce than catchment maps.

Each interactive map in the HTML file displays an R *"plotly"* tool box banner, located in the upper right portion of each plot or map (e.g., Fig. 12), that allows users to interactively control various features of each image. The interactive tool box provides users with a standard set of control options for the display image, including the ability to pan/zoom, box select areas of the image, rescale plot axes, sub-select features from the map legend for display, download 'png' files, and to display the data associated with plot and map features by hovering the mouse cursor over these features (i.e., "hover-text"). The "hover-text" feature allows users to display the data associated with individual data points in plots and the monitoring sites and stream segments in maps.

5.4.1 Mapping sub-directory: (run_id)/maps/Stream

The stream reach maps contain values of the prediction metrics or variables from the *dataDictionary* that are listed in the control setting `master_map_list` in section 8 of the control script. A separate file is generated for each variable, with the name of the variable embeded in the file name: *(run_id)_(variable_name).html* or *(run_id)_(variable_name).pdf*. The `enable_plotlyMaps` setting (Chapter sub-section 4.4.8.4) controls

the generation of either an interactive HTML ("yes") file or a non-interactive PDF ("no") output file. The object binary file *batch_(time).RData* is also created if the maps are saved in batch mode.

5.4.2 Mapping sub-directory: (run_id)/maps/Catchment

The catchment reach maps contain values of the prediction metrics or variables from the *dataDictionary* that are listed in the control setting `master_map_list` in section 8 of the control script. A separate file is generated for each variable, with the name of the variable embeded in the file name: *(run_id)_(variable_name).html* or *(run_id)_(variable_name).pdf*. The `enable_plotlyMaps` setting (section 8) controls the generation of either an interactive HTML ("yes") file or a non-interactive PDF ("no") output file. The object binary file *batch_(time).RData* is also created if the maps are saved in batch mode.

5.4.3 Mapping sub-directory: (run_id)/maps/Interactive

This directory is created if maps are saved (using the *"Save Map"* button) from an R Shiny interactive DSS mapping session (i.e., `enable_ShinyApp<-"yes"`; section 9 of the control script; see Chapter sub-section 4.4.9 for details).

There are three sub-directories created with the names "Stream", "Catchment", and "SiteAttributes", based on the types of maps (*"Map Type"*) that are output, depending on the selected map type.

The output format is controlled by the *"Output Map Format"* setting in the R Shiny mapper interface, with a PDF file output for the *"static"* option (with non-interactive features) and a HTML file output (with interactive features) for the *"plotly"* and *"leaflet"* options. The *"plotly"* map option is not available for the display of catchment metrics because of the lengthy rendering time.

The naming convention for the output files is *(run_id)_(variable_name).pdf* and *(run_id)_(variable_name).html*.

5.4.4 Mapping sub-directory: (run_id)/maps/ESRI_ShapeFiles

There are as many as three sub-directories created, depending on the user selections for the control setting `outputESRImaps` (section 8 of the control script), with the names "prediction", "residuals", and "siteAttributes".

In the "prediction" sub-directory, the following ESRI files are created: "lineShape.(dbf,prj,shp,shx)"", and"polyShape.(dbf,prj,shp,shx)", corresponding to stream and catchment maps respectively.

In the "residuals" sub-directory, the files "residShape.(dbf,prj,shp,shx)" are created for the calibration monitoring sites and populated with data from the *residuals.csv* file.

In the "siteAttributes" sub-direcctory, the files "siteAttrshape.(dbf,prj,shp,shx)" are created for the calibration monitoring sites and populated with variables selected using the `mapSiteAttributes` setting.

The common identification variable in these files includes both the *waterid* and the **originalWaterid**. The *originalWaterid* is the value of the user's original *waterid* variable in the *dataDictionary.csv* in cases where the reach IDs and nodes are renumbered (i.e., where the integer magnitude is greater than 1.0e+06). This variable may be useful as a common identification attribute to digitally link to other user data files (e.g., ESRI).

5.5 Source-change management scenarios directory: (run_id)/scenarios

The following files are output to each subdirectory for the user-specified scenario name as given in the *scenario_name* control setting or specified in the R Shiny interface. See Chapter sub-section 4.4.9 for details on the control settings given in section 9 of the control script.

Note that the files include both the *waterid* and the **originalWaterid** common identification variables. The *originalWaterid* is the value of the user's original *waterid* variable in the *dataDictionary.csv* in cases where the reach IDs and nodes are renumbered (i.e., where the integer magnitude is greater than 1.0e+06). This

variable may be useful as a common identification attribute to digitally link to other user data files (e.g., ESRI).

5.5.1 (scenario_name)_(run_id)_(variable_name).pdf (.html)

A separate stream and/or catchment map is output for each of the prediction metrics (absolute and/or relative) listed in the control setting `scenario_map_list`. Stream maps are output according to the `output_map_type` setting and stored in separate sub-directories ("Stream", "Catchment") according to the specified type. A separate file is generated for each variable, with the name of the variable embeded in the file name: *(scenario_name)_(run_id)_(variable_name).html* or *(scenario_name)_(run_id)_(variable_name).pdf*. The `enable_plotlyMaps` setting (Chapter subsection 4.4.8.4) controls the generation of either a HTML file with interactive features ("yes") or a PDF file with non-interactive or static features ("no").

These HTML and PDF files are also output from the R Shiny interactive DSS mapper (i.e., `enable_ShinyApp<-"yes"`) when the maps are saved using the *"Save Map"* button (see section 9 of the control script and Chapter sub-section 4.4.9 for details). The files are written to sub-directories ("Stream" and/or "Catchment") within the scenario directory that is named by the user (i.e., entry for the *"Enter Scenario Name"* option).

5.5.2 (scenario_name)_(run_id)_scenario_metainfo.txt

The text file records the values for the following control settings for the user scenario: `select_scenarioReachAreas`, `select_targetReachWatersheds`, and `LanduseConversion`. Additionally, the following settings for the R Shiny session for the scenario are recorded: *Source* and *PercentChange*. Reaches with unadjusted prediction metrics are shown in the maps by the category "No Change" and assigned a ratio of 1.0.

An example of the contents of a metainfo text file is shown below.

```
  select_scenarioReachAreas select_targetReachWatersheds
1               all reaches                        import

   Source PercentChange LanduseConversion
1 pasture           -20         sugarcane
```

5.5.3 (scenario_name)_(run_id)_predicts_load_scenario.csv

Absolute load prediction metrics: The file contains the standard list of *sparrowNames* described in section 5.3.1 above, and includes similar load prediction metrics as shown in section 5.3.1.1 above.

The load predictions include the following variables, with the naming syntax as shown in the excerpt from the control script section 7 below. These predictions are corrected for log-retransformation bias as is done for the standard predictions (without changed sources).

Note that the conditioned (monitoring-adjusted) predictions are excluded from the output because the scenarios can only be evaluated using simulated (i.e., unconditioned) model predictions that preserve a spatial mass balance in the loads. Also, the model prediction of the delivery fraction is unaffected by a change scenario and thus excluded from the output.

```
# Load prediction names and explanations
#   pload_total                Total load (fully decayed)
#   pload_(sources)            Total source load (fully decayed)
#   pload_nd_total             Total load delivered to streams (no stream decay)
#   pload_nd_(sources)         Total source load delivered to streams (no stream decay)
#   pload_inc                  Total incremental load delivered to reach
#                                 (with 1/2 of reach decay)
#   pload_inc_(sources)        Source incremental load delivered to reach
#                                 (with 1/2 of reach decay)
#   pload_inc_deliv            Total incremental load delivered to terminal reach
```

```
#   pload_inc_(sources)_deliv  Total incremental source load delivered to terminal reach
#   share_total_(sources)      Source shares for total load (percent)
#   share_inc_(sources)        Source shares for incremental load (percent)
```

5.5.4 (scenario_name)_(run_id)_predicts_load_scenario_units.csv

The names and units of the load predictions are output with the following two column labels:

Prediction Metric Name = the load prediction variable short name, shown in the previous section 5.5.3, and stored in the *(run_id)_predictScenarios.list* binary object variable *oparmlist* as described in section 5.5.9.

Units = the load prediction variable units as stored in the *(run_id)_predictScenarios.list* binary object variable *loadunits* as described in section 5.5.9.

5.5.5 (scenario_name)_(run_id)_predicts_loadchg_scenario.csv

Relative load change prediction metrics: The file contains the standard list of *sparrowNames* described in section 5.2.1 above, and includes the identical load prediction metrics as shown in section 5.5.3 above, with the values expressed as a ratio of the updated prediction (changed according to the scenario) to the baseline (unchanged) prediction metric.

A `NA` is reported for land-use source loads in cases where the ratio of the changed load to the baseline (unchanged) load is equal to infinity–i.e., where the baseline load is equal to zero. These cases reflect reaches where the converted land use for the scenario has an area equal to zero (and zero contaminant load) prior to the application of the scenario.

5.5.6 (scenario_name)_(run_id)_predicts_yield_scenario.csv

Absolute yield prediction metrics: The file contains the standard list of *sparrowNames* described in section 5.3.1 above, and includes similar yield prediction metrics as shown in section 5.3.1.3 above.

The yield predictions include the following variables, with the naming syntax as shown in the excerpt from the control script section 7 below. These predictions are corrected for log-retransformation bias as is done for the standard predictions (without changed sources).

Note that the conditioned (monitoring-adjusted) predictions are excluded from the output because the scenarios can only be evaluated using simulated (i.e., unconditioned) model predictions that preserve a spatial mass balance in the loads.

```
# Yield prediction names and explanations
#   concentration              Flow-weighted concentration based on decayed total load
#                                 and mean discharge
#   yield_total                Total yield (fully decayed)
#   yield_(sources)            Total source yield (fully decayed)
#   yield_inc                  Total incremental yield delivered to reach
#                                 (with 1/2 of reach decay)
#   yield_inc_(sources)        Total incremental source yield delivered to reach
#                                 (with 1/2 of reach decay)
#   yield_inc_deliv            Total incremental yield delivered to terminal reach
#   yield_inc_(sources)_deliv  Total incremental source yield delivered to
#                                 terminal reach
```

5.5.7 (scenario_name)_(run_id)_predicts_yield_scenario_units.csv

The names and units of the load predictions are output with the following two column labels:

Prediction Metric Name = the load prediction variable short name, shown in the previous section 5.5.6, and stored in the *(run_id)_predictScenarios.list* binary object variable *oyieldlist* as described in section 5.5.9.

Units = the load prediction variable units as stored in the *(run_id)_predictScenarios.list* binary object variable *yieldunits* as described in section 5.5.9.

5.5.8 (scenario_name)_(run_id)_predicts_yieldchg_scenario.csv

Relative yield change prediction metrics: The file contains the standard list of *sparrowNames* described in section 5.2.1 above, and includes the identical yield prediction metrics as shown in section 5.5.6 above, with the values expressed as a ratio of the updated prediction (changed according to the scenario) to the baseline (unchanged) prediction metric.

A `NA` is reported for land-use source yields in cases where the ratio of the changed yield to the baseline (unchanged) yield is equal to infinity–i.e., where the baseline yield is equal to zero. These cases reflect reaches where the converted land use for the scenario has an area equal to zero (and zero contaminant yield) prior to the application of the scenario.

5.5.9 (scenario_name)_(run_id)_predictScenarios.list (R binary object)

The file is created as an archive with key scenario control settings and the load and yield prediction variables (from Chapter sub-sections 5.5.3 and 5.5.6) that are output from the execution of a source-change scenario evaluation. The object is output to the subdirectory for the user-specified scenario name as given in the *scenario_name* control setting or specified by the user in the R Shiny interface. The absolute prediction metrics are corrected for log-retransformation bias as is done for the standard predictions (without changed sources).

The content of the object includes:

select_scenarioReachAreas = a scenario control setting that identifies whether the scenario feature is to be disabled ("none") or whether scenarios are to be applied to "all reaches" or "selected reaches" above user-defined watershed outlets.

select_targetReachWatersheds = a scenario control setting that defines the watershed outlets (i.e., targeted reach locations) that are used to identify the hydrologically-connected upstream reaches in the watersheds where the source-change scenarios are applied.

scenario_name = a scenario control setting that specifies the user-selected name for an evaluated scenario.

scenario_map_list = a scenario control setting that specifies the type of predictions to map for stream reaches and output to a PDF file in batch mode.

scenario_sources = a scenario control setting that identifies one or more source variables in the SPARROW model that are to be evaluated in the management source-change scenario.

scenario_factors = a scenario control setting that specifies the factors (i.e., multipliers) by which each source is either reduced or increased according to the user's source-change scenario.

landuseConversion = a scenario control setting that specifies a land-use source in the SPARROW model to which a complimentary area conversion is applied that is equal to the change in area of the *scenario_sources* variable.

oparmlist = a character vector with the names of the variables associated with the model load predictions for management scenarios as described in Chapter sub-section 5.5.3.

loadunits = a character vector with the units of the load predictions corresponding to the variable names given in the *oparmlist* vector.

predmatrix = a NxR matrix with the values of the load predictions, where N is the number of reaches and R is the number of the load prediction variables corresponding to the names given in the *oparmlist* vector.

oyieldlist = a character vector with the names of the variables associated with the model yield predictions for management scenarios as described in Chapter sub-section 5.5.6.

yieldunits = a character vector with the units of the yield predictions corresponding to the variable names given in the *oyieldlist* vector.

yldmatrix = a NxR matrix with the values of the yield predictions, where N is the number of reaches and R is the number of the yield prediction variables corresponding to the names given in the *oyieldlist* vector.

predict.source.list = a list containing 7 vector elements, consisting of the variable names associated with the loads and yields for sources.

predmatrix_chg = a NxR matrix containing the load prediction metrics as shown in section 5.5.3 above for N reaches and R prediction variables, with the values expressed as a ratio of the updated prediction (changed according to the scenario) to the baseline (unchanged) prediction metric.

yldmatrix_chg = a NxR matrix containing the yield prediction metrics as shown in section 5.5.6 above for N reaches and R prediction variables, with the values expressed as a ratio of the updated prediction (changed according to the scenario) to the baseline (unchanged) prediction metric.

scenarioFlag = an automatically derived reach-level identifier that flags the reaches where the scenario is applied, based on both the *select_targetReachWatersheds* and *select_scenarioReachAreas* control settings.

5.5.10 (scenario_name)_(run_id)_DataMatrixScenarios.list (R binary object)

The binary list object is output to the contains two data elements, with changed sources according to the user-specified management scenario, that serve as input to the simulation of the load and yield predictions in the *predictScenarios.R* function. The object is output to the subdirectory for the user-specified scenario name as given in the *scenario_name* control setting or specified by the user in the R Shiny interface. The content includes:

dataNames = the names of the FIXED and REQUIRED system variables and explanatory variables associated with the user-specified model.

data = a data matrix of reach values (rows=number of reaches, columns=number of variables), with the names of the column variables given in the **dataNames** variable.

5.6 Batch directory: (run_id)/batchSessionInfo

5.6.1 (run_id)_log.txt

The text file provides a record of the data import and model execution steps, the reported execution times for selected modeling procedures, and any error messages that occur.

5.6.2 (run_id).RData

An R binary datafile with the objects and variables produced during the batch execution of the RSPARROW model control settings and procedures.

5.7 Model comparison directory: (modelComparison_name)

The following files are created in a new directory with the name `results\modelComparison_name` in cases where a user selects to compare the results of the executed model with multiple previously executed models, as specified by the control settings for `compare_models` (see Chapter sub-section 4.4.11.4):

5.7.1 (modelComparison_name)_summary.txt

The text file list in sequence for the user-specified models the following metrics from the *(run_id)_summary.txt* file (see Chapter sub-section 5.2.4.2):

- **Model performance metrics**: Number of observations and parameters, degrees of freedom, Sum of Squares of Error, Mean Square Error, Root Mean Square Error, R-Squared Adjusted, R-Squared Yield, and Percent Bias

- **Parameter estimate metrics**: parameter name and type, mean estimate, standard error, t-statistics and p-values, VIF, and parameter description and parameter units
- **Model residuals metrics**: Eigenvalue Spread, Normal PPCC, Shipiro-Wilks W test statistic and p-value, and the mean exponentiated weighted error or Smearing Estimate bias-retransformation correction factor

5.7.2 (modelComparison_name)_ModelPerformanceMonitoringAdj.csv

The CSV file contains the **model performance metrics** as described above, with an additional column for the user-specified `compare_models`.

5.7.3 (modelComparison_name)_ParameterEstimates.csv

The CSV files associated with *(run_id)_summary.txt* file output are stored in the "(run_id)/estimate/summaryCSV"" sub-directory and combined in the model comparison. The combined CSV file contains the **parameter estimate metrics** as described above, with an additional column for the user-specified `compare_models`.

5.7.4 (modelComparison_name)_EigenValueSpread.csv

The CSV file contains the **model residuals metrics** as described above, with an additional column for the user-specified `compare_models`.

5.7.5 (modelComparison_name)_EuclideanMoransI.csv

The CSV file contains results from the spatial autocorrelation Moran's I test for the compared models.

6 Tutorial: Executing and interpreting a series of models that build in complexity

6.1 Introduction

This tutorial uses the Midwest total nitrogen model (Robertson and Saad, 2011) to illustrate the steps involved in developing a SPARROW model. We generally recommend that SPARROW model development begin with a relatively simple model and that users incrementally build and evaluate models with increasing complexity as illustrated in this chapter. Model complexity increases with the addition of explanatory variables and functional components (e.g., in-stream decay) as well as with the use of more nonlinear functional expressions. Using an incremental approach to develop model complexity (starting with very simple models) has the advantage of allowing users to systematically evaluate and understand the effect of additional complexity on the statistical performance and physical interpretability of the model.

Sub-section 6.2 demonstrates this incremental approach to model building. The tutorial presents statistical information on the model diagnostics to illustrate the effects of the additional explanatory variables on model performance and the interpretability of the models. Two of the tutorial models (3, 7) are also presented to illustrate the potential sensitivity of model results to the use of different initial values for explanatory variables that have lower levels of statistical significance (also see sub-section 4.4.4.4).

All of the control files and model output files are available for the eight models demonstrated in the tutorial. These model files are stored in the *"run_id"* sub-directories under the "UserTutorial/results" directory. The highlighted results (text, plots, maps) in this chapter can be viewed in the *(run_id)_diagnostic_plots.html* and *(run_id)_summary.txt* files in each of the "run_id" (model1 to model8) sub-directories.

Users can load the control settings for any of the eight models into the RStudio active RSPARROW control script (*sparrow_control.R* located in the "UserTutorial/results" directory) according to the following steps:

- Set the `copy_PriorModelFiles` setting to the model *"run_id"* of choice and enter the user's pathname for the `path_master` setting in the active control script (section 11 of the control script; Chapter sub-section 4.4.11.3).
- Execute the control script (i.e., select *Source* in RStudio). This will overwrite the active control script (and all input control files) in the *"results"* directory with the versions of these files from the prior model. Note that the user's pathname specified for the `path_master` setting in the active control script will be retained and is not overwritten.
- Once execution has completed and the prior model settings are copied into the active control script in the RStudio session, other model control settings can be kept or modified to perform various model applications, such as estimation, prediction, mapping, or evaluations of management scenarios in the R Shiny interactive mapper.

Sub-section 6.3 is supplementary information that illustrates the use of the tutorial data in the *data1.csv* file to automatically create the model input control files (*parameters.csv*, *design_matrix.csv*). A generic description of the approach is given in step 3 of the checklist in Chapter sub-section 4.2.4. The supplementary information in 6.2 is intended to demonstrate the types of Console messages that appear during the execution of these steps.

Details on the SPARROW total nitrogen training model can be found in Robertson and Saad (2011; a PDF of the published article is provided in the UserTutorial directory). The model was estimated using 708 calibration monitoring sites in watersheds of the midwestern United States, including USA drainage to the Great Lakes, based on a 1:500,000 scale representation of streams. The final model includes five nitrogen sources, five land-to-water delivery variables, two in-stream decay variables, and one reservoir decay variable.

The tutorial model includes only about 11,000 reaches, such that the model estimation and generation of predictions and maps execute relatively efficiently, allowing users to easily become familiar with RSPARROW control settings and capabilities.

6.2 Development of the models

6.2.1 Model 1: Incremental drainage area as the sole explanatory variable

An informative baseline model can be estimated by using the incremental drainage area as the only predictor variable in the model. All water-quality models, including SPARROW, can reliably accumulate area and chemical mass in a river network. The accumulation of mass typically provides a reasonable approximation of the total load, which is highly correlated with total drainage area. For this reason, using the reach incremental drainage area as an explanatory variable can serve as an initial baseline predictor of the total stream load. The model will likely display large prediction inaccuracies (i.e., high bias, low precision) in many streams and offers no interpretable information about the factors (sources, physical processes) that control spatial variability in stream loads. However, the model performance metrics of a drainage-area model provide a benchmark against which to evaluate the improved performance of more complex models.

To execute this model, *'parmInit'* was set to 500 kg/km2/yr (chosen arbitrarily) and *'parmMin'* and *'parmMax'* were set to 0 and a large value (10,000 kg/km2/yr), respectively, in the *parameters.csv* file. Note that if the control setting `edit_Parameters<-"yes"` is selected, then the file will appear as a "popup" on the screen; once entries are completed, users should save the file (*users should consult sub-section 4.4.4.3 for details on the selection of parameters and setting of the intial and minimum/maximum values for parameters*).

Note that the summary statistics for model performance and coefficient estimates as well as the plots and maps shown in this section for model 1 can be obtained from the *"model1_summary.txt"* and *"Model1_diagnostic_plots.html"* files in the "/results/Model1/estimate" sub-directory.

The observed total nitrogen loads at the 708 calibration sites, to which the model is fit, are shown in Fig. 16a. The loads span over five orders of magnitude and indicate the combined influence of watershed sources and streamflow (watershed size). High loads are observed across the corn-belt and also in larger rivers, such as the Upper Mississippi and Ohio. Total nitrogen yields (see Fig. 16b) are more indicative of watershed sources, and display large values in the corn-belt areas of the Midwest, with much more moderate to low values of yield in the Appalachians and Upper Midwest areas in the states of Wisconsin and Minnesota. These plots are available in the *TN_model1_diagnostic_plots.html* file in the "results/estimate" sub-directory, based on the use of the setting `map_siteAttributes.list<-c("meanload","meanyield")` in the control script (section 8). Both variables were defined in the *userModifyData.R* script.

RSPARROW reports model performance for both the conditioned (monitoring-adjusted loads) and unconditioned (simulation) predictions that are associated with estimating/calibrating the model and executing the model in simulation mode, respectively (see below). Comparing the peformance metrics for conditioned and unconditioned predictions provides additional information about the magnitude of the improved accuracy of the predictions of stream loads and yields that results from using the more accurate monitored-adjusted (conditioned) loads. These higher levels of accuracy (higher R-Squared and lower RMSE) for the drainage area model provide a lower limit against which the metrics associated with more complex models can be compared.

Below are the model performance metrics and estimated coefficient for the incremental drainage area, followed by diagnostic plots for model performance (Fig. 17), and maps of the observed yields and model residuals for the calibration sites (Fig. 18). The estimated coefficient for the model source variable, the incremental drainage area, is 869 kg/km2/year. This yield serves as a constant source input to streams in each of reach catchment.

```
 MODEL ESTIMATION PERFORMANCE (Monitoring-Adjusted Predictions)
MOBS NPARM  DF      SSE       MSE      RMSE       RSQ RSQ-ADJUST RSQ-YIELD PERCENT BIAS
 708     1 707 432.7155 0.6120446 0.7823328 0.8255356  0.8255356 0.4345638    0.5762475

 MODEL SIMULATION PERFORMANCE (Simulated Predictions)
MOBS NPARM  DF      SSE      MSE     RMSE       RSQ RSQ-ADJUST    RSQ-YIELD PERCENT BIAS
 708     1 707 769.3498 1.088189 1.043163 0.6898097  0.6898097 -0.005321489     15.67349

  Simulated predictions are computed using mean coefficients from the NLLS model
```

```
    that was estimated with monitoring-adjusted (conditioned) predictions

  PARAMETER SUMMARY
  PARAMETER ESTIMATE PARM TYPE EST TYPE  INITIAL VALUE MIN MAX    PARAMETER UNITS
1 demiarea  869.1212 SOURCE    Estimated 500           0   10000  kg/km2/year

  PARAMETER ESTIMATES
  PARAMETER PARM TYPE ESTIMATE SE(Jcb) T(Jcb) P-VALUE(Jcb) VIF PARAMETER UNITS
1 demiarea  SOURCE    869.1212 36.2157 23.998 0            1   kg/km2/year
```

The overall statistical fit of the drainage-area model initially appears to be generally acceptable, based on the conventional measures of model performance for the values of log load (RMSE=0.782; RSQ=0.825; PERCENT BIAS=0.57). Plots of the observed vs. predicted loads and the associated residuals for the loads (Fig. 17) confirm that the model fit seems generally unbiased although with notable scatter in the predictions. However, a closer look at the measures of model performance in simulation mode, using simulated predictions (i.e., unconditioned predictions which are not adjusted for monitoring load values), indicate that the skill of the model is actually quite poor. The RMSE (1.04) of the simulated load predictions is about 30 percent larger than the RMSE reported for the estimated model (which is based on the use of conditioned or monitoring-adjusted predictions). The R-squared value for the yield predictions in simulation mode (RSQ-YIELD) is also negative, indicating that the predictive capability of the model for yields is virtually zero. Thus, the larger reported predictive accuracy of the estimated drainage-area model stems from the use of conditioned predictions and use of the more accurate monitored loads as an upstream source to the model.

A closer look at performance metrics based on the yield predictions (mass per unit area) for the estimated model (monitoring-adjusted predictions) also indicate that the model fit to the observed data is generally quite marginal. The RSQ-YIELD is only 0.434, indicating that less than one-half of the spatial variability in nitrogen yield can be explained by the model. Appreciable spatial biases are also evident in the predicted values of yield (Figs. 16b, 18). For example, in streams with high observed values of yield–the corn belt of the Upper Mississippi (red symbols in Fig. 16b), the model consistently underpredicts (blue and green symbols in Fig. 18). By contrast, in streams with low observed values of yield (blue, yellow, and green symbols in Fig. 16b), the model consistenly overpredicts (red and yellow symbols in Fig. 18). This under- and over-prediction of yields can be seen in the plots of observed vs. predicted yield and the associated residuals (Fig. 17b,d).

The drainage-area model illustrates that the accumulation of mass in spatially explicit watershed models (SPARROW and other modes) can give predictions of total load that reasonably approximate the observed values of stream load (e.g., high R-squared values for load); however, this model has no capabilities to explain water-quality processes (e.g., `RSQ-YIELD`=-0.005 for the simulated unconditioned predictions).

Differences in the `ESTIMATION` (conditioned predictions) and `SIMULATION` (unconditioned predictions) performance metrics for the drainage area model (and any SPARROW model specification) provide an overall measure of the effects of the conditioned (monitoring-adjusted) predictions on model accuracy. Furthermore, the drainage-area model provides a lower limit on the accuracy that users can compare for both the `ESTIMATION` and `SIMULATION` metrics when assessing the improved performance of more complex models.

6.2.2 Model 2: Land-use source variables (four land-use types)

Land-use data are commonly available for most watersheds and can serve as surrogates of pollution sources in a SPARROW model, although a model with mass-based sources is generally preferred. A land-use based model is also an informative initial model to evaluate, especially given that water-quality exports are commonly reported in the literature for uniform land cover types based on field measurements and experimental catchment studies (e.g., Beaulac and Reckhow, 1982). These provide reference information to compare with the estimated SPARROW model coefficients (mass per unit area per time) for similar land-use types.

An initial land-use model is illustrated using the following land use types as source variables: `urban`, `crops`,

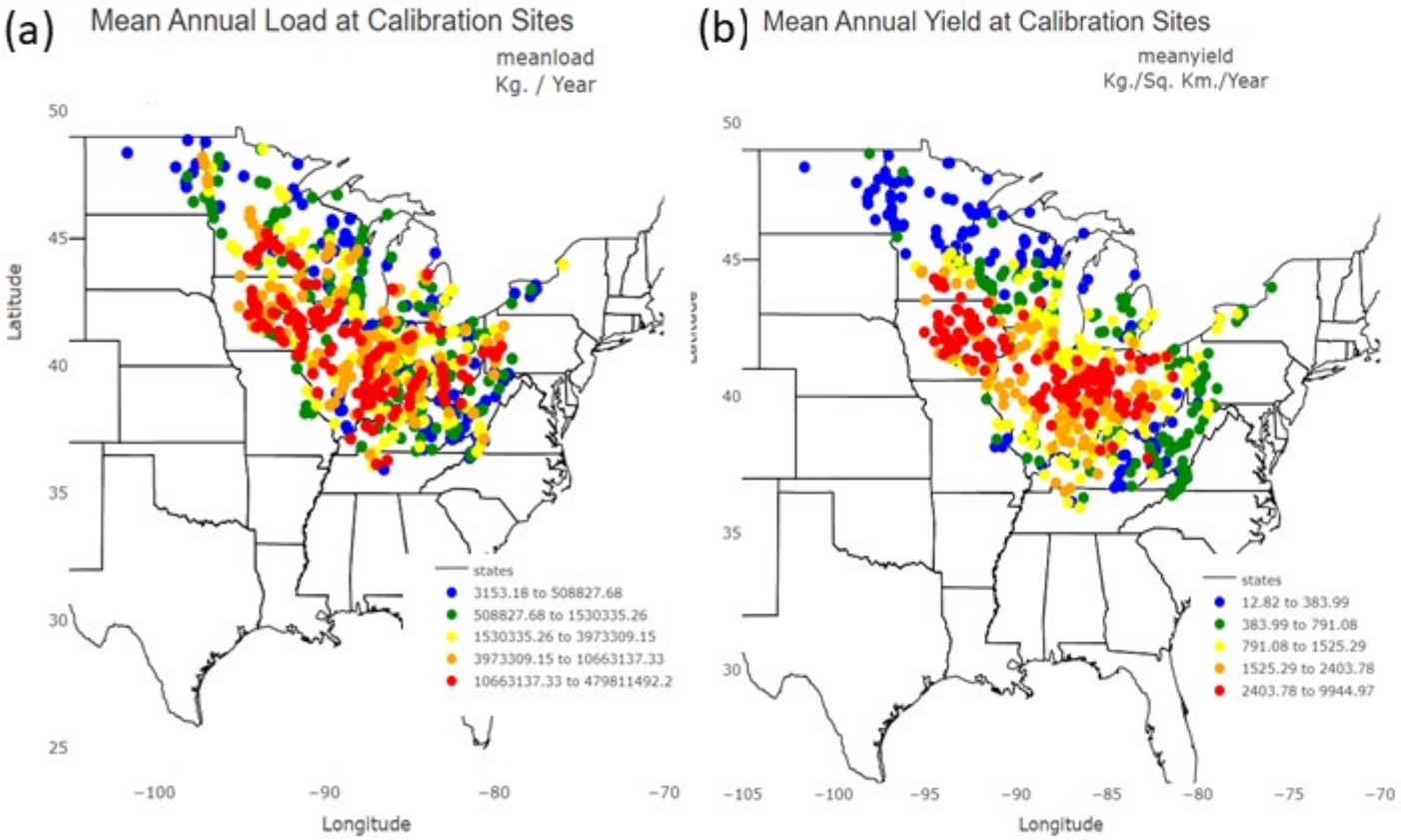


Figure 16: **Maps of the observed mean annual total nitrogen load and yield at the 708 calibration monitoring sites used to estimate the total nitrogen model 1.** (a) Mean annual total nitrogen load (kg/yr);(b) Mean annual total nitrogen yield (kg/km2/yr).

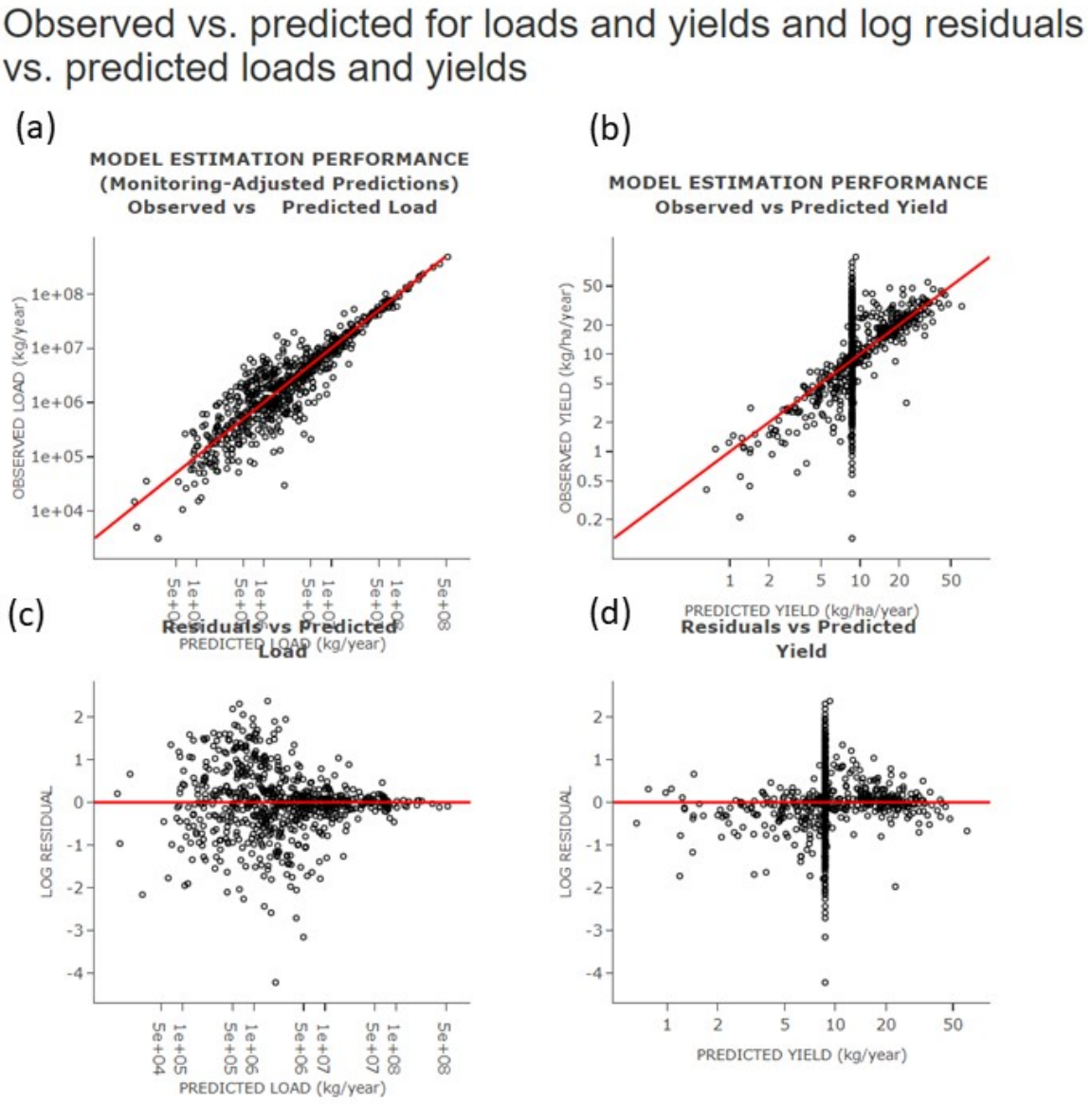


Figure 17: **Diagnostic plots for the RSPARROW total nitrogen model 1 with incremental drainage area as the only predictor variable.** Model performance plots for the monitoring-adjusted (conditioned) predictions are shown for: (a) observed load vs. predicted load; (b) observed yield vs. predicted yield; (c) log residuals vs. predicted load; and (d) log residuals vs. predicted yield.

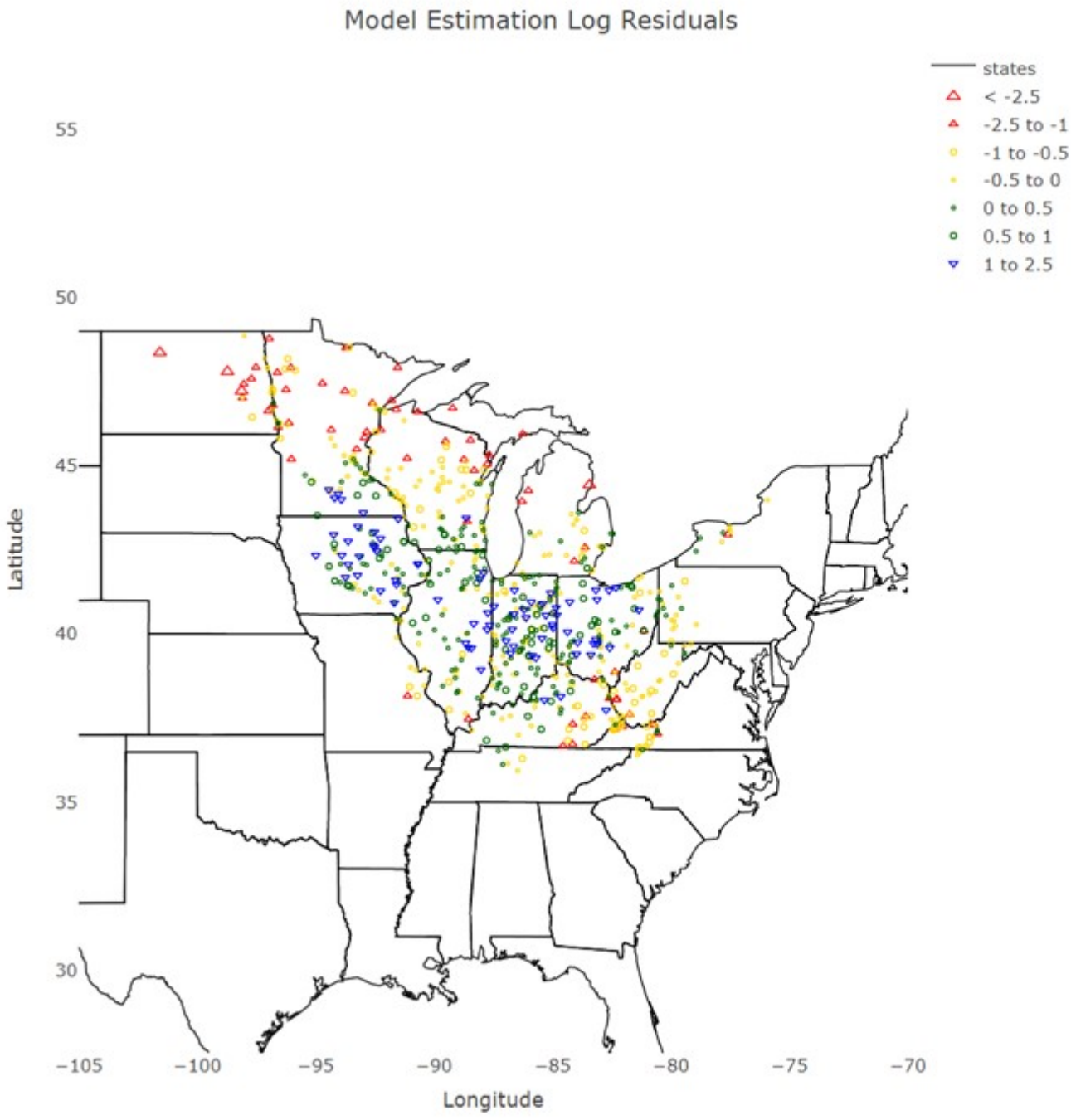


Figure 18: **Calibration site map of the model residuals for the total nitrogen model 1.** Overpredictions of load are shown by negative residual values and underpredictions of load are shown by postive residual values.

pasture, and forest. The table below gives the model performance metrics and estimated coefficients. The model performance metrics indicate a generally good fit to the load observations, with RMSE of 0.582 and a yield R-squared of 0.687. The RMSE of this model is about 25% lower and yield R-squared about 50% higher as compared to that for the incremental-area model 1. All of the land-use coefficients are statistically significant, with mean estimates of the total nitrogen export (kg/km2/yr) that fall within levels reported in the literature for watersheds that are dominated by these land uses (Beaulac and Reckow, 1982).

```
MODEL ESTIMATION PERFORMANCE (Monitoring-Adjusted Predictions)
 MOBS NPARM  DF      SSE       MSE      RMSE       RSQ RSQ-ADJUST RSQ-YIELD PERCENT BIAS
  708     4 704 239.0523 0.3395629 0.5827203 0.9036177   0.903207 0.6876266    -1.456124

MODEL SIMULATION PERFORMANCE (Simulated Predictions)
 MOBS NPARM  DF      SSE       MSE      RMSE       RSQ RSQ-ADJUST RSQ-YIELD PERCENT BIAS
  708     4 704 408.6079 0.5804089 0.7618457 0.8352554  0.8345534 0.4660656   -0.4884183

   Simulated predictions are computed using mean coefficients from the NLLS model
     that was estimated with monitoring-adjusted (conditioned) predictions

  PARAMETER SUMMARY
  PARAMETER ESTIMATE  PARM TYPE EST TYPE  INITIAL VALUE MIN MAX    DESCRIPTION
1 urban     1805.4018 SOURCE    Estimated 1             0   10000  Urban lands
2 crops     1954.5202 SOURCE    Estimated 1             0   10000  Cropland
3 pasture    665.4326 SOURCE    Estimated 1             0   10000  Pasture land
4 forest     221.3215 SOURCE    Estimated 1             0   10000  Forested land

  PARAMETER ESTIMATES
  PARAMETER PARM TYPE ESTIMATE  SE       T      P-VALUE VIF     DESCRIPTION PARAMETER UNITS
1 urban     SOURCE    1805.4018 294.3414  6.134 0       1.85665 Urban lands   kg/ha/yr
2 crops     SOURCE    1954.5202 102.4911 19.070 0       1.26607 Cropland      kg/ha/yr
3 pasture   SOURCE     665.4326 142.9547  4.655 0       1.52432 Pasture land  kg/ha/yr
4 forest    SOURCE     221.3215  30.3965  7.281 0       1.57780 Forested land kg/ha/yr

  EigenValue Spread  Normal PPCC  SWilks W      P-Value   Mean Exp Weighted Error
           6.019206    0.9291126 0.8812804 7.722224e-23                  1.14185
```

Diagnostic plots for load the yield metrics and residuals provide information on the model performance (Fig. 19). The plots of observed vs. predicted load (Fig. 19a) and yield (Fig. 19b) show reasonable fit with little bias in the fit to the observations across the range of load and yield values. The reported aggregate bias is about 1.5% for the model estimation performance (based on conditioned or monitoring-adjusted loads), while the bias is about 0.5% for the model simulation performance (based on unconditioned predictions). There is some modest evidence of heteroscedasticity in the residuals (Fig. 19c), with evidence of smaller variance in the larger watersheds with large mean load values.

One notable feature is evidence of some severe underprediction in the yields (Fig. 19b) in the range of predicted values from 1000 to 2000 kg/km2/yr, as demonstrated by the clustering of predictions nearly perpendicular to the red one-to-one line. Many of these underpredictions occur in watersheds with higher percentages of agricultural land (>80% of the incremental area above monitoring sites). The underpredictions are evident in the diagnostic boxplots of the ratio of the observed to predicted loads by deciles of the percentage of drainage in agricultural land (these are not shown but available in the diagnostic plot PDF file). This illustrates the limitation of a land-use model in agricultural drainages where the variety of crop and land management practices result in wide ranging nitrogen yields that a basic land-use based model is unable to accurately describe.

A closer examination of the diagnostic plots of the observed vs. predicted loads for the four major river basins in the Midwest (Fig. 20) shows generally reasonable fits in aggregate to the observed data in the Ohio and Upper Mississippi River basins (Fig. 20b,c). In the Great Lakes region (Fig. 20a), there's evidence of

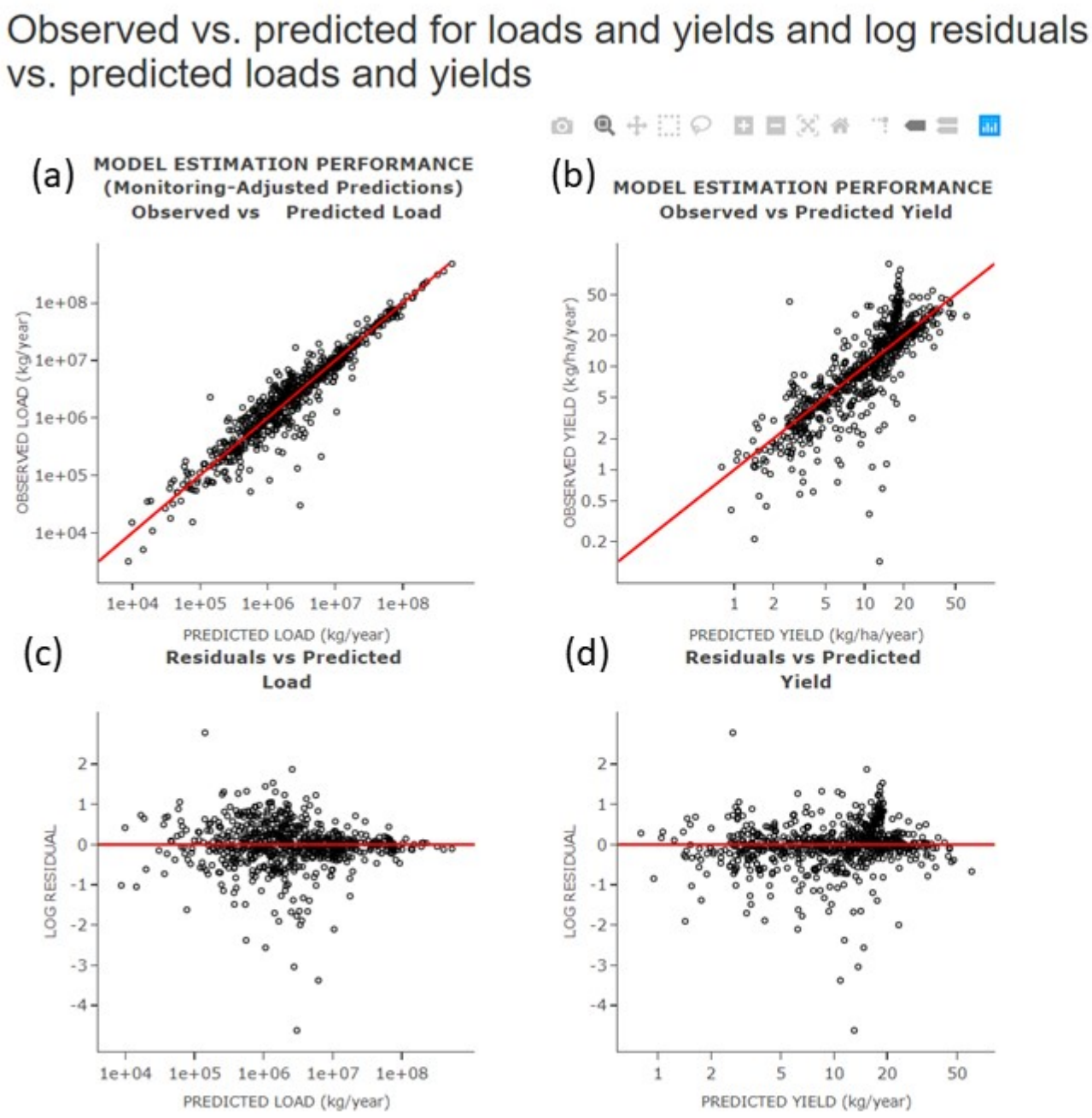


Figure 19: **Diagnostic plots for the RSPARROW total nitrogen model 2 with land use area as the explanatory variables.** Model performance plots for the monitoring-adjusted (conditioned) predictions are shown for: (a) observed load vs. predicted load; (b) observed yield vs. predicted yield; (c) log residuals vs. predicted load; and (d) log residuals vs. predicted yield.

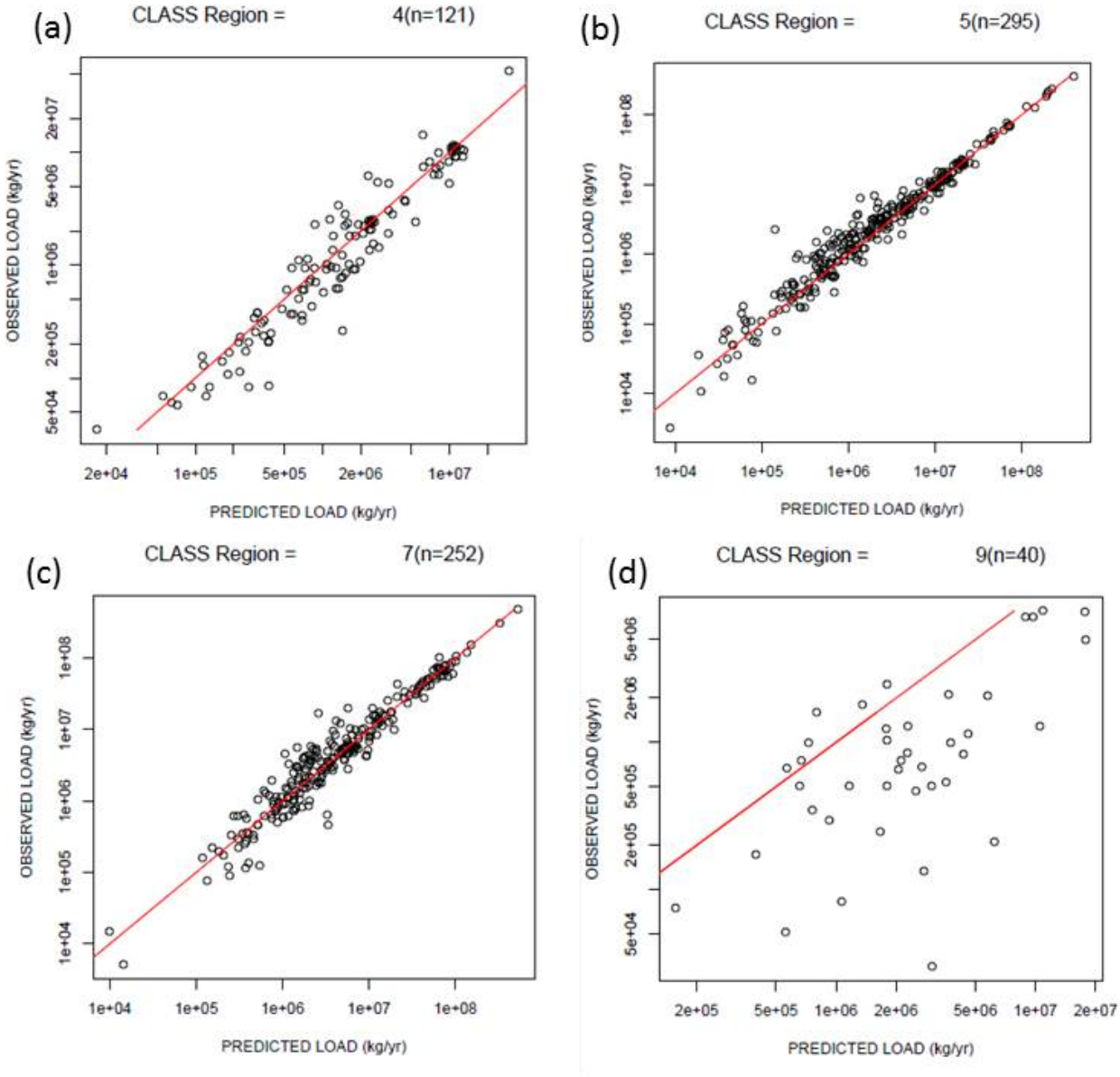


Figure 20: **Plots of the observed vs. predicted loads for the total nitrogen model 2 for the four major rivers basins of the Midwest:** (a) Great Lakes region (HUC2=4); (b) Ohio River Basin (HUC2=5); (c) Upper Mississippi River Basin (HUC2=7); and (d) Red-Rainy Basin (HUC2=9).

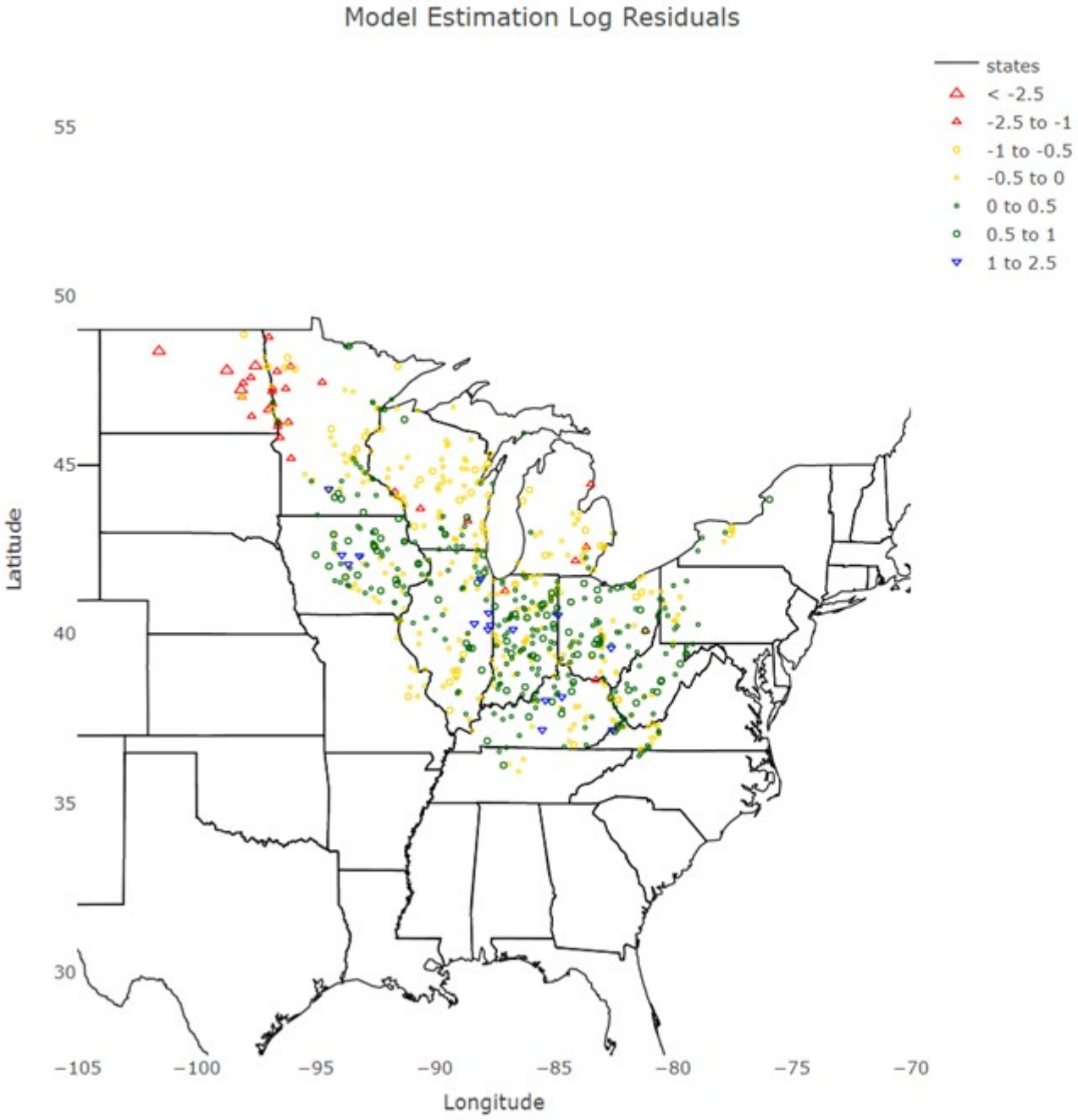


Figure 21: **Calibration site map for the model residuals of the total nitrogen model 2.** Overpredictions are shown by negative values; underpredictions are shown by postive values.

overprediction in the watersheds with smaller loads, while in the Red-Rainy basin (Fig. 20d) the model fits very poorly with overpredictions at virtually all of the 40 calibration sites.

The map of the model residuals for the calibration sites (Fig. 21) shows some evidence of sub-regional biases in the model predictions within the four major regional basins. This includes widespread underpredictions of load in streams in the states of Iowa, Illinois, Indiana, and parts of Ohio, although these are of a small magnitude. This is consistent with the previously cited evidence of underpredictions in agriculturally dominated watersheds. A large overprediction of stream loads in the Red-Rainy basin, located in the northeastern portion of the Midwest region and modeled domain, is also apparent.

6.2.3 Model 3: Land-use source variables (six land-use types)

Two land uses, `shrubgrass` (a combination of the two land uses) and `barren` lands, were added to the model. The table below gives the model performance metrics and estimated coefficients. The model performance metrics, RMSE of 0.578 and yield R-squared of 0.693, are only about 1% improved over that for the four land-use model. The prediction bias is relatively unchanged from that for the four land-use model. Among the added source variables, only the `barren` land use is statistically signficant; the `shrubgrass` land use variable is highly insignificant (p=0.97).

```
  MODEL ESTIMATION PERFORMANCE (Monitoring-Adjusted Predictions)
 MOBS NPARM  DF      SSE       MSE      RMSE       RSQ RSQ-ADJUST RSQ-YIELD PERCENT BIAS
  708     6 702 234.8774 0.3345832 0.5784317 0.9053009  0.9046264  0.693082    -1.432451

  MODEL SIMULATION PERFORMANCE (Simulated Predictions)
 MOBS NPARM  DF      SSE       MSE     RMSE       RSQ RSQ-ADJUST RSQ-YIELD PERCENT BIAS
  708     6 702 402.3707 0.5731777 0.757085 0.8377701  0.8366147 0.4742158   -0.4128916

   Simulated predictions are computed using mean coefficients from the NLLS model
     that was estimated with monitoring-adjusted (conditioned) predictions

  PARAMETER SUMMARY
  PARAMETER  ESTIMATE   PARM TYPE EST TYPE  INITIAL VALUE MIN MAX    DESCRIPTION
1 urban      1732.83140 SOURCE    Estimated 1             0   50000  Urban lands
2 crops      1961.22458 SOURCE    Estimated 1             0   50000  Cropland
3 pasture     724.96951 SOURCE    Estimated 1             0   50000  Pasture land
4 forest      165.07230 SOURCE    Estimated 1             0   50000  Forested land
5 shrubgrass    0.00137 SOURCE    Estimated 1             0   50000  Combined shrub/grass
6 barren     8359.90207 SOURCE    Estimated 1             0   50000  Barren land

  PARAMETER ESTIMATES
  PARAMETER  PARM TYPE ESTIMATE   SE        T      P-VALUE VIF     DESCRIPTION
1 urban      SOURCE    1732.83140  285.5235  6.069 0.00000 1.82964 Urban lands
2 crops      SOURCE    1961.22458  101.7586 19.273 0.00000 1.28865 Cropland
3 pasture    SOURCE     724.96951  142.7412  5.079 0.00000 1.51654 Pasture land
4 forest     SOURCE     165.07230   32.4603  5.085 0.00000 1.80071 Forested land
5 shrubgrass SOURCE       0.00137    0.0409  0.034 0.97318 1.00526 Combined shrub/grass
6 barren     SOURCE    8359.90207 3018.4737  2.770 0.00576 1.32356 Barren land

  EigenValue Spread  Normal PPCC  SWilks W      P-Value   Mean Exp Weighted Error
           6.620065    0.9374364 0.8778245 3.859367e-23                 1.143579
```

The effect of a highly statistically insignificant variable (`shrubgrass`; p=0.973) on the stability of the estimated coefficients and residuals for model 3 is demonstrated through a series of three models: 3a, 3b, and 3c. The results for these models are only discussed here; details for the model summary metrics can be examined in the saved model sub-directories.

In model 3a, model 3 is re-estimated with six land-use sources using initial values of 0.1 rather than 1.0. Several statistical measures indicate that model 3a provides a less acceptable model fit than that of model 3. First, model 3a has several unusually large diagnostic values associated with the over-prediction of nitrogen load at the predominantly forested (92%) Upper Twin Creek at McGaw OH station, including values for the standardized residual (-48.4), leverage (0.998 exceeds critical value of 0.025), and Cook's D (390,190, with $p<0.0001$). Second, the estimated mean coefficient for `barren` land is sensitive to the change in the initial value, with the magnitude of the coefficient reduced by about 30% to 5,522 kg/km2/year. The other land use coefficients do not display sensitivity to the initial value. Finally, the RMSE (0.5791) is slightly higher than that for model 3 (RMSE=0.5784), suggesting that the model converged to a local minima.

In model 3b, the statistically insignificant `shrubgrass` land use variable in model 3a has been removed. Using the initial values of 0.1 now gives a more acceptable model fit. The RMSE for model 3b (0.5780) is slightly less than that observed for model 3 (0.5784). The excessively large overprediction of the load at the Upper Twin Creek site is now appreciably smaller (standardized residual = -1.407). The mean coefficient for the `barren` land use (9,186) is also within about 10% of the magnitude of the estimated coefficient in model 3 (8,359).

In model 3c, the statistical stability of the fit of model 3b is demonstrated by changing the intial values of the land-use sources to 1.0. Results for model 3c are very similar to those observed for model 3b, including the RMSE (0.57797 vs. 0.57798), mean coefficient value for the `barren` land use (9,186 vs. 10,000), and the bias-retransformation correction factor (`Mean Exp Weighted Error`; 1.131 vs. 1.131), respectively.

6.2.4 Model 4: Mass-based source variables only

The model was estimated using four major mass-based sources. These included municipal and industrial wastewater discharges of nitrogen (`point` sources), atmospheric wet deposition of nitrogen (`ndep`), and nitrogen in the wastes from livestock (confined operations; `MANC_N`) and farm fertilizers (`FARM_N`). Mass-based sources are generally preferable to using land-use surrogates for sources in SPARROW models; the mass source data typically provides more spatially specific information about the inputs that contributes to improved prediction accuracy and model interpretability.

The table below gives the model performance metrics and estimated coefficients. The model performance metrics indicate a generally good fit to the load observations, with RMSE of 0.528 and a yield R-squared of 0.743. The RMSE and yield R-squared reflect about a 10% reduction in model error and increase in explanatory power, respectively, as compared to that for the land-use area-based model. However, the prediction bias of the model is somewhat higher than that for the land-use area-based models (models 2 and 3).

Diagnostic plots for load the yield metrics and residuals provide information on the model performance (Fig. 22). The plots of observed vs. predicted load (Fig. 22a) and yield (Fig. 22b) show some modest improvement in the fit to the observations across the range of load and yield values compared with the fits for the land-use model. Most notable is a reduction in the severity of the underprediction in the yields (Fig. 22b) in the range of predicted values from 1000 to 2000 kg/km2/yr, which was evident for the land-use model. Although underprediction still occurs in these predominantly agricultural watersheds, the source model likely provides a more precise characterization of nitrogen sources from farm fertilizers and livestock wastes in the these areas. Although not shown here, the diagnostic plots of the observed vs. predicted loads for the four major river basins in the Midwest show only very modest improvements in the model fit in these regions.

```
 MODEL ESTIMATION PERFORMANCE (Monitoring-Adjusted Predictions)
MOBS NPARM  DF      SSE      MSE      RMSE       RSQ RSQ-ADJUST RSQ-YIELD PERCENT BIAS
 708     4 704 196.5484 0.279188 0.5283825 0.9207546  0.9204169 0.7431671    -2.683733

 MODEL SIMULATION PERFORMANCE (Simulated Predictions)
MOBS NPARM  DF      SSE       MSE      RMSE       RSQ RSQ-ADJUST RSQ-YIELD PERCENT BIAS
 708     4 704 330.0107 0.4687653 0.6846643 0.8669446  0.8663776 0.5687697    -9.070599

  Simulated predictions are computed using mean coefficients from the NLLS model
```

```
     that was estimated with monitoring-adjusted (conditioned) predictions

PARAMETER ESTIMATES
  PARAMETER PARM TYPE ESTIMATE SE     T      P-VALUE VIF     DESCRIPTION
1 point     SOURCE    0.81317  0.1463  5.557 0e+00   1.07964 Wastewater discharge
2 ndep      SOURCE    0.43016  0.0291 14.769 0e+00   1.36010 Atmospheric deposition N
3 MANC_N    SOURCE    0.25330  0.0625  4.051 6e-05   1.88254 Livestock manure N
4 FARM_N    SOURCE    0.19184  0.0164 11.716 0e+00   1.99585 Fertilizer N use

  EigenValue Spread  Normal PPCC  SWilks W      P-Value   Mean Exp Weighted Error
           6.717078    0.9410088 0.8935373 1.029132e-21                  1.113028
```

All of the source coefficients are statistically significant. The coefficient units are dimensionless because both source and stream load mass have identical units, with coefficient values physically interpreted as the mean fraction of the mass input delivered to streams.

A point-source coefficient of 1.0 is theoretically expected, given that the point source data measure direct discharges to streams; this interpretation assumes that the effluent discharges and model specification are reasonably accurate. The estimated value of 0.81 has a 95% confidence interval that includes 1.0; thus, providing generally acceptable confirmation of this hypothesis.

The other coefficients are less than 1.0 as expected, because removal processes have a prominent effect on nitrogen delivery to streams from these sources. The deposition source includes wet forms of nitrate only (e.g., dry N and ammonium are excluded from the deposition mass inputs). Thus, it is likely than much less than the estimated 43% of the unknown but larger total mass deposition inputs are delivered to streams (dry deposition is roughly equivalent to wet in the eastern United States, so the fraction of the actual total could be less than one half of that estimated by SPARROW). Coefficients for the agricultural sources suggest that less than a quarter of the fertilizer inputs and livestock wastes are delivered to streams. These estimates provide an approximate aggregate accounting for a variety of complex removal processes, including crop removal, conservation management practices, legume fixation, and supplemental use of manure fertilizer.

6.2.5 Model 5: Addition of aquatic decay variables

In this model, aquatic decay variables were added to a model with the same four mass-based sources, including municipal and industrial wastewater discharges of nitrogen (point sources), atmospheric deposition of nitrogen (wet only), and nitrogen in the wastes from livestock and farm fertilizers.

For streams, three volume-dependent reaction-rate coefficient constants (`rchdecay1`, `rchdecay2`, `rchdecay3`), expressed as the rate per day of mean annual water travel time, were estimated for the following stream sizes: small streams with mean discharge less than 1.13 m3/s; medium streams with mean discharge greater than 1.13 m3/s and less than 1.93 m3/s; and large sterams with mean discharge greater than 1.93 m3/s. An exponential rate expression was used according to the standard formulation in the RSPARROW control script (equation 1.30, Schwarz et al., 2006; control setting `reach_decay_specification` described in Chapter sub-section 4.4.4.1). The equation requires reach-level estimates of the mean annual water travel time, based on an estimated mean annual stream water velocity and the reach length. The R ifelse statements necessary to define the explanatory variables (`rchdecay1`, `rchdecay2`, `rchdecay3` in the *data1.csv* file) are shown in the *userModifyData.R* script as commented statements (i.e., preceded by a '#' symbol); the variables are conditioned on both the mean annual streamflow (`meanq`) and the reach type indicator for streams (`rchtype`).

For reservoirs, a single mass-transfer coefficient (`iresload`), expressed as meters per year, was estimated according to the standard formulation provided in the RSPARROW control script (equation 1.34, Schwarz et al., 2006; control setting `reservoir_decay_specification` described in Chapter sub-section 4.4.4.1). The equation requires reservoir measures of the areal hydraulic load, the water flushing rate of the reseroir, computed as the ratio of the outflow discharge to the reservoir surface area. The R ifelse statements necessary to define the explanatory variable (`iresload` in the *data1.csv* file) are shown in the *userModifyData.R* script as commented statements (i.e., preceded by a '#' symbol); the variable is conditioned on both the areal

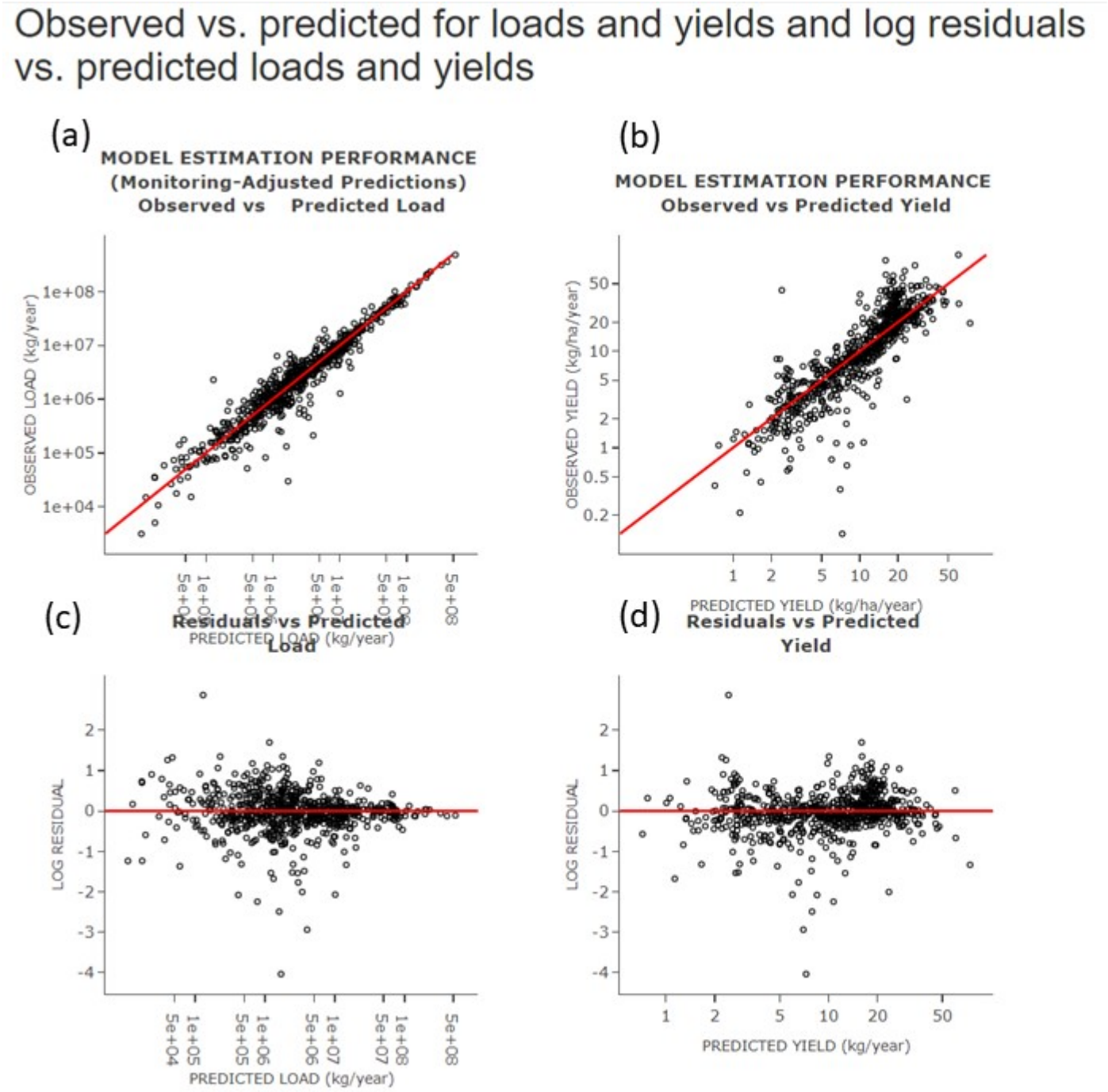


Figure 22: **Diagnostic plots for the RSPARROW total nitrogen model 4 with mass-based sources as the explanatory variables.** Model performance plots for the monitoring-adjusted (conditioned) predictions are shown for: (a) observed load vs. predicted load; (b) observed yield vs. predicted yield; (c) log residuals vs. predicted load; and (d) log residuals vs. predicted yield.

hydraulic load (`hload`) and the reach type indicator for reservoirs (`rchtype`).

The table below gives the model performance metrics and estimated coefficients. The model performance metrics indicate a generally good fit to the load observations, with RMSE of 0.463 and a yield R-squared of 0.804. The RMSE and yield R-squared reflect about a 8-12% improvement in the model error and explanatory power of the model as compared to that for the source-only model 4, with the prediction bias of the model generally about the same. Although not shown here, the diagnostic plots of the observed vs. predicted loads for the four major river basins in the Midwest show generally very modest improvements in the model fit for the four regions.

```
  MODEL ESTIMATION PERFORMANCE (Monitoring-Adjusted Predictions)
 MOBS NPARM  DF     SSE       MSE      RMSE       RSQ RSQ-ADJUST RSQ-YIELD PERCENT BIAS
  708     8 700 150.232 0.2146171 0.4632679 0.9394287   0.938823 0.8036895     2.432207

  MODEL SIMULATION PERFORMANCE (Simulated Predictions)
 MOBS NPARM  DF      SSE       MSE      RMSE       RSQ RSQ-ADJUST RSQ-YIELD PERCENT BIAS
  708     8 700 215.7677 0.3082396 0.5551933 0.9130057  0.9121357 0.7180529     14.57711

   Simulated predictions are computed using mean coefficients from the NLLS model
     that was estimated with monitoring-adjusted (conditioned) predictions

PARAMETER ESTIMATES
   PARAMETER PARM TYPE ESTIMATE SE     T      P-VALUE VIF     DESCRIPTION
1 point     SOURCE     0.78865 0.1330  5.930 0.00000 1.07615 Wastewater discharge
2 ndep      SOURCE     0.55179 0.0366 15.093 0.00000 1.55823 Atmospheric deposition N
3 MANC_N    SOURCE     0.22109 0.0638  3.466 0.00056 1.88481 Livestock manure N
4 FARM_N    SOURCE     0.28103 0.0212 13.257 0.00000 2.49874 Fertilizer N use
5 rchdecay1 STRM       0.66137 0.1078  6.134 0.00000 1.20285 Small stream decay
6 rchdecay2 STRM       0.37918 0.1054  3.599 0.00034 1.37701 Medium stream decay
7 rchdecay3 STRM       0.03184 0.0204  1.563 0.11846 1.47450 Large stream decay
8 iresload  RESV      14.75500 2.6157  5.641 0.00000 1.15018 Reservoir decay

  EigenValue Spread  Normal PPCC  SWilks W      P-Value   Mean Exp Weighted Error
           10.95707    0.9712467 0.9377858 1.297841e-16                 1.123383
```

All of the source coefficients are highly statistically significant (<0.001), with the exception of the reaction rate coefficient (`rchdecay3`) for large rivers (p=0.118). The magnitudes of the source coefficients changed slightly from the previous source-only model 4, with the largest change observed for the farm fertilizer use, which increased in magnitude. Changes would generally be expected in the source coefficients, given that the model now provides an explicit accounting of nitrogen losses in aquatic systems that were previously imprecisely accounted for by the sources and/or model error.

The stream reaction rates range from 0.66 per day of water travel time in small streams (`rchdecay1`) to 0.03 per day of water travel time in large rivers (`rchdecay3`). The reaction rates decline in magnitude from small streams to large rivers, a pattern is frequently observed in SPARROW models (Preston et al., 2011a). This inverse relation is also consistent with the volume-dependent expression of the decay rates and with literature estimates of nitrogen loss in streams related to denitrification effects (e.g., Alexander et al., 2000; 2008). The lower statistical significance of the reaction rate coefficient for large rivers (`rchdecay3`) is typical of most SPARROW models because of the expected small magnitude of the rate, especially relative to its standard error.

The reservoir mean annual mass-transfer coefficient (`iresload`) is estimated to be nearly 15 meters per day, with a relatively small standard error. This is generally within the range of literature estimates (e.g., Alexander et al., 2008).

6.2.6 Model 6: Addition of land-to-water delivery variables (Final Model)

The model presented in this sub-section contains the same explanatory variables of the final model reported by Robertson and Saad (2011). Based on prior exploratory analyses by the authors, the following land-to-water delivery variables were found to be statistically important predictors of stream total nitrogen loads: stream drainage density (`ldrainden`), mean annual precipitation (`PPT30MEAN`), mean annual temperature (`meanTemp`), percentage of incremental reach area in tile drainage (`tiles_perc`), and soil clay expressed as a fraction of the incremental reach area (`soil_CLAYAVE`).

The model was also executed with the following source, stream decay, and land-to-water attributes:

(1) An agricultural source variable, "nitrogen fixation" (`Fixation`), was added to the model, based on the authors' investigations. The variable is expressed as the mass of nitrogen contained in legume crops, based on the land area repoted to be in legume production. The variable is assumed to be potentially related to nitrogen fixation. The authors found the variable to be statistically significant and that it reduced prediction biases in agricultural areas of the basin.

(2) A log transformed stream drainage density variable (`ldrainden`) was used in the model because the authors found that a log transform of the drainage density produced a more symmetrical distribution of the values. This is a desirable objective in SPARROW modeling to reduce the potential influence of more extreme values on the statistical fit to the data when using the exponential land-to-water delivery function (Schwarz et al., 2006). Note that it's also generally preferable to express the land-to-water variables according to a measure of intensity or density (e.g., percent of area or a mass quantity per unit area) to provide a more standardized and symmetrical expression of the variable.

(3) The large-river decay variable (`rchdecay3`) was removed because it was statistically insignificant ($p>0.10$).

(4) All models with land-to-water delivery variables were executed using the control setting `if_mean_adjust_delivery_vars<-"yes"`. This setting performs a mean-adjustment to the land-to-water delivery variables by subtracting the mean of the variable over the full spatial domain from each reach-level value of the variable. Mean-adjustment is recommended to ensure that the reported values of the mean coefficients for the sources can be accurately compared and interpreted across different models (i.e., when testing different land-to-water variables or comparing sources across different models). The adjustment standardizes each source coefficient to the mean of the land-to-water delivery values for the full spatial domain, allowing such comparisons to be reliably made.

(5) The model is executed with the source and land-to-water delivery interaction matrix as shown below. All of the land-to-water delivery factors are allowed to interact with the non-point sources in the model. No interaction is allowed for the wastewater load discharge (point-source) variables, given that the loads are discharged directly to streams and should not be influenced by climate, soils, and tile drainage practices.

```
# model6_design_matrix.csv (source and land-to-water delivery interaction matrix)
sparrowNames    ldrainden   PPT30MEAN meanTemp tiles_perc   soil_CLAYAVE
point           0             0         0          0             0
ndep            1             1         1          1             1
MANC_N          1             1         1          1             1
FARM_N          1             1         1          1             1
Fixation        1             1         1          1             1
```

The summary table below gives the model performance metrics for model 6. The model `ESTIMATION` performance metrics indicate a generally good fit to the load observations, with RMSE of 0.408 and a yield R-squared of 0.849, with a bias of nearly -2% (small average over-prediction of load). The RMSE and yield R-squared reflect a 12% reduction in model error and 6% increase in explanatory power, respectively, as compared to that for the source and aquatic decay model (model 5) presented in the previous subsection. The prediction bias of the model is less than 2%, which is less than reported for the prior mass-based models. Compared with the incremental drainage area only model (model 1), the RMSE of the final model shown

below has an RMSE that is more than 50% lower and a yield R-squared that is larger by a factor of 1.95.

```
MODEL 6 PERFORMANCE SUMMARY

 MODEL ESTIMATION PERFORMANCE (Monitoring-Adjusted Predictions)
MOBS NPARM  DF      SSE       MSE      RMSE       RSQ RSQ-ADJUST RSQ-YIELD PERCENT BIAS
 708    13 695 115.6873 0.1664566 0.4079909 0.9533566  0.9525512 0.8488295    -1.877961

 MODEL SIMULATION PERFORMANCE (Simulated Predictions)
MOBS NPARM  DF      SSE       MSE      RMSE       RSQ RSQ-ADJUST RSQ-YIELD PERCENT BIAS
 708    13 695 155.4753 0.2237054 0.4729751 0.9373147  0.9362324  0.796838      -4.9201

  Simulated predictions are computed using mean coefficients from the NLLS model
    that was estimated with monitoring-adjusted (conditioned) predictions

      EigenValue Spread  Normal PPCC  SWilks W      P-Value   Mean Exp Weighted Error
         82.30764    0.9553319  0.927996 6.290676e-18                  1.081618
```

The model ESTIMATION performance metrics, reported in the table above, are based on the use of the conditioned (monitoring-adjusted) predictions. The conditioned predictions describe the most accurate reach predictions for the purpose of quantifying river loads and model coefficients, which quantify the effects of contaminant sources and hydrological and biogeochemical processes on stream quality; thus, the predictions are used to estimate/calibrate the model. The associated ESTIMATION metrics can be used to assess the adequacy of the overall model fit to the observations.

The SIMULATION performance metrics are based on the unconditioned predictions with monitoring conditioning effects removed (i.e., the effects of substituting the observed station loads for the model predicted loads). The simulation metrics provide the best representation of the predictive skill of the estimated model at the monitored locations, and give a generally preferred estimate of the expected average accuracy of the model when applied to unmonitored stream reaches. The metrics are also well suited for comparing the prediction accuracy of different models because the conditioning effects are removed. For model 6, the accuracy in SIMULATION mode is slightly lower than that reported for the model in ESTIMATION mode, with the model performance in SIMULATION mode showing a somewhat higher model RMSE (~15%) and prediction bias. Also, the scatterplot of the conditioned (monitoring-adjusted load) vs. unconditioned (simulated load) predictions (see p. 10 of *Model6/estimate/Model6_diagnostic_plots.html*) visually confirms the general agreement in these predictions, as they display relatively uniform and modest differences along the line of equivalence.

Diagnostic plots for load the yield metrics and residuals provide information on the model performance (Fig. 23). The plots of observed vs. predicted load (Fig. 23a) and yield (Fig. 23b) show notable improvement in the fit to the observations across the range of load and yield values compared with the fits for the mass source-only model 4. Compared with model 4, the observed vs. predicted fits for load and yield for model 6 display a much tighter fit to the one-to-one line, with less overall variation and deviation from the line.

There's also reduction in the underprediction in the yields (Fig. 23b) in the range of predicted values from 1000 to 2000 kg/km2/yr, which was evident for the land-use models (model 2 and 3). Although some underprediction still occurs in these predominantly agricultural watersheds, model 6 provides much less biased predictions than observed for the earlier models. This model would appear to provide an improved characterization of nitrogen sources from farm fertilizers and livestock wastes in the these areas, after accounting for the additional effects of climate (precipitation, temperature) and landscape (soil clay, tile drainage) variables that affect nitrogen delivery to streams.

Re-visiting the diagnostic plots of the observed vs. predicted loads for the four major river basins in the Midwest (Fig. 24) indicates that the model fits to the observed data are much improved over that for the previous models. All of the plots display much tighter fits to the one-to-one lines. In addition, the Red-Rainy basin (Fig. 24d) shows a much improved fit of the observed data, although overpredictions are still more common at the calibration sites and there's much lower model precision overall across the 40 calibration sites.

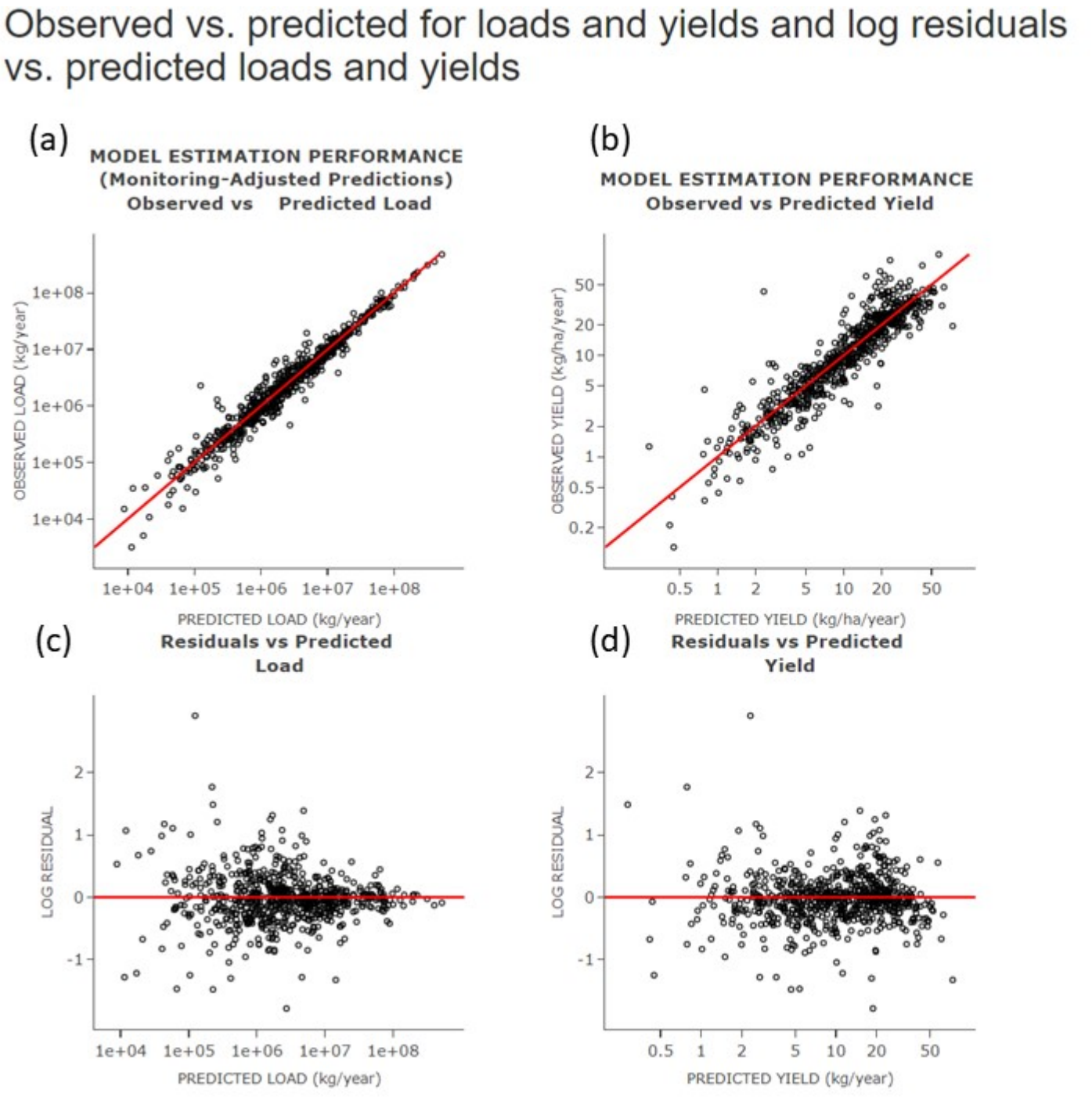


Figure 23: **Diagnostic plots for the RSPARROW total nitrogen model 6 with source, land-to-water delivery, and aquatic decay explanatory variables.** Model performance plots for the monitoring-adjusted (conditioned) predictions are shown for: (a) observed load vs. predicted load; (b) observed yield vs. predicted yield; (c) log residuals vs. predicted load; and (d) log residuals vs. predicted yield.

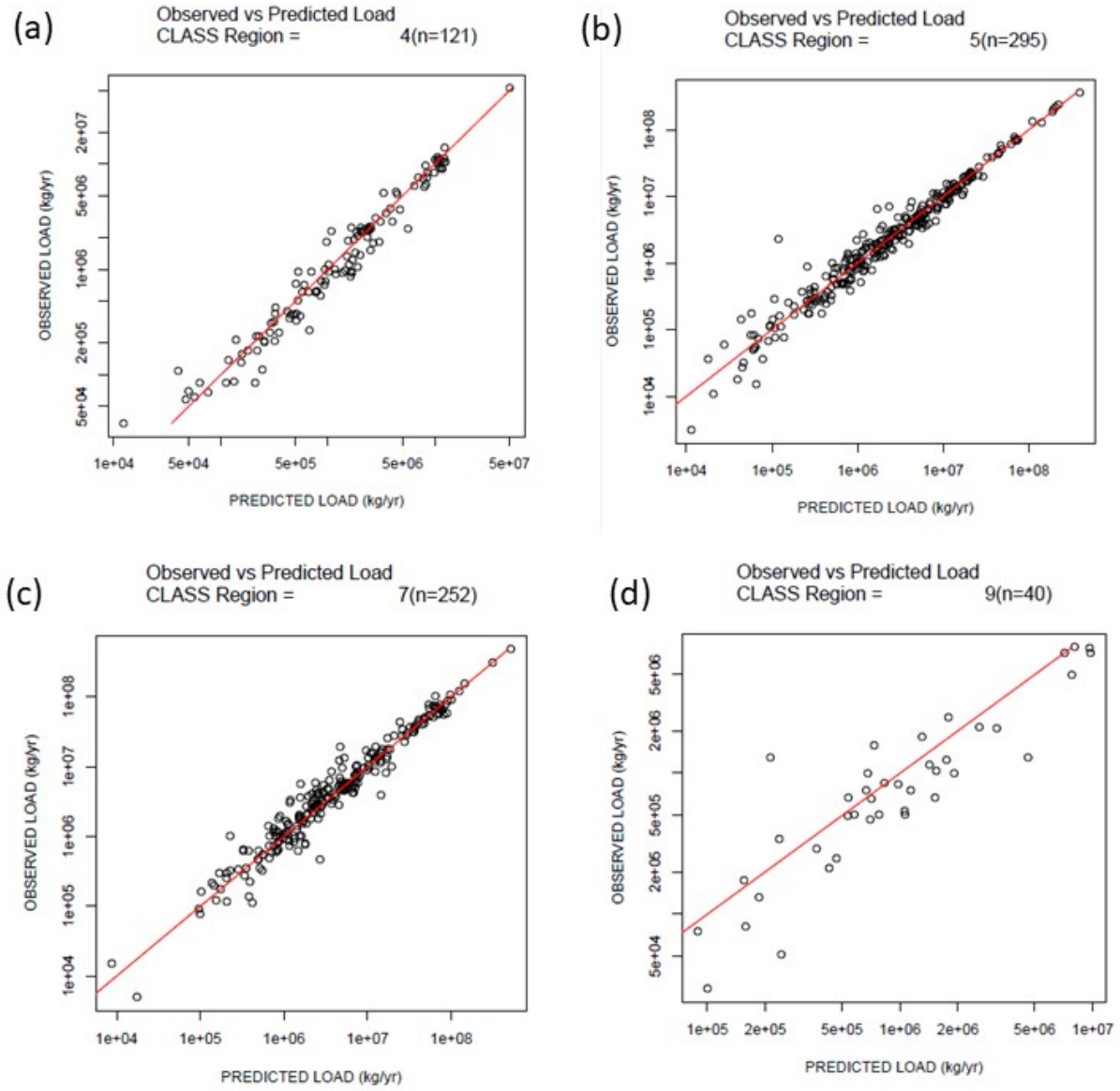


Figure 24: **Plots of the observed vs. predicted loads for the four major rivers basins of the Midwest for the total nitrogen model 6 with source, land-to-water delivery, and aquatic decay explanatory variables:** (a) Great Lakes region (HUC2=4); (b) Ohio River Basin (HUC2=5); (c) Upper Mississippi River Basin (HUC2=7); and (d) Red-Rainy Basin (HUC2=9).

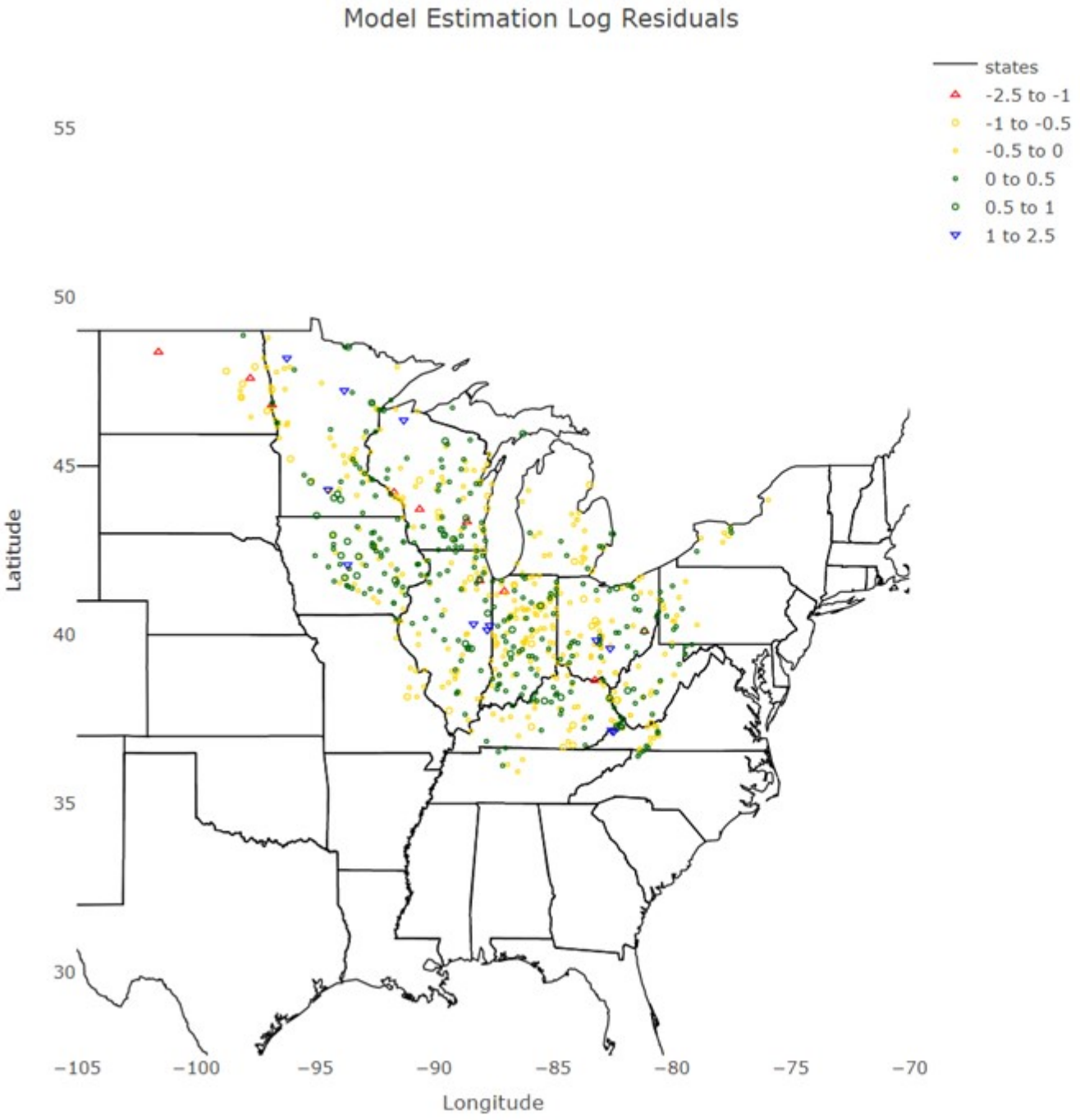


Figure 25: **Calibration site map for the model residuals for the total nitrogen model 6 with source, land-to-water delivery, and aquatic decay explanatory variables.** Overpredictions are shown by negative values; underpredictions are shown by postive values.

The map of the model residuals for the calibration sites (Fig. 25) also displays much more spatial balance in the occurrence of model over- and under-predictions as compared to the patterns for the earlier less complex models. Some clustering of large residuals associated with over-predictions are apparent in certain areas of the Midwest, such as in the Red-Rainy and Ohio basins.

The estimated mean coefficients for model 6, reported in the table below, are all statistically significant for p-values less than 0.06. Adding the land-to-water delivery factors caused the large river decay coefficient to be less statistically signficant than in the prior model and it was removed from the final model by the authors. The magnitude of the source coefficients are largely unchanged with the exception of the farm fertilizer coefficient which falls by more than one half. All of the land-to-water delivery factors have positive signs with the exception of temperature; all of the signs have reasonable physical interpretations as to their effects on nitrogen delivery to streams. Precipitation, clay soils, and tile drainage are estimated to enhance delivery to streams, based on their positive coefficients. The inverse relation of temperature with nitrogen delivery could potentially be indicative of a biologically mediated response of nitrogen to temperature that would be associated, for example, with denitrification. The addition of the land-to-water delivery factors led to slight decreases in the magnitudes of the in-stream and reservoir removal rates. This is a generally expected response, given that a more complex mass-balance model with landscape features is likely to attribute additional nitrogen losses within watersheds to landscape processes, thereby lowering the magnitude of the nitrogen loss in aquatic systems. The addition of the land-to-water delivery factors also allows for a much more spatially variable response of the model to source and landscape factors.

The addition of the nitrogen fixation source variable (`Fixation`) contributes to some increased levels of multicollinearity in the model, as measured by the eignvalue spread (now 82 vs. 11 in the previous model) and a several-fold increase in the VIF (Variance Inflation Factor) for the farm fertilizer use source. The nitrogen fixation also shows a much larger VIF than is reported for other variables. The collinearity between the farm ferilizer use (`FARM_N`) and nitrogen fixation (`Fixation`) sources is relatively high (VIF values of 10 for each coefficient); however, not sufficient to cause statistical insignificance for these variables. Farm fertilizer use is still highly significant (p=0.0065), whereas the nitrogen fixation source is moderately significant (p=0.0502).

PARAMETER SUMMARY (MODEL 6)

	PARAMETER	ESTIMATE	PARM TYPE	EST TYPE	INITIAL VALUE	MIN	MAX
1	point	0.80022	SOURCE	Estimated	0.01	0	10000
2	ndep	0.51288	SOURCE	Estimated	0.01	0	10000
3	MANC_N	0.29239	SOURCE	Estimated	0.01	0	10000
4	FARM_N	0.12047	SOURCE	Estimated	0.01	0	10000
5	Fixation	6.78721	SOURCE	Estimated	1.00	0	10000
6	ldrainden	0.12705	DELIVF	Estimated	0.00	-10000	10000
7	PPT30MEAN	0.00158	DELIVF	Estimated	0.00	-10000	10000
8	meanTemp	-0.03866	DELIVF	Estimated	0.00	-10000	10000
9	tiles_perc	1.13357	DELIVF	Estimated	0.00	-10000	10000
10	soil_CLAYAVE	0.01450	DELIVF	Estimated	0.00	-10000	10000
11	rchdecay1	0.41906	STRM	Estimated	0.01	0	10000
12	rchdecay2	0.22990	STRM	Estimated	0.01	0	10000
13	iresload	6.44912	RESV	Estimated	0.01	0	10000

PARAMETER ESTIMATES (MODEL 6)

	PARAMETER	PARM TYPE	ESTIMATE	SE	T	P-VALUE	VIF	DESCRIPTION
1	point	SOURCE	0.80022	0.1120	7.143	0.00000	1.11171	Wastewater discharge
2	ndep	SOURCE	0.51288	0.0378	13.571	0.00000	2.60675	Atmospheric deposition N
3	MANC_N	SOURCE	0.29239	0.0588	4.970	0.00000	2.13773	Livestock manure N
4	FARM_N	SOURCE	0.12047	0.0441	2.729	0.00650	10.81413	Fertilizer N use
5	Fixation	SOURCE	6.78721	3.4592	1.962	0.05015	10.08344	Cropland area (fixation)
6	ldrainden	DELIVF	0.12705	0.0579	2.193	0.02863	2.27866	Drainage density
7	PPT30MEAN	DELIVF	0.00158	0.0003	5.814	0.00000	4.09339	Precipitation
8	meanTemp	DELIVF	-0.03866	0.0206	-1.880	0.06049	3.56965	Air temperature

```
9  tiles_perc   DELIVF     1.13357 0.1270  8.926 0.00000  1.69179 Tile drainage
10 soil_CLAYAVE DELIVF     0.01450 0.0041  3.571 0.00038  1.66033 Soil clay content
11 rchdecay1    STRM       0.41906 0.0911  4.601 0.00000  1.32673 Small stream decay
12 rchdecay2    STRM       0.22990 0.0900  2.555 0.01083  1.36065 Medium stream decay
13 iresload     RESV       6.44912 1.6191  3.983 0.00008  1.27076 Reservoir decay
```

6.2.7 Model 7: Effects of the initial parameter values on estimated coefficient metrics

The use of different initial coefficient values (*parmInit*) for the more marginally significant variables can potentially have a large influence on the estimated coefficient metrics (mean, standard error, t-statistics) of these variables. As an illustration of the potential effect, model 6 was re-estimated (as model 7) using an initial coefficient value (*parmInit*) for the `Fixation` source of 0.01 rather than a value of 1.0 as used for model 6. The results are shown below.

One concern with the results is that the RMSE of model 7 increases slightly, suggesting that the model has converged only to a local optima.

Moreover, the statistical significance of the two moderately collinear agricultural/farm sources in model 6 (`FARM_N` and `Fixation` with VIFs of 10) now display very different levels of statistical significance. Farm fertilizer (`FARM_N`) is now highly insignificant (p=0.87 vs. p=0.0065 in the previous model). `Fixation` is now highly statistically significant (p=0.0026), whereas its p-value was only moderately significant (p=0.05) in the previous model. The estimated mean coefficient for `Fixation` is also one-half of the previous mean estimate.

Additionally, the use of an alternative initial coefficient value for `Fixation` of 0.1 (model 7a; not shown) rather than 0.01 produces a model with an even higher RMSE (0.4090), with many more coefficients now showing statistical insignificance, including fixation (p=0.43), precipitation (p=0.99), and temperature (p=0.99).

Thus, in this example, a larger initial coefficient value (i.e., 1.0 or larger) is required for the `Fixation` source (as illustrated in model 6) in order to obtain a model with lower model errors (RMSE) and more stable coefficient metrics. As further confirmation of this, we find that model 6, in which an initial value of 1.0 is used for the `Fixation` source, gives results that are much more consistent with those estimated by SAS SPARROW (see the results for model 8 in the next sub-section 6.2.8).

Our recommendation for users is to be especially vigilant in evaluating the effects of the initial parameter values on the performance metrics (RMSE) and coefficient metrics of models that include explanatory variables with marginal (e.g., 0.01<p<0.10) or weak (p>0.10) levels of statistical signficance. This also extends to models where strong collinearities may be present in the explanatory variables. Multiple model runs, with alternative initial coefficient values for the marginally and weakly significant coefficients, may be required to ensure that the estimated coefficient metrics are stable and to increase the likelihood that a global minimization of the model errors is achieved. The models should also be evaluated by excluding statistically insignificant (e.g., p>0.10) and highly collinear variables. For further details, see the guidelines in the sub-section entitled "Model convergence" in Chapter 4.4.4.4.

```
 MODEL ESTIMATION PERFORMANCE (Monitoring-Adjusted Predictions)
MOBS NPARM  DF     SSE       MSE      RMSE       RSQ RSQ-ADJUST RSQ-YIELD PERCENT BIAS
 708    13 695 115.921 0.1667928 0.4084027 0.9532624  0.9524554 0.8485242    -1.888227

 MODEL SIMULATION PERFORMANCE (Simulated Predictions)
MOBS NPARM  DF      SSE       MSE      RMSE       RSQ RSQ-ADJUST RSQ-YIELD PERCENT BIAS
 708    13 695 153.7851 0.2212736 0.4703973 0.9379961  0.9369256 0.7990465     -4.92071

  Simulated predictions are computed using mean coefficients from the NLLS model
    that was estimated with monitoring-adjusted (conditioned) predictions

 PARAMETER SUMMARY
  PARAMETER    ESTIMATE PARM TYPE EST TYPE  INITIAL VALUE MIN    MAX
```

```
1  point         0.80097 SOURCE    Estimated 0.01               0 10000
2  ndep          0.50327 SOURCE    Estimated 0.01               0 10000
3  MANC_N        0.30825 SOURCE    Estimated 0.01               0 10000
4  FARM_N        0.16568 SOURCE    Estimated 0.01               0 10000
5  Fixation      3.02633 SOURCE    Estimated 0.01               0 10000
6  ldrainden     0.14123 DELIVF    Estimated 0.00          -10000 10000
7  PPT30MEAN     0.00161 DELIVF    Estimated 0.00          -10000 10000
8  meanTemp     -0.04442 DELIVF    Estimated 0.00          -10000 10000
9  tiles_perc    1.15268 DELIVF    Estimated 0.00          -10000 10000
10 soil_CLAYAVE  0.01232 DELIVF    Estimated 0.00          -10000 10000
11 rchdecay1     0.39836 STRM      Estimated 0.01               0 10000
12 rchdecay2     0.21842 STRM      Estimated 0.01               0 10000
13 iresload      5.98941 RESV      Estimated 0.01               0 10000

  PARAMETER ESTIMATES
   PARAMETER    PARM TYPE ESTIMATE SE     T      P-VALUE VIF     DESCRIPTION
1  point        SOURCE     0.80097 0.1087  7.371 0.00000 1.10445 Wastewater discharge
2  ndep         SOURCE     0.50327 0.0353 14.255 0.00000 2.46295 Atmospheric deposition N
3  MANC_N       SOURCE     0.30825 0.0592  5.203 0.00000 2.10579 Livestock manure N
4  FARM_N       SOURCE     0.16568 1.0000  0.166 0.86845 7.80026 Fertilizer N use
5  Fixation     SOURCE     3.02633 1.0000  3.026 0.00257 6.22380 Cropland area (fixation)
6  ldrainden    DELIVF     0.14123 0.0571  2.474 0.01360 2.32183 Drainage density
7  PPT30MEAN    DELIVF     0.00161 0.0002  8.404 0.00000 4.03387 Precipitation
8  meanTemp     DELIVF    -0.04442 0.0218 -2.041 0.04168 3.78111 Air temperature
9  tiles_perc   DELIVF     1.15268 0.1016 11.340 0.00000 1.62801 Tile drainage
10 soil_CLAYAVE DELIVF     0.01232 0.0041  3.011 0.00270 1.63543 Soil clay content
11 rchdecay1    STRM       0.39836 0.0870  4.577 0.00001 1.34860 Small stream decay
12 rchdecay2    STRM       0.21842 0.0731  2.988 0.00290 1.36560 Medium stream decay
13 iresload     RESV       5.98941 1.5394  3.891 0.00011 1.25794 Reservoir decay

  EigenValue Spread  Normal PPCC  SWilks W      P-Value   Mean Exp Weighted Error
          52.96279    0.9559388  0.925831 3.354124e-18                 1.084267
```

6.2.8 Model 8: Execution of a SAS SPARROW model in RSPARROW

For users interested in obtaining model performance and coefficient metrics in RSPARROW that are as close as possible to those reported by SAS SPARROW, the mean estimates of the SAS coefficients should be entered (with four or more significant figures) as the initial coefficient values (*parmInit*) for the RSPARROW model estimation. Users should consult the instructions in Chapter sub-section 4.5 for details on the export of the data from SAS and the setup and execution of the SAS SPARROW model in RSPARROW.

To illustrate how the SAS and RSPARROW model results compare, model 6 (the Robertson and Saad 2011 model) was first re-estimated in SAS. The results are shown below in the table.

```
SPARROW model results from a re-estimation of the Robertson and Saad (2011) model in SAS:

  N Obs  DF Model  DF Error    SSE       MSE     Root MSE  R-Square  Adj R-Sq  Yld R-Sq
    708        13       695 115.66357 0.1664224  0.407949 0.9533662  0.952561 0.8488606

              Parameter      Estimate  Std Err   t Value   Pr > |t|  VIF (NC) Min EigVec
              BPCS_NO2_T    0.7893708 0.1125306 7.0147191 5.477E-12 1.1010894  -0.005346
              BTIN_02       0.5130504 0.0398389 12.878114         0 2.9021076  0.0450888
              BMANC_N       0.2909272 0.0554258 5.2489531 2.0345E-7 2.2911542  0.0262912
              BFARM_N       0.1306394 0.0382902 3.4118235 0.0006827 17.354761  -0.714151
              BFIXATION     6.2506424  2.967479 2.1063814 0.0355292 16.555793  0.6896448
```

```
BLDRAINDEN    0.1344048 0.0568554 2.3639781  0.018354 2.7883571  0.0135412
BPPT30MEAN    0.0015921 0.0002679 5.9420546 4.4511E-9 5.4600976  0.0300577
BMEANTEMP     -0.041234 0.0195713 -2.106868 0.0354868 5.2705828  0.0790908
BTILES_PERC   1.1328867 0.1274706 8.8874355         0  1.747011  -0.018006
BSOIL_CLAYAVE 0.0139265   0.00405 3.4386419 0.0006195 1.8781992  0.0611179
BRCHDECAY1    0.4242405 0.1004357 4.2240014 0.0000272 1.4173997  0.0137026
BRCHDECAY2    0.2330996  0.096381 2.4185211 0.0158397 1.4733468  -0.006045
BRESDECAY     6.7102929 1.4529773 4.6183053 4.6098E-6 1.2824011  -0.000091

                    Eigen Sprd Norm PPCC  SWilks W  P-Value  Min Eigval
                     142.19389 0.9599948 0.9248122 2.506E-18  0.0318722
```

Next, model 6 was executed in RSPARROW using the SAS estimated mean coefficients (shown in the table above) as the initial coefficient values (*parmInit* column of the *parameters.csv* file) for the RSPARROW model. The RSPARROW results are reported below, using the model name ("run_id") of model 8. The performance metrics for the RSPARROW and SAS models are virtually identical: SSE (115.6632 vs. 115.6635), RMSE (0.407948 vs. 0.407949), and yield R-squared (0.848861 vs. 0.848861), respectively. These metrics also compare favorably to those for model 6, the RSPARROW model identified as having the lowest RMSE for this specification. This indicates that the use of a larger initial value is required for `Fixation` (e.g, 1.0 rather than 0.01) to obtain model outcomes with this specification that are similar to those in SAS.

The coefficient means reported for model 8 for RSPARROW and SAS are all virtually the same or within one percent of each other, whereas larger differences are observed for the standard errors, which in turn are responsible for any observed differences in the t-statistics and p values.

Compared to SAS SPARROW, the RSPARROW standard errors are within 2 to 13 percent for the land-to-water delivery factors and aquatic decay coefficients, whereas the more statistically significant source coefficients are within 1 to 8 percent of one another. The less statistically significant and moderately collinear source coefficients (`FARM_N`, `Fixation`) show differences in the standard errors of 13 percent.

```
RSPARROW MODEL 8

  MODEL ESTIMATION PERFORMANCE (Monitoring-Adjusted Predictions)
 MOBS NPARM  DF      SSE      MSE      RMSE       RSQ RSQ-ADJUST RSQ-YIELD PERCENT BIAS
  708    13 695 115.6632 0.1664219 0.4079484 0.9533663  0.9525611  0.848861    -1.811554

  MODEL SIMULATION PERFORMANCE (Simulated Predictions)
 MOBS NPARM  DF      SSE       MSE      RMSE      RSQ RSQ-ADJUST RSQ-YIELD PERCENT BIAS
  708    13 695 154.9041 0.2228836 0.4721055 0.937545  0.9364666 0.7975844    -4.676958

   Simulated predictions are computed using mean coefficients from the NLLS model
     that was estimated with monitoring-adjusted (conditioned) predictions

  PARAMETER SUMMARY
    PARAMETER        ESTIMATE PARM TYPE EST TYPE   INITIAL VALUE   MIN   MAX
1   point         0.789478293 SOURCE    Estimated  0.7893708         0 10000
2   ndep          0.513053881 SOURCE    Estimated  0.5130504         0 10000
3   MANC_N        0.290962380 SOURCE    Estimated  0.2909272         0 10000
4   FARM_N        0.130656653 SOURCE    Estimated  0.1306394         0 10000
5   Fixation      6.251067012 SOURCE    Estimated  6.2506424         0 10000
6   ldrainden     0.134403345 DELIVF    Estimated  0.1344048    -10000 10000
7   PPT30MEAN     0.001592016 DELIVF    Estimated  0.0015921    -10000 10000
8   meanTemp     -0.041235668 DELIVF    Estimated -0.0412340    -10000 10000
9   tiles_perc    1.132953147 DELIVF    Estimated  1.1328867    -10000 10000
10  soil_CLAYAVE  0.013925452 DELIVF    Estimated  0.0139265    -10000 10000
```

```
11 rchdecay1     0.424093222 STRM      Estimated  0.4242405         0 10000
12 rchdecay2     0.233019423 STRM      Estimated  0.2330996         0 10000
13 iresload      6.710379414 RESV      Estimated  6.7102929         0 10000

  PARAMETER ESTIMATES
   PARAMETER    PARMTYP ESTIMATE  SE         T        P-VALUE       VIF   DESCRIPTION
1  point        SOURCE  0.7894782 0.1109742  7.114067 2.809752e-12  1.097 Wastewater dischg.
2  ndep         SOURCE  0.5130538 0.0387939 13.225101 0.000000e+00  2.586 Atmospheric dep. N
3  MANC_N       SOURCE  0.2909623 0.0603589  4.820531 1.759621e-06  2.146 Livestock manure N
4  FARM_N       SOURCE  0.1306566 0.0486947  2.683179 7.466072e-03 10.443 Fertilizer N use
5  Fixation     SOURCE  6.2510670 3.8782871  1.611811 1.074570e-01  9.543 Cropland area
6  ldrainden    DELIVF  0.1344033 0.0580737  2.314357 2.093881e-02  2.275 Drainage density
7  PPT30MEAN    DELIVF  0.0015920 0.0002907  5.474763 6.124257e-08  4.185 Precipitation
8  meanTemp     DELIVF -0.0412356 0.0218832 -1.884352 5.993447e-02  3.712 Air temperature
9  tiles_perc   DELIVF  1.1329531 0.1279701  8.853261 0.000000e+00  1.698 Tile drainage
10 soil_CLAYAVE DELIVF  0.0139254 0.0042230  3.297509 1.025123e-03  1.650 Soil clay content
11 rchdecay1    STRM    0.4240932 0.0918879  4.615331 4.674301e-06  1.335 Small stream decay
12 rchdecay2    STRM    0.2330194 0.0904737  2.575548 1.021367e-02  1.357 Medium stream decay
13 iresload     RESV    6.7103794 1.6661960  4.027365 6.261769e-05  1.272 Reservoir decay

  EigenValue Spread  Normal PPCC  SWilks W      P-Value   Mean Exp Weighted Error
           78.34086    0.9595854 0.9273744 5.244195e-18                 1.0825335
```

Although the Jacobian estimates of the first-order partial derivatives for the explanatory variables were generally similar between SAS SPARROW and RSPARROW, moderate numerical differences were observed among individual stations (RSPARROW Jacobian gradients are computed from the R package *numDeriv*). As a result, some differences were observed for the VIF and eigenvalue spread metrics that are derived from the Jacobian estimates. In the model 8 example, the smaller values reported for the VIFs were similar between SAS and RSPARROW, whereas larger VIF values as observed for `FARM_N`, `Fixation`, `PPT30MEAN`, and `meanTemp` showed larger differences, with the RSPARROW values showing somewhat smaller numerical values. In addition, the eigenvalue spread for RSPARROW was about one-half of the value estimated by SAS SPARROW.

Comparisons of the spatial autocorrelation results (Moran's I test) for model 8 for RSPARROW and SAS SPARROW indicated that the test statistic and p-values were very similar in cases where the test results were statistically significant ($p<0.10$). This occurred for model 8 residuals for distance weighting functions (`MoranDistanceWeightFunc` control setting) that gave greater weight to more distant sites (e.g., inverse square root of distance); RSPARROW (t statistic=5.10, p-value=3.286e-07), SAS SPARROW (t statistic=4.75, p-value=1.989e-06). When distance weighting functions were used that give substantially less weight to the residuals for distant sites (e.g., inverse distance, inverse distance squared), the Moran's I test results for both SAS and RSPARROW were statistically insignificant ($p>0.30$) but larger numerical differences were observed in the test statistic values and p-values (e.g., p-values = 0.71 versus 0.31 for the inverse distance function).

Other metrics reported for the residuals as shown above closely match those reported by SAS (i.e., `Normal PPCC`, `SWilks W`, and its `P-Value`) as well as the `Mean Exp Weighted Error` (i.e., bias retransformation correction factor), which is 1.08253 for RSPARROW and 1.08326 for SAS SPARROW.

6.2.9 Evaluating source-change management scenarios

Two versions of the model (model 6 and 3) are used to illustrate the execution of source-change scenarios in the decision support tool using the R Shiny interactive mapper. Settings in section 9 of the control script (and the *userModifyData.R* script) can also be used to execute source-change scenarios, but the R Shiny interactive mapper is more efficient and easier to use. The documentation (Chapter 4.4.9) should be consulted for details about the general approach used to identify watersheds and reaches for scenarios and information on the types of prediction metrics available for output for scenario results.

Execution of source-change scenarios in the interactive mapper and manually in the control script can be applied to source inputs expressed in units of either *mass* (e.g., fertilizer use) or *land area* (e.g., land cover/use). For scenarios with area-based land-use sources, users have the option to convert the land area from one land-use type to another, or users can change the land-use source loading per unit area through changes in the mean estimates of the land-use model coefficients (this latter option can only be executed in the R Shiny interactive mapping interface). For scenarios that involve changes in the area of a land-use source, users are required to select a land use source to receive an equivalent areal change in the opposite direction (i.e., land conversion). This ensures that the total drainage area in the scenario is unchanged; for example, linking reductions in forested land with increases in crop land that are equivalent in areal units.

Source-change scenarios are executed for user-defined watershed locations, source types, and change magnitudes. In the R Shiny interactive mapper, users first specify the reach outlets (i.e., targeted reach locations) for watersheds where the source-change scenarios are to be applied. RSPARROW automatically identifies all hydrologically connected reach segments upstream of the outlet reaches. Users then enter tabular information to specify the details of the source-change scenario attributes, including variable names and conditions for the sources, source-change factors, and reach locations within the targeted watersheds where the scenarios apply.

Note that the effects of a user management scenario are automatically simulated for all hydrologically-connnected reaches located downstream of the user's targeted watersheds, with the most downstream reach location defined by the terminal reaches (e.g., coastal fall-line or international boundaries, inland lakes).

6.2.9.1 Source reductions applied to one source and multiple reach locations

To illustrate the mapper features, a set of hypothetical source-change scenarios illustrate evaluations of the effect of a 25 percent reduction in atmospheric deposition inputs of nitrogen mass (`ndep`) on the mean annual total nitrogen loads in streams, based on the use of model 6. The evaluations show the effects of the source reduction for three separate spatial domains: all reaches in the midwestern modeled region, the Ohio River Basin, and two small watersheds within the Ohio Basin.

Scenario application to all streams in the midwestern modeled region

The first scenario evaluates the effects of source reductions applied uniformly across all reaches in the modeled domain of midwestern streams. The Shiny interactive mapper interface displays the settings for this example (see Fig. 26).

The *"Select Target Reach Watersheds"* setting of *"default"* selects the terminal reaches for the Midwest model as the targeted watershed outlet reaches. All hydrologically connected upstream reaches are automatically selected for potential use in the scenario.

The setting *"Select reaches for applying the scenario (within the targeted watersheds)"* controls the specification of additional details of the scenarios, including the sources, source-change factors, and reach locations within the targeted watersheds for applying the scenarios. The selection of *"all reaches"* applies the scenario to all reaches within the targeted waterheds.

The *"Select Sources and Percent Change (+/-) Factors"* tabular interface allows users to select from one to all model sources for applying the scenario; in this case, the deposition source `ndep` is selected. A *"-25"* value is entered for the *"PercentChange"* tab, which corresponds to a 25% reduction in the atmospheric nitrogen deposition source entering streams. A right click on the table allows a user to add a source and specify additional sources to which a change percentage can be applied.

The selected *"Mapping Variable Type"* is a measure of the relative change in load, with the specific metric *ratio_total* selected. This measures the large watershed-scale effect of the scenario on the total load, inclusive of the mass contributed from all upstream sources and reaches. The prediction metric is computed as the ratio of the updated model prediction of load (resulting from the changed sources) to the baseline (unchanged) model load prediction. A complementary incremental ratio is also provided as an optional metric to measure of the "local" effect of the scenario on loads delivered to streams from the incremental drainage area associated with individual reaches.

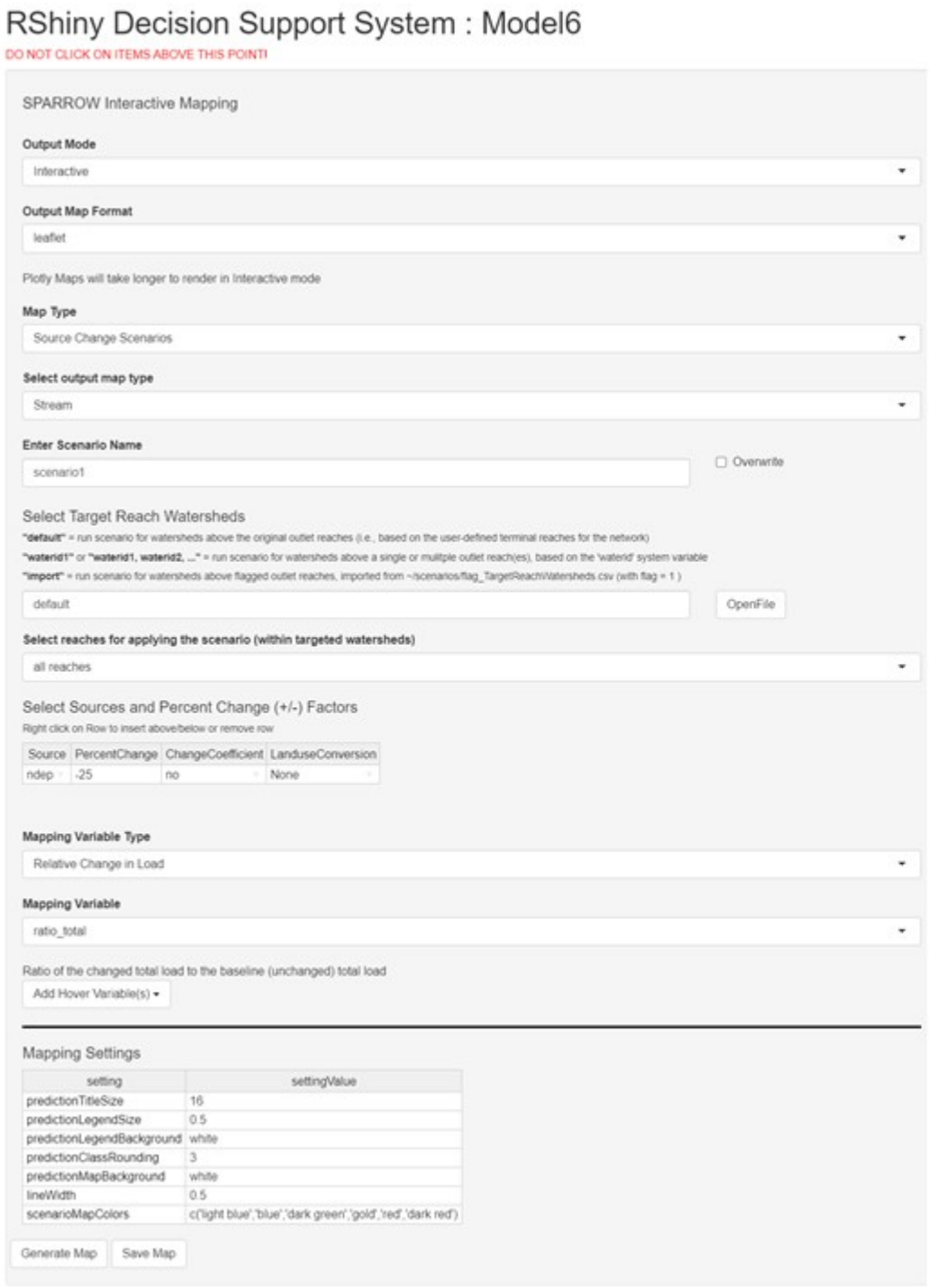


Figure 26: **R Shiny interactive mapper interface with settings to evaluate a 25 percent reduction in the atmospheric deposition nitrogen source in all stream reaches for model 6.**

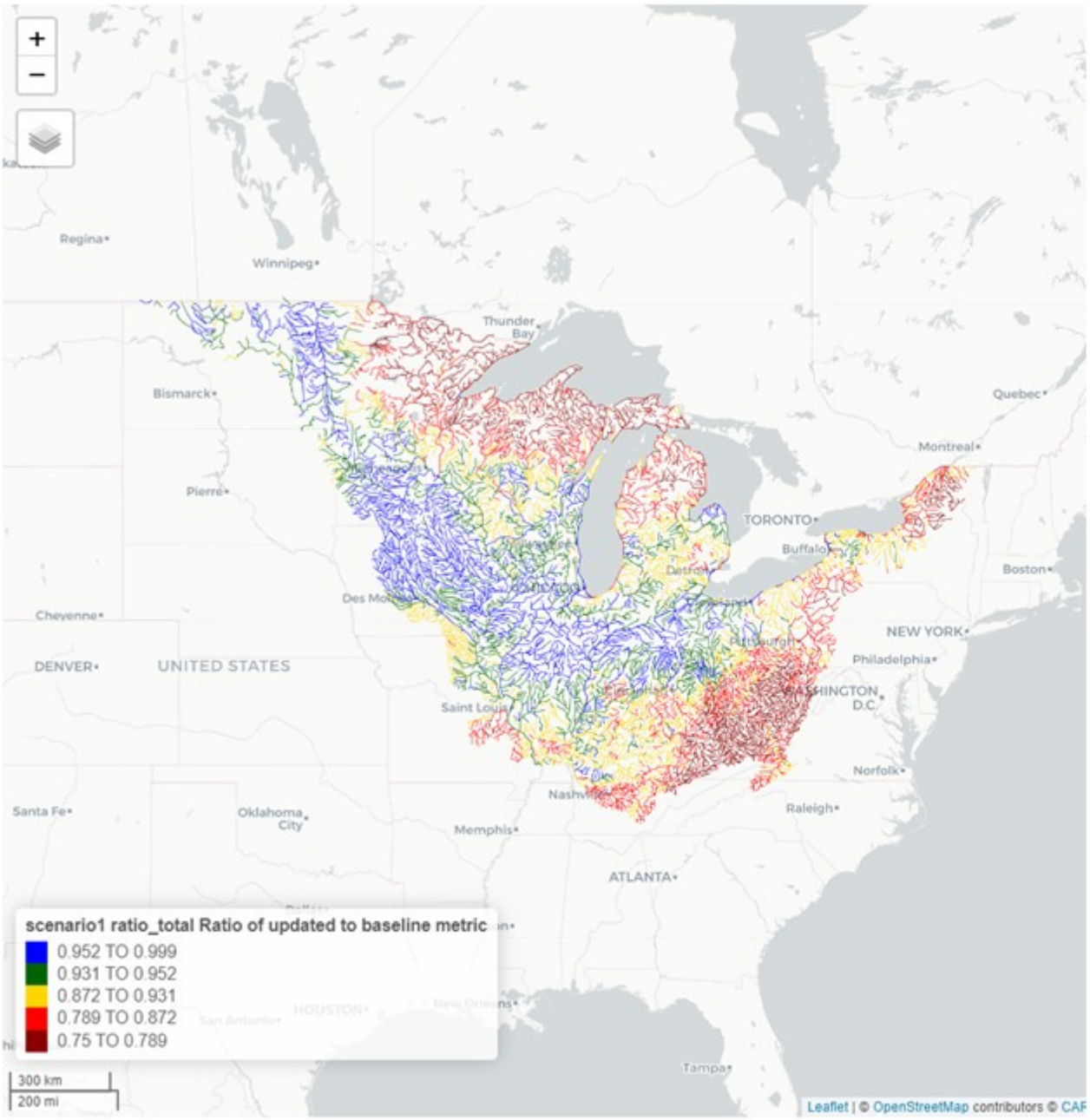


Figure 27: **R Shiny interactive map of the modeled domain, displaying the relative effects on total nitrogen load of a 25 percent reduction in the atmospheric deposition nitrogen source in all reaches for model 6. The load effects are reported as a ratio of the updated (changed) model prediction of load to the baseline model load prediction. The R Shiny control settings for the evaluated scenario conditions are shown in the previous figure.**

Execution of the scenario displays a map (see Fig. 27) of the relative measure of the hypothetical effect of a 25 percent reduction in atmospheric nitrogen deposition applied uniformly across all streams in the Midwest region. Ratios of 0.75 are expected to appear in streams where nitrogen deposition is the only or predominant upstream source. In streams where other sources contribute larger quantities of nitrogen, ratios higher than 0.75 would be expected.

A variety of functional controls are available in the Shiny menu (*"Mapping Settings"*) to allow customization of the map background color, binning breakpoints and intervals, colors, numerical rounding, and text and symbol/line sizes, as shown in Fig. 26. The number of breakpoint intervals is determined by specifying the number of colors (*"scenarioMapColors"*). For scenario maps, the first color selected applies to the "No change" class for streams (selection of a light background color for this class is recommended–e.g., light blue)

Scenario application to the Ohio River Basin

This example scenario is applied only to streams in the Ohio River Basin. The scenario settings are shown in Figure 28. The *"Select Target Reach Watersheds"* setting of *"default"* uses the terminal reaches for the Midwest model as the targeted downstream watershed outlet reaches.

To restrict the application to the Ohio Basin, the option *"selected reaches"* is used for the setting *"Select reaches for applying the scenario (within the targeted watersheds)"*. This choice provides users with two options for entering the scenario attributes via the setting *"Apply same reach selection criteria to all selected sources (yes/no)"*. A *"yes"* response is selected, which applies the same reach selection to all sources using two separate tables (one for sources and one for the reach selection criteria); note that a *"no"* response allows users to apply different reach selection criteria and source-change percentages to each source in the same table. In the *"Reach Selection Criteria"* table, the reach identifier variable *"huc2"*, a hydrologic unit code for USA watersheds, is used to select reaches in the Ohio Basin (i.e., huc2=5).

Results for a 25% reduction in the nitrogen deposition source (`ndep`) in the reaches of the Ohio Basin are shown in Figure 29. No reduction is applied to all other reaches outside of the Ohio River Basin, as indicated by a light blue color.

Scenario application to the Wabash River and New River watersheds

This example scenario evaluates the effect of a reduction in nitrogen deposition on total loads in two selected watersheds (i.e., sub-basins) within the Ohio River Basin. The settings for this scenario are shown in Figure 30.

In this case, the *waterid* values associated with the outlet reaches of the two watersheds are specified for the *"Select Target Reach Watersheds"* setting. Note that the *waterid* values for these outlet reaches can be found by searching for the name of the watersheds associated with the *rchname* system variable column in the *run_id/scenarios/flag_TargetReachWatersheds.csv* file and locating the reach with the largest total drainage area for the *demtarea* system variable column.

The option *"all reaches"* is used for the setting *"Select reaches for applying the scenario (within the targeted watersheds)"*, which applies the scenario conditions to all reaches within the two selected watersheds in the Ohio Basin.

Results for a 25% reduction in the nitrogen deposition source (`ndep`) are shown in Figure 31. The results are reported for the *ratio_total* prediction metric. This provides relative measure of the watershed-scale effect of the scenario on the total load contributed from all upstream sources and reaches in the two watersheds.

One important attribute of the *ratio_total* metric is its utility for evaluating the downstream effect of the management scenario on nearby and distant reaches and waterbodies that are located downstream of the watersheds where the management scenario is applied. In the example, the relative effects of nitrogen deposition reductions in the two watersheds are shown for all downstream reaches to the terminal reach location on the Mississippi River. The results (Fig. 31) demonstrate the diminishing downstream effects of a hypothetical source reduction in the two watersheds. These diminishing effects occur because the magnitude of the load contributions from unchanged (i.e., unmanaged) sources and watersheds increases in downstream reaches that have larger contributing drainage areas.

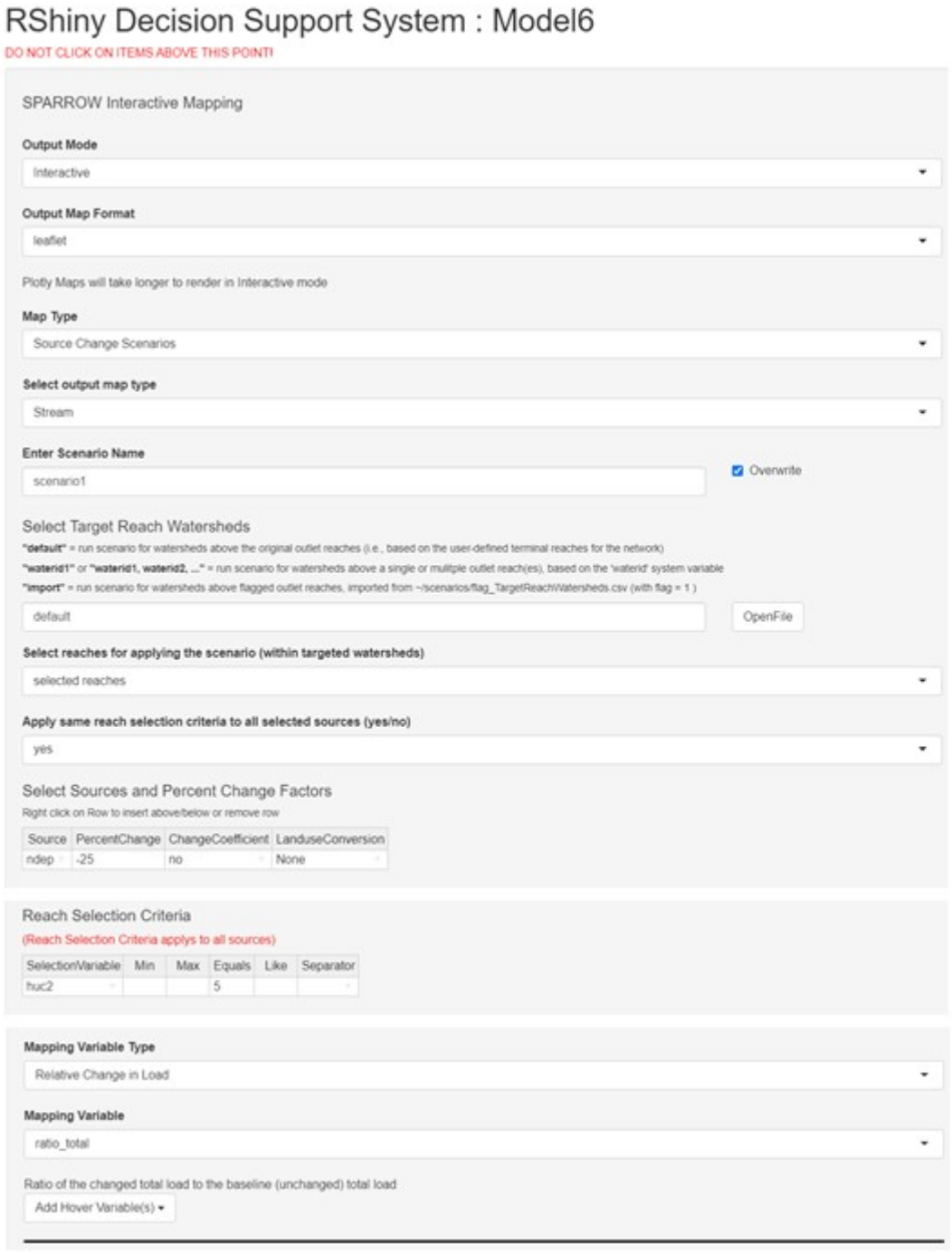


Figure 28: **R Shiny interactive mapper interface with settings to evaluate a 25 percent reduction in the atmospheric deposition nitrogen source in stream reaches of the Ohio River Basin (huc2=5) for model 6.**

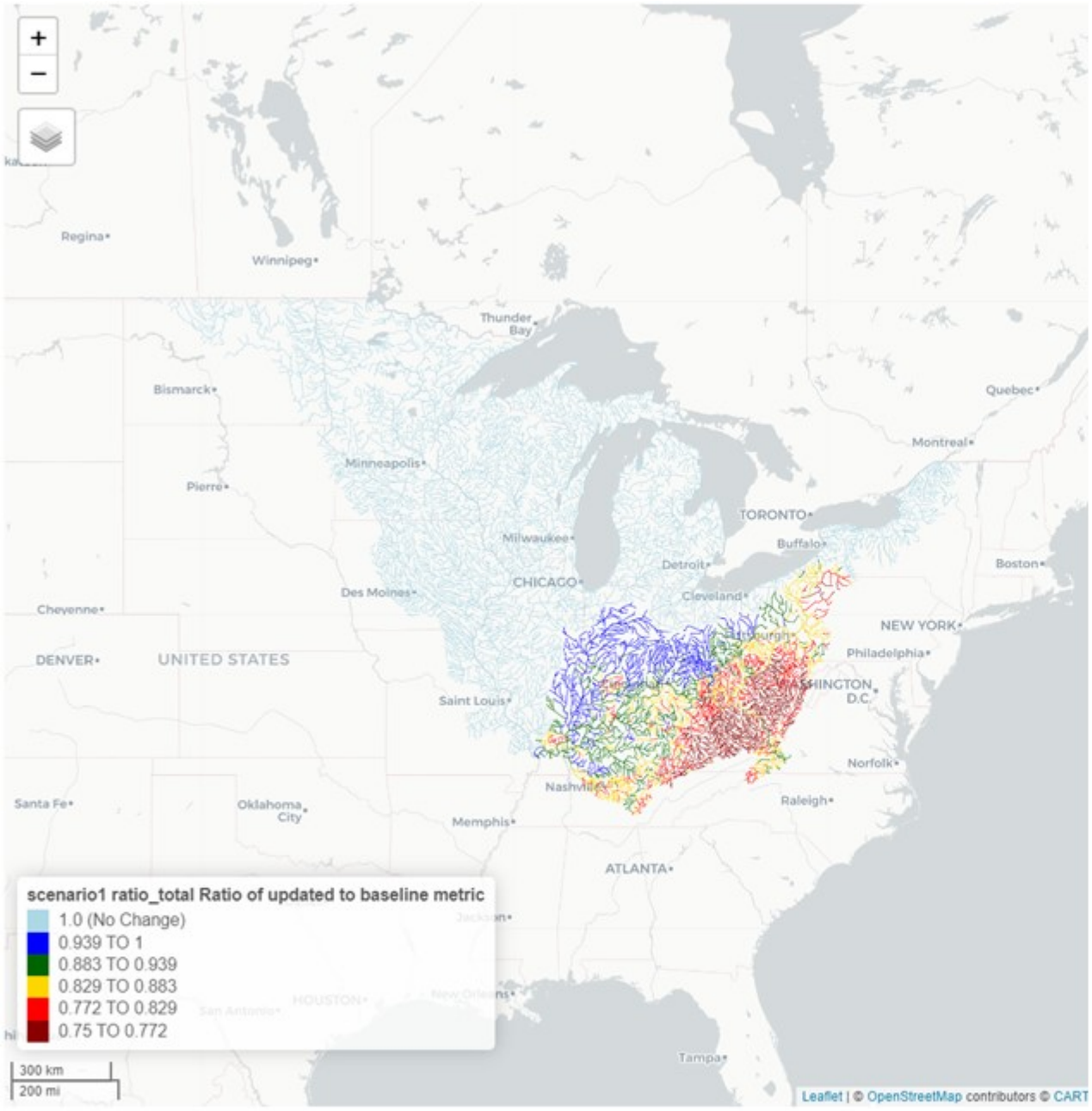


Figure 29: **R Shiny interactive map of the modeled domain, displaying the relative effects on total nitrogen load of a 25 percent reduction in the atmospheric deposition nitrogen source in stream reaches in the Ohio River Basin (huc2=5) for model 6. The load effects are reported as a ratio of the updated (changed) model prediction of load to the baseline model load prediction. The R Shiny control settings for the evaluated scenario conditions are shown in the previous figure.**

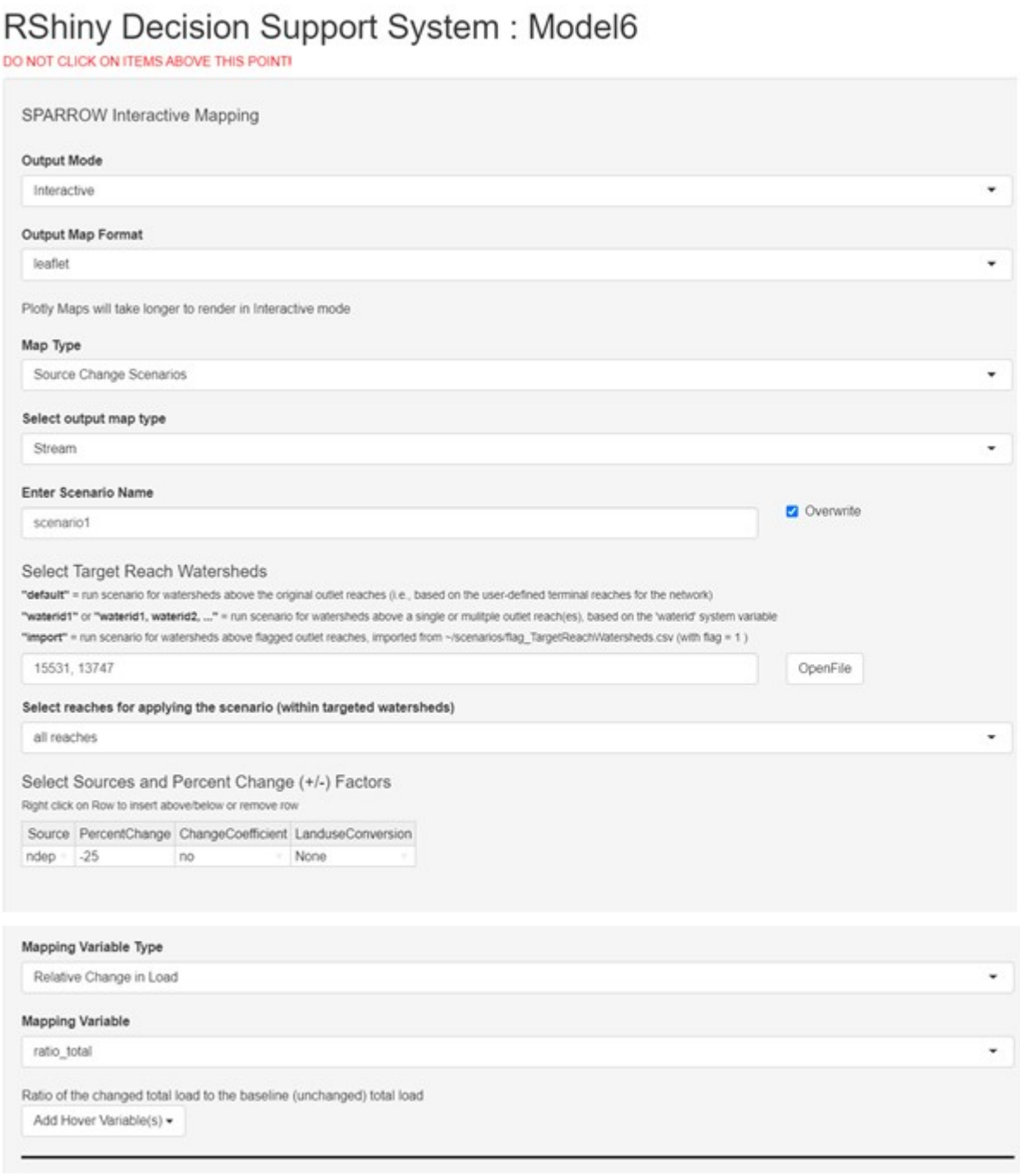


Figure 30: **R Shiny interactive mapper interface with settings to evaluate a 25 percent reduction in the atmospheric deposition nitrogen source in stream reaches of the Wabash River (waterid=15531) and New River (waterid=13747) Basins for model 6.**

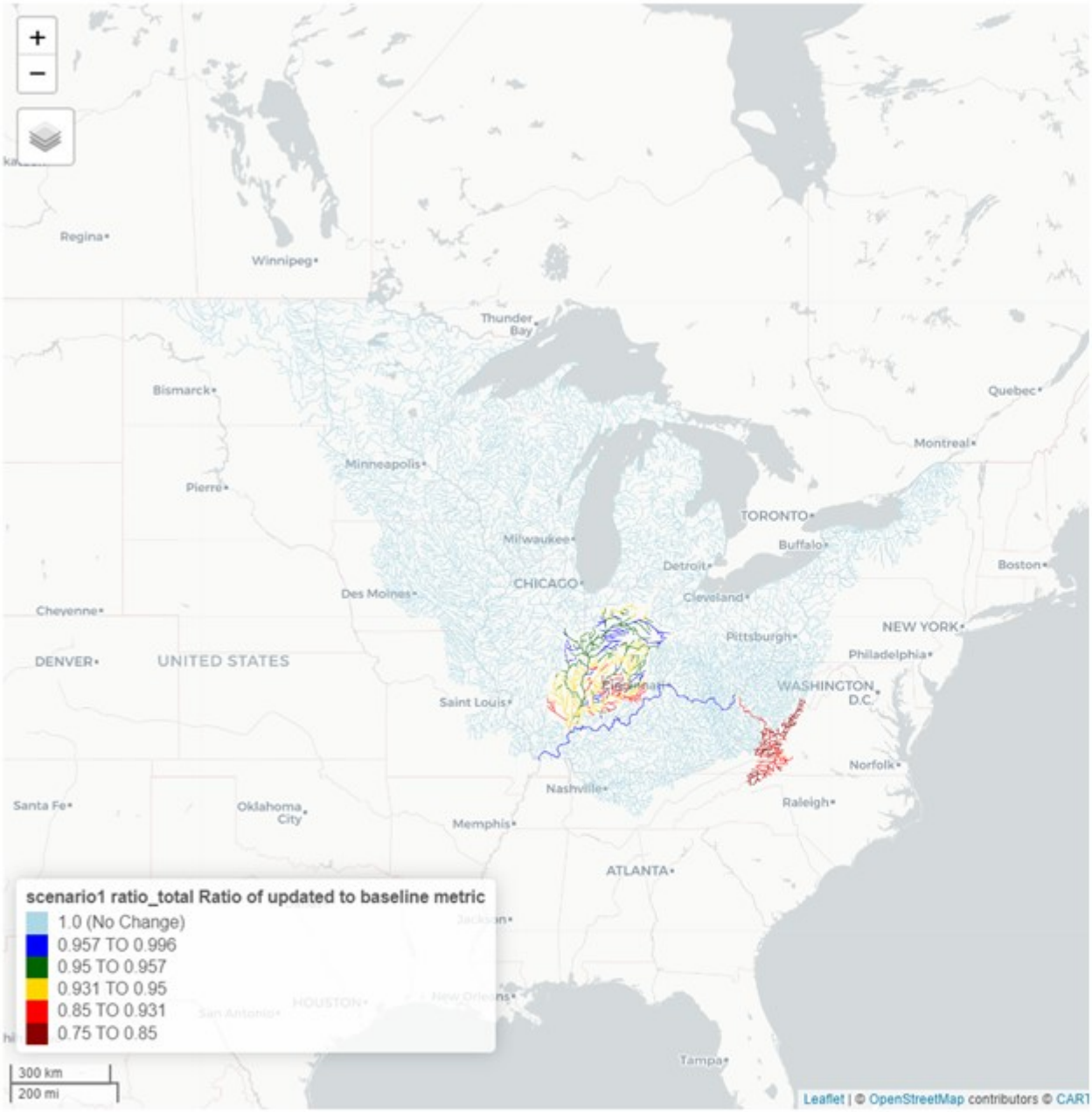


Figure 31: **R Shiny interactive map of the modeled domain, displaying the relative effects on total nitrogen load of a 25 percent reduction in the atmospheric deposition nitrogen source in stream reaches in the Wabash River (waterid=15531) and New River (waterid=13747) Basins for model 6. The load effects are reported as a ratio of the updated (changed) model prediction of load to the baseline model load prediction. The R Shiny control settings for the evaluated scenario conditions are shown in the previous figure.**

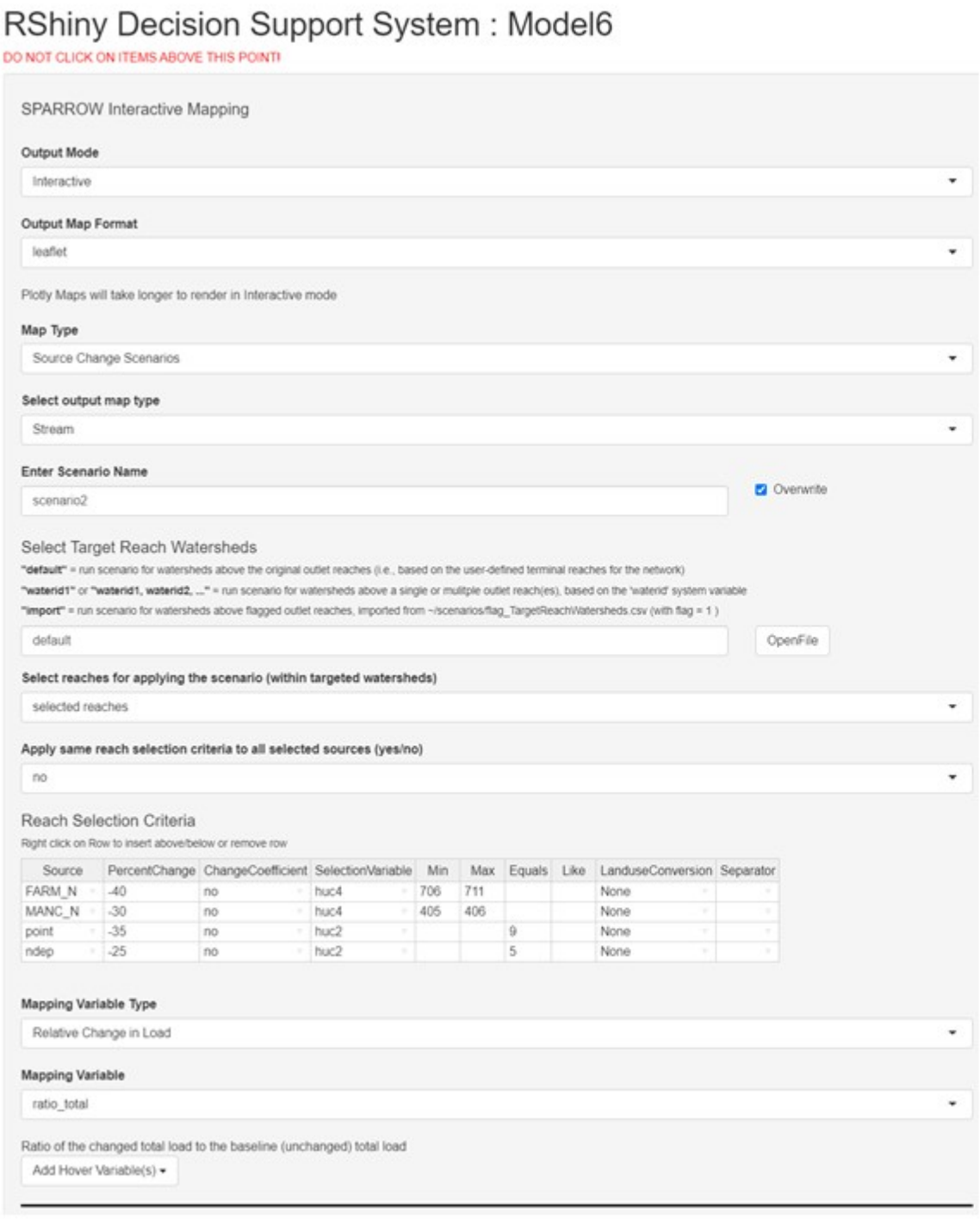


Figure 32: **R Shiny interactive mapper interface with settings to evaluate multiple sources, source-change percentages, and reach locations for the midwestern total nitrogen model 6.**

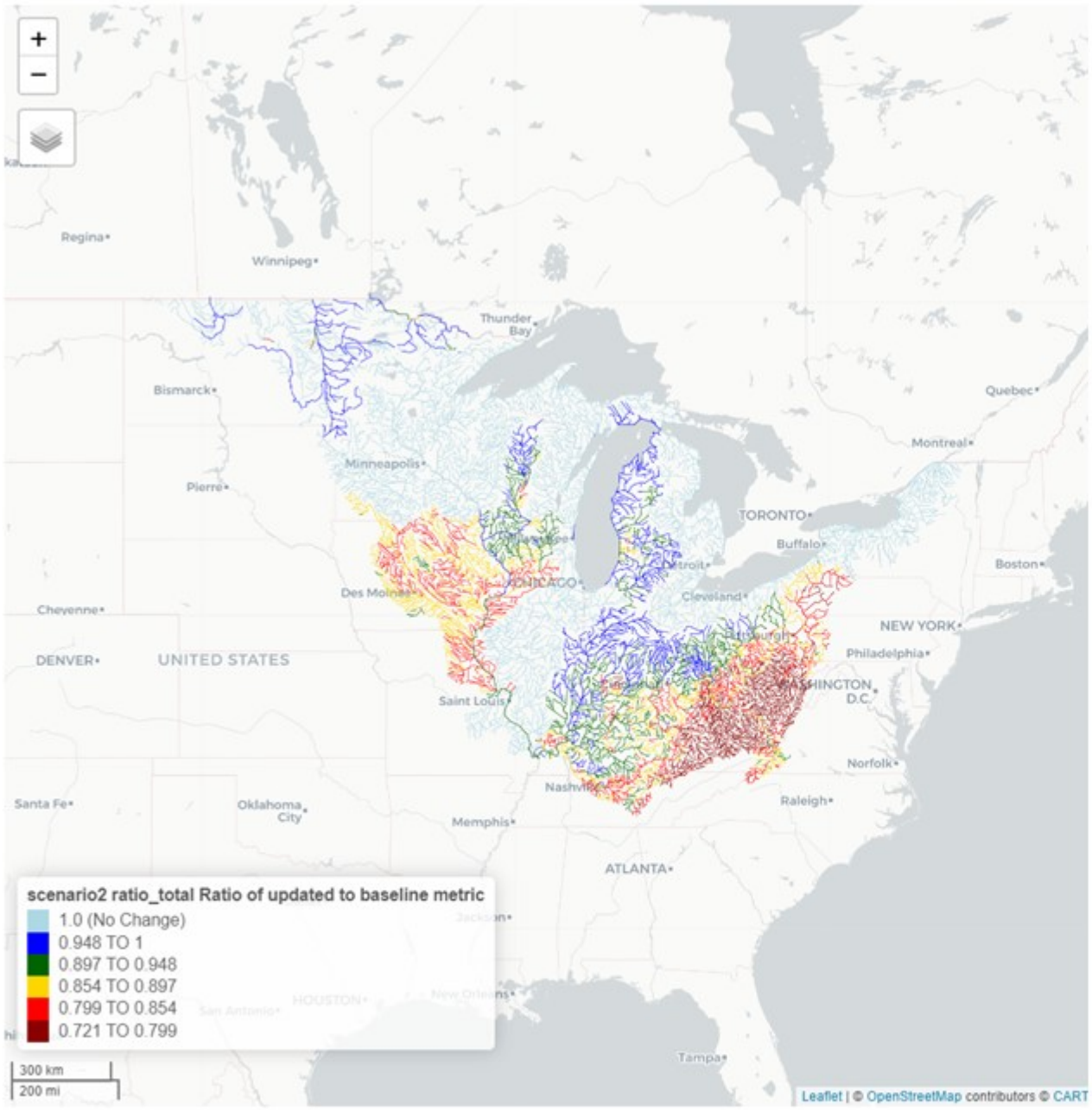


Figure 33: **R Shiny interactive map of the modeled domain, displaying the relative effects on total nitrogen load of hypothetical reductions in multiple sources and reach locations. The load effects are reported as a ratio of the updated (changed) model prediction of load to the baseline model load prediction. The R Shiny control settings for the evaluated scenario conditions are shown in the previous figure.**

6.2.9.2 Source reductions applied separately to multiple sources and reach locations

In this example, the effects on stream loads of hypothetical source reductions are illustrated for a complex set of conditions that include multiple sources with different reduction percentages applied to multiple river basins (see Fig. 32). The scenario is applied using model 6 and includes the following conditions:

- 40% reduction in farm nitrogen fertilizer (`FARM_N`) in central reaches of the Upper Mississippi Basin (huc4 0708 to 0711)
- 30% reduction in confined manure nitrogen (`MANC_N`) in a portion of the Great Lakes basin in western Michigan (huc4 0405 to 0406)
- 35% reduction in municipal point-source nitrogen (`point`) in the Red-River Basin, located in the northern portion of the model domain (huc2 = 9)
- 25% reduction in atmospheric nitrogen deposition (`ndep`) in stream reaches of the Ohio River Basin, located in the eastern portion of the model domain (huc2 = 5)

The source reductions are specified for multiple sources, source reduction percentages, and locations by specifying the option *"selected reaches"* for the setting *"Select reaches for applying the scenario (within the targeted watersheds)"*. Additionally, the option *"no"* is specified for the setting *"Apply same reach selection criteria to all selected sources (yes/no)"*. This response enables a user to apply different reach selection criteria and source-change percentages to each source in the same table. Furthermore, in the *"Reach Selection Criteria"* table, the reach identifier variables *"huc2"* and *"huc4"*, hierarchical hydrologic unit codes for different sized watersheds, are used to select reaches for inclusion in the scenario.

Results of the hypothetical source reductions on stream loads are shown in Figure 33. The largest reductions in stream nitrogen loads, associated with the reductions in sources, are 28% (i.e., ratio of the updated load to the baseline load of 0.72). Many of the larger reductions occur in watersheds of the Ohio and Upper Mississippi River Basins, where atmospheric deposition and farm fertilizer are predominant sources, respectively.

6.2.9.3 Scenario changes applied to the area associated with land-use sources

This example illustrates the effects on stream loads of hypothetical changes in the area associated with selected land-use sources. The application of this change scenario results in a land-use conversion of the drainage area, such that the total land area is unchanged in the scenario. For example, this could include scenarios that evaluate the effects on stream water quality of a hypothetical decrease in forested land and conversion of this land area to urban land.

The control settings for this scenario are shown in Figure 34. Results of the relative effects on total load (*ratio_total*) of the hypothetical reduction in crop land area are shown in Figure 35.

Using the land-use model 3 for the Midwest, the scenario applies a 30% reduction in crop land area (`crops`), with the affected crop land areas converted to a `shrubgrass` cover by specifying the *LanduseConversion* land cover type as `shrubgrass`. RSPARROW ensures that the total drainage area is unchanged in the scenario simulation, with the area reduction in crop land (associated with the 30% reduction) corresponding to an equivalent increase in the area of shub-grass land cover.

Furthermore, the scenario is only applied to reaches in the watersheds of the Upper Mississippi regional basin (i.e., huc2=7), with the terminal reaches used to identify the targeted reach watersheds. This is enabled by the following selections:

- The *"default"* option is specified for the *"Select Target Reach Watersheds"* setting, such that the terminal reaches for the modeled spatial domain are used in the scenario to identify the upstream targeted watersheds. This option is used as a convenience and avoids any need to enter the specific *waterid* for the outlet reach of the Upper Mississippi River Basin; instead, the reach hydrologic identifier, *huc2*, is used to designate the reaches where the scenario is applied.
- The *"selected reaches"* option is specified for the setting *"Select reaches for applying the scenario (within the targeted watersheds)"*, with the option *"no"* specified for the setting *"Apply same reach selection*

criteria to all selected sources (yes/no)". This control setting allows the source type, source-change percentage, and reach selection criteria to be specified in the same table. The hydrological *huc2* variable is used as the reach *SelectionVariable* in the table to restrict the scenario to the Upper Mississippi River basin (i.e., huc2=7).

6.2.9.4 Scenario changes applied to the mass associated with land-use sources

This example demonstrates a scenario that evaluates the downstream effects on stream loads of hypothetical changes in the contaminant mass that is exported by a land-use source. Thus, the scenario evaluates changes in the contaminant loading per unit area of a land-use source. This has relevance to management scenarios that might, for example, evaluate the effects on stream water quality of the implementation of "best management practices" on pasture land or the construction of wastewater treatment factilities, which would be expected to reduce the contaminant loadings to streams. The decision tool implements these land-use based scenarios by changing the SPARROW estimated model coefficient associated with a given land-use source according to the user-specified percentage change value.

The control settings for this scenario are shown in Figure 36. Results of the relative effects on total load (*ratio_total*) of the hypothetical reduction in nitrogen load from the pasture land area source are shown in Figure 37.

Using the land-use model 3 for the Midwest, the scenario applies a 30% reduction in the nitrogen export from the pasture land source (`pasture`) in the model. The mean nitrogen export associated with pasture land was originally estimated to be 724.9 kg/km2/year. Thus, under a 30% reduction scenario, the mean nitrogen export would be 507.4 kg/km2/year.

As in the previous example (sub-section 6.2.9.3), the scenario is only applied to reaches in the watersheds of the Upper Mississippi regional basin (i.e., huc2=7), with the terminal reaches used to identify the targeted reach watersheds. The following selections were made to enable the reach targeting:

- The *"default"* option is specified for the *"Select Target Reach Watersheds"* setting, such that the terminal reaches for the modeled spatial domain are used in the scenario to identify the upstream targeted watersheds. This option is used as a convenience and avoids any need to enter the specific *waterid* for the outlet reach of the Upper Mississippi River Basin; instead, the reach hydrologic identifier, *huc2*, is used to designate the reaches where the scenario is applied.
- The *"selected reaches"* option is specified for the setting *"Select reaches for applying the scenario (within the targeted watersheds)"*, with the option *"no"* specified for the setting *"Apply same reach selection criteria to all selected sources (yes/no)"*. This control setting allows the source type, source-change percentage, and reach selection criteria to be specified in the same table. The hydrological *huc2* variable is used as the reach *SelectionVariable* in the table to restrict the scenario to the Upper Mississippi River basin (i.e., huc2=7).

In the *"Reach Selection Criteria"* menu, note that "yes" is entered for the *"ChangeCoefficient"* to enable the application of the change percentage to the model coefficient for the pasture land source. Also, "None" is entered for the *"LanduseConversion"* because no changes in the area of the source are applied for this scenario.

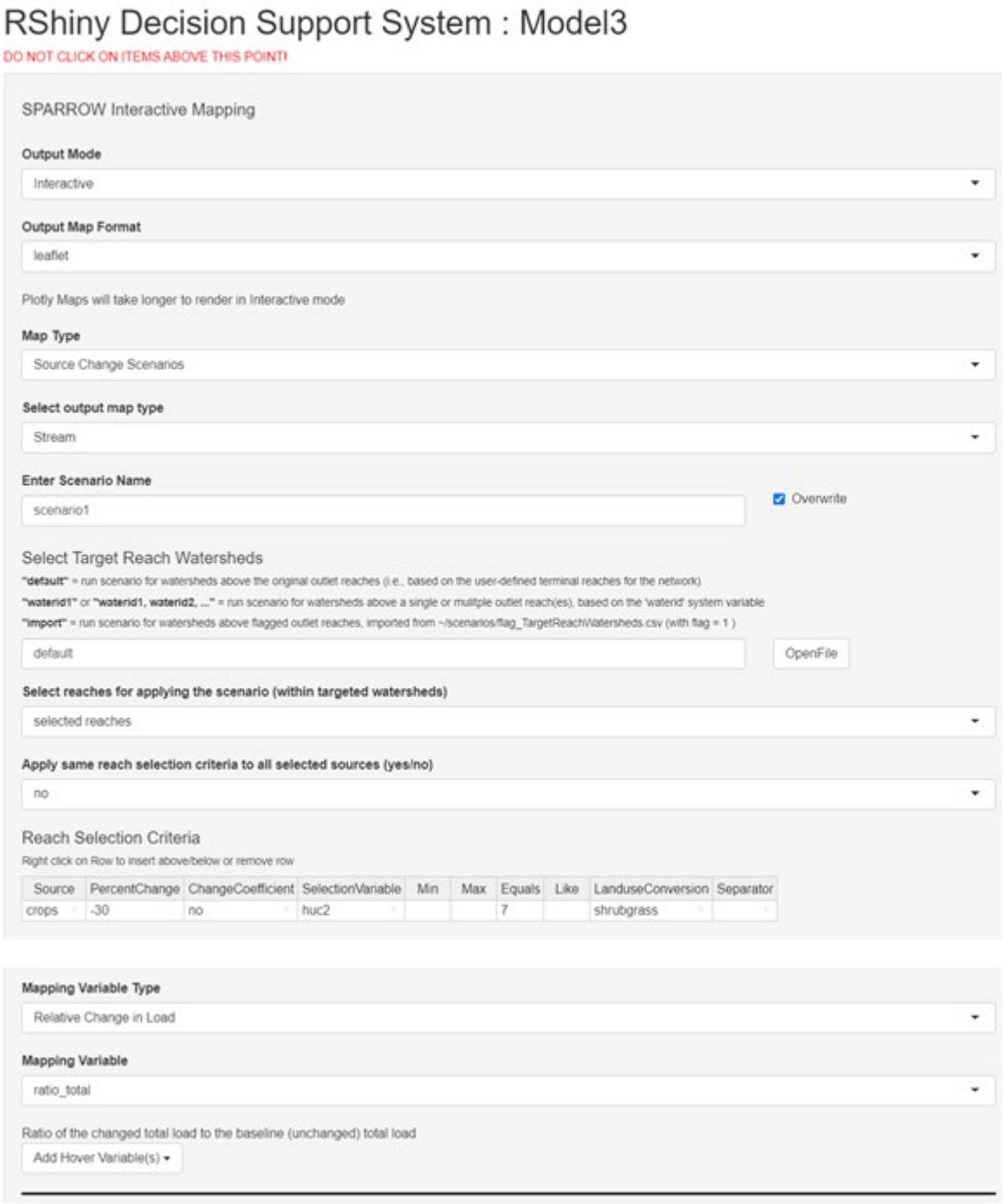


Figure 34: **R Shiny interactive mapper interface with settings to evaluate a 30 percent reduction in the cropland area source in stream reaches of the Upper Mississippi River Basin (huc2=5) for model 3.**

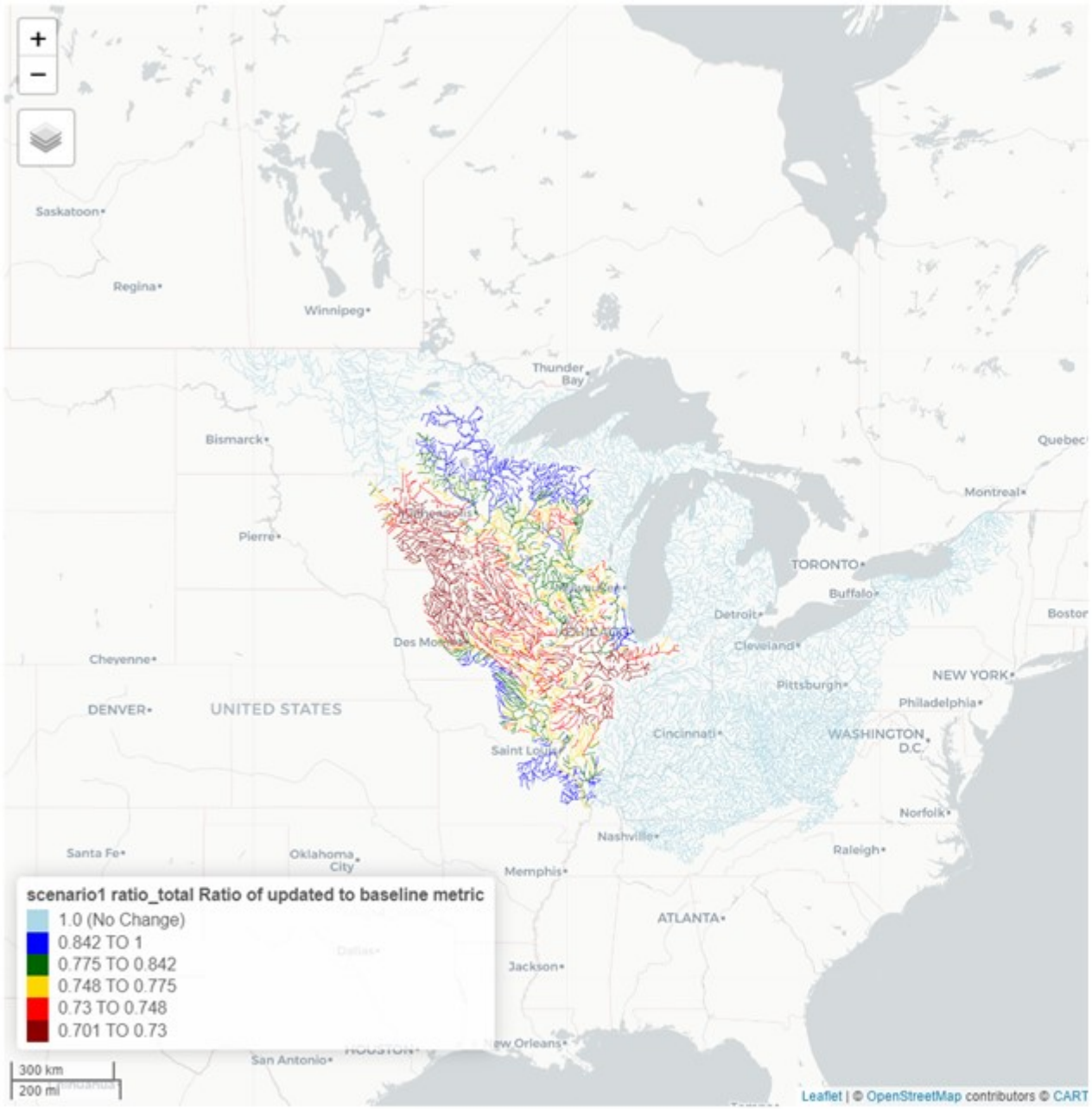


Figure 35: **R Shiny interactive map of the modeled domain, displaying the relative effects on total nitrogen load of a 30 percent reduction in the cropland area source in stream reaches in the Upper Mississippi River basin for model 3. The load effects are reported as a ratio of the updated (changed) model prediction of load to the baseline model load prediction. The R Shiny control settings for the evaluated scenario conditions are shown in the previous figure.**

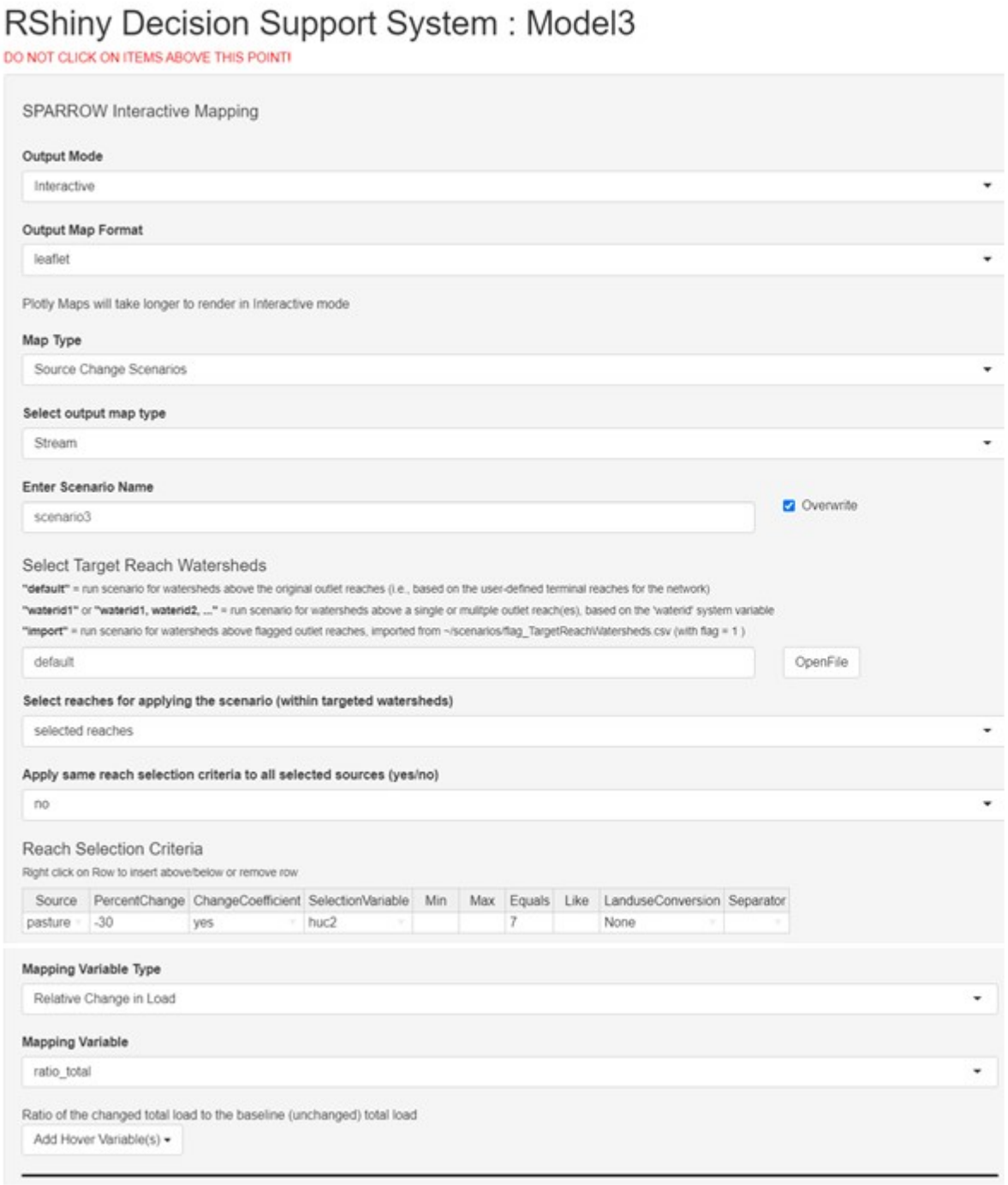


Figure 36: **R Shiny interactive mapper interface with settings to evaluate a 30 percent reduction in export of nitrogen from the pasture land source in stream reaches of the Upper Mississippi River Basin (huc2=5) for model 3.**

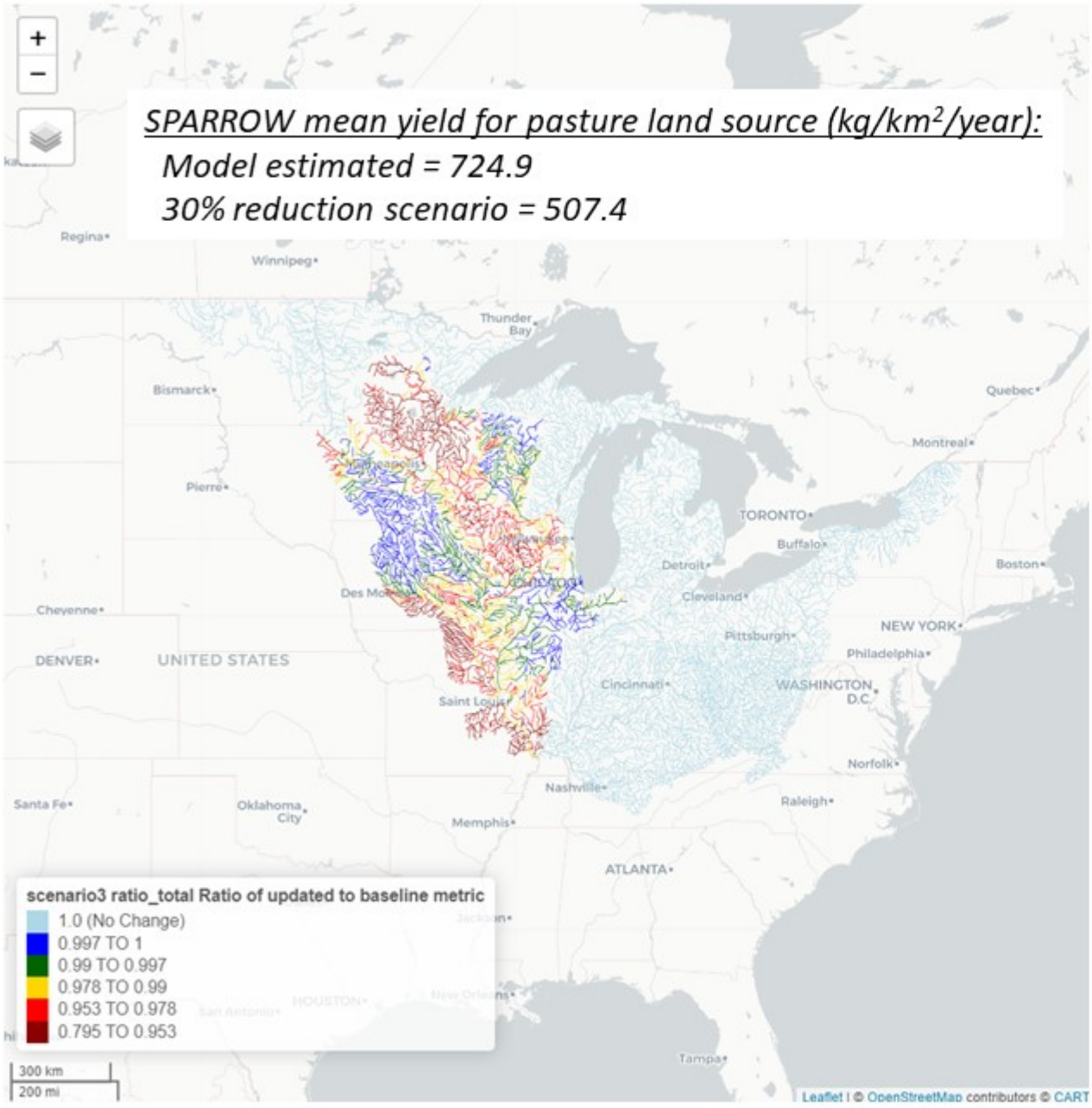


Figure 37: **R Shiny interactive map of the modeled domain, displaying the relative effects on total nitrogen load of a 30 percent reduction in nitrogen load from the pasture land source in stream reaches in the Upper Mississippi River basin for model 3. The load effects are reported as a ratio of the updated (changed) model prediction of load to the baseline model load prediction. The R Shiny control settings for the evaluated scenario conditions are shown in the previous figure.**

6.3 Illustration of the automated creation of model input control files

The supplementary material in this section is presented to illustrate the types of messages that are ouput to the Console window when users employ control settings in section 1 of the control script to automatically create the dataDictionary and parameter input files from the *data1.csv* file.

Note that the data and parameter input control files for the SPARROW tutorial model are provided to users in final form and require no setup using the automated control settings illustrated in this section. To execute the steps below, users will need to temporarily place the existing final copies of the *dataDictionary.csv*, *parameters.csv*, and *design_matrix.csv* files in another directory and delete the files in the *User Directory/results* sub-directory.

Users should also review step 3 in the checklist in Chapter sub-section 4.2.4 for details on the setup of the input control files.

6.3.1 Creating the dataDictionary.csv file

Users can optionally create the data dictionary file using the following control setting in section 1 of the control script.

```
# Control script:
   #Create an initial Data Dictionary file from the data1 column names
   #This file will have to be edited prior to executing RSPARROW
   create_initial_dataDictionary<-"yes"
```

This setting will print the following messages to the Console window, and create the *dataDictionary.csv* file.

```
NO PARAMETERS FILE FOUND IN RESULTS DIRECTORY.
~/results/

Did you save the active control file
~\results\sparrow_control.R
and all '*.csv' control files?

1: Yes, I have saved all control files. Continue the current run.
2: No, I haven't saved all control files.  Cancel current run.

MISSING REQUIRED sparrowNames :
waterid
calsites

MISSING FIXED sparrowNames :
rchname
depvar
depvar_se
target

INITIAL dataDictionary FILE :
~/results/dataDictionary.csv AVAILABLE FOR EDIT

RUN EXECUTION TERMINATED
```

Execution of the control script with the setting `create_initial_dataDictionary<-"yes"` automatically creates the *dataDictionary.csv* file in the *"results"* sub-directory, using column header names in the user's `input_data_fileName` (e.g., *data1.csv* file) and internal RSPARROW system variable names. Additionally, the column information for the REQUIRED and FIXED system variable types (*varType*) are filled-in in cases where the *data1.csv* header names match the pre-defined sparrowNames for these variable types (as shown in Table 4; note that this is not required but will make the process faster). Where no match is found for these

variable types, an NA will appear in the *data1UserNames* column. All other variables in the *data1.csv* file will be assigned to the *data1UserNames* column with NA values for all other columns.

Additional editing is required to fill-in the NA values by completing the variable types (*varType*) and internal R variable names (*sparrowNames*). Users should populate missing variables in the *sparrowNames* column with the most appropriate *data1UserNames* variable name from the *data1.csv* file.

At this stage, it is also recommended that users add the *varType* for the largest possible set of model explanatory variables to the dictionary (i.e., SOURCE, DELIVF, STRM, and RESV) before attempting to create the parameter control files (at a minimum, one SOURCE and DELIVF *varType* must be added to the file).

Users have the option at this step, or in a later step, to add variable names to the *sparrowNames* column that are not present in the *data1.csv* file but created in the *userModifyData* script; a NA should be assigned to the *data1UserNames* for these variables. This includes the definition of land-use variables, which are used in the model diagnostics and are recommended. The variable units (*varunits*) and descriptive information (*explanation*) are optional but recommended at this stage to provide full documentation.

6.3.2 Creating the parameter input control files: parameters.csv and design_matrix.csv

For this illustration, users can replace the *dataDictionary.csv* with the version of the *dataDictionary.csv* in the *model1* subdirectory.

The following control setting in section 1 of the control script allows users to optionally create the two control input files.

```
# Control script:
   #Create an initial parameter and design_matrix files from names in the Data Dictionary file
   #The parameter names must be listed for both the sparrowNames and data1UserNames and the
   #  varType should be defined as SOURCE, DELIVF, STRM, or RESV to populate parmType in the
   #  (run_id)_parameters.CSV
   #The initial file will have to be edited prior to executing RSPARROW
   create_initial_parameterControlFiles<-"yes"
```

Provided at least one SOURCE *varType* variable is defined in the *dataDictionary.csv* file, the parameter control files will be created upon subsequent execution of the control script, with the following messages appearing in the Console window (note that at least one DELIVF *varType* is required to subsequently execute reach connectivity checks or to convert geographic shape files to R binary object files as required for all RSPARROW mapping).

```
# Console messages:

NO PARAMETERS FILE FOUND IN RESULTS DIRECTORY.
~/results/

Did you save the active control file
~\results\sparrow_control.R
and all '*.csv' control files?

1: Yes, I have saved all control files. Continue the current run.
2: No, I haven't saved all control files.  Cancel current run.

Selection: 1

INITIAL DESIGN_MATRIX FILE : ~/results/
    design_matrix.csv AVAILABLE FOR EDIT

INITIAL PARAMETERS FILE : ~/results/
```

```
    parameters.csv AVAILABLE FOR EDIT

RUN EXECUTION TERMINATED
```

The execution of the control script with the setting `create_initial_parameterControlFiles<-"yes"` automatically creates the *parameters.csv* and *design_matrix.csv* control input files in the *"results"* sub-directory, using the contents of the *dataDictionary.csv* file. Execution also fills-in in the *sparrowNames* and *parmType* columns in the CSV files using the *dataDictionary.csv* definitions.

Note that additional editing of the columns and cells in the created parameter files will not be necessary to convert geographic shape files to R binary object files (section 8 of the control script) and to verify reach network connectivity (section 2 of the control script). However, additional editing will be required to the columns in these files to execute user-specified models (see Chapter sub-section 4.4.4). The *parmUnits* and *description* columns in the *parameters.csv* file are optional but recommended.

Finally, note that a parameter *varType* must be created for at least one SOURCE variable in the *dataDictionary.csv* file for the parameter control files to be automatically created using the control setting `create_initial_parameterControlFiles<-"yes"`. Failure to to do this will cause the error message illustrated below to appear and will cause the script execution to terminate.

```
# Console messages:

NO PARAMETERS FILE FOUND IN RESULTS DIRECTORY.
~/results/

Did you save the active control file
~\results\sparrow_control.R
and all '*.csv' control files?

1: Yes, I have saved all control files. Continue the current run.
2: No, I haven't saved all control files.  Cancel current run.

NO VALID varTypes FOUND.

NEW PARAMETER CONTROL FILES NOT CREATED.

SET varType = 'SOURCE','DELIVF','STRM', or, 'RESV' TO CREATE PARAMETER CONTROL FILES.

RUN EXECUTION TERMINATED.
```

7 RSPARROW design elements for experienced R users and developers

RSPARROW was developed as a system of interconnected R functions and scripts written according to standardized open-source R protocols. This system includes code metadata and help files for all functional routines. This section details the inner-workings of the system and introduces new developer tools and RSPARROW code-search features that facilitate future collaborative development of the RSPARROW system and SPARROW modeling techniques. The code-search tools are structured in a generic manner that also supports their application to R functions outside of the RSPARROW library.

7.1 Library dependences

The RSPARROW library dependencies include the following standard R packages: car (>= 3.0-8), data.table (>= 1.12.8), data.tree (>= 0.7.11), devtools (>= 2.3.0), dplyr (>= 1.0.0), evaluate (>= 0.14), formatR (>= 1.7), gear (>= 0.3.4), ggplot2 (>= 3.3.2), gplots (>= 3.0.3), highr (>= 0.8), htmltools (>= 0.5.0), htmlwidgets (>= 1.5.1), inline (>= 0.3.15), knitr (>= 1.29), leaflet (>= 2.0.3), leaflet.extras (>= 1.0.0), magrittr (>= 1.5), maptools (>= 1.0-1), mapview (>= 2.7.8), markdown (>= 1.1), nlmrt (>= 2016.3.2), numDeriv (>= 2016.8-1.1), OpenMx (>= 2.17.4), plotly (>= 4.9.2.1), plyr (>= 1.8.6), rgdal (>= 1.5-10), rhandsontable (>= 0.3.7), rmarkdown (>= 2.3), roxygen2 (>= 7.1.0), rstan (>= 2.19.3), rstudioapi (>= 0.11), sf (>= 0.9-4), shiny (>= 1.4.0.2), shinycssloaders (>= 0.3), shinyWidgets (>= 0.5.3), sp (>= 1.4-2), spdep (>= 1.1-3), stringr (>= 1.4.0), svDialogs (>= 1.0.0), svGUI (>= 1.0.0), tools (>= 4.0.2)

Execution of the RSPARROW control script runs the R statement shown below (`devtools::install_deps`), which automatically installs (on the first run of the script) the most recent versions of the required R functions in the user's default R library (as specified in RStudio). The installed R library packages are then loaded into the RStudio session by the R statement `devtools::load_all`.

```
# Install required packages (Section 12 of the control script)
#   this is a one time process unless a new version of R is installed
#   or more recent packages are found on Cran
#   packages previously installed by user will be skipped
if(!"devtools" %in% installed.packages()){install.packages("devtools")}
suppressWarnings(devtools::install_deps(path_master, upgrade = "ask", type="binary"))

# Load RSPARROW functions (These 2 lines should ALWAYS be run together)
suppressWarnings(remove(list="runRsparrow"))
devtools::load_all(path_master,recompile = FALSE)
```

During subsequent executions of the control script, users are notified of the availability of more recent versions of the required R packages, based on comparisons of the installed versions with those available on the Cran server. If a newer version is available, then the user is asked whether the installed R package should be updated (as illustrated in Chapter sub-section 1.3.5 for the *plyr* R package). Users are not required to update the R packages to versions that are more recent than those listed for the RSPARROW compatibility versions in the DESCRIPTION file; RSPARROW compatibility with updated R package versions is not guaranteed.

Note that RStudio will print the following message in a yellow banner when the RSPARROW control script is opened in RStudio for the first time with a new R version: *"Packages devtools and rstudioapi required but are not installed."* Users should ignore this message and instead execute (*"Source"*) the control script, which will install these and other required R packages (**User caution:** Package installation using the yellow banner prompt may exclude certain required packages such as *"backports"*).

Users are also directed to select *"No"* as the response to the following question, if it appears in a pop-up box in RStudio during the installation of R library dependencies: *"Do you want to install from sources the packages which need compilation?"*. A *"No"* response installs R packages from binary versions, which may not include the most recent version of a package. The selection of *"Yes"* requires that users have previously set up *Rtools* (an R developer package) to compile R packages from their source, otherwise the installation of

R packages from their source will fail.

7.2 Library function organization and order of execution

7.2.1 Function and routine types

There are 163 functions and routines in the RSPARROW system, classified according to 11 functional types. These type categories allow developers to search the functions (and their dependencies) by type when using the developer tools in Section 7.2.2. Some functions/routines are classified according to more than one category.

The function types are:

manageDirVars - Directory and path variable setup functions that are executed prior to any part of the SPARROW model execution (16 total functional categories).

sparrowSetup - Used to create, import, check, and format control files for SPARROW execution (38).

sparrowExec - Core functional routines that execute SPARROW models (42).

shiny - Used by the R Shiny interactive decision support tool (27).

batch - Used for batch execution of R scripts using the *Rscript.exe* executable (6).

errorTrap - Tests control settings and files for critical errors and stops execution if a critical error is detected (21).

utility - Used throughout the RSPARROW system to perform minor computational or administrative tasks; these include variable input arguments (23).

lists - Creates control setting lists that organize the control settings for import into multiple functions and routines (8).

rmd - R Markdown functions for creating HTML output of plots and maps (14).

fortran - Fortran subroutines that perform many of the core computational tasks in SPARROW modeling. These tasks typically entail repeated calculations (e.g., accumulation of loads in the reach network) that are more efficiently performed in Fortran. The subroutines are stored as DLL files in the RSPARROW_master/src directory (6).

external - Functions that can be used outside of the RSPARROW system and/or within the *userModifyData.R* control script (6).

A complete table of all functions and routines classified by type is shown in Figures 38, 39, 40, and 41. The table is also stored in the *RSPARROW_master/inst/tables/funcTypes.csv* file.

7.2.2 Developer tools

The R functions in the RSPARROW master directory are executed in a specific order when a SPARROW model is executed. Because making an update to such a system can be intimidating and result in unforeseen errors, we have provided tools (R functions) for developers to make the system links among the RSPARROW functions transparent.

Developers should note that these tools are coded in a generic manner and are not specific to execution only on the RSPARROW system. The tools can be readily executed on any directory of interconnected Rscripts (e.g., R library) and will function in exactly the same way as described in the sections below.

7.2.2.1 findCodeStr() function

The `findCodeStr()` function allows developers to quickly and easily search the RSPARROW library routines (functions) for common character strings to support the following tasks:

- Find all instances where a given routine is executed.

routine	manageDirVars	sparrowSetup	sparrowExec	shiny	batch	errorTrap	utility	lists	rmd	fortran	external
accumulateIncrArea.R		1									
addMarkerText.R							1				
addVars.R	1										
allowRemoveRow.R				1			1				
applyUserModify.R		1									
areColors.R						1	1				
assignIncremSiteIDs.R		1									
batchGeoLines.R	1				1						
batchlineShape.R	1				1						
batchMaps.R					1						
batchpolyShape.R	1				1						
batchRun.R					1						
calcClassLandusePercent.R		1									
calcDemtareaClass.R		1									
calcHeadflag.R		1									
calcIncremLandUse.R							1				
calcTermflag.R		1									
checkAnyMissingSubdataVars.R						1					
checkBinaryMaps.R						1	1				
checkClassificationVars.R		1				1					
checkData1NavigationVars.R		1				1					
checkDrainageareaErrors.R		1				1					
checkDrainageareaErrors.Rmd		1				1			1		
checkDrainageareaErrorsChild.Rmd									1		
checkDrainageareaMapPrep.R		1				1					
checkDupVarnames.R		1				1					
checkingMissingVars.R		1				1					
checkMissingData1Vars.R		1				1					
checkMissingSubdataVars.R		1									
compileALL.R				1			1				
compileInput.R				1			1				
controlFileTasksModel.R			1								
convertHotTables.R				1			1				
copyPriorModelFiles.R		1									
correlationMatrix.R			1								
createDataMatrix.R		1									
createDirs.R	1										
createInitialDataDictionary.R		1									
createInitialParameterControls.R		1									
createInteractiveChoices.R				1			1				
createMasterDataDictionary.R		1									
createRTables.R				1			1				
createSubdataSorted.R		1									
createVerifyReachAttr.R		1									

Figure 38: **RSPARROW library functions and routine types, organized by eleven functional categories. Users can access this table in the RSPARROW funcTypes.csv file.**

routine	manageDirVars	sparrowSetup	sparrowExec	shiny	batch	errorTrap	utility	lists	rmd	fortran	external
dataInputPrep.R		1									
deleteFiles.R	1										
deliv_fraction.for										1	
diagnosticClassLandChild.Rmd									1		
diagnosticClassvarChild.Rmd									1		
diagnosticContiguousChild.Rmd									1		
diagnosticCorrChild.Rmd									1		
diagnosticDiagMapChild.Rmd									1		
diagnosticMapAttrChild.Rmd									1		
diagnosticMaps.R			1								
diagnosticPlotsNLLS.R			1								
diagnosticPlotsNLLS.Rmd			1						1		
diagnosticPlotsValidate.R			1								
diagnosticSensitivity.R			1								
diagnosticSensitivity.Rmd			1						1		
diagnosticSensParamChild.Rmd									1		
diagnosticSpatialAutoCorr.R			1								
diagnosticSpatialAutoCorr.Rmd			1						1		
dropFunc.R				1			1				
eigensort.R			1								
errorOccurred.R						1	1				
estimate.R			1								
estimateBootstraps.R			1								
estimateFeval.R			1								
estimateFevalNoadj.R			1								
estimateNLLSmetrics.R			1								
estimateNLLStable.R			1								
estimateOptimize.R			1								
estimateWeightedErrors.R			1								1
executeRSPARROW.R			1								
executionTree.R											1
findCodeStr.R											1
findControlFiles.R	1										
findMinMaxLatLon.R							1				
findScriptName.R	1										
fixDupLatLons.R		1				1					
generateInputLists.R								1			
getCharSett.R								1			
getNumSett.R								1			
getOptionSett.R								1			
getShortSett.R								1			
getSpecialSett.R						1		1			
getVarList.R								1			
getYesNoSett.R								1			
goShinyPlot.R				1							

Figure 39: **RSPARROW library functions and routine types, organized by eleven functional categories (continued). Users can access this table in the RSPARROW funcTypes.csv file.**

routine	manageDirVars	sparrowSetup	sparrowExec	shiny	batch	errorTrap	utility	lists	rmd	fortran	external
handsOnMod.R				1			1				
handsOnUI.R				1			1				
hydseq.R		1									
hydseqTerm.R		1									
importCSVcontrol.R							1				
interactiveBatchRun.R				1	1						
isScriptSaved.R	1										
makePaths.R	1										
mapBreaks.R							1				
mapSiteAttributes.R			1								
modelCompare.R			1								
mptnoder.for										1	
named.list.R							1				1
openDesign.R	1										
openParameters.R	1										
openVarnames.R	1										
outputSettings.R		1									
plotlyLayout.R							1				
predict.R											
predictBoot.R			1								
predictBootsOutCSV.R			1								
predictBootstraps.R			1								
predictMaps.R			1								
predictMaps.Rmd			1						1		
predictOutCSV.R			1								
predictScenarios.R			1								
predictScenariosOutCSV.R			1								
predictScenariosPrep.R			1								
predictSensitivity.R			1								
predictSummaryOutCSV.R			1								
ptnoder.for										1	
read_dataDictionary.R		1									
readData.R		1									
readDesignMatrix.R		1									
readParameters.R		1									
removeObjects.R	1										
replaceData1Names.R		1									
replaceNAs.R							1				1
runBatchShiny.R				1							1
runRsparrow.R			1								
selectAll.R				1			1				
selectCalibrationSites.R			1								
selectDesignMatrix.R			1								
selectParmValues.R			1								
selectValidationSites.R			1								

Figure 40: **RSPARROW library functions and routine types, organized by eleven functional categories (continued). Users can access this table in the RSPARROW funcTypes.csv file.**

routine	manageDirVars	sparrowSetup	sparrowExec	shiny	batch	errorTrap	utility	lists	rmd	fortran	external
setMapDefaults.R		1				1					
setNLLSWeights.R			1								
setupMaps.R		1									
shapeFunc.R				1							
shinyErrorTrap.R				1		1					
shinyMap2.R				1							
shinySavePlot.Rmd				1					1		
shinyScenarios.R				1							
shinyScenariosMod.R				1							
shinySiteAttr.R				1							
sites_incr.for										1	
sourceRedFunc.R				1							
startEndmodifySubdata.R		1									
startModelRun.R			1								
streamCatch.R				1							
sumIncremAttributes.R		1									
syncVarNames.R	1										
testCosmetic.R				1		1					
testRedTbl.R				1		1					
testSettings.R	1					1					
tnoder.for										1	
unPackList.R							1				
updateVariable.R				1			1				
validateFevalNoadj.R			1								
validateMetrics.R			1								
validCosmetic.R				1		1					
validSetting.R				1		1					
verifyDemtarea.R		1									

Figure 41: **RSPARROW library functions and routine types, organized by eleven functional categories (continued). Users can access this table in the RSPARROW funcTypes.csv file.**

- Determine all arguments for a given routine.
- Find all routines that use a given control setting or functional argument.
- Determine all locations within the RSPARROW system where a given string occurs.

The `findCodeStr()` function will return a data.frame of files and line numbers where the string is found (or arguments in the case of searching for routine arguments), making it a powerful tool for an R developer.

7.2.2.1.1 findCodeStr() arguments

Functional arguments for the `findCodeStr()` function are:

`path_master` is the path to the directory on which the code search is executed. Note that `path_master` is reset internally during a model execution to *RSPARROW_master/R*. Therefore, to execute a search of the entire *RSPARROW_master* directory, which includes the batch R scripts, users should set this argument equal to the `path_main`, which is a global variable that is stored after execution of the RSPARROW control script (*sparrow_control.R*).

`str` indicates the character string to find.

`strType` is the type of string search that is executed. The options include:

- "routine" to find all instances of a routine (`str`) being executed;
- "args" to find all function arguments for the routine (`str`);
- "setting" to find all functions that use the setting (`str`) as an argument;
- "all" to find all instances of the (`str`) to be located.

Note that the "routine" and "args" options are only valid on functional routines–i.e., routines that are defined as an R function. Batch R scripts and Fortran routines are not functional routines and cannot be searched with "routine" and "args" options. Also, the functional routines *estimateFeval.R* and *estimateFevalNoadj.R* are not executed in the traditional way and therefore should be searched with a `strType` equal to "all". Note that all string searches are case dependent and follow standard `regexpr` search syntax.

7.2.2.1.2 findCodeStr() example

Consider the case in which an R developer wants to add an argument to the *mapBreaks.R* function. Without the `findCodeStr()` function, the developer would have to manually search all 135 R scripts and 7 batch Rscripts for every instance where the *mapBreaks* function was executed. This is not only time consuming, but prone to error as it is possible that one or more executions might be missed, resulting in execution errors. By contrast, the `findCodeStr()` function allows users to simply execute the R statement as shown below in the RStudio Console. Now it is a simple operation to add the new argument to all executions of the mapBreaks.R routine allowing the update to propagate through the entire system so that it can be executed without error. Note that because all string searches are case dependent, a search for `mapbreaks.R` would yield no results, while a search for `mapBreaks.R` will output results.

```
path_master <- "~/RSPARROW_master"
path_main <- path_master
devtools::load_all(path_main,recompile = FALSE)
```

```
## Loading RSPARROW
```

```
findCodeStr(path_master, "mapBreaks.R", "routine")
```

```
##                  file line
## 1 mapSiteAttributes.R  101
## 2       predictMaps.R  447
## 3       predictMaps.R  475
## 4       predictMaps.R  476
```

```
## 5       predictMaps.R  497
## 6       predictMaps.R  498
```

7.2.2.2 executionTree() function

The `executionTree()` function was developed to provide R developers with a complete picture of all functional and routine executions and interdependences among functions in order of occurrence after the execution of the RSPARROW control script. This extends the capabilities of the `findCodeStr()` function, which is very useful for isolated updates but would require many repeated executions of the function to obtain a full picture of how the RSPARROW system operates.

The `executionTree()` function has the capability to trace functional executions forward from a given function that is specified by the user (i.e., see `startRoutine` argument below).

The function outputs either a "data.tree" format or a "data.table" format with line numbers, associated with the executions of "all" or a "subset" of the R scripts, batch R scripts, and Fortran routine executions,

7.2.2.2.1 executionTree() arguments

The `executionTree()` functional arguments are:

`path_main` - the path to the RSPARROW_master directory.

`startRoutine` - the starting function name for tracing subsequent executions. The default setting for `startRoutine` is *runRsparrow.R*, which will trace the entire RSPARROW system library.

`includeTypes` - the function/routine types that are included in the search. Ten `includeTypes` are available, as listed in Section 7.2.1 and the *RSPARROW_master/inst/tables/funcTypes.csv* file. This argument allows users to trim the execution trace so that only functions of interest are displayed. The setting `includeTypes="all"` displays all ten function/routine types, whereas the default setting includes eight types and excludes the "utility" and "external" types.

`includeList` - a character string of functions/routines to include, even if they are not in `includeTypes`. The default setting is `NA`, which will only include functions/routines from the `includeTypes` setting.

`excludeList` - a character string of functions/routines to exclude from the `includeTypes` setting. The default is `c("errorOccurred.R","named.list.R")`. This allows the user to trim down the execution tree further so that only the specific functions of interest are displayed.

`allOccurances` - a TRUE/FALSE setting that indicates whether all executions are to be found or only the first occurrence in each function/routine. This is only an option if the `outputType` is "data.table". The default setting is FALSE.

`outputType` - indicates whether a data.tree of executions or a data.table with line numbers of executions will be output. The default setting is "data.tree".

`pruneTree` indicates the maximum node level at which the data.tree will be pruned. The default setting is NA, which includes all nodes.

`treeLimit` indicates the maximum number of lines of the data.tree to print. The default setting is NA, which will print the entire tree.

7.2.2.2.2 executionTree() examples

Execution of the `executionTree()` function with the default arguments (as shown below) provides a current and up-to-date version of the program map for the entire RSPARROW system because the map is generated dynamically via the `executionTree()` function. Whenever new routines are added, the tree will grow.

Note that all functions/routines are listed in the order of first execution within the parent function/routine. There may be multiple executions of any given function/routine within the parent script. For a full list of all executions, "data.table" should be specified as the `outputType`.

```
path_master <- "~/RSPARROW_master"
path_main <- path_master
devtools::load_all(path_main,recompile = FALSE)
```

```
## Loading RSPARROW
```

```
executionTree(path_master)
```

```
##                                                    levelName
## 1   runRsparrow.R
## 2    ¦--getCharSett.R
## 3    ¦--getNumSett.R
## 4    ¦--getOptionSett.R
## 5    ¦--getShortSett.R
## 6    ¦--getYesNoSett.R
## 7    °--executeRSPARROW.R
## 8        ¦--copyPriorModelFiles.R
## 9        ¦--findControlFiles.R
## 10       ¦--openParameters.R
## 11       ¦--openDesign.R
## 12       ¦--openVarnames.R
## 13       ¦--removeObjects.R
## 14       ¦--isScriptSaved.R
## 15       ¦--setMapDefaults.R
## 16       ¦--testSettings.R
## 17       ¦   ¦--getYesNoSett.R
## 18       ¦   ¦--getCharSett.R
## 19       ¦   ¦--getOptionSett.R
## 20       ¦   ¦--getShortSett.R
## 21       ¦   ¦--getNumSett.R
## 22       ¦   °--getSpecialSett.R
## 23       ¦       °--areColors.R
## 24       ¦--makePaths.R
## 25       ¦--getCharSett.R
## 26       ¦--getNumSett.R
## 27       ¦--getOptionSett.R
## 28       ¦--getShortSett.R
## 29       ¦--getYesNoSett.R
## 30       ¦--generateInputLists.R
## 31       ¦--createInitialDataDictionary.R
## 32       ¦   ¦--readData.R
## 33       ¦   °--getVarList.R
## 34       ¦--createInitialParameterControls.R
## 35       ¦--addVars.R
## 36       ¦   °--getVarList.R
## 37       ¦--setupMaps.R
## 38       ¦   ¦--outputSettings.R
## 39       ¦   ¦   ¦--getCharSett.R
## 40       ¦   ¦   ¦--getNumSett.R
## 41       ¦   ¦   ¦--getShortSett.R
## 42       ¦   ¦   ¦--getYesNoSett.R
## 43       ¦   ¦   °--getOptionSett.R
## 44       ¦   ¦--batchGeoLines.R
## 45       ¦   ¦--batchlineShape.R
```

```
## 46       ¦   °--batchpolyShape.R
## 47       ¦--createDirs.R
## 48       ¦   °--syncVarNames.R
## 49       ¦       °--getVarList.R
## 50       ¦--deleteFiles.R
## 51       ¦--dataInputPrep.R
## 52       ¦   ¦--readData.R
## 53       ¦   ¦--read_dataDictionary.R
## 54       ¦   ¦   °--getVarList.R
## 55       ¦   ¦--checkDupVarnames.R
## 56       ¦   ¦--replaceData1Names.R
## 57       ¦   ¦   °--getVarList.R
## 58       ¦   ¦--checkData1NavigationVars.R
## 59       ¦   ¦--createVerifyReachAttr.R
## 60       ¦   ¦   ¦--hydseq.R
## 61       ¦   ¦   ¦   ¦--getVarList.R
## 62       ¦   ¦   ¦   °--accumulateIncrArea.R
## 63       ¦   ¦   ¦--calcTermflag.R
## 64       ¦   ¦   ¦   °--getVarList.R
## 65       ¦   ¦   ¦--calcHeadflag.R
## 66       ¦   ¦   ¦   °--getVarList.R
## 67       ¦   ¦   ¦--accumulateIncrArea.R
## 68       ¦   ¦   °--verifyDemtarea.R
## 69       ¦   ¦       °--checkDrainageareaErrors.R
## 70       ¦   ¦           ¦--checkBinaryMaps.R
## 71       ¦   ¦           °--checkDrainageareaErrors.Rmd
## 72       ¦   ¦               °--checkDrainageareaErrorsChild.Rmd
## 73       ¦   ¦                   °--checkDrainageareaMapPrep.R
## 74       ¦   °--checkMissingData1Vars.R
## 75       ¦       ¦--checkingMissingVars.R
## 76       ¦       ¦   °--getVarList.R
## 77       ¦       °--getVarList.R
## 78       ¦--startModelRun.R
## 79       ¦   ¦--createMasterDataDictionary.R
## 80       ¦   ¦   °--getVarList.R
## 81       ¦   ¦--readParameters.R
## 82       ¦   ¦   °--getVarList.R
## 83       ¦   ¦--selectParmValues.R
## 84       ¦   ¦--readDesignMatrix.R
## 85       ¦   ¦   °--getVarList.R
## 86       ¦   ¦--selectDesignMatrix.R
## 87       ¦   ¦--createSubdataSorted.R
## 88       ¦   ¦--applyUserModify.R
## 89       ¦   ¦--startEndmodifySubdata.R
## 90       ¦   ¦--checkClassificationVars.R
## 91       ¦   ¦--checkMissingSubdataVars.R
## 92       ¦   ¦   ¦--checkingMissingVars.R
## 93       ¦   ¦   ¦   °--getVarList.R
## 94       ¦   ¦   °--getVarList.R
## 95       ¦   ¦--selectCalibrationSites.R
## 96       ¦   ¦   °--assignIncremSiteIDs.R
## 97       ¦   ¦       °--sites_incr.for
## 98       ¦   ¦--selectValidationSites.R
## 99       ¦   ¦   °--assignIncremSiteIDs.R
```

```
## 100      ¦   ¦       °--sites_incr.for
## 101      ¦   ¦--checkAnyMissingSubdataVars.R
## 102      ¦   ¦   ¦--checkingMissingVars.R
## 103      ¦   ¦   ¦   °--getVarList.R
## 104      ¦   ¦   °--getVarList.R
## 105      ¦   ¦--createDataMatrix.R
## 106      ¦   ¦   °--getVarList.R
## 107      ¦   ¦--calcDemtareaClass.R
## 108      ¦   ¦--correlationMatrix.R
## 109      ¦   ¦   °--sumIncremAttributes.R
## 110      ¦   ¦--setNLLSWeights.R
## 111      ¦   ¦   °--assignIncremSiteIDs.R
## 112      ¦   ¦       °--sites_incr.for
## 113      ¦   ¦--controlFileTasksModel.R
## 114      ¦   ¦   ¦--estimate.R
## 115      ¦   ¦   ¦   ¦--estimateOptimize.R
## 116      ¦   ¦   ¦   ¦   °--estimateFeval.R
## 117      ¦   ¦   ¦   ¦       °--tnoder.for
## 118      ¦   ¦   ¦   ¦--estimateNLLSmetrics.R
## 119      ¦   ¦   ¦   ¦   ¦--getVarList.R
## 120      ¦   ¦   ¦   ¦   ¦--eigensort.R
## 121      ¦   ¦   ¦   ¦   ¦--estimateFeval.R
## 122      ¦   ¦   ¦   ¦   ¦   °--tnoder.for
## 123      ¦   ¦   ¦   ¦   °--estimateFevalNoadj.R
## 124      ¦   ¦   ¦   ¦       °--tnoder.for
## 125      ¦   ¦   ¦   ¦--validateMetrics.R
## 126      ¦   ¦   ¦   ¦   ¦--getVarList.R
## 127      ¦   ¦   ¦   ¦   °--validateFevalNoadj.R
## 128      ¦   ¦   ¦   ¦       °--tnoder.for
## 129      ¦   ¦   ¦   ¦--estimateNLLStable.R
## 130      ¦   ¦   ¦   ¦--diagnosticPlotsNLLS.R
## 131      ¦   ¦   ¦   ¦   °--diagnosticPlotsNLLS.Rmd
## 132      ¦   ¦   ¦   ¦       ¦--checkBinaryMaps.R
## 133      ¦   ¦   ¦   ¦       ¦--diagnosticMapAttrChild.Rmd
## 134      ¦   ¦   ¦   ¦       ¦   °--mapSiteAttributes.R
## 135      ¦   ¦   ¦   ¦       ¦       °--checkBinaryMaps.R
## 136      ¦   ¦   ¦   ¦       ¦--diagnosticCorrChild.Rmd
## 137      ¦   ¦   ¦   ¦       ¦--diagnosticClassvarChild.Rmd
## 138      ¦   ¦   ¦   ¦       ¦--diagnosticClassLandChild.Rmd
## 139      ¦   ¦   ¦   ¦       ¦--diagnosticContiguousChild.Rmd
## 140      ¦   ¦   ¦   ¦       °--diagnosticDiagMapChild.Rmd
## 141      ¦   ¦   ¦   ¦           °--diagnosticMaps.R
## 142      ¦   ¦   ¦   ¦--diagnosticSensitivity.R
## 143      ¦   ¦   ¦   ¦   °--diagnosticSensitivity.Rmd
## 144      ¦   ¦   ¦   ¦       ¦--predictSensitivity.R
## 145      ¦   ¦   ¦   ¦       ¦   ¦--getVarList.R
## 146      ¦   ¦   ¦   ¦       ¦   °--ptnoder.for
## 147      ¦   ¦   ¦   ¦       °--diagnosticSensParamChild.Rmd
## 148      ¦   ¦   ¦   ¦--diagnosticPlotsValidate.R
## 149      ¦   ¦   ¦   ¦   °--diagnosticPlotsNLLS.Rmd
## 150      ¦   ¦   ¦   ¦       ¦--checkBinaryMaps.R
## 151      ¦   ¦   ¦   ¦       ¦--diagnosticMapAttrChild.Rmd
## 152      ¦   ¦   ¦   ¦       ¦   °--mapSiteAttributes.R
## 153      ¦   ¦   ¦   ¦       ¦       °--checkBinaryMaps.R
```

```
## 154      ¦   ¦   ¦   ¦       ¦--diagnosticCorrChild.Rmd
## 155      ¦   ¦   ¦   ¦       ¦--diagnosticClassvarChild.Rmd
## 156      ¦   ¦   ¦   ¦       ¦--diagnosticClassLandChild.Rmd
## 157      ¦   ¦   ¦   ¦       ¦--diagnosticContiguousChild.Rmd
## 158      ¦   ¦   ¦   ¦       °--diagnosticDiagMapChild.Rmd
## 159      ¦   ¦   ¦   ¦           °--diagnosticMaps.R
## 160      ¦   ¦   ¦   ¦--estimateFevalNoadj.R
## 161      ¦   ¦   ¦   ¦   °--tnoder.for
## 162      ¦   ¦   ¦   ¦--estimateFeval.R
## 163      ¦   ¦   ¦   ¦   °--tnoder.for
## 164      ¦   ¦   ¦   ¦--predict.R
## 165      ¦   ¦   ¦   ¦   ¦--getVarList.R
## 166      ¦   ¦   ¦   ¦   ¦--ptnoder.for
## 167      ¦   ¦   ¦   ¦   ¦--mptnoder.for
## 168      ¦   ¦   ¦   ¦   °--deliv_fraction.for
## 169      ¦   ¦   ¦   °--predictSummaryOutCSV.R
## 170      ¦   ¦   ¦       °--calcClassLandusePercent.R
## 171      ¦   ¦   ¦           °--accumulateIncrArea.R
## 172      ¦   ¦   ¦--diagnosticSpatialAutoCorr.R
## 173      ¦   ¦   ¦   °--diagnosticSpatialAutoCorr.Rmd
## 174      ¦   ¦   ¦       °--fixDupLatLons.R
## 175      ¦   ¦   ¦--estimateBootstraps.R
## 176      ¦   ¦   ¦   ¦--getVarList.R
## 177      ¦   ¦   ¦   °--estimateFeval.R
## 178      ¦   ¦   ¦       °--tnoder.for
## 179      ¦   ¦   ¦--predict.R
## 180      ¦   ¦   ¦   ¦--getVarList.R
## 181      ¦   ¦   ¦   ¦--ptnoder.for
## 182      ¦   ¦   ¦   ¦--mptnoder.for
## 183      ¦   ¦   ¦   °--deliv_fraction.for
## 184      ¦   ¦   ¦--predictOutCSV.R
## 185      ¦   ¦   ¦   °--getVarList.R
## 186      ¦   ¦   ¦--predictBootstraps.R
## 187      ¦   ¦   ¦   ¦--getVarList.R
## 188      ¦   ¦   ¦   °--predictBoot.R
## 189      ¦   ¦   ¦       ¦--getVarList.R
## 190      ¦   ¦   ¦       ¦--ptnoder.for
## 191      ¦   ¦   ¦       ¦--mptnoder.for
## 192      ¦   ¦   ¦       °--deliv_fraction.for
## 193      ¦   ¦   ¦--predictBootsOutCSV.R
## 194      ¦   ¦   ¦   °--getVarList.R
## 195      ¦   ¦   ¦--outputSettings.R
## 196      ¦   ¦   ¦   ¦--getCharSett.R
## 197      ¦   ¦   ¦   ¦--getNumSett.R
## 198      ¦   ¦   ¦   ¦--getShortSett.R
## 199      ¦   ¦   ¦   ¦--getYesNoSett.R
## 200      ¦   ¦   ¦   °--getOptionSett.R
## 201      ¦   ¦   ¦--batchMaps.R
## 202      ¦   ¦   ¦   °--predictMaps.R
## 203      ¦   ¦   ¦       ¦--checkBinaryMaps.R
## 204      ¦   ¦   ¦       °--predictMaps.Rmd
## 205      ¦   ¦   °--predictScenarios.R
## 206      ¦   ¦       ¦--predictScenariosPrep.R
## 207      ¦   ¦       ¦   °--hydseqTerm.R
```

```
## 208      ¦   ¦       ¦       °--getVarList.R
## 209      ¦   ¦       ¦--ptnoder.for
## 210      ¦   ¦       ¦--mptnoder.for
## 211      ¦   ¦       ¦--deliv_fraction.for
## 212      ¦   ¦       ¦--predictScenariosOutCSV.R
## 213      ¦   ¦       ¦   °--getVarList.R
## 214      ¦   ¦       ¦--outputSettings.R
## 215      ¦   ¦       ¦   ¦--getCharSett.R
## 216      ¦   ¦       ¦   ¦--getNumSett.R
## 217      ¦   ¦       ¦   ¦--getShortSett.R
## 218      ¦   ¦       ¦   ¦--getYesNoSett.R
## 219      ¦   ¦       ¦   °--getOptionSett.R
## 220      ¦   ¦       ¦--batchMaps.R
## 221      ¦   ¦       ¦   °--predictMaps.R
## 222      ¦   ¦       ¦       ¦--checkBinaryMaps.R
## 223      ¦   ¦       ¦       °--predictMaps.Rmd
## 224      ¦   ¦       °--predictMaps.R
## 225      ¦   ¦           ¦--checkBinaryMaps.R
## 226      ¦   ¦           °--predictMaps.Rmd
## 227      ¦   ¦--outputSettings.R
## 228      ¦   ¦   ¦--getCharSett.R
## 229      ¦   ¦   ¦--getNumSett.R
## 230      ¦   ¦   ¦--getShortSett.R
## 231      ¦   ¦   ¦--getYesNoSett.R
## 232      ¦   ¦   °--getOptionSett.R
## 233      ¦   ¦--modelCompare.R
## 234      ¦   °--runBatchShiny.R
## 235      ¦       °--shinyBatch.R
## 236      ¦           °--shinyMap2.R
## 237      ¦               ¦--createInteractiveChoices.R
## 238      ¦               ¦--createRTables.R
## 239      ¦               ¦--streamCatch.R
## 240      ¦               ¦   ¦--dropFunc.R
## 241      ¦               ¦   °--handsOnUI.R
## 242      ¦               ¦--shinySiteAttr.R
## 243      ¦               ¦   ¦--dropFunc.R
## 244      ¦               ¦   °--handsOnUI.R
## 245      ¦               ¦--shinyScenarios.R
## 246      ¦               ¦   ¦--handsOnUI.R
## 247      ¦               ¦   °--dropFunc.R
## 248      ¦               ¦--shapeFunc.R
## 249      ¦               ¦--selectAll.R
## 250      ¦               ¦--updateVariable.R
## 251      ¦               ¦--shinyScenariosMod.R
## 252      ¦               ¦   ¦--handsOnMod.R
## 253      ¦               ¦   ¦   °--allowRemoveRow.R
## 254      ¦               ¦   ¦--testCosmetic.R
## 255      ¦               ¦   ¦   ¦--compileInput.R
## 256      ¦               ¦   ¦   ¦--convertHotTables.R
## 257      ¦               ¦   ¦   ¦--getNumSett.R
## 258      ¦               ¦   ¦   °--getSpecialSett.R
## 259      ¦               ¦   ¦       °--areColors.R
## 260      ¦               ¦   °--validCosmetic.R
## 261      ¦               ¦--testCosmetic.R
```

```
## 262      ¦                ¦   ¦--compileInput.R
## 263      ¦                ¦   ¦--convertHotTables.R
## 264      ¦                ¦   ¦--getNumSett.R
## 265      ¦                ¦   °--getSpecialSett.R
## 266      ¦                ¦       °--areColors.R
## 267      ¦                ¦--validCosmetic.R
## 268      ¦                ¦--testRedTbl.R
## 269      ¦                ¦   °--convertHotTables.R
## 270      ¦                °--goShinyPlot.R
## 271      ¦                    ¦--compileALL.R
## 272      ¦                    ¦   °--compileInput.R
## 273      ¦                    ¦--convertHotTables.R
## 274      ¦                    ¦--shinyErrorTrap.R
## 275      ¦                    ¦--predictMaps.R
## 276      ¦                    ¦   ¦--checkBinaryMaps.R
## 277      ¦                    ¦   °--predictMaps.Rmd
## 278      ¦                    ¦--mapSiteAttributes.R
## 279      ¦                    ¦   °--checkBinaryMaps.R
## 280      ¦                    ¦--sourceRedFunc.R
## 281      ¦                    °--predictScenarios.R
## 282      ¦                        ¦--predictScenariosPrep.R
## 283      ¦                        ¦   °--hydseqTerm.R
## 284      ¦                        ¦       °--getVarList.R
## 285      ¦                        ¦--ptnoder.for
## 286      ¦                        ¦--mptnoder.for
## 287      ¦                        ¦--deliv_fraction.for
## 288      ¦                        ¦--predictScenariosOutCSV.R
## 289      ¦                        ¦   °--getVarList.R
## 290      ¦                        ¦--outputSettings.R
## 291      ¦                        ¦   ¦--getCharSett.R
## 292      ¦                        ¦   ¦--getNumSett.R
## 293      ¦                        ¦   ¦--getShortSett.R
## 294      ¦                        ¦   ¦--getYesNoSett.R
## 295      ¦                        ¦   °--getOptionSett.R
## 296      ¦                        ¦--batchMaps.R
## 297      ¦                        ¦   °--predictMaps.R
## 298      ¦                        ¦       ¦--checkBinaryMaps.R
## 299      ¦                        ¦       °--predictMaps.Rmd
## 300      ¦                        °--predictMaps.R
## 301      ¦                            ¦--checkBinaryMaps.R
## 302      ¦                            °--predictMaps.Rmd
## 303      °--outputSettings.R
## 304          ¦--getCharSett.R
## 305          ¦--getNumSett.R
## 306          ¦--getShortSett.R
## 307          ¦--getYesNoSett.R
## 308          °--getOptionSett.R
```

The `executionTree()` function does not have to be executed with output generated for the entire RSPARROW library. Entering a different `startRoutine` argument allows the user to trace executions forward from that function location only. As an illustration, if the user is only interested in functional and routine dependencies associated with the *predictScenarios.R* functions, then the following arguments could be used in the function to generate the list.

```
path_master <- "~/RSPARROW_master"
path_main <- path_master
devtools::load_all(path_main,recompile = FALSE)
```

```
## Loading RSPARROW
```

```
executionTree(path_master,"predictScenarios.R",includeTypes = 'all', excludeList = NA)
```

```
##                           levelName
## 1  predictScenarios.R
## 2    ¦--unPackList.R
## 3    ¦--predictScenariosPrep.R
## 4    ¦   ¦--unPackList.R
## 5    ¦   ¦--hydseqTerm.R
## 6    ¦   ¦   ¦--unPackList.R
## 7    ¦   ¦   °--getVarList.R
## 8    ¦   ¦       °--named.list.R
## 9    ¦   °--named.list.R
## 10   ¦--ptnoder.for
## 11   ¦--mptnoder.for
## 12   ¦--deliv_fraction.for
## 13   ¦--named.list.R
## 14   ¦--predictScenariosOutCSV.R
## 15   ¦   ¦--unPackList.R
## 16   ¦   ¦--getVarList.R
## 17   ¦   ¦   °--named.list.R
## 18   ¦   °--named.list.R
## 19   ¦--outputSettings.R
## 20   ¦   ¦--unPackList.R
## 21   ¦   ¦--getCharSett.R
## 22   ¦   ¦--getNumSett.R
## 23   ¦   ¦--getShortSett.R
## 24   ¦   ¦--getYesNoSett.R
## 25   ¦   °--getOptionSett.R
## 26   ¦--batchMaps.R
## 27   ¦   ¦--unPackList.R
## 28   ¦   °--predictMaps.R
## 29   ¦       ¦--unPackList.R
## 30   ¦       ¦--checkBinaryMaps.R
## 31   ¦       ¦--mapBreaks.R
## 32   ¦       ¦   °--named.list.R
## 33   ¦       ¦--addMarkerText.R
## 34   ¦       ¦   °--named.list.R
## 35   ¦       °--predictMaps.Rmd
## 36   ¦           °--addMarkerText.R
## 37   ¦               °--named.list.R
## 38   °--predictMaps.R
## 39       ¦--unPackList.R
## 40       ¦--checkBinaryMaps.R
## 41       ¦--mapBreaks.R
## 42       ¦   °--named.list.R
## 43       ¦--addMarkerText.R
## 44       ¦   °--named.list.R
## 45       °--predictMaps.Rmd
```

```
## 46          °--addMarkerText.R
## 47              °--named.list.R
```

If more detailed information on function executions is needed that includes a display of all executions and line numbers, then the following statement will generate a data.table, as shown for the *estimateNLLSmetrics.R* function.

```
path_master <- "~/RSPARROW_master"
path_main <- path_master
devtools::load_all(path_main,recompile = FALSE)
```

```
## Loading RSPARROW
```

```
executionTable<-executionTree(path_master,startRoutine="estimateNLLSmetrics.R",
                              allOccurances = TRUE,outputType = "data.table")
print(executionTable)
```

```
##                  routine            executes1 line1  executes2 line2
## 1: estimateNLLSmetrics.R         getVarList.R    94       <NA>    NA
## 2: estimateNLLSmetrics.R          eigensort.R   275       <NA>    NA
## 3: estimateNLLSmetrics.R      estimateFeval.R   437 tnoder.for   134
## 4: estimateNLLSmetrics.R estimateFevalNoadj.R   524 tnoder.for   129
## 5: estimateNLLSmetrics.R      estimateFeval.R   524 tnoder.for   134
```

7.3 Function metadata

All functional routines have `rmarkdown` style metadata including function descriptions, parameter (argument) descriptions, and return object descriptions at the top of each routine. Following the function description, additional information is printed on what routines execute a given routine and what routines are executed by the given routine. CSV files of all code metadata by routine (or by parameter) are stored in the RSPARROW_master/inst/tables directory.

An example of the code metadata format is given below for the `findCodeStr()` function, with the corresponding Help file display in RStudio shown in Figure 42.

```
#'@title findCodeStr
#'@description find all instances where a given routine is executed, determine all
#`             arguments for a given routine, find all routines that use a given control
#`             setting or functional argument, or determine all locations within the
#`             system a given string occurs
#'@param path_master character string path to RSPARROW_master directory.  Internally reset
#'       to 'RSPARROW_master/R/' subdirectory
#'@param str character string to find in function
#'@param strType type of string search, 'routine' indicates that all instances of a routine
#'       (str) being executed will be found, 'args' indicates that all function arguments
#`       for the routine (str) are found, 'setting' indicates that all functions that use
#`       the setting (str) as an argument are found, and 'all' indicates that all instances
#`       of the str should be found
```

7.4 The modifyUserData.R script

The *userModifyData.R* control script is executed if `if_userModifyData<-"yes"`. This control script is not a functional routine, but is a series of sourced R statements that allow the user to perform calculations or conditional tasks with all variables available in the *dataDictionary.csv* control file.

The *userModifyData.R* control file is input into the RSPARROW system as text using `readLines()` in the *applyUserModify.R* function. This function adds the necessary text prior to and after the *userModifyData.R* script to transform the script into a functional routine called `modifySubdata()`.

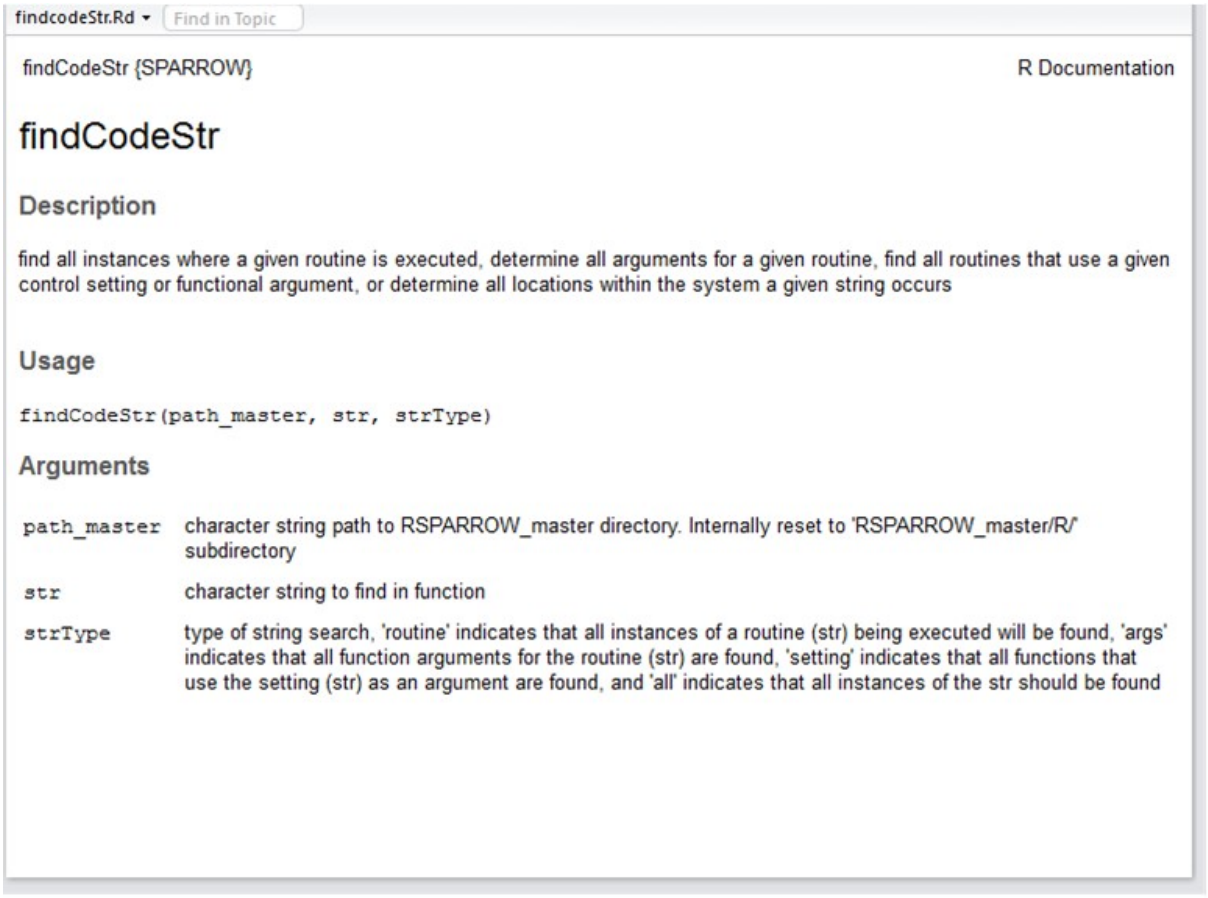


Figure 42: **Example meta-data Help listing in the RStudio window for the RSPARROW findCodeStr function. The listing is displayed for the RStudio Console command "?findCodeStr".**

An excerpt from the *applyUserModify.R* function is shown below. This function loads all the variables from the *dataDictionary.csv* as vectors prior to executing the statements in *userModifyData.R* routine. The function also applies all edits to the variables to the *subdata* object that is returned as output from the `modifySubdata()` function.

```
  #read userModifyData file as text
  userMod <- readLines(pathToUserMod)

#-------------------------------------------------------------------------------------------
  #header text
  top<-"modifySubdata <- function(betavalues,data_names,subdata,class_landuse,lon_limit,
                                  file.output.list) {
unPackList(list=list(datalstreq=data_names$sparrowNames,
                     betavalues=betavalues$sparrowNames,
                    file.output.list = file.output.list),
          list(subdata = subdata,
               subdata = subdata,
              NA))
"
#-------------------------------------------------------------------------------------------
#footer text
  bottom<-  "
#check for missing landuse class
missingLanduseClass<-class_landuse[which(!class_landuse %in% data_names$sparrowNames)]
  if (length(na.omit(missingLanduseClass))!=0){
  for (i in 1:length(missingLanduseClass)){
  cat('\n FATAL ERROR : MISSING class_landuse : ',missingLanduseClass[i],'\n ',sep='')
  cat('\n \n')
  }
  }
  #-----------------------------------------------------
  # substitute 0.0 for NAs for user-selected parameters
  # set NAs for explanatory variables associated with the selected parameters
eval(parse(text=paste('replaceNAs(named.list(',paste(paste('\"',
betavalues$sparrowNames[betavalues$parmMax != 0],'\"',sep=''),collapse=','),'))',sep='')))

  #-----------------------------------------------------
  # Transfer global variables to SUBDATA
  # Refresh variables in 'subdata' (this allows subsequent use of subdata values)
  #  (accounts for any modification to these variables to replace NAs or
  #   following calculations in the data modifications section)
  datalstreq <- data_names$sparrowNames
  for (i in 1:length(datalstreq)) {
  dname <- paste('subdata$',datalstreq[i],' <- ',datalstreq[i],sep='')
  eval(parse(text=dname))
  }

  #-----------------------------------------------------
  # Ensure that variables associated with user-selected parameters are
  #   reassigned to SUBDATA
  for (i in 1:length(betavalues$sparrowNames)) {
  if(betavalues$parmMax[i]>0){
  dname <- paste('subdata$',betavalues$sparrowNames[i],' <-
              ',betavalues$sparrowNames[i],sep='')
```

```
  eval(parse(text=dname))
  }
  }

  return(subdata)
  }"
  #-----------------------------------------------------
  #add header and footers
  userMod<-paste(top,"\n", paste(userMod,collapse="\n"),"\n", bottom)

  #evaluate modifySubdata as text
  eval(parse(text = userMod))

  #create subdata
  subdata<-modifySubdata(betavalues,data_names,subdata,class_landuse,lon_limit,
                         file.output.list)
```

7.5 The R Shiny modular design

The R Shiny app in the RSPARROW system allows users to easily display maps of the SPARROW prediction metrics, site attributes, and source-change management scenario predictions with little to no knowledge of R syntax. The parent app is the *shinyMap2.R* routine, which is divided into modules according to the type of map being generated. The app uses the *rhandsontable* R package to create dynamic user input tables for cosmetic map settings and source-change scenario setup, and includes dynamic error trapping to highlight invalid user entries and prevent app failure.

The `executionTree()` of the shiny app is shown below.

```
path_master <- "~/RSPARROW_master"
path_main <- path_master
devtools::load_all(path_main,recompile = FALSE)
```

```
## Loading RSPARROW
```

```
executionTree(path_main,"shinyMap2.R")
```

```
##                                levelName
## 1  shinyMap2.R
## 2   ¦--createInteractiveChoices.R
## 3   ¦--createRTables.R
## 4   ¦--streamCatch.R
## 5   ¦   ¦--dropFunc.R
## 6   ¦   °--handsOnUI.R
## 7   ¦--shinySiteAttr.R
## 8   ¦   ¦--dropFunc.R
## 9   ¦   °--handsOnUI.R
## 10  ¦--shinyScenarios.R
## 11  ¦   ¦--handsOnUI.R
## 12  ¦   °--dropFunc.R
## 13  ¦--shapeFunc.R
## 14  ¦--selectAll.R
## 15  ¦--updateVariable.R
## 16  ¦--shinyScenariosMod.R
## 17  ¦   ¦--handsOnMod.R
## 18  ¦   ¦   °--allowRemoveRow.R
```

```
## 19  ¦   ¦--testCosmetic.R
## 20  ¦   ¦   ¦--compileInput.R
## 21  ¦   ¦   ¦--convertHotTables.R
## 22  ¦   ¦   ¦--getNumSett.R
## 23  ¦   ¦   °--getSpecialSett.R
## 24  ¦   ¦       °--areColors.R
## 25  ¦   °--validCosmetic.R
## 26  ¦--testCosmetic.R
## 27  ¦   ¦--compileInput.R
## 28  ¦   ¦--convertHotTables.R
## 29  ¦   ¦--getNumSett.R
## 30  ¦   °--getSpecialSett.R
## 31  ¦       °--areColors.R
## 32  ¦--validCosmetic.R
## 33  ¦--testRedTbl.R
## 34  ¦   °--convertHotTables.R
## 35  °--goShinyPlot.R
## 36      ¦--compileALL.R
## 37      ¦   °--compileInput.R
## 38      ¦--convertHotTables.R
## 39      ¦--shinyErrorTrap.R
## 40      ¦--predictMaps.R
## 41      ¦   ¦--checkBinaryMaps.R
## 42      ¦   °--predictMaps.Rmd
## 43      ¦--mapSiteAttributes.R
## 44      ¦   °--checkBinaryMaps.R
## 45      ¦--sourceRedFunc.R
## 46      °--predictScenarios.R
## 47          ¦--predictScenariosPrep.R
## 48          ¦   °--hydseqTerm.R
## 49          ¦       °--getVarList.R
## 50          ¦--ptnoder.for
## 51          ¦--mptnoder.for
## 52          ¦--deliv_fraction.for
## 53          ¦--predictScenariosOutCSV.R
## 54          ¦   °--getVarList.R
## 55          ¦--outputSettings.R
## 56          ¦   ¦--getCharSett.R
## 57          ¦   ¦--getNumSett.R
## 58          ¦   ¦--getShortSett.R
## 59          ¦   ¦--getYesNoSett.R
## 60          ¦   °--getOptionSett.R
## 61          ¦--batchMaps.R
## 62          ¦   °--predictMaps.R
## 63          ¦       ¦--checkBinaryMaps.R
## 64          ¦       °--predictMaps.Rmd
## 65          °--predictMaps.R
## 66              ¦--checkBinaryMaps.R
## 67              °--predictMaps.Rmd
```

7.6 Management of RSPARROW system and model variables

7.6.1 The data dictionary variable respository

The *dataDictionary.csv* file provides an active repository of the system and user-defined variable names (see Chapter sub-section 3.2). Access to the contents of this file during an RSPARROW RStudio session streamlines R developer access to SPARROW variables for use in mapping or other types of functions.

The contents of this file (variable names and attributes) are stored in the *data_names* and *subdata* R binary objects during an RStudio session. The *subdata* object is also output to the */data* sub-directory, following the execution of the control script.

7.6.2 Parameter classification conventions

The RSPARROW model parameters are assigned different classifications according to the conceptual description of contaminant sources and landscape and aquatic processes in the SPARROW model. The process classifications of the model parameters are described in Chapter sub-section 3.3 as SOURCE, DELIVF, STRM, RESV, OTHER. The parameter types are also stored in the *SelParmValues* object.

The model parameter classification method used here is employed in the SAS SPARROW system (Schwarz et al., 2006) and was adopted for use in RSPARROW.

7.7 Network navigation using the *hydseq.R* function

The *hydseq.R* and *hydseqTerm.R* functions were adapted from SAS SPARROW code developed by Gregory Schwarz (USGS, 2018, written communication). The *hydseq.R* function is a direct translation from SAS to R of the SAS SPARROW code with no updates or changes.

The *hydseq.R* function creates the hydrological sequence variable `hydseq`. This variable is required to hydrologically sort reach attributes in the SPARROW model input file (*data1.csv*) to ensure that loads are accurately accumulated in the river network during model execution and prediction. The function uses an efficient reach navigation method that creates an indexed list of all reach flowlines immediately upstream of a given `fnode` and immediately downstream of a given `fnode`. Terminal flowlines are identified as flowlines with a `tnode` that has no matching `fnode` in the network. The routine is initialized using a stack of terminal flowlines. Sequential `hydseq` values are then assigned and the stack is repopulated with all flowlines upstream of the current stack. The routine terminates when all flowlines upstream of the terminal flowlines are assigned a value for `hydseq`.

The *hydseqTerm.R* function is used to identify reaches upstream of user-defined watershed outlets where source-change scenarios are applied. The function substitutes a user defined list of target reaches from the control setting `select_targetReachWatersheds` for the terminal reaches and then executes in the same way as *hydseq.R* function. Once the new `hydseq` variable is created, reaches with a value for `hydseq` are flagged as part of the source-change scenario in the `scenarioFlag` object within the `predictScenarios.list`. All reaches outside of the upstream flowlines of the `select_targetReachWatersheds` are assigned a `hydseq = NA`.

7.8 Internal RStudio session objects

The execution of the control script produces various types of R objects that are accessible during the RStudio session. These include 108 control setting objects, including the `settingsEnv` object, which contains a list of all control settings (see *sparrow_control.R* in the active sub-directory for a list). A set of internal objects are designated to store the path names associated with different types of RSPARROW files; these include `path_data`, `path_gis`, `path_main`, `path_results`, `path_src`, `path_user`, and `activeFile`. Chapter 5 also describes binary objects that are output during RSPARROW model execution and stored as external files.

The remaining dataframe objects and binary list objects are described below; these are not stored externally and are only available during an RStudio session.

7.8.1 Dataframe objects

These include the following:

- **betavalues** - data.frame of model parameters from the *parameters.csv* file.
- **data_names** - data.frame of variable metadata from the *data_Dictionary.csv* file.
- **dlvdsgn** - design matrix imported from the *design_matrix.csv* file for selected parameters.
- **data1** - data1 input file as a dataframe with large integer values replaced and *hydseq* values added, if requested.
- **sitedata.landuse** - a data.frame consisting of incremental land-use percentages for calibration sites that is calculated according to the `class_landuse` control setting. The dataframe is assigned to the global environment in `startModelRun.R` and output by the `calcIncremLandUse.R` function.
- **vsitedata.landuse** - a data.frame consisting of incremental land-use percentages for validation sites that is calculated according to the `class_landuse` control setting. The dataframe is assigned to the global environment in `startModelRun.R` and output by the `calcIncremLandUse.R` function.

7.8.2 Binary list objects

These include the following:

- **backupOptions** - A list of the saved user options for customized error handling described in sub-section 4.4.11.6.
- **class.input.list** - A list of control settings related to classification variables that is created and assigned to the global environment by `generateInputLists.R`.

```
class.input.list<-named.list(classvar,class_landuse,class_landuse_percent)
```

- **Csites.list** - A named list of sites for calibration that is created by `selectCalibrationSites.R`. The object contains information on the total set of monitoring sites, available for use in calibration after selection filters are applied (see Chapter sub-section 4.4.3). The contents include the following variables:
  - *waterid* = hydrologically ordered reach identification number (length=number of reaches)
  - *depvar* = model response variable (e.g., mean annual load) (length=number of reaches; 0 for non-monitored reaches)
  - *staid* = hydrologically ordered (upstream to downstream) unique calibration station sequence number (length=number of reaches)
  - *numsites1* = initial monitoring site count.
  - *numsites2* = number of monitoring sites after filtering for small headwater sites.
  - *numsites3* = number of monitoring sites after filtering for minimum number of reaches separating sites.
  - *numsites4* = number of monitoring sites after filtering for minimum incremental area between sites.
  - *nMon* = number of selected calibration sites with non-zero observed loads.
  - *staidseq* = unique station sequence number (*staid*) that is assigned to each reach for the nearest downstream calibration monitoring station (length=number of reaches).
  - *nMoncalsites* = number of calibration sites identified by the *calsites* system variable.
- **Csites.weights.list** - A named list containing information on the model estimation weights for the calibration monitoring sites. The variables, sorted by the hydrologically ordered station identification number *staidseq*, include:

- *NLLS_weights* = User-selected value for the control setting to enable weighted NLLS estimation of a SPARROW model (see section 4 of the control script)
- *tiarea* = sum of the incremental drainage area of reaches located between calibration monitoring sites.
- *count* = data.frame containing the vectors **staidseq** and **nirchs**.
- *weight* = the normalized weight, expressed as being proportional to the reciprocal variance of the model errors when the control setting `NLLS_weights` is set to one of three optional methods for estimating a weighted NLLS. The `NLLS_weights<-"default"` setting applies weights with a value of one in an unweighted NLLS estimation of the SPARROW model. See Chapter sub-section 4.4.4.11 for details on the methods.

- **Csites.weights.lnload.list** - A named list that is saved internally for the `NLLS_weights<-"lnload"` control setting and contains information on the weights for the calibration monitoring sites. The list is created in the *estimateWeightedErrors.R* function. The contents of the list includes:
  - *NLLS_weights* = User-selected value for the control setting to enable weighted NLLS estimation of a SPARROW model (see section 4 of the control script)
  - *weights_nlr* = the normalized weight, expressed as the reciprocal of variance as a function of the log transformed predicted load. See Chapter sub-section 4.4.4.11 for details on the methods.
  - *sqResids* = the square of the log residuals from an unweighted NLLS SPARROW model, the response variable of the weight power function.
  - *lnload* = the log predicted load from an unweighted NLLS SPARROW model, the explanatory variable for the weight power function.
  - *regnls* = an R object list consisting of 6 elements from the weight power function (nonlinear least squares fit using the *nls* R function).
- **estimate.list** - A named list of summary metrics and diagnostic output that are created by the `estimate.R` function. Different objects are contained in the list according to user control settings. The following objects are included in the `estimate.list` for the indicated control settings (see documentation Chapter sub-section 5.2.4. for description of the object contents):
  - The default contents are set to NULL.
  - Control setting `if_estimate_simulation <- "yes"` includes the objects *sparrowEsts* and *JacobResults*, which are required to enable model predictions (`if_predict<-"yes"`).
  - Control setting `if_estimate <- "yes"` includes the objects *sparrowEsts*, *JacobResults* ,*HesResults*, *ANOVA.list* , and *Mdiagnostics.list*; the object *HesResults* is NULL for the control setting `ifHess <- "no"`.
  - Control settings `if_estimate <- "yes"` and `if_validate <- "yes"` include the objects *sparrowEsts*, *JacobResults*, *HesResults*, *ANOVA.list* , *Mdiagnostics.list*, *vANOVA.list*, and *vMdiagnostics.list*.
- **estimate.input.list** - A list of control settings used in estimation and prediction that is created and assigned to the global environment by `generateInputLists.R`.

```
estimate.input.list <- named.list(ifHess, s_offset, NLLS_weights,
                                  if_mean_adjust_delivery_vars,
                                     yieldFactor,ConcFactor,confInterval,
                                     loadUnits,yieldUnits,ConcUnits,MoranDistanceWeightFunc,
                                     incr_delivery_specification,reach_decay_specification,
                                     reservoir_decay_specification)
```

- **file.output.list** - A list of control settings and relative paths used for input and output of external files that is created and assigned to the global environment by `generateInputLists.R`.

```
file.output.list <- named.list(path_master,
                               path_main,
                               path_user,
                               path_data,
                               path_gis,
                               path_results,
                               results_directoryName,
                               data_directoryName,
                               gis_directoryName,
                               run_id,
                               csv_decimalSeparator,
                               csv_columnSeparator)
```

- **mapping.input.list** - A named list of sparrow_control settings for mapping that is created and assigned to the global environment by `generateInputLists.R`.

```
mapping.input.list <- named.list(lat_limit,lon_limit,master_map_list,
                               lineShapeName,lineWaterid,
                               polyShapeName,polyWaterid,LineShapeGeo,
                               LineShapeGeo,CRStext,convertShapeToBinary.list,
                               map_siteAttributes.list,
                               if_verify_demtarea_maps,output_map_type,
                               outputESRImaps,

                               #diagnosticPlots
                               enable_plotlyMaps,add_plotlyVars,pchPlotlyCross,showPlotGrid,
                               loadUnits,
                               yieldUnits,
                               diagnosticPlotPointSize,
                               diagnosticPlotPointStyle,

                               #prediction map settings
                               predictionTitleSize,
                               predictionLegendSize,
                               predictionLegendBackground,
                               predictionMapColors,
                               predictionClassRounding,
                               predictionMapBackground,
                               lineWidth,

                               #residual maps settings
                               residual_map_breakpoints,
                               ratio_map_breakpoints,
                               residualTitleSize,
                               residualLegendSize,
                               residualColors,
                               residualPointStyle,
                               residualPointSize_breakpoints,
                               residualPointSize_factor,
                               residualMapBackground,

                               #siteAttribute maps settings
```

```
                                    siteAttrTitleSize,
                                    siteAttrLegendSize,
                                    siteAttrColors,
                                    siteAttrClassRounding,
                                    siteAttr_mapPointStyle,
                                    siteAttr_mapPointSize,
                                    siteAttrMapBackground,

                                    #scenarios
                                    scenarioMapColors)
```

- **min.sites.list** - A named list of site filtering control settings that is created and assigned to the global environment by `generateInputLists.R`. The settings include: `minimum_headwater_site_area`, `minimum_reaches_separating_sites`, and `minimum_site_incremental_area`.

```
min.sites.list<-named.list(minimum_headwater_site_area,
                           minimum_reaches_separating_sites,
                           minimum_site_incremental_area)
```

- **numsites** - An integer value equal to the number of calibration sites.
- **scenario.input.list** - A list of control settings related to source-change scenarios that is created and assigned to the global environment by `generateInputLists.R`.

```
scenario.input.list<-named.list(select_scenarioReachAreas,
                                select_targetReachWatersheds,
                                scenario_name,
                                scenario_map_list,
                                scenarioMapColors,
                                scenario_sources,
                                scenario_factors,
                                landuseConversion)
```

- **sitedata.demtarea.class** - A vector of decile values corresponding to the total drainage area classes for the calibration sites for model performance plots in the *(run_id)_diagnostic_plots.html* file.
- **sitegeolimits** - A named list of minimum and maximum values of the calibration station latitudes and longitudes that is created by `findMinMaxLatLon.R`.
- **Vsites.list** - A named list of sites for validation that is created by `selectValidationSites.R`. The contents include the following variables:
  - *waterid* = hydrologically ordered reach identification number (length=number of reaches)
  - *staid* = hydrologically ordered (upstream to downstream) unique calibration station sequence number (length=number of reaches)
  - *vstaid* = hydrologically ordered (upstream to downstream) unique calibration station sequence number (length=number of reaches)
  - *depvar* = model response variable (e.g., mean annual load) (length=number of reaches; 0 for non-monitored reaches)
  - *vdepvar* = model response variable (e.g., mean annual load) (length=number of reaches; 0 for non-monitored reaches)
  - *nMon* = number of selected calibration sites with non-zero observed loads.
  - *vic* = number of selected validation sites with non-zero observed loads.

  - *staidseq* = unique station sequence number (*staid*) that is assigned to each reach for the nearest downstream calibration monitoring station (length=number of reaches).
  - *vstaidseq* = unique station sequence number (*staid*) that is assigned to each reach for the nearest downstream validation monitoring station (length=number of reaches).

- **vsitedata.demtarea.class** - A vector of decile values corresponding to the total drainage area classes for validation sites for model performance plots in the *(run_id)_validation_plots.html* file.

7.9 Future software design needs

- *Development of the capability to host the R Shiny Decision Support System (DSS) mapper and existing SPARROW models on a remote Web server.* This capability would eliminate the need for end-users (e.g., water resource managers) to install and maintain copies of R, RStudio, and the SPARROW data on their personal computer. RSPARROW version 1.1, which executes R Shiny from a Web browser that is independent of RStudio, reflects recent changes in the RSPARROW coding architecture that can assist efforts to develop remote server hosting of RSPARROW data and software.
- *Development of a more efficient R-based SPARROW NLLS model estimation algorithm.* The RSPARROW function employs Fortran looping to sum loads within the reach network, as done in the SAS SPARROW Fortran subroutine. However, SAS execution of the SPARROW NLLS function is more than two times faster than that of the RSPARROW NLLS function, *estimateFeval.R*. Evaluations of alternative approaches for performing matrix calculations yielded negligible differences in execution times.
- *More efficient calculation of the second-order Hessian estimates of the model coefficient standard errors.* The SAS SPARROW calculation of these estimates is more than 1.5 times faster than in RSPARROW.
- *Semi-automation of the SPARROW model evaluation and development process, with capabilities to automatically evaluate alternative explanatory variables.* The manual evaluations of candidate models that build in complexity as illustrated in Chapter 6 provide an informative incremental understanding of the merits and limitations of different SPARROW models, based on an examination of many types of model diagnostics. However, this process could be made more efficient through the development of optional methods for automating the testing of explanatory variables.
- *Testing and development of the RSPARROW functions and control script for use on IOS (Apple) and Linx systems.* Two features of the current RSPARROW functions have Windows dependencies: the automated pop-up of CSV control files and the batch-mode operation.

8 REFERENCES

Alexander, R.B., Smith, R.A., and Schwarz, G.E. (2000), Effect of stream channel size on the delivery of nitrogen to the Gulf of Mexico, Nature, v. 403, doi:10.1038/35001562.

Alexander, R.B., Smith, R.A. Schwarz, G.E., Preston, S.D., Brakebill, J.W., Srinivasan, R., and Pacheco, P.A. (2001), Atmospheric nitrogen flux from the watersheds of major estuaries of the United States: An application of the SPARROW watershed model, *in* Nitrogen loading in coastal water bodies: An atmospheric perspective, American Geophysical Union Monograph 57, R. Valigura, R. Alexander, M. Castro, T. Meyers, H. Paerl, P. Stacey, and R.E. Turner (eds.) 119-170.

Alexander, R.B., Elliott, S., Shankar, U., McBride, G.B. (2002), Estimating the sources of nutrients in the Waikato River Basin, New Zealand, Water Resources Research 38: 1268-1290.

Alexander, R.B., Boyer, E.W., Smith, R.A., Schwarz, G.E., and Moore, R.B. (2007), The role of headwater streams in downstream water quality, Journal American Water Resources Association, 43(1): 41-59.

Alexander, R.B., Smith, R.A., Schwarz, G.E., Boyer, E.W., Nolan, J.V., and Brakebill, J.W. (2008), Differences in phosphorus and nitrogen delivery to the Gulf of Mexico from the Mississippi River Basin, Environ. Sci. Technol., v. 42(3), 822-830. DOI:10.1021/es0716103

Alexander, R.B. (2015), Advances in quantifying streamflow variability using hierarchical Bayesian methods with SPARROW, PhD dissertation, The Pennsylvania State University, College of Agricultural Sciences, 281p.

Alexander, R.B., Schwarz, G.E., and Boyer, E.W. (2019a), Advances in quantifying streamflow variability across continental scales: 1. Identifying natural and anthropogenic controlling factors in the USA using a spatially explicit modeling method, Water Resources Research, v. 55, 10,893-10,917.

Alexander, R.B., Schwarz, G.E., and Boyer, E.W. (2019b), Advances in quantifying streamflow variability across continental scales: 2. Improved model regionalization and prediction uncertainties using hierarchical Bayesian methods, Water Resources Research, v. 55, 11,061-11,087.

Anning, D.W. and Flynn, M.E. (2014), Dissolved-solids sources, loads, yields, and concentrations in streams of the conterminous United States, U.S. Geological Survey Scientific Investigations Report 2014-5012, 112 p.

Appling, A.P., Leon, M.C., and McDowell, W.H. (2015), Reducing bias and quantifying uncertainty in watershed flux estimates: The R package loadflex, Ecosphere, 6(12), 1-25, DOI: 10.1890/ES14-00517.1 (also see *loadflexBatch* with extended SPARROW support capabilities to execute for multiple sites, constituents, and models: https://github.com/USGS-R/loadflexBatch)

Beaulac, M.N. and Reckhow, K.H. (1982), An examination of land use nutrient export relationships, Journal of the American Statistical Association, v. 18, p. 1013-1024. DOI: 10.1111/j.1752-1688.1982.tb00109.x

Benoy, G.A., Jenkinson, R.W., Robertson, D.M., and Saad, D.A. (2016), Nutrient delivery to Lake Winnipeg from the Red—Assiniboine River Basin – A binational application of the SPARROW model, Canadian Water Resources Journal / Revue canadienne des ressources hydriques, 41:3, 429-447, DOI: 10.1080/07011784.2016.1178601

Bergamaschi, B.A., Smith, R.A., Sauer, M.J., Shih, J.S., and Ji, L. Chapter 6. Terrestrial fluxes of nutrients and sediment to coastal waters and their effects on coastal carbon storage in the eastern United States (2014), *in* Baseline and projected future carbon storage and greenhouse-gas fluxes in ecosystems of the eastern United States, Zhiliang Zhu and Bradley Reed (Eds.), USGS Professional Paper 1804.

Brakebill, J.W., Ator, S.W., and Schwarz, G.E. (2010), Sources of suspended-sediment flux in streams of the Chesapeake Bay Watershed: A regional application of the SPARROW model, Journal of the American Water Resources Association, 46(4), 757-776.

Brakebill, J.W. and Terziotti, S.E. (2011), A Digital Hydrologic Network Supporting NAWQA MRB SPARROW Modeling–MRB_E2RF1, U.S. Geological Survey Digital Data Release, Reston, VA. https://water.usgs.gov/GIS/metadata/usgswrd/XML/mrb_e2rf1.xml.

Duan, N. (1983), Smearing estimate-A nonparametric retransformation method: Journal of the American Statistical Association, v. 78, p. 605-610.

Duan, W. L.; He, B.; Takara, K.; Luo, P. P.; Nover, D.; Hu, M. C. (2015), Modeling suspended sediment sources and transport in the Ishikari River basin, Japan, using SPARROW, 19, 1293-1306, DOI: https://doi.org/10.5194/hess-19-1293-2015,

Elliott, A.H., Alexander, R.B., Schwarz, G.E., Shankar U., Sukias, J.P.S., and McBride, G.B. (2005), Estimation of nutrient sources and transport for New Zealand using the hybrid physical-statistical model SPARROW, New Zealand Journal of Hydrology, 44, 1-27.

Garcia, A.M., Alexander, R.B., Arnold, J.G., Norfleet, L., White, M.J., Robertson, D.M., and Schwarz, G.E. (2016), Regional effects of agricultural conservation practices on nutrient transport in the Upper Mississippi River Basin, Environmental Science and Technology, 50: 6991-7000. DOI: doi.org/10.1021/acs.est.5b03543 IPDS IP-067273

Harris, S., Elliott, S., McBride, G., Shankar, U., Quinn, J., Wheeler, D., Wedderburn, L., Hewitt, A., Gibb, R., Parfitt, R., Clothier, B., Green, S., Montes de Oca Munguía, O., Dake, C., and Rys, G. (2009), Integrated assessment of the environmental, economic and social impacts of land use change using a GIS format – the CLUES model, New Zealand Agricultural and Resource Economics Society Conference Proceedings, Nelson, New Zealand, August 27-28, 2009. DOI: 10.22004/ag.econ.97166

Helsel, D.R., and Hirsch, R.M. (2002), Statistical methods in water resources, U.S. Geological Survey Techniques of Water-Resources Investigations Book 4, Chapter A3.

Herrmann, M., Najjar, R.G., Kemp, W.M., Alexander, R.B., Boyer, E.W., Cai, Wei-Jun, Griffith, P.C., Kroeger, K.D., McCallister, S.L., and Smith, R.A. (2015), Net ecosystem production and organic carbon balance of U.S. East Coast estuaries: A synthesis approach, Global Biogeochemical Cycles, 29: 96-111. doi.org/10.1002/2013GB004736

Hirsch, Robert M. (2014), Large biases in regression-based constituent flux estimates: Causes and diagnostic tools, Journal of the American Water Resources Association (JAWRA) 50(6):1401-1424. doi: 10.1111/jawr.12195

Hirsch, R.M., and De Cicco, L.A. (2015), User guide to Exploration and Graphics for RivEr Trends (EGRET) and dataRetrieval: R packages for hydrologic data (version 2.0, February 2015), U.S. Geological Survey Techniques and Methods book 4, chap. A10, 93 p. https://dx.doi.org/10.3133/tm4A10. (alse see http://usgs-r.github.io/EGRET/)

Hirsch, R. M., Moyer, D.L., and Archfield, S.A. (2010), Weighted Regressions on Time, Discharge, and Season (WRTDS), With an application to Chesapeake Bay river inputs, Journal of the American Water Resources Association (JAWRA) 46(5):857-880. doi: 10.1111/j.1752-1688.2010.00482.x

Hoos, A.B., and McMahon, G. (2009), Spatial analysis of instream nitrogen loads and factors controlling nitrogen delivery to streams in the southeastern United States using spatially referenced regression on watershed attributes (SPARROW) and regional classification frameworks, 23(16), 2275-2294.

Hoos, A.B., and Roland, V.L. II (2019), Spatially referenced models of streamflow and nitrogen, phosphorus, and suspended-sediment loads in the Southeastern United States, U.S. Geological Survey Scientific Investigations Report 2019–5135, 91 p.

Jobson, H.E. (1996), Prediction of traveltime and longitudinal dispersion in rivers and streams, U.S. Geological Survey Water Resources Investigations 1996-4013, 69p.

Lee C., R.M. Hirsch, G.E. Schwarz, D.J. Holtschlag, S.D. Preston, C.G. Crawford, and Vecchia, S.V. (2016), An evaluation of methods for estimating decadal stream loads, Journal of Hydrology, v. 542, p. 185-203. doi:10.1016/j.jhydrol.2016.08.059

Lee, C.J., Hirsch, R.M., and Crawford, C.G. (2019), An evaluation of methods for computing annual water-quality loads, U.S. Geological Survey Scientific Investigations Report 2019–5084, 59 p. doi: https://doi.org/10.3133/sir20195084.

Leopold, L.B. and Maddock, T. (1953), The hydraulic geometry of stream channels and some physiographic implications, U.S. Geological Survey Professional Paper 252, 57p.

Miller, M.P., Capel, P.D., Garcia, A.M., and Ator, S.W. (2019), Response of nitrogen loading to the Cheasapeake Bay to source reduction and land use change scenarios: A SPARROW-informed analysis, Journal of the American Water Resources Association, 56, 100-112.

Miller, M. P., S.G. Buto, D.D. Susong, and Rumsey, C.A. (2016), The importance of base flow in sustaining surface water flow in the Upper Colorado River Basin. Water Resources Research, (52), 3547-3562.

Moriasi, D.N., J.G. Arnold, M.W. Van Liew, R.L. Bingner, R.D. Harmel, and Veith, T.L. (2007), Model evaluation guidelines for systematic quantification of accuracy in watershed simulations, Transactions of the American Society of Agricultural and Biological Engineers, 50(3), 885-900.

Moore, R.B., C.M. Johnston, K.W. Robinson, and Deacon, J.R. (2004), Estimation of total titrogen and phosphorus in New England streams using spatially referenced regression models, U.S. Geological Survey Scientific Investigations Report 2004-5012.

Najjar, R., Herrmann, M., Alexander, R., Boyer, E.W., Burdige,D., Butman, D., Cai, W.J., Canuel, E. A., Chen, R., Friedrichs., M.A.M., Feagin, R.A., Griffith, P., Hinson, A. L., Holmquist, J.R., Hyde, K., Kemp, W.M., Kroeger, K.D., Mannino, A., McCallister, S.L., McGillis, W.R., M. Mulholland, R., Pilskaln, C., Salisbury, J., Signorini, S., Tian, H., Tzortziou, M., Vlahos, P., Wang, Z.A., and Zimmerman, R.C. (2018), Carbon budget of tidal wetlands, estuaries, and shelf waters of Eastern North America, 32(3), 389-416.

Preston, S.D., Alexander, R.B., and Wolock, D. M. (2011a), Sparrow modeling to understand water-quality conditions in major regions of the United States: A featured collection introduction, Journal of the American Water Resources Association, v. 47(5), 887-890, doi:10.1111/j.1752-1688.2011.00585.x.

Preston, S.D., Alexander, R.B., Woodside, M.D., and Hamilton, P.A. (2011b), SPARROW MODELING–Enhancing understanding of the Nation's water quality, U.S. Geological Survey Fact Sheet 2009-3019.

Qian, S.S., Reckhow, K.H., Zhai, J., McMahon, G. (2005), Nonlinear regression modeling of nutrient loads in streams: A Bayesian approach, 41(7), doi:10.1029/2005WR003986

Robertson, D.M., Saad, D.A, Benoy, G.A., Vouk, I., Schwarz, G.E., Laitta, M.T. (2019) Phosphorus and nitrogen transport in the binational Great Lakes Basin estimated using SPARROW watershed models, Journal of the American Water Resources Association, 55, 1401-1424.

Robertson, D.M., Schwarz, G.E., Saad, D.A, and Alexander, R.B. (2009), Incorporating uncertainty into the ranking of SPARROW model nutrient yields from Mississippi/Atchafalaya River basin watersheds, Journal of the American Water Resources Association, 42, 534-549.

Robertson, D.M., and Saad, D.A. (2011), Nutrient inputs to the Laurentian Great Lakes by source and watershed estimated using SPARROW watershed models, Journal of the American Water Resources Association, 47(5), 1011-1033, doi:10.1111/j.1752-1688.2011.00574.x

Runkel, R.L., Crawford, C.G., and Cohn, T.A. (2004), Load Estimator (LOADEST): A FORTRAN program for estimating constituent loads in streams and rivers, U.S. Geological Survey Techniques and Methods Book 4, Chapter A5, 69p.

Saad, D.A., Schwarz, G.E., Argue, D.M., Anning, D.W., Ator, S.A., Hoos, A.B., Preston, S.D., Robertson, D.M., and Wise, D. (2019), Estimates of long-term mean daily streamflow and annual nutrient and suspended-sediment loads considered for use in regional SPARROW models of the conterminous United States, 2012 base year, U.S. Geological Survey Scientific Investigations Report 2019-5069. https://doi.org/10.3133/sir20195069

Schmadel, N.M., Harvey, J.W., Alexander, R.B., Schwarz, G.E., Moore, R.B., Eng, K., Gomez-Velez, J.D., Boyer, E.W., and Scott, D. (2018), Thresholds of lake and reservoir connectivity in river networks control nitrogen removal, Nature Communications. DOI: 10.1038/s41467-018-05156-x

Schmadel, N.M., Harvey, J.W., Schwarz, G.E., Alexander, R.B., Gomez-Velez, J.D.,Scott, D., and Ator, S.W. (2019), Small ponds in headwater catchments are a dominant influence on regional nutrient and sediment

budgets, Geophysical Research Letters. DOI: 10.1029/2019GL083937

Schwarz, G.E., Hoos, A.B., Alexander, R.B., and Smith, R.A. (2006), The SPARROW surface water-quality model: Theory, application, and user documentation, U.S. Geological Survey Techniques and Methods Report, Book 6, Chapter B3. (USGS SAS SPARROW software: https://www.usgs.gov/software/sparrow-modeling-program)

Smith, R.A., Schwarz, G.E., and Alexander, R.B. (1997), Regional interpretation of water-quality monitoring data, Water Resources Research, v. 33(12), 2781-2798, doi:10.1029/97WR02171

Wellen, C., Arhonditsis, G. B., Labencki, T., and Boyd, D. (2012), A Bayesian methodological framework for accommodating interannual variability of nutrient loading with the SPARROW model, Water Resources Research, 48(10). https://doi.org/10.1029/2012WR011821

Wellen, C., Arhonditsis, G. B., Labencki, T., & Boyd, D. (2014), Application of the SPARROW model in watersheds with limited information: a Bayesian assessment of the model uncertainty and the value of additional monitoring. Hydrological Processes, 28(3), 1260–1283. https://doi.org/10.1002/hyp.9614

9 ACKNOWLEDGMENTS

We thank Matt Miller and Bradley Huffman of the U.S. Geological Survey (USGS) for their reviews and suggestions on the documentation and R code. Additional suggestions and testing of RSPARROW were provided by Ana Garcia, Laura Nagy, Victor Roland, Kristina Hopkins, David Saad, and Allison Appling of the USGS, Marcelo Luiz de Souza and Alexandre de Amorim of the National Water Agency of Brazil (ANA), Eber Josér Anda and Lucus Goncalves of the Geological Survey of Brazil (CPRM), and Jayanthi Srikishen and Mohammad Alhamdan of the Universities Space Research Association at NASA (National Aeronautics and Space Administration). Additional support was provided by Gregory Schwarz, Laura De Cicco, Anne Hoos, and Mike Ierardi of the USGS. Reviews and suggestions were also provided on the R Shiny interface and decision-support tool by Steve Preston, David Wolock, and Michael Woodside of the USGS. We also thank Marcus Beck at the Tampa Bay Estuary Program, who contributed R code for the R Shiny *leaflet* interactive maps in the 1.1 version.

We extend our appreciation to the participants in SPARROW training workshops, held in Brasilia, Brazil (May 29-June 2, 2017; sponsored by ANA and CPRM) and Baltimore, Maryland USA (September 13, 2018; sponsored by the USGS), for their feedback on preliminary versions of the RSPARROW software.

Funding for the production of RSPARROW was provided by the National Water Quality Assessment Project of the USGS National Water Quality Program, with additional support provided by the Integrated Modeling and Prediction Division (IMPD) of the USGS Water Mission Area and the USGS Maryland-Delaware-DC Water Science Center. Funding was also provided by the Brazilian federal agencies ANA and CPRM as part of a collaborative agreement with the U.S. Geological Survey. Additional support for RSPARROW was provided by the Research Opportunities in Space and Earth Sciences (ROSES) 2016, Applied Sciences Water Resources program at NASA.

10 DISCLAIMER

This software has been approved for release by the U.S. Geological Survey (USGS). Although the software has been subjected to rigorous review, the USGS reserves the right to update the software as needed pursuant to further analysis and review. No warranty, expressed or implied, is made by the USGS or the U.S. Government as to the functionality of the software and related material nor shall the fact of release constitute any such warranty. Furthermore, the software is released on condition that neither the USGS nor the U.S. Government shall be held liable for any damages resulting from its authorized or unauthorized use.

Any use of trade, firm, or product names is for descriptive purposes only and does not imply endorsement by the U.S. Government.

11 CITATION FOR THE PUBLICATION

Alexander, Richard B., and Gorman Sanisaca, Lillian. (2019). RSPARROW: An R system for SPARROW modeling [Software release]. U.S. Geological Survey. DOI: https://doi.org/10.5066/P9UAZ6FO