# Development of tools and interfaces for the implementation, pre-processing, and sensitivity analysis of the SWAT model in an R programming environment

Report to the
Austrian Marshall Plan Foundation

Christoph Schürz

March 2019

# Table of Contents

# List of Figures

# Abstract

The assessment of the impact of future system changes, such as climate or land use change on our ecosystems is becoming increasingly important. Integrative ecosystem models, such as the SWAT model, are useful tools for the evaluation of our water resources. A profound knowledge of the uncertainties of model predictions (and the respective sensitivities of model inputs) is however essential for decision making. Unfortunately, uncertainty assessment is not yet standard procedure in today´s model applications. Available software is very limited to specific cases and lacks the possibility for adaptation. An implementation of the SWAT model into the R programming environment however, shows great potential to develop highly flexible workflows for SWAT model applications and the analysis of model results. Based on my recent work in developing SWAT modelling workflows in R, the central goal of this research stay was to increase the chances for uncertainty and sensitivity analysis to become a standard procedure in SWAT model applications and to make these methods easily accessible to SWAT users. With the development of user friendly SWAT modelling workflows in R, involving state-of-the-art methods for uncertainty and sensitivity analysis, the implementation of the most recent developments of the SWAT model and the planned provision of detailed documentation and study material, this research stay has the potential to make an essential contribution to the SWAT community.

# 1 Introduction

## 1.1 Modeling the impacts on our water resources under future change

The management of our landscapes, changes in our agricultural systems and the impact of climate change pose some of the key challenges for maintaining the quality and quantity of our future fresh water resources in the future (Rockström et al., 2009; Steffen et al., 2015). In particular, agricultural practices impact the nitrogen and the phosphorous cycles and a strong future increase is expected globally (Steffen et al., 2015; Sutton and Bleeker, 2013). Thus, the quality of our water resources has been deteriorating on a global scale since the 1990's (UNEP, 2016).

The assessment of the impact of future system changes, such as climate or land use change on our ecosystems is becoming increasingly important. Integrative eco-hydrological models are useful tools assess the quantity and quality of our water resources. A few eco-hydrological models are available to model water and nutrient fluxes on a watershed scale. The available models strongly differ in their complexity and the representation of the relevant environmental processes. While MONERIS (MOdelling Nutrient Emissions in RIver Systems; Venohr et al., 2010) calculates long-term average nutrient budgets from that are introduced into the water bodies via different pathways, more complex models such as the Hydrological Simulation Program FORTRAN (HSPF; Bicknell, 1997), or the Soil and Water Assessment Tool (SWAT; Arnold et al., 1998) simulate environmental processes on a daily scale and spatially distributed.

The SWAT model is by far the most frequently used model in the water quality context on catchment scale (Mannschatz et al., 2016). SWAT is a continuous, process-based semi-distributed model. The processes calculated on the land phase include water balance components such as interception; infiltration; shallow and deep percolation; surface runoff; lateral flow; groundwater flow; plant uptake and evapotranspiration; or the pathways of nutrients such as the input through atmospheric deposition or fertilizer application, the transformation into other forms of a nutrient, and nutrient transport (Neitsch et al., 2011). The model's flexibility to consider the relevant processes of the water and the nutrient cycles in a holistic modeling approach allows to implement SWAT to simulate the impact of future changes (such as land use change, climate change, or urban development) on different variables of the water cycle (Mehdi et al., 2015; Schürz et al., 2019; Wagner et al., 2017), water quality (Guse et al., 2015; Mehdi et al., 2015; Schürz et al., 2019; Teshager et al., 2016), or sediment yield (Bieger et al., 2013).

## 1.2 Flexible open source tools for the challenges in eco-hydrological modeling

To draw informed conclusions from environmental modeling studies a profound knowledge about the relevant processes to simulate is essential. Thus, a model should be able to represent the relevant processes appropriately, but also the impacts on the relevant processes due to possible future changing conditions (Clark et al., 2008, Chiew and Vaze (2015); Milly et al., 2008). Eco-hydrological models are, however, always a simplification of the real world processes. Different model setups and different sets of parameters in a model can perform equally well to reproduce historical observations, an issue well-known as equifinality in the literature (Beven, 1996, 2006; Beven and Freer, 2001; Schulz et al., 1999). Further, the simulation of future changes in climate, or the land use are always uncertain. In environmental impact studies different plausible future changes are typically analyzed with scenario based modeling, where an array of discrete scenarios cover a range of plausible future developments (Clark et al., 2016).To approach such challenges, model parameter calibration and parameter sensitivity analysis, model optimization, ensemble modeling, scenario based modeling are standard elements of most environmental impact assessments. Albeit, these procedures are similar in their implementation and follow straight-forward principles, all modeling studies have their peculiarities and therefore require flexible tools for their analysis.

The SWAT community has been developing tools in the past, that support a modeler in the model setup (using graphical GIS user interfaces such as ArcSWAT (Winchell et al., 2015) and QSWAT (Dile et al., 2016, 2018)), in the evaluation of simulation outputs (White et al., 2014), or in the parameter calibration and sensitivity analysis (Abbaspour, 2015). The available tools strongly promote the user-friendliness of SWAT. Without doubt, these tools are a main reasons for the wide application of the SWAT model. The software SWAT-CUP for instance, provides several methods for model calibration and parameter sensitivity analysis and allows an efficient processing of many standard model applications. Yet, SWAT-CUP is limited to the implemented functionality and does not support any major user defined adaption of the provided methods.

Open source high-level programming languages, such as python (python.org), R (R Core Team, 2019), or julia (Bezanson et al., 2012) are becoming increasingly popular in the environmental sciences, as they provide great flexibility in the preparation and analysis of environmental data. A major strength of these programming environments is the large number packages (or libraries) provided and peer-reviewed by the respective communities. The available packages allow to efficiently implement and apply state-of-the-art methods into any data analysis workflow. The number of R packages hosted on the Comprehensive R Archive Network (CRAN), for instance, grew almost exponentially from under 1000 packages in the year 2007 to over 13000 packages in

2018. Several of the R packages hosted on CRAN are useful for environmental model applications, such as `sensitivity` (Iooss et al., 2018) and `fast` (Reusser, 2015) for parameter sensitivity analysis, `lhs` (Carnell, 2019) or `randtoolbox` (Dutang C. and P., 2018) for parameter sampling and model calibration, or `ggplot2` (Wickham, 2016), `dygraphs` (Vanderkam et al., 2018), `plotly` (Carson Sievert, 2018) and many others to explore and visualize simulation results. An even larger number of R packages is indirectly useful in any data analysis.

Several SWAT implementations and tools are available for python and R. Carla Camargos (2018) presented an implementation of SWAT into python to employ the functionality of the python library `SPOTPY` (Houska et al., 2015) for parameter sensitivity analysis and parameter calibration of SWAT projects. Joseph and Guillaume (2013) provided a step by step recipe to implement a parallelized MCMC algorithm for SWAT in R. R packages, such as `EcoHydRology` (Fuka DR, 2018), `SWATmodel` (Fuka et al., 2014; Fuka, 2014), or `R-SWAT-FME` (Wu and Liu, 2012, 2014) provide solutions to execute and analyze SWAT projects from within an R instance.

## 1.3 Central goals and structure

Despite the availability of solutions to implement the SWAT model in R (or any other open source high-level programming language), none of the available R packages yet provide the flexibility and usability to find a great acceptance in the SWAT community and to sufficiently support the model users with processing their model applications in R. Key to a wider implementation of SWAT modeling workflows in R is the provision of an R package that seamlessly integrates a SWAT project into the R environment. Yet, a sustainable solution that finds great acceptance in the SWAT user community requires also detailed documentation, tutorials and appropriate training. Consequently, the outcome of this work should provide the following:

- The R package `SWATplusR` will be introduced as a new approach to seamlessly integrate the SWAT model into any R programming workflow. The development of `SWATplusR` bases on previously developed R packages, tools and scripts that proved to be essential tools in previous SWAT-R projects.
- The latest version of the SWAT model (SWAT+) will be implemented in `SWATplusR`. The cooperation with the SWAT developers ensures a seamless incorporation of the new model version.
- A detailed documentation of the `SWATplusR` package should provide guidance in the application of the R package functionality. The documentation should further contain examples and tutorials that provide best practice examples for standard modeling tasks such as parameter sensitivity analysis, or model calibration.
- The application of the `SWATplusR` package and typical modeling workflows in the R environment will be prepared in form of on-line learning materials. These will be published openly accessible and free of use.

- Based on the established learning materials a workshop format will be developed that should provide the opportunity to learn these methods first hand on a regular basis (e.g. together with SWAT user conferences)

The following document will cover the above mentioned goals as follows: In section 2 I will, in brief, outline the previous work on R packages that I developed and that were substantially tested and implemented in previous SWAT modeling studies. Section 3 provides an overview for the `SWATplusR` package. I will address the design decisions that were made prior to the development of the `SWATplusR` package to delineate the capabilities but also the limitations of the R package. Further, the R packages' functionality and its implementation in modeling workflows is explained. Section 4 demonstrates the application of the R package and its compatibility with other R packages. Section 4 is basically a collection of study materials that summarizes tasks that are commonly performed in any modeling study are illustrated using `SWATplusR`. That collection of study materials is the result of a first workshop held at the end of my research stay. The materials developed for that course and the experiences and feedbacks collected in this workshop will be the basis for future workshop formats. The outlook for future work and the possible implementation in teaching (e.g. on-line tutorials and workshops) will be addressed in section 5.

# 2 Preceding work and R packages

The SWAT model setup requires very detailed input data, that is often unavailable with an adequate spatial or thematic resolution. In other cases the parameters that are mandatory for the model setup are not available and inferring these data from the available data is necessary. Over the last years our working group developed methods to overcome the limitations of missing input data and to eventually set up SWAT models that adequately represent the study areas by employing limited data that is often available.

Detailed agricultural land use data is usually very limited. Yet, agricultural statistics are often available on different administrative levels. The R package `SWATfarmR` (Schürz et al., 2017a) allows the SWAT user to infer spatially distributed management schedules for the agricultural land uses in a SWAT model setup from agricultural statistics and climate data based on simple rules. SWAT requires detailed soil data for a model setup. In many cases field measurements are unavailable. Though, global soil data sets strongly improved in spatial resolution and prediction accuracy. A very comprehensive global soil data base is provided by the SoilGrids project (Hengl et al., 2017). To make use of these substantial data in SWAT model setup we developed the R package `soilgridr` (Schürz, 2018b) that allows to develop the required SWAT soil input data set from SoilGrids data. Climatic data is often provided as gridded NetCDF or binary data. SWAT however requires the weather input data as time series data for points in space that are assigned to sub units of a model setup. With the `aRastoCAT` (Schürz, 2018a) R package we provide a solution to derive the weather input data for a SWAT model setup from gridded weather data.

With the `SWATpasteR` (Schürz et al., 2017b) R package we developed an R package that allowed to integrate SWAT projects into R programming workflows. It based on a highly rigid data structure and the focus of the R package was to provide an enclosed environment for model calibration and sensitivity analysis. The consequence was, however, that this framework quickly became very inflexible in many applications and always required expert knowledge of the developers in its application. Although it was heavily tested and sucessfully applied in several model applications (Odusanya et al., 2019; e.g.; Schürz et al., 2019) it was never released; mostly due to its inflexibility. Based on all previous R package developments we identified aspects that were limiting for the users, but also processes that were intuitively adopted. Most important was however the conclusion that many steps of a SWAT-R modeling workflow were integrated in several R packages, implemented in inflexible ways, or the workflows were too enclosed in their own environments and the user could therefore not benefit from the functionality of other R packages. Consequently, I decided to rethink the design of the modeling workflows and the corresponding R packages. The `SWATplusR` package as the 'communication interface' between the SWAT project and the R environment is a result of that.

# 3 The ‘SWATplusR‘ package

## 3.1 Design decisions

The main design goal for `SWATplusR` was to reduce the functionality to one central, but essential task, of linking the SWAT project on the local hard drive with the users workflows in the R programming environment. Minimum requirements, such as modifying model parameters, controlling simulation periods and specific settings for the model simulation, or defining model outputs that should be extracted from a simulation, have to be facilitated by the R package in order to seamlessly integrate SWAT in modeling studies that are performed entirely in R. Moreover, the handling of SWAT projects should be free of unintended side effects, intuitive in usability, computationally efficient and should be possible on different operation system platforms. To promote the implementation of these essential key aspects in the development of `SWATplusR` I defined four central design goals for the R package.

### 3.1.1 Safety and usability

A SWAT project consists of a large number of model input files that define specifications such as the model parametrization, the linking between modeling units (e.g. subbasins, aquifers, or hydrological response units (HRUs)), or the driving model inputs (e.g. weather inputs). Modifications in these input files are, once done, hard to track and to reproduce. Thus, a central requirement of `SWATplusR` is that the original SWAT projects should never be altered in any way when the model is processed in R (e.g. when parameters are modified, or simulation periods are set). All modifications in the SWAT input files will be performed in temporary images of the original SWAT project. Further, `SWATplusR` should provide a maximum of usability. All relevant parameters that control a model run are definable in R functions. The terminology of the functions should clearly define their unintended task and all parameters and variables use self-explaining names. Consequently, the code to execute a SWAT model should be easy to read and the implementation of `SWATplusR` should therefore also be accessible to non-expert R users.

### 3.1.2 Compatibility and tidiness

The `data.frame` (R Core Team, 2019) is the standard format to handle and organize data in R. Of course there are further, not less relevant data structures available in R.

Yet, the `data.frame` is the most accessible structure to R users (particularly to users that are new to R). `data.frames` (or modern derivatives such as `tibbles` (Müller and Wickham, 2019)) provide variables in columns and corresponding observations of all variables in rows. Parts of the R community strongly promote a concept for data structures called ‘tidy data’ to easier facilitate data analysis and eventually to save resources on cleaning the data (Ross et al., 2017; Wickham, 2014). The structure of the `SWATplusR` simulation outputs comply with the ‘tidy data’ concept. Independent of the defined SWAT output variables, the simulated output that is returned by the R function has a clear and consistent structure. Each simulated variable forms a table where each simulation is a column and each simulation time step is an observation for the respective time step (see Fig. 3.1). The clear and consistent output design should greatly facilitate the compatibility with other R packages, where often the inputs are required as `data.frames`.

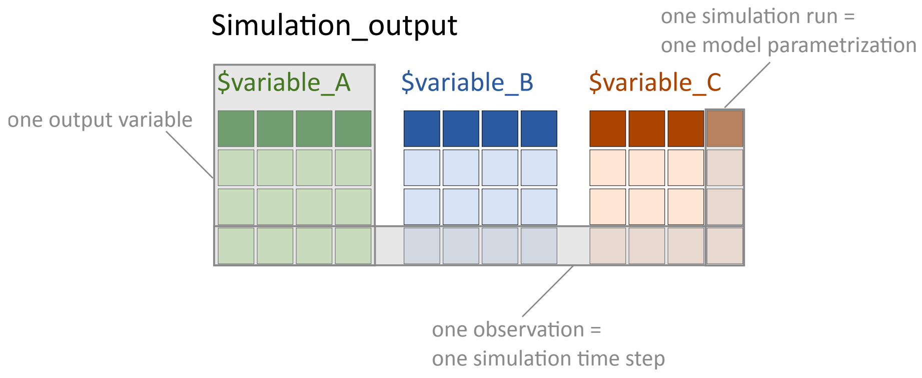

Figure 3.1: The tidy frame work applied to ‘SWATplusR‘.

### 3.1.3 Platform and model revision independence

A great benefit of the R package `SWATmodel` is to ability to execute the SWAT model independent of the operating system. Unfortunately, `SWATmodel` directly implements the source code of simplified revisions of SWAT2009 and SWAT2012 into the R package. Such approach, however, restricts the users to one specific revision of the SWAT model. The SWAT model is, however, constantly updated and a large number of different revisions are in use. Therefore, an R package must be compatible with all recent SWAT model revisions. The goal for `SWATplusR` was to provide both, platform independence (Windows and Unix) and the flexibility to use any recent SWAT model revision in R.

### 3.1.4 Performance and reliability

The computation time of large SWAT model setups can be a limiting factor in many case studies. Model calibration and parameter sensitivity analysis may require several thousand model evaluations. To reduce the total computation time of large SWAT projects and to efficiently use the computational resources, `SWATplusR` provides parallel computing of SWAT simulations. In many SWAT applications a large number of variables are of interest in the analysis. The analysis of a large number of model evaluations and long time series of a large number of variables can easily result in data sets with sizes that cannot be stored in the computers RAM anymore. Therefore, `SWATplusR` also provides the option to store the simulation runs incrementally on the hard driver rather than keeping the simulations in the RAM storage. A benefit of that option is that, large SWAT projects can also be executed on computers with small RAM storage. Further, the incremental saving of simulation results provides safety in the execution of very long simulations, as possible crashes do not result in complete data losses then.

## 3.2 Functionality and programming workflow in R

`SWATplusR` is a package developed for the high-level programming language R. Therefore, to make use of the functionality of `SWATplusR` the installation of R is required. Further, the functionality of `SWATplusR` depends on other R packages, that have to be installed before the user can install `SWATplusR`. More detail on the R package dependencies and the installation of the R packages is outlined in Section 4. `SWATplusR` provides the link between existing SWAT projects and the R programming environment. Therefore, the application of `SWATplusR` requires an existing set up SWAT project folder together with the correct revision of the SWAT executable file.

The defined design goals resulted in a programming workflow provided with `SWATplusR` that is illustrated in Fig. 3.2. With in total 5 functions the functionality was reduced to the essential minimum that is required to perform SWAT modeling task. The sections below outline the available functionality provided by these functions and describes how these functions implement a SWAT project in R. The application of the functions and how the simulation outputs can be utilized in workflows together with other R packages is demonstrated in Section 4.

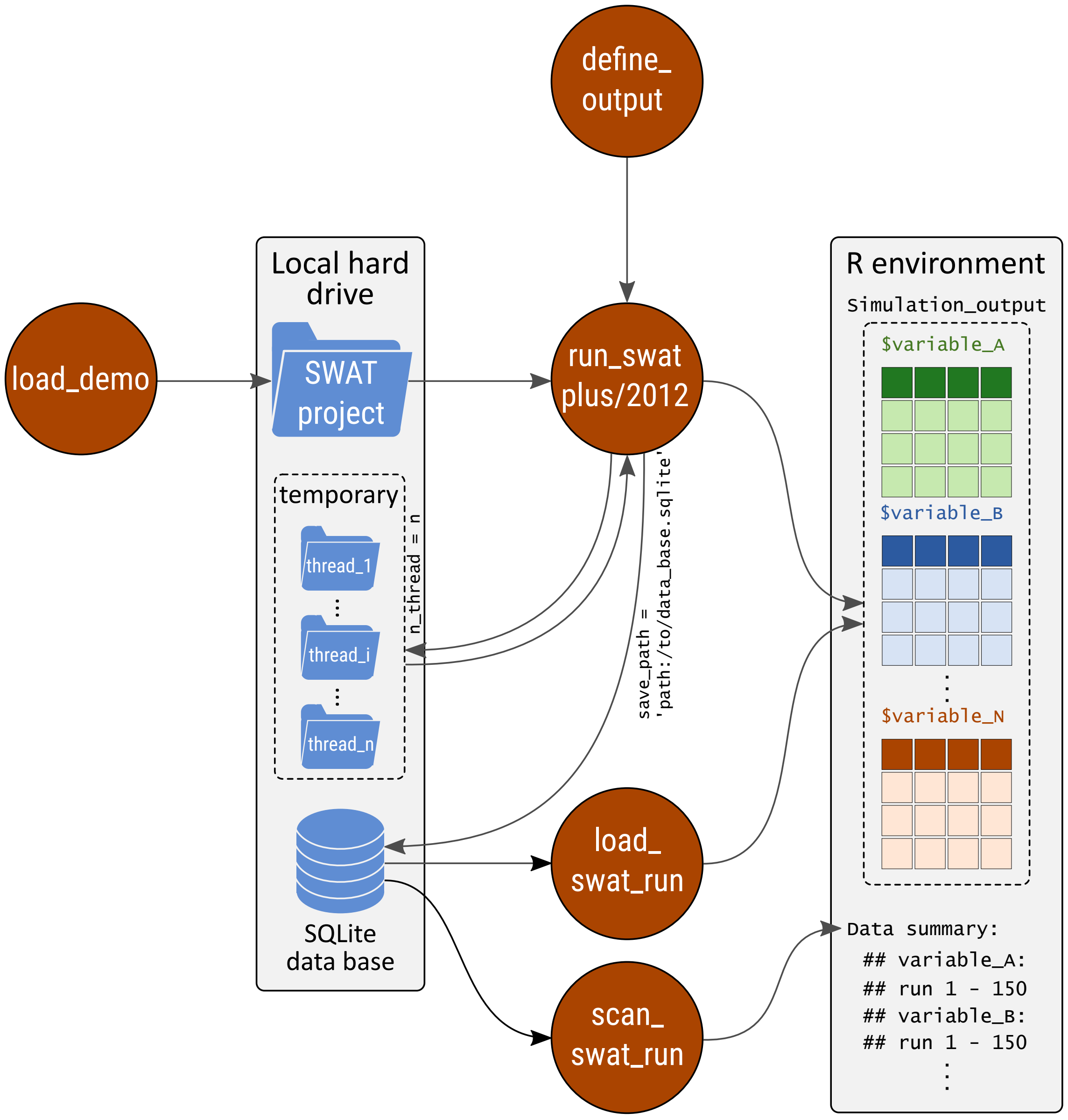

Figure 3.2: Functionality and workflow of the ‘SWATplusR‘ package.

### 3.2.1 The SWAT project and demo data

Every SWAT project is organized in a specific structure of input files in the project folder. The project folder is typically called 'TxtInOut' in most SWAT projects as it is the default name for the Input-Output file folder after setting up a SWAT project with editors such as ArcSWAT or QSWAT. `SWATplusR` requires such a complete set up SWAT project in order to implement it in the workflow in R. As different revisions of the SWAT model might differ in the structure and organization of the model input files, the user always has to provide the corresponding SWAT model executable in the project folder.

To easily test the functionality of `SWATplusR` the R package provides demo data sets with the R package. The functionality of `SWATplusR` is available for the SWAT versions SWAT2012 and SWAT+ and for the operation systems Windows and Linux. Therefore, demo data is available for all combinations of SWAT versions and operation systems. The demo data sets can easily be loaded with the function `load_demo()` (see code box below for usage) and the provision of a path on the local hard drive where the demo data should be stored (see Fig 3.2). The catchment that was selected for demonstration uses is the upper part of the Little River Experimental Watershed (LREW) (Bosch et al., 2007) in Georgia, United States, that has been extensively used in many SWAT modeling studies and that acts as a reference catchment in the SWAT community (Bailey et al., 2017; e.g.; Bieger et al., 2019, 2016; Pfannerstill et al., 2017).

Apart from the demo SWAT projects `SWATplusR` also provides time series data for discharge at the main outlet of the demo catchment to provide the option to test model calibration or parameter sensitivity analysis with the demo data. Shape files of subbasins, HRUs, or the stream network of the set up SWAT projects are provided for any possible visualization tasks in R.

```r
# Function call to load demo data sets

load_demo(type, path = NULL)

# Arguments

# type              Character string that defines the type of demo data
#                   set to be loaded. type = 'SWAT2012' loads a
#                   SWAT2012 project folder. type = 'SWATplus' loads a
#                   SWAT+ project folder. type = 'obs_data' returns the
#                   observation data for the SWAT demo project.
# path              Character string that defines the path where copy
#                   the SWAT demo project.
```

### 3.2.2 Executing SWAT simulations and extracting simulation outputs

The function `run_swat*()` links a SWAT project that is located in the `project_path` on the local hard dive with the R programming environment. As the differences between the SWAT versions SWAT2012 and SWAT+ are drastic (including different input arguments to define and set up a SWAT run), two individual functions to execute a SWAT project with the respective SWAT versions were developed (replace the ‘*’ with ‘2012’ to execute a SWAT2012 project and by ‘plus’ to run a SWAT+ project). The minimum requirement to execute a SWAT model that is located in the `project_path` is to define the `output` of the SWAT simulations that should be returned to R. To define the outputs that are returned from the simulations the user requires the helper function `define_output()`. The function `define_output()` allows the user to define the requested simulation outputs in a clear terminology and translates the output definition into the commands to extract these outputs from the simulation output files in the SWAT project folder. Further this function allows to calculate new variables from several output variables in the output files directly in the output extraction step, when necessary.

The `run_swat*()` functions provide a large number of additional optional parameters to control specific settings of the model execution, the storing of simulation results, or the information that is provided during and after the simulations. Parameters such as the `start_date`, `end_date`, the `output_interval`, or `years_skip` allow control over time period that should be simulated, or the time steps that should be used for writing the simulation outputs (either daily, monthly, or yearly). The parameters `run_index` and `n_thread` define which of the provided parameter sets should be used in the simulation (if multiple parameter sets should be used in the SWAT simulation) and how many parallel threads should be used for the simulation (see Fig 3.2). These two parameters allow an efficient use of computational resources, by using multiple cores on a computer or multiple computers to perform model simulations with multiple model parametrizations. Parameters such as `save_path`, `save_file`, `return_output`, `add_parameter`, or `add_date` provide a lot of control over how and where the simulation results should be saved. A description of the function calls in R and a detailed documentation of all input arguments that can be defined with the function calls are provided in the following code boxes:

```r
# Function call to run a SWAT model

run_swat*(project_path, output, parameter = NULL,
          start_date = NULL, end_date = NULL,
          output_interval = NULL, years_skip = NULL,
          run_index = NULL, run_path = NULL, n_thread = NULL,
          save_path = NULL, save_file = NULL, return_output = TRUE,
          add_parameter = TRUE, add_date = TRUE, refresh = TRUE,
          keep_folder = FALSE, quiet = FALSE)
```

```
# Arguments

# project_path    Path to the SWAT project folder (i.e. TxtInOut)
# output          Define the output variables to extract from the
#                 SWAT model runs. Use define_output() to define an
#                 output.
# parameter       (optional) SWAT model parameters either provided as
#                 named vector or data.frame. If parameter is provi-
#                 ded respective parameters are modified accordingly.
# start_date      (optional) Start date of the SWAT simulation.
#                 Provided as character string in any ymd format (e.g.
#                 'yyyy-mm-dd') or in Date format project are located.
# end_date        (optional) End date of the SWAT simulation. Provi-
#                 ded as character string in any ymd format (e.g.
#                 'yyyy-mm-dd') or in Date format project are located
# output_interval (optional) Time interval in which the SWAT model
#                 outputs are written. Provided either as character
#                 string ("d" for daily, "m" for monthly, or "y" for
#                 yearly) or as # SWAT input values (0 for monthly,
#                 1 for daily, 2 for yearly).
# years_skip      (optional) Integer value that provides the number
#                 of years to be skipped during writing the SWAT
#                 model outputs
# run_index       (optional) Numeric vector (e.g.run_index =
#                 c(1:100, 110, 115)) to run a subset of the provided
#                 parameter sets. If NULL all provided parameter sets
#                 are used.
# run_path        (optional) Character string that provides the path
#                 where the '.model_run' folder is written and the
#                 SWAT models are executed. If NULL '.model_run' is
#                 built in the project folder.
# n_thread        (optional) Number of threads to be used for the
#                 parallel model run. If not provided models are run
#                 on a single core.
# save_path       (optional) Character string to define the path
#                 where the model runs are saved if save_file is de-
#                 fined. If save_path = NULL the save_file is saved
#                 in the 'project_path'.
# save_file       (optional) Character string to define the name of
#                 the file where the simulations are saved.
# return_output   (optional) Logical. Whether outputs should be re-
#                 turned or not. Set return_out = FALSE and provide
#                 save_file if outputs should only be saved on hard
#                 drive. Default = # TRUE
# add_parameter   (optional) Logical. If add_parameter = TRUE used
#                 parameter sets are saved and/or returned together
#                 with the model outputs. Default = TRUE
# add_date        (optional) Logical. If add_date = TRUE a date
#                 column is added to every simulation output table.
#                 Default = TRUE
# refresh         (optional) Logical. refresh = TRUE always forces
#                 that '.model_run' is newly written when SWAT run is
#                 started. Default = TRUE
# keep_folder     (optional) Logical. If keep_folder = TRUE
#                 '.model_run' is kept and not deleted after finishing
#                 model runs. In this case '.model_run' is reused in a
#                 new model run if refresh = FALSE. Default = FALSE
# quiet           (optional) Logical. If quiet = TRUE no messages are
#                 written. Default = FALSE
```

```
# Function call to define a simulation output

define_output(file, variable = NULL, unit = NULL, expression = NULL)

# Arguments

# file            Character string. The SWAT output file where the
#                 output variable is located.
# variable        Character string. Output variable available from
#                 the respective SWAT output file defined with
#                 variable file.
# unit            Numeric vector. The spatial unit file for which the
#                 outputs should be extracted.
# expression      Alternatively to variable and unit an expression
#                 can be defined to extract outputs directly as a
#                 text string.
```

### 3.2.3 Incremental saving, scanning and loading of saved simulations

As defined in the design goals for `SWATplusR` the returned simulations outputs are organized in a ‘tidy’ way. Each simulated variable is organized in an individual `data.frame`, each column in a `data.frame` is one simulation result for the variable, and a row provides the values of all simulations for a time step. The parameters `save_file` and `return_output` from the function `run_swat*()` control how the extracted SWAT simulation outputs are stored. If a `save_file` is defined SQLite data bases are generated in this path on the local hard drive and the simulation results are saved incrementally in the data bases . If the argument `return_output = TRUE` all simulation results are returned back to the R environment after all simulations were performed (see Fig. 3.2). `SWATplusR` provides two functions to access simulation results that were saved locally in data bases. With the function `load_swat_run()` simulation results that are saved in one or many data bases (but are the result from one simulation project) can be loaded into R and further used for any analyses. Large projects require some time to be loaded entirely into the R environment, or are even too large for the RAM storage and cannot be loaded. Therefore, `load_swat_run()` provides the option to define specific variables and specific simulation runs with the respective input arguments `variable` and `run`. In some cases the user does not want to load the simulation runs into R but only wants to inquire which variables and simulation runs are stored in the data base (e.g. when continuing a simulation task). For this tasks `SWATplusR` provides the function `scan_swat_run()` to simply request the meta information of a project data base. A description of the function calls in R and a detailed documentation of all input arguments that can be defined with the function calls are provided in the following code boxes:

```
# Function call to load SWAT runs from a data base

load_swat_run(save_dir, variable = NULL, run = NULL,
  add_parameter = TRUE, add_date = TRUE)

# Arguments

# save_dir         Character string or vector of character strings
#                  that provide the path/s to the save folder/s.
# variable         Output variables that were saved in the SWAT run
#                  and that should be loaded into R.
# run              Numeric vector giving the indexes of the simula-
#                  tions that should be loaded.
# add_parameter    Logical. If add_parameter = TRUE the parameter
#                  set for the SWAT runs is added to the loaded data.
# add_date         Logical. If add_date = TRUE a date column is added
#                  to the simuation results of each variable
```

```
# Function call to scan SWAT runs that are saved in a data base

scan_swat_run(save_dir)

# Arguments

# save_dir         Character string or vector of character strings
#                  that provide the path/s to the save folder/s.
```

# 4 ‘SWATplusR‘ application and course materials

The major strength of the `SWATplusR` R package is the great compatibility with other R packages. Therefore, `SWATplusR` facilitates a very intuitive integration of SWAT projects into any programming and data analysis workflow in R. At the beginning, however, an in particular when a SWAT model user is new to R it can be challenging to perform the intended modeling tasks of a SWAT modeling study in R. To provide guidance and to promote the use of the `SWATplusR` package in the SWAT user community I prepared course material that alleviate the introduction of `SWATplusR` into the users workflows. The study material covers typical tasks that can be part of any SWAT modeling study, such as model calibration, parameter sensitivity analysis, analysis and evaluation of the model results, or the visualization of model results. For every of the listed task I suggest other R packages that can be helpful in approaching the modeling task. Further, I prepare code examples, how the R packages can be implemented together with `SWATplusR`. These code examples are in a way kept rather general and can therefore act as templates that can be easily integrated in other modeling studies and adapted to new requirements. The prepared materials will be provided online and openly accessible. Further, the study materials will be the basis of workshops to demonstrate the use of the `SWATplusR` package in mentioned typical modeling situations, based on small case study examples (and most likely implementing the prepared demo data sets). The sections below provide an excerpt of the prepared study materials.

## 4.1 Package installation

### 4.1.1 SWATplusR

The `SWATplusR` package is hosted in a *github* repository. The R package `devtools` (Wickham et al., 2018a) provides a great set of functions to easily install R packages from other sources that CRAN. To access the functionality of `devtools` the package can be installed from CRAN. After installing `devtools`, `SWATplusR` can be installed using the following commands:

```r
install.packages("devtools")
```

```r
# use the function install_github from the devtools package to install
devtools::install_github("chrisschuerz/SWATplusR")
```

### 4.1.2 Additional packages

Some R packages are useful in any data analysis project, such as most of the R packages from the `tidyverse` (Wickham, 2017). The tidy verse is a suite of several R packages that provide solutions for typical tasks in data analysis, such as `readr` (Wickham et al., 2018b) `dplyr` (Wickham et al., 2019) and `tidyr` (Wickham and Henry, 2018) for data manipulation and data cleaning, `purrr` (Henry and Wickham, 2019) for efficient functional programming, or `ggplot2` for data visualization. The entire `tidyverse` suite can be installed from CRAN as follows.

```r
install.packages("tidyverse")
```

Other R packages are useful for environmental modeling tasks in particular (but not exclusively). A recent article by Slater et al. (2019) gives a comprehensive overview of useful R packages in hydrology. A small selection of R packages that is also mentioned in the article will be briefly addressed here and further used in small examples below.

The `lhs` package (Carnell, 2019) provides different methods to draw Latin Hypercube Samples (LHSs). LHS sampling is a strategy to sample e.g. model parameters, while evenly covering the model parameter space. LHS sampling is often used in hydrological modeling to draw model parameter combinations, for the application of methods of global sensitivity analysis (GSA) such as the method of Sobol (Sobol, 1993, Saltelli et al. (2008)), model calibration e.g. using the SUFI2 algorithm (Abbaspour et al., 2004), or in optimization algorithms such as Shuffled Complex Evolution (SCE; Duan et al., 1993). To use the `lhs` package simply install it from CRAN. A LHS sample can be generated simply by executing the following code:

```r
install.packages("lhs")

# To draw 10 samples for 2 variables:
lhs_sample <- randomLHS(10, 2)
```

The `hydroGOF` package (Mauricio Zambrano-Bigiarini, 2017) provides a comprehensive library of objective functions to evaluate simulated time series based on observation data, that are frequently used in hydrology, such as the Nash Sutcliffe efficiency (NSE; Nash and Sutcliffe, 1970), the Kling Gupta efficiency (KGE; Gupta et al., 2009), or the percentage bias (Gupta et al., 1999). The package can be installed from CRAN.

```r
install.packages("hydroGOF")
```

The `sensitivity` (Iooss et al., 2018) package provides a large variety of methods to perform Global Sensitivity Analysis (GSA). Standard methods for sensitivity analysis that are available with the R package are the method of Sobol (Sobol, 1993), the method of Morris (Morris, 1991), or Delsa (Rakovec et al., 2014). The package can be installed from CRAN.

```
install.packages("sensitivity")
```

The `fast` package (Reusser, 2015) provides a collection of methods to perform the Fourier Amplitude Sensitivity Test (FAST) as a method for GSA. This package implements this method in `R`. The package can be installed from CRAN.

```
install.packages("fast")
```

The `hydromad` package is a comprehensive suite for hydrological simulation, model evaluation, model optimization, and visualization. It is due to its strict structural requirements very limited to the models provided by the package. Some functions, however, such as `SCEoptim()` that implements the SCE algorithm can be easily implemented with any model function. The `hydromad` package is available from hydromad.catchment.org and can be easily installed as follows:

```
install.packages(c("zoo", "latticeExtra", "polynom",
                   "car", "Hmisc","reshape"))

install.packages("hydromad",
  repos="http://hydromad.catchment.org")
```

## 4.2 Loading R packages

R packages have to be loaded at the beginning of each R session in order to access the functionality provided by a package. The command to load an R package is `library()`. Good practice for any study done in R is to load all required packages at the top of an R script:

```
library(SWATplusR)
library(tidyverse)
library(lubridate)
library(forcats)
library(lhs)
library(fast)
library(sensitivity)
library(hydroGOF)
library(hydromad)
```

## 4.3 Exploring the basic `SWATplusR` functionality

### 4.3.1 Loading a demo project

The `SWATplusR` package provides very simple model setups of a head watershed of the Little River Experimental Watershed (LREW). Demo model setups can be retrieved for SWAT2012 and for SWAT+. Currently the SWAT2012 demo is available for Windows and Linux, while the SWAT+ demo is only available for Windows. In updated versions of `SWATplusR`, however, demo projects will be available for all SWAT versions and operating system platforms. A demo project can be loaded with the following command.

```r
# The path where the SWAT demo project will be written
demo_path <- "C:"

# Defining the SWAT version
swat_version <- "plus" #or "2012" on Linux

# The function writes the demo folder to the defined path and
# returns the final path of the project folder in R
proj_path <- load_demo(dataset = "project",
                       swat_version = swat_version,
                       path = demo_path)
```

### 4.3.2 Loading observation data for the LREW watershed

Discharge data for the main outlet of the demo watershed is available with `SWATplusR`. The discharge time series is provided on a daily time step for the time period 1968-01-01 until 2012-12-31. The data set can be used for model evaluation of a demo project. Running the following command loads a tibble with the timeseries of the discharge data in the R working environment.

```r
q_obs <- load_demo(dataset = "observation")

q_obs
```

```
# A tibble: 16,437 x 2
   date       q_out
   <date>     <dbl>
 1 1968-01-01 0.16
 2 1968-01-02 0.570
 3 1968-01-03 0.61
 4 1968-01-04 0.37
 5 1968-01-05 0.25
 6 1968-01-06 0.2
 7 1968-01-07 0.21
 8 1968-01-08 0.22
 9 1968-01-09 0.18
10 1968-01-10 0.34
# ... with 16,427 more rows
```

### 4.3.3 A first SWAT simulation

After loading the demo project it is already possible to perform a first SWAT simulation. The SWAT demo project can be executed from R with the functions `run_swat2012()` to run a SWAT2012 project or `run_swatplus()` to run a SWAT+ project. The minimum requirements to execute a SWAT project are to provide the path to the project folder on the hard drive and to define the simulation outputs that the simulation should return to R. The definition of the simulation output always follows a simple pattern. Each simulated variable (e.g. the discharge, the actual evapotranspiration, the nitrate-nitrogen loads, etc.) has to be defined using the function `define_output()`. The arguments that have to be dfeined are `file` that tells the function into which output file the variable was written, `variable` that is the variable header as it is written in the output file, or the number of the column of a variable in the respective output table, and `unit` that defines one or several spatial units for which the outputs should be extracted. Below is an example to return the discharge ('flow_out') that is written into the 'channel' file at the main outlet (that is the subbasin number 1) of the SWAT+ demo project.

```
q_out <- run_swatplus(project_path = proj_path,
                      output = define_output(file = "channel",
                                             variable = "flo_out",
                                             unit = 1))
```

The function returns the simulation at the catchment outlet as a table providing the simulation dates and the simulated discharge for these time steps.

```
q_out
```

```
# A tibble: 3,653 x 2
  date       flo_out
  <date>       <dbl>
1 2003-01-01    7.17
2 2003-01-02    6.12
3 2003-01-03    4.15
4 2003-01-04    3.15
5 2003-01-05    2.69
6 2003-01-06    2.43
7 2003-01-07    2.28
8 2003-01-08    2.18
```

```
 9 2003-01-09    2.11
10 2003-01-10    2.04
# ... with 3,643 more rows
```

More than one variable that should be returned to R after the simulation have to be defined in a list. When the variables in the list are named, these variable names are then assigned to the variables in the returned output tables, as shown in this example for different water balance components.

```
wb_out <- run_swatplus(project_path = proj_path,
           output = list(precip = define_output(file = "basin_wb",
                                                variable = "precip",
                                                unit = 1),
                         q_sur  = define_output(file = "basin_wb",
                                                variable = "surq_gen",
                                                unit = 1),
                         q_lat  = define_output(file = "basin_wb",
                                                variable = "latq",
                                                unit = 1),
                         eta    = define_output(file = "basin_wb",
                                                variable = "et",
                                                unit = 1)))

wb_out
```

```
wb_out
```

```
# A tibble: 3,653 x 5
   date       precip q_sur q_lat   eta
   <date>      <dbl> <dbl> <dbl> <dbl>
 1 2003-01-01  4.28  0.144 2.28  0.425
 2 2003-01-02  0.387 0.021 1.68  0.951
 3 2003-01-03  0     0.005 1.34  0.081
 4 2003-01-04  0     0.001 1.15  0.742
 5 2003-01-05  0     0     1.03  0.7
 6 2003-01-06  0     0     0.962 0.676
 7 2003-01-07  0     0     0.912 0.574
 8 2003-01-08  0     0     0.874 0.628
 9 2003-01-09  0     0     0.844 0.716
10 2003-01-10  0     0     0.817 0.423
# ... with 3,643 more rows
```

### 4.3.4 Modifying parameters in a simulation

Parameters that should be modified in a SWAT simulation can be provided to the function `run_swat*()` with the argument `parameter`. A single parameter set can be provided as a vector. Many parameter sets that should be used in simulations have to be provided with a `tibble`.

The parameter names, that have to be provided with the parameter set, are very essential as they have to follow a certain syntax that controls the changes that are made for a parameter, such as the type of change, or selecting specific HRUs, subbasins, or land uses for which the parameter should be modified, or which should be excluded from the parameter change. A parameter name can consist of several parts, where some are required and others are optional. The minimum requirement for a parameter name is the actual name of the parameter in the model an the type of change as shown in the example below. The syntax tells the model that the parameter 'CN2' that is defined on HRU level should be changed and that the parameter should be changed by adding an absolute value to the initial parameter value.

```r
par_name <- "cn2.hru|change = abschg"
```

For later analyses it can be a good strategy to assign clear names to the parameters, especially if the modification of a parameter was only done for selected subbasins, or land uses. Individual names can be assigned to a parameter as follows.

```r
par_name <- "my_name::cn2.hru|change = abschg"
```

To define now a single parameter set for a SWAT simulation a named vector has to be generated where the name defines the parameter to be changed and controls the type of change and the value provides the change to be made in the SWAT model setup. Below a single parameter set was generated where all 'CN2' values in the model are reduced by 5 percent and set the 'alpha.gw' value was set to 0.5 globally in the model setup.

```r
par_single <- c("cn2.hru|change = pctchg" = - 5,
                "alpha.gw|change = absval" = 0.5)
```

The SWAT model can be executed with the new parameter set by simply assigning the vector with the parameter values to the input argument `parameter`.

```r
q_out <- run_swatplus(project_path = proj_path,
                      output = define_output(file = "channel",
                                             variable = "flo_out",
                                             unit = 1),
                      parameter = par_single)
```

The same workflow works with several parameter sets when they are defined in a `tibble`. The example below uses the same parameters as above, but in this case a `tibble` was defined with 8 random realizations for each parameter within the defined boundaries.

```
par_set <- tibble("cn2.hru|change = abschg" = runif(8,-15,10),
                  "alpha.gw|change = absval" = runif(8, 0, 1))
```

The implementation in the `run_swat*()` function works the same way. To execute multiple SWAT simulations with many parameter sets the `SWATplusR` package provides the option to run the simulations in parallel. To perform a parallel model execution the number of parallel threads has to be defined with the input argument `n_thread`.

```
q_out <- run_swatplus(project_path = proj_path,
  output = list(q_sur  = define_output(file = "basin_wb",
                                       variable = "surq_gen",
                                       unit = 1),
                q_lat  = define_output(file = "basin_wb",
                                       variable = "latq",
                                       unit = 1)),
                      parameter = par_set,
                      n_thread = 4)
```

Now instead of only providing a table with the simulations, the function returns by default a list that stores the parameter set that was used, a table that shows details of the parameter definition and the simulation results, where again the each variable is stored in one table. As more than one simulation was performed, the columns for the simulation results are now named run_1 to run_8.

```
q_out
```

```
$parameter
$parameter$values
# A tibble: 8 x 2
      cn2  alpha
    <dbl>  <dbl>
1  -1.59  0.819
2   9.06  0.364
3   0.552 0.0511
4  -8.80  0.884
5  -4.50  0.388
6  -2.95  0.969
7 -12.0   0.492
8   5.49  0.701

$parameter$definition
# A tibble: 2 x 4
  par_name parameter file_name change
  <chr>    <chr>     <chr>     <chr>
```

```
1 cn2      cn2       hru       abschg
2 alpha    alpha     gw        absval


$simulation
$simulation$q_sur
# A tibble: 3,653 x 9
   date       run_1 run_2 run_3 run_4 run_5 run_6 run_7 run_8
   <date>     <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 2003-01-01 0.112 0.366 0.152 0.035 0.071 0.091 0.02  0.244
 2 2003-01-02 0.017 0.058 0.022 0.005 0.011 0.014 0.002 0.039
 3 2003-01-03 0.004 0.014 0.005 0.001 0.002 0.003 0.001 0.009
 4 2003-01-04 0.001 0.003 0.001 0     0.001 0.001 0     0.002
 5 2003-01-05 0     0.001 0     0     0     0     0     0.001
 6 2003-01-06 0     0     0     0     0     0     0     0
 7 2003-01-07 0     0     0     0     0     0     0     0
 8 2003-01-08 0     0     0     0     0     0     0     0
 9 2003-01-09 0     0     0     0     0     0     0     0
10 2003-01-10 0     0     0     0     0     0     0     0
# ... with 3,643 more rows

$simulation$q_lat
# A tibble: 3,653 x 9
   date       run_1 run_2 run_3 run_4 run_5 run_6 run_7 run_8
   <date>     <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 2003-01-01 2.33  1.74  2.27  2.49  2.41  2.37  2.52  2.01
 2 2003-01-02 1.72  1.27  1.67  1.87  1.80  1.75  1.90  1.47
 3 2003-01-03 1.38  1.01  1.33  1.52  1.44  1.41  1.54  1.18
 4 2003-01-04 1.18  0.86  1.14  1.30  1.24  1.21  1.33  1.01
 5 2003-01-05 1.06  0.773 1.02  1.17  1.12  1.09  1.20  0.905
 6 2003-01-06 0.989 0.717 0.953 1.09  1.03  1.01  1.11  0.841
 7 2003-01-07 0.937 0.679 0.903 1.03  0.98  0.956 1.05  0.797
 8 2003-01-08 0.898 0.651 0.866 0.98  0.938 0.917 0.999 0.765
 9 2003-01-09 0.867 0.627 0.835 0.943 0.905 0.884 0.961 0.738
10 2003-01-10 0.839 0.607 0.809 0.912 0.876 0.856 0.928 0.715
# ... with 3,643 more rows
```

## 4.4 Typical modeling tasks with SWATplusR

### 4.4.1 Parameter sensitivity analysis

In two examples the `SWATplusR` package will be integrated into parameter sensitivity analysis applying the two standard methods for global sensitivity analysis (GSA), the method of Sobol that is available in the R package `sensitivity` and FAST that is provided with the R package `fast`. The Fourier Amplitude Sensitivity Test (FAST) is a method to perform GSA with few model evaluations. It only requires a few simulations when the number of parameters is low and strongly increases to tenth of thousands of model evaluations for more than 20 parameters. Therefore for our example we select only 7 parameters that are relevant in many model SWAT model applications.

The FAST method requires a specific parameter sampling design. Fortunately, the `fast` package provides a function to sample the model parameters.

```
# The parameters used with their boundaries
par_names <- c("cn2.hru | change = abschg",
               "lat_ttime.hru | change = absval",
               "lat_len.hru | change = absval",
               "k.sol | change = pctchg",
               "z.sol | change = pctchg",
               "esco.hru | change = absval",
               "epco.hru | change = absval")


par_fast <- fast_parameters(
  minimum = c(-15, 0.5,  10, -50, -50, 0,  0),
  maximum = c( 10,  50, 100,  50,  50, 1,  1),
    names = par_names)
```

To perform a FAST analysis for 7 parameters 167 model evaluations are required. The SWAT model is executed with the parameter sets sampled with `fast` and is evaluated using the NSE criterion for daily discharge for the time period 2003 to 2012.

```
q_fast <- run_swatplus(project_path = proj_path,
            output = list(q_out = define_output(file = "channel",
                                      variable = "flo_out",
                                      unit = 3)),
            parameter = par_fast,
            start_date = "2000-01-01",
            end_date = "2012-12-31",
            years_skip = 3,
            n_thread = 4)
```

The NSE function is available from the `hydroGOF` package. To evaluate the simulations with the same time period of the observations, the observation time series was also reduce to 2003 to 2012.

```
q_obs <- filter(q_obs, date >= ymd("2003-01-01"),
                       date <= "2012-12-31")
```

```
nse_fast <- q_fast$simulation$q_out %>%
  select(-date) %>%
  map_dbl(., ~NSE(.x, q_obs$q_out))

sens_fast <- sensitivity(nse_fast, 7)
```

To visualize the calculated sensitivities the `ggplot2` package was used and a bar plot was generated where the sensitivity values of the 7 parameters were sorted and plotted, to rank the parameters.

```
result_fast <- tibble(parameter = q_fast$parameter$definition$par_name,
                      fast      = sens_fast) %>%
  mutate(parameter = factor(parameter) %>% fct_reorder(., fast))
```

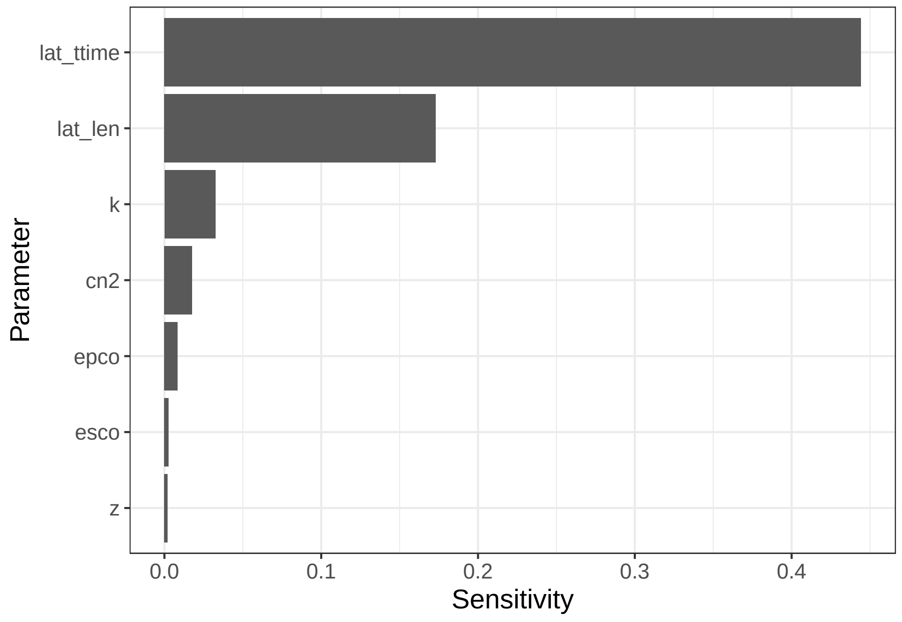

Figure 4.1: Results of the sensitivity analysis using the FAST method.

The method of Sobol is a reference method for GSA. To implement the SWAT model in the GSA workflow with `sensitivity` a function has to be defined that returns the a scalar variable for which the sensitivity is assessed.

```
swat_sobol <- function(par) {
  names(par) <- par_names
  q_sim <- run_swatplus(project_path = proj_path,
                        output = define_output(file = "channel",
                                               variable = "flo_out",
                                               unit = 1),
                        parameter = par,
                        start_date = "2000-01-01",
                        end_date = "2012-12-31",
                        years_skip = 3, n_thread = 4,
                        add_date = FALSE)
  nse_q <- map_dbl(q_sim$simulation$flo_out/8.64,
                   ~ NSE(.x, q_obs$q_out))
  return(nse_q)
}
```

To perform GSA with the method of Sobol two random sets of samples with the same sample size for the parameters that should be analyzed are required.

```
par_bound <- tibble("cn2.hru | change = abschg" = c(-15, 10),
                    "lat_ttime.hru | change = absval" = c(0.5, 50),
                    "lat_len.hru | change = absval" = c(10, 100),
                    "k.sol | change = pctchg" = c(-50, 50),
                    "z.sol | change = pctchg" = c(-50, 50),
                    "esco.hru | change = absval" = c(0, 1),
                    "epco.hru | change = absval" = c(0, 1))

n_par  <- 7
n_samp <- 500

x1 <- data.frame(matrix(runif(n_par * n_samp), nrow = n_samp)) %>%
  set_names(., names(par_bound)) %>%
  map2_dfc(., par_bound, ~ (.x * (.y[2] - .y[1]) + .y[1]))

x2 <- data.frame(matrix(runif(n_par * n_samp), nrow = n_samp)) %>%
  set_names(., names(par_bound)) %>%
  map2_dfc(., par_bound, ~ (.x * (.y[2] - .y[1]) + .y[1]))
```

To perform the sensitivity analysis with using method of Sobol the following command has to be executed. In total 4000 model evaluations are necessary to analyze 7 parameters with 500 Sobol samples.

```
sens_sobol <- sobol(model = swat_nse, X1 = x1, X2 = x2, nboot = 100)
```

Similar to the visualization of the results with the FAST method the results of the GSA with the method of Solbol are plotted.

```
plot_sobol <- sens_sobol$S %>%
  mutate(parameter = rownames(.)) %>%
  mutate(parameter = factor(parameter) %>% fct_reorder(., original))
```

```
ggplot(data = plot_sobol) +
  geom_pointrange(aes(x = parameter, y = original ,
                      ymin = `min. c.i.`, ymax = `max. c.i.`)) +
  coord_flip() +
  xlab("Parameter") +
  ylab("Sensitivity") +
  theme_bw() +
  theme(text = element_text(size=14))
```

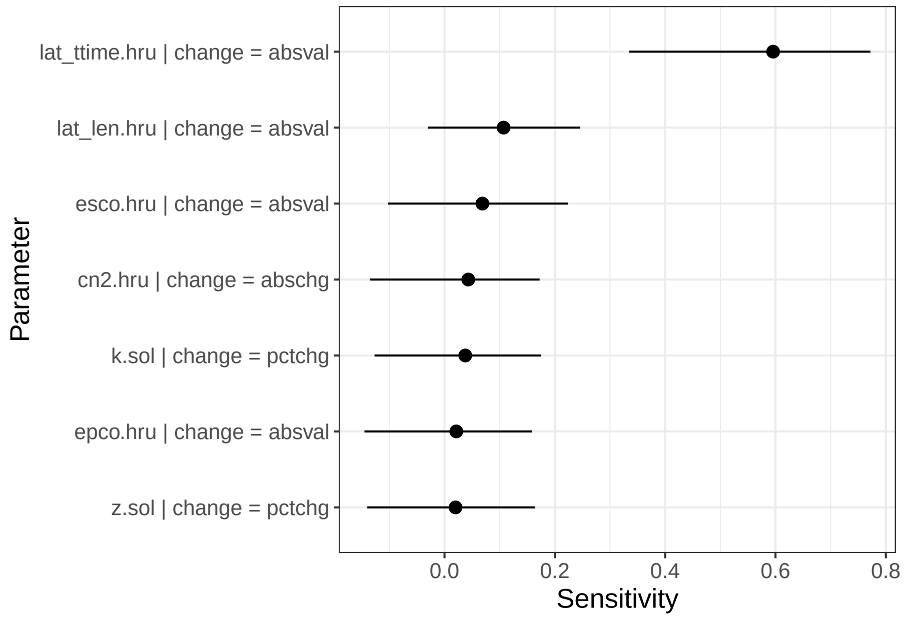

Figure 4.2: Results of the sensitivity analysis using the method of Sobol.

### 4.4.2 Parameter optimization

A way to find a model parametrization that results in model setup that adequately reproduces observation data is to perform parameter optimization. Several routines such as the Shuffled Complex Evolution algorithm (SCE; Duan et al., 1993), or the Dynamically Dimensioned Search algorithm (DDS; Tolson and Shoemaker, 2007) among others are frequently used in hydrological modeling. The two examples below demonstrate the implementation of a 'quasi-Newton' method (Byrd et al., 1995) with the generic `optim()` function and the implementation of the SCE algorithm that is available in the `hydromad` package.

Both optimization algorithms again require a function that returns a scalar for the optimization. Thus, the SWAT model has to be implemented in a function that performs the model evaluation with observation data and returns a measure of goodness-of-fit for which the model parameters will be optimized.

```
swat_optim <- function(par) {
  names(par) <- par_names
  q_sim <- run_swatplus(project_path = "C:/swat_test/swatplus_demo",
                        output = define_output(file = "channel",
                                               variable = "flo_out",
                                               unit = 1),
                        parameter = par,
                        start_date = "2000-01-01",
                        end_date = "2012-12-31",
                        years_skip = 3,
                        quiet = TRUE,
                        keep_folder = TRUE,
                        refresh = FALSE)

  nse_q <- - NSE(q_sim$simulation$flo_out/8.64, q_obs$q_out)

  return(nse_q)
  }
```

The optimization algorithms require the parameters to be optimized, starting values, and upper and lower parameter boundaries to define the range in which the algorithm searches the best parameter set.

```
par_names <- c("cn2.hru | change = abschg",
               "lat_ttime.hru | change = absval",
               "lat_len.hru | change = absval",
               "k.sol | change = pctchg",
               "z.sol | change = pctchg",
               "epco.hru | change = absval",
               "esco.hru | change = absval")

par_init <- c(0, 5, 50, 0 , 0, 0.5, 0.5)
par_lwr  <- c(-15, 0.5,  10, -50, -50, 0, 0)
par_upr  <- c( 10,  50, 100,  50,  50, 1, 1)

names(par) <- par_names
names(par_lwr) <- par_names
names(par_upr) <- par_names
```

To perform the parameter optimization the defined model that should be optimized and the parameters with their initial values and boundaries are implemented as follows.

```
opt_bfgs <- optim(par, swat_opt, method = "L-BFGS-B",
                  lower = par_lwr, upper = par_upr,
                    control = list(reltol = 10^(-3)))

opt_sce  <- SCEoptim(swat_opt, par, lower = par_lwr, upper = par_upr,
                     control = list(reltol = 10^(-3)))
```

### 4.4.3 Parameter sampling and model calibration

A common procedure in hydrological modeling is to draw random samples for a set of parameters, to execute the model with all drawn parameter combinations, evaluate the simulations based on one or several criteria, and select parameter sets that were able to reproduce the observation date sufficiently with the applied model setup. A workflow to perform all these steps is illustrated below. Random samples are often drawn using Latin Hypercube Sampling (LHS) as the LHS sampling strategy evenly distributes the drawn samples over the defined parameter space. In R LHS samples can be drawn with the R package `lhs`.

```
par_bound <- tibble("cn2.hru | change = abschg" = c(-15, 10),
                    "lat_ttime.hru | change = absval" = c(0.5, 50),
                    "lat_len.hru | change = absval" = c(10, 100),
                    "k.sol | change = pctchg" = c(-50, 50),
                    "z.sol | change = pctchg" = c(-50, 50),
                    "esco.hru | change = absval" = c(0, 1),
                    "epco.hru | change = absval" = c(0, 1))

n_par  <- 7
n_samp <- 250

par_lhs <- randomLHS(n = n_samp, k = n_par) %>%
  as_tibble(.) %>%
  map2_dfc(., par_bound, ~ (.x * (.y[2] - .y[1]) + .y[1]))
```

All sampled parameter combinations are implemented in the SWAT+ demo setup and daily discharge is simulated for the period 2003-01-01 until 2012-12-31 (with a warm-up period of 3 years before that). Again the entire table that holds all parameter combinations is provided with the argument `parameter`. In this small example we simply define that the period from 2003-01-01 is a calibration period of 5 years. The following 5 years from 2008 - 2012 are used to validate the simulations that were identified as behavioral in the calibration.

```
q_lhs <- run_swatplus(project_path = proj_path,
           output = list(q_out = define_output(file = "channel",
                                       variable = "flo_out",
                                       unit = 3)),
          parameter = par_lhs,
          start_date = "2000-01-01",
           end_date = "2012-12-31",
          years_skip = 3,
          n_thread = 4)
```

The 250 simulations are evaluated with the NSE criterion and observations of daily discharge at the catchment outlet. To split the data into calibration and validation periods the time series have to be cut by the defined dates. Important to know here is that SWAT+ writes the discharge in $ha \cdot m$. To convert the discharge volume for one day to a mean discharge for that day in $m^3 s^{-1}$ it is necessary to divide by 8.64.

```
q_cal <- q_lhs$simulation$q_out %>%
  filter(date < ymd("2008-01-01"))

q_obs_cal <- q_obs %>%
  filter(date < ymd("2008-01-01"))

nse_cal <- q_cal %>%
  select(-date) %>%
  map_dbl(., ~NSE(.x/8.64, q_obs_cal$q_out))
```

Below the best NSE results from the 250 simulations are given for the calibration period. With a maximum NSE of 0.71 the LHS sample of the selected parameters resulted already in acceptable simulations.

```
head(sort(nse_cal, decreasing = TRUE))
```

```
  run_219   run_198   run_244   run_168   run_016   run_012
0.7110944 0.7006471 0.6990813 0.6966276 0.6941954 0.6890791
```

To validate the selected behavioral simulations the procedure is repeated, but only using the runs that were selected in the calibration step.

```
index_sel <- sort(nse_cal, decreasing = TRUE, index.return = TRUE)
run_sel   <- names(index_sel$x[index_sel$x > 0.5])
index_sel <- index_sel$ix[index_sel$x > 0.5]

q_sel <- q_lhs$simulation$q_out %>%
  select(date, one_of(run_sel))

q_val <- q_sel %>%
  filter(date >= ymd("2008-01-01"))

q_obs_val <- q_obs %>%
  filter(date >= ymd("2008-01-01"))

nse_val<- q_val %>%
  select(-date) %>%
  map_dbl(., ~NSE(.x/8.64, q_obs_val$q_out))
```

The majority of the selceted runs performed well in the validation, with NSE values above 0.5 for almost all runs.

```
head(nse_val)
```

```
  run_219   run_198   run_244   run_168   run_016   run_012 
0.6313062 0.6265993 0.6749064 0.6945464 0.6514865 0.6619073 
```

### 4.4.4 Visualization

To explore the simulations it is strongly recommended to include visualizations in the analysis. Common visualizations that are used in analyses of hydrological simulations are dotty plots and to plot the time series of all model parametrizations that are considered as behavioral. Dotty plots are useful to identify parameter ranges that either improve or decrease the considered criterion that was used to evaluate the simulations. This is an indicator for the parameter sensitivities and can further be used to constrain the parameter ranges in a next iteration step in the model calibration. The plots of simulated time series and their comparison to observation data can provide a lot of insight in the strengths and weaknesses of reproducing observations with a model setup. Below code examples are provided to produce such plots for the small calibration example.

```
dotty_data <- q_lhs$parameter$values %>%
  mutate(nse = nse_cal) %>%
  gather(key = "parameter", value = "value", -nse)

ggplot(data = dotty_data) +
  geom_point(aes(x = value, y = nse)) +
  theme_bw() +
  ylim(c(-5, 1)) +
  ylab("NSE") +
  xlab("Parameter") +
  facet_wrap(parameter ~ ., scales = "free_x" ) +
  theme(text = element_text(size=14))
```

The dotty plot clearly shows a strong impact of the parameter ‘lat_ttime’, that was already shown in the different sensitivity analyses. The other parameters show a rather flat upper boundary in the dotty plots. Thus, these parameters were able (in

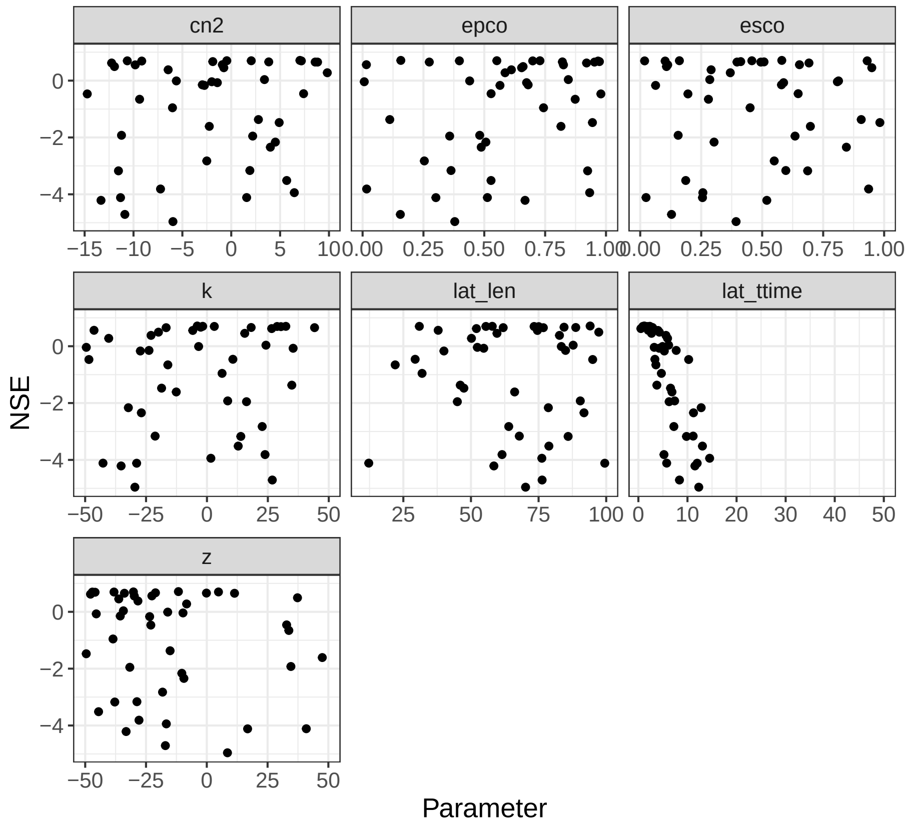

Figure 4.3: Dotty plot for the 250 LHS sampled model parameter sets based on the NSE for daily discharge.

combination with other parameters) to result in good simulations over their entire range.

```
plot_cal <- q_cal %>%
  select(one_of(run_sel)) %>%
  mutate(q_max = pmap_dbl(., max),
         q_min = pmap_dbl(., min)) %>%
  select(matches(run_sel[1]), q_min, q_max) %>%
  mutate_all(funs(./8.64)) %>%
  bind_cols(set_names(q_obs_cal, "date", "q_obs"), .) %>%
  gather(key = "variable", value = "discharge", -date, -q_min, - q_max) %>%
  mutate(period = "calibration")

plot_val <- q_val %>%
  select(-date) %>%
  mutate(q_max = pmap_dbl(., max),
         q_min = pmap_dbl(., min)) %>%
  select(matches(run_sel[1]), q_min, q_max) %>%
  mutate_all(funs(./8.64)) %>%
  bind_cols(set_names(q_obs_val, "date", "q_obs"), .) %>%
  gather(key = "variable", value = "discharge", -date, -q_min, - q_max) %>%
  mutate(period = "validation")

plot_data <- bind_rows(plot_cal, plot_val)

ggplot(data = plot_data) +
  geom_ribbon(aes(x = date, ymin = q_min, ymax = q_max, fill = period)) +
  geom_line(aes(x = date, y = discharge, col = variable, linetype = period), lwd = 0.
  scale_color_manual(values = c("dodgerblue3", "tomato3")) +
  scale_fill_manual(values = c("grey30", "grey70")) +
  xlab("Date (yyyy)") +
  ylab(expression (Discharge~(m^3~s^{-1}))) +
  theme_bw()
```

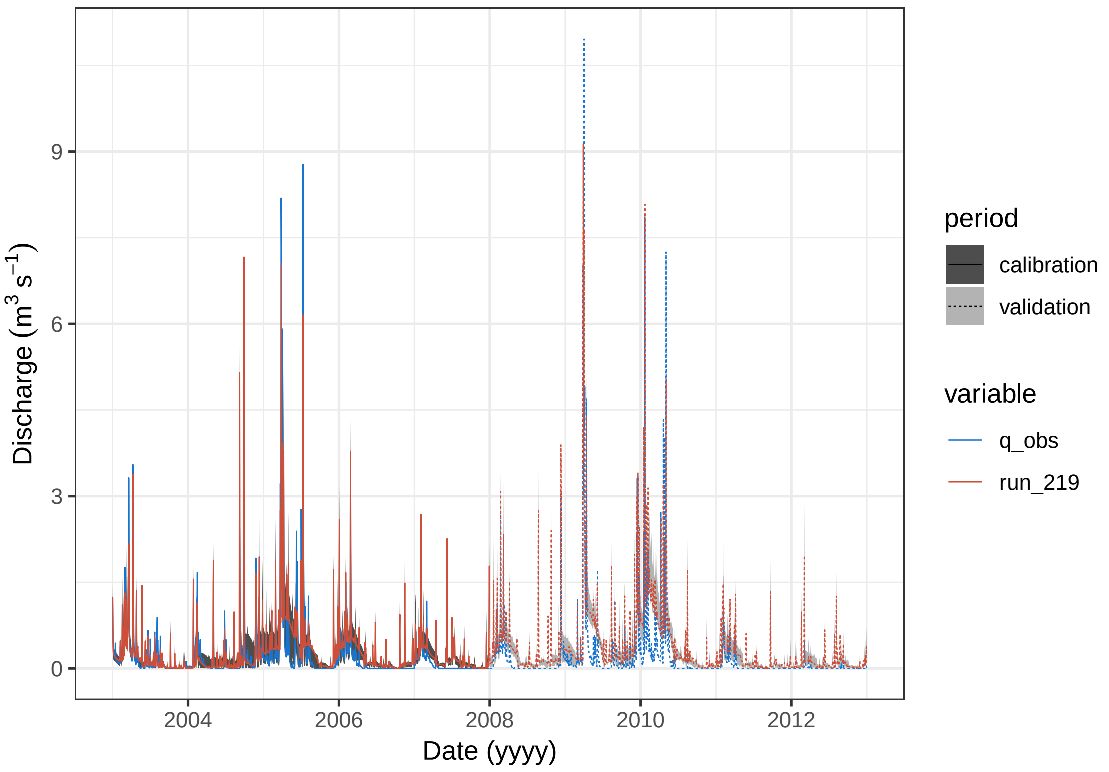

Figure 4.4: Best simulation and ranges for behavioral simulations for discharge atthe main outlet for the calibration and validation periods.

# 5 Outlook

The previous sections outline the routinely challenges in environmental modeling studies and showed with `SWATplusR` a flexible tool to implement solutions into individual programming workflows to approach SWAT modeling tasks in R. The R package provides already the essential functionality in its current form. Following steps of the package development will focus on the improvement of the demo materials that are implemented in the R package and the documentation of the R package. To establish the developed tool in the R community and to find great acceptance with the developed programming workflows the study materials will be improved, workshops will be organized and the publication of the R package in a peer reviewed journal.

The demo materials that are implemented in `SWATplusR` facilitate to test large parts of the package functionality already. An implementation of SWAT+ on Linux platforms is, however, still missing, as no stable Linux executable of SWAT+ yet exists. The development of Linux executables of recent revisions of SWAT+ is currently under development and will be implemented soon. Based on feedback from future SWAT-R workshops and feedback from colleagues that test the R package, further modifications and additions to the R package and the demo materials will be done. The documentation of the functionality of `SWATplusR` is in development and will be available online soon at `https://chrisschuerz.github.io/SWATplusR`.

The study materials will be continuously improved based on feedback from the workshops and feedback that I get from colleagues. Currently study materials are available in a form as outlined in section 4 above. The materials from the first SWAT-R workshop that took place at the Texas A&M Research and Extension Center, Temple, TX in December 2018 (see Fig. 5.1) are freely available online from the course's GitHub repository. In a final step the course materials will be integrated with the R package documentation online. Short tutorials will be developed and structured in chapters that cover collections of programming solutions for typical challenges in environmental modeling (similar to the materials outlined in section 4, but with greater detail). A first test workshop held for the SWAT developers group at the Texas A&M Blackland Research and Extension Center Temple, TX on Dec. 4 2018 (Fig. 5.1) brought valuable feedback for the R package and the course materials. This first workshop forms the basis for further future workshop formats to teach R as a tool for environmental modeling and to introduce SWAT modelers in the functionality of the `SWATplusR` package. This year two further workshops are planned, one for the German SWAT user group at the University of Rostock, Germany in March 2019 and a second at the International SWAT Conference at the University of Natural Resources

Figure 5.1: The first test workshop held for the SWAT developer group at the Texas A&M Blackland Research and Extension Center Temple, TX on Dec 4 2018

and Life Sciences, Vienna, Austria in July 2019 (further information is provided at the conference web page).

# References

Abbaspour, K. C.: SWAT-CUP: SWAT calibration and uncertainty programs, EAWAG, Dübendorf, Switzerland. [online] Available from: `https://swat.tamu.edu/software/swat-cup/` (Accessed 18 September 2018), 2015.

Abbaspour, K. C., Johnson, C. A. and Genuchten, M. T. van: Estimating uncertain flow and transport parameters using a sequential uncertainty fitting procedure, Vadose Zone Journal, 3(4), 1340, doi:10.2136/vzj2004.1340, 2004.

Arnold, J. G., Srinivasan, R., Muttiah, R. S. and Williams, J. R.: Large area hydrologic modeling and assessment part I: model development, Journal of the American Water Resources Association, 34(1), 73–89, doi:10.1111/j.1752-1688.1998.tb05961.x, 1998.

Bailey, R., Rathjens, H., Bieger, K., Chaubey, I. and Arnold, J.: SWATMOD-prep: Graphical user interface for preparing coupled SWAT-MODFLOW simulations, JAWRA Journal of the American Water Resources Association, 53(2), 400–410, doi:10.1111/1752-1688.12502, 2017.

Beven, K.: The limits of splitting: Hydrology, Science of the Total Environment, 183(1-2), 89–97, doi:10.1016/0048-9697(95)04964-9, 1996.

Beven, K.: A manifesto for the equifinality thesis, Journal of Hydrology, 320(1-2), 18–36, doi:10.1016/j.jhydrol.2005.07.007, 2006.

Beven, K. and Freer, J.: Equifinality, data assimilation, and uncertainty estimation in mechanistic modelling of complex environmental systems using the glue methodology, Journal of Hydrology, 249(1), 11–29, doi:https://doi.org/10.1016/S0022-1694(01)00421-8, 2001.

Bezanson, J., Karpinski, S., Shah, V. B. and Edelman, A.: Julia: A Fast Dynamic Language for Technical Computing, arXiv [online] Available from: `http://arxiv.org/abs/http://arxiv.org/abs/1209.5145v1`, 2012.

Bicknell, J. C. K., B. R.; Imhoff: Hydrological simulation program fortran: User's manual for version 11., EPA-68th-01st-6207th ed., Environmental Protection Agency Office of Research; Development. [online] Available from: `https://ntrl.ntis.gov/NTRL/dashboard/searchResults/titleDetail/PB97193114.xhtml` (Accessed 8 September 2014), 1997.

Bieger, K., Hörmann, G. and Fohrer, N.: The impact of land use change in the

Xiangxi Catchment (China) on water balance and sediment transport, Regional Environmental Change, 15(3), 485–498, doi:10.1007/s10113-013-0429-3, 2013.

Bieger, K., Arnold, J. G., Rathjens, H., White, M. J., Bosch, D. D., Allen, P. M., Volk, M. and Srinivasan, R.: Introduction to SWAT+, a completely restructured version of the soil and water assessment tool, JAWRA Journal of the American Water Resources Association, 53(1), 115–130, doi:10.1111/1752-1688.12482, 2016.

Bieger, K., Arnold, J. G., Rathjens, H., White, M. J., Bosch, D. D. and Allen, P. M.: Representing the connectivity of upland areas to floodplains and streams in SWAT+, JAWRA Journal of the American Water Resources Association, doi:10.1111/1752-1688.12728, 2019.

Bosch, D. D., Sheridan, J. M., Lowrance, R. R., Hubbard, R. K., Strickland, T. C., Feyereisen, G. W. and Sullivan, D. G.: Little river experimental watershed database, Water Resources Research, 43(9), doi:10.1029/2006wr005844, 2007.

Byrd, R. H., Lu, P., Nocedal, J. and Zhu, C.: A limited memory algorithm for bound constrained optimization, SIAM Journal on Scientific Computing, 16(5), 1190–1208, doi:10.1137/0916069, 1995.

Carla Camargos, L. B., Tobias Houska: SWAT calibration and uncertainty assessment in parallel using a python tool, in EGU general assembly conference abstracts, vol. 20, Copernicus Publishing, Vienna, Austria., 2018.

Carnell, R.: Lhs: Latin hypercube samples. [online] Available from: `https://CRAN.R-project.org/package=lhs` (Accessed 5 March 2019), 2019.

Carson Sievert, T. H., Chris Parmer: plotly: Create Interactive Web Graphics via 'plotly.js'. [online] Available from: `https://CRAN.R-project.org/package=plotly` (Accessed 4 March 2019), 2018.

Chiew, F. H. and Vaze, J.: Hydrologic nonstationarity and extrapolating models to predict the future: Overview of session and proceeding, in IAHS-aish proceedings and reports, vol. 371, pp. 17–21., 2015.

Clark, M. P., Slater, A. G., Rupp, D. E., Woods, R. A., Vrugt, J. A., Gupta, H. V., Wagener, T. and Hay, L. E.: Framework for Understanding Structural Errors (FUSE): A modular framework to diagnose differences between hydrological models, Water Resources Research, 44(12), doi:10.1029/2007WR006735, 2008.

Clark, M. P., Wilby, R. L., Gutmann, E. D., Vano, J. A., Gangopadhyay, S., Wood, A. W., Fowler, H. J., Prudhomme, C., Arnold, J. R. and Brekke, L. D.: Characterizing Uncertainty of the Hydrologic Impacts of Climate Change, Current Climate Change Reports, 2, 55–64, doi:10.1007/s40641-016-0034-x, 2016.

Dile, Y., Srinivasan, R. and George, C.: QGIS interface for swat (qswat) version 1.7, [online] Available from: `https://swat.tamu.edu/software/qswat/` (Accessed 24

October 2018), 2018.

Dile, Y. T., Daggupati, P., George, C., Srinivasan, R. and Arnold, J.: Introducing a new open source GIS user interface for the SWAT model, Environmental Modelling & Software, 85, 129–138, doi:10.1016/j.envsoft.2016.08.004, 2016.

Duan, Q. Y., Gupta, V. K. and Sorooshian, S.: Shuffled complex evolution approach for effective and efficient global minimization, Journal of Optimization Theory and Applications, 76(3), 501–521, doi:10.1007/bf00939380, 1993.

Dutang C. and P., S.: Randtoolbox: Generating and testing random numbers. [online] Available from: `https://CRAN.R-project.org/package=randtoolbox` (Accessed 5 March 2019), 2018.

Fuka, D. R., Walter, M. T., MacAlister, C., Steenhuis, T. S. and Easton, Z. M.: SWATmodel: A multi-operating system, multi-platform SWAT model package in r, JAWRA Journal of the American Water Resources Association, 50(5), 1349–1353, doi:10.1111/jawr.12170, 2014.

Fuka DR, A. J., Walter MT: EcoHydRology: A community modeling foundation for eco-hydrology. [online] Available from: `https://CRAN.R-project.org/package=EcoHydRology` (Accessed 17 October 2018), 2018.

Fuka, W., DR: SWATmodel: A multi-os implementation of the tamu swat model. [online] Available from: `https://CRAN.R-project.org/package=SWATmodel` (Accessed 17 October 2018), 2014.

Gupta, H. V., Sorooshian, S. and Yapo, P. O.: Status of Automatic Calibration for Hydrologic Models: Comparison with Multilevel Expert Calibration, Journal of Hydrologic Engineering, 4(2), 135–143, doi:10.1061/(ASCE)1084-0699(1999)4:2(135), 1999.

Gupta, H. V., Kling, H., Yilmaz, K. K. and Martinez, G. F.: Decomposition of the mean squared error and NSE performance criteria: Implications for improving hydrological modelling, Journal of Hydrology, 377(1-2), 80–91, doi:10.1016/j.jhydrol.2009.08.003, 2009.

Guse, B., Pfannerstill, M. and Fohrer, N.: Dynamic Modelling of Land Use Change Impacts on Nitrate Loads in Rivers, Environmental Processes, 2(4), 575–592, doi:10.1007/s40710-015-0099-x, 2015.

Hengl, T., De Jesus, J. M., Heuvelink, G. B., Gonzalez, M. R., Kilibarda, M., Blagotić, A., Shangguan, W., Wright, M. N., Geng, X., Bauer-Marschallinger, B., Guevara, M. A., Vargas, R., MacMillan, R. A., Batjes, N. H., Leenaars, J. G., Ribeiro, E., Wheeler, I., Mantel, S. and Kempen, B.: SoilGrids250m: Global gridded soil information based on machine learning, PLoS one, 12(2), 1–40, doi:10.1371/journal.pone.0169748, 2017.

Henry, L. and Wickham, H.: Purrr: Functional programming tools. [online] Available from: `https://CRAN.R-project.org/package=purrr` (Accessed 3 May 2019), 2019.

Houska, T., Kraft, P., Chamorro-Chavez, A. and Breuer, L.: SPOTting model parameters using a ready-made python package, edited by D. Hui, PLOS ONE, 10(12), e0145180, doi:10.1371/journal.pone.0145180, 2015.

Iooss, B., Janon, A., Pujol, G., Khalid Boumhaout, Veiga, S. D., Delage, T., Fruth, J., Gilquin, L., Guillaume, J., Le Gratiet, L., Lemaitre, P., Nelson, B. L., Monari, F., Oomen, R., Rakovec, O., Ramos, B., Roustant, O., Song, E., Staum, J., Sueur, R., Touati, T. and Weber, F.: Sensitivity: Global sensitivity analysis of model outputs. [online] Available from: `https://CRAN.R-project.org/package=sensitivity` (Accessed 5 March 2019), 2018.

Joseph, J. and Guillaume, J.: Using a parallelized MCMC algorithm in r to identify appropriate likelihood functions for SWAT, Environmental Modelling & Software, 46, 292–298, doi:10.1016/j.envsoft.2013.03.012, 2013.

Mannschatz, T., Wolf, T. and Hülsmann, S.: Nexus tools platform: Web-based comparison of modelling tools for analysis of water-soil-waste nexus, Environmental Modelling & Software, 76, 137–153, doi:10.1016/j.envsoft.2015.10.031, 2016.

Mauricio Zambrano-Bigiarini: HydroGOF: Goodness-of-fit functions for comparison of simulated and observed hydrological time series., 2017.

Mehdi, B., Lehner, B., Gombault, C., Michaud, A., Beaudin, I., Sottile, M.-F. and Blondlot, A.: Simulated impacts of climate change and agricultural land use change on surface water quality with and without adaptation management strategies, Agriculture, Ecosystems & Environment, 213, 47–60, doi:10.1016/j.agee.2015.07.019, 2015.

Milly, P. C. D., Betancourt, J., Falkenmark, M., Hirsch, R. M., Kundzewicz, Z. W., Lettenmaier, D. P. and Stouffer, R. J.: Climate change. Stationarity is dead: whither water management?, Science, 319(5863), 573–574, doi:10.1126/science.1151915, 2008.

Morris, M. D.: Factorial sampling plans for preliminary computational experiments, Technometrics, 33(2), 161–174, 1991.

Müller, K. and Wickham, H.: Tibble: Simple data frames. [online] Available from: `https://CRAN.R-project.org/package=tibble` (Accessed 5 March 2019), 2019.

Nash, J. E. and Sutcliffe, J. V.: River flow forecasting through conceptual models part I - A discussion of principles, Journal of Hydrology, 10(3), 282–290, doi:10.1016/0022-1694(70)90255-6, 1970.

Neitsch, S., Arnold, J., Kiniry, J. and Williams, J.: Soil and Water Assessment Tool Theoretical Documentation Version 2009. TR-406 2011, Texas Water Resources Institute; Texas Water Resources Institute, College Station, TX. [online] Available from: `https://swat.tamu.edu/media/99192/swat2009-theory.pdf#urldate#`, 2011.

Odusanya, A. E., Mehdi, B., Schürz, C., Oke, A. O., Awokola, O. S., Awomeso, J. A., Adejuwon, J. O. and Schulz, K.: Multi-site calibration and validation of SWAT with satellite-based evapotranspiration in a data-sparse catchment in southwestern nigeria, Hydrology and Earth System Sciences, 23(2), 1113–1144, doi:10.5194/hess-23-1113-2019, 2019.

Pfannerstill, M., Bieger, K., Guse, B., Bosch, D. D., Fohrer, N. and Arnold, J. G.: How to constrain multi-objective calibrations of the SWAT model using water balance components, JAWRA Journal of the American Water Resources Association, 53(3), 532–546, doi:10.1111/1752-1688.12524, 2017.

R Core Team: R: A language and environment for statistical computing., [online] Available from: `https://www.r-project.org/` (Accessed 5 March 2019), 2019.

Rakovec, O., Hill, M. C., Clark, M. P., Weerts, A. H., Teuling, A. J. and Uijlenhoet, R.: Distributed Evaluation of Local Sensitivity Analysis (DELSA), with application to hydrologic models, Water Resour. Res, 50, 409–426, doi:10.1002/2013WR014063, 2014.

Reusser, D.: Fast: Implementation of the fourier amplitude sensitivity test (fast). [online] Available from: `https://CRAN.R-project.org/package=fast` (Accessed 5 March 2019), 2015.

Rockström, J., Steffen, W., Noone, K., Persson, Chapin, F. S. I., Lambin, E., Lenton, T. M., Scheffer, M., Folke, C., Schellnhuber, H. J., Nykvist, B., Wit, C. A. de, Hughes, T., Leeuw, S. van der, Rodhe, H., Sörlin, S., Snyder, P. K., Costanza, R., Svedin, U., Falkenmark, M., Karlberg, L., Corell, R. W., Fabry, V. J., Hansen, J., Walker, B., Liverman, D., Richardson, K., Crutzen, P. and Foley, J.: Planetary boundaries: Exploring the safe operating space for humanity, Ecology and Society, 14(2), doi:10.5751/es-03180-140232, 2009.

Ross, Z., Wickham, H. and Robinson, D.: Declutter your r workflow with tidy tools, PeerJ, doi:10.7287/peerj.preprints.3180v1, 2017.

Saltelli, A., Ratto, M., Andres, T., Campolongo, F., Cariboni, J., Gatelli, D., Saisana, M. and Tarantola, S.: Global Sensitivity Analysis. The Primer, John Wiley & Sons, Ltd, Chichester, UK., 2008.

Schulz, K., Beven, K. and Huwe, B.: Equifinality and the problem of robust calibration in nitrogen budget simulations, Soil Science Society of America Journal, 63(6), 1934–1941, doi:10.2136/sssaj1999.6361934x, 1999.

Schürz, C.: ARastoCAT: A raster climate data aggregation tool. [online] Available from: `https://github.com/chrisschuerz/aRastoCAT` (Accessed 5 March 2019a), 2018.

Schürz, C.: Soilgridr: Working with soilgrids data in r. [online] Available from: `https://github.com/chrisschuerz/solACE` (Accessed 5 March 2019b), 2018.

Schürz, C., Strauch, M., Mehdi, B. and Schulz, K.: SWATfarmR: A simple rule-based scheduling of management operations for swat, in Book of abstracts for the 2017 swat conference, Wersaw, Poland., 2017a.

Schürz, C., Strauch, M., Mehdi, B. and Schulz, K.: SWATpasteR: Parallel swat execution and sensitivity testing in r, in Book of abstracts for the 2017 swat conference, Wersaw, Poland., 2017b.

Schürz, C., Hollosi, B., Matulla, C., Pressl, A., Ertl, T., Schulz, K. and Mehdi, B.: A comprehensive sensitivity and uncertainty analysis for discharge and nitrate-nitrogen loads involving multiple discrete model inputs under future changing conditions, Hydrology and Earth System Sciences, 23(3), 1211–1244, doi:10.5194/hess-23-1211-2019, 2019.

Slater, L. J., Thirel, G., Harrigan, S., Delaigue, O., Hurley, A., Khouakhi, A., Prodoscimi, I., Vitolo, C. and Smith, K.: Using r in hydrology: A review of recent developments and future directions, Hydrology and Earth System Sciences Discussions, 1–33, doi:10.5194/hess-2019-50, 2019.

Sobol, I. M.: Sensitivity analysis for nonlinear mathematical models, Mathematical Modelling and Computational Experiments, 4(1), 407–414, doi:10.18287/0134-2452-2015-39-4-459-461., 1993.

Steffen, W., Richardson, K., Rockstrom, J., Cornell, S. E., Fetzer, I., Bennett, E. M., Biggs, R., Carpenter, S. R., Vries, W. de, Wit, C. A. de, Folke, C., Gerten, D., Heinke, J., Mace, G. M., Persson, L. M., Ramanathan, V., Reyers, B. and Sorlin, S.: Planetary boundaries: Guiding human development on a changing planet, Science, 347(6223), 1259855–1259855, doi:10.1126/science.1259855, 2015.

Sutton, M. A. and Bleeker, A.: The shape of nitrogen to come, Nature, 494(7438), 435–437, doi:10.1038/nature11954, 2013.

Teshager, A. D., Gassman, P. W., Schoof, J. T. and Secchi, S.: Assessment of impacts of agricultural and climate change scenarios on watershed water quantity and quality, and crop production, Hydrology and Earth System Sciences, 20(8), 3325–3342, doi:10.5194/hess-20-3325-2016, 2016.

Tolson, B. A. and Shoemaker, C. A.: Dynamically dimensioned search algorithm for computationally efficient watershed model calibration, Water Resources Research, 43(1), doi:10.1029/2005wr004723, 2007.

UNEP: A Snapshot of the World's Water Quality: Towards a global assessment., [online] Available from: `https://uneplive.unep.org/media/docs/assessments/`

unep_wwqa_report_web.pdf (Accessed 28 January 2019), 2016.

Vanderkam, D., Allaire, J., Owen, J., Gromer, D. and Thieurmel, B.: Dygraphs: Interface to 'dygraphs' interactive time series charting library. [online] Available from: https://CRAN.R-project.org/package=dygraphs (Accessed 4 March 2019), 2018.

Venohr, M., Hirt, U., Hofmann, J., Opitz, D., Gericke, A., Wetzig, A., Ortelbach, K., Natho, S., Neumann, F. and Hürdler, J.: The Model System MONERIS, Version 2.14.1vba, Leibniz-Institute of Freshwater Ecology; Inland Fisheries (IGB-Berlin), Berlin, Germany. [online] Available from: http://www.moneris.igb-berlin.de/tl_files/data_moneris/data_publikationen/Moneris%20Handbuch/Handbuch_englisch12_03.2010.pdf (Accessed 3 April 2017), 2010.

Wagner, P. D., Bhallamudi, S. M., Narasimhan, B., Kumar, S., Fohrer, N. and Fiener, P.: Comparing the effects of dynamic versus static representations of land use change in hydrologic impact assessments, Environmental Modelling and Software, 1–9, doi:10.1016/j.envsoft.2017.06.023, 2017.

White, M. J., Harmel, R. D., Arnold, J. G. and Williams, J. R.: SWAT check: A screening tool to assist users in the identification of potential model application problems, Journal of Environment Quality, 43(1), 208, doi:10.2134/jeq2012.0039, 2014.

Wickham, H.: Tidy data, Journal of Statistical Software, 59(10), doi:10.18637/jss.v059.i10, 2014.

Wickham, H.: Ggplot2: Elegant graphics for data analysis, Springer-Verlag New York. [online] Available from: http://ggplot2.org, 2016.

Wickham, H.: Tidyverse: Easily install and load the 'tidyverse'. [online] Available from: https://CRAN.R-project.org/package=tidyverse (Accessed 5 March 2019), 2017.

Wickham, H. and Henry, L.: Tidyr: Easily tidy data with 'spread()' and 'gather()' functions. [online] Available from: https://CRAN.R-project.org/package=tidyr (Accessed 3 May 2019), 2018.

Wickham, H., Hester, J. and Chang, W.: Devtools: Tools to make developing r packages easier. [online] Available from: https://CRAN.R-project.org/package=devtools (Accessed 5 March 2019a), 2018.

Wickham, H., Hester, J. and Francois, R.: Readr: Read rectangular text data. [online] Available from: https://CRAN.R-project.org/package=readr (Accessed 5 March 2019b), 2018.

Wickham, H., François, R., Henry, L. and Müller, K.: Dplyr: A grammar of data manipulation. [online] Available from: https://CRAN.R-project.org/package=

dplyr (Accessed 5 March 2019), 2019.

Winchell, M., Srinivasan, R., Di Luzio, M. and Arnold, J. G.: ArcSWAT 2012.10.19 Interface for SWAT2012, [online] Available from: http://swat.tamu.edu/software/arcswat/ (Accessed 27 January 2017), 2015.

Wu, Y. and Liu, S.: Automating calibration, sensitivity and uncertainty analysis of complex models using the r package flexible modeling environment (FME): SWAT as an example, Environmental Modelling & Software, 31, 99–109, doi:10.1016/j.envsoft.2011.11.013, 2012.

Wu, Y. and Liu, S.: Improvement of the r-SWAT-FME framework to support multiple variables and multi-objective functions, Science of The Total Environment, 466-467, 455–466, doi:10.1016/j.scitotenv.2013.07.048, 2014.