SWAT+ modeling protocol for the assessment of water and nutrient retention measures in small agricultural catchments

This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant agreement No. 862756.

Christoph Schürz[1] Natalja Čerkasova[2] Csilla Farkas[3] Attila Nemes[4]
Svajunas Plunge[5] Michael Strauch[6] Brigitta Szabó[7] Mikołaj Piniewski[8]

2022-12-20

[1]Helmholtz Centre for Environmental Research - UFZ, `christoph.schuerz@ufz.de`
[2]Klaipeda University, Marine Research Institute, `natalja.cerkasova@ku.lt`
[3]Norwegian Institute of Bioeconomy Research, `csilla.farkas@nibio.no`
[4]Norwegian Institute of Bioeconomy Research, `attila.nemes@nibio.no`
[5]Warsaw University of Life Sciences, `svajunas_plunge@sggw.edu.pl`
[6]Helmholtz Centre for Environmental Research - UFZ, `michael.strauch@ufz.de`
[7]Centre for Agricultural Research, Institute for Soil Sciences, `szabo.brigitta@atk.hu`
[8]Warsaw University of Life Sciences, `mikolaj_piniewski@sggw.edu.pl`

Contents

Please check the latest document version with DOI: doi.org/10.5281/zenodo.7462415

Project Consortium

Document Information

Program	EU Horizon 2020 Research and Innovation Action H2020-EU.3.2.1.1 (SFS-23-2019)
Grant agreement No.	862756
Project acronym	OPTAIN
Project full name	Optimal strategies to retain and re-use water and nutrients in small agricultural catchments across different soil-climatic regions in Europe
Start of the project	September 2020
Duration	60 months
Project coordination	Prof. Dr. Martin Volk Helmholtz-Centre for Environmental Research GmbH - UFZ www.optain.eu
Deliverable	D4.2: Modelling protocols. Part 2 SWAT+ modelling protocol
Work package	WP4: Integrated assessment of NSWRM
Task	Task 4.2: Development of modelling protocols
Lead beneficiary	NIBIO
Author(s)	Christoph Schürz (UFZ), Natalja Čerkasova (KU), Csilla Farkas (NIBIO), Attila Nemes (NIBIO), Svajunas Plunge (WULS), Michael Strauch (UFZ), Brigitta Szabó (ATK), Mikołaj Piniewski (WULS)
Contributor(s)	Petr Fucik (VUMOP), Luka Honzak (UL)
Quality check	Felix Witing (UFZ), Martin Volk (UFZ)
Planned delivery date	Month 26 (October 2022)
Actual delivery date	20/12/2022
Citation	Schürz, C., Čerkasova, N., Farkas, C., Nemes, A., Plunge, S., Strauch M., Szabó, B., Piniewski, M. (2022), SWAT+ modeling protocol for the assessment of water and nutrient retention measures in small agricultural catchments. Deliverable D4.2 EU Horizon 2020 OPTAIN Project, Grant agreement No. 862756
Dissemination level*	PU

*PU = Public; PP = Restricted to other program participants (including the Commission Services; CO = Confidential, only for members of the consortium (including the Commission Services).

Deliverable status

Version	Date	Author(s)/Contributor(s)	Notes
0.5	30.11.2022	Christoph Schürz (UFZ), Natalja Čerkasova (KU), Csilla Farkas (NIBIO), Attila Nemes (NIBIO), Svajunas Plunge (WULS), Michael Strauch (UFZ), Brigitta Szabó (ATK), Mikołaj Piniewski (WULS)	First complete draft
0.6	08.12.2022	Felix Witing (UFZ), Martin Volk (UFZ)	Revision
0.7	16.12.2022	Christoph Schürz (UFZ), Natalja Čerkasova (KU), Csilla Farkas (NIBIO), Attila Nemes (NIBIO), Svajunas Plunge (WULS), Michael Strauch (UFZ), Brigitta Szabó (ATK), Mikołaj Piniewski (WULS)	Second draft
0.8	08.12.2022	Felix Witing (UFZ), Martin Volk (UFZ)	Revision
0.9	20.12.2022	Christoph Schürz (UFZ), Natalja Čerkasova (KU), Csilla Farkas (NIBIO), Attila Nemes (NIBIO), Svajunas Plunge (WULS), Michael Strauch (UFZ), Brigitta Szabó (ATK), Mikołaj Piniewski (WULS)	Third draft
1.0	20.12.2022	Felix Witing (UFZ), Martin Volk (UFZ)	Final

Abbreviation list

APEX Agricultural Policy / Environmental eXtender
AWC Available Water Capacity
BMPs Best Management Practices
COCOA COntiguous object COnnectivity Approach
CS Case Study
DA Drainage Area
DEM Digital Elevation Model
DT Decision Table
EMEP European Monitoring and Evaluation Programme
ET Evapotranspiration
ETa Actual Evapotranspiration
EU-HYDI European Hydropedological Data Inventory
EURO-CORDEX European branch of the Coordinated Regional Climate Downscaling project
FC Water Content at Field Capacity
GCM General Circulation Model
GIS Geographic Information System
GUI Graphical User Interface
HG Hydraulic Geometry
HRUs Hydrologic Response Units
HSG Hydrologic Soil Groups
HYSOGs250m Global gridded hydrologic soil groups dataset for curve-number-based runoff modelling
LAI Leaf Area Index
LIDAR Light Detection and Ranging
MARG Multi-Actor Reference Group
NSE Nash Sutcliffe Efficiency
NSWRMs Natural/Small Water Retention Measures
OPTAIN OPtimal strategies to retAIN and re-use water and nutrients in small agricultural catchments across different soil-climatic regions in Europe
P Phosphorus
PET Potential Evapotranspiration
PM Penman-Monteith
PTF Pedotransfer Function
RCM Regional Climate Model
RCP Representative Concentration Pathway
RTUs Routing Units
SA Sensitivity Analysis
SC Soft Calibration
SONAR Sound Navigation and Ranging
SWAP Soil Water Atmosphere Plant
SWAT Soil and Water Assessment Tool
SWAT+ Soil and Water Assessment Tool Plus
UA Uncertainty Analysis
USLE Universal Soil Loss Equation
WB Water Balance
WLP Water Content at Wilting Point
WP Work Package
WWTPs Waste Water Treatment Plants

Chapter 1

Introduction

This SWAT+ modelling protocol (further - protocol) was designed for guiding model setup development and model calibration in 14 European Case Study (CS) sites participating in the modelling component of the EU funded research and innovation project OPtimal strategies to retAIN and re-use water and nutrients in small agricultural catchments across different soil-climatic regions in Europe (OPTAIN). These 14 CSs are small agricultural catchments (ranging in size from 21 to 254 km^2) located in three biogeographical regions of Europe and 12 different countries (Fig. 1.1). The main topic of OPTAIN are Natural/Small Water Retention Measures (NSWRMs), which are a relatively new concept. These are small and multi-functional measures for the retention/management of water and nutrients in the landscape, thus addressing drought/flood control, management of water quality problems, climate change adaptation, biodiversity restoration, etc.

1.1 Position of the protocol within OPTAIN

The protocol is an output of the Work Package (WP) 4 “Integrated assessment of NSWRMs” and belongs to OPTAINs Task 4.2 “Development of modelling protocols”. It is most closely connected to the Task 4.4 “Assessment of NSWRM effectiveness at the catchment scale”. Basically, within this task the case study modellers are supposed to set up and calibrate their SWAT+ models as well as apply them for running scenarios related to climate change and implementation of NSWRM, following the recommendations from the protocol. This will ensure a harmonised modelling approach which is one of the core concepts OPTAIN.

The protocol is also related to other activities and outputs of OPTAIN. It is strongly linked to WP3 “Retrieval of modelling data and solutions to overcome data scarcity”, which has already provided three deliverables with valuable inputs and tools in the context of this protocol:

1. D3.1 “Climate scenarios for integrated modelling” (Honzak and Pogačar, 2022);
2. D3.2 “Solutions to overcome data scarcity” (Szabó et al., 2022);
3. D3.3 “Created data pre-processors successfully applied for input data restructuring” (Čerkasova et al., 2022).

Within WP2 “Measures and indicators” the following deliverables provided an important foundation for the work in WP4:

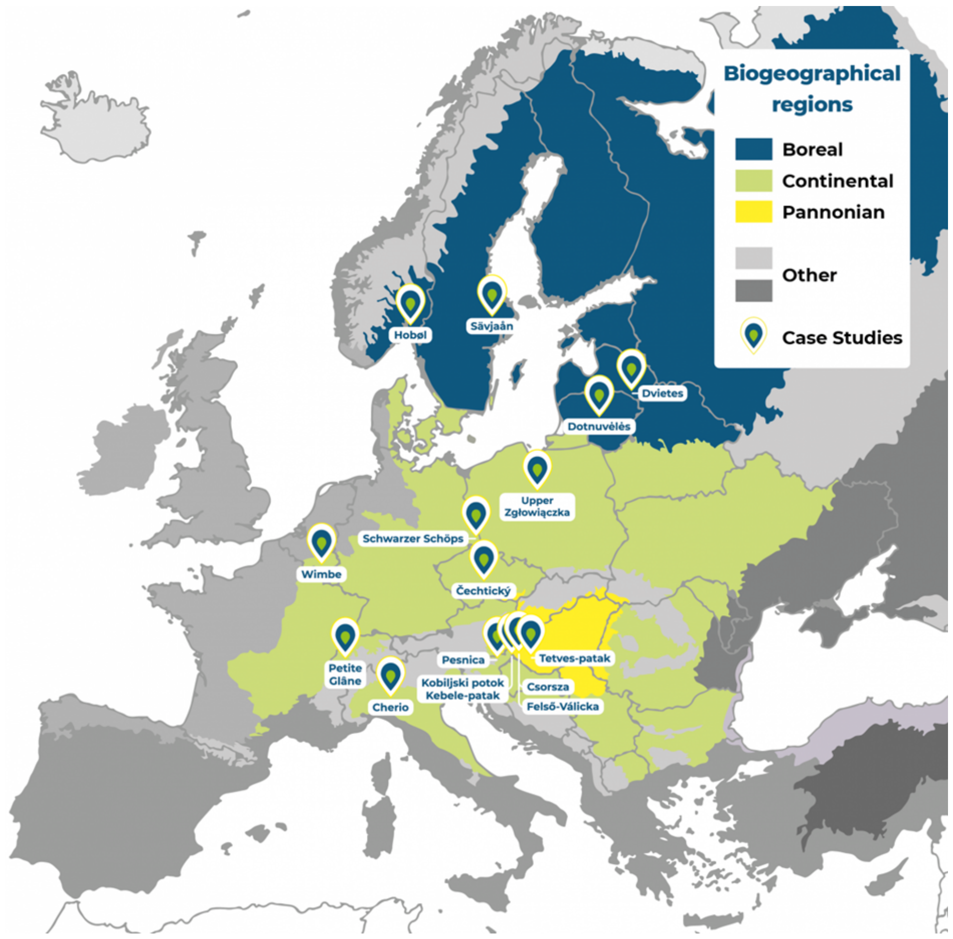


Figure 1.1: Location of 14 OPTAIN case studies in Europe.

1. Deliverable D2.1 "Coherent catalogue with a selection of most promising NSWRM including results from Multi-Actor Reference Group (MARG) exchanges" specified the OPTAIN understanding of NSWRM and selected the measures that will be modelled in the case studies (Lemann et al., 2022).

2. Deliverable D2.3 "Participatory modelling settings and standardised guidelines for parametrisation of measures" provided relevant information about parametrisation of selected NSWRM in SWAT+ (Marval et al., 2022).

1.2 Aim of the protocol

The Soil and Water Assessment Tool Plus (SWAT+) (Bieger et al., 2017) is a continuation of the Soil and Water Assessment Tool (SWAT) (Arnold et al., 1998; Arnold and Fohrer, 2005) model development, this new model version brings extensive changes with new concepts into a well-established modelling tool. New changes might provide large advancements in enhanced options of NSWRMs modelling. Biophysical simulation of NSWRM functioning and performance under different climatic conditions can be performed using models such as SWAT+. However, a consistent and state-of-the-art methodological guidance on best modelling practices as well as tools for aiding the modelling process to access the full potential of the changes brought with the new model version are yet missing.

To close these gaps, the main aim of this document was to prepare a comprehensive SWAT+ modelling protocol supported with additional tools to facilitate the modelling process of NSWRM, with a focus on small agricultural catchments in Europe. Although there are many publications, tools and user groups available to support SWAT model users, they are spread over many sources and may in some cases provide misleading or inadequate recommendations. Thus, we here intend to integrate state-of-the-art recommendations from literature with the unique expertise of the co-authors of this document in order to lay out a fit-for-purpose way of setting up and calibrating the SWAT+ model. In addition, the protocol for the first time introduces the new SWAT+ model setup approach COntiguous object COnnectivity Approach (COCOA). COCOA can represent features in the landscape at the field scale and accounts for the connectivity between land phase objects (Hydrologic Response Units (HRUs)). It is a fundamental change in process-based hydrological modelling that will allow for a more realistic representation of the measures in the model setup as well as more realistic model outputs related to simulated effectiveness of measures.

Although the protocol was designed specifically as a resource for the OPTAIN modellers and case studies, it also represents a valuable source of information and tools for the wider SWAT+ modellers' community. In particular, the target audience of this guideline are all SWAT+ modellers worldwide who apply their model in small agricultural catchments and focus on application of measures and climate change impact assessment. However, in some aspects, the protocol may be more relevant for European model users due to the very nature of data availability in Europe.

1.3 Available tools and guidance

1.3.1 SWAT+ model

SWAT is a conceptual, continuous time watershed model applied to simulate the quality and quantity of surface and ground water (Arnold et al., 1998; Arnold and Fohrer, 2005). The model has a great number of successful application examples for different eco-hydrological questions in multiple geographical regions spanning over more than two decades (Akoko et al., 2021; Gassman et al., 2014; Tan et al., 2020, 2019). The SWAT model code has undergone some major modifications assigned to versions of SWAT2000, SWAT2005, SWAT2009, SWAT2012. However, to meet present and future challenges in water resources modelling and in management of global users' community, a completely restructured version named SWAT+ has been recently developed (Bieger et al., 2017). One of the main improvements was increased flexibility of water routing across a landscape providing a better representation of connectivity (Bieger et al., 2019; White et al., 2022), which is further enhanced in the newly developed COCOA approach and tool (presented in this protocol in section 2.1). Another important advancement was the implementation of decision tables, which allows including complex land management operations and reservoir management in the model (Arnold et al., 2018). An advanced soft calibration routine has been incorporated that helps the modellers to check the water and mass balance elements and prevents unrealistic parametrisation via auto-calibration (White et al., 2022).

1.3.2 Available tools

The SWAT model is a command line tool using text input and output files. Therefore, several tools have been prepared and are currently available for the SWAT+ users to help with model application. They could be roughly divided into three main categories:

1) Model setup preparation (handling Geographic Information System (GIS) data, delineating HRUs, watershed configuration, etc.).

2) Parameter editing and model running (opening/editing model databases, rerunning model with updated parameters).
3) Calibration/validation and other (running parameter optimization algorithms, calculating parameter sensitivities and model efficiency coefficients).

Among the most broadly used tools in the first category is the QGIS interface for SWAT+ (QSWAT+), which is the primary tool (the only one publicly available and working at the time of writing the protocol for general SWAT+ users) used by SWAT+ modellers in preparing baseline model setups. However, in the OPTAIN project an alternative tool for building model setups called `SWATbuildR` was developed and is further described in the 2.2 chapter.

The second category of available tools includes the SWAT+ Editor, whose main role is updating model databases, running scenarios, checking model outputs, etc. At the time of writing the protocol no publicly available alternative to this tool was available.

SWAT+ model users have a wider choice of tools in the third category:

- SWATplusR is a tool for running the SWAT+ model in a R environment. It is useful for analysing model sensitivity, running calibration and validation as well as simulating scenarios. The greatest strength of this tool comes from the availability of multiple other tools that are accessible in the R environment, which could be integrated with SWAT+ model applications. SWATplusR is recommended to be used for calibration and uncertainty analysis within OPTAIN.
- SWATplus-CUP is among the most sophisticated and widely used Graphical User Interface (GUI)-type tools available for the calibration of SWAT+ models. It is a continuation of the SWAT-CUP software that gained significant popularity among SWAT users globally in the recent decade. It provides various algorithms for model calibration, validation, sensitivity analysis and uncertainty analysis. The noticeable drawback compared to other tools is that it is proprietary, requiring purchasing a license for working with it.
- SWAT+ Toolbox is a user-friendly tool for SWAT+ model adaptations including sensitivity, calibration and output checking capabilities[1].
- IPEAT+ is a FORTRAN-based, built-in, and open source optimization and automatic calibration tool of SWAT+ (Yen et al., 2019).

These are the default tools for SWAT+ model users to start building their model applications, if no specifically tailored options are available.

1.3.3 SWAT+ documentations and source code

A key source for modellers preparing SWAT+ model setups is the SWAT+ input/output documentation, which explains model file structures, parameter definitions, and even some theoretical background. However, at the time of writing this protocol, the SWAT+ documentation was still under development, so alternative sources could be the SWAT documentations (input/output for SWAT2012 and theoretical for SWAT2019). The SWAT+ model source code is probably the most relevant resource for programmers, but it is as well useful for model users. Understanding the source code is important if something is missing in the model documentation or if the documentation has not yet been updated for the latest revision.

Some useful materials could also be found on the official SWAT+ model website, e.g. the Hydrological Modelling Using SWAT+ Training Manual or links to other manuals, video lectures and user groups. The reader is encouraged to explore these resources, especially as a part of self-training at the beginners' level.

[1]During preparation of the protocol this tool lacked stability and update to be able running with the latest SWAT+ release.

1.3.4 Available modelling protocols and guidelines

As the SWAT+ model is relatively new and in the active development stage, few literature sources are yet available for the tool. However, several documents tailored for the SWAT model could be useful for the SWAT+ model users. From general sources, one of the most comprehensive guidelines for modellers is published in the article NRES-21 Hydrology committee of ASABE (2017) available on SWAT website. The article leads modellers through all important steps in a model application with general tips and recommendations.

SWAT calibration techniques and important parameters to be manipulated during model calibration are explained in a slide format as SWAT calibration techniques. The importance of using soft data in addition to hard data in model calibration is well described by Arnold et al. (2016). Useful guidelines on Best Management Practices (BMPs) modelling are provided in the document of Conservation Practice Modeling Guide for SWAT and Agricultural Policy / Environmental eXtender (APEX) by Waidler et al. (2009), although in this case some of the representations of conservation practices may not fit to SWAT+. The SWAT+ and Soil Water Atmosphere Plant (SWAP) retention measure implementation handbook (Marval et al., 2022) developed within the OPTAIN project provides SWAT+ relevant information about the parametrisation of NSWRM.

1.3.5 Useful information in published scientific literature

Peer-reviewed scientific literature on the SWAT+ model is growing rapidly, but is still much less abundant than on the SWAT model. Consequently, the identified references were mostly indented for the SWAT model. However, numerous recommendations provided therein should be also useful for modellers working with the SWAT+ model.

For general recommendations on SWAT model calibration and uncertainty assessment, Abbaspour et al. (2015) provide information using an European scale model as an example. Despite the large scale, several aspects such as the inclusion of crop yields in calibration, trials with different input datasets, hints about using parameters in calibration could also be applicable for small-scale model applications.

Additional SWAT calibration and uncertainty assessment recommendations could be found in Abbaspour et al. (2017) article named "A Guideline for Successful Calibration and Uncertainty Analysis for Soil and Water Assessment: A Review of Papers from the 2016 International SWAT Conference". According to the authors of this paper:

> "(...) focus on calibration and uncertainty analysis highlighting some serious issues in the calibration of distributed models. A protocol for calibration is also highlighted to guide the users to obtain better modeling results."

However, provided guidelines and protocols are directly relevant to SWAT-CUP software users. In OPTAIN a novel calibration workflow has been developed that employs the `SWATPlusR` tool (presented in chapter 6.2.4).

Arnold et al. (2012b) provided guiding principles on the SWAT model use, calibration and validation. Established "good" model calibration and validation practices are described in the paper.

Five additional general review papers could be useful. First one by Harmel et al. (2014) provides a review and recommendations of hydrologic/water quality models evaluation, interpretation, and communicating of performance. Second by Fu et al. (2018) provides a comprehensive review of catchment-scale water quality and erosion models (including SWAT) while discussing main modelling steps and challenges. The third paper by Jakeman et al. (2006) lays down a good model development practice, which envelopes also reporting of results and review of models. Fourth paper by Fu et al.

(2020) reviews existing practices and challenges of watershed water quality modelling, and discusses areas of potential improvement. Lastly, we would like to point to a publication by Moriasi et al. (2015) on model performance and evaluation criteria, which are important for the calibration and validation steps.

1.4 Workflow of SWAT+ model setup in OPTAIN

SWAT/SWAT+ is by far the most popular eco-hydrological model (Mannschatz et al., 2016). To a large extent a common, "conventional" modelling workflow can be found in the SWAT literature. SWAT/SWAT+ models are set up employing one of the graphical interfaces ArcSWAT, QSWAT, or QSWAT+ which all follow the same principles in model setup. For SWAT+ models any further parametrization is done with the SWAT+ Editor. Model calibration is typically performed with SWAT-CUP/SWATplus-CUP (Abbaspour, 2015) using the Sequential uncertainty fitting method (SUFI II, Abbaspour et al. (2004), Abbaspour et al. (1997)). A single best model is selected then to perform scenario runs. In OPTAIN we designed and propose a novel workflow which may follow some similar principles, but differs substantially from a conventional SWAT modelling procedure in the following aspects:

- The baseline model setup is performed with a newly developed R-based tool that implements a novel philosophy of contiguous object-based connectivity. Thus, the established model setups will structurally substantially differ from model setups which are set up with the currently used GIS based interface QSWAT+;
- The novel structural concept which is applied in the model setup allows a better spatial representation of landscape features and \acr{NSWRMs]. Thus, the implementation of structural \acr{NSWRMs] in model setups will strongly differ from a conventional parametric approach and can hopefully improve the model representation of such measures;
- A conventional model setup initializes many model parameters with generic literature values (which may not fit regional characteristics) and modellers tend to ignore them in the model parametrization. In OPTAIN a lot of attention is paid to using best available data, methods and approaches for model parametrization to limit the use of generic initial parameter values;
- A special attention is paid to agricultural management practices, with a workflow including a new crop classification tool and a tool that allows for a more realistic execution of management operations in SWAT+;
- A new workflow for model calibration is proposed that emphasizes the role of input data verification and soft calibration, while focusing hard calibration on processes and their related parameters.

As aforementioned, this protocol was designed to deliver guidance on using best modelling practices and ensuring a high methodological standard and comparability across case studies within the OPTAIN project. Therefore case studies are expected to go through (or consider) each step provided in the workflow when developing their models. The protocol structure is built to be aligned with the workflow steps. The following workflow steps should be considered:

- 1. **Input data collection and preparation**
  - 1.1. `SWATbuildR` input data preparation
  - 1.2. Weather data preparation

- 2. **Model setup preparation**
  - 2.1. Model setup with `SWATbuildR`
  - 2.2. Weather data importing
  - 2.3. ***Initial model run***
- 3. **Model parametrization**
  - 3.1. Land use *'landuse.lum'*
  - 3.2. Channel properties *'hyd-sed-lte.cha'*
  - 3.3. Crops *'plant.ini/plants.plt'*
  - 3.4. Soil physical data *'soils.sol'*
  - 3.5. Soil chemical data *'nutrient.sol'*
  - 3.6. Impoundments *'hydrology.res'*
  - 3.7. Water withdrawals *'water_rights.wro'*
  - 3.8. Point sources *'recall.rec'*
  - 3.9. Tile drainage *'tiledrain.str'*
  - 3.10. Atmospheric deposition *'atmo_dep'*
  - 3.11. Additional settings *'codes.bsn/parameters.bsn'*
- 4. **Agricultural management**
  - 4.1. Data preparation
  - 4.2. Setup of crop schedules
  - 4.3. Builing management files with `SWATfarmR`
- 5. **Decision tables**
  - 5.1. Land Use and Management *'lum.dtl'*
  - 5.2. Land Use Scenarios *'scen_lu.dtl'*
  - 5.3. Reservoir Release *'res.dtl'*
- 6. **Model verification, calibration and validation**
  - 6.1. Calibration/validation data
  - 6.2. Model verification
  - 6.3. Soft calibration
  - 6.4. Hard calibration
  - 6.5. Model validation
- 7. **Implementation of scenarios**
  - 7.1. Climate scenarios
  - 7.2. NSWRM scenarios
  - 7.3. Combined scenarios
  - 7.4. Uncertainty analysis

Chapter 2

Baseline model setup

As the baseline model setup we define a SWAT+ model setup which includes all inputs to form an executable SWAT+ model. The minimum input definitions include all spatial objects (HRUs, Routing Units (RTUs), channels, reservoirs, aquifers) which define the modelled landscape features and the connections between all spatial objects. Further such a model setup already includes a basic parametrization of the spatial objects, mostly spatial terrain and soil properties which can be derived from the required raster inputs and weather inputs.

For the definition of the spatial objects and the connections between spatial objects we developed a novel method to set up SWAT+ models which follows the COCOA approach. This concept uses the field scale to discretize features in the landscape. These landscape units form a contiguous representation of the landscape. Based on the terrain properties all landscape features are connected with contiguous neighbor objects. COCOA is implemented in the model setup tool `SWATbuildR`.

The following sections give an introduction to the conceptual framework of COCOA and show the differences to a conventional model setup with QSWAT+. The next two sections cover the model setup procedure with the `SWATbuildR`, where first the preparation of the required model inputs is described and then the actual model setup process is explained for the currently available script version (version 1.0.4) of `SWATbuildR`. The last section in this chapter covers the weather input data and the weather generator data which must be implemented to have an executable SWAT+ model setup.

2.1 Contiguous object connectivity approach

The SWAT+ model setups consist of spatial objects that represent the land areas, channels, reservoirs, aquifers, and other components of a watershed system. Spatial objects can be mutually connected to enable different types of fluxes between them, such as surface runoff, lateral flow, and groundwater recharge. The introduction of spatial objects and their connectivity is a substantial improvement of the SWAT model and allows for a flexible representation of hydrological catchment systems.

Yet, the model setup process with the QSWAT+ v2.2 GIS interface is still based on concepts that were established for SWAT2012 model setups (such model setups or the model setup process is referred to as ‘conventional’ in the following). Thus, the conventional model setup procedure with QSWAT+ does not take full advantage of the flexibility of the model in configuring a watershed. The smallest spatial land phase unit that has a spatial reference in such a conventional setup is the routing unit (RTU). RTUs usually contain multiple HRUs, which lump areas with the same land use, soil and similar slope in an unit without a spatial reference. Thus, HRUs usually aggregate areas in an RTU, which are scattered across the landscape, and which are not connected.

Although agricultural fields could be defined as unique and separated HRUs also in conventional SWAT model setups, the representation of the landscape is still rather abstract in such model setups, as the connectivity between land units is not accounted for at all. Therefore, spatial structural measures can only be implemented in an abstract way and measures are usually represented by changes of model parameter values that impact the hydrological processes of interest. Parameter values often do not have a clear physical reference and are empirical and abstract instead. Yet, literature values may be available for most of the model parameters that represent certain measures.

In OPTAIN we have two spatial scales of interest - the catchment scale, but also the field scale. The catchment scale model applications are used to investigate the sum of the field scale effects (for instance introduced by NSWRMs) at the catchment scale. Thus, conventional model setups, which strongly aggregate the landscape at the scale of interest, may be inappropriate to evaluate the implementation of NSWRMs. Consequently, a new approach for SWAT+ model setups is being developed in OPTAIN that can represent features in the landscape at the field scale and accounts for the connectivity between land phase objects (HRUs). With this approach each individual landscape feature in a land cover layer needs to be a contiguous spatial unit and is eventually considered as a unique object in the SWAT+ model setup. Contiguous units are here defined as in itself contained areas with clearly defined borders to neighbouring units (which is different to the fragmented representation of the landscape within conventional model setups). The connectivity between contiguous objects is calculated based on terrain properties which are calculated with a raster DEM.

When the different spatial configurations of SWAT+ model setups are initiated conventionally in QSWAT+ or with COCOA, they lead to a different representation of NSWRMs. Different landscape representations within SWAT are not necessarily a novel concept (example in Rathjens and Oppelt (2012) paper). Also for SWAT2012, approaches were documented to implement a hillslope discretization in order to simulate the retention behaviour of grassland and filter strips for runoff from e.g. more erosion prone areas (see e.g. Appendix B in SWAT 2012 Input/Output documentation). Yet, this modification in the model setup was rarely implemented, as it requires substantial manual modifications in the input files. COCOA aims to automatize these modifications in the model setup process.

COCOA represents the landscape with greater detail. In such a setup, spatial structural measures (and potential locations of measures) are represented by unique spatial objects. The implementation of a measure (e.g., the transformation of an agricultural area to a retention area) is then for example represented by a transformation in the land use of the respective spatial object. The effectiveness of the measure in such a model setup is then strongly impacted by the connectivity between the spatial objects and the retention properties of a certain land use.

For COCOA, it is necessary that all measures, which are going to be modelled in later scenario simulations, are represented as unique polygons in the land use map. Such an extended land use map must be used as input for setting up the SWAT+ models.

In the following text, we will focus on the SWAT+ model setups following the COCOA approach. The tool that was developed to set up SWAT+ models in R following the COCOA approach is called `SWATbuildR`.

2.2 `SWATbuildR` input data preparation

`SWATbuildR` connects spatial objects such as land, water, or aquifer objects to build a SWAT+ model setup. The spatial objects are organized in GIS vector layers, where the different spatial objects require to be of a specific geometry type and must have certain attributes. Land objects (including open water surfaces) for example must be in a polygon format and must at least have a land use type defined, while channel objects must be defined as line geometries, with the attribute whether a channel is a surface channel or an object that transfers water underground.

Besides the spatial location and the extent of spatial objects a `SWATbuildR` model setup requires a Digital Elevation Model (DEM) and soil information to derive terrain and soil attributes for the spatial objects. The DEM and the soil layer must be provided as raster layers in the model setup process.

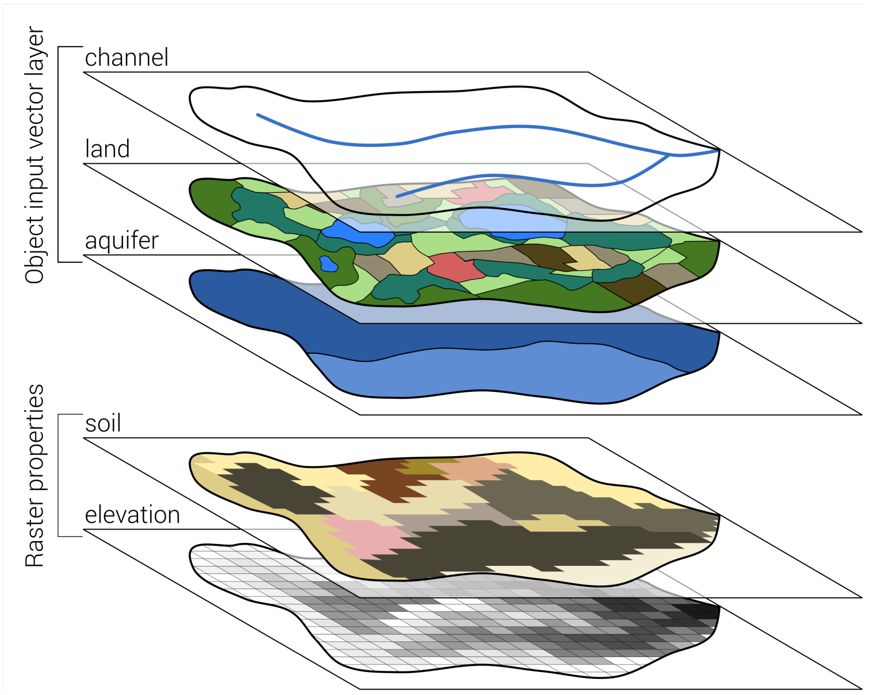


Figure 2.1: Required vector and raster inputs for a SWAT+ model setup with `SWATbuildR`.

This section covers the input data required for setting up a SWAT+ model with `SWATbuildR`. It defines the general required data structure of the model input layers and provides some insight in the data preparation and potential issues which may arise in the input data preparation. The section is organised in a way to first describe the required raster inputs DEM and soil, continued by all vector object layers, which are the basin boundary, land objects, channel and aquifer objects.

The model setup process with the `SWATbuildR` focuses on the connections between spatial objects in a SWAT+ model setup. The setup process only partly parameterizes spatial objects and adds only a few spatial properties which can be derived from the DEM and soil layers. A final `SWATbuildR` setup is written into a *'sqlite'* database which can be further edited with the SWAT+Editor. Thus, any further definition of a model setup is done with the SWAT+Editor. The editing of `SWATbuildR` setups is discussed in the section on parameter customization.

2.2.1 Digital Elevation Model (DEM)

A DEM is relevant to derive spatial attributes for the spatial objects of a SWAT+ model setup. In the `SWATbuildR` setup process the terrain is further highly relevant to calculate the connections and flow fractions between land objects. An adequate representation of the terrain in a catchment requires to the use of a high-resolution DEM product.

Many national authorities in the EU (Kakoulaki et al., 2021) provide LIDAR-based DEM products with high spatial resolutions (e.g. 1 or 2 m). A high spatial resolution can be computationally expensive in processing the terrain information. Unless the processing of the terrain data at its original resolution is infeasible the original spatial resolution should be kept.

`SWATbuildR` was conceptualised to represent landscape characteristics in SWAT+ model setups at the field plot scale. Thus, the interaction of features in the landscape should also be represented in the terrain properties. Features, such as road dams and similar structures are hydrologically effective as they separate the landscape and can substantially control the runoff processes. Eventually, the DEM must be capable of representing the hydrologically effective landscape features on the field plot scale. A resampling of the DEM may result in a substantial information loss and features such as road dams may not be well presented in the resampled DEM anymore. Figure 2.2 shows the same detail of a Light Detection and Ranging (LIDAR)-based DEM with its original spatial resolution of 2 m (left) and resampled to 10 m (right). The original DEM product shows hydrologically effective road dams which would guide the flow from adjacent areas along the road dam in drainage channels and would prevent flow from passing the dam. While the larger structures are still present in the resampled DEM, the effect of less pronounced structures might get lost when a DEM with a coarser resolution would be used in the model setup process.

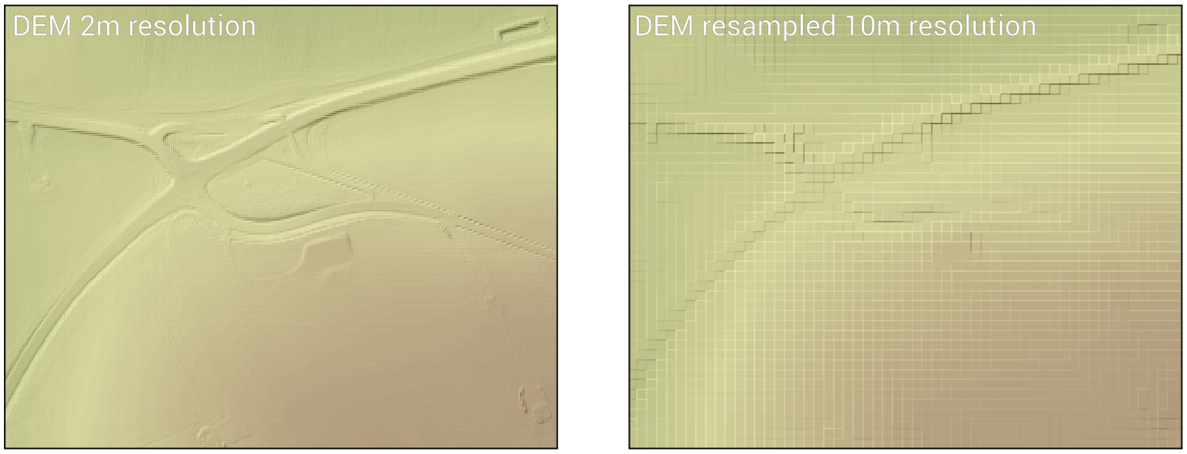


Figure 2.2: Comparison of the representation of hydrologically effective landscape features with different DEM resolutions. Left a LIDAR-based DEM product with its original resolution of 2 m. Right the DEM resampled to a spatial resolution of 10 m.

As a guidance for the user of `SWATbuildR` it is recommended to preferably use DEM products with high spatial resolutions of 1 to 5m. If no other products are available, the spatial resolution of the DEM may not exceed 10m.

2.2.2 Soil data

`SWATbuildR`requires the same soil data input data that is required for a SWAT model setup with other widely used model setup tools, such as ArcSWAT, QSWAT, or QSWAT+. The soil data comprises of (1) a *'GeoTiff'* raster layer that defines the spatial locations of the soil classes (with integer ID values), (2) a lookup table in *'csv'* format, which links the IDs of the soil classes with their names, and a (3) *usersoil* table in *'csv'* format, which provides physical and chemical parameters of all soil layers for each soil class that is defined in the raster layer and the lookup table. An example soil dataset can be acquired e.g. from the *'ExampleDatasets'* folder which comes with an installation of SWAT+.

Additionally, soil chemical data to initialize soil nutrient modelling could be used as input data to SWAT+. Especially important is the soil phosphorus content, because processes governing phosphorus movement are sensitive to initial values. Soil physical data and soil chemical data are described in the Model parametrization chapter of this protocol. This section focuses only on the GIS map with soil classes.

Similar to the DEM, the soil data must represent the spatial heterogeneity of soil classes at the field plot scale. Figure 2.3 a) gives an example of soil data which has a comparable spatial representation to the scale of the spatial objects. The soil properties control the partitioning into different runoff fractions and the transport processes of nutrients. Coarse soil information and consequently low heterogeneity in the soil properties may not represent the impacts of the soil on the hydrological processes well. Thus, the user should aim to implement detailed soil datasets (available as national level) with an adequate spatial representation in the model setup with `SWATbuildR`.

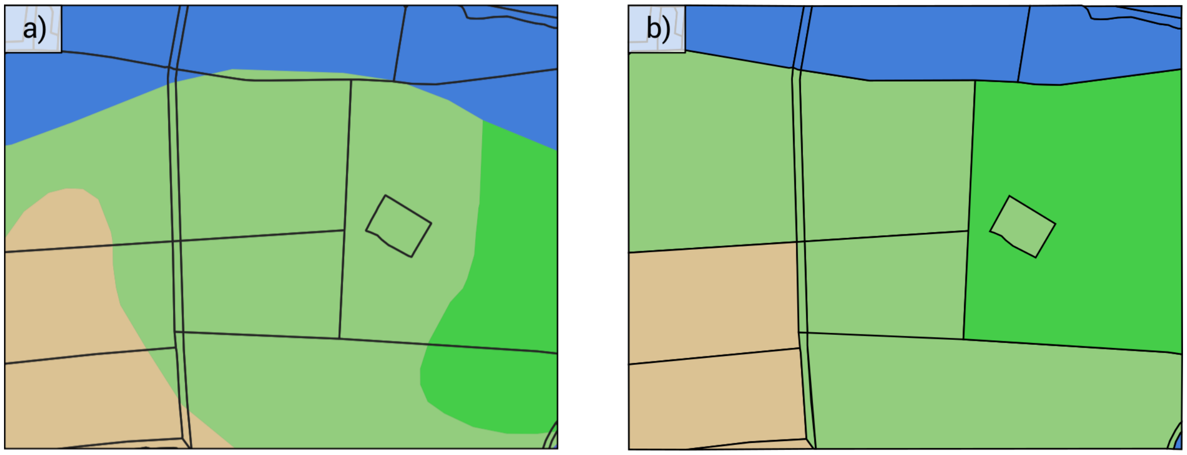


Figure 2.3: Spatially adequate representation of soil heterogeneity on field plot scale. The different colours represent different soil classes. The black border lines represent the boundaries of spatial objects (e.g. agricultural fields). a) the original soil input layer. b) the dominant soil aggregation as performed by `SWATbuildR`.

While working with local soil datasets, one of the issues which might be encountered is that national soil databases might only cover agricultural land, while lacking soil information for urban and forest areas. In some countries forest services may provide access to forest soil maps that can be merged with the agricultural soil maps. If no better data is available, such gaps can be filled by using global (e.g. SoilGrids, or other dataset available from ISRIC - World Soil Information) or continental (e.g. European Soil Database, etc.) datasets. Another approach is to use proxy data to obtain soil types. For instance such proxy data are forest plant habitat datasets, which could be used to derive soil types.

National or regional soil maps are often available with different spatial resolutions. In general, for model application in small agricultural catchments (several km^2 to a few hundred km^2), more detailed maps, corresponding to scale of at least 1: 100 000 would be advised. The use of soil information on a scale which is much more detailed than the scale of the spatial objects in a model setup is however not recommended. It might be difficult to provide parameters to the *usersoil* table for less common soil types. Moreover, by default `SWATbuildR` uses the dominant soil class for each spatial object in the model setup process (see Figure 2.3b). Thus, a great share of the detailed information would get lost during the model setup process. One exception from that rule may be soil physical data which are available in a raster format. For such soil data averaging the parameters for each spatial land object may be an appropriate approach. Furthermore, in cases when the soil map is too detailed, it might make sense to carry out reclassification of the raw soil map before loading it to the `SWATbuildR` in order to aggregate soil units into larger objects based on their characteristics. If there are too many unique soil classes in the soil map, firstly, many of them may actually have similar properties (so they can be clustered) and secondly, it may be difficult to identify parameters for some rare soil types or for soil types with missing characteristics (see more in Soil physical data section).

2.2.3 Watershed boundary

The catchment boundary of the watershed that is modelled is a required input for `SWATbuildR`. The watershed boundary must be available as a single polygon GIS vector layer. The current version of `SWATbuildR` only handles single catchments and therefore checks that one feature is provided with the boundary layer. No attributes are required for this single feature. The boundary layer will be used to define the `SWATbuildR` project coordinate reference system and to crop all other GIS layers to this boundary.

One source for the watershed boundary to be used in a `SWATbuildR` project can be water authorities or environmental agencies. In some situations the study area may however not be covered by any "official" dataset, because it is not represented as a watershed in national datasets due to its small size or the study area's catchment outlet does not match with the official data. If this is the case, the basin boundary can also be delineated from the DEM. GIS provide hydrological toolboxes to perform watershed delineation. QSWAT+ for example provides a full workflow for watershed delineation in QGIS which generates a basin boundary that can be further used in the `SWATbuildR` model setup process. ArcGIS Hydrology toolset or QGIS SAGA - Terrain Analysis: Hydrology could be used for this task as well.

Several checks will be performed for the basin boundary input layer during the `SWATbuildR` model setup process. The input layer checks are documented in section 2.3.3.1 in more detail. The most critical layer property for the basin boundary with respect to the input layer checks is the single polygon property. This can particularly cause issues for basin boundaries which were generated by a GIS-based watershed delineation. Single pixels of the underlying DEM layer can result in a crossing of the basin boundary in vertices (see darker red pixel in Figure 2.4). This results however in a multi-polygon layer and for further processing such single "pixels" must be removed.

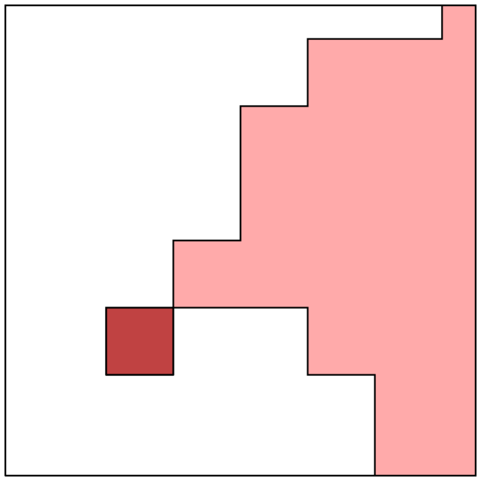
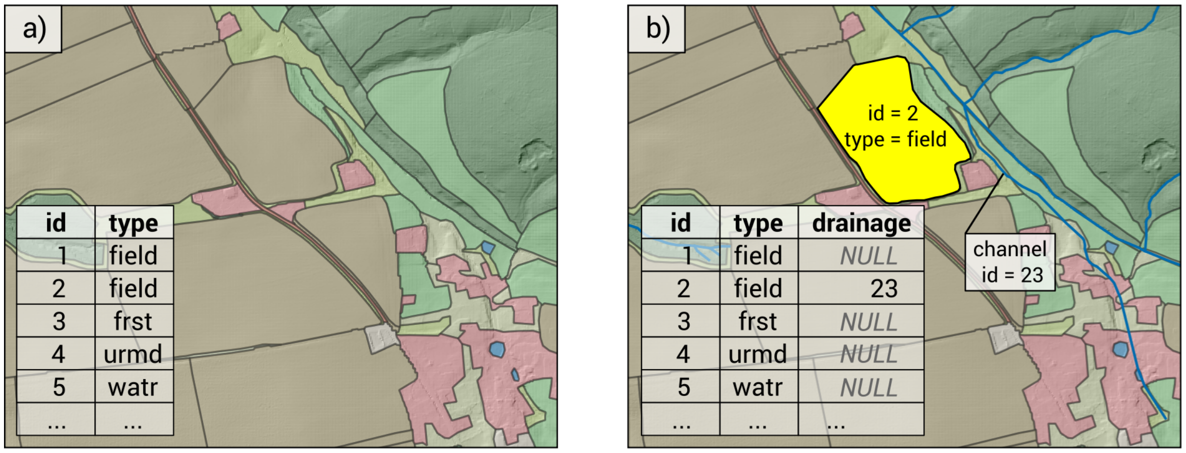


Figure 2.4: Detail of a basin boundary layer that was generated by a GIS-based watershed delineation. The darker red polygon results from a single pixel of the DEM. It should be removed from the basin boundary layer for further processing.

2.2.4 Land object input

The land object input layer defines all surfaces in the landscape of a watershed, including all existing land surfaces and standing water bodies as well as existing and/or planned (in modelling scenario runs) NSWRMs placements. The input layer must comprise all individual spatial units in the landscape for which the water balance components and the interaction with other spatial objects are simulated. Eventually, the land layer must be a seamless contiguous representation of the landscape.

The land layer must be provided as a GIS polygon vector layer. Each feature that is defined by the input layer will form an individual spatial object in the final SWAT+ model setup. Each feature requires to have the attributes `id` which must be a unique ID number as well as a `type` column that defines the land use of a spatial unit (see Figure 2.5 a) for an example. `drainage` column is optional, if tile drainage is important for hydrological processes in a catchment. `id` of channel receiving tile drain water from a field should be provided here.



id	type
1	field
2	field
3	frst
4	urmd
5	watr
...	...

id	type	drainage
1	field	*NULL*
2	field	23
3	frst	*NULL*
4	urmd	*NULL*
5	watr	*NULL*
...	...	...

Figure 2.5: Land object polygons with attribute table. a) shows a land layer without tile drainage implemented. b) shows a case where the agricultural field with the land `id = 2` has tile drainage activated and drains the tile flow into the channel `id = 23`.

Besides others, OPTAIN investigates hydrological processes and nutrient transport from field plots and the interaction with other landscape features such as hedges. The project analyses for instance in scenarios the impact of the implementation of NSWRMs on water and nutrient retention. The delineation of spatial units in the landscape by pre-processing defines the spatial scale of the SWAT+ model setup. The individually modelled land objects must therefore represent landscape features on the field plot scale. Further, a model setup must already include land objects which can be transformed to objects that simulate NSWRMs.

2.2.4.1 Data sources

Considering the spatial scale of landscape representation in OPTAIN, the use of for instance continental or global scale data might not be suitable. Thus, national or locally available information on land use is mandatory. In some countries such data might be dispersed across different institutions. For instance, a detailed crop dataset could be located under the institution responsible for farmer subsidies, forest - under the institution responsible for national forest management, urban - under institution for urban planning, water - under water management institutions, etc.

If land use data sources are from different national/local institutions, they have to be combined into one layer (usually applying GIS `union` functions). Procedures need to be agreed on how to deal with overlapping areas (by two or more data sets) or gaps (areas not covered by selected local data sets). Moreover, sliver polygons (very small polygons in vector data) and various topographical errors could be generated by joining different data sets. Thus, it is important to check and clean such errors before proceeding further with data preparation.

2.2.4.2 Delineation of land objects

Land use is usually not static and can change over time. While changes in major water surfaces, forest, and urban areas follow slower trends and could be assumed constant for modelling purposes, agricultural field plots might be split differently in every growing season. Thus, the delineation of the individual land objects must find a good compromise between the representation of relevant landscape features and accounting for changes in the landscape.

The delineated polygons for the land objects are a static feature in a `SWATbuildR` model setup. Therefore, all features in the landscape which should be considered as individual entities in the landscape or which develop into individual features (e.g. changing land use or the implementation of a structural NSWRM measure in OPTAIN) during the simulation period must be delineated as individual units in a model setup.

For agricultural field plots, which may have slightly changing field boundaries every year, this compromise in their representation is met in OPTAIN by using a separation into individual fields in one year where data is available (or developing a compromise for several years of data) and keeping those field boundaries constant in the land input layer. As land use change is not the major focus in OPTAIN, the land unit boundaries for other land uses such as urban areas, open water surfaces, or forest land are static in the land layer.

The land layer example in Figure 2.5 illustrates the degree of abstraction in the land object delineation which was aimed for in OPTAIN. The scale of interest in this case is the field plot scale. In Figure 2.5 agricultural field plots are individual polygon features. For areas with land uses other than agriculture the aim is to generate units at a similar spatial scale. This determines how abstract or complex land uses such as urban areas should be represented. In urban areas, for example, diverse land uses, such as sealed surfaces (e.g. roads), buildings or green areas must be grouped to blocks which comprise heterogeneous areas in an urban land use class. Also complex patterns of different forest types may be grouped into one type of forest to reduce the number of land objects - given that this grouping does not affect any of the modelling goals.

Homogeneous, large forests can however also form large, single land units. To maintain land units of comparable size, such large land objects must be split into several units. Maintaining similar sizes for all spatial units and avoiding land units in a model setup can be crucial in the routing of water between spatial objects. The routing of large amounts of water from large land objects into small land units can cause issues in the calculation of the units' water balances and may result in implausible simulations of hydrological processes. A practical approach to split large forests into smaller units is to delineate local catchments in a forest and subdivide a forest into its local catchments (see Figure 2.6 a) as an example). Another approach could be to split a forest based on features such as forest roads (similar to the grouping of urban areas) to form blocks of forest land which eventually result in areas comparable to field plots in size.

COCOA connects neighbouring land objects based on topography. With this approach long land objects such as roads or long tree hedges can be particularly problematic, because they can cause unrealistic water transfers in the landscape. For example if a long road polygon crosses an entire simulated catchment, while receiving water from land objects at one end of the road polygon and this road polygon routes water to other land objects at its other end. This is unlikely to be realistic for

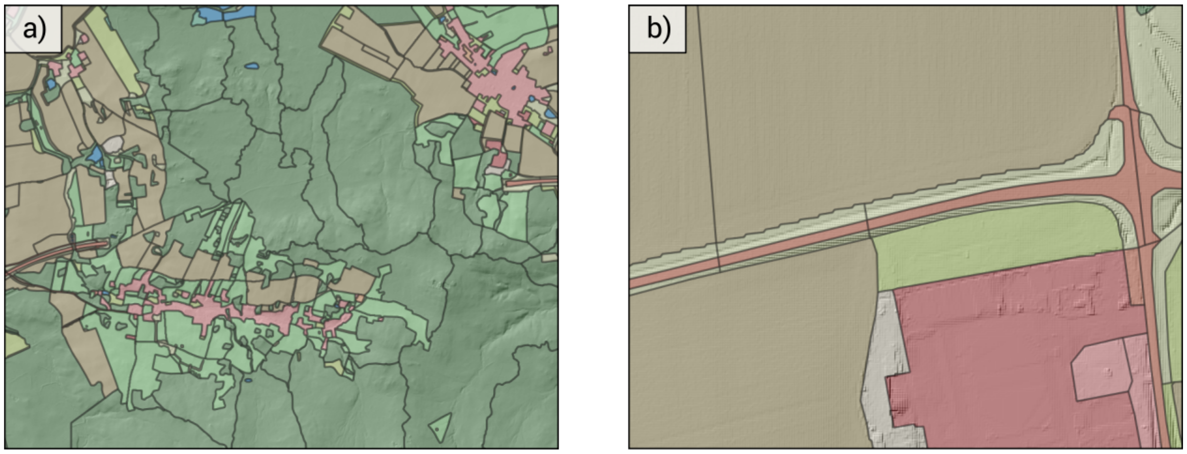


Figure 2.6: Details of a land object layer. a) shows the division of a large forest (dark green area) into smaller units based on local catchments. b) shows the splitting of long objects e.g. roads into shorter segments based on borders of neighbouring units.

water to be transferred across the entire catchment. Such issue must be considered in the delineation of the land objects and long features such as roads must be split into shorter segments. To minimise the effect of unrealistic water transfers, splitting such long land objects at edge points of other polygons is considered good practice (see Figure 2.6b) for an example).

To obtain SWAT+ models with acceptable run time (<15 min for a simulation period of 10-15 years), the final land use vector layer should preferably not consist of more than 5,000 - 8,000 polygons. Limiting the number of land objects gives some reference for the degree of abstraction for the implementation of landscape features in the land input layer (some strategies for the delineation were mentioned above). A final effective strategy can be to merge small polygons into larger neighbouring polygons or grouping them into more general land use groups (see e.g. urban land use), where the individual effect of a land use is not relevant to the overall modelling task. Considering the calculation of the water routing between spatial objects, delineated land polygons should not be smaller than two to five times the DEM pixel size. Smaller objects may even not be represented in the final model setup, but may increase the time required for the model setup process.

An important issue of land use is that it is not constant. Water, forest, and urban areas might not change much in area between years and could be assumed constant for modelling purposes (although it is wise to check long term trends in these categories). However, cropland might be changing every year and in some cases even more than once per year. Thus, it is important, if possible, to collect field-based crop maps for all years within the modelling period. When such data is not available, remote sensing methods (see section 4.1.1) could be applied to generate it. Even though eventually a crop map representing a single year could be used for setting up the model, full crop datasets are important later on for building realistic rotation patterns in management schedules or decision tables.

In contrast to the QSWAT+ based setup, in which a conversion of a vector to a raster land use map would be typically needed, in the COCOA approach field boundary GIS data (in vector file) have to be obtained and/or prepared. As fields would become the smallest modelling unit (instead of HRUs), it is important that they represent areas of the same crops with as little variation in soil types and slope as possible. In case of large fields of the same crop with large variation of soils and/or slope, splitting the fields into smaller parts should be considered.

2.2.4.3 Standing water bodies and the land use `type = 'watr'`

In contrast to the model setup process with QSWAT+, surface water objects (reservoirs/ponds/lakes) are not provided in a separate layer. In the COCOA concept the land use layer is a contiguous representation of the landscape, which also includes standing water surfaces. Although the `type` attribute for land surfaces can have any label, water surfaces require to have the specific label *'watr'* for their attribute `type`. This is necessary as all standing water surfaces will eventually become part of the water objects in the `SWATbuildR` model setup and will be reservoir objects in the final SWAT+ model setup. Standing water bodies with `type = 'watr'` can be part of the water object network and can therefore be directly connected to channels (which is the case when a water surface covers a channel of the channel layer or if channels start or end in a land object with `type = 'watr'`). Standing water bodies can however also be water surfaces which are not connected to the channel network. In such a case it will become a reservoir in the final SWAT+ model setup, which can receive water from land objects, but only emits water through transpiration.

2.2.4.4 Tile drainage option

By default no tile drainage is considered for land objects in the model setup process with `SWATbuildR`. Depending on topographic conditions and soil properties soils might be drained with existing tile drains systems, which has to be considered in the SWAT+ model setup. By default a land object routes all runoff fractions (surface flow, lateral flow and tile flow which is zero) into the neighbouring land and water objects. If tile drain is activated for a land object it specifically routes surface and lateral flow into the neighbouring objects, but routes its tile flow into a defined channel.

Tile drainage maps can usually be obtained from either water authorities, environmental agencies or municipalities. These could be polygon features with fields equipped with tile drains or line features with the drainage pipe network. There are methods for identification of tile drainage from remote sensing products (Gökkaya et al., 2017), but their use requires time and expertise.

Tile flow for a specific land object into a channel object is defined with the attribute `drainage` in the land object layer. Figure 2.5 shows two different configurations of the same land object layer. While in Figure 2.5a no tile drainage is considered for any land object (absent `drainage` attribute), the attribute table in Figure 2.5b includes the column drainage. For most land objects the attribute `drainage = NULL`, which translates to a deactivated tile drainage option. For the agricultural field with the `id = 2` the `drainage` attribute is set to `23`. This means that the tile flow from this field is routed into the channel with the `id = 23`. If no data on the drainage pipe network that would enable a correct attribution of drained fields to channels are available, then a compromise solution may be to identify the nearest channel from each drained field.

The land layer only considers the connectivity of tile drained land units and does not require any information on the parameters of the tile drain network. All tile drained land units are parameterized with a default configuration of the tile drain network. The tile drain parametrization must be adjusted by the user at a later step in the model setup process. Tile drain parameters will be further discussed in section 3.9.

2.2.4.5 Existing and potential structures to retain water and nutrients

The main focus area of OPTAIN is to identify efficient techniques for the retention and reuse of water and nutrients, and select NSWRMs at farm and catchment level and optimise their spatial allocation and combination based on environmental and economic sustainability indicators. Measures related to agricultural management, such as cover crops or reduced tillage, can for example be modelled by directly changing management schedules or by triggering certain operations with SWAT+ decision tables. The implementation of such an agricultural measure does not structurally affect a SWAT+ model setup. Thus, their later implementation in scenarios only requires to make changes in the management of the respective land objects.

Structural NSWRMs, such as riparian buffer strips, hedges, grassed waterways, constructed wetlands, two-staged ditches and retention ponds however, must be considered as individual objects in a model setup. Existing landscape features must be defined as individual polygons in the land input layer (e.g. riparian buffers with green polygons in Figure 2.7). Potential locations for structural NSWRMs must also be delineated as individual land units. These units receive the initial land cover within the defined unit (e.g. the orange, yellow, and purple polygons in Figure 2.7). The implementation of a structural measure in a scenario case can for example be performed by changing the land use (e.g. transformation of an erosion prone zone of an agricultural plot to a grassed waterway) or by changing the land object to a reservoir object and adjust the routing (e.g. for the implementation of a retention pond).

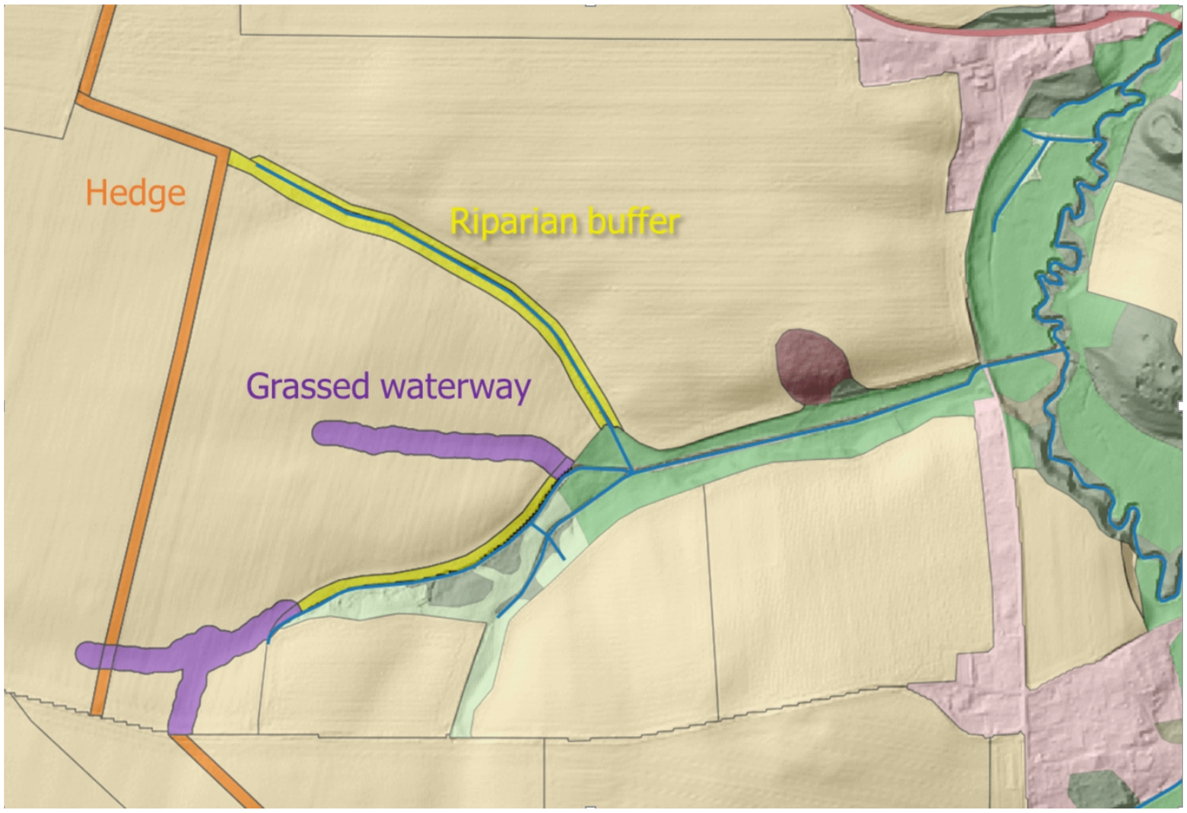


Figure 2.7: Land input layer with potential locations for structural measures highlighted. Polygons were defined to transform parts of agricultural fields into hedges (orange), riparian buffers (yellow), or grassed waterways (purple). Existing riparian buffers are delineated with green polygons.

The allocation of structural measures should be justified by site-specific criteria (e.g. structural measures addressing soil erosion only on vulnerable sites within cropland with certain soil properties and topography). The potential sites for implementing new measures or - at least - the methodology to map these sites should be confirmed by the local stakeholder group of an OPTAIN case study (see also 2nd MARG meeting – guidelines[1]).

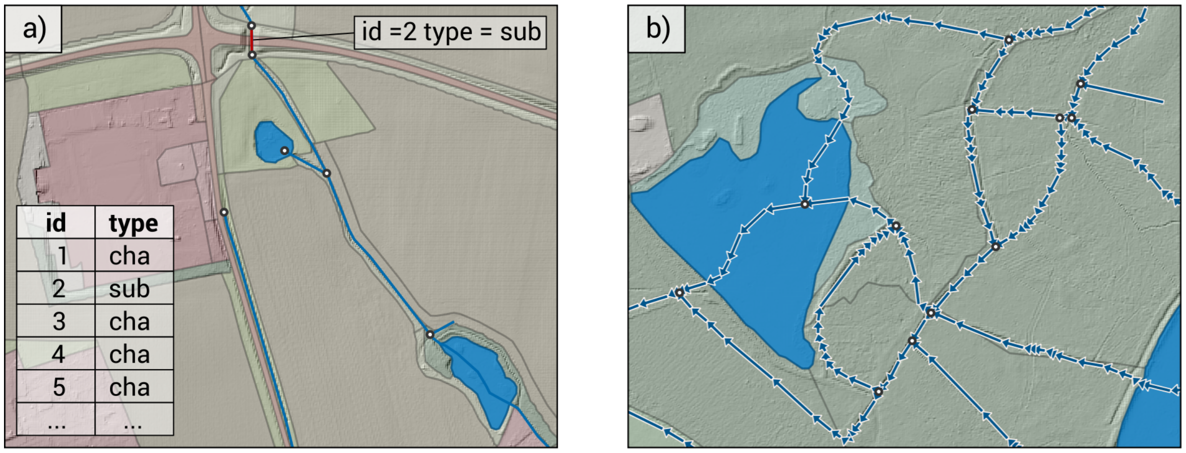
The OPTAIN deliverable D2.3 (Marval et al., 2022) includes a comprehensive collection of structural

[1]The link is only accessible by the OPTAIN consortium partners.

NSWRMs and agricultural management related measures. The document provides guidance for the implementation of NSWRMs in the landscape and documents the implementation of measures in SWAT+ model setups.

2.2.5 Channel object input

The channel object input layer defines the network of all surface (open water courses) and subsurface (e.g., pipes, underpasses) waterways. The channel object input must be provided as a GIS line vector layer. Each feature that is defined by the input layer will form an individual spatial object in the final SWAT+ model setup. Each feature requires to have the attributes `id`, which must be a unique ID number, as well as `type` that defines the type of channel. `type` attribute has two options, where `type = cha` defines a surface channel and `type = sub` defines a subsurface water course (see Figure 2.8 a) for an example). The major difference between these two channel types is that a surface channel can receive water from neighbouring land and water objects, while `type = sub` channels only connect with other water objects (thus emulating subsurface pipes which do not receive water from the land surface). Together with the standing water bodies (land objects of `type = watr`) the channel network forms the water object network in the `SWATbuildR` model setup process. While reservoirs can be unconnected, each channel must at least be connected to one other channel or to a reservoir. In the current version of `SWATbuildR` the number of channels that can be input in a model setup is limited to 9999 channel objects.



id	type
1	cha
2	sub
3	cha
4	cha
5	cha
...	...

Figure 2.8: Examples for channel networks. a) shows a part of a channel network with its attribute table. The object with `id = 2` in this example is a road underpass and is therefore of `type = sub` (subsurface water course). b) shows a more complex case of a channel network with natural and artificial surface channels, which are connected with ponds. The visualization of channels as arrows can help to visually validate a channel network.

To develop a correct flow routing between the water objects in the model setup process, the directions of the channels are relevant. The direction of a line segment is defined by the order of setting the line nodes. Channel segments must always end in the start points of new channel segments to correctly link two channels. In the model setup process the correct linking between water objects is substantially tested and will be explained with more detail in section 2.3.3.3. A visualization of channels as arrows is helpful to visually validate the directions of channel segments in a channel network to check if the flow would be routed through the water objects in the correct sequence. Figure 2.8b) shows a more complex example of a channel network, which includes natural and artificial water courses. Figure 2.8b) also shows the water objects from the land layer, which will be assembled with the channel network to form the water object network. In such a complex case the arrow visualization greatly supports the visual validation of the channel network. It shows that in each node (grey/white circles) at least one channel ends and points into the node and at least one channel starts (pointing away from the node).

The visualization of line segments as arrows should be available in practically every GIS. In QGIS for example this visualization type can by activated in `Properties... > Symbology > Symbol layer type > Arrow`.

2.2.5.1 Data sources and data processing

Stream network data are usually available on national levels from water management authorities or environmental agencies. Yet, national stream networks may not cover all channels, which should be included in model setups at such a high spatial resolution. Small headwater streams, high order streams, drainage ditches, or roadside drainage channels may be missing from stream network data sets. If available from other authorities such data must be included in the channel network. If further data is not available it may be necessary to manually add channels which affect the runoff processes. Road dams, for example, form barriers for surface flow and are barriers for the connectivity between spatial objects.

Usually areas along road dams are drained by ditches along the dams. In a COCOA model setup not including road dams can strongly affect the hydrology of the simulated catchment. Figure 2.9 shows the area around the road underpass which was illustrated in Figure 2.8a. Figure 2.9 now includes the flow accumulation based on the topography of this area. From the flow accumulation it is apparent that flow would accumulate along the road dam and it is very likely that there is a road ditch which is missing in the channel network. To better represent the flow routing these ditches should be added manually.

The Figures 2.8 and 2.9 also illustrate how channel segments should be separated. In order to keep the number of channel objects in the final model setup low an unnecessary splitting of channel segments should be avoided. A splitting into individual channels is required at confluence points with other channels or at points where in-stream observation data is available. If a node in the channel network is missing at the location of observations no simulation outputs would be written for that location and the observation data can only be compared to other locations. A channel split at intersections with standing water bodies is not necessary (channels cross ponds in Figures 2.8 for example). The intersection of channels and reservoirs is performed in the model setup process automatically.

2.2.6 Point sources locations

Point sources are in general municipal or industrial wastewater treatment plants that discharge treated sewage into the stream network. For the `SWATbuildR`, only location data of point sources is required. All other relevant parameters such as the volume and chemical characteristics of discharged water belong to the section on point sources. It is worth checking if acquired locations intersect or are within close proximity of the existing stream network (pipe location is required, not the address of the treatment plant).

2.2.7 Aquifer objects

Although SWAT+ would allow the implementation of multiple aquifers in a model setup, the current version of `SWATbuildR` only adds a single aquifer for the entire catchment to the model setup. All land

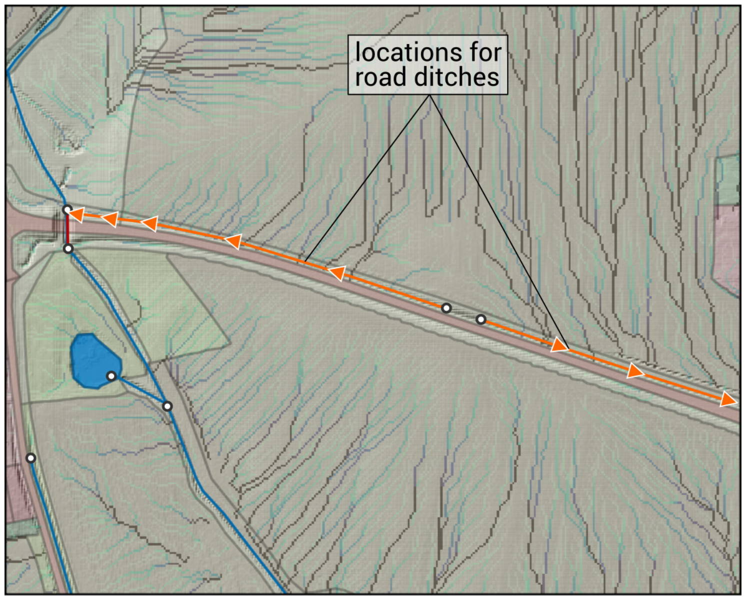


Figure 2.9: Example for a location in a catchment where flow would accumulate along a road dam based on the flow accumulation, but a road ditch is missing and should be added (orange lines).

and water objects will be linked to this single aquifer. As the sizes of the OPTAIN study sites range between approximately 20 to 250 km^2 the use of a single aquifer can be justified.

2.3 Model setup with `SWATbuildR`

`SWATbuildR` is a tool to generate SWAT+ model setups following the contiguous object connectivity approach (COCOA). In the model setup process with `SWATbuildR` the prepared land and channel object input layers and the DEM and soil raster layers are combined to generate the spatial objects of a SWAT+ model setup and to calculate the connections and flow fractions between the generated spatial objects. The generated SWAT+ input files are stored in an *'sqlite'* data base which is readable and therefore further editable with the SWAT+Editor. `SWATbuildR` was developed and written in the script programming language `R` and is currently available in a script version.

The `SWATbuildR` script version consists of an RStudio project (`buildr_script.Rproj`) with three `R` script files.

The following section goes through all steps of the `SWATbuildR` script to set up a SWAT+ model implementing COCOA. The `SWATbuildR` script version is currently under development and is tested internally by the OPTAIN case study groups using the `SWATbuildR` version 1.0.3. The text below refers to the functionality of the 1.0.3. This current version and future updated versions of `SWATbuildR` and a demo input data set to follow the model setup steps will be made available in a public repository after its test phase.

2.3.1 The `settings.R` and `function.R` inputs

The `SWATbuildR` script version should be started by opening the `buildr_script.Rproj` in RStudio. At the beginning of the main script `build_cocoa.R` two additional scripts are sourced which are called `settings.R` and `function.R`. The scripts are defined with relative paths, so the user needs to make sure that the script `build_cocoa.R` is in the same project folder with `settings.R` and `function.R`.

```r
# Load paths and parameter settings ---------------------------------
source('./settings.R')
# Load R packages and functions -------------------------------------
source('./functions.R')
```

The file `functions.R` collects all functions and subroutines, which are called and used in the main model builder script `build_cocoa.R`. It also includes a routine to install and load all required `R` packages. The functions in `functions.R` should not be modified by the user. They can however be useful to look into for debugging.

`settings.R` defines paths and settings which are required for the model setup. The adjustments which are shown in the code boxes below need to be done in the file `settings.R`. The first definition `project_path` is the path where the new `SWATbuildR` project (defined with `project_name`) should be generated and the name of the new project. On Windows computers the user needs to be aware to use front slashes '/' and no back slashes ('\') to define a path, as back slashes do have specific functions as control characters in `R`.

```r
project_path <- 'Define:/your/new/path'
project_name <- 'demo_project'
```

Next the settings define where the DEM and soil raster layers and the SWAT format soil data csv files are located on the hard drive.

```r
## DEM raster layer path
dem_path <- './test_data/dem/dem_j.tif'
##Soil raster layer and soil tables paths
soil_layer_path  <- './test_data/soil/soil_j.tif'
soil_lookup_path <- './test_data/soil/soil_lookup.csv'
soil_data_path   <- './test_data/soil/usersoil_lrew.csv'
```

To set up a COCOA model setup at least vector layer inputs of the basin boundary and the land layer (which includes optional reservoirs defined by the attribute `type = watr`) are required. These paths to these two inputs are defined with `bound_path` and `land_path`, respectively. A channel layer is an optional input if channels are present in the simulated catchment (which may be the default case). If no channels should be included in the model setup set `channel_path <- NULL`.

```r
## Catchment boundary vector layer path, all layers will be masked
## by the basin boundary
bound_path <- './test_data/basin/bsn_boundary_dem_10m.shp'
## Land input vector layer path
land_path <- './test_data/land/lulc_j.shp'
# land_path <- './test_data/land/lulc_drain.shp'
## Channel input vector layer path
channel_path <- './test_data/channel/streams_j_nhd_edit.shp'
```

A few additional settings must be defined to run the model setup procedure. The first input argument is `project_layer`, which controls whether input layers should be reprojected to the same coordinate reference system (CRS) as the basin boundaries CRS (which eventually be the `SWATbuildR` project's reference system).

By default `project_layer` is set `TRUE` and all input layers will be reprojected. This may be OK in most cases, but a visual validation of the projected input layers is strongly recommended. If this option should be deactivated in order to ensure that all inputs are in the same reference system at the time of reading the data, `project_layer` should be set `FALSE`.

In the next step, the main outlet of the catchment must be defined. The main outlet can be either one channel or a single reservoir. The main outlet is defined with one of the two input arguments `id_cha_out` and `id_res_out`. Only one `id_*` should be defined while the second argument must be set to `NULL`. In the example below the channel with the channel `id = 10` is defined as the catchment outlet and `id_res_out` is therefore set `NULL`. The id values used to define the main outlet must match the `id` of the channel or reservoir in the channel or land input layer.

```
id_cha_out <- 10

id_res_out <- NULL
```

The calculation of connections between spatial objects can for single objects result in many connections to other objects with small flow fractions. Small fractions of flow between spatial objects may however not be relevant and it is preferable to only keep the few relevant connections of an object in the final model setup. For each spatial object the connections to other objects will be evaluated in a model setup step. The largest flow fraction of that spatial object will be assigned a value 1 and all other flow fractions will be normalised with respect to the largest flow fraction. A connection to other objects will be removed if the flow fraction relative to the largest flow fraction of that spatial object is lower than the value defined with `frc_thres`. Default `frc_thrs` is set to

```
frc_thres <- 0.3
```

The model setup process saves each step of the model setup in the projects' `data_path`. The saved data is organised in a way that all raster data is saved in *'GeoTiff'* format in *'./data/raster'*, the vector data is saved in ESRI shape file format in the folder *'./data/vector'* and all tabular data will be saved in the database *'./data/tables.sqlite'*. As the results of each step in the model setup is saved, it is possible to continue the model setup process at any point in the script.

2.3.2 Routines for checking and preparing the vector input data

All spatial object input vector layers and the DEM and soil raster layers are read and thoroughly checked before they are used in the model setup process. In the following the routines to check the polygon and line vector layers are explained in more detail. A better insight in what the actual checks analyse can be helpful to correct the identified issues.

2.3.2.1 Topological checks for polygon layers with `check_polygon_topology()`

After reading the polygon input layers `bound`, which is the basin boundary, and `land`, which is the land object input layer, a sequence of topological checks is performed with the polygon features of the input layers. Both input layers are checked with the function `check_polygon_topology()`. The function has the three required input arguments `shp`, `vct_path`, and `label`. `shp` is the shape file, which was loaded as an `sf` (i.e. simple features, a data.frame with spatial properties) object into `R` that should be checked by the function. `data_path` is the path to the `SWATbuildR` project's data folder. As for the two polygon layers `bound` and `land` different checks might be relevant some additional optional input arguments are available. If the number of features should be checked the input argument `n_feat` must be defined. For the basin boundary for example it is important that it consists of only one polygon feature and thus `n_feat = 1` must be set. `area_fct` is a factor which is multiplied with the Q_{25} quantile value of the areas of land polygons. The routine checks whether areas can be identified with areas smaller than $area\_fct \cdot Q_{25,area}$. The `cvrg_frc` is used as a threshold to check whether the sum of polygon area of a layer is above a fraction of the catchment area. With the input argument `checks` eight different input layer checks and operations applied to the vector layer can be activated.

Below is an example on how to check the loaded `land` layer. In this example the check function would raise an issue if polygons with an area smaller than $0.05 \cdot Q_{25,area}$ are present and if the coverage of the land polygons is lower than 99.9% of the basin defined in the `bound` layer. With `checks` all checks/vector operations but the second one (which is the number of features check) are activated for the land layer. In the following all eight checks/operation are described and supported with examples of identified issues.

```
check_polygon_topology(shp = land, data_path = data_path,
                       label = 'land',
                       area_fct =  0.05, cvrg_frc = 99.9,
                       checks = c(T,F,T,T,T,T,T,T))
```

`checks[1]` Intersection with the basin boundary: The first option in `checks` controls whether an intersection with the basin boundary should be performed before other operations and checks are executed. For layers except the basin boundary itself this operation should be performed.

`checks[2]` Check number of features: The second option controls whether the number of features of the input layer should be compared and should match the value defined with `n_feat`. This option is relevant for the basin boundary layer `bound`, which must have exactly `n_feat = 1` polygon feature which defines the simulated catchment.

`checks[3]` Check for *MULTIPOLYGON* features: All features of a polygon input layer must be of type *POLYGON*. Layer preprocessing, but also the intersection with the basin boundary can result in a conversion of polygons into *MULTIPOLYGON* features. The example below (Figure 2.10) shows polygons of a land layer (grey polygons) together with the basin boundary (black line) of the catchment after the intersection (intersected parts of the polygons in red and orange colours). The two details show that because of the shape of the basin boundary and the land polygons the polygons were split into one large part and several smaller parts. The details in Figure 2.10) should emphasise that intersecting might result in very small fragments and thus they are sometimes difficult to identify.

If *MULTIPOLYGON* features are identified in the check routine of a polygon vector layer, these issues have to be resolved in the original input layer and the topological checks have to be repeated before continuing with the `SWATbuildR` script. The performance of following operations applied to an input layer before loading it can reduce the risk of having *MULTIPOLYGON* features:

- Intersect the input polygon layer with the basin boundary. In QGIS the processing tool `Intersection` can be used

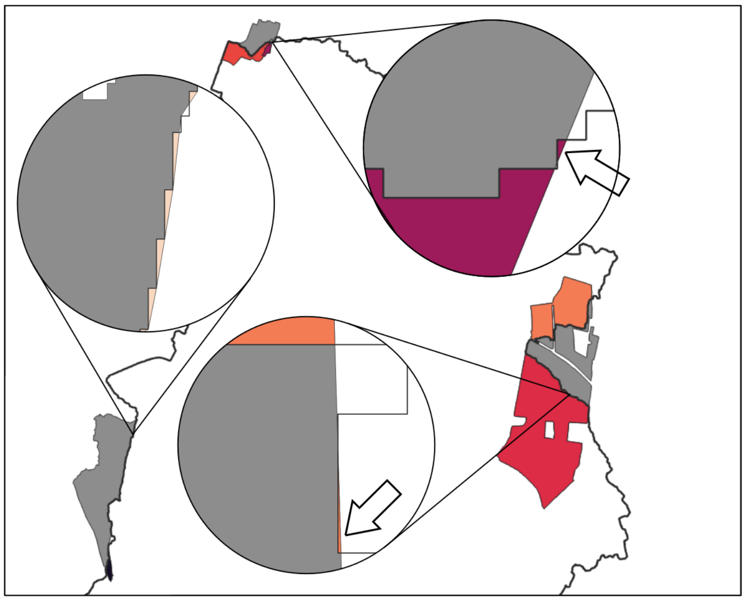


Figure 2.10: Identification of MULTIPOLYGON issues for an example land layer after the intersection with the basin boundary. The details point to the small fragments which were generated during the intersection and which have to be removed.

- Convert the intersected layer to type 'POLYGONS'. In QGIS there is the processing tool `Convert geometry type` to do this.
- Filter the converted layer for very small polygons which have e.g. areas of smaller than 2 times the pixel size of the DEM raster which is used in the `SWATbuildR` project and delete them.

`checks[4]` Check for invalid features: Option 4 of `checks` is to activate/deactivate a validity check of the feature geometries. Digitizing a polygon layer can result in invalid polygons. Figure 2.11 provides a few examples for invalid polygons.

If invalid polygons are identified in an input layer, issues such as self or ring intersections have to be identified and fixed before continuing with the `SWATbuildR` script.

`checks[5]` Identifying small features: Option 5 of `checks` identifies small features in a polygon input layer. Very small polygon features should be avoided for example in the land input layer, as the routing of water from large spatial objects into small ones can cause issues in the calculated water balances. Further, small features can be unwanted artefacts in an input layer that result from the layer preprocessing.

This check function should identify small objects and recommends to remove these small features from the input layer before continuing with the model setup process.

If this option is active (`checks[5] = TRUE`) the 25% quantile of the polygon areas is calculated. The Q_{25} value is then multiplied with the input argument `area_fct` to control the threshold for the identification of small objects. Default `area_fct` is set to 0.05. Thus, all polygons are identified as small polygons that have an area of smaller than $area\_fct \cdot Q_{25,area}$.

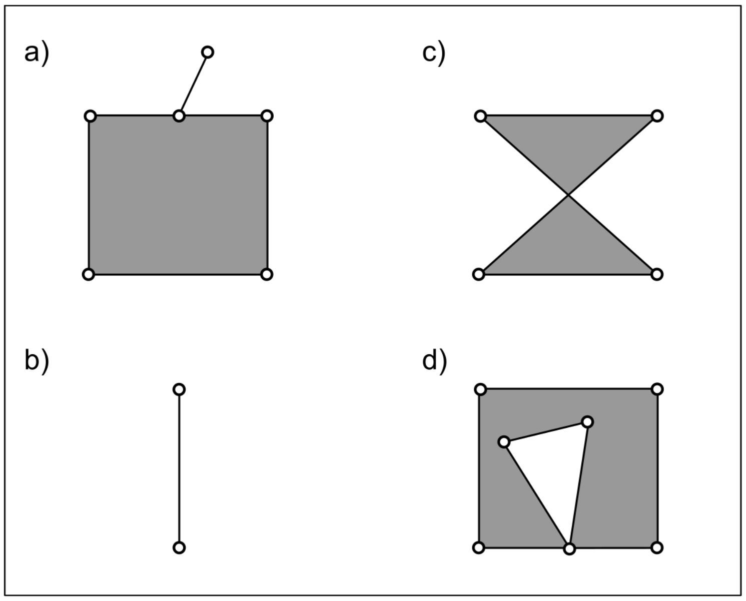


Figure 2.11: Examples for invalid polygons which can have “dangling segments” (a), no area (b), or self-intersect (c and d).

All polygons which were identified as small polygons are collected in a GIS vector layer for further inspection by the user. After removing all unwanted small objects it may be in the users’ interest to keep some small object. In this case the user can manually decrease the value of `area_fct` in order to not trigger an issue message.

`checks[6]` **Identifying covered features:** Option 6 of `checks` activates/deactivates a routine to identify features, which are covered by other features. Covered features can result from mistakes in the digitization process of e.g. the land cover layer. In the final layer product such features can be hard to identify. This option identifies the covered features and provides a basis for the user to manually fix the covered features before continuing with the model setup process.

`checks[7]` **Identifying overlapping features:** Similar to option 6, option 7 activates/deactivates a check routine for overlapping features. The reasons for overlaps can be the same as for feature coverage, the results of the two checks can however slightly differ. Therefore, always both checks should be performed.

`checks[8]` **Comparing layer coverage to basin area:** Polygon input layers should ideally cover the simulated catchment (which is defined by the basin boundary) entirely. Minor deviations between the basin area and the area which is covered by the layer polygons can occur (for example because of deleting small artefacts). Thus, it is recommended to check for a coverage close to 100%. The coverage threshold is defined with the input argument `cvrg_frc` which has a default setting of 99.9 (%).

2.3.2.2 Topological checks for line feature layers with `check_line_topology()`

Similar to the polygon input layers the channel line input layer must pass topological checks before the layer can be implemented in the model setup process. Some of the routines follow the same procedure as the topological polygon checks. Different to the polygon checks, for line features all implemented checks are always performed. Hence, no input argument is available which could activate/deactivate certain checks and layer operations. In the following, all checks and layer operations are documented which are performed with line vector layers.

Intersection with the basin boundary: Before performing any operations for a line input layer, the layer is intersected with the basin boundary. This step is primarily done to delete channel segments, which are not part of the simulated basin. All removed channel features, which were removed due to the intersect, are written into a *gpkg* ("GeoPackage") layer if a manual inspection of the removed features is intended.

The intersection with the basin boundary also helps to identify potential issues with the used stream network. If the basin boundary and the stream network do not match, an intersection could cut channels and result in *MULTILINE* features.

Check for *MULTILINE* features: After the intersection the line features are analysed for *MULTILINE* features. These are features that include several line objects. Figure 2.12 shows two very simple examples how *MULTILINE* features can be present in the channel input layer. The green example shows three channel segments (indicated by start and end points), which are collected in one feature. The green stream should form two separate channels in the model setup and should be split at the confluence point with the blue channels. The upper two segments should be merged to one segment of the remaining feature. The blue example channel is a channel that intersects the basin boundary and would therefore cause a *MULTILINE* feature after the intersection with the basin boundary. In general, a stream network must not intersect the basin boundary. As a consequence, the channel layer and the basin boundary should be checked for plausibility and data quality. If *MULTILINE* features are identified they are written into a *gpkg* layer, which should help in the analysis of the channel input layer.

Check for invalid features: For all line features a validity check is performed. Similar to the polygon features, line segments would be identified here if they self-intersect or if for example nodes of the lines are duplicated. Any identified invalid features are written into a *gpkg* layer, which should help in the analysis of the channel input layer.

Identify short features: This routine identifies short features in the line input layer. Although short channel segments are not necessarily problematic, a large number of channels can affect the computation time of the final SWAT+ model setup. Short line features can also be artefacts of the digitization process of the channels. This routine should help to identify these features.

For the identification of short line features the 25% quantile of the lengths is calculated. The Q_{25} value is then multiplied with the input argument `length_fct` to control the threshold for the identification of small objects. Default `length_fct` is set to 0.05. Thus, all lines are identified as short features with a length shorter than $length\_fct \cdot Q_{25,length}$.

All line features which were identified as short features are written into a *gpkg* layer for further inspection by the user. After removing all unwanted short objects it may be in the users interest to keep some short line features. In this case the user can manually decrease the value of `length_fct` in order to not trigger an issue message (`length_fct = 0` deactivates this check function).

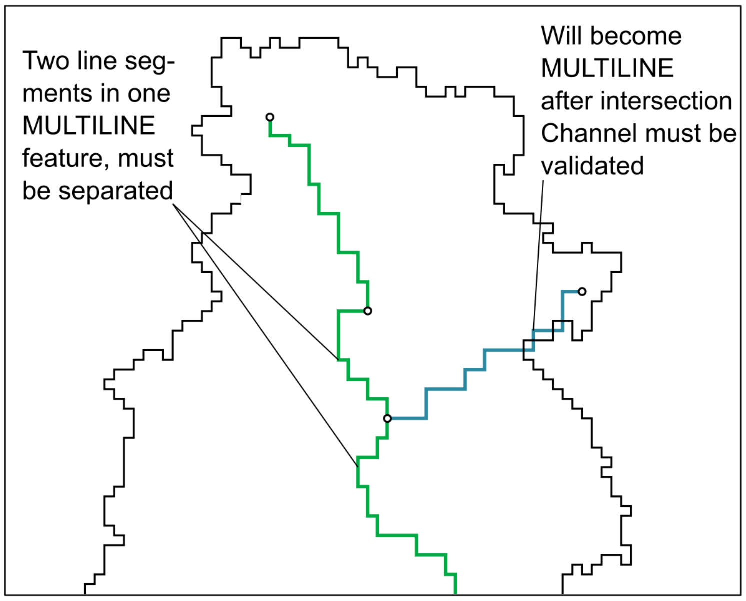

Two line segments in one MULTILINE feature, must be separated
Will become MULTILINE after intersection Channel must be validated

Figure 2.12: Examples for channels that would be identified as *MULTILINE* features.

Identify crossing features: In most cases, channel line features should not cross. Therefore, a routine is implemented to identify crossing line features. There are, however, cases where crossings should be allowed, such as culverts under an open channel, pipes crossing at different depths. If crossings should be allowed the user can set the input argument `can_cross = TRUE`. This deactivates the routine for crossing checks. It is advised to keep the default value `can_cross = FALSE` to identify any unwanted crossings and deactivate this routine in a second run if no unwanted river crossings are present in the input layer. All crossing line features are written into a *gpkg* layer for further inspection by the user.

2.3.2.3 Checking layer attributes

The vector input layers for land and channel objects require to have the attributes `id` and `type` (see also the sections 2.2.4 and 2.2.5 for further information). Additionally, the object IDs for land and the channel features in each layer must be unique.

The function `check_layer_attributes()` checks if the attribute columns exist in the input layers and if the `id` values are unique. The function additionally has the input argument `type_to_lower`. This is useful if the user wishes to transform the `type` labels to all lower case words. This option is set to `TRUE` for the channel layer as it requires lower case inputs. The default setting for the land layer is `FALSE`, because the definition of the labels is the users' choice. But if a conversion to lower case letters is preferred by the user (e.g. when converting from SWAT2012 to SWAT+ typically all upper case labels should be converted to all lower case) the conversion by setting `type_to_lower = TRUE` in the `check_layer_attributes()` function call for the land layer can be activated.

2.3.2.4 Checking and projecting the layer reference system

For the model setup process all input layers (vector and raster data) must be in the same coordinate reference system (CRS). The CRS to which all layers are compared (and projected) is the basin boundaries CRS, which will be used as the project CRS.

The function `check_project_crs()` compares the the `layers` CRS to the project CRS, which is provided to the function in the 'well known text' (wkt) format with the input argument `proj_wkt`. In the settings the variable `project_layer` was defined default to `TRUE`. This controls that a layer will be reprojected if the CRS differs to the `proj_wkt`. The input argument `label` is only relevant for printing output messages to the console. The `type` must be either `'raster'` or `'vector'`. But these settings are already correctly set in the model building script and should not be changed by the user. Below is an example of the function call for checking and reprojecting the CRS of the land layer.

```
check_project_crs(layer = channel, proj_wkt = proj_wkt,
                  proj_layer = project_layer,
                  label = 'channel layer', type = 'vector')
```

2.3.3 Reading vector input data

The `SWATbuildR` script reads the required vector inputs basin boundary and land and the optional channel layer, performs all topological and other test, prepares the layers and writes the layers into the *'./data/vector'* folder of the `SWATbuildR` project for further use in the model setup process. The following section goes through the reading and data preparation of the layers as it is performed in the script version of `SWATbuildR`. Detailed information on the checks can be found in the previous section above.

2.3.3.1 Basin boundary layer `bound`

The basin boundary is primarily used to crop all other spatial data to the same outer boundary which defines the study site. The basin boundary is read as an `sf` object and can be in any vector data format that can be read by `read_sf()` and is assigned to the variable `bound`. The path to the basin boundary vector file is defined with the `bound_path` in the `settings.R` All attributes of `bound` will be discarded and only its geometry will be further used. Thus, there are no attribute requirements for the basin boundary input. The CRS of the basin boundary is stored in the well known text format (wkt) in the variable `proj_wkt` and defines the `SWATbuildR` projects CRS.

```
bound <- read_sf(bound_path) %>% select()

proj_wkt <- st_crs(bound)$wkt
check_polygon_topology(layer = bound, data_path =  data_path, label = 'basin',
                       n_feat = 1, checks = c(F,T,T,T,F,F,F,F))
```

For `bound` the topological checks 2, 3, and 4 will be performed with `check_polygon_topology()`. The features will be checked if there is exactly one feature (`n_feat = 1`) in the data set, whether the basin boundary feature is of type *MULTIPOLYGON*, and if the polygon is valid. If all checks for the basin boundary are successful, the basin boundary is written to the *'./data/vector'* folder in ESRI *shp* format with the file name *'basin.shp'*. If any of the checks identified an issue the *'basin_topological_issues.gpkg'* is written to *'./data/vector'*. The user can open this file in any GIS and use the file to identify the sources of the identified issues and resolve them before rerunning the code for reading the basin boundary and performing the checks. The example below shows the output the function `check_polygon_topology()` if all topological checks for the basin layer were successful.

```
check_polygon_topology(layer = bound, data_path =  data_path, label = 'basin',
                       n_feat = 1, checks = c(F,T,T,T,F,F,F,F))

#> Running topological checks and modifications for the basin layer:
#>
#> Analyzing basin layer for specific number of features...
#>   v  Number of features correct.
#> Analyzing basin layer for MULTIPOLYGON features...
#>   v  No MULTIPOLYGON features identified.
#> Analyzing basin layer for invalid features...
#>   v  No invalid features identified.
#>
#>
#>   v  All checks successful! Saving checked basin layer.
```

2.3.3.2 Land layer `land`

The land object input layer defines all surfaces in the landscape of a watershed, including all land surfaces and standing water bodies. The land layer is read as an `sf` object and can be in any vector data format that can be read by `read_sf()`. The path to the land vector file is defined with the `land_path` in the `settings.R`. After reading the land layer the layer attributes are checked if they contain an `id` and a `type` column and if the provided `id`s are unique for all features. The land layer is assigned to the variable `land`. With `check_project_crs()` the CRS of land is compared with the project CRS `proj_wkt` and by default projected to this reference system.

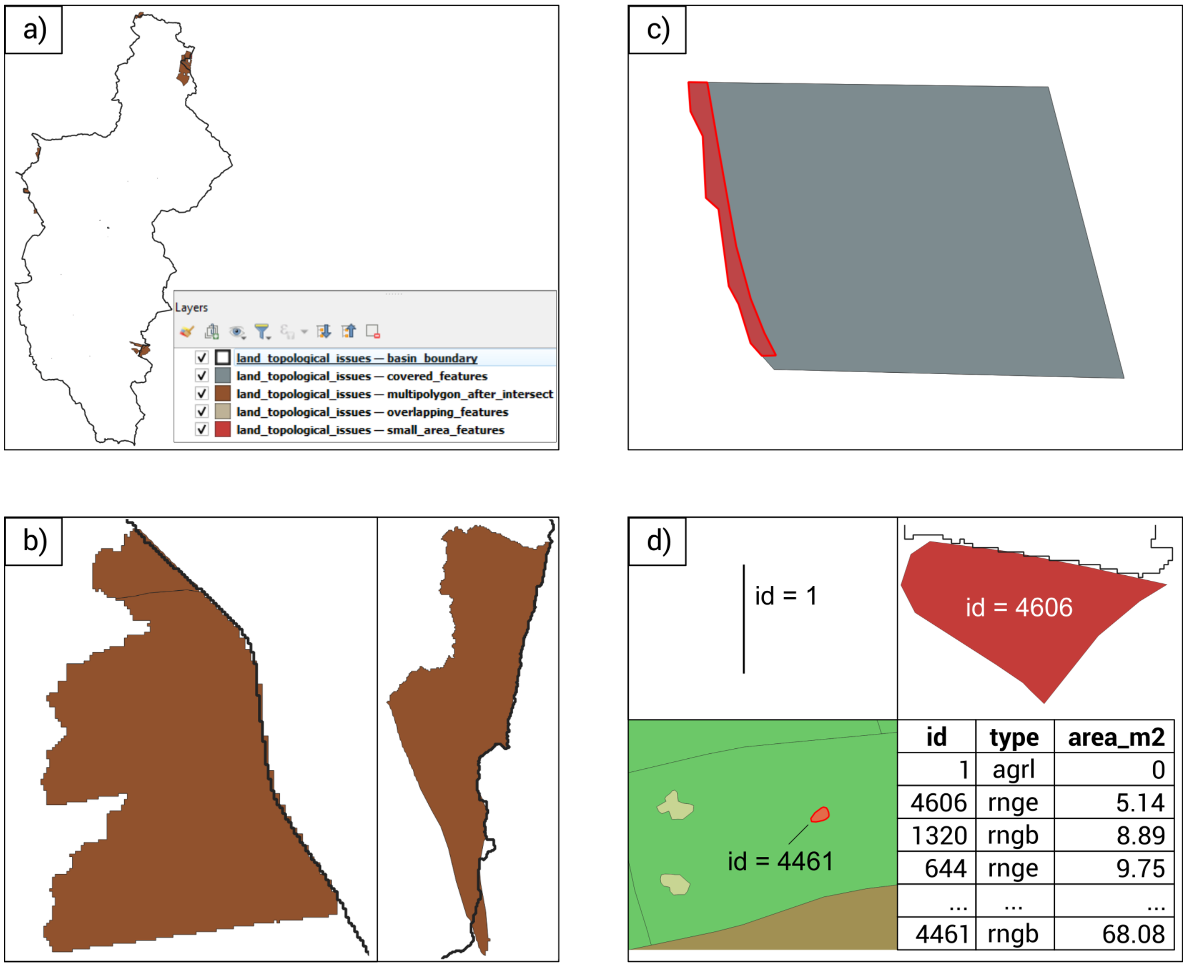
```
land <- read_sf(land_path) %>%
  check_layer_attributes(., type_to_lower = FALSE) %>%
  check_project_crs(layer = ., proj_wkt = proj_wkt, proj_layer = project_layer,
                    label = 'land layer', type = 'vector')
```

For the land layer a very comprehensive set of topological checks is performed that was described in detail in section 2.3.2.1. Below a typical example of a first round of checks for a land layer with ~5000 features is shown. The printed output is more comprehensive compared to the checks of the basin boundary layer. In a first step the intersection with the basin boundary layer saved in *'./data/vector/basin.shp'* is performed. The output of the check routine provides the information that 33 features were removed from the layer. These were features which are available in the land layer, but are located entirely outside of the basin area. The analysis of the geometry type identified 18 *MULTIPOLYGON* features. Setting `area_fct = 0.05` resulted in a threshold of 69 m^2 for the identification of small features. In total 15 features had areas smaller than 69 m^2. These features have to be analysed manually. Small features, which are in fact artefacts of the digitization process, should then be removed. After the manual inspection `area_fct` can be set to 0 to ignore all remaining small features which should remain as objects in the model setup. This decision has to be made with care as features with areas in the range of the pixel size of the used DEM layer can cause issues at a later step of the model setup process. One covered and two overlaps were identified between features. These are likely the same objects. And have to be inspected manually in any GIS. The layer coverage of the land layer was at least 99.9% and therefore the check was successful.

```
check_polygon_topology(layer = land, data_path = data_path, label = 'land',
                       area_fct =  0.05, cvrg_frc = 99.9,
                       checks = c(T,F,T,T,T,T,T,T))
#> Running topological checks and modifications for the land layer:
#>
#> Intersection of land layer with basin boundary layer...
#>   v  Intersection completed. 33 features removed from the land layer (located
#>   outside of the basin boundary).
#> Analyzing land layer for MULTIPOLYGON features...
#>   x  18 MULTIPOLYGON features identified after intersection with basin boundary.
#> Analyzing land layer for invalid features...
#>   v  No invalid features identified.
#> Analyzing land layer for very small feature areas...
#>   x  15 features identified with an area < 69 m² (is 0.05 * Q25 of all areas)
#>   after intersection with basin boundary.
#> Analyzing land layer for features covered by other features...
#>   x  1 feature covered by other feature identified.
#> Analyzing land layer for overlapping features...
#>   x  2 overlaps between features identified.
#> Analyzing land layer coverage with basin boundary...
#>   v  Layer coverage OK.
#>
#> Error in check_polygon_topology(layer = land, data_path = data_path, label =
↪  "land",  :
#>
#>
#> Topological issues for the land layer identified!
#>
#> Writing the layer land_topological_issues.gpkg into
#>  'Define:/your/path/demo_project/data/vector'
#>
#> Load the .gpkg layer in a GIS to analyse the features that cause issues.
#> Fix the issues in the land layer before proceeding with the model setup.
```

As issues were identified for the land layer, the routine triggers an error message and provides further information on how to proceed. The topological check routine generated the layer *'land_topological_issues.gpkg'* and wrote it into the projects vector data folder *'Define:/your/path/demo_project/data/vector'*. The layer should now be loaded from there in any GIS and inspected together with the land input layer, which was loaded in the script and which caused the issues. The layer *'land_topological_issues.gpkg'* can help to identify the features in the land input layer which should be fixed. Please make sure to perform all changes in the actual land input layer and not for the layer *'land_topological_issues.gpkg'* as changes there would be ineffective for resolving issues.

Figure 2.13 highlights a few details of the written geopackage *'land_topological_issues.gpkg'*. The loaded geopackage includes five layers including the basin boundary and layers for each type of topological issue identified in the check routine (Figure 2.13 a)). Only a few polygons actually cause issues. Most of them are located along the basin boundary. This indicates already that these polygons remain as small *MULTIPOLYGON* artefacts in the land layer after the intersection with the basin boundary. Figure 2.13 b) shows two examples for such polygons. In such cases the remaining parts of these polygons are negligible for the final model setup and should be removed entirely. The empty pixels are ideally filled with the neighbouring features or can be left blank if it will not be too many in the end (issue of area coverage). One feature was covered by another feature in the land layer. This and the covering feature are shown in Figure 2.13 c). It seems that the overlap/coverage resulted from a mistake when the small polygon was added to the land layer. Figure 2.13 d) shows a few examples of features which were identified as small features together with the attribute table. The feature with `id = 1` has an area of 0. Zooming into the feature makes clear that this is a polygon without an area and should be removed. The feature with the `id = 4606` would create a small polygon after the intersection with the basin boundary. It is also recommended to remove this feature. The feature with the `id = 4461` was also identified as a small feature and has an area of ~68 m^2. The object is however a landscape feature, which should remain in the land layer and would therefore not be removed. After checking all identified features the input argument `area_fct` in `check_polygon_topology()` will be set to 0 to not trigger this issue again.



✓ ☐	land_topological_issues — basin_boundary
✓ ▨	land_topological_issues — covered_features
✓ 🟫	land_topological_issues — multipolygon_after_intersect
✓ ▨	land_topological_issues — overlapping_features
✓ 🟥	land_topological_issues — small_area_features

id	type	area_m2
1	agrl	0
4606	rnge	5.14
1320	rngb	8.89
644	rnge	9.75
...	...	...
4461	rngb	68.08

Figure 2.13: Examples for issues identified in the topology checks performed for the land layer above. a) shows all layers of *'land_topological_issues.gpkg'*, which include the basin boundary and the identified grouped issues. b) shows two polygons, which were identified to result in *MULTIPOLYGON* features after the intersection with the basin boundary. c) shows the covered/overlapping features identified. d) shows examples for small identified features and the corresponding attribute table.

After fixing all issues, loading the updated land layer and rerunning the topological checks, all checks were successful. The checked land layer is written to the *'./data/vector'* folder in ESRI *shp* format with the file name *'land.shp'*.

```
land <- read_sf(land_path) %>%
  check_layer_attributes(., type_to_lower = FALSE) %>%
  check_project_crs(layer = ., proj_wkt = proj_wkt, proj_layer = project_layer,
                    label = 'land layer', type = 'vector')

check_polygon_topology(layer = land, data_path = data_path, label = 'land',
                       area_fct =  0, cvrg_frc = 99.9,
                       checks = c(T,F,T,T,T,T,T,T))

#> Running topological checks and modifications for the land layer:
#>
#> Intersection of land layer with basin boundary layer...
#>   v  Intersection completed.
#> Analyzing land layer for MULTIPOLYGON features...
#>   v  No MULTIPOLYGON features identified.
#> Analyzing land layer for invalid features...
#>   v  No invalid features identified.
#> Analyzing land layer for very small feature areas...
#>   v  No small features identified.
#> Analyzing land layer for features covered by other features...
#>   v  No covered features identified.
#> Analyzing land layer for overlapping features...
#>   v  No overlapping features identified.
#> Analyzing land layer coverage with basin boundary...
#>   v  Layer coverage OK.
#>
#>
#>   v  All checks successful! Saving intersected land layer.
```

Split land layer into HRU (land) and reservoir (water) objects: The land input layer includes all land surfaces and standing water surfaces. In a SWAT+ model HRUs objects represent land surfaces, while standing water bodies are simulated with reservoir objects. Thus, it is necessary to separate the two SWAT+ object types into separate data sets. This step is performed with the function `split_land_layer()`. It only requires the `vector_path` as an input argument. The function loads the *'land.shp'* vector layer from there and extracts all features, which are of `type == 'watr'`. These features receive new `ids` (from 1 to the number of features, while keeping also the initial land layer ids) and will be written into the layer *'res.shp'*. The remaining land features receive new `ids` as well and will be written to the layer *'hru.shp'* in the *'./data/vector'* folder.

```
split_land_layer(data_path)
```

2.3.3.3 Channel layer `channel`

The channel layer is an optional input (although most model setups will have channel objects). If no channel input should be read, the `channel_path` in `settings.R` should be set `NULL`. If a path of a channel vector layer is assigned to `channel_path` the file is read as an `sf` object. The channel layer can be in any vector data format that can be read by `read_sf()`. After reading the channel layer, the layer attributes are checked if they contain an `id` and a `type` column and if the provided `id`s are unique for all features. The channel layer is assigned to the variable `channel`. With `check_project_crs()` the CRS of `channel` is compared with the project CRS `proj_wkt` and by default projected to this reference system.

The topological checks which are described in section 2.3.2.2 are performed for the channel layer after reading and checking layer attributes and CRS. Below an example of the outputs printed for topological checks of a channel layer is shown. In the first step the intersection with the basin boundary layer saved in *'./data/vector/basin.shp'* is performed. The output of the check routine provides the information that 33 features were removed from the layer. These were features, which are available in the land layer but are located entirely outside of the basin area. The analysis of the geometry type identified 18 *MULTIPOLYGON* features.

```
if(!is.null(channel_path)) {
  channel <- read_sf(channel_path) %>%
    check_layer_attributes(., type_to_lower = TRUE) %>%
    check_project_crs(layer =., proj_wkt = proj_wkt, proj_layer = project_layer,
                      label = 'channel layer', type = 'vector')
  check_line_topology(layer = channel, data_path = data_path,
                      label = 'channel', length_fct = 0.05, can_cross = FALSE)
}
#> Running topological checks and modifications for the channel layer:
#>
#> Intersection of channel layer with basin boundary layer...
#>   v  Intersection completed. 616 features removed from the channel layer
#>   (located outside of the basin boundary).
#> Analyzing channel layer for MULTILINE features...
#>   x  3 MULTILINE features identified after intersection with basin boundary.
#> Analyzing channel layer for invalid features...
#>   v  No invalid features identified.
#> Analyzing channel layer for very short feature lengths...
#>   x  1 feature identified with a length < 4.4 m (is 0.05 * Q25 of all lengths)
#>   after intersection with basin boundary.
#> Analyzing channel layer for crossing features...
#>   v  No crossing features identified.
```

After fixing all issues, loading, the updated land layer and rerunning the topological checks, all checks were successful. The checked channel layer is written to the *'./data/vector'* folder in the ESRI *shp* format with the file name *'land.shp'*.

```
if(!is.null(channel_path)) {
  channel <- read_sf(channel_path) %>%
    check_layer_attributes(., type_to_lower = TRUE) %>%
    check_project_crs(layer =., proj_wkt = proj_wkt, proj_layer = project_layer,
                      label = 'channel layer', type = 'vector')
  check_line_topology(layer = channel, data_path = data_path,
                      label = 'channel', length_fct = 0, can_cross = FALSE)
}
#> Running topological checks and modifications for the channel layer:
#>
#> Intersection of channel layer with basin boundary layer...
#>   v  Intersection completed.
#> Analyzing channel layer for MULTILINE features...
#>   v  No MULTILINE features identified.
#> Analyzing channel layer for invalid features...
#>   v  No invalid features identified.
#> Analyzing channel layer for very short feature lengths...
#>   v  No small features identified.
#> Analyzing channel layer for crossing features...
#>   v  No crossing features identified.
#>
#>   v  All checks successful! Saving intersected channel layer.
```

2.3.4 Checking the water object connectivity

After the water objects `res` and `channel` are checked individually and saved to vector layers, the connection between all water objects is analysed with the function `check_cha_res_connectivity()`. The function loads the written `res` and `channel` layers from *'./data/vector'*. In a first step the water objects are prepared, channels are clipped and split by the reservoir objects to form the combined water object network. In the connectivity analysis the spatial relationship between all water objects is analysed. All channel objects must be connected at its end point to a start point of another channel feature or to a reservoir. Reservoirs which receive water from channels must also be connected to channels which leave that reservoir.

The routine identified four channel segments and one reservoir from the input layers above to be not connected to other water objects. The layer *'water_connectivity_issues.gpkg'* was written into the projects vector data folder *'Define:/your/path/demo_project/data/vector'* and should now be loaded in any GIS together with the channel and the reservoir input layers to inspect and fix the identified issues.

```
check_cha_res_connectivity(vct_path, id_cha_out, id_res_out)
#> Preparing channel and reservoir features...
#>   v  OK!
#> Analyzing connectivity of water object network...
#>   x  4 channel segments identified not connected to other channels or reservoirs.
#>   x  1 reservoir identified to be not connected to other channels or reservoirs.
#>
#> Error in check_cha_res_connectivity(vct_path, id_cha_out, id_res_out) :
#>
#> Connectivity issues for the water objects identified!
#>
#> Writing the layer water_connectivity_issues.gpkg into
#> 'Define:/your/path/demo_project/data/vector'
#>
#> Load the .gpkg layer in a GIS to analyse the features that cause issues.
#> Fix the issues in the land and channel layers before proceeding with
#> the model setup.
```

Figure 2.14 shows some of the identified issues from the water object connectivity check. Figure 2.14 a) shows a simple example where the direction of a headwater channel is not correct and the channel points away from the channel network. The direction of the channel has to be reversed. This can for instance be done with the `Reverse line` tool in QGIS (see Figure 2.15). The example b) is already a more complex situation. Three channels point to one node but no channel points away from this node. Thus, the direction of at least one channel is incorrect. By just evaluating the line features two options to change channel directions would be possible to result in a "correct" channel network (simply based on feature relationship). Here, a check of the elevations at the endpoints is helpful and clearly shows that the lower left channel has to be modified. Example c) shows the unconnected reservoir. In this example it is very likely that a channel is missing which connects the flow from the reservoir with the channel network. In this case a channel must be added to fill this gap.

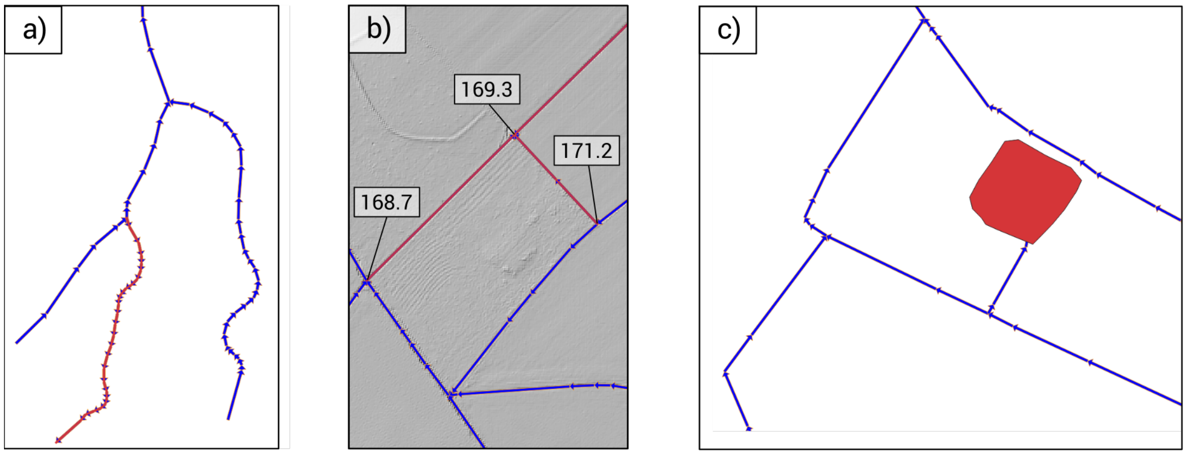


Figure 2.14: Examples for issues identified in the water object connectivity analysis. a) and b) shows issues with the channel connectivity. c) shows the identified unconnected reservoir.

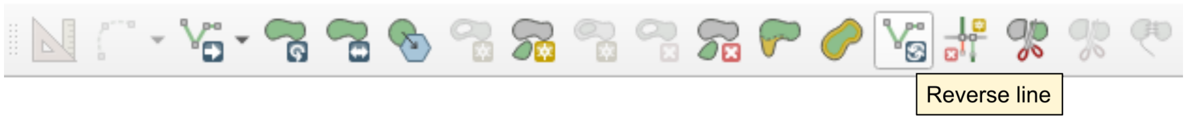


Figure 2.15: Screenshot of the `Reverse line` tool in QGIS.

After fixing all issues and rerunning the topological checks, all checks were successful. The checked channel and reservoir layers are updated and written to the *'./data/vector'* folder in the ESRI *shp* format with the file names *'cha.shp'* and *'res.shp'*.

```
check_cha_res_connectivity(data_path, id_cha_out, id_res_out)

#> Preparing channel and reservoir features...
#>   v  OK!
#> Analyzing connectivity of water object network...
#>   v  No disconnected channels identified.
#>   v  No disconnected reservoirs identified.
#>
#>   v  Water object connectivity check successful!
```

2.3.5 Reading raster input data

A SWAT+ model setup requires terrain and soil properties, which are assigned to the spatial objects of a model setup. The `SWATbuildR` script version reads a Digital Elevation (DEM) and a soil raster layer together with the soil input tables in a SWAT format. After reading the layer, the CRS as well as the coverage of the raster inputs are analysed and the relevant terrain and soil inputs are prepared for the SWAT+ model setup. The prepared raster data are written into the *'./data/raster'* folder of the `SWATbuildR` project for further use in the model setup process. The following section gives an overview of an additional raster check function (`check_raster_coverage()`) and then goes through the reading and data preparation of the layers as performed in the script version of `SWATbuildR`.

2.3.5.1 Checking coverage of raster input with `check_raster_coverage()`

For a SWAT+ model setup it would be ideal that the entire study region is completely covered with terrain and soil data. Yet, small gaps in the raster data may be present (e.g., missing soil data for mostly inundated areas, missing data in urban areas). `SWATbuildR` sets strict requirements for data coverage and requests that each land object is covered by a minimum fraction of its area with soil and terrain data. This check of the input data is performed with the function `check_raster_coverage()`. Below is the example for the analysis of the soil raster layer coverage. The coverage of the `soil` layer is analysed with the `hru` layer, which only includes land surfaces (after the split into `hru` and `res`). The coverage with soil data is tested with a data coverage fraction of `cov_frc = 0.75`, which means that the area of each polygon in `hru` must be covered by at least 75% with raster data from `soil`. If this condition is not `TRUE` for all spatial objects the check function triggers an error and provides information which polygons were not adequately covered with raster data. In the example, data is missing for the polygons with the `ids` 15, 35, and 36.

```r
check_raster_coverage(rst = soil, vct_layer = 'hru', data_path = data_path,
                      label = 'soil', cov_frc = 0.75)
#> Error in check_raster_coverage(rst = soil, vct_layer = "hru",
#> data_path = data_path,  :
#>
#> The units with the following ids do have a coverage by the soil layer
#> of less than 75%:
#> 15, 35, 36
#> Please update the soil layer for better coverage before you continue!
```

2.3.5.2 DEM raster layer (`dem`)

The DEM raster is read with the function `rast()` from the `terra` package. The DEM raster layer can be in any format that is readable with `terra::rast()`. In the example below the `dem_path` points to a GeoTiff file and was defined in `settings.R`. After reading the raster input the layer CRS is compared to the project CRS `proj_wkt` and by default (`project_layer = TRUE`) the `dem` layer is reprojected to the project CRS if they differ.

```r
dem <- rast(dem_path) %>%
  check_project_crs(layer = ., proj_wkt = proj_wkt, proj_layer = project_layer,
                    label = 'basin boundary', type = 'raster')

## Prepare terrain properties from the DEM and save them to the raster folder
prepare_dem_terrain_properties(rst_path, bound)
```

With `check_raster_coverage()` the coverage of the `dem` layer is analysed for all spatial objects that are defined in the `land` layer, which was saved in the `vct_path`. `land` includes all land as well as all standing water objects. A SWAT+ setup requires spatial terrain information for both types of spatial objects (also for channels, but the condition for channels is fulfilled if it is for all land objects). The data coverage for each spatial object in `land` is set to a strict value of `cov_frc = 0.95`.

```
check_raster_coverage(rst = dem, vct_layer = 'land', data_path = data_path,
                      label = 'dem', cov_frc = 0.95)
```

After passing all checks, the terrain slope (in percent) is calculated and both the `dem` and slope are written to the projects raster folder *'./data/raster'*. In this step the original `dem` file is written as the GeoTiff file *'dem.tif'* and the terrain slope is calculated and written as *'slope.tif'*.

```
save_dem_slope_raster(dem_path, data_path)
```

2.3.5.3 Soil input layer (`soil`)

The `soil` raster is read with the function `rast()` from the `terra` package. The soil layer can be in any raster format that is readable with `terra::rast()`. The `soil_layer_path` is defined in the `settings.R`. After reading the raster input the layer CRS is compared to the project CRS `proj_wkt` and is (default `project_layer = TRUE`) reprojected to the project CRS if the project CRS and the soil CRS differ.

```
soil <- rast(soil_layer_path) %>%
  check_project_crs(layer = ., proj_wkt = proj_wkt, proj_layer = project_layer,
                    label = 'soil layer', type = 'raster')
```

Soil information has to be prepared only for land surfaces. Thus, the coverage of the soil input data is analysed with the `hru` layer, which excludes the standing water surfaces. Also the coverage fraction is defined less strict compared to the `dem` with `cov_frc = 0.75`.

```
check_raster_coverage(rst = soil, vct_layer = 'hru', data_path = data_path,
                      label = 'soil', cov_frc = 0.75)
```

With `save_soil_raster()` the soil information is transformed to the grid of the `dem` elevation data, so that all raster data is available on the same spatial grid (of the `dem` which will be used for several later operations). The `soil` layer is written as the GeoTiff file *'soil.tif'* to the projects raster folder *'./data/raster'*.

```
save_soil_raster(soil, data_path)
```

2.3.5.4 Aggregate terrain and soil information for all HRUs

With `aggregate_hru_dem_soil()` terrain properties and soil type information is aggregated for each land object in the layer `hru`. For each land object the mean elevation and mean slope are calculated from the raster layers *'dem.tif'* and *'slp.tif'* (available for the `rst_path`) using `terra::zonal()` statistics. For each land object also the modal value of the soil classes (using the layer *'soil.tif'* in `rst_path`) is calculated and thus the dominant soil within each land polygon is assigned to each land object. The aggregation of the soil data from the soil raster input to dominant soils is illustrated in Figure 2.3.

```
aggregate_hru_dem_soil(data_path)
```

The resulting table is written in *'Define:/your/path/demo_project/data'* into the projects *tables.sqlite* data base. The table assigns each HRU `id` a value of mean elevation `elev`, slope `slp` and the raster value of the dominant soil as they were given in *'soil.tif'*. Below an example for the soil terrain properties table is given.

```
#> # A tibble: 373 × 4
#>       id  elev  slope  soil
#>    <dbl> <dbl>  <dbl> <dbl>
#>  1     1  141. 0.0265     6
#>  2     2  142. 0.0266     6
#>  3     3  138. 0.0212     6
#>  4     4  139. 0.0348     6
#>  5     5  135. 0.0311    10
#>  6     6  138. 0.0263     6
#>  7     7  134. 0.0258     6
#>  8     8  141. 0.0257     6
#>  9     9  139. 0.0333     6
#> 10    10  133. 0.0324     6
#> # ... with 363 more rows
```

If the land input layer for example includes polygons, which cannot be represented on the `dem` layer raster grid (e.g. because due to the polygon shape the polygon would result in no pixels after rasterizing the layer), the error below would be triggered. Together with the error message, a vector layer *'hru_no_dem_soil.gpkg'* is written into *'Define:/your/path/demo_project/data/vector'*, which helps to identify the problematic polygons. These have to be fixed in the `land` input layer. The layer has to be reloaded and processed in the script before continuing with the model setup process.

```
aggregate_hru_dem_soil(data_path)

#> Error in  aggregate_hru_dem_soil(data_path) :
#>
#> No elevation/slope/soil assigned to some HRUs!
#>
#> Writing the layer hru_no_dem_soil.gpkg into
#>  'Define:/your/path/demo_project/data/vector'
#>
#> Load the .gpkg layer in a GIS to see which land objects cause the issue.
#> This issue typically occurs for very small features that cannot be represented
#> on the DEM raster grid.
#>
#> Please fix this issue in the land input layer and redo the previous steps.
```

2.3.6 Land object connectivity

A key function of `SWATbuildR` is the definition of the connectivity between spatial objects. The connectivity between spatial objects is defined by the type of flux (e.g. total flux, surface or lateral runoff, tile flow, or groundwater recharge) and a fraction which defines the amount of a certain flux that is sent from one spatial object to another object.

`SWATbuildR` generates the connectivity between spatial objects in a two stage approach. i) The connectivity between land objects is calculated based on the terrain, which is derived from the DEM and the spatial outline of the land object polygons. Water objects are considered to be sinks for fluxes from land objects and do not further send any fluxes to land objects. ii) For the water objects which receive fluxes from land objects the water object network is calculated, which defines the connections between channels and reservoirs.

In this section the calculation of the land object connectivity with `SWATbuildR` is documented. The water object connectivity is outlined in the following section.

2.3.6.1 Preparation of terrain inputs

The calculation of fluxes between spatial objects employs a flow accumulation and a flow pointer layer, which are calculated based on the input DEM raster layer. The DEM must be hydrologically conditioned before it is used in the calculation of flow accumulation and direction. Local sinks, depressions and discontinuities in flow paths along the hill slopes must be eliminated. Water objects (channels and reservoirs) are burnt-in in the dem layer to enforce water objects to be local sinks for the fluxes from land objects.

All operations to prepare the catchment's terrain properties for the calculation of the land object connectivity are included in the workflow of `prepare_land_terrain()`. `prepare_land_terrain()` loads the DEM layer which is saved as *'./data/raster/dem.tif'*. The hydrological conditioning fills in a first step single cell pits using the `whitebox` tools function `wbt_fill_single_cell_pits()`. In a next step, the pit filled DEM is breached with `wbt_breach_depressions_least_cost()` implementing the least-cost path analysis proposed by Lindsay and Dhun (2015). After the least-costs breaching local depressions will likely remain. Some of those depressions will be removed in a third cleaning step using the fill depression routine of Wang and Liu (2006) implemented with the `whitebox` tools function `wbt_fill_depressions_wang_and_liu()`. The combination of breaching and pit filling that was implemented here was proposed by J.P. Gannon in his hydroinformatics online resources which can be found here. This resource also provides a simple explanation for sink filling and depression breaching. The hydrologically conditioned DEM layer is written into the projects raster data folder *'./data/raster'* as *'dem_fill_brch.tif'*.

```
prepare_terrain_land(data_path)
```

The breached and filled DEM layer is further processed by burning in the channels and reservoirs. For the channel burn-in only surface channels `type = cha` are used and subsurface channel objects (`type = sub`) are excluded. To ensure that the burnt-in channel objects form continuous depressions in the DEM it is buffered by the raster pixel dimension. Figure 2.16 a) and b) shows the differences between the original DEM input layer and the breached and sink filled DEM layer with the burnt-in water objects. The differences which result from the breaching and sink filling are hard to identify and, for example, are visible for the structures in the lower left corner of Figure 2.16 b) where single structures were smoothed out to minimize the risk of discontinued flow. The burnt-in water objects are clearly visible. In the upper part of the figure a gap between burnt-in channels is visible. The reason for the gap is a short subsurface channel section in the flow network, which will not be considered as a potential sink in the calculation of the land object connectivity. The hydrologically conditioned DEM with burnt-in water objects is written as *'dem_watr_burn.tif'* into *'./data/raster'*.

With the *'dem_watr_burn.tif'* layer the D8 flow accumulation and the D8 flow pointer (O'Callaghan and Mark, 1984) is calculated with the `whitebox` functions `wbt_d8_flow_accumulation()` and

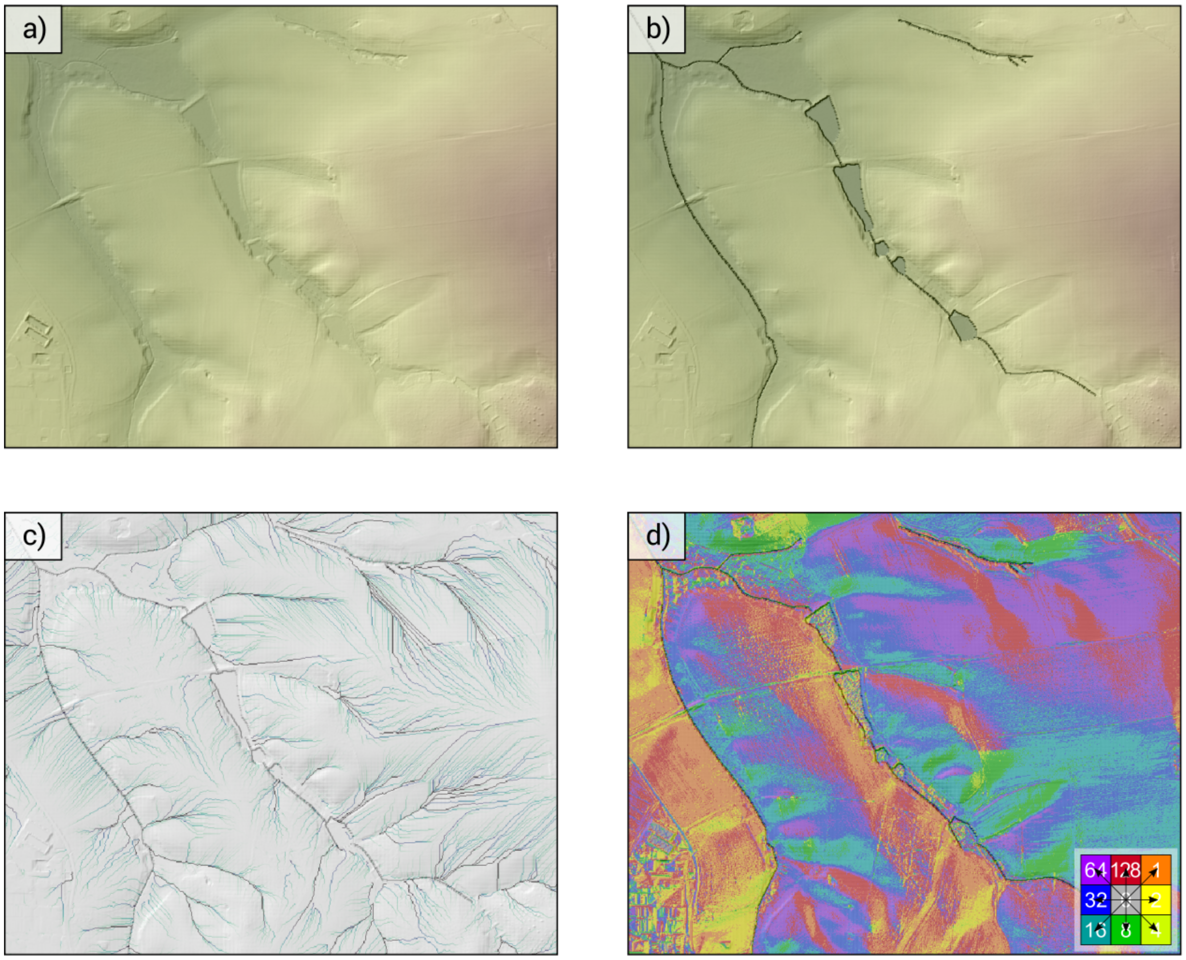


Figure 2.16: Products of the `prepare_terrain_land()` routine. a) shows the original `dem` layer. b) shows the breached and sink filled DEM layer with the burnt-in water objects. c) shows the calculated flow accumulation. d) shows the calculated D8 flow pointer.

`wbt_d8_pointer()`. Both layers are written into the project's raster data folder *'./data/raster'* as *'flac_dem_watr_burn.tif'* and *'fpnt_dem_watr_burn.tif'*, respectively. Examples for flow accumulation and flow pointer are shown in Figure 2.16 c) and d).

2.3.6.2 Calculation of the land object connections

The calculation of the land object connectivity implements the processed land object layer *'./data/vector/hru.shp'* and the prepared flow accumulation and flow pointer raster layers *'flac_dem_watr_burn.tif'* and *'fpnt_dem_watr_burn.tif'* (Figure 2.17 a)). To prepare the calculation of the connectivity the land layer is superimposed with the surface channels (`type = cha`) and the reservoirs. A combined object `id` layer raster is generated, which represents the spatial locations of all land, surface channel, and reservoir objects. The combined layer is rasterised with the DEM raster to be on the exact same grid as the flow accumulation and flow pointer layers.

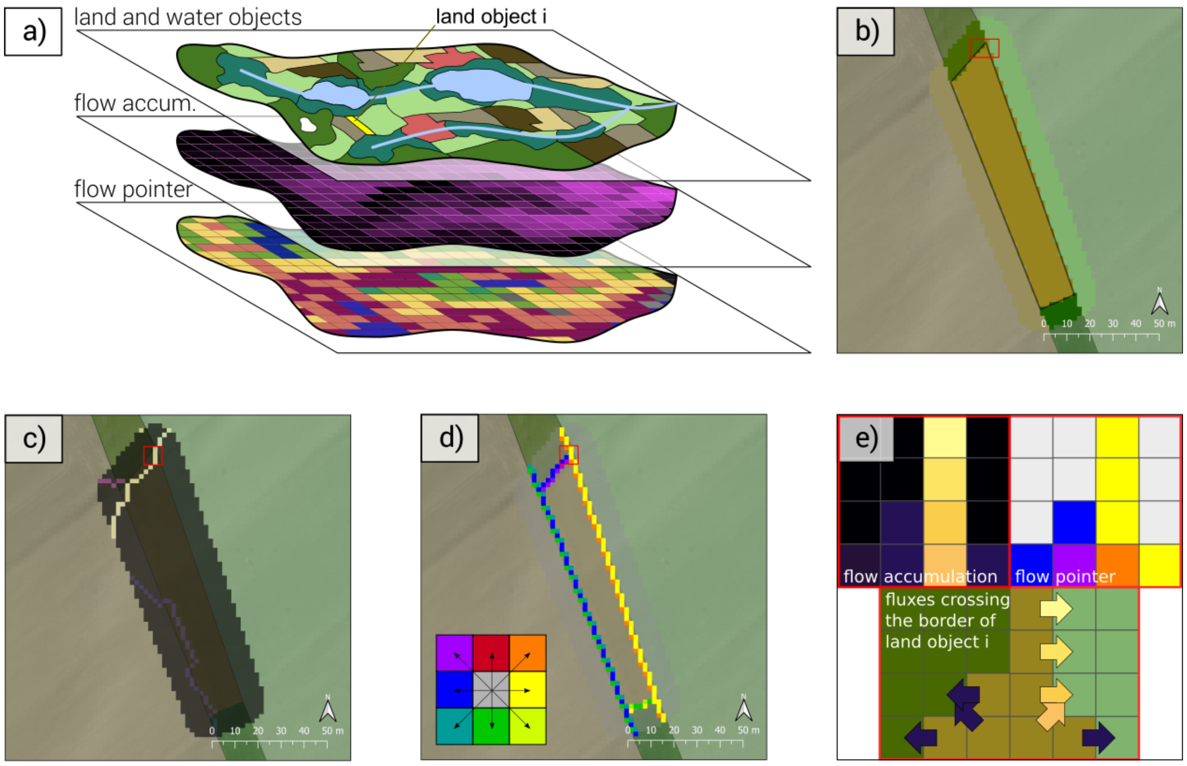


Figure 2.17: Workflow steps of `calculate_land_connectivity()`. a) shows the three input layers land and water objects, flow accumulation, and D8 flow pointer. b) shows the extracted land object i (small rectangular field) with its surrounding, which is highlighted in yellow in a). c) and d) show the flow accumulation and the flow directions along the edges of object i. e) shows the calculation of fluxes crossing the object border pixels in the detail which is highlighted with red boxes in b), c), and d).

The calculation of the land object connectivity is processed iteratively, for each land object individually. For each land object, the accumulated flow which leaves that land object is calculated in the following command sequence:

- Extract the land object i from the raster `id` layer and generate a buffer area of 5 raster pixels around the object to include information on the neighbouring objects (Figure 2.17 b)).
- Extract the flow accumulation and the flow pointer for the same raster extent (Figure 2.17 c) and d)).
- Identify the edges of the land object i and only use the flow direction of the object's edge pixels in the flow pointer layer (Figure 2.17 d))
- Based on the directions of the D8 flow pointer identify for each edge pixel the `id` of the neighbouring object to which each edge pixel points to.
- Extract the flow accumulations and group them based on the identified neighbour pixel `ids` (Figure 2.17 e)).
- Sum-up the flow accumulation values grouped by the receiving spatial object `id`.

Depending on the number of land objects this process can take between 10 minutes (for a few 100 land objects) to a few hours (for a few thousand land objects). The routine writes the progress of the calculation into the command line, as shown below. After the calculation of the flow accumulation sums that leave the land objects, the net sums of fluxes are calculated for all land objects. Along a border between two land objects it is possible that fluxes pass the border in both directions. The difference between the flux sums from one object into the other object and vice versa is calculated and used as the net flux between these objects.

In a final step, all land objects are analysed for 'sink units'. Based on the terrain properties it is possible that a land object receives fluxes from neighbouring land objects, but no fluxes leave this unit. Such a unit is then considered as a 'sink unit'. It is possible that such local sinks exist in the landscape. Although the hydrological conditioning of the DEM should have reduced the risk of having sinks in the terrain they still can occur. In the example below 192 sink units were identified. The land units are written into the layer *'land_no_connection.gpkg'* in *'./data/vector'* and should be further analysed before proceeding with the model setup.

```
calculate_land_connectivity(data_path)

#> Calculating land object connectivity:
#>  Land object 2 of 5179   Time elapsed: 11S   Time remaining: 1H 18M 6S
#>
#>  Completed 5179 Land objects in 1H 0M 18S
#>
#> Cleaning up land object connectivities...
#>   v  Done!
#>
#> Analyzing land objects for 'sink units':
#>   X 192 land objects with no connections identified.
#>
#>  The identified units are sinks and do not further route receiving water!
#>  The connections of these units have to be resolved manually.
#>  You can resolve this issue in the following ways:
#>
#>  - Edit the land input layer and adjust the boundaries of these units to better
#>    fit the flow accumulation and the flow pointer.
#>
#>  - Add additional connections manually (routine for that will be implemented
↪  soon).
#>
#>  - Leave units unconnected. Some land objects may be actual sinks in the
↪  landscape.
```

```
#>
#>  Use the layer 'land_no_connection.gpkg' that was written to
#>   'Define:/your/path/demo_project/data/vector
#>  together with the layers 'flac_dem_watr_burn.tif', 'fpnt_dem_watr_burn.tif'
#>  and the DEM 'dem_watr_burn.tif' which were saved in
#>   'Define:/your/path/demo_project/data/raster
#>  to analyse the sink land objects.
```

Figure 2.18 shows two examples with land objects that do not further route fluxes and were identified as sink units (red polygons). Both Figure 2.18 a) and b) show that often the reason for identified local sinks are artificial structures such as roads. Issues along such structures must be revised manually. In situations along road dams that create sink units there is very likely a drainage ditch implemented, as the slope water must be drained. Thus, in such cases drainage channels must be added to the channel setup to better represent the actual situation.

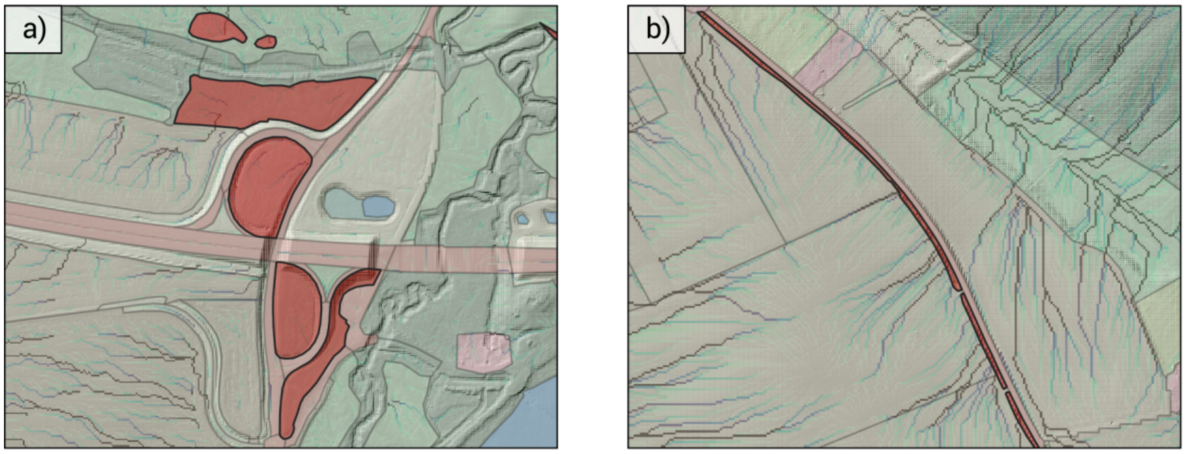


Figure 2.18: Examples for land objects which were identified as sink units (dark red polygons). a) Several patches between motorway exits are not connected due to higher elevation of the surrounding artificial structures. b) Often objects along roads are not connected, due to the elevated road dam and a missing drainage channel along the road which would drain the slope water.

If either the local sinks are justifiable in the model setup or were fixed the result of the land connectivity calculation is written into the *'tables.sqlite'* data base with the table name *'connect_ids'*. Below an example for a `connect_ids` table is given. It provides the information of the sending unit `id_from` and the receiving unit `id_to`. The `flow_acc` is the summed net flow accumulation. The lines 6 to 10 in the table show negative numbers for the receiving `id_to` objects. This indicates that these objects are water objects. To differentiate between different object types, in this table land objects have positive numbers, channels ids have values between -1 and -9999 (- channel id value) and reservoirs do have values smaller than -10000 (-10000 - reservoir id value).

```
#> # A tibble: 16,111 × 3
#>     id_from  id_to flow_acc
#>       <int>  <dbl>    <dbl>
#>  1        1      2    10901
#>  2        1    248     9915
#>  3       1   2809     3670
#>  4       1   3502     3067
#>  5       1   4555      482
#>  6       3 -10128      520
#>  7       3 -10127       48
#>  8       3   -792      254
#>  9       3   -386       10
#> 10       3   -236      480
#> # ... with 16,101 more rows
```

2.3.6.3 Reducing the number of connections

If a land object has several neighbours it can also have multiple connections to other spatial objects. Some of the calculated connections will likely have very low flow accumulation sums compared to the few dominant fluxes. In the final SWAT+ model setup it may, however, only be relevant to keep the dominant connections of the spatial objects. Also, if the land objects have many connections to other objects there is an increased risk of loop routing, which means that water is routed between spatial objects and again ends up in the initially sending object. Thus, a fraction of water is then routed in an infinite loop. Reducing the number of connections can also reduce the potential for such infinite loop routing. `reduce_land_connections()` eliminates connections between objects with very low flow fractions. Therefore the *'connect_ids'* table is loaded and for each object (`id_from`) normalised fractions are calculated based on `flow_acc`. The flow fractions are normalised based on the largest fraction that is sent from an object to receiving objects. Thus the largest fraction for each object is 1. All other fractions can vary between 0 and 1. `frc_thres` defines the threshold to keep or eliminate an object connection. `frc_thres` is defined in *'settings.R'* and the default value is 0.3. This means that the connection is eliminated if the flow fraction of a connection is lower than 30% of the largest flow fraction of this object. After reducing all connections with small flow fractions the flow fractions are normalised in a way that the sum of flow fractions which leave a land object is 1. The updated connections table is sent to the function `check_infinite_loops()` to analyse the reduced set of connections for infinite loop routing (see next section).

```
reduce_land_connections(data_path, frc_thres) %>%
  check_infinite_loops(., data_path, 'Land')

#> Analyzing land objects for infinite loop routing:
#>  Land object 2 of 5179   Time elapsed: 11S   Time remaining: 9M 36S
#>
#> Completed 5179 Land objects in 11M 1S
#>
#>   X  198 Land objects identified where water is routed in loops.
#>
#>  You can resolve this issue in the following ways:
#>
#> - Use the layer 'land_infinite_loops.gpkg' that was written to
#>   'Define:/your/path/demo_project/data/vector
#>    to identify land polygons that cause the issue and split them to break the
#>    loops.
#>    This would require to restart the entire model setup procedure!
#>
#>  - Increase the value of 'frc_thres'.
#>    This reduces the number of connections of each land unit (maybe undesired!)
#>    and can remove the connections that route the water in loops.
#>
#>  - Continue with the model setup (only recommended for small number of identified
↪
#>  units!).
#>    The function 'resolve_loop_issues()' will then eliminate a certain number of
#>    connections.
```

2.3.6.4 Check and resolve infinite loops

A SWAT+ model setup must not have any infinite loop connections. If infinite loops are present in the connections between spatial objects of a SWAT+ model setup, executing the model will trigger an error. Thus, all infinite loops must be identified and resolved before the model setup process can be continued.

The routine `check_infinite_loops()` uses the reduced land object connection table and propagates the fluxes starting from each spatial object into the receiving objects. This procedure is an iterative process and the fluxes are further propagated through the spatial object until a flux reaches a sink object (e.g. a channel, reservoir, or a land sink object). In the example above 198 land objects were identified which are part of infinite loops. If infinite loops were identified, the identified land objects are written to the layer *'land_infinite_loops.gpkg'* into *'./data/vector'*

If `check_infinite_loops()` identifies infinite loops the routine `resolve_loop_issues()` tries to resolve the identified infinite loops by eliminating individual connections between spatial objects. The routine ranks the connections which were identified to be part of infinite loops based on the flow fraction (to more likely remove connections with low flow fractions), the number of times a certain connection was identified in loops (the removal of one connection can then resolve several loops), and the area of spatial objects (the impact of smaller objects is lower to the overall result than that of larger ones). Iteratively, connections are eliminated from the connections table until no infinite loops are present in the connections. The routine prints the eliminated connections into the console, so that the user can follow the process of the elimination, as shown in the example below. In the example 55 connections were eliminated with flow fractions lower than 1. Unfortunately, also eight connections had to be removed with flow fractions of 1 which therefore generated sink land objects. Yet, given the total number of 5179 land objects and a total number of 16110 connections between land objects in this example eliminating 63 connections and creating eight sink units is an acceptable modification of the overall object connectivity.

```
resolve_loop_issues(data_path)

#> Trying to resolve the identified infinite loops by removing connections...
#>     Removing connection ID  4113 to ID  2151 (fraction = 0.073)
#>     Removing connection ID   126 to ID   125 (fraction = 0.129)
#>     Removing connection ID  1173 to ID   637 (fraction = 0.152)
#>     Removing connection ID  1413 to ID   705 (fraction = 0.152)
#>     Removing connection ID  4564 to ID  4100 (fraction = 0.139)
#>     Removing connection ID  1086 to ID  2649 (fraction = 0.203)
#>     Removing connection ID  2321 to ID  2318 (fraction = 0.236)
#>     Removing connection ID  4113 to ID   891 (fraction = 0.154)
#>     Removing connection ID  1830 to ID  4948 (fraction = 0.153)
#>     Removing connection ID  3099 to ID  1168 (fraction = 0.179)
#>     Removing connection ID  1797 to ID  1116 (fraction = 0.216)
#>     Removing connection ID  1435 to ID  3935 (fraction = 0.196)
#>     Removing connection ID  5120 to ID  5122 (fraction = 0.189)
#>     Removing connection ID  3332 to ID  1350 (fraction = 0.203)
#>     Removing connection ID  1912 to ID  5045 (fraction = 0.219)
#>     Removing connection ID  2125 to ID  2513 (fraction = 0.241)
#>     Removing connection ID   544 to ID  1236 (fraction = 0.244)
#>     Removing connection ID  2406 to ID  3582 (fraction = 0.238)
#>     Removing connection ID  3075 to ID  3074 (fraction = 0.245)
#>     Removing connection ID  3881 to ID   668 (fraction = 0.246)
#>     Removing connection ID  3071 to ID  1958 (fraction = 0.371)
#>     Removing connection ID  1426 to ID  3548 (fraction = 0.231)
#>     Removing connection ID  3838 to ID  1665 (fraction = 0.224)
#>     Removing connection ID  3144 to ID  3532 (fraction = 0.243)
#>     Removing connection ID  3597 to ID  3943 (fraction = 0.328)
#>     Removing connection ID  4866 to ID  4863 (fraction = 0.269)
#>     Removing connection ID  1608 to ID  1606 (fraction = 0.261)
#>     Removing connection ID  1279 to ID  2859 (fraction = 0.287)
#>     Removing connection ID  5134 to ID  5133 (fraction = 0.282)
#>     Removing connection ID  3084 to ID   598 (fraction = 0.276)
#>     Removing connection ID  1797 to ID  1793 (fraction = 0.277)
#>     Removing connection ID  4499 to ID  1007 (fraction = 0.203)
#>     Removing connection ID  2351 to ID   923 (fraction = 0.312)
#>     Removing connection ID  3310 to ID   860 (fraction = 0.333)
#>     Removing connection ID  1825 to ID  4535 (fraction = 0.336)
#>     Removing connection ID  1760 to ID  4163 (fraction = 0.342)
#>     Removing connection ID  1946 to ID  4472 (fraction = 0.323)
#>     Removing connection ID   336 to ID   887 (fraction = 0.352)
#>     Removing connection ID  2682 to ID  2681 (fraction = 0.448)
#>     Removing connection ID   349 to ID  1280 (fraction = 0.381)
#>     Removing connection ID   521 to ID  4144 (fraction = 0.487)
#>     Removing connection ID  4361 to ID  4499 (fraction = 0.406)
#>     Removing connection ID  2310 to ID  3862 (fraction = 0.448)
#>     Removing connection ID  2086 to ID  2110 (fraction = 0.429)
#>     Removing connection ID  4193 to ID  1775 (fraction = 0.446)
#>     Removing connection ID  2385 to ID   887 (fraction = 0.399)
#>     Removing connection ID  1413 to ID  4982 (fraction = 0.471)
#>     Removing connection ID   359 to ID  2682 (fraction = 0.465)
#>     Removing connection ID   126 to ID   141 (fraction = 0.452)
#>     Removing connection ID  4872 to ID  4878 (fraction = 0.533)
#>     Removing connection ID   514 to ID  3862 (fraction = 0.623)
#>     Removing connection ID  4479 to ID  1826 (fraction = 0.516)
#>     Removing connection ID  2862 to ID   731 (fraction = 0.669)
#>     Removing connection ID  2408 to ID    11 (fraction = 0.697)
#>     Removing connection ID  2943 to ID  3656 (fraction = 0.52)
#>     Removing connection ID  4162 to ID  1760 (fraction = 1 CAUTION sink unit!)
#>     Removing connection ID  3696 to ID  2617 (fraction = 1 CAUTION sink unit!)
#>     Removing connection ID  4766 to ID  4558 (fraction = 1 CAUTION sink unit!)
#>     Removing connection ID   638 to ID  2762 (fraction = 1 CAUTION sink unit!)
#>     Removing connection ID  2784 to ID  1248 (fraction = 1 CAUTION sink unit!)
#>     Removing connection ID  1770 to ID  1221 (fraction = 1 CAUTION sink unit!)
#>     Removing connection ID  4647 to ID  5146 (fraction = 1 CAUTION sink unit!)
#>     Removing connection ID  4873 to ID  3315 (fraction = 1 CAUTION sink unit!)
#>
#> 55 connections removed.
#>  The land units for which connections were removed were written into the layer
#>  'removed_connection' in the file 'resolve_loops.gpkg' saved in
#>  Define:/your/path/demo_project/data/vector
#>
#> 8 connections removed which created land units without connections!
#>  The land units that caused the issue were written into the layer
#>  'create_sink_connection' in the file 'resolve_loops.gpkg' saved in
#>  Define:/your/path/demo_project/data/vector
#>  These issues have to be resolved manually in the land input layer!
```

Figure 2.19 shows two examples with land objects that are part of infinite loops. The land objects are shown after the infinite loops were resolved with `resolve_loop_issues()`. The connections for green objects were successfully eliminated. Red polygons show land objects which became sink units after the elimination of their connection. As with the identified sink objects, infinite loops are often generated along road dams where the fluxes would be “trapped” between elevated artificial structures if no artificial drainage would be present. Thus, also some of the infinite loops can be resolved by adding drainage ditches to the channel input layer (e.g. Figure 2.19 a)). Other situations (e.g. Figure 2.19 b)) are not so clear to resolve. In some situations for example flow trajectories (in the flow accumulation layer) are visible but stop in the middle of a land object and therefore can be the reason for unresolvable loop routing or sink units. Non-continuing flow paths can result from imperfect hydrological conditioning of the `dem` layer or from spatial features in the terrain in general. Such situations can only be resolved manually.

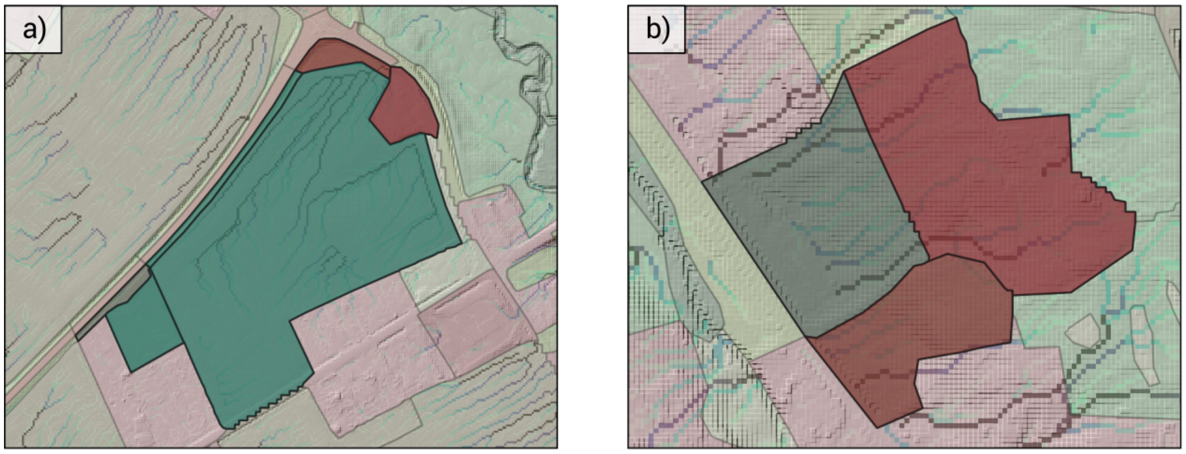


Figure 2.19: Examples for land objects which are part of infinite loops. Grey objects were kept after resolving the loops. Connections for green objects were successfully eliminated. Red polygons show land objects which became sink units after the elimination of their connection. a) Several patches between road dams. b) Triplet of land objects which only route into their neighbour and will in any case generate sink units when eliminating connections.

After eliminating connections which cause loop routing the reduced connections table is written into the *'tables.sqlite'* data base with the table name *'rtu.connect_ids'*. This table will be further used to generate the routing unit connections table *'rout_unit.con'* of the SWAT+ model setup.

2.3.7 Water object connectivity

The connections between water objects were already partly analysed after reading the channel data and combining the channels with the extracted reservoir objects to form the water object network (see section 2.3.4). Also the connection between the water objects is only dependent on the intersection of channel and reservoir objects and the direction of channel objects. `build_water_object_connectivity()` transforms the `cha` and `res` vector input layers and generates the SWAT+ connectivity input files for the channel and the reservoir objects. The generated SWAT+ input tables are written into the *'tables.sqlite'* database with the table name *'cha.chandeg_con_out'* and *'res.reservoir_con_out'*, respectively.

```
build_water_object_connectivity(data_path)
```

The example table below shows the *'cha.chandeg_con_out'* table for the model setup which is used in this demo.

```
#> # A tibble: 1,000 × 7
#>       id order obj_typ obj_id hyd_typ  frac chandeg_con_id
#>    <int> <int> <chr>    <int> <chr>   <dbl>          <int>
#>  1     1     1 res         13 tot         1              1
#>  2     2     1 sdc        141 tot         1              2
#>  3     3     1 sdc        268 tot         1              3
#>  4     4     1 sdc        537 tot         1              4
#>  5     5     1 sdc          4 tot         1              5
#>  6     6     1 sdc          3 tot         1              6
#>  7     7     1 sdc        641 tot         1              7
#>  8     8     1 sdc        646 tot         1              8
#>  9     9     1 sdc         84 tot         1              9
#> 10    10     1 sdc        795 tot         1             10
#> # ... with 990 more rows
```

2.3.7.1 Check for infinite loop routing

Although the connectivity between water objects was already analysed an analysis for infinite loop routing was not performed yet. With the same approach as for the land objects the `check_infinite_loops()` routine is used to propagate the fluxes between the water objects to identify whether any flux returns back to one of the sending objects. The analysis of any potential issues is similar as with the land objects. Before using `check_infinite_loops()` with the water objects the connectivity tables *'cha.chandeg_con_out'* and *'res.reservoir_con_out'*, which resulted from `build_water_object_connectivity()`, are processed with `prepare_water_links()` to get the data format which is required for the infinite loops check.

```
prepare_water_links(data_path) %>%
  check_infinite_loops(., data_path, 'Water', Inf)
#> Analyzing water objects for infinite loop routing:
#>  Water object 2 of 996   Time elapsed: 7S   Time remaining: 15M 58S
```

2.3.7.2 Terrain information for water objects

As the connectivity calculation for water objects does not require any terrain information, all calculations of terrain properties for channels and reservoir objects are preformed after defining the connectivity and checking for infinite loops. Reservoir objects only require elevation information. Channels, however, additionally require channel slope and the channels' contributing areas as further information. The contributing areas are further used to provide first estimates for the channel depths, channel widths and their ratios. The calculation of channel depths and widths is performed using the equations derived by Bieger et al. (2015) for natural streams in the conterminous US. As the equations have been specifically derived for US rivers, the calculated values should only be considered as first estimates and must be revisited by the user. Further, the estimates will very likely be incorrect for artificial channels. These must be adjusted in a later step of the model parametrization.

As the used channel input layer and the DEM must not necessarily hydrologically correspond, a workaround was necessary to get acceptable estimates for the channel contributing areas. For the estimation iso-basins (basins of equal size) with an approximate area of $500m^2$ are computed based on the DEM with the `whitebox` function `wbt_isobasins()`. For these basins the routing and their flow accumulation is calculated. The iso-basins are intersected with the channels and the contributing area of the channels is estimated based on the distribution of the flow accumulation values of the channel/iso-basin intersection. Although these estimates may not be the exact contributing areas of the channels, it is more robust when the channel line features and the flow accumulation which results from the used DEM do not perfectly match. Yet in any case, contributing areas and estimated channel dimensions must be checked manually e.g. when working with the SWAT+Editor or the model text input files.

```
prepare_terrain_water(data_path)

#> Calculating iso-basins from catchment DEM...
#>   v  Done
#> Calculating flow accumulation for iso-basins...
#>   v  Done
#> Generating flow accumulation raster...
#>   v  Done
#> Calculating channel contributing areas...
#>   v  Done
```

2.3.8 Generate SWAT+ land object inputs

SWAT+ land objects require information on soil, soil water retention, topography, and land use. These information are distributed over several SWAT+ input files. Topographical information for land objects such as slope (`slp`) or slope length (`slp_len`) is parameterized in the file *'topography.hyd'*. Hydrological characteristics of all land surfaces are defined in *'hydrology.hyd'*. The soil parametrization of a model setup is given in the input file *'soils.sol'*. Land use and land management are defined with the input file *'landuse.lum'*, which again points to several parameter files which define the plant communities for a land use (*'plant.ini'*), the land management (*'management.sch'*), land use/management/soil specific curve number values (*'cntable.lum'*), or any conservation practices (different *'.str'* files).

The `SWATbuildR` model setup processes generates all land object inputs, which can be safely derived from the provided spatial input layers and input tables. Other inputs such as the user defined land use label can be rather ambiguous and a certain land management cannot be delineated from the land use labels (only a few labels such as `watr` do have a clearly defined function otherwise the user is free to define land use labels). Therefore, most land use related parametrization is left empty and has to be defined by the user in a later step of the model setup process.

This section documents the SWAT+ input tables which are generated from the input data and which are required to define the `hru` land objects of a SWAT+ model setup.

2.3.8.1 Soil inputs

Up to this point in the model development only the soil raster layer was read, which defines the spatial location of soil classes. The soil classes must be linked to soil physical and chemical properties. Other SWAT model setup tools, such as QSWAT+, require a lookup table that links the raster integer values with names of soil classes and a soil data input table, which provides soil parameterisations for the different layers of the soil classes. Often the soil data for SWAT projects are already available in this specific format. Thus, `SWATbuildR` uses the same data structure for the soil input data (see section 2.2.2 for further information).

The function `build_soil_data()` prepares the soil information and arranges the data format that is used by SWAT+Editor *sqlite* data bases. The routine reads the unique soil ids that were assigned to the HRUs in `hru_terrain_soil` and extracts the required information from the soil lookup table (which must be available in `soil_lookup_path`) and the soil data table (which must be available in `soil_data_path`). `build_soil_data()` generates four soil tables and writes them into the *tables.sqlite* data base for their use in the further model setup. `soil.lookup` stores the user provided soil lookup table. `soil.soils_sol` and `soil.soils_sol_layer` provide the soil data in the SWAT+Editor format and `soil.ids` links the HRU `ids` with the soil ids.

```
build_soil_data(soil_lookup_path, soil_data_path, data_path)
```

2.3.8.2 Land use inputs

For land uses only a very basic setup is performed with `SWATbuildR`. The routine `build_landuse()` reads the `hru.shp` layer from the vector data in the `data_path` and prepares a blank *'landuse.lum'* table for all unique land uses which were provided with the `type` attribute of the land object input layer. Land uses which are assigned to land objects with the tile drainage option activated (see section 2.2.4.4) are duplicated and the suffix *'_drn'* is added to the land use label. To these land uses a default parametrization for tile drainage is assigned in the column `tile_id`. To all land uses a default plant community is assigned in the column `plnt_com_id` of *'landuse.lum'* except to land uses that use generic SWAT+ urban land use labels. These land uses do not receive an initial plant community (this has to be checked and verified by the user e.g. with the SWAT+Editor when spatial objects are further parameterized). All other fields of *'landuse.lum'* are left empty and have to be set by the user e.g. with the SWAT+Editor when spatial objects are further parameterized.

```
build_landuse(data_path)
```

The table below shows an example for the `landuse.landuse_lum` table that is written to the *tables.sqlite* database. It shows examples for two land uses (`agrl1` and `agrl2`) which were assigned to land objects in the land input layer with the drainage option activated. These two land uses are duplicated in the `landuse.landuse_lum` with the suffix *'_drn'*. The land uses `urml` and `utrn` do not have initial plant communities assigned as they are default SWAT+ urban land uses.

```
#>      id name           cal_group plnt_com_id mgt_id cn2_id cons_prac_id urban_id
↪  urb_ro ov_mann_id tile_id sep_id vfs_id grww_id bmp_id description
#>    <dbl> <chr>              <int>       <dbl>  <int>  <int>        <int>    <int>
↪  <int>      <int>   <dbl>  <int>  <int>   <int>  <int> <chr>
#> 1      1 agrl1_lum             NA           1     NA     NA           NA       NA
↪  NA          NA      NA     NA     NA      NA     NA ""
#> 2      2 agrl1_drn_lum         NA           1     NA     NA           NA       NA
↪  NA          NA       1     NA     NA      NA     NA ""
#> 3      3 agrl2_lum             NA           1     NA     NA           NA       NA
↪  NA          NA      NA     NA     NA      NA     NA ""
#> 4      4 agrl2_drn_lum         NA           1     NA     NA           NA       NA
↪  NA          NA       1     NA     NA      NA     NA ""
#> 5      5 agrl3_lum             NA           1     NA     NA           NA       NA
↪  NA          NA      NA     NA     NA      NA     NA ""
#> 6      6 agrl4_lum             NA           1     NA     NA           NA       NA
↪  NA          NA      NA     NA     NA      NA     NA ""
#> 7      7 frsd_lum              NA           1     NA     NA           NA       NA
↪  NA          NA      NA     NA     NA      NA     NA ""
#> 8      8 frse_lum              NA           1     NA     NA           NA       NA
↪  NA          NA      NA     NA     NA      NA     NA ""
#> 9      9 frst_lum              NA           1     NA     NA           NA       NA
↪  NA          NA      NA     NA     NA      NA     NA ""
#> 10    10 past_lum              NA           1     NA     NA           NA       NA
↪  NA          NA      NA     NA     NA      NA     NA ""
#> 11    11 rngb_lum              NA           1     NA     NA           NA       NA
↪  NA          NA      NA     NA     NA      NA     NA ""
#> 12    12 shrb_lum              NA           1     NA     NA           NA       NA
↪  NA          NA      NA     NA     NA      NA     NA ""
#> 13    13 urml_lum              NA          NA     NA     NA           NA       NA
↪  NA          NA      NA     NA     NA      NA     NA ""
#> 14    14 utrn_lum              NA          NA     NA     NA           NA       NA
↪  NA          NA      NA     NA     NA      NA     NA ""
#> 15    15 wehb_lum              NA           1     NA     NA           NA       NA
↪  NA          NA      NA     NA     NA      NA     NA ""
#> 16    16 wetf_lum              NA           1     NA     NA           NA       NA
↪  NA          NA      NA     NA     NA      NA     NA ""
```

2.3.8.3 hru object inputs

The function `build_hru_input()` generates the `hru` object specific input tables `hru.hru_con`, `hru.hru_data_hru`, `hru.topography_hyd`, `hru.hydrology_hyd`, and `hru.topo_id` and writes them into the *tables.sqlite* data base.

```
build_hru_input(data_path)
```

`hru.hru_con` is the SWAT+Editor formatted file that will be translated to the *'hru.con'* in the final SWAT+ model setup. In the `SWATbuildR` model setup no connectivity is defined for `hru`s, but each HRU is assigned to a single routing unit. Thus, the essential information that is provided in `hru.hru_con` are the `area`, the elevation (`elev`) and the coordinates of the centroid point of each object (`lat`, `lon`). As for all other objects no weather stations are linked with the spatial objects. The link to weather station data will be assigned with the SWAT+Editor at a later step of the model setup procedure.

```
#> # A tibble: 373 × 12
#>       id name   gis_id  area   lat   lon  elev wst_id cst_id  ovfl  rule hru_id
#>    <int> <chr>   <int> <dbl> <dbl> <dbl> <dbl>  <int>  <int> <int> <int>  <int>
#>  1     1 hru001     NA  6.47  31.8 -83.8  141.     NA     NA     0     0      1
#>  2     2 hru002     NA 20.9   31.7 -83.8  142.     NA     NA     0     0      2
#>  3     3 hru003     NA  6.50  31.7 -83.8  138.     NA     NA     0     0      3
#>  4     4 hru004     NA 11.5   31.7 -83.8  139.     NA     NA     0     0      4
#>  5     5 hru005     NA  1.63  31.7 -83.8  135.     NA     NA     0     0      5
#>  6     6 hru006     NA  1.42  31.7 -83.8  138.     NA     NA     0     0      6
#>  7     7 hru007     NA  8.18  31.7 -83.8  134.     NA     NA     0     0      7
#>  8     8 hru008     NA 22.1   31.7 -83.8  141.     NA     NA     0     0      8
#>  9     9 hru009     NA  5.20  31.7 -83.8  139.     NA     NA     0     0      9
#> 10    10 hru010     NA 10.1   31.7 -83.8  133.     NA     NA     0     0     10
#> # ... with 363 more rows
```

`hru.hru_data_hru` is the SWAT+Editor formatted file that will be translated to the *'hru-data.hru'* in the final SWAT+ model setup. Each line defines an `hru` object and points to parametrizations of an `hru`'s topography (`topo_id` points to positions in `hru.topography_hyd`), hydrology (`hydro_id` points to positions in `hru.hydrology_hyd`), soil layers (`soil_id` points to positions in `soil.soils_sol`), and the land use (`lu_mgt_id` points to positions in `landuse.landuse_lum`). The integer id values in the example below point to the `ids` in the respective input tables.

```
#> # A tibble: 373 × 11
#>       id name   topo_id hydro_id soil_id lu_mgt_id soil_plant_init_id
↪  surf_stor_id snow_id field_id description
#>    <dbl> <chr>    <int>    <int>   <int>     <int>              <int>
↪  <int>   <int>    <int> <chr>
#>  1     1 hru001       1        1       5         6                 NA
↪  NA        1       NA ""
#>  2     2 hru002       2        2       5         1                 NA
↪  NA        1       NA ""
#>  3     3 hru003       3        3       5         2                 NA
↪  NA        1       NA ""
#>  4     4 hru004       4        4       5         1                 NA
↪  NA        1       NA ""
#>  5     5 hru005       5        5       8        10                 NA
↪  NA        1       NA ""
#>  6     6 hru006       6        6       5         3                 NA
↪  NA        1       NA ""
#>  7     7 hru007       7        7       5         3                 NA
↪  NA        1       NA ""
#>  8     8 hru008       8        8       5         6                 NA
↪  NA        1       NA ""
#>  9     9 hru009       9        9       5         5                 NA
↪  NA        1       NA ""
#> 10    10 hru010      10       10       5         3                 NA
↪  NA        1       NA ""
#> # ... with 363 more rows
```

`hru.topography_hyd` is the SWAT+Editor formatted file that will be translated to the *'topography.hyd'* in the final SWAT+ model setup. In the current version of `SWATbuildR` most of the topographic parameters are default values. Only the slope (`slp`) values are the mean slopes of the land objects that were derived from the zonal statistics of the DEM.

```
#>       id name          slp slp_len lat_len dist_cha depos type
#>    <int> <chr>       <dbl>   <dbl>   <dbl>    <dbl> <dbl> <chr>
#>  1     1 topohru001 0.0265      30      30      121     0 hru
#>  2     2 topohru002 0.0266      30      30      121     0 hru
#>  3     3 topohru003 0.0212      30      30      121     0 hru
#>  4     4 topohru004 0.0348      30      30      121     0 hru
#>  5     5 topohru005 0.0311      30      30      121     0 hru
#>  6     6 topohru006 0.0263      30      30      121     0 hru
#>  7     7 topohru007 0.0258      30      30      121     0 hru
#>  8     8 topohru008 0.0257      30      30      121     0 hru
#>  9     9 topohru009 0.0333      30      30      121     0 hru
#> 10    10 topohru010 0.0324      30      30      121     0 hru
#> # ... with 736 more rows
```

`hru.hydrology_hyd` is the SWAT+Editor formatted file that will be translated to the *'hydrology.hyd'* in the final SWAT+ model setup. For almost all parameters the same initial parameter values were used which are also assigned by QSWAT+. The parameters `cn3_swf`, `perco`, and `latq_co` receive different initial parameter values, with respect to the runoff and leaching potential of an HRU. All HRUs are classified to have a high, moderate, or low leaching potential and a high, moderate, or low runoff potential. The classification was performed according to (Thompson et al., 2020), which is based on the mean slope and the hydrologic soil group of an HRU. Low and high leaching potentials translate to a small (0.05) and large (0.90) initial values for `perco`. Low and high runoff potentials translate to small (0.01) and large (0.90) initial values for `latq_co` and large (0.95) and small (0.00) initial values for `perco`, respectively. The example below shows that most parameters were initialized with the same default values, except for `cn3_swf`, `perco`, and `latq_co`, which were initialized based on the HRU's leaching and runoff potentials.

```
#> # A tibble: 373 × 16
#>       id name   lat_ttime lat_sed can_max  esco  epco orgn_enrich orgp_enrich
↪  cn3_swf bio_mix perco lat_orgn lat_orgp harg_pet latq_co
#>    <int> <chr>      <dbl>   <dbl>   <dbl> <dbl> <dbl>       <dbl>       <dbl>
↪  <dbl>   <dbl> <dbl>    <dbl>    <dbl>    <dbl>   <dbl>
#> 1      1 hyd001         0       0       1  0.95     1           0           0
↪  0.95      0.2  0.9        0        0        0    0.01
#> 2      2 hyd002         0       0       1  0.95     1           0           0
↪  0.95      0.2  0.9        0        0        0    0.01
#> 3      3 hyd003         0       0       1  0.95     1           0           0
↪  0.95      0.2  0.05       0        0        0    0.01
#> 4      4 hyd004         0       0       1  0.95     1           0           0
↪  0.95      0.2  0.9        0        0        0    0.01
#> 5      5 hyd005         0       0       1  0.95     1           0           0
↪  0.3       0.2  0.05       0        0        0    0.2
#> 6      6 hyd006         0       0       1  0.95     1           0           0
↪  0.95      0.2  0.9        0        0        0    0.01
#> 7      7 hyd007         0       0       1  0.95     1           0           0
↪  0.95      0.2  0.9        0        0        0    0.01
```

```
#>  8      8 hyd008          0        0        1  0.95     1            0            0
↪  0.95      0.2  0.9          0        0        0    0.01
#>  9      9 hyd009          0        0        1  0.95     1            0            0
↪  0.95      0.2  0.9          0        0        0    0.01
#> 10     10 hyd010          0        0        1  0.95     1            0            0
↪  0.95      0.2  0.9          0        0        0    0.01
#> # ... with 363 more rows
```

2.3.9 Generate SWAT+ water object inputs

The SWAT+ water objects channel and reservoir are basically defined by three inputs; (1) an object pointer file which defines the objects and points to other inputs (*'channel-lte.cha'*, and *'reservoir.res'*), (2) a connectivity file which defines the connectivity to other objects (*'chandeg.con'*, and *'reservoir.con'*), and (3) an input file, which defines the objects' geometries and hydrological properties (*'hyd-sed-lte.cha'*, and *'hydrology.res'*). There are other inputs that for example initialize nutrient concentrations or the water level in a reservoir. These inputs will be initialized with default parameterisations by `SWATbuildR`.

2.3.9.1 Channel inputs (`cha`)

The channel inputs are prepared in the SWAT+ input table format with `build_cha_input()`. The function uses the channel properties which were generated with `prepare_terrain_water()` that are the mean channel elevation (`elev`), channel length (`len`), contributing area (`area`), slope (`slp`), channel width (`wd`), channel depth (`dp`) and the width depth ratio (`wd_rto`). Other channel parameters are parameterized with default values as they are also used by QSWAT+ and the SWAT+Editor.

```
build_cha_input(data_path)
```

The prepared SWAT+ channel input tables are written to the *tables.sqlite* data base with the names *'cha.channel_lte_cha'*, *'cha.chandeg_con'*, and *'cha.hyd_sed_lte_cha'*. The examples below show the data structure of these input tables for a demo model setup.

```
# cha.channel_lte_cha

#> # A tibble: 52 × 7
#>       id name   init_id hyd_id sed_id nut_id description
#>    <int> <chr>    <int>  <int>  <int>  <int> <chr>
#>  1     1 cha01        1      1     NA      1 ""
#>  2     2 cha02        1      2     NA      1 ""
#>  3     3 cha03        1      3     NA      1 ""
#>  4     4 cha04        1      4     NA      1 ""
#>  5     5 cha05        1      5     NA      1 ""
#>  6     6 cha06        1      6     NA      1 ""
#>  7     7 cha07        1      7     NA      1 ""
#>  8     8 cha08        1      8     NA      1 ""
#>  9     9 cha09        1      9     NA      1 ""
#> 10    10 cha10        1     10     NA      1 ""
#> # ... with 42 more rows
```

```
# cha.chandeg_con

#> # A tibble: 52 × 12
#>       id name  gis_id   area   lat   lon  elev wst_id cst_id  ovfl  rule lcha_id
#>    <int> <chr>  <int>  <dbl> <dbl> <dbl> <dbl>  <int>  <int> <int> <int>   <int>
#>  1     1 cha01     NA  8.87   31.7 -83.7  117.     NA     NA     0     0       1
#>  2     2 cha02     NA 17.8    31.7 -83.7  113.     NA     NA     0     0       2
#>  3     3 cha03     NA  0.386  31.7 -83.7  118.     NA     NA     0     0       3
#>  4     4 cha04     NA 10.1    31.7 -83.7  115.     NA     NA     0     0       4
#>  5     5 cha05     NA 21.1    31.7 -83.7  111.     NA     NA     0     0       5
#>  6     6 cha06     NA  0.483  31.7 -83.7  124.     NA     NA     0     0       6
#>  7     7 cha07     NA  0.430  31.7 -83.7  120.     NA     NA     0     0       7
#>  8     8 cha08     NA 10.8    31.7 -83.7  114.     NA     NA     0     0       8
#>  9     9 cha09     NA 22.3    31.7 -83.7  110.     NA     NA     0     0       9
#> 10    10 cha10     NA 25.3    31.7 -83.7  107.     NA     NA     0     0      10
#> # ... with 42 more rows
```

```
# cha.hyd_sed_lte_cha

#> # A tibble: 52 × 25
#>       id name     order    wd     dp     slp     len  mann     k erod... cov_f...
↪  wd_rto
#>    <int> <chr>    <chr> <dbl>  <dbl>   <dbl>   <dbl> <dbl> <dbl>   <dbl>   <dbl>
↪  <dbl>
#>  1     1 hydcha01 ""    1.15  0.179  1.38e-3 0.287   0.05     1    0.01   0.005
↪  6.43
#>  2     2 hydcha02 ""    1.47  0.208  2.24e-3 1.10    0.05     1    0.01   0.005
↪  7.08
#>  3     3 hydcha03 ""    0.382 0.0918 1.32e-2 0.962   0.05     1    0.01   0.005
↪  4.16
#>  4     4 hydcha04 ""    1.20  0.184  2.82e-3 1.04    0.05     1    0.01   0.005
↪  6.54
#>  5     5 hydcha05 ""    1.56  0.215  1.94e-4 0.970   0.05     1    0.01   0.005
↪  7.25
#>  6     6 hydcha06 ""    0.413 0.0963 1.34e-2 0.967   0.05     1    0.01   0.005
↪  4.29
#>  7     7 hydcha07 ""    0.397 0.0940 1.17e-2 1.31    0.05     1    0.01   0.005
↪  4.22
#>  8     8 hydcha08 ""    1.23  0.187  6.96e-4 0.0762  0.05     1    0.01   0.005
↪  6.61
#>  9     9 hydcha09 ""    1.59  0.218  2.96e-3 0.872   0.05     1    0.01   0.005
↪  7.30
#> 10    10 hydcha10 ""    1.66  0.224  1.68e-3 1.67    0.05     1    0.01   0.005
↪  7.43
#> # ... with 42 more rows, 12 more variables: eq_slp <dbl>, d50 <dbl>, clay <dbl>,
↪  carbon ,
#> #   dry_bd <dbl>, side_* <dbl>, bed_l* <dbl>, fps <dbl>, fpn <dbl>, p_conc <dbl>,
↪
#> #   p_bio <dbl>, description <chr>
```

2.3.9.2 Reservoir inputs (res)

The reservoir inputs are prepared in the SWAT+ input table format with `build_res_input()`. The function calculates the reservoir area (`area`) based on the size of the reservoir polygons. `SWATbuildR` provides very simplistic estimates for the reservoir areas and volumes: for the condition that the the principal spillway is reached `area_ps = area` and `vol_ps  = area_ps * 10`. The areas and volumes at emergency spillway estimated with `area_es = area_ps * 1.15` and `vol_es  = area_es* 10`. Other reservoir parameters are parameterized with default values as they are also used by QSWAT+ and the SWAT+Editor.

```
build_res_input(data_path)
```

The prepared SWAT+ reservoir input tables are written to the *tables.sqlite* data base with the names *'res.reservoir_res'*, *'res.reservoir_con'*, and *'res.hydrology_res'*. The examples below show the data structure of these input tables for a demo model setup.

```
# res.reservoir_res

#> # A tibble: 22 × 8
#>       id name  init_id hyd_id rel_id sed_id nut_id description
#>    <int> <chr>   <int>  <int>  <int>  <int>  <int> <chr>
#>  1     1 res01       1      1     39      1      1 ""
#>  2     2 res02       1      2     39      1      1 ""
#>  3     3 res03       1      3     39      1      1 ""
#>  4     4 res04       1      4     39      1      1 ""
#>  5     5 res05       1      5     39      1      1 ""
#>  6     6 res06       1      6     39      1      1 ""
#>  7     7 res07       1      7     39      1      1 ""
#>  8     8 res08       1      8     39      1      1 ""
#>  9     9 res09       1      9     39      1      1 ""
#> 10    10 res10       1     10     39      1      1 ""
#> # ... with 12 more rows
```

```
# res.reservoir_con

#> # A tibble: 22 × 12
#>       id name  gis_id    area   lat   lon  elev wst_id cst_id  ovfl  rule res_id
#>    <int> <chr>  <int>   <dbl> <dbl> <dbl> <dbl>  <int>  <int> <int> <int>  <int>
#>  1     1 res01     NA 14.1     31.7 -83.8  131.     NA     NA     0     0      1
#>  2     2 res02     NA  0.0814  31.7 -83.7  141.     NA     NA     0     0      2
#>  3     3 res03     NA 10.6     31.7 -83.8  128.     NA     NA     0     0      3
#>  4     4 res04     NA  3.49    31.7 -83.7  128.     NA     NA     0     0      4
#>  5     5 res05     NA  1.57    31.7 -83.8  132.     NA     NA     0     0      5
#>  6     6 res06     NA  0.614   31.7 -83.8  132.     NA     NA     0     0      6
#>  7     7 res07     NA  1.40    31.7 -83.7  122.     NA     NA     0     0      7
#>  8     8 res08     NA  2.49    31.7 -83.7  122.     NA     NA     0     0      8
#>  9     9 res09     NA  0.922   31.7 -83.7  129.     NA     NA     0     0      9
#> 10    10 res10     NA  1.34    31.7 -83.7  120.     NA     NA     0     0     10
#> # ... with 12 more rows
```

```
# res.hydrology_res

#> # A tibble: 22 × 12
#>       id name  yr_op mon_op area_ps  vol_ps area_es  vol_es     k evap_co shp_co1
↪  shp_co2
#>    <int> <chr> <int>  <int>   <dbl>   <dbl>   <dbl>   <dbl> <dbl>   <dbl>   <dbl>
↪  <dbl>
#> 1      1 res01     1      1 14.1    141.    16.2    162.        0     0.6       0
↪  0
#> 2      2 res02     1      1  0.0814   0.814  0.0936   0.936     0     0.6       0
↪  0
#> 3      3 res03     1      1 10.6    106.    12.2    122.        0     0.6       0
↪  0
#> 4      4 res04     1      1  3.49    34.9    4.01    40.1       0     0.6       0
↪  0
#> 5      5 res05     1      1  1.57    15.7    1.81    18.1       0     0.6       0
↪  0
#> 6      6 res06     1      1  0.614    6.14   0.706    7.06      0     0.6       0
↪  0
#> 7      7 res07     1      1  1.40    14.0    1.60    16.0       0     0.6       0
↪  0
#> 8      8 res08     1      1  2.49    24.9    2.87    28.7       0     0.6       0
↪  0
#> 9      9 res09     1      1  0.922    9.22   1.06    10.6       0     0.6       0
↪  0
#> 10    10 res10     1      1  1.34    13.4    1.54    15.4       0     0.6       0
↪  0
#> # ... with 12 more rows
```

2.3.10 Generate SWAT+ routing unit inputs (`rout_unit`)

A `SWATbuildR` model setup generates a model structure where one HRU is assigned to one routing unit (RTU). The hydrological land phase processes are calculated for the HRUs. The routing units take care of the routing of fluxes from land objects to other spatial objects. As single HRUs and their corresponding RTUs match spatially, most of the spatial properties that where calculated for the HRUs are simply copied and assigned to the corresponding RTUs. Similar to other spatial objects an RTU is defined by an object pointer file, which defines the objects and points to other inputs (*'rout_unit.rtu'*), and a connectivity file, which defines the connectivity to other objects (*'rout_unit.rtu'*). The RTU objects' topographical properties are the same as their corresponding HRUs and were already written in the table *'hru.topography_hyd'*.

2.3.10.1 Processing `rout_unit` connectivity

`build_rout_con_out()` uses the calculated and cleaned land object connectivities which were saved as *'rtu.connect_ids'* and transforms them into the SWAT+Editor input table format. By default all calculated flow fractions are considered to be total runoff (`tot`), which means that the flow fraction applies to surface (`sur`) and lateral flow (`lat`). If a land object uses the tile drainage option the flow fractions are written separately. While again `sur` and `lat` are sent to other spatial objects according to their calculated routing fractions and tile flow (`til`) is sent to the channel which was defined as the recipient of the tile flow (`drainage` attribute in the land input layer).

```
build_rout_con_out(data_path)
```

The processed connection table for routing units is written as *'rtu.rout_unit_con_out'* into the projects *'tables.sqlite'* data base. Below an example for the calculated routing units is illustrated. The example shows the differences in routing with and without the tile flow option used.

```
#> # A tibble: 936 × 7
#>       id order obj_typ obj_id hyd_typ  frac rtu_con_id
#>    <int> <int> <chr>    <dbl> <chr>   <dbl>      <int>
#>  1     1     1 ru         140 tot     1              1
#>  2     2     2 aqu          1 rhg     1              1
#>  3     3     1 ru         140 tot     0.217          2
#>  4     4     2 ru         141 tot     0.396          2
#>  5     5     3 ru         143 tot     0.387          2
#>  6     6     4 aqu          1 rhg     1              2
#>  7     7     1 ru         157 sur     1              3
#>  8     8     2 ru         157 lat     1              3
#>  9     9     3 sdc         26 til     1              3
#> 10    10     4 aqu          1 rhg     1              3
#> # ... with 926 more rows
```

2.3.10.2 `rout_unit` input files

With `build_rout_input()` the routing unit input files are generated in the required SWAT+Editor format. The prepared SWAT+ routing unit input tables are written to the *tables.sqlite* data base with the names *'rtu.rout_unit_rtu'*, *'rtu.rout_unit_con'*, *'rtu.rout_unit_ele'*, and *'rtu.field_fld'*. *'rtu.rout_unit_rtu'* defines the routing units and points to the other files for topography *'hru.topography_hyd'* and the definition of fields *'rtu.field_fld'*. *'rtu.rout_unit_con'* provides some spatial information and links to the connectivity file *'rtu.rout_unit_con_out'*. *'rtu.rout_unit_ele'* defines the elements which are grouped in each routing unit. In the case of a `SWATbuildR` setup only one HRU element is part of an RTU. The examples below show the data structure of these input tables for a demo model setup.

```
build_rout_input(data_path)
```

```
# rtu.rout_unit_rtu

#> # A tibble: 373 × 6
#>       id name   dlr_id topo_id field_id description
#>    <int> <chr>   <int>   <int>    <int> <chr>
#>  1     1 rtu001     NA     374        1 ""
#>  2     2 rtu002     NA     375        2 ""
#>  3     3 rtu003     NA     376        3 ""
#>  4     4 rtu004     NA     377        4 ""
#>  5     5 rtu005     NA     378        5 ""
#>  6     6 rtu006     NA     379        6 ""
#>  7     7 rtu007     NA     380        7 ""
#>  8     8 rtu008     NA     381        8 ""
#>  9     9 rtu009     NA     382        9 ""
#> 10    10 rtu010     NA      383       10 ""
#> # ... with 363 more rows
```

```
#rtu.rout_unit_con

#> # A tibble: 373 × 12
#>       id name   gis_id  area   lat   lon  elev wst_id cst_id  ovfl  rule rtu_id
#>    <int> <chr>   <int> <dbl> <dbl> <dbl> <dbl>  <int>  <int> <int> <int>  <int>
#>  1     1 rtu001     NA  6.47  31.8 -83.8  141.     NA     NA     0     0      1
#>  2     2 rtu002     NA 20.9   31.7 -83.8  142.     NA     NA     0     0      2
#>  3     3 rtu003     NA  6.50  31.7 -83.8  138.     NA     NA     0     0      3
#>  4     4 rtu004     NA 11.5   31.7 -83.8  139.     NA     NA     0     0      4
#>  5     5 rtu005     NA  1.63  31.7 -83.8  135.     NA     NA     0     0      5
#>  6     6 rtu006     NA  1.42  31.7 -83.8  138.     NA     NA     0     0      6
#>  7     7 rtu007     NA  8.18  31.7 -83.8  134.     NA     NA     0     0      7
#>  8     8 rtu008     NA 22.1   31.7 -83.8  141.     NA     NA     0     0      8
#>  9     9 rtu009     NA  5.20  31.7 -83.8  139.     NA     NA     0     0      9
#> 10    10 rtu010     NA 10.1   31.7 -83.8  133.     NA     NA     0     0     10
#> # ... with 363 more rows
```

```
# rtu.rout_unit_ele

#> # A tibble: 373 × 7
#>       id name   rtu_id obj_typ obj_id  frac dlr_id
#>    <int> <chr>   <int> <chr>    <int> <dbl>  <int>
#>  1     1 hru001      1 hru          1     1     NA
#>  2     2 hru002      2 hru          2     1     NA
#>  3     3 hru003      3 hru          3     1     NA
#>  4     4 hru004      4 hru          4     1     NA
#>  5     5 hru005      5 hru          5     1     NA
#>  6     6 hru006      6 hru          6     1     NA
#>  7     7 hru007      7 hru          7     1     NA
#>  8     8 hru008      8 hru          8     1     NA
#>  9     9 hru009      9 hru          9     1     NA
#> 10    10 hru010     10 hru         10     1     NA
#> # ... with 363 more rows
```

```
# rtu.field_fld

#> # A tibble: 373 × 5
#>       id name     len    wd   ang
#>    <int> <chr>  <dbl> <dbl> <dbl>
#>  1     1 fld001   500   100    30
#>  2     2 fld002   500   100    30
#>  3     3 fld003   500   100    30
#>  4     4 fld004   500   100    30
#>  5     5 fld005   500   100    30
#>  6     6 fld006   500   100    30
#>  7     7 fld007   500   100    30
#>  8     8 fld008   500   100    30
#>  9      9 fld009   500   100    30
#> 10     10 fld010   500   100    30
#> # ... with 363 more rows
```

2.3.10.3 Landscape unit input files

For output printing the routing units are interpreted as landscape units. These files are not relevant for the model execution, but only for output printing. For a correct implementation into the SWAT+Editor data base structure these files will be generated as well. The landscape units are defined with a landscape unit definition file *'ls_unit.def'* and a landscape unit elements file *'ls_unit.ele'*. Both input tables are generated with `build_ls_unit_input()` and are written into the projects *'tables.sqlite'* data base.

```
build_ls_unit_input(data_path)
```

2.3.11 Generate SWAT+ aquifer inputs (aqu)

A `SWATbuildR` model setup generates in its current version only one aquifer for the entire basin. The aquifer is defined with the aquifer definition file *'aquifer.aqu'* and the aquifer connectivity file *'aquifer.con'*. Both files are generated for a single aquifer object for the entire catchment area with the function `build_single_aquifer_files()`. All aquifer parameters are default values as they would be also set with QSWAT+ and the SWAT+Editor, except for the aquifer `area`, which is the basin area, and the elevation (`elev`), which is the average basin elevation.

```
build_single_aquifer_files(data_path)
```

The SWAT+ aquifer input tables are written as *'aqu.aquifer_aqu'* and *'aqu.aquifer_con'* into the projects *'tables.sqlite'* data base. Below the definition of a single aquifer for a demo project is shown.

```
#aqu.aquifer_aqu

#> # A tibble: 1 × 18
#>      id name  init_id gw_flo dep_bot dep_wt no3_n sol_p carbon flo_dist bf_max
↪  alpha_bf revap rchg_dp spec_yld hl_no3n flo_min revap_min
#>   <dbl> <chr>   <dbl>  <dbl>   <dbl>  <dbl> <dbl> <dbl>  <dbl>    <dbl>  <dbl>
↪  <dbl> <dbl>   <dbl>    <dbl>   <dbl>   <dbl>     <dbl>
#> 1     1 aqu1        1   0.05      10     10     0     0      0        0      1
↪  0.048  0.02    0.05    0.003       0       5         3
```

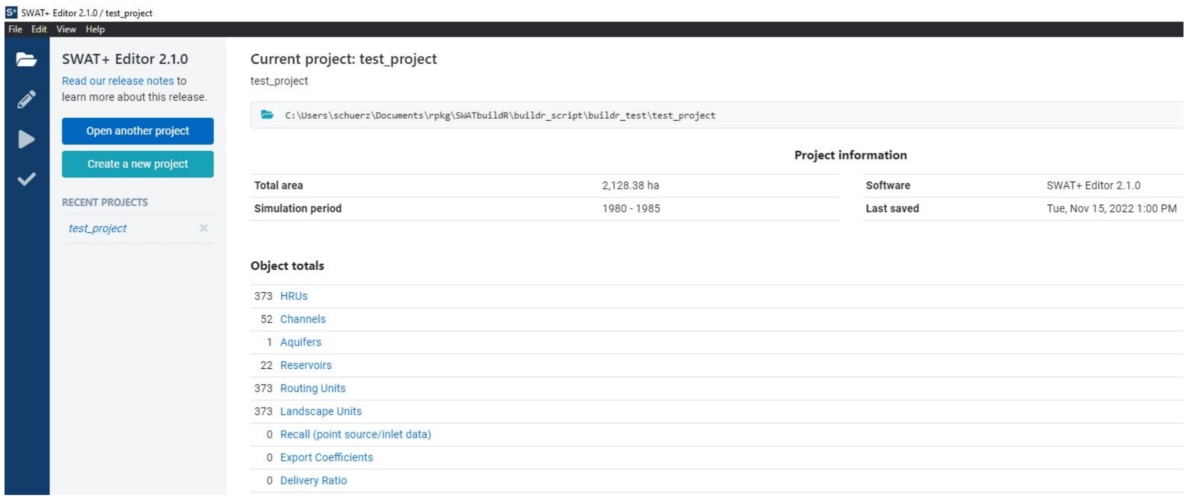
```
#aqu.aquifer_con

#> # A tibble: 1 × 12
#>      id name  gis_id  area   lat   lon  elev wst_id cst_id  ovfl  rule aqu_id
#>   <int> <chr>  <int> <dbl> <dbl> <dbl> <dbl>  <int>  <int> <int> <int>  <int>
#> 1     1 aqu1      NA 2203.  31.7 -83.7  128.     NA     NA     0     0      1
```

2.3.12 Generating the SWAT+Editor data base

After preparing all model input files they are written into an *'.sqlite'* database, which can be opened with the SWAT+Editor for further editing of the SWAT+ model setup. `create_swatplus_database()` reads all prepared input tables from *'tables.sqlite'* and generates a SWAT+Editor project in the `project_path` with the `project_name` (in this case e.g *'Define:/your/new/path/demo_project.sqlite'*).

```
create_swatplus_database(project_path, project_name)
```

Figure 2.20 shows the generated SWAT+Editor database of a **`SWATbuildR`** project after loading it with the SWAT+Editor. It shows the typical structure of a **`SWATbuildR`** project, which has the same number of HRUs and routing units and only one aquifer.



Current project: test_project

test_project

C:\Users\schuerz\Documents\rpkg\SWATbuildR\buildr_script\buildr_test\test_project

Project information

Total area	2,128.38 ha	Software	SWAT+ Editor 2.1.0
Simulation period	1980 - 1985	Last saved	Tue, Nov 15, 2022 1:00 PM

Object totals

373	HRUs
52	Channels
1	Aquifers
22	Reservoirs
373	Routing Units
373	Landscape Units
0	Recall (point source/inlet data)
0	Export Coefficients
0	Delivery Ratio

Figure 2.20: Screenshot of **`SWATbuildR`** project after writing the SWAT+Editor project data base and loading it in the SWAT+Editor v2.1.0.

2.4 Weather data

`SWATbuildR` does not have an option for loading weather data. However, it could be done either using the SWAT+ Editor for this task (see guidelines) or the R package **`svatools`** (description in this chapter and online documentation). Difference between two options is that SWAT+ Editor requires users to prepare all weather files according to required structure and calculate weather generator[2] parameters by themselves, while functions in **`svatools`** requires users to put weather data in simple Excel template and the rest is taken care of by provided functions.

This section discusses issues related to using the observed weather data for calibration/validation. The use of future climate data for modelling is discussed here.

2.4.1 Weather time series

Several daily weather variables are necessary for the SWAT+ model. They are minimum and maximum temperature, precipitation, solar radiation, wind speed and relative humidity. The first three are

[2]Module of SWAT used to generate missing weather values.

required, the next three are optional, depending on the Potential Evapotranspiration (PET) method chosen (see Additional settings). The data should cover the same time range and should allow for splitting into a calibration and validation period (plus 2-3 years of the warm-up period). Ideally, around 15-20 years of weather data are recommended, and 10 years as a bare minimum. The simulation period does not have to be exactly the same in different case studies, because of potentially non-overlapping data availability. If possible, the most recent data should be preferred. The modellers should note that the choice of the calibration and validation period will also determine the period of availability of other dynamic model inputs such as crop rotations, and optionally, point sources and atmospheric deposition. In OPTAIN it is recommended to provide data for all meteorological variables, since future climate projections will also include all variables.

There are many potential issues while preparing weather data. One of the most common is incorrect units. The correct units should be:

- Precipitation - daily total (mm/day);
- Temperature - daily minimum and maximum (°C);
- Wind speed - daily average (m/s) at 2 m height;
- Relative humidity - daily average fraction;
- Solar radiation - daily total (MJ/m^2).

Other potential problems might be missing values, which have to be filled, or suspicious values that should be corrected. The value *-99* could be used instead of a missing value, which would trigger the built-in weather generator to generate a value for this day from statistical weather parameters. However, the weather generator might not be the best option to fill missing or suspicious values as it generates somewhat random weather conditions. It should not be a big problem for filling small data gaps, but filling larger chunks of time series with the weather generator in the case of precipitation would definitely create problems in model calibration and validation. Thus, a far better option is to look for adjacent meteorological stations to fill data gaps. The closer they are to the selected watershed, the better. However, one should be aware that in mountainous areas not only distance, but also elevation is a crucial factor.

In case if no local meteorological data are available or usable, global or regional datasets could be applied. For example, ERA5-Land dataset provides for public use hourly meteorological data from 1950 to 2-3 months before the presence with a spatial resolution of 0.1 degrees. As part of the work in WP3 the data has been downloaded for all SWAT+ variables, all spatial domains covering each of the 14 OPTAIN case studies, for the time period 1981-2021. The data available for all OPTAIN project partners as well as on on ZENODO (see more in Climate change section). Another example is CFSR Global Weather Data for SWAT 1979-2014, which provides weather data already in SWAT model format. However, these products may have significant bias when compared to local station data, so it is advised to evaluate them before using them in the model.

If station data are used as input, in general it is recommended to use data from all available stations, in order to capture existing gradients in weather parameters, although in very small catchments spatial variability may be low. Again, it might be more important for precipitation than for other variables. Sometimes it is necessary to use weather stations outside the catchment. Interpolation of daily time series of weather data may be a suitable option, especially if at least several weather stations are available and they have different periods of data availability and/or many missing values. Interpolation, even using a simple method such as Thiessen polygon or Inverse Distance Weighted, would then generate gap-free time series and at the same time allow for capturing spatial variability at a daily time scale.

An important feature of interpolation is spatial resolution. This is also important if external gridded climate products (available in many countries) are used as input. In such cases virtual stations representing the grid centre points should be used instead of weather stations. For precipitation, resolution of virtual stations could be higher than for other variables.

Model input format is quite specific, requiring one file for information about the weather stations (id, name, latitude, longitude, elevation) and then for each variable and each station a time series in a separate file. Preparation of these files requires a lot of repetitive work with spreadsheets. Thus, in order to avoid random errors as well as to document how raw data are handled to get model input and also save time, scripting of this process is recommended.

Lastly, it is important to mention that quality check and cleaning meteorological data by identifying wrong or suspicious values is crucial. This can be done comparing data from different sources, plotting data in multiple ways, devising outlier tests, etc. Data needs to be properly examined before using it as model input.

For this the R package `svatools` prepared within OPTAIN project could be used. It provides functions for loading weather data from a given excel template, plotting them into interactive plots and aggregating by various methods to identify possible problems. Also this package provides functions to prepare, write weather station and weather generator input files directly into SWAT+ setups. The package can be installed in R using the following lines:

```r
devtools::install_github("biopsichas/svatools")
##It also needs euptf2 library, which is used for the soil parameters preparation
devtools::install_github("tkdweber/euptf2")
```

Documentation for its use for weather data is provided on the package website.

Data (for station locations and timeseries) for this package should be prepared in the provided Excel template and then could be easily loaded with one function `load_template()`.

```r
library(svatools)
temp_path <- system.file("extdata", "weather_data.xlsx",
                         package = "svatools")
## temp_path is path to a filled template
## weather_data.xlsx is excel template for weather data
## example is installed with package
met_lst <- load_template(temp_path)
```

Loaded data could be plotted in many ways (different methods of aggregation and for different time steps) using plotting function for one dataset `plot_weather()` or for two datasets `plot_weather_compare()` to compare results after data alterations or to compare data from two sources.

```r
plot_weather(met_lst, "PCP", "month", "sum")
plot_weather_compare(met_lst, met_lst2, "PCP", "month", "mean",
                     "dataset1", "dataset2")
```

Interpolation between stations could be done by `interpolate()` function of `svatools` package. The function uses the inverse distance weighting method. However, it additionally needs DEM and basin boundary data. GIS data should be provided in the same coordinate system. `interpolate()` function will generate SWAT+ model input files for virtual stations according to the grid spacing parameter provided.

```
result <- interpolate(met_lst, "./output/",  basin_path, DEM_path,
            grid_spacing = 2000)

##Interpolation results have to be transformed for use with other functions.
##Transforming interpolation result data structure
met_lst <- transform_to_list(result, start_date, end_date)
```

2.4.2 Weather Generator

Weather generator input consists of long-term monthly statistics of selected weather parameters. SWAT+ requires weather generator data even if all weather time series are free of gaps. Two situations in which SWAT+ would use this input are: (1) monthly temperature data would be used for heat unit calculation; (2) maximum (monthly) half-hour rainfall parameter (*pcp_hhr*) would be used for peak flow and erosion rate estimation. For the last parameter, sub-daily (ideally 30-minute or higher resolution) precipitation data for as many years as possible is required. Alternatively, in some countries peak rainfall intensity data may be available in national climate atlases or for urban drainage applications.

Dew-point temperature data required by weather generator may be missing in some countries. However, it could be easily calculated from relative humidity and temperature by using the dewpoint estimation program available on the SWAT website or by other methods.

The `svatools` package provides two options for preparing weather generator parameters. `prepare_wgn()` function could be used directly with loaded weather data to automatically prepare weather statistical parameters. It can be used for many stations. However, if some variables are missing for some stations, a function should be provided with information, which station's data should be used to fill in the missing variable.

```
wgn <- prepare_wgn(met_lst,
                   PCP = met_lst$data$ID9$PCP)
##If missing precipitation data in this example
##station ID9 data should be used instead.
```

Another option in `svatools` package is the function `write_wgnmaker_files()`, which prepares files for the WGNmaker excel macro tool available on the official SWAT model website. This tool also allows an easy preparation of required statistical parameters for the SWAT+ weather generator module, yet an additional step of applying it is needed. `write_wgnmaker_files()` generates data only for one station.

```
wgn_stations_lst <- list(PCP = "ID12", SLR = "ID11",
                         RELHUM = "ID12", TMP_MAX = "ID12",
                         TMP_MIN = "ID12", WNDSPD = "ID12",
                         MAXHHR = "ID12")
###wgn_stations_lst provides information, which station's data
###should be used if for particular variable
write_wgnmaker_files("./output/", met_lst,
                     wgn_stations_lst, "Station1")
```

2.4.3 Updating setup

Updating the model setup with prepared weather files could be done with SWAT+Editor. However, `svatools` `add_weather()` package offers simple one-step solution. If the weather data were loaded with `load_template()` or prepared with `interpolate()` + `transform_to_list()`, and weather generator parameters were prepared with `prepare_wgn()` function, the `add_weather()` function could be used directly to print all required files into the model setup folder and to update the model setup `.sqlite` database. The only condition is that the following code should be used for the setup that doesn't yet have any weather data in it.

```
##Path to .sqlite
db_path <- "./output/test/project.sqlite"
add_weather(db_path, met_lst, wgn)
```

Chapter 3

Model parametrization

The baseline model setup process described in the previous chapter (of which most steps are done with `SWATbuildR`) ends with the generated SWAT+ input files that are stored in an 'sqlite' data base which is readable and therefore further editable with the SWAT+Editor. With the weather data loaded, the model is able to run, but a lot of model input parameters and options have to be specified. The model setup derived from `SWATbuildR` is "raw" meaning that it has either default or empty values for various parameters or some functions are not "active" (e.g. point sources, water withdrawals, etc.). This chapter provides a thorough overview of all relevant aspects of the model parametrisation that the SWAT+ modeller has to consider before running the calibration (please note, however, that agricultural land management and decision tables are included in separate chapters). It directly points to most relevant input files and parameters and provides some recommendations on data sources and pre-processing aspects.

The majority of changes in parameters/options discussed in this chapter can be implemented via SWAT+ Editor that has an online documentation (although, as of November 2022, not all functions were included there). However, the user is free to implement changes directly in the 'sqlite' data base (more information here) or by directly manipulating the SWAT+ input files (see IO documentation).

3.1 Land use

So far, our model setup accounts for a precise spatial distribution of land use by using a high resolution land use / land cover map as model input (section 2.2). The different land use classes must now be further described by choosing appropriate parameter values and management schedules. In the file *'hru-data.hru'*, each land object points to a certain land-use-management (*lu_mgt*), which needs to be described in file *'landuse.lum'*. *'landuse.lum'* itself is pointing to different sets of land-use related parameters and management schedules. It is important to note that the baseline model setup via `SWATbuildR` generated a *'landuse.lum'* file without any pointers to parameters or schedules (i.e., there is no default setting as provided with a QSWAT+ setup). Therefore, it is even more important to carefully study the official SWAT+ land-use-management documentation. This section describes the most relevant parameter settings in the file *'landuse.lum'*, while management schedules are addressed in later sections (4.2 and 4.3).

Non-cropland areas can be described using generic *lu_mgt* classes for forest (e.g. *frst_lum*) or other semi-natural land covers (e.g. *rngb_lum*), grassland (e.g. *past_lum*), barren land (*bsvg_lum*) and urban areas (e.g. *urmd_lum*, *utrn_lum*). In contrast, cropland in OPTAIN is described by a large number of individual fields, which can have their own individual management, resulting in a large number of field-specific *lu_mgt* classes, such as *field_1_lum*, *field_2_lum*, etc. Even grassland or pasture may have their own management on the field-level (only if appropriate). Due to the large number of *lu_mgt* classes, it is not convenient to edit the *'landuse.lum'* file with the SWAT+ Editor. We recommend to edit the file in Excel or (even better) to manipulate it using R. Example R code is provided at the end of this section.

3.1.1 Plant community (*plnt_com*)

Plant communities in SWAT+ define all plant or land cover types that can occur within the simulation period for a given *lu_mgt* class. Column *plnt_com* in the *'landuse_lum'* file points to the plant community defined in the *'plant.ini'* file. This file includes also all variables needed for initializing plant growth. In OPTAIN, the user does not need to edit the *'plant.ini'* file. It will be automatically updated when using the `SWATfarmR` package. However, it is necessary to prepare the `SWATfarmR` input table as described in section 4.2.

3.1.2 Management schedules (*mgt*)

Management schedules are described in sections 4.2 and 4.3.

3.1.3 Curve Numbers (*cn2*)

SWAT+ uses the SCS curve number method to partition precipitation into surface runoff and infiltration (+ interception). The curve number is thus a central model parameter (especially for OPTAIN) as it has a direct impact on the water retention of a given land object (a field, or a part of a field representing an NSWRM). The curve number is a function of the soil's permeability, land use and antecedent soil water conditions. Model users have to define *cn2* values, i.e. curve number values for antecedent soil moisture condition II (average moisture condition). The SWAT+ input/output documentation provides a table for various land covers and hydrologic soil types A-D. This table is also given in the SWAT+ input file *'cntable.lum'*. The user needs to assign the most representative set of *cn2* values to each *lu_mgt* class (column *cn2* in *'landuse.lum'*). In case, none of the sets provided in *'cntable.lum'* are representative for a desired land cover type (e.g. a certain NSWRM), users can define their own set of *cn2* values. It is necessary that *'cntable.lum'* lists *cn2* values for all relevant land cover/management types; this includes all NSWRM scenario land cover/management types.

3.1.4 Conservation Practices (*usle_p*)

The Practice factor in the Universal Soil Loss Equation (*usle_p*) reduces the amount of soil erosion due to a given conservation practice. It is the ratio of the erosion resulting from the described practice to that which would occur with up-and-down slope cultivation. For conservation practices, their typical *usle_p* and maximum slope length (*slp_len_max*) values are given in file *'cons_practice.lum'*. It applies the same as for *cn2*. Users need to assign the most representative set of parameter values to each of their *lu_mgt* classes (column *cons_prac* in *'landuse.lum'*). In case, none of the sets provided in *'cons_practice.lum'* are representative for a desired land management type (e.g. a certain NSWRM), users can define their own conservation practice. It is necessary that *'cons_practice.lum'* provides parameter values for all relevant land cover/management types; this includes all NSWRM scenario land cover/management types.

3.1.5 Manning's n (*ovn*)

Manning's roughness coefficient for overland flow (*ovn*) controls the routing of surface runoff. It is thus another important land-use related parameter which needs to be defined with great care. *'ovn_table.lum'* lists typical *ovn* values and ranges for various land cover and tillage types (including the amount of residuals left on the field). Users need to assign the most representative set of *ovn* values to each of their *lu_mgt* classes (column *ov_mann* in *'landuse.lum'*). In case, none of the sets provided in *'ovn_table.lum'* are representative for a desired land cover/tillage types (e.g. a certain NSWRM), users can define their own set of *ovn* values. *'ovn_table.lum'* must list *ovn* values for all relevant land cover/management types; this includes all NSWRM scenario land cover/management types.

3.1.6 Urban parameters

Although not in focus of OPTAIN, urban areas have to be parameterized as well in order to ensure meaningful catchment balances of water and nutrient fluxes. File *'urban.urb'* lists a set of 10 parameters for typical urban land use/cover classes. It is mandatory to assign the most representative set to each of your urban *lu_mgt* classes (column *urban* in *'landuse.lum'*).

3.1.7 Further specifications (*tile*, *sep*, *vfs*, *grww*, *bmp*)

In contrast to the aforementioned parameters, columns *tile*, *sep*, *grww*, *bmp* in the *'landuse.lum'* file can remain empty (*NULL*) if these are not relevant in the case study.

tile is important if tile drainage should be considered on specific agricultural fields. In such a case, it is necessary that *tile* points to representative parameters of the user's tile drain system in file *'tiledrain.str'*, as described in section 3.9 and the SWAT+ input/output documentation.

sep can be specified if onsite waste water systems are relevant in a case study. *sep* points to file *'septic.sep'*, characterising water, nutrient, sediment, and bacteria related parameters of different septic systems. Users can also define their own septic systems if desired. However, septic parameter values might be hard to guess or generalize and the algorithm was never tested in Europe. The topic is addressed in more detail in chapter *Domestic waste from disconnected areas* in section 3.8.

vfs may become relevant for simulating edge-of-field filter strips in a parametric approach. This can be advantageous over the COCOA approach, see D2.3 SWAT+ and SWAP retention measure implementation handbook (Marval et al., 2022). With a parametric approach, filter strips can be modelled by either changing the label in column *vfs* when the measure is implemented (which then points to the measure in file *'filterstrip.str'*) or by initialising the measure in the default configuration and setting the value of the *flag_fs* to 1 when the measure should be activated and reset it to 0 when removing the buffer strip. If *vfs* is used, it is important that the right filter strip parameters are defined in file *'filterstrip.str'*.

grww may become important if grassed waterways should be modelled in a parametric approach (which is not recommended for OPTAIN and its COCOA approach, where grassed waterways should modelled as land objects with their own land-cover related parameters and their own connectivity to other objects based on their individual spatial position within a landscape).

However, due to the lack of generic solutions to model swales with COCOA, the parametric way using the grassed waterway parametrization might be the only feasible solution to model this type of NSWRM. Column *grww* in *'landuse.lum'* points to file *'grassedww.str'*, which includes typical parameters for grassed waterways on low, medium and high slopes.

bmp offers the possibility to consider conservation practices which are unsupported by SWAT+. However, approximate removal efficiencies must be known and specified by constituent in file *'bmpuser.str'*.

3.1.8 Example R code to manipulate the landuse.lum file

```r
# R packages ----------------------------------------------------
library(tidyverse)
# library(data.table)
library(vroom)

# Project path --------------------------------------------------
proj_path <- 'C:/Define/your/path'
# ---------------------------------------------------------------

# Functions ------------------------------------------------------
read_tbl <- function(tbl_name, proj_path, row_data_start, row_col_names) {
  tbl_path <- paste(proj_path, tbl_name, sep = '/')
  col_names <- vroom_lines(tbl_path, skip = row_col_names - 1, n_max = 1) %>%
    str_trim(.) %>%
    str_split(., '[:space:]+') %>%
    unlist()

  tbl <- vroom_lines(tbl_path, skip = row_data_start - 1) %>%
    str_trim(.) %>%
    str_split(., '\t[:space:]+|[:space:]+')

  is_num <- tbl[[1]] %>% as.numeric() %>% suppressWarnings() %>% map_lgl(.,
↪  ~!is.na(.x)) %>%  which()

  tbl <- tbl %>%
    map(., ~ set_names(.x, col_names)) %>%
    map_df(., bind_rows) %>%
    mutate(across(all_of(is_num), ~ as.numeric(.x)))

  return(tbl)
}

# Read landuse.lum -----------------------------------------------
lum <- read_tbl('landuse.lum', proj_path, 3, 2)
lum_head <- vroom_lines(paste(proj_path, 'landuse.lum', sep = '/'), n_max = 1) %>%
  paste0(., ', edited manually on ', Sys.time())

# Define pointers in landuse.lum ----------------------------------

## cn2
lum$cn2[which(substr(lum$name,1,5)=='field')] <- 'rc_strow_g'
lum$cn2[which(substr(lum$name,1,4)=='frst')] <- 'wood_f'
lum$cn2[which(substr(lum$name,1,4)=='orcd')] <- 'woodgr_f'
lum$cn2[which(substr(lum$name,1,4)=='rngb')] <- 'brush_f'
lum$cn2[which(substr(lum$name,1,4)=='rnge')] <- 'brush_f'
lum$cn2[which(substr(lum$name,1,4)=='wetl')] <- 'wood_p'
lum$cn2[which(substr(lum$name,1,4)=='bsvg')] <- 'fal_bare'
lum$cn2[which(substr(lum$name,1,4)=='urld')] <- 'farm'
lum$cn2[which(substr(lum$name,1,4)=='urmd')] <- 'dirtroad'
```

```
lum$cn2[which(substr(lum$name,1,4)=='utrn')] <- 'urban'
lum$cn2[which(substr(lum$name,1,12)=='meadow_2cuts')] <- 'pasth'
lum$cn2[which(substr(lum$name,1,12)=='meadow_3cuts')] <- 'pasth'
lum$cn2[which(substr(lum$name,1,12)=='meadow_4cuts')] <- 'pasth'

## cons_prac
lum$cons_prac[which(substr(lum$name,1,5)=='field')] <- 'up_down_slope'
lum$cons_prac[which(substr(lum$name,1,4)=='frst')] <- 'up_down_slope'
lum$cons_prac[which(substr(lum$name,1,4)=='orcd')] <- 'up_down_slope'
lum$cons_prac[which(substr(lum$name,1,4)=='rngb')] <- 'up_down_slope'
lum$cons_prac[which(substr(lum$name,1,4)=='rnge')] <- 'up_down_slope'
lum$cons_prac[which(substr(lum$name,1,4)=='wetl')] <- 'up_down_slope'
lum$cons_prac[which(substr(lum$name,1,4)=='bsvg')] <- 'up_down_slope'
lum$cons_prac[which(substr(lum$name,1,4)=='urld')] <- 'up_down_slope'
lum$cons_prac[which(substr(lum$name,1,4)=='urmd')] <- 'up_down_slope'
lum$cons_prac[which(substr(lum$name,1,4)=='utrn')] <- 'up_down_slope'
lum$cons_prac[which(substr(lum$name,1,12)=='meadow_2cuts')] <- 'up_down_slope'
lum$cons_prac[which(substr(lum$name,1,12)=='meadow_3cuts')] <- 'up_down_slope'
lum$cons_prac[which(substr(lum$name,1,12)=='meadow_4cuts')] <- 'up_down_slope'

## ov_mann
lum$ov_mann[which(substr(lum$name,1,5)=='field')] <- 'convtill_nores'
lum$ov_mann[which(substr(lum$name,1,4)=='frst')] <- 'forest_med'
lum$ov_mann[which(substr(lum$name,1,4)=='orcd')] <- 'forest_light'
lum$ov_mann[which(substr(lum$name,1,4)=='rngb')] <- 'forest_light'
lum$ov_mann[which(substr(lum$name,1,4)=='rnge')] <- 'densegrass'
lum$ov_mann[which(substr(lum$name,1,4)=='wetl')] <- 'forest_light'
lum$ov_mann[which(substr(lum$name,1,4)=='bsvg')] <- 'fallow_nores'
lum$ov_mann[which(substr(lum$name,1,4)=='urld')] <- 'shortgrass'
lum$ov_mann[which(substr(lum$name,1,4)=='urmd')] <- 'range_sparse'
lum$ov_mann[which(substr(lum$name,1,4)=='utrn')] <- 'urban_asphalt'
lum$ov_mann[which(substr(lum$name,1,12)=='meadow_2cuts')] <- 'densegrass'
lum$ov_mann[which(substr(lum$name,1,12)=='meadow_3cuts')] <- 'densegrass'
lum$ov_mann[which(substr(lum$name,1,12)=='meadow_4cuts')] <- 'densegrass'

## urban
lum$urban[which(substr(lum$name,1,4)=='urld')] <- 'urld'
lum$urban[which(substr(lum$name,1,4)=='urmd')] <- 'urmd'
lum$urban[which(substr(lum$name,1,4)=='utrn')] <- 'utrn'


# Write new landuse.lum ---------------------------------------------
fmt_nam <- c('%-28s', '%-9s', rep('%17s', 12))
fmt_val <- c('%-33s', '%-4s', rep('%17s', 12))

lum_names <- colnames(lum) %>%
  map2_chr(., fmt_nam, ~sprintf(.y, .x)) %>%
  paste(., collapse = ' ')

lum_lines <- lum %>%
  map2_df(., fmt_val, ~sprintf(.y, .x)) %>%
  apply(., 1, paste, collapse = ' ')

lum_lines <- c(lum_head, lum_names, lum_lines)

write_lines(lum_lines, paste(proj_path, 'landuse2.lum', sep = '/'))
#check landuse2.lum in a text editor before replacing the old
##landuse.lum file
```

3.2 Channel properties

Channel parameters in SWAT+ are included in the *hyd-sed-lte.cha* file. The section deals with geometric and hydraulic parameters, that affect mainly channel routing processes, and indirectly the calibration of discharge.

Most important geometric parameters include bankfull width (wd) and bankfull depth (bd) for each channel segment. By default, in `SWATbuildR` these parameters are estimated using empirical equations (3.1) from the study of Bieger et al. (2015) for the United States, which require the Drainage Area (DA) in hectares contributing water to the channel:

$$
\begin{aligned}
wd &= 2.70 * (DA * 10^{-2})^{0.352} \\
bd &= 0.30 * (DA * 10^{-2})^{0.213}
\end{aligned}
\tag{3.1}
$$

Such equations belong to the so-called Hydraulic Geometry (HG) relationships that relate channel geometric parameters to DA or bankfull discharge. The resulting parameters may have unrealistic values for a given catchment, since such relationships are highly region-specific (Figure 3.1). Hence, it is suggested to compare the derived values with the measured ones, and in case of significant disagreement, to replace them with more accurate ones.

In an ideal situation, which rarely will be the case in practice, measured data for all or most of channel segments would be available. An example could be the combination of LIDAR (for above waterline) and Sound Navigation and Ranging (SONAR) data (for underwater), which could be used for extracting channel parameters. More likely the measured data are cross-sectional profiles in certain places from which bankfull widths and depths may be directly read (although deciding what “bankfull” means in practice for natural channels may be at times challenging). SWAT+ considers each channel segment as an individual, trapezoidal routing object, so if more data are available, they should be averaged or a representative cross-section should be selected.

A more common approach may be to derive a HG relationship similar as the one shown in Eq. (3.1) using local data. In such relationships, DA can be considered as a surrogate for bankfull discharge that enables making prediction also for ungauged sites (i.e. nearly all channel segments) (Bieger et al., 2015). The generic form of HG relationship is usually a power function:

$$y = a \cdot x^b \tag{3.2}$$

where y is the dependent variable (bd or wd), x = DA is the independent variable of drainage area, a is a coefficient indicating the intercept of the regression line, and b is an exponent representing the slope of the regression line. The values of the coefficient a and the exponent b should be determined by least-squares regression analysis using the available empirical data after the logarithmic transformation to allow the application of linear techniques (Bieger et al., 2015).

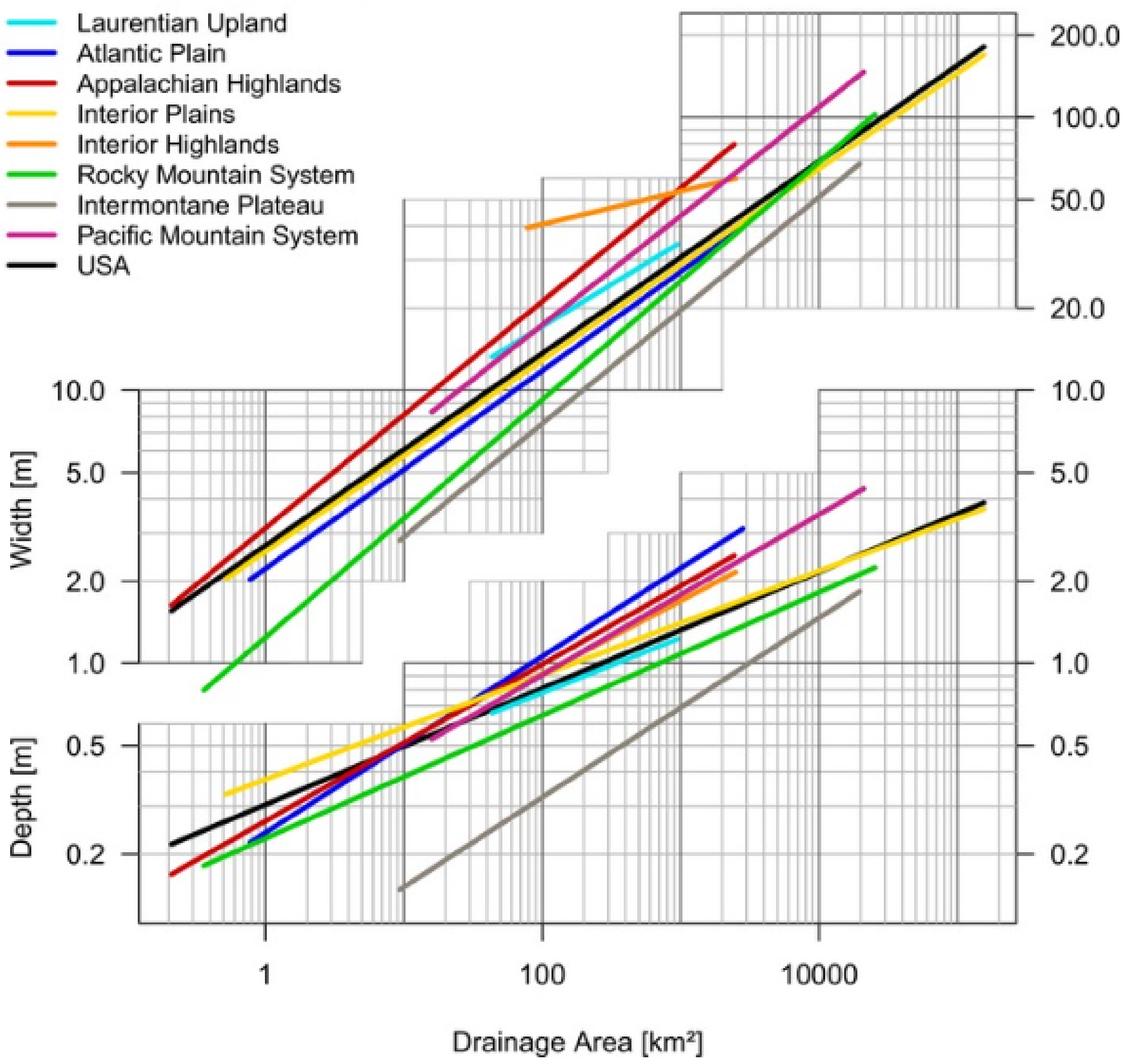


Figure 3.1: Example curves relating bankfull width and depth to drainage area for US regions (adapted from Bieger et al. (2015)).

In the absence of measured cross-sectional data from the analysed catchment, data from neighbouring catchments can also be used as the HG relationships tend to be regional. In worst case, *wd* can be estimated based on high quality orthophotos, but to estimate *bd* measured data are indispensable. As always, the more the data the better, but the equation can be fit even for a few measurements. It is recommended, though, that the data include a wide range of DA values.

In case drainage or roadside ditches are included in the channel network, they may not follow the HG relationship, but instead, they may have constant dimensions. In this case, they should be parametrized manually, while HG relationships should be applied to the natural channel segments only.

Two other parameters related to channel geometry are channel length (*len*) and width-to-depth ratio (*wd_rto*). The former should have correct values, while the latter should be updated after any change in either *wd* or *bd*.

SWAT+ uses the Manning's equation to calculate average flow velocity in each channel. Two main hydraulic input parameters in this equation are channel slope (*slp*) and Manning's roughness coefficient (*mann*). While slope values are derived directly from the DEM and channel layer, Manning's roughness coefficient values may be adjusted by the model user based on look-up tables. In most cases, for natural channels, the values range between 0.03 and 0.06.

Another parameter directly affecting channel routing is the effective hydraulic conductivity in the main channel alluvium (*k*). For perennial streams with continuous groundwater contribution it should be set to 0. In contrast, losing streams will normally have positive *k* values, depending on the stream bed material (see .rte chapter of the SWAT2012 documentation for look-up tables). Both *mann* and *k* are frequently used in the calibration of discharge.

Modifying channel parameters (geometric, roughness) may be one option, though not perfect, to represent hydromorphological NSWRMs, such as channel restoration or channel remeandering, in the SWAT+ model setup.

3.3 Crops

All crop-associated parameters can be found in the *plants.plt* file. In general, it is recommended to use the locally measured crop data, if such exist. Crop data, available from reference sites or calibrated, using site-specific model parameters could also be used. In most of the cases, however, some basic plant properties are monitored only, or no measured plant data are available at all.

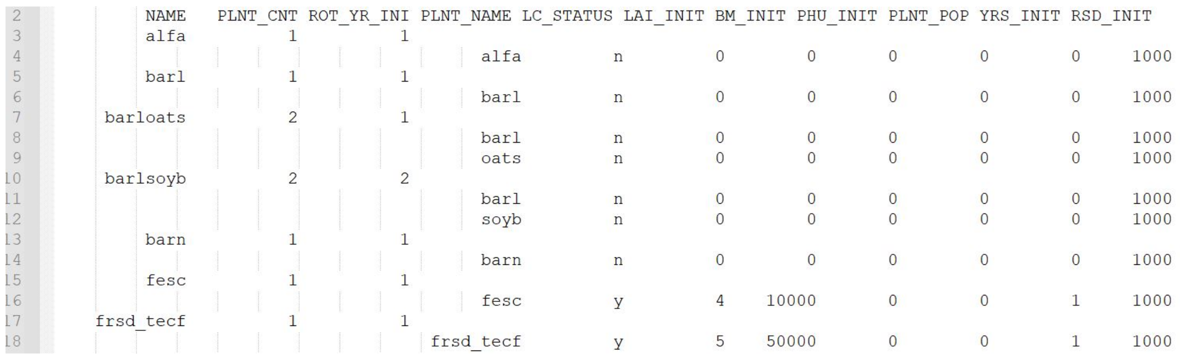
The majority of SWAT+ plant parameters are shared with other crop-growth models, such as APEX, EPIC, ALMANAC and others. Besides crop-growth and catchment scale hydrological models, also many field- or profile-scale soil hydrological models like SWAP, HYDRUS and COUP have crop routines of various complexity, and their plant parameters partly overlap with those, used in SWAT+.

All this knowledge can help modellers to develop and define parameters for unavailable crop types, which are not present in the default SWAT+ database (*plants.plt* file). Here, we will cover the SWAT+-specific parameters, which are uncommon and might require manual adjustment.

SWAT+ specific plant parameters

In the *plants.plt* file, the heat units to maturity (*phu_mat*) were changed to days to maturity (*days_mat*). The concept of heat units to maturity was developed for annual crops and we use heat units for the entire growing season for native perennials and native annuals. By inputting days to maturity, we can include different crop varieties as defined by length of growing season (for example, corn varieties for 120-, 110-, 100- and 90-day varieties). The algorithm currently uses monthly weather generator parameters to initiate the plant growth, hence an accurate weather generator is required. For information on other parameters, refer to the current input documentation.

Crop initialization parameters can be found in the *plant.ini* file. This file will define the initial conditions of plants, as well as the newly added plant community initialization. The PLANT_COV input in the *landuse.lum* file points to the name in the *plant.ini* file. PLNT_NAME input in the *plant.ini* file points to *plants.plt* file, where all the plant/crop names should be defined. Plant initialization files can be constructed to allow decision tables for planting and harvesting to be used. Below is a sample *plant.ini* file:



```
      NAME  PLNT_CNT ROT_YR_INI PLNT_NAME LC_STATUS LAI_INIT BM_INIT PHU_INIT PLNT_POP YRS_INIT RSD_INIT
      alfa         1          1
                                    alfa         n        0       0        0        0        0     1000
      barl         1          1
                                    barl         n        0       0        0        0        0     1000
  barloats         2          1
                                    barl         n        0       0        0        0        0     1000
                                    oats         n        0       0        0        0        0     1000
  barlsoyb         2          2
                                    barl         n        0       0        0        0        0     1000
                                    soyb         n        0       0        0        0        0     1000
      barn         1          1
                                    barn         n        0       0        0        0        0     1000
      fesc         1          1
                                    fesc         y        4   10000        0        0        1     1000
 frsd_tecf         1          1
                               frsd_tecf         y        5   50000        0        0        1     1000
```

Figure 3.2: Snippet of the plants.ini file

In the *plant.ini* snippet above, the *alfa* example is typical of any native perennial plant, which is not growing at the start of the simulation. The *fesc* example is a typical example of a plant, which is growing at the start of the simulation, and has accumulated some biomass. The *barloats* example initializes a barley-oats rotation with barley growing the first year (indicated by ROT_YR_INI=1). The *barlsoyb* example initializes a barley-soybean rotation with soybeans growing the first year (ROT_YR_INI =2).

A good practice to model and parametrize plant growth and yields is to make sure that:

1. Plants/crops are correctly initialized (the plant communities are present in the *plant.ini* file, with necessary and accurate initialization data);
2. Plants/crops are correctly parametrized (the plant is present in the *plants.plt* file with the necessary relevant parameters);
3. Plant/crop management is set up appropriately. See the agricultural management chapter of this protocol.

Check the model evaluation chapter for a step-by-step check-point guide on crop growth modelling with SWAT+.

Alternative sources of plant data

Soil hydrological models can be relevant sources of plant data. The example input files or databases, related to these models might incorporate crops, that are not available in the SWAT, APEX etc. databases. On the other hand, region-specific calibrated crop parameters of these models can be used to adjust the available SWAT+ plant data to the study region or site. This, however, only concerns joint parameters, which are commonly the maximum leaf area index, land cover factor, maximum crop height, maximum rooting depth, albedo and the harvest index or yield response.

The SWAP crop database, being developed within the OPTAIN project contains data for various plants, including forests, grasslands and agricultural crops. This database can be used for cross-validating and completing the SWAP and SWAT+ crop parameters. The database contains both, static and dynamic crop parameters, the latter being given as a function of the crop development stage. The crop database is stored in an Excel format and available for the OPTAIN consortium partners.

Another useful and rather complex set of crop data is related to the crop development model WOFOST, also used by the SWAP model when the advanced crop routine is selected: https://github.com/ajwdewit/WOFOST_crop_parameters and https://github.com/ajwdewit/WOFOST/tree/master/cropd

3.4 Soil physical data

Several NSWRMs - mostly the management measures - influence the hydraulic processes of agricultural fields. In order to analyse the effectiveness of NSWRMs in retaining water and nutrients, it is important to have field-based soil physical and chemical input data for the target area.

The soil physical input data is provided in the *soils.sol* file. These parameters are already required by the `SWATbuildR` program during the step of loading soil data. The *usersoil* table required by `SWATbuildR` is exported to the *soils.sol* file. However, it often happens that users do not have all parameter values of all of their soil classes within the soil map at the initial stage of the model setup, so we present these inputs in this part of the protocol and add some clarifications to support the proper use of soil data. Information on the soil data related SWAT+ model setup is described in SWATbuildR input data preparation section of this protocol.

The following soil properties are required as soil physical data – SWAT+ and SWAT2012 acronyms are added in brackets to ease identification of the parameters:

- soil hydrologic group (HYDGRP),
- maximum rooting depth of the soil profile (dp_tot or SOL_ZMX) – it has to be equal with the depth from soil surface to the bottom of the deepest soil layer ,
- depth from soil surface to bottom of the layer (dp or SOL_Z),
- moist bulk density (bd or SOL_BD),
- available water capacity (awc or SOL_AWC),
- saturated hydraulic conductivity (soil_k or SOL_K),
- organic carbon content (carbon or SOL_CBN),
- clay, silt and sand content (clay or SOL_CLAY, silt or SOL_SILT, sand or SOL_SAND),
- rock fragment content (rock or SOL_ROCK),
- moist soil albedo (alb or SOL_ALB),
- Universal Soil Loss Equation (USLE) soil erodibility factor (usle_k or USLE_K) of each soil layer.

These soil properties are considered to influence the movement of water and air in the soil profile, and thus have impact on the soil hydrologic processes in the Hydrological Response Unit (HRU). Further optional soil properties are soil name, fractions of porosity from which anions are excluded, potential crack volume of the soil profile, texture and user comments. Data for maximum 25 soil layers for each soil profiles can be included in the usersoil table (Arnold et al., 2012a).

The basic soil properties – e.g. soil organic carbon content, particle size distribution – are usually locally available, but information on some soil physical and hydraulic properties are often missing. These parameters can be computed based on national guidelines (e.g.: Ad-hoc-AG Boden (2005)) or with Pedotransfer Function (PTF)s, which are widely-used indirect techniques enabling the soil properties to be predicted by using easily-retrievable basic soil information. An alternative can be the use of open access international data if local or national information is lacking.

Hereinafter we provide i) information for the correct use of soil data, ii) suggestions on how to derive missing soil physical input data and iii) possible tools to compute missing data.

3.4.1 Basic soil physical properties

Figure 3.3 shows the main steps used in OPTAIN for deriving the basic soil properties. If basic soil physical properties – such as soil organic carbon content, sand, silt and clay content, rock fragments content – are missing, the use of the SoilGrids dataset (Poggio et al., 2021) is recommended to use, which is freely available from the ISRIC website.

Organic carbon content

It is important to clarify what data related to organic carbon content is available. In the case of soil organic matter or humus content, the data has to be converted into organic carbon content (carbon or SOL_CBN) with equation (3.3).

$$SOL\_CBN = 0.58 \cdot humus \tag{3.3}$$

where SOL_CBN ($mass\%$) is soil organic carbon content and $humus$ ($mass\%$) is soil organic matter content.

Particle size distribution

It is a known obstacle in international soil-related research that different countries – and often different institutions within a country – measure particle-size distribution by different standards and often represent it according to different classification systems. Historically, European countries adapted different size standards for the description of soils, which is best depicted in Weynants et al. (2013). Table 27.1 therein depicts that often several different measurement/data patterns exist even within a country.

The SWAT+ model and some of its built-in calibrated parameterizations work with the definitions used by the FAO-USDA particle-size classification system that defines clay content as the mass of solids (individual particles) that are <0.002 mm, silt as the mass of solids in the 0.002 – 0.05 mm size range, and sand content as the mass of solids in the 0.05 – 2 mm size range (Food and Agriculture Organization, 1990; USDA, 1951). Particles sized above 2 mm are considered as gravel or stones.

Nemes and Rawls (2006) demonstrated that a lack of particle-size data conversion will increase the risk of introducing bias in any follow-up application, while an interpolation based conversion introduces some random error, but reduces the chances of bias in subsequent applications. Since particle size data and derived soil texture information is being used further as basis of SWAT+ model parameterization, conversion of un-matching soil particle-size data using an interpolation technique is desirable.

When measured points are sparse – i.e. only the triplet of sand, silt and clay content available – , a k-nearest neighbour type pattern recognition algorithm (termed 'similarity procedure' in the original publication) (Nemes et al., 1999) could be used. Details of this technique, its development and assessment for the estimation and interpolation of soil physical and hydraulic properties, and tests performed to evaluate its capabilities and robustness can be found in Nemes et al. (1999), Nemes et al. (2006b), Nemes et al. (2006a) and Nemes et al. (2010).

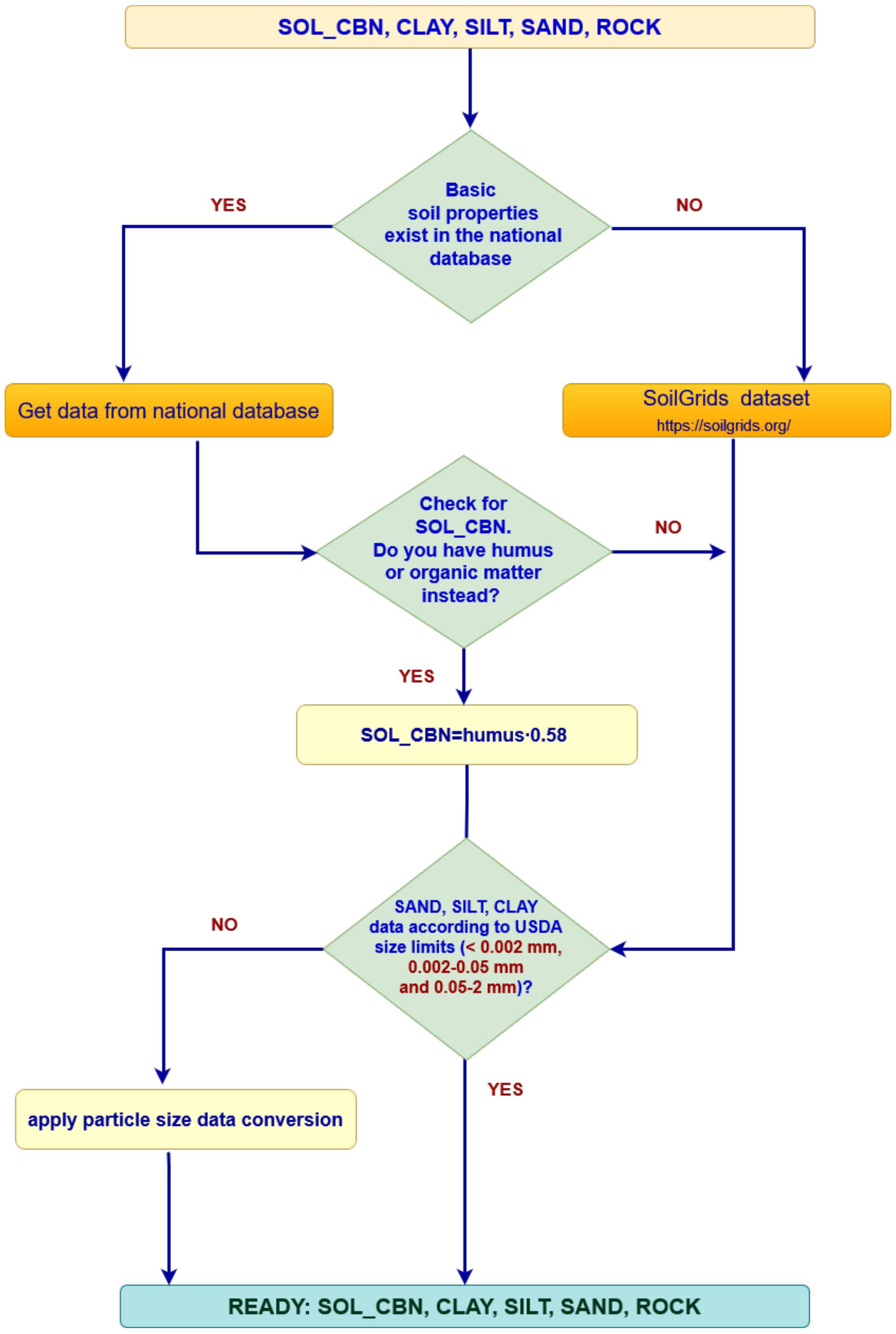

SOL_CBN, CLAY, SILT, SAND, ROCK
Basic soil properties exist in the national database
YES
NO
Get data from national database
SoilGrids dataset
https://soilgrids.org/
Check for SOL_CBN. Do you have humus or organic matter instead?
NO
YES
SOL_CBN=humus·0.58
SAND, SILT, CLAY data according to USDA size limits (< 0.002 mm, 0.002-0.05 mm and 0.05-2 mm)?
NO
YES
apply particle size data conversion
READY: SOL_CBN, CLAY, SILT, SAND, ROCK

Figure 3.3: Flowchart of deriving basic soil properties for the SWAT+ model. SOL_CBN: soil organic carbon content (*mass* %), CLAY: clay content (*mass* %), SILT: silt content (*mass* %), SAND: sand content (*mass* %), ROCK: rock fragment content (> 2 *mm*) (*mass* %) .

The particle-size interpolations in OPTAIN were made using a custom written MATLAB code that is made available for further use on ZENODO (Nemes, 2022).

Bulk density

In the case of bulk density it is recommended to compute it with a pedotransfer function based on local organic matter content and/or particle size distribution data. This way local variability might be better explained than retrieving bulk density from a global soil dataset. There are several PTFs available from the literature (Abbaspour et al., 2019). Based on their accuracy test on the European Hydropedological Data Inventory (EU-HYDI) (Weynants et al. (2013)) equation (3.4) (Alexander, 1980) could be used for European catchments.

$$BD_{dry} = 1.72 - 0.294 \cdot (SOL\_CBN)^{0.5} \tag{3.4}$$

The moist bulk density (bd or SOL_BD) can be considered to be synonym with the effective bulk density, which can be computed with the method of Wessolek et al. (2009):

- for soils with organic carbon content > 1%:

$$SOL\_BD = BD_{eff} = BD_{dry} + 0.009 \cdot clay \tag{3.5}$$

- for soils with organic carbon content <= 1%:

$$SOL\_BD = BD_{eff} = BD_{dry} + 0.005 \cdot clay + 0.001 \cdot silt \tag{3.6}$$

where BD_{eff} $(g\ cm^{-3})$ is effective bulk density, BD_{dry} $(g\ cm^{-3})$ is the dry bulk density, clay is clay content ($mass\,\%$, $< 0.002\ mm$), silt is silt content ($mass\,\%$, 0.002-0.05 mm).

Moist soil albedo of the top layer

Any of the equations presented by Abbaspour et al. (2019) could be used to calculate the moist soil albedo of the top layer (alb or SOL_ALB). For these calculations field capacity (Water Content at Field Capacity (FC)) is required, which is derived by the approach described under the Soil hydraulic properties subchapter. In OPTAIN we recommend to use the equation of Gascoin et al. (2009):

$$Albedo = 0.15 + 0.31 \cdot e^{-12.7 \cdot FC} \tag{3.7}$$

USLE soil erodibility (K) factor

It is recommended to use the equations presented by Abbaspour et al. (2019) for the prediction of the Universal USLE soil erodibility (K) factor (usle_k or USLE_K) ($\frac{t \cdot ha \cdot h}{ha \cdot MJ \cdot mm}$). If the unit is in $\frac{t \cdot acre \cdot h}{houndreds\ of\ acre \cdot foot-tonf \cdot inch}$, it has to be multiplied with 0.1317 to get the unit used by SWAT+ (Foster et al., 1981). The computation requires sand ($mass\,\%$, 0.05-2 mm), silt ($mass\,\%$, 0.002-0.05 mm), clay ($mass\,\%$, $< 0.002\ mm$) and organic carbon (OC, $mass\,\%$) content of the soil (Sharpley and Williams, 1990).

$$USLE\_K = E_S \cdot E_{C-T} \cdot E_{OC} \cdot E_{HS} \tag{3.8}$$

where

$$E_S = 0.2 + 0.3 \cdot e^{-0.0256 \cdot sand \cdot (1 - \frac{silt}{100})}$$

$$E_{C-T} = (\frac{silt}{clay + silt})^{0.3}$$

$$E_{OC} = 1 - (\frac{0.25 \cdot OC}{OC + e^{3.72 - 2.95 \cdot OC}})$$

$$E_{HS} = 1 - \frac{0.7 \cdot (1 - \frac{sand}{100})}{(1 - \frac{sand}{100}) + e^{-5.51 + 22.9 \cdot (1 - \frac{sand}{100})}}$$

3.4.2 Soil hydraulic properties

It is recommended to derive the soil hydraulic parameters – namely the available water capacity (awc or SOL_AWC) and hydraulic conductivity (soil_k or SOL_K) – from the parameters of the van Genuchten model (Genuchten, 1980) because this way the parameters will be self-consistent, and rely on dynamic criterion based on soil internal drainage dynamics (Assouline and Or, 2014; Nasta et al., 2021). The main steps are demonstrated in Figure 3.4.

Available water capacity (AWC)

Plant available water capacity (awc or SOL_AWC) is defined by the water content at field capacity and at wilting point with the following equation:

$$SOL\_AWC = FC - WLP \tag{3.9}$$

where:

FC: water content at field capacity

WLP: water content at wilting point, which corresponds to water content at -15,000 cm matric potential head.

Main steps to compute AWC

1. Predict parameters of the van Genuchten model

First, predict parameters of the van Genuchten model – i.e. θ_r, θ_s, α and n – that describes the full soil water retention curve (equation (3.10)).

$$\theta(\psi) = \theta_r + \frac{\theta_s - \theta_r}{(1 + (\alpha \cdot \psi^n))^m}$$

$$m = 1 - \frac{1}{n} \tag{3.10}$$

where ψ is matric potential (cm), θ_r ($cm^3\ cm^{-3}$) and θ_s ($cm^3\ cm^{-3}$) are the residual and saturated soil water contents, respectively, α (cm^{-1}) is a scale parameter, m (-) and n (-) are shape parameters.

The approach to predict parameters of the van Genuchten model depends on the type of input data available for the prediction. As a minimum requirement soil texture classes and topsoil/ subsoil distinction have to be available for the prediction. Hereinafter two main approaches are recommended regarding the input data availability.

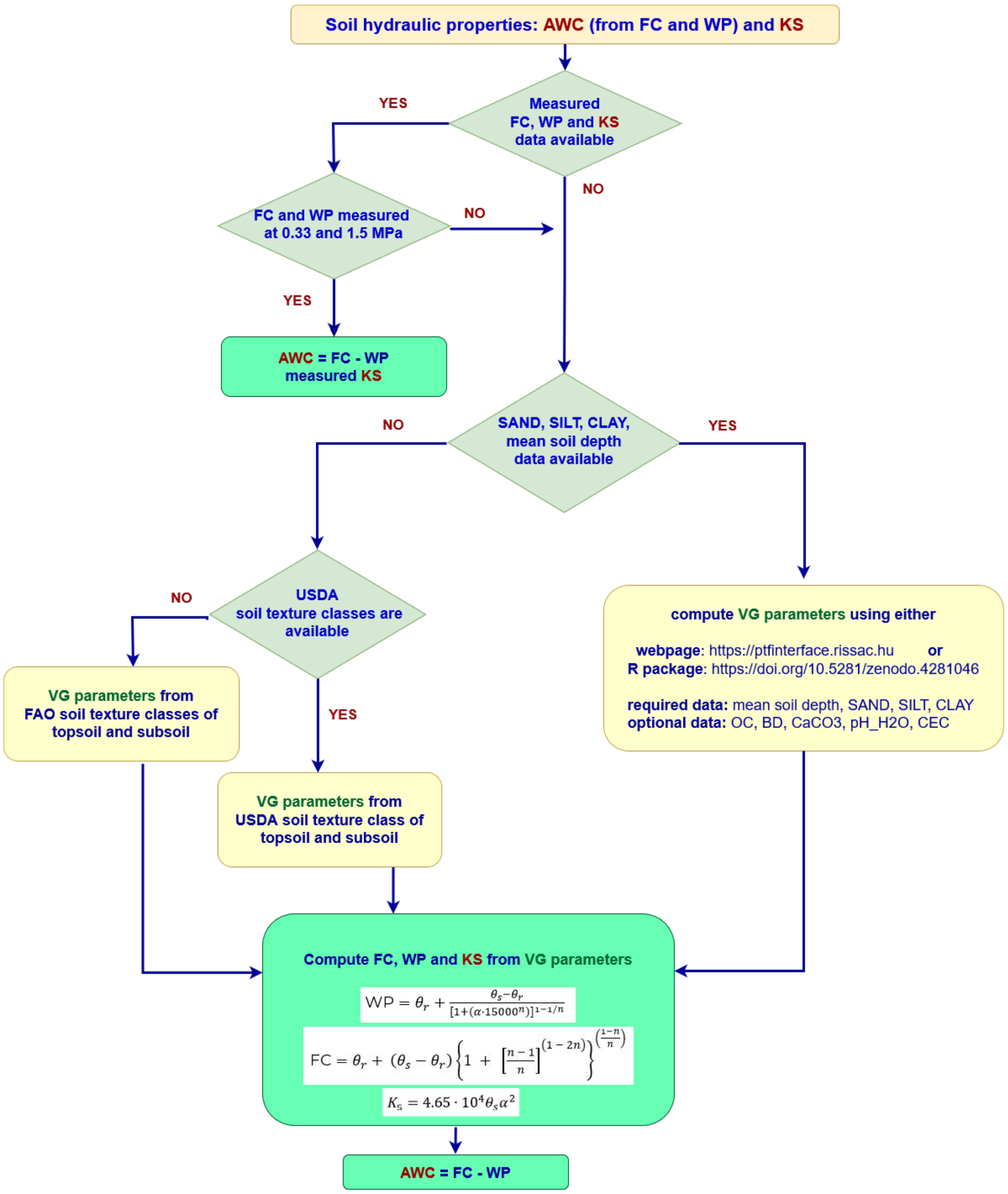

Soil hydraulic properties: AWC (from FC and WP) and KS
Measured FC, WP and KS data available
YES
NO
FC and WP measured at 0.33 and 1.5 MPa
NO
YES
AWC = FC - WP
measured KS
SAND, SILT, CLAY, mean soil depth data available
NO
YES
USDA soil texture classes are available
NO
YES
compute VG parameters using either
webpage: https://ptfinterface.rissac.hu or
R package: https://doi.org/10.5281/zenodo.4281046
required data: mean soil depth, SAND, SILT, CLAY
optional data: OC, BD, CaCO3, pH_H2O, CEC
VG parameters from FAO soil texture classes of topsoil and subsoil
VG parameters from USDA soil texture class of topsoil and subsoil
Compute FC, WP and KS from VG parameters
$\mathrm{WP} = \theta_r + \frac{\theta_s - \theta_r}{[1+(\alpha \cdot 15000^n)]^{1-1/n}}$
$\mathrm{FC} = \theta_r + (\theta_s - \theta_r)\left\{1 + \left[\frac{n-1}{n}\right]^{(1-2n)}\right\}^{\left(\frac{1-n}{n}\right)}$
$K_s = 4.65 \cdot 10^4 \theta_s \alpha^2$
AWC = FC - WP

Figure 3.4: Flowchart of deriving soil hydraulic properties. AWC: available water capacity, FC: water content at field capacity, KS: saturated hydraulic conductivity, VG parameters: parameters of the van Genuchten model, WP: water content at wilting point.

a) If only soil texture classes and topsoil/subsoil distinction are available, the `euptfv1` class pedotransfer function (which is a look up table approach) (Tóth et al., 2015) could be used to predict the parameters of the van Genuchten model. Table 3.1 and Table 3.2 show the look up tables depending on the type of soil texture classification. Topsoil is the surface layer, all the layers below that are subsoils. The modified FAO and the USDA texture classes could be derived based on clay, silt and sand content with the `soiltexture` R package (Moeys, 2014).

Table 3.1: Look up table to assign the parameters of the van Genuchten model to the FAO soil texture classes. Units of the parameters: θ_r ($cm^3\ cm^{-3}$), θ_s ($cm^3\ cm^{-3}$), α (cm^{-1}), n (−), m (−).

	Modified FAO texture classes	θ_r	θ_s	α	n	m
Topsoils						
	coarse	0.045	0.438	0.0478	1.3447	0.2563
	medium	0.000	0.459	0.0309	1.1920	0.1611
	medium fine	0.000	0.432	0.0094	1.2119	0.1749
	fine	0.000	0.478	0.0403	1.1176	0.1053
	very fine	0.000	0.522	0.0112	1.1433	0.1253
	organic	0.111	0.697	0.0069	1.4688	0.3192
Subsoils						
	coarse	0.057	0.404	0.0426	1.5349	0.3485
	medium	0.000	0.428	0.0347	1.1725	0.1471
	medium fine	0.000	0.418	0.0066	1.2173	0.1785
	fine	0.000	0.430	0.0011	1.2290	0.1863
	very fine	0.000	0.511	0.0002	1.4048	0.2882
	organic	0.000	0.835	0.0113	1.2256	0.1841

Table 3.2: Look up table to assign the parameters of the van Genuchten model to the USDA soil texture classes. Units of the parameters: θ_r ($cm^3\ cm^{-3}$), θ_s ($cm^3\ cm^{-3}$), α (cm^{-1}), n (−), m (−).

	USDA texture classes	θ_r	θ_s	α	n	m
Topsoils						
	sand	0.061	0.411	0.0258	1.8005	0.4446
	loamy sand	0.052	0.475	0.0341	1.4846	0.3264
	sandy loam	0.000	0.441	0.0750	1.1904	0.1599
	loam	0.000	0.491	0.0347	1.1931	0.1618
	silt loam	0.000	0.424	0.0074	1.2545	0.2029
	silt	0.009	0.465	0.0042	1.4853	0.3267
	sandy clay loam	0.000	0.409	0.0700	1.1335	0.1178
	clay loam	0.000	0.465	0.1284	1.1160	0.1040
	silty clay loam	0.000	0.463	0.0107	1.1892	0.1591

	USDA texture classes	θ_r	θ_s	α	n	m
	sandy clay	0.192	0.523	0.0351	1.4455	0.3082
	silty clay	0.000	0.455	0.0309	1.1110	0.0999
	clay	0.000	0.499	0.0234	1.1200	0.1072
	organic	0.111	0.697	0.0069	1.4688	0.3192
Subsoils						
	sand	0.034	0.368	0.0356	1.7767	0.4372
	loamy sand	0.037	0.423	0.0419	1.4222	0.2968
	sandy loam	0.000	0.437	0.0681	1.1966	0.1643
	loam	0.000	0.432	0.0336	1.1701	0.1454
	silt loam	0.000	0.422	0.0077	1.2483	0.1989
	silt	0.009	0.465	0.0042	1.4853	0.3267
	sandy clay loam	0.000	0.384	0.0717	1.1206	0.1076
	clay loam	0.000	0.413	0.0227	1.1191	0.1064
	silty clay loam	0.000	0.408	0.0032	1.1993	0.1662
	sandy clay	0.000	0.365	0.0016	1.1812	0.1534
	silty clay	0.000	0.442	0.0003	1.3861	0.2786
	clay	0.000	0.461	0.0004	1.3027	0.2323
	organic	0.000	0.835	0.0113	1.2256	0.1841

b) If information on mean soil depth, sand, silt and clay content is available, it is recommended to use the `euptfv2` pedotransfer functions (Szabó et al., 2021). The minimum input requirements are:

- mean soil depth (cm), i.e.: $top\ depth\ of\ the\ layer + \frac{bottom\ depth\ of\ layer - top\ depth\ of\ the\ layer}{2}$,
- percentages of clay ($<0.002\ mm$), silt ($0.002 - 0.05\ mm$) and sand ($0.05 - 2\ mm$) content.

Optional further inputs:

- organic carbon content ($mass\ \%$), bulk density ($g\ cm^{-3}$), calcium carbonate content ($mass\ \%$), pH in water ($-$), cation exchange capacity ($cmol^{(+)}\ kg^{-1}$).

Tools to use `euptfv2`:

- user friendly web interface (Szabó et al., 2019),
- `euptf2` R package to use the pedotransfer functions, archived on Zenodo (Weber et al., 2020).

2. Compute FC and WLP from the van Genuchten parameters

For the computation of Available Water Capacity (AWC), Water Content at Wilting Point (WLP) and FC is required. For the computation of WLP use the predicted van Genuchten parameters in the following equations:

$$WLP = \theta_r + \frac{\theta_s - \theta_r}{(1 + (\alpha \cdot 15000^n))^{(1-\frac{1}{n})}} \tag{3.11}$$

For sake of simplicity, FC is often estimated at a fixed soil matric head (e.g. -500 cm, or -330 cm, or -100 cm), depending mainly on the dominant soil textural class in the soil profile, but the FC value can also be predicted through the physically-based analytical equation proposed by Assouline and Or (2014):

$$FC = \theta_r + (\theta_s - \theta_r) \cdot (1 + (\frac{n-1}{n})^{(1-2\cdot n)})^{(\frac{1-n}{n})} \tag{3.12}$$

The meaning of the parameters has already been given under equation (3.10). This computation of FC is self-consistent, dynamic criterion based on soil internal drainage dynamics is applied.

3. Compute Available soil water capacity (AWC)

Based on the FC and WLP computed in step 2, AWC is computed using equation (3.9) (in $cm^3\,cm^{-3}$). The values can then be used in SWAT+ (in $mm\,H_2O\;mm\,soil^{-1}$) without the need for unit conversion.

Saturated hydraulic conductivity (KS)

The saturated hydraulic conductivity (soil_K or SOL_K) can be computed from parameters of the van Genuchten model – which were predicted above for AWC from basic soil properties – by the equation of Guarracino (2007):

$$K_s = 4.65 \cdot 10^4 \cdot \theta_s \cdot \alpha^2$$

$$SOL\_K = K_s \cdot 0.416667 \tag{3.13}$$

where K_s is the saturated hydraulic conductivity expressed in units of $cm\,day^{-1}$, SOL_K is the saturated hydraulic conductivity in $mm\,h^{-1}$.

Hydrologic Soil Group

The Hydrologic Soil Groups (HSG) are based on the infiltration characteristic of the soil and include four groups having similar runoff potential. The groups are defined based on the saturated hydraulic conductivity, depth to high water table and depth to water impermeable layer. More details can be found in U.S. Department of Agriculture Natural Resources Conservation Service (2009). For defining the HSG, the following data has to be considered: depth to water table (m), map of saturated hydraulic conductivity ($\mu m\;s^{-1}$), maximum rooting depth of the soil profile (SOL_ZMAX). The main steps of recommended workflow to define HSG are:

- add the depth to water table to each soil type of the *usersoil* table,
- set the minimum saturated hydraulic conductivity for soil layers 0-50 cm, 0-60 cm and 0-100 cm,
- set the depth of the layer which has KS < 0.01 ($\mu m\;s^{-1}$) (0.036 $mm\;h^{-1}$), this is the depth to impermeable layer,
- define HSG codes based on maximum depth to impermeable soil layer, minimum KS value for different soil depth ranges and depth to water table according to Table 7-1 Criteria for assignment of HSG of the U.S. Department of Agriculture Natural Resources Conservation Service (2009) (Table 3.3).

If no local or regional data is available for depth to water table, the Global Patterns of Groundwater Table Depth dataset (Fan et al., 2013) could be used. If only SoilGrids dataset is used as soil input data – i.e.: no local data is available – , than it is recommended to retrieve the HSG from the global gridded hydrologic soil groups dataset for curve-number-based runoff modeling (HYSOGs250m) dataset (Ross et al., 2018).

Table 3.3: Definition of soil hydrologic groups based on U.S. Department of Agriculture Natural Resources Conservation Service (2009).

Depth to water impermeable layer[1] (cm)	Depth to high water table[2] (cm)	K_s of least transmissive layer in depth range ($\mu m\, s^{-1}$)	K_s depth range (cm)	HSG[3]
<50	—	—	—	D
50 to 100	<60	>40.0	0 to 60	A/D
		>10.0 to $\leq$ 40.0	0 to 60	B/D
		>1.0 to $\leq$ 10.0	0 to 60	C/D
		$\leq$ 1.0	0 to 60	D
	$\geq$ 60	>40.0	0 to 50	A
		>10.0 to $\leq$ 40.0	0 to 50	B
		>1.0 to $\leq$ 10.0	0 to 50	C
		$\leq$ 1.0	0 to 50	D
>100	<60	>10.0	0 to 100	A/D
		>4.0 to $\leq$ 10.0	0 to 100	B/D
		>0.40 to $\leq$ 4.0	0 to 100	C/D
		$\leq$ 0.40	0 to 100	D
	60 to 100	>40.0	0 to 50	A
		>10.0 to $\leq$ 40.0	0 to 50	B
		>1.0 to $\leq$ 10.0	0 to 50	C
		$\leq$ 1.0	0 to 50	D
—	>100	>10.0	0 to 100	A
		>4.0 to $\leq$ 10.0	0 to 100	B
		>0.40 to $\leq$ 4.0	0 to 100	C
		$\leq$ 0.40	0 to 100	D

[1] An impermeable layer has a K_s less than 0.01 $\mu m\, s^{-1}$ (0.0014 $inch\, h^{-1}$) or a component restriction of fragipan; duripan; petrocalcic; orstein; petrogypsic; cemented horizon; densic material; placic; bedrock, paralithic; bedrock, lithic; bedrock, densic; or permafrost. [2] High water table during any month during the year. [3] Dual HSG classes are applied only for wet soils (water table less than 60 cm (24 in)). If these soils can be drained, a less restrictive HSG can be assigned, depending on the K_s.

Tools to derive soil physical properties

Availability of tools applicable for European catchments:

- the equations suggested above are implemented into an R script available at ZENODO with example input and output files (Szabó and Mészáros, 2022);
- the `svatools` R package's function `get_soil_parameters()` and `get_hsg()` provides a simplified version of the equations suggested above. It allows automatic computation of moist bulk density, available water capacity, saturated hydraulic conductivity, moist soil albedo, USLE K factor, maximum rooting depth and hydrologic soil groups for the SWAT+ *usersoil* table using just *SOL_Z*, *SOL_ZMAX*, *CLAY*, *SILT*, *SAND* and *OC* (organic carbon or *SOL_CBN*) parameters for each soil layer and water table depth of the soil types. The function needs a filled excel template with listed parameters filled for available soil layers of each soil type available (or

to be used) in a selected catchment. An example workflow for preparing all necessary SWAT+ soils parameters is provided in the svatools website. The package is archived and could be refereed using Zenodo.

```r
library(svatools)
library(euptf2)
temp_path <- system.file("extdata", "soil_parameters.xlsx",
                         package = "svatools")
soil <- get_soil_parameters(temp_path)
```

3.5 Soil chemical data

Regarding the nutrient content of the topsoil layer, the SWAT+ model can use information on the initial soil nutrient content as input stored in the *nutrient.sol* file. This input is optional for the model, else it uses default values if initial soil nutrient information is not provided (Arnold et al., 2012a). Soil chemical data includes the following properties: initial nitrate, organic nitrogen, labile phosphorus, organic phosphorus concentration of the surface soil layer. If the analysis focuses on nutrient retention, it is important to derive approximate soil nutrient maps for the target area instead of using the model's default values.

3.5.1 Soil phosphorus content

The SWAT+ model requires the labile Phosphorus (P) content (lab_p) of the surface layer in ppm for initialization of the different P-pools. The model's default value is 5 ppm. Labile-P is the amount of P that is available for plants and microorganisms. Consequently, it's the sum of inorganic and organic P absorbed in the soil in a way that it can easily enter the soluble phase (Costa et al., 2016).

According to Chaubey et al. (2006) labile P is the P extracted by an anion exchange resin (Sharpley et al., 1984) and therefore represents solution P plus weakly adsorbed P. Within SWAT the pool "solution P" (Figure 3.5) is actually labile P in conformance with the original EPIC version of the P module as described in Jones et al. (1984) and Sharpley et al. (1984). The initial concentration of solution P in SWAT is of particular relevance, as it will be used for the model-internal initialisation of both mineral P pools (active & stable).

Several methods exist for the determination of different P formats, but the level of P analysed highly depends on the method used for its determination. Most commonly used P test methods are:

- Acid ammonium acetate lactate extraction (AL method; Egnér et al. (1960)), which is applied in the OPTAIN CSs of Belgium (Flanders), Hungary, Lithuania, Norway, Slovenia and Sweden,
- Sodium bicarbonate extraction (Olsen method; Olsen et al. (1954)), which is applied globally. It is the official soil P test in Denmark, England, France, Italy, Spain and this method is used in the LUCAS Topsoil survey (Tóth et al. (2014)).

Both, AL and Olsen P, are approximating plant available P. Most often measured soil P content is not available for the whole modelling target area. However, it is an important input data to analyse the nutrient retention in the CSs. As the LUCAS Topsoil Survey dataset (Tóth et al., 2013) contains measured Olsen-P content data of about 20,000 soil samples in Europe, it can be used to derive approximate maps of soil P content for data scarce areas. A common method is provided for producing Olsen-P maps for any area of the EU member states based on the LUCAS dataset. For the prediction

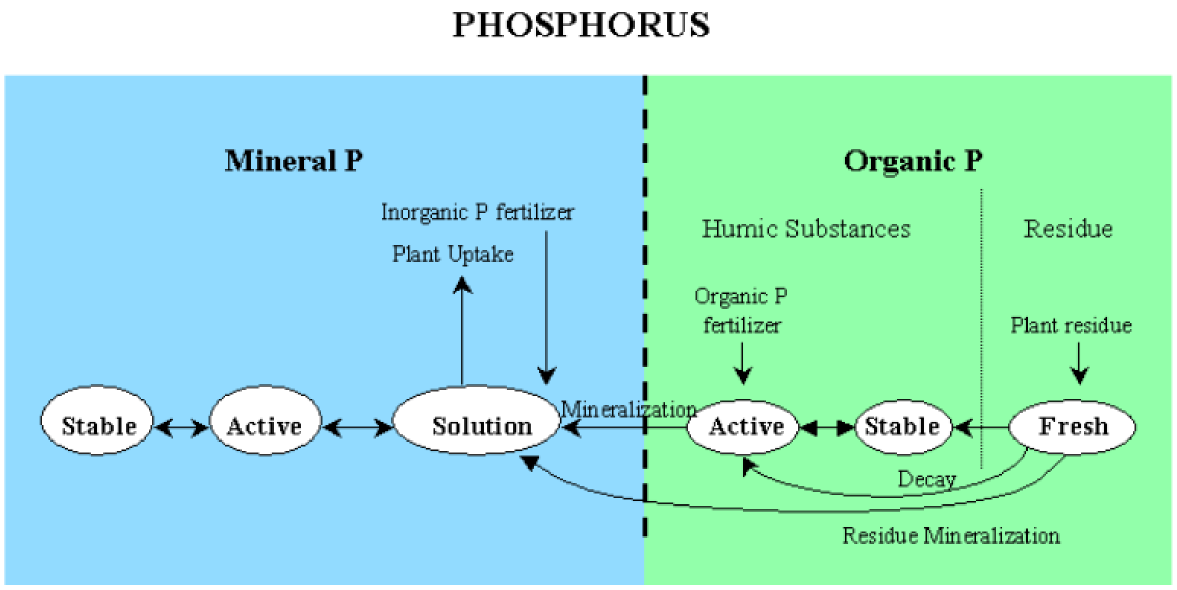


Figure 3.5: SWAT soil phosphorus pools and processes that move phosphorus in and out of pools. Source: SWAT 2009 theoretical documentation.

of the P content of the surface soil layer the geometric mean Olsen P values are calculated by land use/land cover categories using the LUCAS Topsoil Survey dataset. If a local measured dataset is available for some areas or fields of the catchment, those should replace the mean values. This method requires the delineation of land use/ land cover categories of the land use map available for the target area and land use/ land cover categories available in the LUCAS dataset.

Detailed information about a possible workflow, required input data and data preparation are provided in a guideline (Szabó et al., 2022) with R script at Zenodo.

3.5.2 Other soil nutrients

The nitrate content is highly variable in space and time and the dynamic of its amount is significantly influenced by nitrogen fertilization (Zhu et al., 2021). It is thus recommended to use the default value of the SWAT+ model in the case of missing data and use locally available information on nitrogen fertilization in the management table of SWAT+. If there is no measured data for organic nitrogen and organic phosphorus content it is recommended to use the default value, because SWAT+ will initialise the values for the user. The initialization will be based on routines and assumptions that were carried over from the EPIC model and/or CENTURY.

- **Soil carbon**: although not in the nutrients initialization file, SWAT+ will initialize the soil carbon for lower layers using exponential decrease, which the user can control by adjusting the *EXP_CO* (depth coefficient to adjust nutrient concentrations for depth) parameter in the soil nutrients (*nutrients.sol*) file. This way only the topsoil *carbon* parameter value is required by the model for the *nutrient.sol*, but for the computation of soil physical properties in the *usersoil* table it is required for each layer of each soil types. The value for the organic carbon content of the soil layer should be an input in the *soils.sol* database (soil database).
- **Mineral NO3 pool** (*nitrate*): similar like the soil carbon, the mineral NO3 pool will be redistributed over the soil layers using exponential decrease, controlled by *EXP_CO* parameter.

- **Labile P in soil surface** (*lab_p*): if no concentration will be provided, the model will assume a 5 mg/kg default value. Its possible derivation is described in the section above. The distribution will be adjusted with depth in the same way as the other pools.
- **Fraction of soil humus that is active** (*FR_HUM_ACT*): the value can vary from 0 to 1. If the value is "0", then the model will assume a 0.02 default value.
- **Humus C:N ratio** (*HUM_C_N*) and **Humus C:P ratio** (*HUM_C_P*): although active in the setup, currently the model assumes 10:1 C:N ratio and 80:1 C:P ratio for the initialization of stable and active humus pools, while other assumptions are made for the initialization of passive and slow humus pools, microbial, metabolic, structural and lignin litter pools. Therefore, we recommend leaving the default values of 10 and 80 in the model setup step.
- **Other soil nutrient pools** i.e., water soluble pools, are not currently in use. Therefore, the values for *INORGP*, *WATERSOL_P*, *H3A_P*, *MEHLICH_P*, *BRAY_STRONG_P* are only placeholders and should be set to zero.

3.6 Reservoirs

Reservoirs were extensively discussed in the model setup chapter, but mainly from the point of view of their GIS features and connectivity. It was also mentioned that `SWATbuildR` writes the reservoir input files with some default values (with an exception of the area that is read directly from the GIS). The values of at least some of these parameters should be updated as a part of the model parametrization described in this section.

The term "reservoirs" is here used in the SWAT+ context, that does not differentiate between the size nor between the natural or managed character of the water body. In other words, both large lakes and tiny ponds will be represented in the model setup by the same "reservoir" feature. Also it does not matter if the object is natural or man-made (i.e. constructed sedimentation ponds or reservoirs with outflow release rules). It is up to the modeller to reflect the character of the given water body by its parameters or additional features. Finally, in contrast to previous SWAT model versions, for model setups created with `SWATbuildR`, the type of connectivity with channel network does not even matter. In older SWAT versions, objects located off the channel network were called "ponds" and served a different purpose than "reservoirs" located on the channel network. Now, both types of objects are called reservoirs and their role is handled by the object connectivity.

Reservoir-related processes such as their water balance, sediment and nutrient budgets did not change substantially between SWAT2012 and SWAT+, therefore it is wise to rely on valuable literature studies that investigated reservoirs in SWAT. The paper of Jalowska and Yuan (2019) is highly recommended in this respect, as it guides the reader through other literature on the use of reservoirs in SWAT, it contains useful suggestions on reservoir parametrization, as well as a possible calibration workflow for catchments in which impoundments play an important role. Another remarkable paper is the one of Jingwen Wu (2022), focusing on reservoir operation functions in SWAT+.

The most important reservoir input parameters are stored in the *'hydrology.res'* file. Particular attention should be paid to reservoir storage capacities represented by two parameters: *vol_ps* and *vol_es*, meaning reservoir volume at principal and emergency spillways, respectively. The emergency spillway concept is related to flood control, and therefore it might be easy to obtain the values of both parameters for flood control reservoirs. For other types, it might be that only one volume will be available (usually representing principal spillway), and some assumptions have to be made to derive volume at emergency spillway. In practice, *vol_es* is often set as something in the range between 1.1**vol_ps* and 1.2**vol_ps*. The area at emergency spillway *area_es* should be changed accordingly with respect to the area at principal spillway *area_ps*.

Two other parameters from the *'hydrology.res'* file that control the reservoir water budget are hydraulic conductivity of the reservoir bottom (*k*) and evaporation coefficient (*evap_co*). It should not be expected that measured data would be available for them, in most cases. For seepage, related *soil_k* values from the underlying soil could be used as a proxy (see soil parameter chapter). Baldan et al. (2021) assumed the value of *k* equal to 1 mm/h when parametrizing sedimentation ponds as NSWRM measures.

Reservoir decision tables in SWAT+ handle the process of setting reservoir outflow rules. This is a difference compared to SWAT2012, in which several different (simple) outflow simulation methods were available.

Reservoirs also play an important role for sediment and nutrients transport. Relevant files are *'sediment.res'* and *'nutrients.res'*. For sediment, the critical parameter is *sed_amt* (Equilibrium sediment concentration in the reservoir), controlling the sediment settling process. Baldan et al. (2021) set its value to 80 mg/l in their case study with sedimentation ponds in Austria. For N and P, the key parameters are settling rates. Both of them can be specified as seasonally variable, since SWAT+ distinguishes between the mid-year settling period and the rest of the year. The user should define the start and end of the higher settling period to reflect the impact of temperature and other seasonal factors.

Last but not least, the reservoir parameters should be initialized in the *'om_water.ini'* file, both in terms of capacity, sediment and nutrients. It is likely that using the warm-up period should help to decrease the importance of the initialization parameters, however care should be taken since the behaviour of reservoir (storage, settling) could depend on the initial values.

3.7 Water diversions

This functionality was developed to allow users to set up water rights object and provide more flexibility in assigning water rights to individual fields (HRUs). The SWAT+ Water Allocation Module is still work in progress and is not functional in the current revision of SWAT+Editor or other software, apart from the SWAT+ itself (November, 2022). This option must be used if either irrigation or municipal withdrawals or water transfers are present in the CS. The functionality allows the user to "move" the water across different parts of the basin in a specific order.

The combination of the water diversion functionality being relatively new, the SWAT+ flexibility, the lack of documentation, and numerous possibilities for water transfers within the basin, makes it difficult to give an example to all possible use cases. At this stage, we recommend contacting the SWAT+ development team in case of a complex water diversion system within the CS. Here, a description of the necessary files and their formats for a general water withdrawal setup approach is provided.

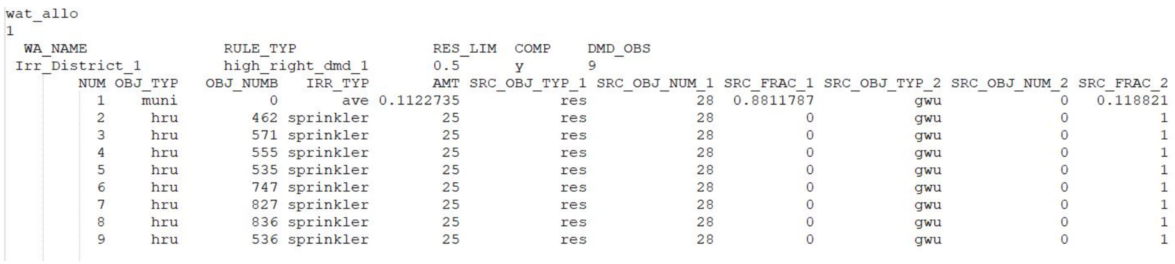
Recommended workflow to setup water diversion in your model:

1. Create the water rights object following the example provided in this chapter (*'water_rights.wro'* file).
2. Alter the standard flo_con_std Decision Table to meet your requirements or create your own DT. Make sure that the name of the DT is the same as in the *'water_rights.wro'* file created in step 1.
3. Input the name of the created in step 2 DT in the *'chandeg.con'* file in the appropriate channel number.
4. Save all the changes and run the model.

Note, that this functionality can be setup only manually (as of November 2022). The SWAT+Editor does not support this functionality.

Step 1. Create the water rights object

First, a water rights object has to be defined in *water_rights.wro* (Figure 3.7):



```
wat_allo
1
  WA_NAME                RULE_TYP               RES_LIM  COMP    DMD_OBS
 Irr_District_1         high_right_dmd_1       0.5      y       9
       NUM OBJ_TYP   OBJ_NUMB   IRR_TYP       AMT SRC_OBJ_TYP_1 SRC_OBJ_NUM_1 SRC_FRAC_1 SRC_OBJ_TYP_2 SRC_OBJ_NUM_2 SRC_FRAC_2
         1    muni          0       ave 0.1122735           res            28  0.8811787           gwu             0   0.118821
         2     hru        462 sprinkler        25           res            28          0           gwu             0          1
         3     hru        571 sprinkler        25           res            28          0           gwu             0          1
         4     hru        555 sprinkler        25           res            28          0           gwu             0          1
         5     hru        535 sprinkler        25           res            28          0           gwu             0          1
         6     hru        747 sprinkler        25           res            28          0           gwu             0          1
         7     hru        827 sprinkler        25           res            28          0           gwu             0          1
         8     hru        836 sprinkler        25           res            28          0           gwu             0          1
         9     hru        536 sprinkler        25           res            28          0           gwu             0          1
```

Figure 3.6: Example of the water_rights.wro file

First lines describe the source object and number of demand objects:

Field	Description	Type
WA_NAME	name of the water allocation object	string
RULE_TYP	rule type to allocate water (DT name, i.e.: "*high_right_dmd_1*")	string
DMD_OBS	number of demand objects	integer
COMP	allow compensation flag (y=yes, n=no)	string
DMD_OBS	number of demand objects	integer

Demand: can be irrigation demand from an HRU, municipal demand, or demand to transfer to another source object (channel, reservoir, aquifer). For irrigation demand, the HRU number, decision table for triggering irrigation, and irrigation depth (mm) are input. For municipal demand, a so-called muni number is input (see '*water_rights.wro*' above). The user then has the option to input an average daily demand or a recall name that can be daily, monthly, or annual.

Next, the demand objects are described with the allocation rule-set:

Field	Description	Type
NUM	demand object number	integer
OBJ_TYP	object type (channel=cha; reservoir=res; aquifer=aqu; hru=HRU)	string
OBJ_NUM	number of the object type	integer
AMT	amount: m^3 per day for *muni* and mm for *HRU*	real

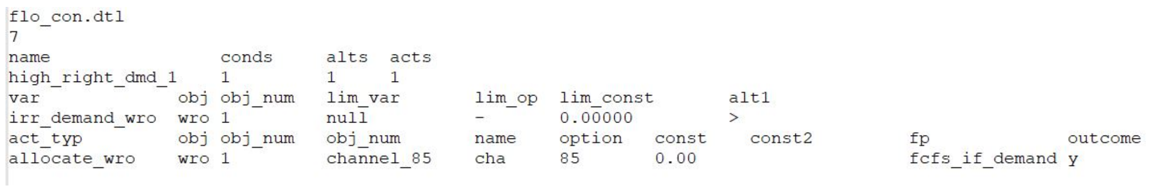
Next, the **sources** are defined (see table below). The sources are listed in order of selection and the fraction from each source is input. The other input is compensation – if other sources are not available, the current source may be allowed to compensate for the demand. The model goes through all sources in the order listed and allocates based on the fractions. The model loops through the sources again, checking to see if compensation is allowed.

Field	Description	Type
SRC_OBJ_TYP_X	source type (channel=cha; reservoir=res; aquifer=aqu; unlimited source=gwu)	string
SRC_OBJ_NUM_X	number of the source type	integer
SRS_FRAC_X	fraction of the demand to be satisfied	real

Step 2. Define the *flo_con* Decision Table

Refer to the standart flo_con_std Decision Table and make adjustments to meet specific requirements of your CS, or create your own DT. Make sure that the name of the DT is the same as in the *'water_rights.wro'* file created in step 1. In this case the name is "*high_right_dmd_1*" (see Figure below).

The decision table for *high_right_dmd_1* is in *flo_con.dtl*:

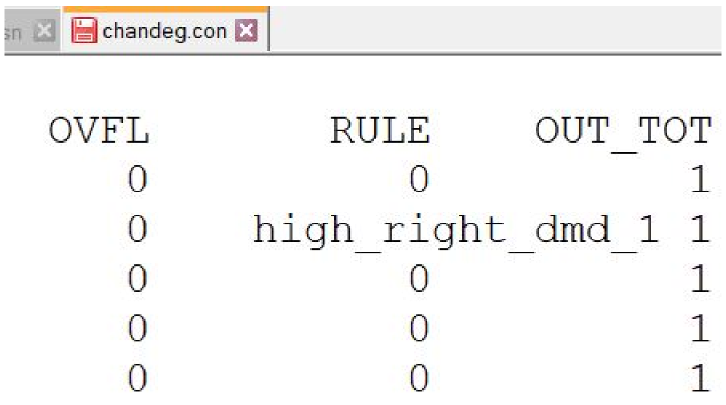


```
flo_con.dtl
7
name                 conds      alts  acts
high_right_dmd_1     1          1     1
var             obj obj_num     lim_var       lim_op  lim_const       alt1
irr_demand_wro  wro 1           null          -       0.00000         >
act_typ         obj obj_num     obj_num       name    option    const    const2         fp              outcome
allocate_wro    wro 1           channel_85    cha     85        0.00                    fcfs_if_demand  y
```

Figure 3.7: Example of the flow_con decision table.

Step 3. Input the name of the created in step 2 DT in the *'chandeg.con'* file

The decision table for conditioning the amount transferred from the channel directly to the HRU is input in the connect file for the channel (*'chandeg.con'*) in column 'RULE' (see 3.8). The *'chandeg.con'* file contains the channel properties for the modelled basin. The developed DT name for water diversion has to be an input to the channel, where the diversion occurs. This way the model, rather than rout the water normally, will reach into the DT, where the rules are defined. Based on the water rights object, the re-routing will be distributed on the current conditions for the time step (day).



```
OVFL          RULE      OUT_TOT
   0             0            1
   0  high_right_dmd_1        1
   0             0            1
   0             0            1
   0             0            1
```

Figure 3.8: Example of the connect file, which includes a water diversion DT.

Step 4. Save all the changes and run the model.

After all the manual edits have been performed, the files can be saved and SWAT+ can be executed from the model directory. Because those edits have been setup manually, any alteration via the SWAT+Editor will overwrite these changes. Hence, a backup of the specific files or the entire model is always recommended. If your CS requires a more elaborate setup – contact the SWAT+ development team with request for assistance.

3.8 Point sources

Most of the domestic waste produced by human settlements and commercial activities are collected by sewerage networks and treated by Waste Water Treatment Plants (WWTPs) before being discharged to the surface water (Vigiak et al., 2020). Such facilities, releasing connected and treated waste, are in SWAT+ referred to as point sources. They are not the main focus in OPTAIN, since the project is dealing with small agricultural catchments where diffuse pollution from agriculture should dominate. In all scenario or optimisation runs in OPTAIN, no assumptions about changes in point source loadings will be made and no measures will be implemented to reduce pollution from these sources. Hence, it sounds as if point sources could even be neglected in the model setups. One reason why they should not be neglected is model calibration. In calibration, we are comparing simulated nutrient loads to observed loads. If significant amounts of loadings from point sources are not accounted for in the model setup, automatic calibration will likely drive unrealistic changes in some of the model parameters in order to compensate for these unaccounted loads. To prevent from such situations, all major point sources should be represented in the model. Another reason is augmentation of streamflow due to point source discharges, particularly visible during low flow periods. Research in a medium-sized catchment in Poland showed that up to 100% of gauged streamflow originated from WWTPs in a dry August of 2015 (Somorowska and Łaszewski, 2017).

SWAT+ requires input data on measured volumes of effluent and loads of sediment, nutrients and other constituents for each point source. Point source locations were already discussed in Point sources locations section. The loads are calculated based on volumes and average concentrations. The latter depend on WWTP treatment level (T1 - primary, T2 - secondary or T3 - tertiary). This information is not required by SWAT, but should be indirectly reflected in the effluent load data. In the absence of load data, this information may also help to get a rough estimate of treatment removal efficiencies, based for example on data assembled for a Europe-wide study on domestic waste emissions to European waters presented in Table 3.7 (Vigiak et al., 2020). Removal efficiencies together with the number of population connected to a WWTP and basic characteristics of domestic sewage (e.g. average annual per capita production of 4.5 kg N and 0.55 kg P; (Jönsson and Vinnerås, 2004)) could then be used to estimate the effluent loads.

Table 3.7: Treatment removal efficiencies [%] per treatment level T and constituent adopted from Vigiak et al. (2020).

Treatment level	N	P	BOD5
Septic tank	25	30	40
T1 - primary	25	30	50
T2 - secondary	55	60	94
T3 - tertiary	80	60	96
T3P - tertiary with P removal	80	90	96

There are four options for temporal resolution of input data: constant (average), annual, monthly and daily. If the first option is used, point source parameters are interpreted by SWAT+ as the so-called export coefficients. For all remaining three options with temporal data, the so-called *recall* files are used (e.g. *recall.rec* with point source IDs and *.rec* files with time series data). A manual how to prepare the input data is included in the SWAT+Editor documentation.

As a matter of fact, the most problematic issue about point sources may be data acquisition, which is highly country-specific. Among various potential data owners are: (1) environmental agencies, (2) water authorities, (3) municipal authorities, (4) WWTP managers, (5) statistical offices, or (6) companies. Often, data from more than one source are available and they may be different. Asking local experts about reliability of different data sources is recommended.

It is important to check if the year of construction (or major modernization/upgrade) of WWTP intersects with the planned calibration/validation period. If this is the case, only temporal input data options (as time series) should be used, to reflect occurrence/change of new loadings during the simulation period.

A very common problem will be that only a part of the required input parameters might be available. For example, only total forms of N and P or only BOD could be available, whereas the model requires a division into organic and mineral form of both N and P (and in case of nitrogen, specifically NH_4, NO_3 and NO_2). Proxy solutions should be applied in such cases. For example, unreported parameters could be "extrapolated" from other WWTPs with similar conditions or taken from the literature or experts.

Although the safest solution is always to include all existing point sources, a threshold approach may be used to decide whether a certain WWTP is significant enough to have a measurable effect on the model outputs. If only data on the amount of effluent are available, they can be compared to the data from the main discharge gauge used in calibration. An example criterion for neglecting a given point source could be: average WWTP discharge is less than 2% of the minimum of the average monthly flows.

It has to be taken into account to keep the balance between water withdrawals and point sources. Water that is discharged from domestic WWTPs to stream network in the studied catchment must have been withdrawn from a certain source. If this source (aquifer or surface water) is inside the catchment, then it should be accounted for in the water withdrawals. Typically, the amount withdrawn should be somewhat higher than the amount discharged as point source, but this may not always be the case due to differences between water supply and sewer networks, as well as possible additional sewage transported by vacuum trucks.

3.8.1 Domestic waste from disconnected areas

In remote rural and peri-urban areas households are often not connected to sewerage network and are equipped with individual systems, which may have one of two forms: (1) septic systems collecting and treating domestic waste before releasing it to the environment or (2) cesspits collecting waste without treatment. Septic systems can have variable performances in nutrient and pathogen removal, in worst case they can be failing and causing major water quality problems in surface waters (Withers et al., 2014). Cesspits can also be problematic, as in theory they should be sealed and sewage should be transported via vacuum trucks to WWTPs (and thus be included in WWTP load statistics). However, leaking cesspits remain to be an issue in some countries (Grochowska et al., 2020).

Representation of these pollution sources in process-based water quality models such as SWAT has always been challenging. SWAT2012 provided an option to model septic systems using the so-called "biozone" algorithm (Jeong et al., 2011). The review of the SWAT Literature Database showed that only a handful of papers applied this option, of which none dealt with Europe. For this reason, process-based simulation of septic systems is not feasible in OPTAIN and pollution from this source will be not accounted and contributing to the overall uncertainty. To quantify this uncertainty, one possibility is to estimate average loads from these sources using some basic assumptions about the population not connected to sewerage system in the catchment, per capita emission of N, P and BOD5 and septic tank treatment levels (see e.g. (Vigiak et al., 2020) for a possible approach). The annual loads calculated in this way will be "raw", i.e. describing the septic tank effluent that is released to the soil. The effective load of pollutants reaching the surface waters would depend on numerous factors, such as the distance to nearest stream, groundwater depth and soil permeability (Withers et al., 2014), which are hard to be simplified as average reduction coefficients. In any case, the calculated value could at least be represented as a fraction of the total watershed load, which would be a measure of uncertainty in this case.

3.9 Tile drainage

Tile drainage processes in agriculture-dominated catchments could have significant impact on water and pollutant routing as well as plant growth. Thus, correct representation of tile drains is crucial, especially in flat areas with heavy soils (dominated by silt or/and clay).

Information about location of tile drains was already required by the `SWATbuildR` to be included in the land object layer which was discussed in the Land object input section. In this section the focus is on tile drainage parameters required by SWAT+. The model provides two options for simulating tile drains that are controlled by *tile_drainage* parameter in the *'codes.bsn'* file (see section Additional parameters). The default option is using a simple model adopted from swat2005 (*tile_drainage* = 0), but the recommended method is to use the Hooghoudt and Kirkham equations included already in SWAT2012 (Moriasi et al., 2012) (*tile_drainage* = 1). Tile drainage activation is done in the *'landuse.lum'* file by providing *tile* parameter (name of tile drain parameter set saved in the *'tiledrain.str'* file) for selected land use. Table 3.8 presents parameter names from the *'tiledrain.str'* file, their default values and recommended ranges. Some of these parameters, e.g. those related to tile drainage system dimensions (*dp*, *rad*, *dist*) should be more easy to identify based on locally available data while others may be more problematic. Parameter *dp* typically varies between 800 and 1200 mm, although in deeper and heavier flatland soil it could be deeper (1200 - 1500 mm). Parameters *t_fc* and *drain* are very much depending on the soil hydraulic conductivity and the shallow water table dynamics (more considerations of the latter in Skaggs (2017)). It could be calculated if drainage parameters are known and otherwise, the values of 48-64 h for average soils and 60 - 80 h for heavier soils could be tested.

Table 3.8: Tile drain model parameters in *'tiledrain.str'* file.

Description	**Parameter**	**Default**	**Range**
Depth of drain tube from the soil surface (mm)	dp	1000	0 - 6000
Time to drain soil to field capacity (hrs)	t_fc	48	0 - 100
Drain tile lag time (hrs)	lag	24	0 - 100
Effective radius of drains (mm)	rad	30	3 - 40
Distance between two drain tubes or tiles (mm)	dist	15000	7600 - 30000
Drainage coefficient (mm/day)	drain	10	10 - 51
Pump capacity (mm/hr)	pump	1	0 - 10

Description	Parameter	Default	Range
Multiplication factor to determine lateral ksat from SWAT ksat input value	lat_ksat	1	0.01 - 4

In addition, attention should be paid to *PERCO* parameter (percolation coefficient) from the *'hydrology.hyd'* file. This parameter replaced *DEP_IMP* (depth to impervious layer) from SWAT2012, the parameter that had to be set to an appropriate value in order for tile drainage to function. PERCO controls percolation from the soil bottom and can be used to limit percolation if an impermeable layer or high water table is present (Wagner et al., 2022). Thus it makes sense to set it to a different value for drained HRUs. It is recommended to set this parameter to the value of 0.05 for drained HRUs.

Tile drainage was so far tested in two published SWAT+ studies (Bailey et al., 2022; Wagner et al., 2022) and one preprint (Sharma et al., 2022). For example Wagner et al. (2022) applied $t_{fc} = 24h$ and $lag = 48h$ in a northern German case study with a large proportion of drained fields. All these studies evaluated tile drainage component of SWAT+ and are thus a useful resource for case studies in which tile drainage is implemented in the model setup. Some of the parameters from Table 3.8 could be tested in calibration, but tile drain outflow data or at least tile drain average contribution to runoff would be desired to better constrain the parameters.

Setting appropriate values to tile drainage parameters is particularly important if some drainage-related NSWRMs, such as controlled outflow from drainage systems, are to be simulated in the studied catchment.

3.10 Atmospheric deposition

Although atmospheric deposition of nitrogen has been declining in Europe since 1990s due to reduced emissions, catchment response to current deposition levels can be variable (Kaste et al., 2020). A source apportionment study performed in three catchments in Europe showed that atmospheric deposition constituted between 3 and 16% of total nitrogen inputs (Grizzetti et al., 2005). Setting atmospheric deposition to appropriate level should help better constrain other parameters responsible for nitrogen transport processes in the catchment.

SWAT+ considers nitrogen deposition (NH_4 and NO_3), but not phosphorous deposition. Both wet and dry deposition is taken into account. Wet deposition is absorption of compounds by rain and snow as they fall and is expressed in mg/l, whereas dry deposition is direct adsorption of compounds to water and land surfaces, expressed in $kg/ha/year$.

SWAT+ allows constant, annual or monthly input data on atmospheric deposition. The choice is controlled by *atmo* parameter in the *'codes.bsn'* file (see more Additional settings section). Deposition data have similar status as the weather station data, so parameters can be entered for different stations (real or virtual), if data are available, although for small catchments using data from the nearest station should be sufficient. The details how to prepare the input file *'atmo.cli'* are explained in the SWAT+ input/output documentation.

Atmospheric deposition data should be available from the environmental agencies. In case of missing values, a good alternative is data from "the co-operative programme for monitoring and evaluation of the long-range transmission of air pollutants in Europe", inofficially European Monitoring and Evaluation Programme (EMEP). The EMEP portal provides gridded 0.1° resolution netCDF data over Europe. The parameters of interest are: dry deposition of oxidized nitrogen per m^2

grid (*DDEP_OXN_m2Grid*), wet deposition of oxidized nitrogen (*WDEP_OXN*), dry deposition of reduced nitrogen per m^2 grid (*DDEP_RDN_m2Grid*), wet deposition of reduced nitrogen (*WDEP_RDN*).

3.11 Additional settings

General attributes of a catchment (parameters and codes) are defined in the basin input files called: *'parameters.bsn'* and *'codes.bsn'*, respectively. These attributes control a range of physical processes at the catchment level. Since they are automatically set to the default or recommended values listed in the variable documentation, the user should review them and adjust if needed before running the simulation. Here we focus on "codes" which are a special type of a parameter that can only have a finite number of integer values (typically: 1, 2, 3, ...) that represent a certain option that the model would use for simulation of a certain process. A general overview of available codes and their meaning is provided both in the SWAT+ Editor Documentation and the SWAT+ IO Documentation, although there exist small discrepancies between these two sources and the actual input file codes.bsn.

Not all of these codes will be relevant in OPTAIN. Several deal exclusively with the sub-daily simulation option (which is outside the scope in OPTAIN), others are in testing phase or even not active. In the text below we focus on a subset of codes that seem most relevant for consideration by OPTAIN modellers.

3.11.1 PET method

The parameter called *pet* provides an option for choosing one of four methods of estimating PET (0 - Priestley-Taylor; 1 - Penman-Monteith; 2 - Hargreaves; 3 - read in potential ET values). It is recommended to use the Penman-Monteith (PM) method for the following reasons:

- it will allow to take the benefit of the full range of climate variables that were bias-corrected in WP3;
- PM method is recommended by FAO and most physically-based out of available options;
- PM method is currently the only one that would allow for including the physiological effect of elevated atmospheric CO_2 on actual Evapotranspiration (ET) and plant growth in SWAT+ (Gunn et al., 2021).

3.11.2 Channel routing method

The parameter called *rte_cha* provides an option for choosing one of two channel water routing methods (0 - variable storage; 1 - Muskingum). The choice of the routing method may have an impact on streamflow results, and issues were identified in both of these options in some of the SWAT2012 revisions (Nguyen et al., 2018). Although these methods were not sufficiently tested, it is recommended to start with Muskingum as the first choice method in OPTAIN, and apply the variable storage method in case if the former is causing problems.

3.11.3 Stream water quality

The parameter called *wq_cha* provides an option for choosing whether in-stream transformation of nutrients using the QUAL2E algorithms is active (1) or not (0). There are very few studies that evaluate the effect of using in-stream nutrient transformations on simulated processes and model performance. Yuan and Chiang (2015) showed that there is a strong effect on nutrient parameter sensitivities. Due to very limited testing of these options in SWAT+, it is recommended to test both of them in OPTAIN.

3.11.4 Daily curve number calculation

The parameter called *cn* provides an option for choosing the daily curve number calculation method (0 - as a function of soil moisture, which is the original SCS approach; 1 - as a function of plant evapotranspiration; 2 - like option 0 but retention is adjusted for mildly-sloped tile drained watersheds). Calculation of the daily CN value as a function of plant ET was added because the default method was predicting too much runoff in shallow soils, while for the plant ET method, daily CN value is less dependent on soil storage and more dependent on antecedent climate (Williams et al., 2012). This method also allows to use the parameter *CNCOEF* (plant ET curve number coefficient in *'hydrology.hyd'* file), an ET weighting coefficient used to calculate the retention coefficient in SCS equation, in model calibration. Overall, based on own experience and some studies showing superior behaviour of plant ET method (Yen et al., 2015), this one is recommended in OPTAIN.

3.11.5 Soil phosphorus calculation

The parameter called *soil_p* provides an option for choosing one of the two soil phosphorous routines (0 - an “old” one from SWAT2005; 1 - the “new” one, based on the paper of White et al. (2010)). Using the “new” soil P routine is recommended in OPTAIN.

3.11.6 Using lapse rates for weather data

The parameter called *lapse* provides an option for using temperature and precipitation lapse rates for weather data (0 do not use lapse rates; 1 - use lapse rates). This option may be worth testing in catchments with substantial elevation gradient and a limited amount of weather stations. If this option is active, *tlaps* and *plaps* parameters should be defined in *'parameters.bsn'* file.

3.11.7 Plant growth stress (de)activation

The parameter called *nostress* provides an option for activating or deactivating plant growth stresses (0 - all stresses applied; 1 - turn off all plant stress; 2 - turn off nutrient plant stress only). By default, *nostress* should be set to 0, however, for special purposes related to model verification, it could be set to 1 or 2.

3.11.8 Other codes

Some codes were already covered in other sections, for example *tile_drainage* was discussed in Tile drainage section, whereas *atmo_dep* in Atmosperic deposition section.

Other codes from the *'codes.bsn'* file could be kept with the default values. They are either in the testing phase or do not seem relevant for OPTAIN.

Chapter 4

Agricultural land management

Simulating agricultural management is one of the biggest strengths of the SWAT+ model. It allows for a detailed consideration of many operations relevant for the management of crops and soils, such as tillage, sowing/planting, fertilisation, irrigation, and harvest - each defined for a specific time in the year or conditioned on state variables such as soil moisture or plant nutrient demand. In order to assess the effectiveness of water and nutrient retention measures, our ambition in OPTAIN is to represent the current management practices as well as possible for each agricultural field. This requires local data (e.g. the type and timing of operations based on local stakeholder knowledge) and - in case of limited data availability - also other sources of information such as remote sensing data to derive crop rotations over a certain time period.

This chapter provides guidance on how the agricultural input data collected in OPTAIN can be efficiently translated into field-specific management operation schedules for full crop rotations as required by SWAT+.

The procedure described in this section aims at using the `SWATfarmR` R package to write the final SWAT+ management schedules. We strongly recommend using this package in OPTAIN for the following reasons:

1. It allows for a randomization of operation dates within a given time window to ensure that a certain operation is not applied on only one specific date across the whole catchment (which would be unrealistic and flaw the modelling results, e.g. leading to nitrate concentration peaks due to a common date of fertiliser application).
2. It allows constrain the operations only for suitable weather conditions (e.g. no rain event on a given day and the days immediately before) to make sure operations do not take place when the soil is too wet.
3. Using decision tables to address the issues in reasons 1 and 2 would be also possible and even smarter from an algorithmic point of view (e.g. because the simulated soil moisture could be directly taken into account). However, the SWAT+ models being set up for OPTAIN are expected to include a large number of routing units (land and water objects) and decision tables would further increase computation time, which is unfavourable for any hard calibration and the later multi-objective optimization of NSWRM implementation (as these tasks require a large number of SWAT+ simulations).

4.1 Agricultural management data

4.1.1 Crop data

In Europe, crop data at sufficient resolution and quality can be hard to get. In the best case, field-level crop information for at least five consecutive years can be obtained from the Integrated Administration and Control System (IACS) of the Common Agricultural Policy. However, these data are highly protected and only a few case studies might get access. If no crop data is available, remote sensing data could be used to derive crop rotations on field-level for a certain time period as required in OPTAIN. Scripting languages such as Python, R, JavaScript, or tools available in QGIS and ESRI tools could be used to locate, download, clean, correct, and classify aerial images or radar data to obtain needed classes. Tools such as Google Earth Service make all operations run on a cloud, thus saving from the need of having large, computationally capable machines. Yet, the application of remote sensing adds another level of uncertainty as classification algorithms would involve different types of classification errors.

Within the OPTAIN project deliverable D3.2, a Google Earth Engine-based script was developed to predict crop types with the random forest method based on time series reflectance data of Sentinel 1A and 1B satellite radar images and the harmonised version of the Land Use / Cover Area frame statistical Survey (LUCAS) dataset (D'Andrimont et al., 2020). The script is accessible on ZENODO. It was provided with an R script to merge the local land use map with the derived crop maps. Detailed information is given in the report by Szabó et al. (2022). Required inputs are:

- a table which shows how to link the merged land use map categories (defined based on LUCAS documentation and CORINE terminology) with the SWAT+ crop type codes,
- the crop map derived in Google Earth Engine Platform with the crop classification script,
- the land use map of the catchment,
- the field boundary map of the catchment (see also section 2.2; field boundaries may change from year to year; it is therefore sufficient to use the specific boundaries of one recent year; if field boundaries are not available as shapefile, users need to delineate them manually based on a recent satellite imagery).

Output of the Google Earth Engine-based script is a series of crop maps from the year 2015 to 2021. Preliminary testing of this tool in several OPTAIN case studies showed that relying solely on LUCAS training data may produce inaccurate results. In this case it is recommended to add additional local training data. If this is not possible, manual correction of the script-based map is suggested.

In case you received very specific crop type information from a local dataset, it is advisable to reclassify them into a manageable number of crop types (especially 'rare' crop types with a low percentage (e.g. <5%) on total area should be assigned to a similar 'main' crop type). As an example, the original crop dataset for the German OPTAIN case study included 79 crop types which were reclassified into nine 'main' crop types. Spelt and durum wheat had very low percentages in this case and were thus classified as winter wheat.

To feed the field-level crop data into SWAT+, each case study is requested to include crop and, if appropriate, certain management information in the land use map. Each year, for which data could be obtained - from remote sensing or local datasets, must have its own column in the attribute table of the land use shapefile, specifying the crop name for the respective field polygons. As field boundaries may change from year to year but our field polygons are assumed to be static within the baseline period, it is recommended to assign the majority crop to each field polygon.

Moreover, the sequence of retrieved crop data for a certain period (e.g. 2015 to 2020) must be extrapolated (i.e. repeating the available sequence) to cover (at least!) the period from 1988 to 2020. If data for 2021 are available, they should remain in the attribute table. This allows for a total simulation period of 33 years where at least a part of this period contains the 'true' crop information in historic simulations (under observed climate) which can be used for model calibration, validation, and for defining the status-quo measure effectiveness. It is important to derive a 33-year crop sequence because this sequence can be directly used in later climate scenario simulations, which usually cover a period of 30 years (plus 3 years warm-up).

Figure 4.1 shows an exemplary scheme for such an extrapolation. Starting point in both directions, back in history until year 1988 and forwards to year 2020 (optional 2021), should always be the period for which the data were retrieved. Gaps within this period should be filled and any implausibilities (e.g. due to errors in the classification, such as forest or pasture for certain years in the sequence) should be corrected with plausible crops before extrapolation. Table 4.1 might help to identify suitable crops for filling gaps or correcting errors in the crop classification.

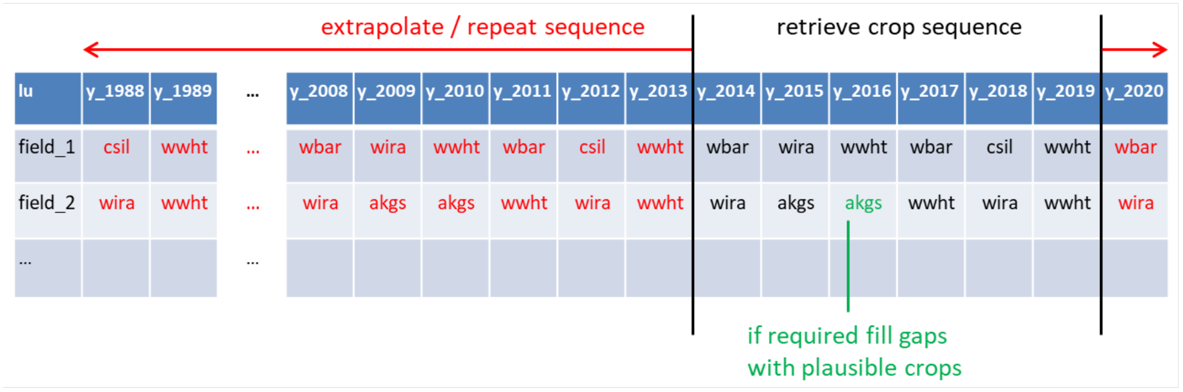
Table 4.1: Suitability of crop combinations (source: (Padel, 2001), modified).

Following crop	wwht	wbar	barl	wiry	oats	csil	fpea	alfa	akgs	pota	sgbt	wira
Winter wheat (wwht)	-	–	-	o	o	o	++	o	o	++	o	o
Spring wheat (swht)	–	–	-	o	o	++	+	++	++	++	++	++
Winter barley (wbar)	o	–	–	o	o	–	++	o	o	–	–	–
Spring barley (barl)	o	–	o	o	o	++	-	–	o	+	++	++
Winter rye (wiry*)	o	o	o	o	o	o	++	o	o	o	-	-
Oats (oats)	o	o	o	o	-	++	++	++	++	++	++	++
Corn (csil)	++	++	++	++	++	-	++	o	o	++	++	++
Alfalfa (alfa)	+	o	++	++	o	o	–	o	–	++	++	++
Farml. grass (akgs*)	o	o	++	++	++	o	o	o	o	++	++	++
Peas (fpea)	++	+	++	++	++	++	–	-	+	++	++	++
Potatoes (pota)	++	+	++	++	++	++	++	++	++	-	++	++
Beets (sgbt)	++	++	++	++	++	++	++	++	++	++	–	–
Rapeseed (wira*)	++	++	++	++	++	++	++	++	++	++	–	–

++ Good;
+ Good, but unnecessary. Other crops make better use of the preceding one. Could be used in combination with catch crop or green manure;
o Possible;
- Limited applications – not advisable if preceding crop is harvested late, in dry areas, if pest risk exists (mainly nematodes), or if danger of lodging (e.g. spring barley after legumes);
– inadvisable;
* crop customized for German case study and added to SWAT+ plant database.

Repeating the available crop sequences to generate the full 33-year sequences can also cause implausible or inadvisable crop combinations (e.g. winter rape following winter rape due to repeating a sequence starting and ending with winter rape). However, such 'rotation errors' can be ignored as they have no (or negligible) effects on the simulated yields. For OPTAIN it is important that the period of available data is represented with the correct crops per year. For the full 33-year sequences it is only important that overall crop shares are representative for the respective fields.

Each year column must start with the prefix *'y_'*, followed by the year. For writing agricultural management schedules via the `SWATfarmR` R package (section 4.3), it is also necessary that the attribute table of the land use map contains column *'lu'* with unique names (or codes) for each polygon representing a unique crop sequence or management (e.g., *'field_1'*, *'field_2'*, etc.). It is mandatory that



lu	y_1988	y_1989	...	y_2008	y_2009	y_2010	y_2011	y_2012	y_2013	y_2014	y_2015	y_2016	y_2017	y_2018	y_2019	y_2020
field_1	csil	wwht	...	wbar	wira	wwht	wbar	csil	wwht	wbar	wira	wwht	wbar	csil	wwht	wbar
field_2	wira	wwht	...	wira	akgs	akgs	wwht	wira	wwht	wira	akgs	akgs	wwht	wira	wwht	wira
...			...													

Figure 4.1: Example scheme for generating the 33-year period from 1988 to 2020, based on the period of retrieved crop information.

the names for all land objects which include a crop rotation start with the same prefix (e.g. *'field'*). Names of each crop or crop-management combination in the year columns (see for example Figure 4.2) must equal the names provided in the management schedule tables described in section 4.2. Keep in mind that a certain field can be split into multiple parts to obtain placeholders for measure implementation in later scenario runs (section 2.2) or smaller and more compact polygons to avoid routing problems. All parts of the same field should have the same crop sequence and therefore the same name. Likewise, all land objects with a certain generic management should have the same name (e.g. *'pasture'* or *'meadow'* for permanent grassland with or without grazing, respectively; or *'meadow_2cuts'*, *'meadow_3cuts'*, *'meadow_4cuts'*, etc. if you can differentiate certain intensity types of meadows).

4.1.2 Workflow summary

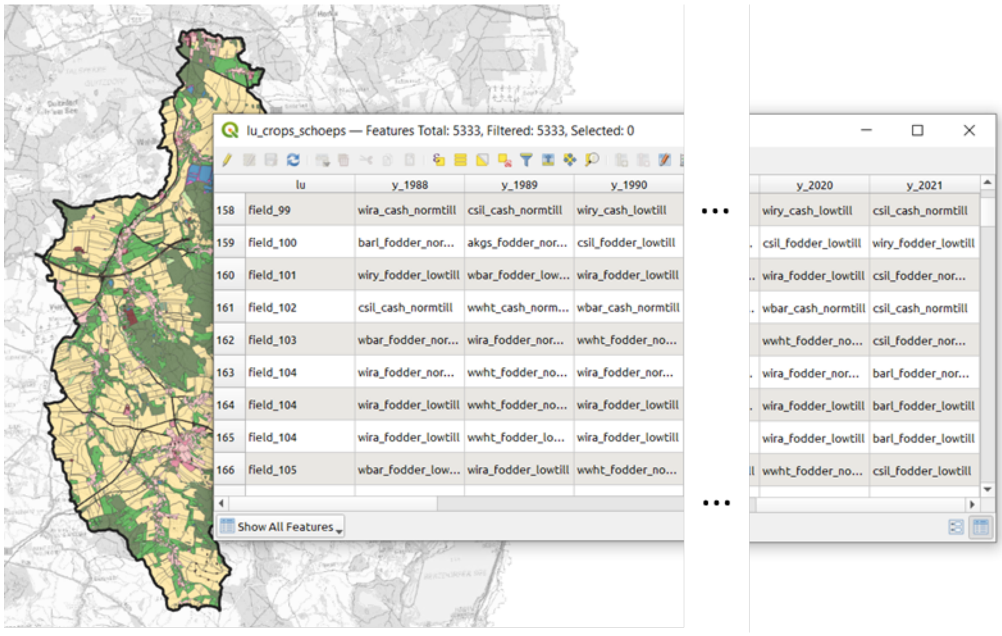
1. Retrieve a crop sequence for every field and a recent time period (minimum 5 years) from local datasets or remote-sensing based crop classification.
2. Check for implausible crops or land covers in the sequence, correct if necessary.
3. Fill gaps with plausible crops.
4. Extrapolate to cover the period from 1988 to 2020 (or 2021, if data are available).
5. Include the full crop sequence in your land-use map attribute table.
6. Meet conventions for naming columns, land objects and crop(-management) types.

4.2 Crop management schedules and crop rotations

4.2.1 SWAT+ management operations

The inputs for management operations are found in the *'management.sch'* file. SWAT+ can simulate different types of management operations. The type of operation simulated is identified by the code given for the MGT_OP/OP_TYP variable. The operations must be listed in chronological order starting in January for each year of rotation for management operations. The current list of operations is as follows:

- *plnt* - plant
- *harv* - harvest only


lu_crops_schoeps — Features Total: 5333, Filtered: 5333, Selected: 0

	lu	y_1988	y_1989	y_1990	…	y_2020	y_2021
158	field_99	wira_cash_normtill	csil_cash_normtill	wiry_cash_lowtill	…	wiry_cash_lowtill	csil_cash_normtill
159	field_100	barl_fodder_nor...	akgs_fodder_nor...	csil_fodder_lowtill		csil_fodder_lowtill	wiry_fodder_lowtill
160	field_101	wiry_fodder_lowtill	wbar_fodder_low...	wira_fodder_lowtill		wira_fodder_lowtill	csil_fodder_nor...
161	field_102	csil_cash_normtill	wwht_cash_norm...	wbar_cash_normtill		wbar_cash_normtill	csil_cash_normtill
162	field_103	wbar_fodder_nor...	wira_fodder_nor...	wwht_fodder_no...		wwht_fodder_no...	csil_fodder_nor...
163	field_104	wira_fodder_nor...	wwht_fodder_no...	wira_fodder_nor...		wira_fodder_nor...	barl_fodder_nor...
164	field_104	wira_fodder_lowtill	wwht_fodder_no...	wira_fodder_lowtill		wira_fodder_lowtill	barl_fodder_lowtill
165	field_104	wira_fodder_lowtill	wwht_fodder_lo...	wira_fodder_lowtill		wira_fodder_lowtill	barl_fodder_lowtill
166	field_105	wbar_fodder_low...	wira_fodder_lowtill	wwht_fodder_no...	…	wwht_fodder_no...	csil_fodder_lowtill

Figure 4.2: Example attribute table of a land-use map containing 34-year crop sequence from 1988 to 2021. The only convention on crop names is that they must equal the names provided in the management schedule table (section 4.2). Crop names can be also crop-management combinations (e.g. *'wwht_fodder_normtill'* referring to a winter wheat management under conventional tillage (in contrast to lowtill) to produce animal fodder, which implies applying some organic fertiliser, in contrast to cash crop production). Multiple land objects of one and the same field (here *'field_104'*) should have the same crop information.

- *kill* - kill
- *hvkl* - harvest and kill
- *till* - tillage
- *irr* - irrigation
- *fert* - fertiliser application
- *pest* - pesticide application
- *graz* - grazing
- *burn* - burn
- *swep* - street sweep
- *skip* - skip to end of the year.

The table below explains some of the operations and their parameters.

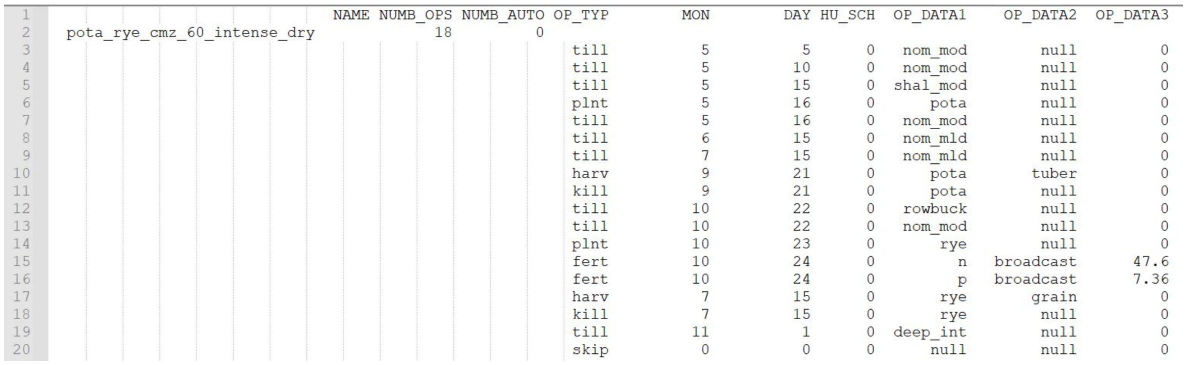
OP_TYP	OP_DATA1	OP_DATA2	OP_DATA3	Link to file	Explanation
fert	Fertiliser type from fert data base, i.e. “n”			*fertlizer.frt*	Fertiliser application
		surface application fraction from chem_app data base	amount applied in kg/ha	*chem_app.ops*	
plnt	Type of plant, i.e. *pota*			*plant.ini*	Plant one plant or entire community
harv	Type of plant, i.e. *pota*			*plant.ini*	Harvest only operation
		harvest specific type.		*harv.ops*	Plant part to be harvested: *biomass, grain, residue, tree, tuber, peanuts, stripper, picker* [1]
kill	Type of plant, i.e. *pota*			*plant.ini*	kill operation
hvkl	Type of plant, i.e. *pota*			*plant.ini*	harvest and kill operation
		harvest specific part of plant		*harv.ops*	Plant part to be harvested: *biomass, grain, tuber*
till	Type of plough			*tillage.til*	Tillage operation
irrm	irrigation amount (mm) from *irr.ops* data base			*irr.ops*	Date scheduled irrigation operation

[1]Most of the harvested parts are not supported (as of Nov-2022) and are placeholders for future development. Functioning harvest operations are: *biomass*, *grain*, *tuber*.

OP_TYP	OP_DATA1	OP_DATA2	OP_DATA3	Link to file	Explanation
pest	Sequential pesticide type from pest data base			*pesticide.pst*	pesticide application operation
		surface application option from *chem_app* data base	amount applied in kg/ha	*chem_app.ops*	
graz	Type of grazing operation		Amount of days that the operation occurred	*graze.ops*	grazing operation
burn	Burn type from *fire* data base			*fire.ops*	Burning of biomass
swep	Type of sweep operation			*sweep.ops*	street sweeping (only if iurban=2)
skip					skip a year

The management can be set to use Decision Tables or a fixed schedule.

Below in Figure 4.3 is a sample of *'management.sch'* file where all the operations are scheduled by fixed dates.

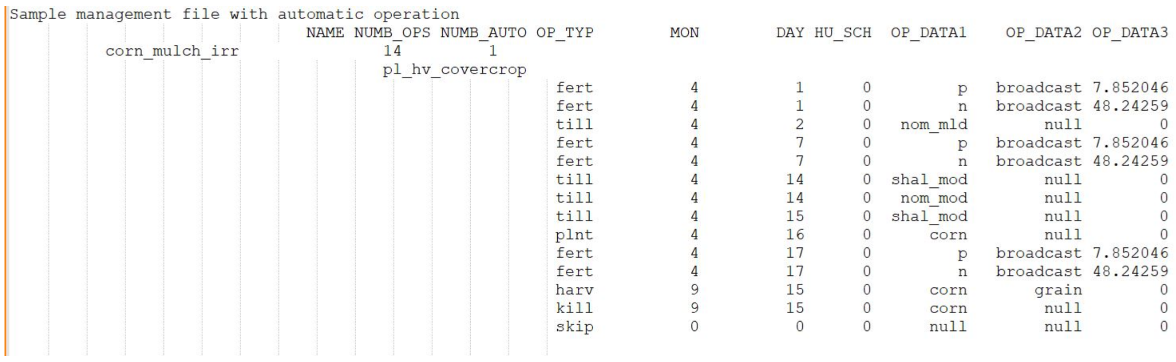


```
                           NAME NUMB_OPS NUMB_AUTO OP_TYP       MON        DAY HU_SCH  OP_DATA1     OP_DATA2  OP_DATA3
 pota_rye_cmz_60_intense_dry           18         0
                                                    till          5          5      0   nom_mod         null         0
                                                    till          5         10      0   nom_mod         null         0
                                                    till          5         15      0  shal_mod         null         0
                                                    plnt          5         16      0      pota         null         0
                                                    till          5         16      0   nom_mod         null         0
                                                    till          6         15      0   nom_mld         null         0
                                                    till          7         15      0   nom_mld         null         0
                                                    harv          9         21      0      pota        tuber         0
                                                    kill          9         21      0      pota         null         0
                                                    till         10         22      0   rowbuck         null         0
                                                    till         10         22      0   nom_mod         null         0
                                                    plnt         10         23      0       rye         null         0
                                                    fert         10         24      0         n    broadcast      47.6
                                                    fert         10         24      0         p    broadcast      7.36
                                                    harv          7         15      0       rye        grain         0
                                                    kill          7         15      0       rye         null         0
                                                    till         11          1      0  deep_int         null         0
                                                    skip          0          0      0      null         null         0
```

Figure 4.3: Snippet of the *'management.sch'* file

Automatic management operations can be defined using Decision Tables. The number of automatic operations is defined under the NUMB_AUTO column. Next line lists the DTs that are going to be in use for that management schedule. An example of a management schedule with an automatic procedure is shown below (Figure 4.4). See the NUMB_AUTO column and the name of the DT on the next line.

The example shows simulation of a corn crop using fixed management with a rye cover crop planted as an automatic procedure (see the DT chapter for more details). More examples of fixed management can be found at the SWAT+ Example datasets, i.e. fixed management for several crops in the US.



```
Sample management file with automatic operation
                                         NAME NUMB_OPS NUMB_AUTO OP_TYP        MON        DAY HU_SCH  OP_DATA1    OP_DATA2 OP_DATA3
        corn_mulch_irr                         14          1
                                               pl_hv_covercrop
                                                             fert             4          1      0         p   broadcast 7.852046
                                                             fert             4          1      0         n   broadcast 48.24259
                                                             till             4          2      0   nom_mld        null        0
                                                             fert             4          7      0         p   broadcast 7.852046
                                                             fert             4          7      0         n   broadcast 48.24259
                                                             till             4         14      0  shal_mod        null        0
                                                             till             4         14      0   nom_mod        null        0
                                                             till             4         15      0  shal_mod        null        0
                                                             plnt             4         16      0      corn        null        0
                                                             fert             4         17      0         p   broadcast 7.852046
                                                             fert             4         17      0         n   broadcast 48.24259
                                                             harv             9         15      0      corn       grain        0
                                                             kill             9         15      0      corn        null        0
                                                             skip             0          0      0      null        null        0
```

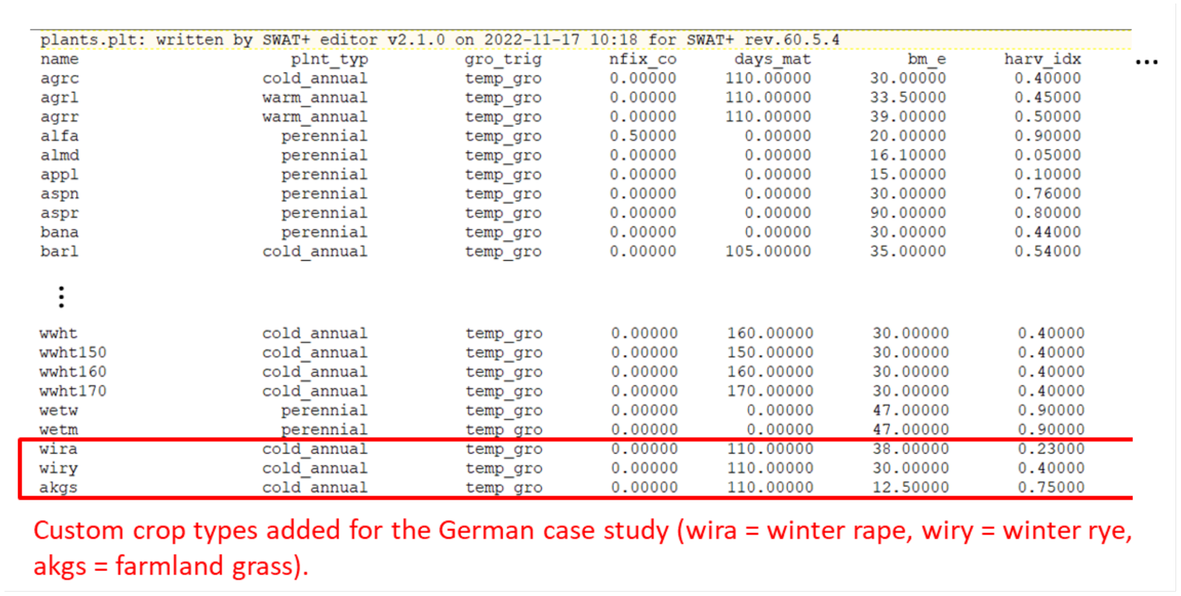
Figure 4.4: Snippet of the *'management.sch'* file with a decision table

4.2.2 Databases referring to agricultural management

The management operations described above often refer to specific crop, fertiliser, or tillage types which are described by specific parameters. These parameters are listed in respective databases which can (and should) be edited by demand.

4.2.2.1 Plant database

The plant database (*'plants.plt'*) is described in section 3.3. It contains about 260 land cover and plant types (or varieties). However, only a few of them might be suitable to represent European crop types. The user has to ensure that the database contains all crops that are included in the land-use/crop map described in the crop data section). If a certain crop type is not listed in the database, users can add a new crop type (see for example Figure 4.5) and define all relevant parameters (if available) or simply choose a different but similar crop type listed in the database (or solve the issue via reclassification as described in the previous section).



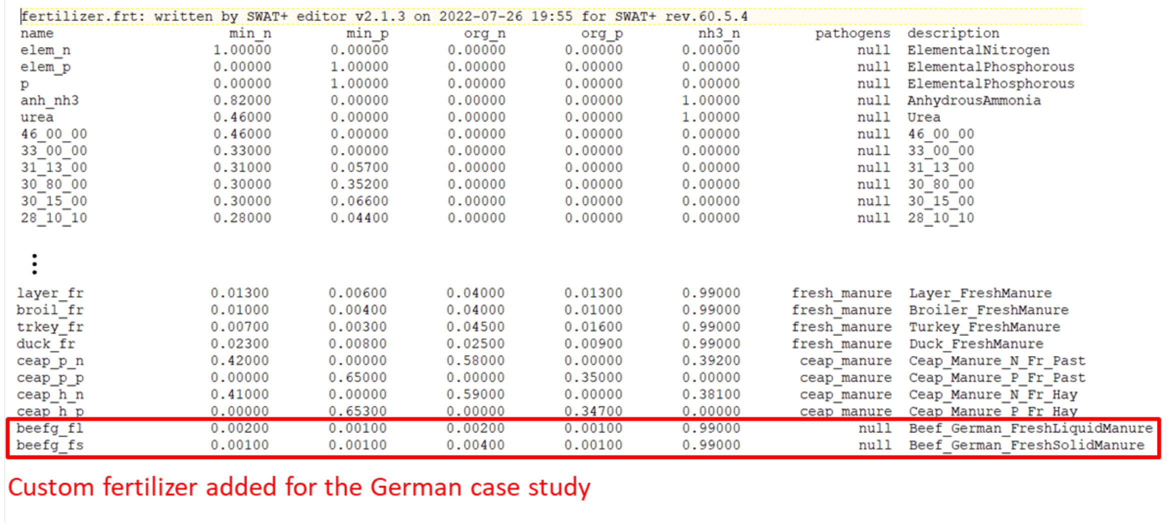
```
plants.plt: written by SWAT+ editor v2.1.0 on 2022-11-17 10:18 for SWAT+ rev.60.5.4
name                      plnt_typ           gro_trig      nfix_co      days_mat         bm_e         harv_idx    ...
agrc                   cold_annual           temp_gro      0.00000     110.00000      30.00000       0.40000
agrl                   warm_annual           temp_gro      0.00000     110.00000      33.50000       0.45000
agrr                   warm_annual           temp_gro      0.00000     110.00000      39.00000       0.50000
alfa                     perennial           temp_gro      0.50000       0.00000      20.00000       0.90000
almd                     perennial           temp_gro      0.00000       0.00000      16.10000       0.05000
appl                     perennial           temp_gro      0.00000       0.00000      15.00000       0.10000
aspn                     perennial           temp_gro      0.00000       0.00000      30.00000       0.76000
aspr                     perennial           temp_gro      0.00000       0.00000      90.00000       0.80000
bana                     perennial           temp_gro      0.00000       0.00000      30.00000       0.44000
barl                   cold_annual           temp_gro      0.00000     105.00000      35.00000       0.54000

  ⋮

wwht                   cold_annual           temp_gro      0.00000     160.00000      30.00000       0.40000
wwht150                cold_annual           temp_gro      0.00000     150.00000      30.00000       0.40000
wwht160                cold_annual           temp_gro      0.00000     160.00000      30.00000       0.40000
wwht170                cold_annual           temp_gro      0.00000     170.00000      30.00000       0.40000
wetw                     perennial           temp_gro      0.00000       0.00000      47.00000       0.90000
wetm                     perennial           temp_gro      0.00000       0.00000      47.00000       0.90000
wira                   cold_annual           temp_gro      0.00000     110.00000      38.00000       0.23000
wiry                   cold_annual           temp_gro      0.00000     110.00000      30.00000       0.40000
akgs                   cold_annual           temp_gro      0.00000     110.00000      12.50000       0.75000
```

Custom crop types added for the German case study (wira = winter rape, wiry = winter rye, akgs = farmland grass).

Figure 4.5: *'plants.plt'* file of the German case study.

4.2.2.2 Fertiliser database

The fertiliser database (*'fertilizer.frt'*) contains 59 types of fertiliser, ranging from one-component mineral fertilisers for N and P and different mixtures to different types of organic fertilisers. The fertilisers differ in the fraction of mineral and organic N and P components. Users need to define the appropriate fertilizer type in each fertiliser operation. If an appropriate type of fertiliser is missing, users can add their own case-study specific fertiliser by simply copying and modifying an existing entry (see for example Figure 4.6).



```
fertilizer.frt: written by SWAT+ editor v2.1.3 on 2022-07-26 19:55 for SWAT+ rev.60.5.4
name                 min_n      min_p      org_n      org_p      nh3_n     pathogens  description
elem_n             1.00000    0.00000    0.00000    0.00000    0.00000          null  ElementalNitrogen
elem_p             0.00000    1.00000    0.00000    0.00000    0.00000          null  ElementalPhosphorous
p                  0.00000    1.00000    0.00000    0.00000    0.00000          null  ElementalPhosphorous
anh_nh3            0.82000    0.00000    0.00000    0.00000    1.00000          null  AnhydrousAmmonia
urea               0.46000    0.00000    0.00000    0.00000    1.00000          null  Urea
46_00_00           0.46000    0.00000    0.00000    0.00000    0.00000          null  46_00_00
33_00_00           0.33000    0.00000    0.00000    0.00000    0.00000          null  33_00_00
31_13_00           0.31000    0.05700    0.00000    0.00000    0.00000          null  31_13_00
30_80_00           0.30000    0.35200    0.00000    0.00000    0.00000          null  30_80_00
30_15_00           0.30000    0.06600    0.00000    0.00000    0.00000          null  30_15_00
28_10_10           0.28000    0.04400    0.00000    0.00000    0.00000          null  28_10_10

 ⋮

layer_fr           0.01300    0.00600    0.04000    0.01300    0.99000  fresh_manure  Layer_FreshManure
broil_fr           0.01000    0.00400    0.04000    0.01000    0.99000  fresh_manure  Broiler_FreshManure
trkey_fr           0.00700    0.00300    0.04500    0.01600    0.99000  fresh_manure  Turkey_FreshManure
duck_fr            0.02300    0.00800    0.02500    0.00900    0.99000  fresh_manure  Duck_FreshManure
ceap_p_n           0.42000    0.00000    0.58000    0.00000    0.39200   ceap_manure  Ceap_Manure_N_Fr_Past
ceap_p_p           0.00000    0.65000    0.00000    0.35000    0.00000   ceap_manure  Ceap_Manure_P_Fr_Past
ceap_h_n           0.41000    0.00000    0.59000    0.00000    0.38100   ceap_manure  Ceap_Manure_N_Fr_Hay
ceap_h_p           0.00000    0.65300    0.00000    0.34700    0.00000   ceap_manure  Ceap_Manure_P_Fr_Hay
beefg_fl           0.00200    0.00100    0.00200    0.00100    0.99000          null  Beef_German_FreshLiquidManure
beefg_fs           0.00100    0.00100    0.00400    0.00100    0.99000          null  Beef_German_FreshSolidManure
```

Figure 4.6: *'fertilizer.frt'* file of the German case study.

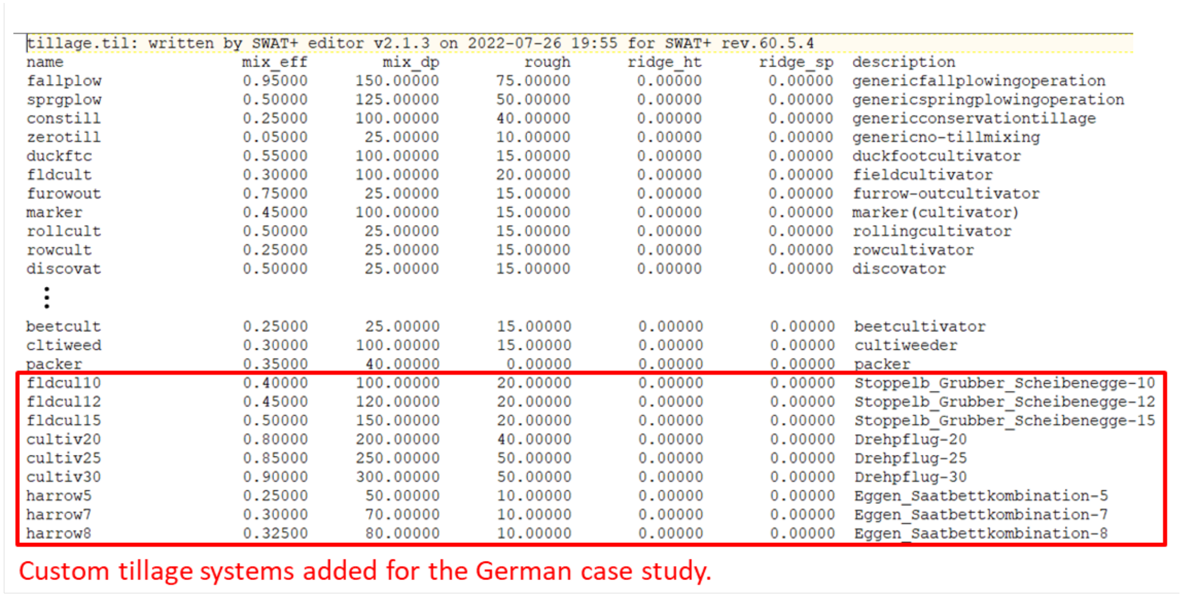
4.2.2.3 Tillage database

The tillage database (*'tillage.til'*) contains 78 types of tillage systems, each differing in their mixing efficiency and mixing depth as well as the random roughness of the tilled soil. Users need to define the appropriate tillage system type in each tillage operation. If an appropriate type is missing, users can add their own case-study specific tillage system by simply copying and modifying an existing entry (see for example Figure 4.7).

4.2.3 Crop-specific management operation table in OPTAIN

In OPTAIN, we have to schedule management operations for each field representing the (unique) crop sequence of that field as derived from local data or the remote-sensing based classification (section 2.3.2.2). This requires generating hundreds or thousands of different management operation schedules (depending on the number of fields) which can be daunting. To automate this process, we recommend to provide representative management operation schedules for each relevant crop in a certain format. Using an R script (described in section 4.3, these single-year crop schedules will be combined to field specific rotation schedules and stored in a huge *.csv* file that can be directly used by the `SWATfarmR` R package to write the SWAT+ *'management.sch'* file.

For all crops present in the land-use crop map, it is required to provide a representative schedule of typical management operations. For this task, it is recommended to ask local experts from the stakeholder group (e.g. farm advisors) for assistance and confirmation that the defined operations



```
tillage.til: written by SWAT+ editor v2.1.3 on 2022-07-26 19:55 for SWAT+ rev.60.5.4
name                   mix_eff      mix_dp       rough    ridge_ht    ridge_sp  description
fallplow               0.95000   150.00000    75.00000     0.00000     0.00000  genericfallplowingoperation
sprgplow               0.50000   125.00000    50.00000     0.00000     0.00000  genericspringplowingoperation
constill               0.25000   100.00000    40.00000     0.00000     0.00000  genericconservationtillage
zerotill               0.05000    25.00000    10.00000     0.00000     0.00000  genericno-tillmixing
duckftc                0.55000   100.00000    15.00000     0.00000     0.00000  duckfootcultivator
fldcult                0.30000   100.00000    20.00000     0.00000     0.00000  fieldcultivator
furowout               0.75000    25.00000    15.00000     0.00000     0.00000  furrow-outcultivator
marker                 0.45000   100.00000    15.00000     0.00000     0.00000  marker(cultivator)
rollcult               0.50000    25.00000    15.00000     0.00000     0.00000  rollingcultivator
rowcult                0.25000    25.00000    15.00000     0.00000     0.00000  rowcultivator
discovat               0.50000    25.00000    15.00000     0.00000     0.00000  discovator
  ⋮
beetcult               0.25000    25.00000    15.00000     0.00000     0.00000  beetcultivator
cltiweed               0.30000   100.00000    15.00000     0.00000     0.00000  cultiweeder
packer                 0.35000    40.00000     0.00000     0.00000     0.00000  packer
fldcul10               0.40000   100.00000    20.00000     0.00000     0.00000  Stoppelb_Grubber_Scheibenegge-10
fldcul12               0.45000   120.00000    20.00000     0.00000     0.00000  Stoppelb_Grubber_Scheibenegge-12
fldcul15               0.50000   150.00000    20.00000     0.00000     0.00000  Stoppelb_Grubber_Scheibenegge-15
cultiv20               0.80000   200.00000    40.00000     0.00000     0.00000  Drehpflug-20
cultiv25               0.85000   250.00000    50.00000     0.00000     0.00000  Drehpflug-25
cultiv30               0.90000   300.00000    50.00000     0.00000     0.00000  Drehpflug-30
harrow5                0.25000    50.00000    10.00000     0.00000     0.00000  Eggen_Saatbettkombination-5
harrow7                0.30000    70.00000    10.00000     0.00000     0.00000  Eggen_Saatbettkombination-7
harrow8                0.32500    80.00000    10.00000     0.00000     0.00000  Eggen_Saatbettkombination-8
```

Custom tillage systems added for the German case study.

Figure 4.7: *'tillage.til'* file of the German case study.

are indeed typical for the case study. All of the individual crop-specific operation schedules must be compiled in one .csv as exemplarily shown in Figure 4.8. Here, the order of crops is not important.

Column *'crop_mgt'* lists the names of each crop type (+ management type if desired). The names must match the names used in the crop sequences of the land use crop map (section 4.1).

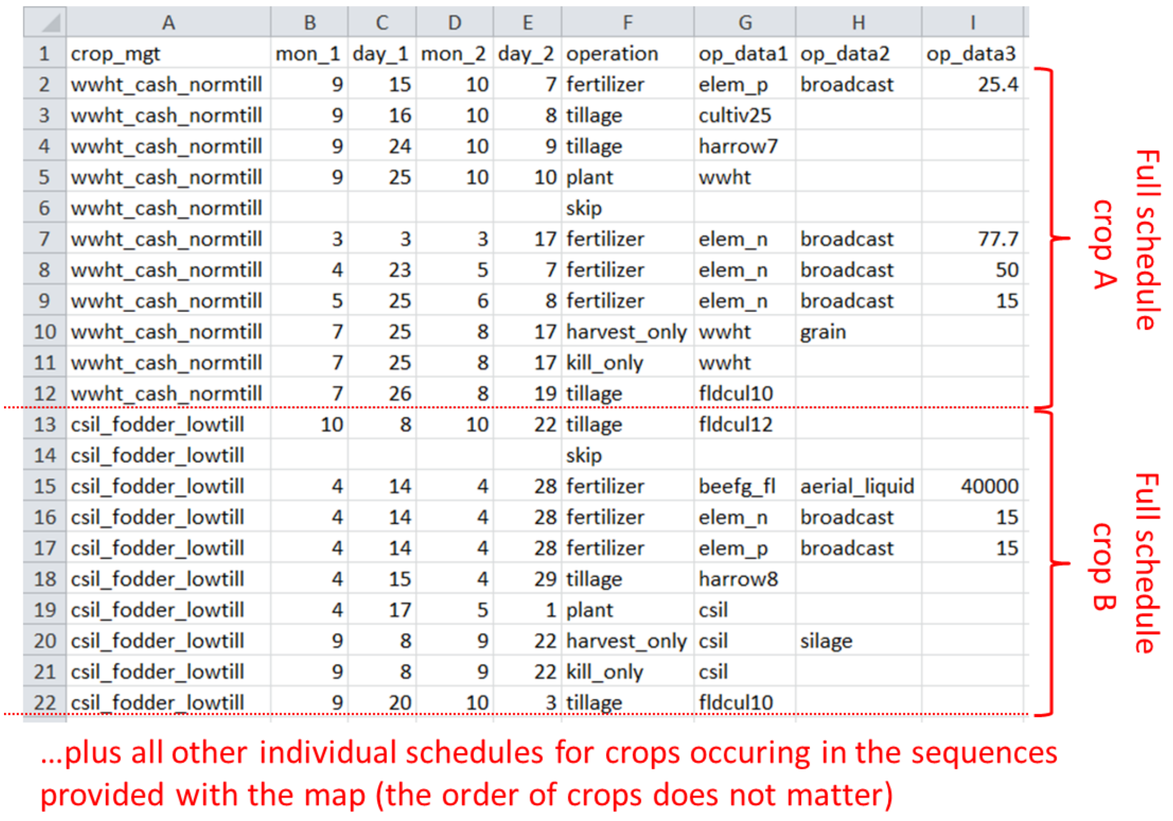
Columns *mon_1* and *day_1* define the start of a representative time window for a specific operation. *mon_2* and *day_2* define the end of this time window.

Column *operation* lists all relevant operations that should be considered for a certain crop. The operations need to be labelled in the format of SWATfarmR. *op_data1*, *op_data2*, and *op_data3* further specify the operations using the SWAT+ codes or numeric values (e.g. for the amount of applied fertiliser) as previously described in this section.

All operations within a single-year crop schedule must be sorted in chronological order, starting with the first crop-specific operation. Note, the first crop-specific operation is usually not the first operation occurring in a year. Usually, the first operation for a specific crop is a fertiliser or tillage operation in autumn, i.e. after the harvest of the preceding crop.

Obviously, it is mandatory to include a *plant* operation, otherwise there will be no crop growth. It is also mandatory to include a *kill* operation at the end of the schedule. Otherwise, subsequent crops in a rotation will not grow in the SWAT+ model simulations. For all crop schedules it is strongly recommended to not use the combined *harvest_kill* operation to allow later verification of plant maturity for the last harvest operation (section model verification. Therefore, it is mandatory to always use a *harvest_only* operation and if it is the final harvest operation for that crop it should be followed by a *kill* operation with the same time window . *kill* must be included, but not necessarily as the last operation of the schedule. This could be also a tillage operation, which is usually applied after harvesting a specific crop, independent from the type of the subsequent crop (see for example Figure 4.8).

Mandatory is also a line with a *skip* in column *operation* to indicate the turn of the year for each schedule. *skip* should never be in the last position. If it cannot be placed in-between other operations,



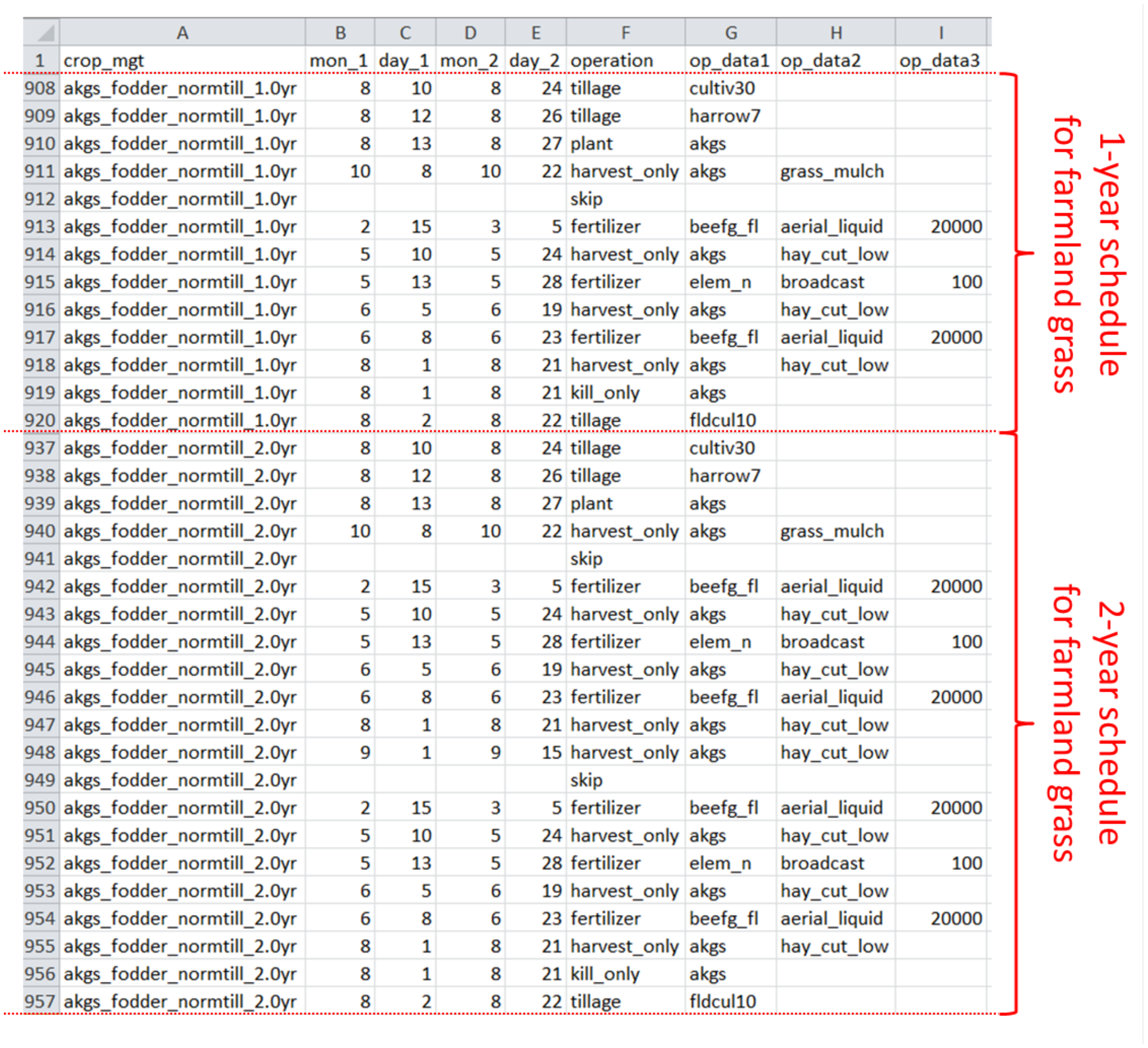
	A	B	C	D	E	F	G	H	I
1	crop_mgt	mon_1	day_1	mon_2	day_2	operation	op_data1	op_data2	op_data3
2	wwht_cash_normtill	9	15	10	7	fertilizer	elem_p	broadcast	25.4
3	wwht_cash_normtill	9	16	10	8	tillage	cultiv25		
4	wwht_cash_normtill	9	24	10	9	tillage	harrow7		
5	wwht_cash_normtill	9	25	10	10	plant	wwht		
6	wwht_cash_normtill					skip			
7	wwht_cash_normtill	3	3	3	17	fertilizer	elem_n	broadcast	77.7
8	wwht_cash_normtill	4	23	5	7	fertilizer	elem_n	broadcast	50
9	wwht_cash_normtill	5	25	6	8	fertilizer	elem_n	broadcast	15
10	wwht_cash_normtill	7	25	8	17	harvest_only	wwht	grain	
11	wwht_cash_normtill	7	25	8	17	kill_only	wwht		
12	wwht_cash_normtill	7	26	8	19	tillage	fldcul10		
13	csil_fodder_lowtill	10	8	10	22	tillage	fldcul12		
14	csil_fodder_lowtill					skip			
15	csil_fodder_lowtill	4	14	4	28	fertilizer	beefg_fl	aerial_liquid	40000
16	csil_fodder_lowtill	4	14	4	28	fertilizer	elem_n	broadcast	15
17	csil_fodder_lowtill	4	14	4	28	fertilizer	elem_p	broadcast	15
18	csil_fodder_lowtill	4	15	4	29	tillage	harrow8		
19	csil_fodder_lowtill	4	17	5	1	plant	csil		
20	csil_fodder_lowtill	9	8	9	22	harvest_only	csil	silage	
21	csil_fodder_lowtill	9	8	9	22	kill_only	csil		
22	csil_fodder_lowtill	9	20	10	3	tillage	fldcul10		

Full schedule crop A
Full schedule crop B
...plus all other individual schedules for crops occuring in the sequences provided with the map (the order of crops does not matter)

Figure 4.8: Snippet of an example management operation schedule file (*.csv*) in the format required to generate the SWATfarmR input table automatically (see next section). Shown here are examples for only two crops. The full file must contain schedules for all crops relevant in the case study.

it must be placed in the first line. Otherwise, the final rotation schedules written to the `SWATfarmR` input *.csv* file could include severe chronological errors.

The fact that *plant* and *kill* must be included in the schedule of a specific crop requires careful attention for crops with variable length of the growing period, such as farmland grass, which can be either killed already after one year of growing, or remain on the field for further two, three, or four years before it is killed. In such a case, multiple operation schedules have to be developed (one for each possible period length, see Figure 4.9).

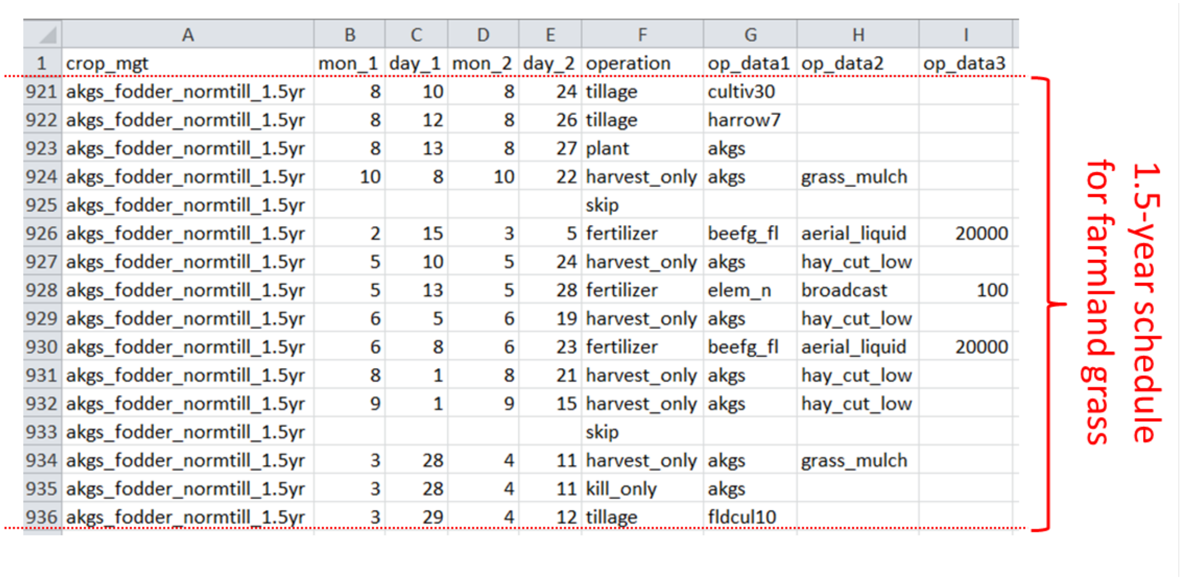

1-year schedule for farmland grass
2-year schedule for farmland grass

	A	B	C	D	E	F	G	H	I
1	crop_mgt	mon_1	day_1	mon_2	day_2	operation	op_data1	op_data2	op_data3
908	akgs_fodder_normtill_1.0yr	8	10	8	24	tillage	cultiv30		
909	akgs_fodder_normtill_1.0yr	8	12	8	26	tillage	harrow7		
910	akgs_fodder_normtill_1.0yr	8	13	8	27	plant	akgs		
911	akgs_fodder_normtill_1.0yr	10	8	10	22	harvest_only	akgs	grass_mulch	
912	akgs_fodder_normtill_1.0yr					skip			
913	akgs_fodder_normtill_1.0yr	2	15	3	5	fertilizer	beefg_fl	aerial_liquid	20000
914	akgs_fodder_normtill_1.0yr	5	10	5	24	harvest_only	akgs	hay_cut_low	
915	akgs_fodder_normtill_1.0yr	5	13	5	28	fertilizer	elem_n	broadcast	100
916	akgs_fodder_normtill_1.0yr	6	5	6	19	harvest_only	akgs	hay_cut_low	
917	akgs_fodder_normtill_1.0yr	6	8	6	23	fertilizer	beefg_fl	aerial_liquid	20000
918	akgs_fodder_normtill_1.0yr	8	1	8	21	harvest_only	akgs	hay_cut_low	
919	akgs_fodder_normtill_1.0yr	8	1	8	21	kill_only	akgs		
920	akgs_fodder_normtill_1.0yr	8	2	8	22	tillage	fldcul10		
937	akgs_fodder_normtill_2.0yr	8	10	8	24	tillage	cultiv30		
938	akgs_fodder_normtill_2.0yr	8	12	8	26	tillage	harrow7		
939	akgs_fodder_normtill_2.0yr	8	13	8	27	plant	akgs		
940	akgs_fodder_normtill_2.0yr	10	8	10	22	harvest_only	akgs	grass_mulch	
941	akgs_fodder_normtill_2.0yr					skip			
942	akgs_fodder_normtill_2.0yr	2	15	3	5	fertilizer	beefg_fl	aerial_liquid	20000
943	akgs_fodder_normtill_2.0yr	5	10	5	24	harvest_only	akgs	hay_cut_low	
944	akgs_fodder_normtill_2.0yr	5	13	5	28	fertilizer	elem_n	broadcast	100
945	akgs_fodder_normtill_2.0yr	6	5	6	19	harvest_only	akgs	hay_cut_low	
946	akgs_fodder_normtill_2.0yr	6	8	6	23	fertilizer	beefg_fl	aerial_liquid	20000
947	akgs_fodder_normtill_2.0yr	8	1	8	21	harvest_only	akgs	hay_cut_low	
948	akgs_fodder_normtill_2.0yr	9	1	9	15	harvest_only	akgs	hay_cut_low	
949	akgs_fodder_normtill_2.0yr					skip			
950	akgs_fodder_normtill_2.0yr	2	15	3	5	fertilizer	beefg_fl	aerial_liquid	20000
951	akgs_fodder_normtill_2.0yr	5	10	5	24	harvest_only	akgs	hay_cut_low	
952	akgs_fodder_normtill_2.0yr	5	13	5	28	fertilizer	elem_n	broadcast	100
953	akgs_fodder_normtill_2.0yr	6	5	6	19	harvest_only	akgs	hay_cut_low	
954	akgs_fodder_normtill_2.0yr	6	8	6	23	fertilizer	beefg_fl	aerial_liquid	20000
955	akgs_fodder_normtill_2.0yr	8	1	8	21	harvest_only	akgs	hay_cut_low	
956	akgs_fodder_normtill_2.0yr	8	1	8	21	kill_only	akgs		
957	akgs_fodder_normtill_2.0yr	8	2	8	22	tillage	fldcul10		

Figure 4.9: Examples for a multi-year management schedule (here, a 2-year farmland grass schedule next to a 1-year schedule) in the format required to generate the SWATfarmR input table automatically (see next section). Farmland grass has a variable growing period in the German case study.

The multi-year problem can be even more complicated because perennials such as farmland grass can either be killed in autumn (before the subsequent winter crop) or in spring (before the summer crop). If relevant in a case study, it is recommended to develop schedules also for such cases (see for example Figure 4.10).

Note, the suffixes indicating the length of the growing period (e.g. _1.5yr or _2yr) are only necessary

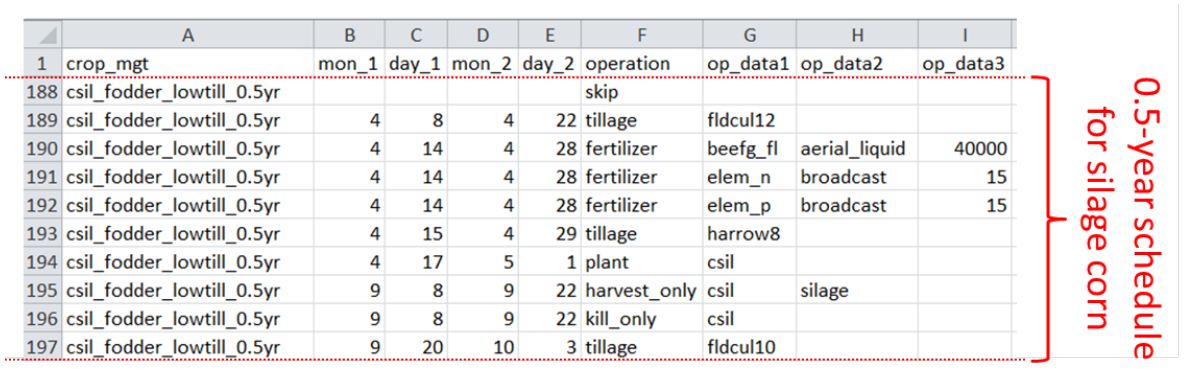


	A	B	C	D	E	F	G	H	I
1	crop_mgt	mon_1	day_1	mon_2	day_2	operation	op_data1	op_data2	op_data3
921	akgs_fodder_normtill_1.5yr	8	10	8	24	tillage	cultiv30		
922	akgs_fodder_normtill_1.5yr	8	12	8	26	tillage	harrow7		
923	akgs_fodder_normtill_1.5yr	8	13	8	27	plant	akgs		
924	akgs_fodder_normtill_1.5yr	10	8	10	22	harvest_only	akgs	grass_mulch	
925	akgs_fodder_normtill_1.5yr					skip			
926	akgs_fodder_normtill_1.5yr	2	15	3	5	fertilizer	beefg_fl	aerial_liquid	20000
927	akgs_fodder_normtill_1.5yr	5	10	5	24	harvest_only	akgs	hay_cut_low	
928	akgs_fodder_normtill_1.5yr	5	13	5	28	fertilizer	elem_n	broadcast	100
929	akgs_fodder_normtill_1.5yr	6	5	6	19	harvest_only	akgs	hay_cut_low	
930	akgs_fodder_normtill_1.5yr	6	8	6	23	fertilizer	beefg_fl	aerial_liquid	20000
931	akgs_fodder_normtill_1.5yr	8	1	8	21	harvest_only	akgs	hay_cut_low	
932	akgs_fodder_normtill_1.5yr	9	1	9	15	harvest_only	akgs	hay_cut_low	
933	akgs_fodder_normtill_1.5yr					skip			
934	akgs_fodder_normtill_1.5yr	3	28	4	11	harvest_only	akgs	grass_mulch	
935	akgs_fodder_normtill_1.5yr	3	28	4	11	kill_only	akgs		
936	akgs_fodder_normtill_1.5yr	3	29	4	12	tillage	fldcul10		

Figure 4.10: Examples for a 1.5-year farmland grass schedule which might be needed for the case that a summer crop follows within the rotation.

in the management schedule table but not in the land use crop map. The length of the growing period will be detected automatically when scanning the sequences obtained from the map.

Unfortunately, multi-year farmland grass can also cause problems in crop rotations if the schedule for a following summer crop already starts in autumn (often, a tillage operation is applied in autumn even if the summer crop is planted in spring). For such cases, it is necessary to add an additional schedule for all relevant summer crops without starting operations in autumn. The adapted summer crop schedules should be added to the crop management table with suffix '_0.5yr' (e.g. 'csil_0.5yr'). An example is shown in Figure 4.11). Here, the adapted schedule for silage corn (*csil*) is exactly the same as shown in Figure 4.8, with the exception that the tillage operation (tillage type *fldcul12*) has been moved to April. Adapted 'half-year' schedules thus begin always with a *skip* line.



	A	B	C	D	E	F	G	H	I
1	crop_mgt	mon_1	day_1	mon_2	day_2	operation	op_data1	op_data2	op_data3
188	csil_fodder_lowtill_0.5yr					skip			
189	csil_fodder_lowtill_0.5yr	4	8	4	22	tillage	fldcul12		
190	csil_fodder_lowtill_0.5yr	4	14	4	28	fertilizer	beefg_fl	aerial_liquid	40000
191	csil_fodder_lowtill_0.5yr	4	14	4	28	fertilizer	elem_n	broadcast	15
192	csil_fodder_lowtill_0.5yr	4	14	4	28	fertilizer	elem_p	broadcast	15
193	csil_fodder_lowtill_0.5yr	4	15	4	29	tillage	harrow8		
194	csil_fodder_lowtill_0.5yr	4	17	5	1	plant	csil		
195	csil_fodder_lowtill_0.5yr	9	8	9	22	harvest_only	csil	silage	
196	csil_fodder_lowtill_0.5yr	9	8	9	22	kill_only	csil		
197	csil_fodder_lowtill_0.5yr	9	20	10	3	tillage	fldcul10		

Figure 4.11: Examples for a 0.5-year summer crop schedule.

Management operation schedules must be also developed for winter cover crops if they are relevant in a case study (or if they will be relevant as a retention measure). Unfortunately, cover crops cannot be detected automatically in the crop sequences provided in the land use map, because they grow in between the periods of two main crops. This (important) issue still needs to be addressed in future developments of our scrip-based solution to generate SWAT+ management schedules.

4.2.4 Management operation table for generic land-use in OPTAIN

Generic land-use classes without crop rotations can also include management operations. Pastures and meadows, for example, should always be simulated with typical management operations (e.g. grazing, hay cutting, fertilisation). Even if there is no management to be defined for a certain land-use class (such as forest), plant growth must be initialized (section 3.3). Since the `SWATfarmR` R package will also write the plant.ini file, we need to define here the initial plant growth parameters. Similarly to the crop management table, a generic land-use management table needs to be compiled for all non-cropland classes which include a vegetation cover (all classes except urban and water, and possibly barren land). An example file is shown in Figure 4.12).

In contrast to the crop management table, *skip* lines are not required and not allowed. Instead, the user has to define initial plant growth with label *initial_plant* in column *'operation'*, the respective plant community in column *'op_data1'* and the parameter values needed for plant growth initialization, all listed as a vector in column *'op_data2'* (separated by comma). The values refer to parameters *lai_init*, *bm_init*, *phu_init*, *plnt_pop*, *yrs_init*, *rsd_init* in the order of occurrence. More details on these parameters are provided in the SWAT+ land-use-management documentation.

4.2.5 Workflow summary

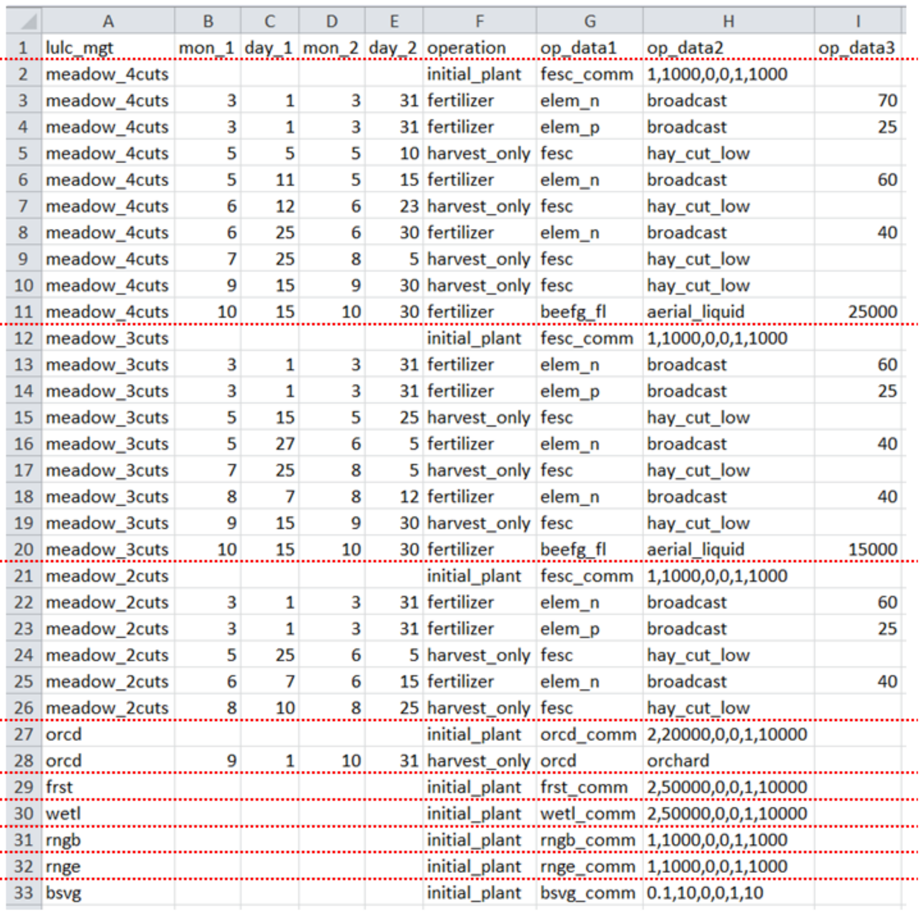
1. Make yourself familiar with the agricultural management operations supported by SWAT+.
2. Make yourself familiar with the plant, tillage, and fertiliser databases. Add custom entries in case you need it.
3. Develop representative management operation schedules for each crop (or crop-management) type that is listed in the land-use crop map and compile them in a crop management table. Follow the conventions for specifying operations.
4. Develop representative generic management operation schedules (e.g. for pasture or meadows) and initialise plant growth for all non-crop land-use classes. Follow the naming conventions and compile all non-cropland classes in a generic land-use management table.

4.3 Development of management schedules with SWATfarmR

With the land-use crop map and the management operations file, it is now possible to generate the management input file to run the `SWATfarmR` R package. Since each agricultural field might have its own crop rotation and the number of fields might range from several hundreds to thousands, it is recommended to use `R` to generate the `SWATfarmR` input file.

The following R code will generate the `SWATfarmR` management input file directly from (i) the sequence of crops provided in the land-use crop map and (ii) the crop management file containing the corresponding individual crop management schedules. The script will not only combine the individual crop-specific management schedules to full rotation schedules for a desired simulation period, it will also check for any date conflicts among operations (as conflicts might occur among the last and first operations of consecutive crops) and automatically adjust the conflicting dates (if desired, otherwise the individual crop schedules must be adjusted manually). The script will also translate the settings in the generic land-use management file into the format of `SWATfarmR`.

The `SWATfarmR_input` script version consists of an RStudio project (`SWATfarmR_input.Rproj`) with 2 `R` script files and an input data folder ('input_data') containing demo data. For the moment, the script



	A	B	C	D	E	F	G	H	I
1	lulc_mgt	mon_1	day_1	mon_2	day_2	operation	op_data1	op_data2	op_data3
2	meadow_4cuts					initial_plant	fesc_comm	1,1000,0,0,1,1000	
3	meadow_4cuts	3	1	3	31	fertilizer	elem_n	broadcast	70
4	meadow_4cuts	3	1	3	31	fertilizer	elem_p	broadcast	25
5	meadow_4cuts	5	5	5	10	harvest_only	fesc	hay_cut_low	
6	meadow_4cuts	5	11	5	15	fertilizer	elem_n	broadcast	60
7	meadow_4cuts	6	12	6	23	harvest_only	fesc	hay_cut_low	
8	meadow_4cuts	6	25	6	30	fertilizer	elem_n	broadcast	40
9	meadow_4cuts	7	25	8	5	harvest_only	fesc	hay_cut_low	
10	meadow_4cuts	9	15	9	30	harvest_only	fesc	hay_cut_low	
11	meadow_4cuts	10	15	10	30	fertilizer	beefg_fl	aerial_liquid	25000
12	meadow_3cuts					initial_plant	fesc_comm	1,1000,0,0,1,1000	
13	meadow_3cuts	3	1	3	31	fertilizer	elem_n	broadcast	60
14	meadow_3cuts	3	1	3	31	fertilizer	elem_p	broadcast	25
15	meadow_3cuts	5	15	5	25	harvest_only	fesc	hay_cut_low	
16	meadow_3cuts	5	27	6	5	fertilizer	elem_n	broadcast	40
17	meadow_3cuts	7	25	8	5	harvest_only	fesc	hay_cut_low	
18	meadow_3cuts	8	7	8	12	fertilizer	elem_n	broadcast	40
19	meadow_3cuts	9	15	9	30	harvest_only	fesc	hay_cut_low	
20	meadow_3cuts	10	15	10	30	fertilizer	beefg_fl	aerial_liquid	15000
21	meadow_2cuts					initial_plant	fesc_comm	1,1000,0,0,1,1000	
22	meadow_2cuts	3	1	3	31	fertilizer	elem_n	broadcast	60
23	meadow_2cuts	3	1	3	31	fertilizer	elem_p	broadcast	25
24	meadow_2cuts	5	25	6	5	harvest_only	fesc	hay_cut_low	
25	meadow_2cuts	6	7	6	15	fertilizer	elem_n	broadcast	40
26	meadow_2cuts	8	10	8	25	harvest_only	fesc	hay_cut_low	
27	orcd					initial_plant	orcd_comm	2,20000,0,0,1,10000	
28	orcd	9	1	10	31	harvest_only	orcd	orchard	
29	frst					initial_plant	frst_comm	2,50000,0,0,1,10000	
30	wetl					initial_plant	wetl_comm	2,50000,0,0,1,10000	
31	rngb					initial_plant	rngb_comm	1,1000,0,0,1,1000	
32	rnge					initial_plant	rnge_comm	1,1000,0,0,1,1000	
33	bsvg					initial_plant	bsvg_comm	0.1,10,0,0,1,10	

Figure 4.12: Examples for a generic land-use management file. Columns are similar to the crop management table. However, here initial plant growth has to be specified by providing the name of the plant community and the parameter values needed for plant growth initialization. A *skip* line is not necessary (and not allowed).

version is available for OPTAIN partners at the UFZ cloud. In the long term, it will be integrated within the official SWATfarmR package.

This section goes through all steps of the main script `write_SWATfarmR_input.R` to generate the `SWATfarmR` management input file.

4.3.1 Load functions and packages

The main script first calls the script `functions_write_SWATfarmR_input.R`. The script is defined with a relative path, so please make sure that the functions script and the main script are in the same folder and you started the `SWATfarmR_input` script version by starting the RStudio project `SWATfarmR_input.Rproj`.

The functions file collects all functions which are called and used in the main script. It also includes routines to install and load all required `R` packages (`foo1()` and `foo2()`). The functions in `functions_write_SWATfarmR_input.R` should not be modified by the user. They can however be useful to look into for debugging.

```r
# Load functions and packages
↪  -------------------------------------------------------
source('./functions_write_SWATfarmR_input.R')

foo1(c("sf" , "tidyverse" , "lubridate", "reshape2", "remotes", "dplyr"))
foo2("HighFreq")
```

4.3.2 Define input files

Next the input files have to be defined. These are the inputs described in the previous section: (1) the land-use crop map, (2) the crop management file, and (3) the generic land-use management file.

```r
# Define input
↪  files---------------------------------------------------------------

## Land-use crop map
## Demo data: lu_crops_CS1.shp
lu_shp <- './input_data/your_lu_crops.shp'

## Crop management table
## Make sure it includes all crops of your lu map
## Demo data: mgt_crops_CS1.csv
mgt_csv <- './input_data/your_mgt_crops.csv'

## Generic land use management table
## Make sure it includes all non-cropland classes with a vegetation cover
## Demo data: mgt_generic_CS1.csv
lu_generic_csv <- './input_data/your_mgt_generic.csv'
```

4.3.3 Further settings

To generate the SWATfarmR input according to your needs, a few variables have to be defined.

In later SWAT+ model runs, the *'management.sch'* input file (which contains the crop rotations generated by `SWATfarmR`) does not include any year information. The model will simply start with the first crop listet in the management schedules, independent from which simulation period has been defined. To ensure that the right crop grows at the right year on a given field, it is mandatory to update the *'management.sch'* file for each simulation period. That means, this script and the `SWATfarmR` need to be executed each time before you run SWAT+ for another simulation period. Here you define the starting and ending year of your crop rotation to make sure they are consistent with the later simulation period in SWAT+.

```
# Define
↪  variables-----------------------------------------------------------------

## Simulation period
start_y <- 2012 #starting year (consider at least 3 years for warm-up!)
end_y <- 2020 #ending year
```

Then you have to define the common prefix of your cropland hrus (all land objects names in the landuse crop map must start with this prefix).

```
## Prefix of land objects which include a crop rotation
hru_crops <- 'field'
```

Finally you have to deal with the problem of multi-year farmland grass (please follow the comments in the Rcode below and the instructions for the crop management file given in the previous section).

```
## Multi-year farmland grass
## Did you define any multi-year farmland grass schedules? 'y' (yes), 'n' (no)
m_yr_sch_existing <- 'y'

## If yes, define also the following variables. If not, skip next four lines
crop_myr <- 'akgs' # name of your farmland grass
max_yr <- 5 # maximum number of years farmland grass can grow before it is killed
↪  (should be <8)
## Do your multi-year farmland grass schedules consider the type of the following
↪  crop (summer or winter crop)?
## (e.g., a '_1.5yr' schedule with a kill op in spring allows for planting a summer
↪  crop immediately afterwards)
## If yes, you must define your summer crops
crop_s <- c('sgbt','csil','barl')
## Do your summer crop schedules usually start with an operation in autumn (e.g.
↪  tillage)?
## To combine them with farmland grass, it is necessary that you provide
↪  'half-year-schedules'
## ('half-year-schedules' are additional summer crop schedules without operations in
↪  autumn)
## The adapted schedules should be added to the crop management table with suffix
↪  '_0.5yr' (e.g. 'csil_0.5yr')
## If additional 'half-year-schedules' are not needed, because your normal summer
↪  crop schedules
## do not start in autumn, type 'n'
additional_h_yr_sch_existing <- 'y' # 'y' (yes), 'n' (no)
```

4.3.4 Run functions to generate the SWATfarmR input files

Now you can execute all further lines without modification.

At first, the input data are read into `R`.

```r
# Read input data ----------------------------------------------------------------

## Read land-use crop map shapefile and drop geometry
lu <- st_drop_geometry(read_sf(lu_shp))

## Read crop management .csv table
mgt_crop <- read.csv(mgt_crop_csv, as.is=T)

## Read generic land use management .csv table
mgt_generic <- read.csv(mgt_generic_csv, as.is=T)
```

Before building the rotation schedules, a check function can be used to see if your crop management file meets all conventions for the *skip* line. To recall the instructions of the previous section: (1) A *skip* line must be included in each individual crop schedule to indicate the change of years. (2) The *skip* line should not be positioned at the last position of the crop schedule.

```r
# Check for correct positioning of 'skip' line ------------------------------------
check_skip <- check_skip_position()
```

As a result, the function returns a message if the check was successful or not. If not, it names the crop for which a *skip* line is missing or set at the wrong (i.e. the last) position. Users might also check file *'check_skip.csv'* which is created when running the function.

The next function combines all individual crop schedules to full crop rotation schedules according to the sequences given in the land-use crop map.

```r
# Build schedules for crop sequences ----------------------------------------------
rota_schedules <- build_rotation_schedules()
```

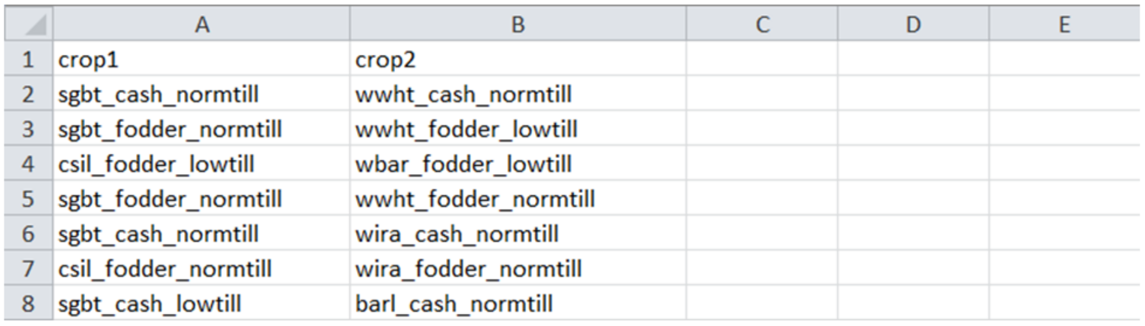
It is very likely that simply combining individual crop schedules causes date conflicts. Especially the dates for the last and first operations of consecutive crops might overlap. The next function identifies all cases with overlapping dates.

```r
# Check for date conflicts --------------------------------------------------------
check_date_conflicts()
```

If there are any conflicts, a return message will tell you. Then you can look into two *'.csv'* files to study the date conflicts.

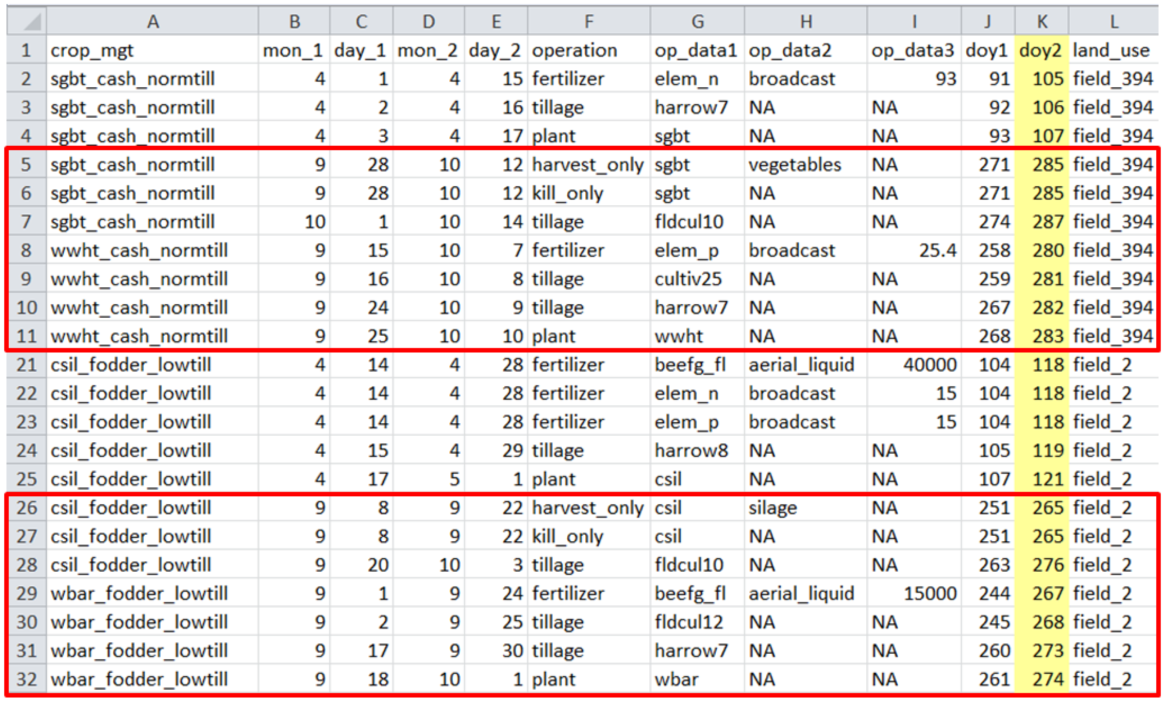
File *'crop_comb_conflict.csv'* lists all crop combinations which caused a date overlap (see for example Figure 4.13).

The conflicts can be studied in more detail in file *'mgt_conflict.csv'*, which lists the full crop schedules where a date overlap has been identified. The date overlap is identified through the day of the year for



	A	B	C	D	E
1	crop1	crop2			
2	sgbt_cash_normtill	wwht_cash_normtill			
3	sgbt_fodder_normtill	wwht_fodder_lowtill			
4	csil_fodder_lowtill	wbar_fodder_lowtill			
5	sgbt_fodder_normtill	wwht_fodder_normtill			
6	sgbt_cash_normtill	wira_cash_normtill			
7	csil_fodder_normtill	wira_fodder_normtill			
8	sgbt_cash_lowtill	barl_cash_normtill			

Figure 4.13: Snippet of an example *'crop_comb_conflict.csv'* file, indicating conflicts among preceding (*'crop1'*) and following crops (*'crop2'*).



	A	B	C	D	E	F	G	H	I	J	K	L
1	crop_mgt	mon_1	day_1	mon_2	day_2	operation	op_data1	op_data2	op_data3	doy1	doy2	land_use
2	sgbt_cash_normtill	4	1	4	15	fertilizer	elem_n	broadcast	93	91	105	field_394
3	sgbt_cash_normtill	4	2	4	16	tillage	harrow7	NA	NA	92	106	field_394
4	sgbt_cash_normtill	4	3	4	17	plant	sgbt	NA	NA	93	107	field_394
5	sgbt_cash_normtill	9	28	10	12	harvest_only	sgbt	vegetables	NA	271	285	field_394
6	sgbt_cash_normtill	9	28	10	12	kill_only	sgbt	NA	NA	271	285	field_394
7	sgbt_cash_normtill	10	1	10	14	tillage	fldcul10	NA	NA	274	287	field_394
8	wwht_cash_normtill	9	15	10	7	fertilizer	elem_p	broadcast	25.4	258	280	field_394
9	wwht_cash_normtill	9	16	10	8	tillage	cultiv25	NA	NA	259	281	field_394
10	wwht_cash_normtill	9	24	10	9	tillage	harrow7	NA	NA	267	282	field_394
11	wwht_cash_normtill	9	25	10	10	plant	wwht	NA	NA	268	283	field_394
21	csil_fodder_lowtill	4	14	4	28	fertilizer	beefg_fl	aerial_liquid	40000	104	118	field_2
22	csil_fodder_lowtill	4	14	4	28	fertilizer	elem_n	broadcast	15	104	118	field_2
23	csil_fodder_lowtill	4	14	4	28	fertilizer	elem_p	broadcast	15	104	118	field_2
24	csil_fodder_lowtill	4	15	4	29	tillage	harrow8	NA	NA	105	119	field_2
25	csil_fodder_lowtill	4	17	5	1	plant	csil	NA	NA	107	121	field_2
26	csil_fodder_lowtill	9	8	9	22	harvest_only	csil	silage	NA	251	265	field_2
27	csil_fodder_lowtill	9	8	9	22	kill_only	csil	NA	NA	251	265	field_2
28	csil_fodder_lowtill	9	20	10	3	tillage	fldcul10	NA	NA	263	276	field_2
29	wbar_fodder_lowtill	9	1	9	24	fertilizer	beefg_fl	aerial_liquid	15000	244	267	field_2
30	wbar_fodder_lowtill	9	2	9	25	tillage	fldcul12	NA	NA	245	268	field_2
31	wbar_fodder_lowtill	9	17	9	30	tillage	harrow7	NA	NA	260	273	field_2
32	wbar_fodder_lowtill	9	18	10	1	plant	wbar	NA	NA	261	274	field_2

Figure 4.14: Snippet of an example *'mgt_conflict.csv'* file. The red boxes highlight operations with a date conflict.

the ending date of the defined operation time windows (*'doy2'*). *'doy2'* must increase monotonically. Any drop within a crop rotation causes a date conflict (see for example Figure 4.14).

All conflicts must be solved, otherwise the `SWATfarmR` will later break up with an error or generate wrong schedules. The user should therefore carefully study the type of date overlaps. Severe overlaps (> multiple months) should be solved manually by editing the crop management file (e.g. adapting dates of existing schedules or add new schedules to avoid these conflicts). However, in most cases the conflicts should be rather small (i.e., dates are overlapping only by a few days or weeks). The next function solves these minor conflicts automatically by shifting the conflicting dates of both the following and the preceding operation iteratively (by one day up and down, respectively) as long as necessary until the conflict is solved.

```
# Solve minor date conflicts (where only a few days/weeks are overlapping)---------
rota_schedules <- solve_date_conflicts()
```

After that, you should run the check function again to make sure all conflicts have been solved.

```
# Check again for date conflicts --------------------------------------------------
check_date_conflicts()
```

If all conflicts have been solved, you can run the last function to generate the `SWATfarmR` input file, called *'farmR_input.csv'*.

```
# Write the SWAT farmR input table ------------------------------------------------
write_farmR_input()
```

It has to be noted that this file ignores the various `SWATfarmR` options to adjust functions, weights, and variables for a more reasonable scheduling of management operations. For example, the filter attribute column is left empty and does not specify any spatial rules (thus, the defined rules apply for all hrus of the same land-use class). Moreover, the following snippet of the `SWATfarmR` conditioning function will be used for each management operation: **'(1 - w_log(pcp, 0, 7)) * (1 - w_log(api, 5, 20))'**. This condition function uses logarithmic functions to define the probabilities for applying an operation depending on (1) the precipitation (pcp) on a given day (with a very high probability (0.99) for pcp = 0 mm , and very low probability (0.01) for pcp = 7 mm) and (2) the antectedent precipitation index (api; for more details on api, please study the SWATfarmR maunual). If required (e.g. if climate conditions in a case study are very special and lead to an unrealistic scheduling of operations), variables and parameter values need to be adjusted manually in the *'farmR_input.csv'* file.

4.3.5 Run the SWATfarmR

A full manual on how to use the `SWATfarmR` R package (see workflow in Figure 4.15) is provided at its public github repository.

Please follow the instructions given there.

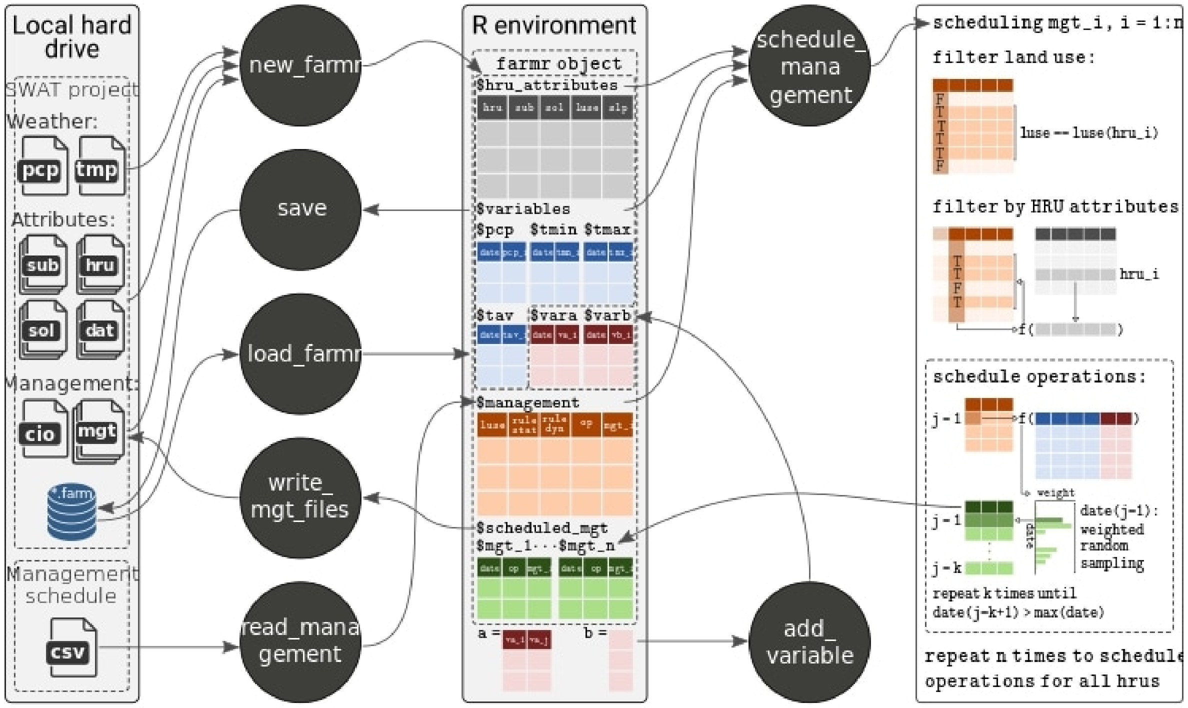


Figure 4.15: Workflow of the `SWATfarmR` R package.

Chapter 5

Decision tables

There are several types of Decision Tables (DT) in SWAT+. They are used to trigger certain operations or adjustments to the model. Those are:

1. **Land Use and Management** – called from *lum.dtl* and control actions and conditions on an HRU basis.
    - Plant/harvest – everything, from a single summer/winter crop to complex crop rotations.
    - Irrigation – unlimited and multiple sources. When parameterizing irrigation in DT the channels cannot be used as water source (since it is flowing water, no rights – first simulated, first serve).
    - Controlled drainage.
    - Grazing.
    - Fertilizer application.
    - Tillage – i.e. fall plow, spring plow, mulch till, no till.
    - CN2 update.
2. **Land Use Scenarios** – called from *scen_lu.dtl* and are triggered outside the routing loop. Scenarios can be variations of:
    - Change of the entire land use.
    - Change hru fractions.
    - Change the USLE P factor – i.e. terracing, contouring, strip cropping.
    - Installation of structures – i.e. install tile.
3. **Reservoir Release** – called from *res.dtl* and trigger the reservoir operations, such as:
    - Demand based release.
    - Direct withdrawal for water rights object.
    - Direct withdrawal for hru (from *lum.dtl*).
4. **Water Diversions** – called from *flo_con.dtl* to condition fractions of flow sent to each outflow object. Water allocation using decision tables is complicated and has its own format. See the Water withdrawals chapter for an in-depth description.

For an explanation on the DT setup in SWAT+Editor, refer to the current documentation. Currently, the functionality of SWAT+Editor does not support the full spectrum of possibilities of SWAT+, hence manual DT creation is advised. We provide some examples of each of the DTs here:

- Standard Land Use and Management DT - are used to define certain automatic operations. Within OPTAIN fixed scheduling will be used, hence only basic information is provided in this protocol.
- Standard Land Use Scenarios DT - once defined, they should be activated by adding the scen_lu.dtl in *file.cio* and enabled in scen_dtl.upd file.
- Standard Reservoir Operation DT - once defined, the name of the reservoir operation DT should be provided in the REL column in the *reservoir.res* file. Note that the reservoir operation DT maybe used for both reservoirs and ponds.
- Standard Water control DT - the functionality was developed to allow users to set up water rights object and provide more flexibility in assigning water rights to individual fields (HRUs). See the Water withdrawals chapter of this document for more information.

Within OPTAIN, DTs will be used seldomly and based on the specific needs of individual CSs. This chapter provides an overview for DTs to determine, if the possibilities provided within SWAT+ are useful in your specific case.

5.1 Land Use and Management DT

The Land Use management decision tables can be used to model several or all the land management operations in an automatic way based on certain conditions. The names of the land use management DTs need to be provided in the *management.sch* file. Management DTs can be flexible and adapted to user needs. If the management is defined as fixed schedule, the DTs can be used to supplement the schedule with auto-operations, i.e. auto-irrigation or auto-fertilization. Although the setup procedure is present in the SWAT+Editor, its functioning has not been sufficiently tested, hence a manual definition of the Land Use Management DT is recommended.

Recommended workflow:

1. Determine what type of automatic management will be implemented. Note that several management DTs could be used at the same time. The standard set of automatic procedures is:

- Irrigation scheduling;
- Planting or harvesting operations;
- Fertilization scheduling;
- Control drainage;
- Hay cutting;
- Grazing;
- Plowing/Tillage;
- Some advance features – i.e. schedule future fertilizer application based on soil tests.

2. Determine the timing (start date or/and triggers) and the location (HRU or HRU group).
3. Refer to the Standard Land Use and Management DTs for examples and find the necessary example that would fit your purpose.
4. Adapt an example or create a new *lum.dtl* file and add it to your model working directory.

5. Apply the management operation for the selected HRUs. This can be done manually, or using the SWAT+Editor
6. Once the definition and the setup are complete, *management.sch* and *lum.dtl* files are written, check if the DT was incorporated correctly. An example of the *management.sch* file with an automatic irrigation operation embedded as a decision table is provided (see Figure @ref(fig:fig-Management.sch-with-DTL)).

5.2 Land Use Scenarios DT

Within the OPTAIN we did not yet agree on the usage of Land Use scnenarios via DTs. Hence, the information provided in this chapter is for general purpose only.

There are three general changes that can be updated in the land use scenario decision tables:

1. Change entire land use and management,
2. Change the USLE P factor for terracing, contouring, and strip cropping,
3. Installation of structures.

Any change in the landuse will trigger an HRU re-initialization with updated management.

Scenarios are different than a new model setup in a way that the changes are performed mid-simulation. At the moment (November 2022), the functionality of the Land Use Scenario DTs has been tested and verified. As aforementioned - although the setup procedure is present in the SWAT+Editor, its functioning has not been sufficiently tested, hence a manual Land Use Scenario DT setup approach is recommended.

Recommended workflow:

1. Determine what type of scenario will be implemented. Note that many scenarios could be run at the same time.
2. Determine the timing (start date or trigger) and the location (HRU type and management, slope, soil, or LU type) of scenario (-s).
3. Refer to Standard Land Use Scenarios DT for examples and find the necessary example that would fit your purpose.
4. Adapt an example or create a new scen_lu.dlt file and add it to your model working directory.
5. Once defined, they should be activated by adding the scen_lu.dtl line in *file.cio* and enabled in scen_dtl.upd file by including a list of active Land Use Scenario DTs.

5.3 Reservoir Release Decision Tables

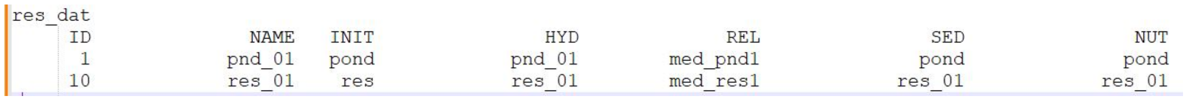
There are multiple ways that a reservoir release DT can be customized. For practical applications, use the provided examples within your SWAT+ installation directory of tested and verified DTs, and simply adapt them to your specific case by changing the names, pointers (IDs), conditions and actions. Once defined, the name of the reservoir release DT should be provided in the REL column in the reservoir.res file. Note that the reservoir release DT maybe used for both reservoirs and ponds.

Recommended workflow:

1. Determine the reservoir name and number for which the release DT should be defined.
2. Determine your preferred way of defining the release rates (described below).
3. Alter the example DT with your reservoir name and number, conditions, actions and alternatives.
4. Include the DT in your project folder.
5. Add the reservoir release DT name to the REL column in the reservoir.res file or the appropriate column in the SWAT+Editor.
6. Save (or generate) the necessary files of your model.

Example setup

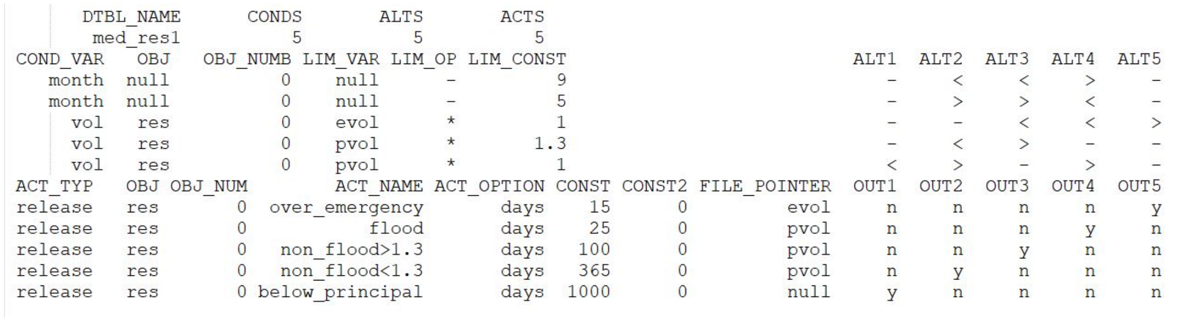
A snippet of the *reservoir.res* file is provided below (Figure 5.5).



```
res_dat
    ID           NAME   INIT          HYD          REL          SED          NUT
     1         pnd_01   pond       pnd_01     med_pnd1         pond         pond
    10         res_01    res       res_01     med_res1       res_01       res_01
```

Figure 5.1: Example of the *reservoir.res* file

The name of the DT used in this example is "med_pnd1" and "med_res1". The DT name is provided in the "REL" column of the file. The "med_res1" DT is given below (Figure 5.6):



```
     DTBL_NAME      CONDS      ALTS      ACTS
      med_res1          5         5         5
COND_VAR   OBJ   OBJ_NUMB LIM_VAR LIM_OP LIM_CONST                          ALT1  ALT2  ALT3  ALT4  ALT5
   month  null          0    null      -         9                             -     <     <     >     -
   month  null          0    null      -         5                             -     >     >     <     -
     vol   res          0    evol      *         1                             -     -     <     <     >
     vol   res          0    pvol      *       1.3                             -     <     >     -     -
     vol   res          0    pvol      *         1                             <     >     -     >     -
ACT_TYP   OBJ OBJ_NUM        ACT_NAME ACT_OPTION CONST CONST2 FILE_POINTER  OUT1  OUT2  OUT3  OUT4  OUT5
release   res       0  over_emergency       days    15      0         evol     n     n     n     n     y
release   res       0           flood       days    25      0         pvol     n     n     n     y     n
release   res       0    non_flood>1.3      days   100      0         pvol     n     n     y     n     n
release   res       0    non_flood<1.3      days   365      0         pvol     n     y     n     n     n
release   res       0 below_principal       days  1000      0         null     y     n     n     n     n
```

Figure 5.2: Example of reservoir release DT

In this example, a simple release rate in "days" is defined based on the volume of the reservoir ((principal volume (*pvol*) and emergency volume (*evol*)) and months of the year. Other release rates that may be used to define reservoir operations are:

1. *rate* - constant rate,
2. *dyrt* - drawdown days,
3. *inflo_rate* - outflow is equal to the inflow rate,
4. *days*, *dyrt* - drawdown days and constant rate,
5. *irrig_dmd* - *wro* - irrigation demand from water resource object, or *hru* - demand for single hru, which allows a fraction (usually > 1.0) of the demand (m^3) to be released,
6. *meas* - daily, monthly or annual measured release rates based on data in the *recall* files.

1.Constant Release Rate: The action type specifies a constant release rate. The reservoir object number can be the specific reservoir simulated or can be set to 0 so it can be used by multiple impoundments. The rate action option is used to set the constant daily release rate in m$^3/s$. In this example, the rate is set at 2.5 m$^3/s$.

ACT_TYP	OBJ	OBJ_NUM	ACT_NAME	ACT_OPTION	CONST	CONST2	FILE_POINTER
release	res	0	over_emer	**rate**	2.5	0	null

2. Drawdown Days: This action is a target approach. The target volume options are principal volume (*pvol*), emergency volume (*evol*), and zero volume (*null*). Water is released from the reservoir to reach the target volume in the given number of drawdown days. For example, when the volume is over the emergency spillway volume, water is released so the reservoir volume will be at the emergency volume in 5 days (1/5 of volume over emergency is released).

ACT_TYP	OBJ	OBJ_NUM	ACT_NAME	ACT_OPTION	CONST	CONST2	FILE_POINTER
release	res	0	over_emer	**days**	5	0	evol
release	res	0	over_prim	**days**	15	0	pvol
release	res	0	under_prim	**days**	100	0	null

3. Outflow Equal Inflow: This action (*inflo_rate*) sets daily outflow to daily inflow. A minimum release can also be specified (i.e. 2.5 m^3/s). The release is set as the maximum of inflow and the minimum release.

ACT_TYP	OBJ	OBJ_NUM	ACT_NAME	ACT_OPTION	CONST	CONST2	FILE_POINTER
release	res	0	out=in	**inflo_rate**	2.5	0	null

4. Drawdown Days + Constant Rate: This option uses the drawdown day release added to a constant release. Parameters were developed for 150 reservoirs in the U.S. as part of NAM (National Agroecosystem Model) (White et al., 2022). The paper by (Jingwen Wu, 2022) describes the parameter development for such applications. In this example, for reservoir 101, with volume at multiple use, non-flood conditions, the drawdown days are set to 12 with 1.5 m$^3/s$ added to determine total release volume.

ACT_TYP	OBJ	OBJ_NUM	ACT_NAME	ACT_OPTION	CONST	CONST2	FILE_POINTER
release	res	101	multiple_use_nf	**dyrt**	12	1.5	null

5. Irrigation Demand: The *irrig_dmd* option allows water to be released based on irrigation water demand. In this example, water is released at the rate of 1.2 times the irrigation demand of water rights object 1.

ACT_TYP	OBJ	OBJ_NUM	ACT_NAME	ACT_OPTION	CONST	CONST2	FILE_POINTER
release	res	0	irrig_release	**irrig_dmd**	1.2	1	wro

6. Measured Outflow: Daily, monthly, or annual releases can be input by the user. The measured output file is specified in the FILE_POINTER input.

ACT_TYP	OBJ	OBJ_NUM	ACT_NAME	ACT_OPTION	CONST	CONST2	FILE_POINTER
release	res	1	meas_daily	**meas**	0	0	res1_daily
release	res	5	meas_monthly	**meas**	0	0	res5_monthly
release	res	7	meas_annual	**meas**	0	0	res7_annual

The model finds the appropriate file in *recall.rec* and sets daily outflow according to data in each recall file.

Chapter 6

Model evaluation

Model evaluation in a very general sense assesses how the SWAT+ model setups are able to reproduce observable environmental variables with simulated outputs. Observation data which is compared to the model simulations can have contrasting characters, whether they represent hard measurable data or are just soft information that can be related to model outputs. Section 6.1 addresses different kinds of data that can be implemented in model calibration and validation.

Section 6.2 covers the large field of model calibration and outlines the calibration workflow that was proposed as a harmonized workflow for all OPTAIN case studies. The developed workflow is separated in three main steps, where the first step in the procedure is a comprehensive model setup verification (section 6.2.2), followed by a soft calibration procedure (section 6.2.3) and a hard calibration that fine tunes model parameters in order to represent observations of in-stream discharge, nutrient, or sediment concentrations/loads adequately well (section 6.2.4). To validate a calibrated model setup with observation data that has not been used in the model calibration is considered to be good practice. Thus, all OPTAIN case studies will perform model validation which is outlined in section 6.3.

6.1 Data

The first step in preparation of calibration data is identification of available data, which could be used for model calibration and validation. Generally, relevant data fall under two categories: hard and soft data. Hard calibration data are long-term, measured time series, typically at a certain point within a catchment, e.g. the main outlet (Arnold et al., 2016). For models such as SWAT+ these are most frequently streamflow or water quality parameters (N, P, sediment concentrations or loads). The source of hard data is typically hydrometeorological service or agencies responsible for water quality monitoring; alternatively, monitoring performed for research purposes. Hard data can be used to evaluate model confidence level using different model performance indices (Bennett et al., 2013; Knoben et al., 2019; Moriasi et al., 2015, 2007).

Soft data are defined as “information on individual processes within a budget that may not be directly measured within the study area, may be just an average annual estimate, and may entail considerable uncertainty” (Arnold et al., 2016). Soft data includes, for instance, information on ET (which might be estimated from remote sensing products or literature), baseflow ratio (can be obtained using digital filters from streamflow), other groundwater related information (e.g. drain tile flow), share of flow fractions (e.g. average share of lateral flow, surface flow), Leaf Area Index (LAI) development, crop yields (available from agricultural statistics), erosion rates, nutrient uptake, denitrification rates (available from prior studies or literature), even nutrient load estimations based on different discharge levels and measured concentrations (e.g. for different seasons) and literature values on the effects of measures. Several possible sources of soft data can be distinguished: refereed literature; engineering, technical, and research reports; unpublished documents (theses and dissertations); and field surveys (Arnold et al., 2016). Remote sensing data (e.g. ET, soil moisture) could be used either as hard (time series) or soft (time-averaged) data, but due to small catchment scale and predominantly coarse resolution of these sources, they are not recommended to be used in OPTAIN.

6.1.1 Hard data collection and quality

Traditionally, catchment-scale models were calibrated using hard data only but numerous studies pointed at short-sightedness of this approach (Arnold et al., 2016; Seibert and McDonnell, 2003). The major risk is that achieving optimal statistics in hard calibration does not guarantee an accurate representation of internal watershed processes, which in consequence may lead to a lack of meaningful results for any model-based scenarios. Using soft data can help constrain ranges of sensitive parameters influencing these processes. Since scenarios involving both single and multiple NSWRM and future climate change are fundamental in OPTAIN, the calibration approach presented in this protocol should not be solely focused on hard data and should account for evaluation of internal processes as much as possible. The advantage of using SWAT+ over previous SWAT versions is that it allows an easy inclusion of soft data in setting key parameters to appropriate levels, before pursuing hard calibration.

There are many issues to think about during preparation of hard data for calibration. Among the most important ones is the question if there is sufficient data for model calibration and validation. Traditionally, at least several years of measured data representing average, wet and dry conditions are considered sufficient for a hard calibration of a hydrological model. For sites with no record or with only a short record, methods for reconstruction of streamflow records could be applied. The simple methods rely on transfer of information from nearby flow gauges via: (1) the use of drainage area ratios, (2) the use of estimates of monthly means and standard deviations based on regional streamflow-basin characteristics models, and (3) two different methods of using the cross-correlation of flow records (Hirsch, 1979). Of course, using such approaches increases the modelling uncertainty.

The next important question is evaluation of data quality and suitability for modelling objectives. Below are example questions that should be considered in this step:

- How were data collected (i.e., timing, frequency, composite or grab samples)?
- Were handling/storage, processing, analysis, and QA/QC methods suitable for intended purpose and modeling objectives?
- What is the measurement uncertainty?
- Is the quality of the measured data sufficient given the intended model use?
- Is there a consistency among different data sources (including units)?
- How to identify potential outliers and should they be included or excluded from the data set?

To facilitate answering to some of these problems, several functions were included in the R package `svatools` with the aim of analysing and pre-processing of calibration data as shown at the following examples:

- loading GIS and time series data from excel templates;

```
library(svatools)
temp_path <- system.file("extdata", "calibration_data.xlsx",
                         package = "svatools")
##temp_path is path sting to calibration data excel file
cal_data <- load_template(temp_path, epsg_code = 4326)
```

- plotting measured data from different sources on one figure in order to select more consistent and reliable data;

```
##Example for plotting data of two stations
plot_cal_data(cal_data$data, c("3","10"))
```

- defining realistic values (e.g. within realistic bounds, positive);

```
##Example plot for one data rich station could be used as
##interactive tool to explore potential data problems
plot_cal_data(cal_data$data, c("4"))
```

- checking if total N or P are not smaller than their forms (e.g. mineral, organic);

```
##Example provides figure comparing changes in Pmin/TP ratio
##for each month
plotP <- plot_fractions(cal_data$data,
                       station = c("4"),
                       c("PT"), c("P-PO4"))
plotP$fraction

##Same figure but providing Nmin to TN monthly ratio changes
plotN <- plot_fractions(cal_data$data,
               station = c("4"), c("NT"),
               c("N-NO3", "N-NH4", "N-NO2"))
plotN$fraction
```

- defining if zero values could be used (or limit of quantification/detection is more appropriate as the lowest level);

```
##Generally, zeros should not be allowed in case of water quality.
##Better practice replacing them with half of limit of quantification/detection.
##clean_wq function could be used in this case.
cal_data$data <- clean_wq(cal_data$data, 0.5)
```

- identifying outliers (e.g. as values outside mean plus n times of standard deviation);

```
##Outliers identified as values by 3 standard deviations from mean.
lst <- clean_outliers(cal_data$data, times_sd = 3)
##Printing data to be removed.
print(head(lst$dropped))
##Updating calibration data after removal of outliers.
cal_data$data <- lst$newdf
```

- plotting the same data in different ways to identify problems.

```
##Example plotting monthly regression figure (after previous
##function) Mineral vs. total phosphorus or nitrogen
plotP$regression
plotN$regression

##Example of plotting monthly summaries for station 4
plot_monthly(cal_data$data, station = "4",
             drop_variables = c("Q"))

##Example of putting time series and station location data
##on an interactive map to explore.
plot_map(cal_data$data, cal_data$stations, reach_sf, basin_sf)
```

Such simple functions allow us to quickly identify and correct problems, thus saving time and avoiding more problems at later stages.

Streamflow and other hydrological data examination can be helpful to get a better understanding of the dynamics of hydrological processes in the catchment that may appear useful in calibration. For instance, observed streamflow and precipitation time series can be plotted together (as a hydrograph combined with a hyetograph) to assess the strength of rainfall-runoff relationship for different type of events as well as to identify potential anthropogenic effects modifying the flow regime (e.g. dams, water abstractions, water transfers, etc.). Ideally, if such anthropogenic effects exist in the studied catchment, they should be included in the model setup. However, if this is difficult, there are numerous methods for streamflow naturalization available (Terrier et al., 2021).

It is important to keep in mind that enough data should be available and set aside for model validation, which could not be used in calibration. Calibration and validation data could be separated in time (different time periods) or space (different locations), or both. Additionally, for larger watersheds multi-site calibration and validation should be applied, if there are enough data.

6.1.2 Soft data collection and quality

Soft calibration planned in OPTAIN will, as a minimum, require three data items:

1. Water yield ratio [-] – the ratio of average annual water yield (total of surface, lateral, tile, perc) to average annual precipitation.
2. Baseflow ratio [-] – the ratio of average annual baseflow (total of lateral, tile, perc) to average annual water yield.
3. Average annual crop yields [t/ha] for different crops included in the model setup.

Water yield ratio is here defined using SWAT+ language, but it could be approximated by a ratio of average annual discharge measured at the catchment outlet to average annual precipitation. If discharge is known to be heavily modified by upstream human pressure (withdrawals, point sources), a naturalized discharge should be used instead. National or regional hydrological atlases could also be a useful source to identify this index, but such data can be interpolated in space and thus a more rough approximation of the conditions of studied catchment.

There are dozens of methods in hydrology to derive the baseflow ratio for a catchment. More sophisticated methods would require additional data and involve separate modelling, which may be not feasible in OPTAIN. Therefore more simple approaches are recommended. Digital baseflow separation filters such as the one available on the SWAT+ website, conceptually based on the work of Arnold et al. (1995), are a good example. According to developers of this tool, the fraction of water yield contributed by baseflow should fall somewhere between the value for *Baseflow Fr1* and *Baseflow Fr2*, unless baseflow in the studied catchments is from aquifers recharged by precipitation falling outside the catchment.

Crop yield data are of different nature than water yield and baseflow ratios. It may be hard, if not impossible, to get the yield data exactly matching the model simulation time for all fields inside the studied catchment. Data sources may be country-specific, but typically agricultural census data should be available in each country. The problem may be that they are available for large administrative units such as the NUTS2 regions in the EU. Therefore, more representative data may be available from the local agriculture advisory services, farmer organisations or individual farmers.

It should be kept in mind that SWAT+ crop yield is on the dry weight basis, whereas typical crop yield statistics are on the fresh weight basis. Thus, raw observed data should be multiplied by a crop-specific conversion factor before calibration. Since these factors may slightly differ between countries, it is recommended to search for national data sources on humidity degrees of different crops. Alternatively, EU-standard humidity degrees for various crops could be used (EUROSTAT, 2020).

6.2 Calibration

All OPTAIN CSs focus on different environmental fluxes and how they are affected by NSWRM which will be simulated with the SWAT+ model setups. While some CSs must address soil erosion and sediment loads as their main issues, other case studies must for example focus on nitrate losses or water scarcity. Regardless of the CSs main focus areas, all model setups must adequately represent the study catchment's water balance and dominant runoff processes. To harmonise the SWAT+ model calibration activities in OPTAIN a general calibration workflow was developed which should establish a certain standard for all SWAT+ model setups and their model performance.

6.2.1 Calibration workflow

Standard model calibration procedures strongly rely on the explanatory value of specific hydrological performance metrics and thresholds which accept or reject a model performance. The calibration procedure in OPTAIN aims to put a strong focus on the plausible simulation of the relevant eco-hydrological processes which can hardly be reflected by single model performance metrics. Considerations of all relevant processes resulted in a comprehensive multi-step calibration workflow which can be summarised in three phases (see Figure 6.1).

i) After the model setup is completed a model verification should identify any potential wrong inputs, which would result in implausible model simulations. This is particularly relevant for complex inputs such as management operations or decision rules which may be interpreted in a different way by the model as intended.

ii) SWAT+ provides soft calibration routines for the separation of water in the hydrological system into different water balance components and for adjusting simulated crop yields. The soft calibration routines suggest initial values for a few dominant model parameters to support the hard calibration procedure.

iii) In the hard calibration functional groups of model parameters are fit to improve the simulation of different hydrological processes, sediment yield or the transport of nutrients such as phosphorus and nitrate-N. In the following sections all three phases of the proposed calibration procedure will be addressed in detail.

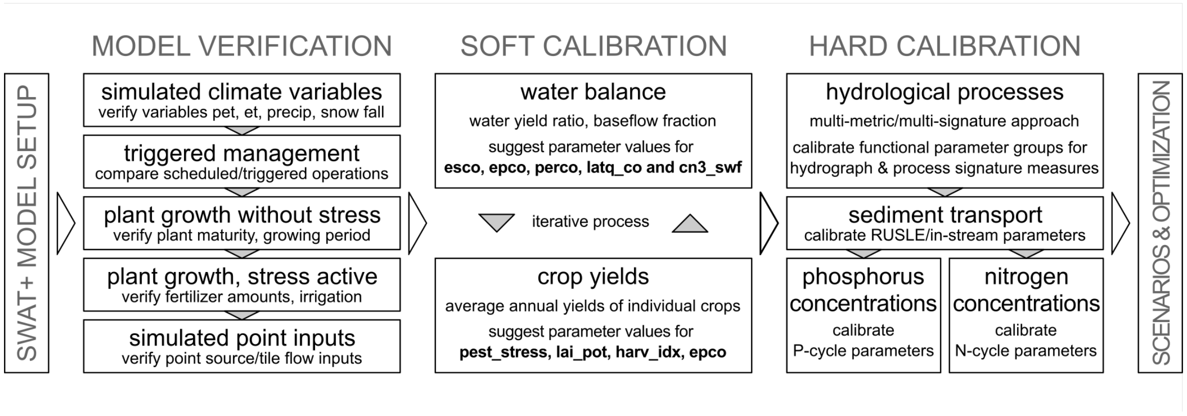

MODEL VERIFICATION
SOFT CALIBRATION
HARD CALIBRATION
SWAT+ MODEL SETUP
simulated climate variables
verify variables pet, et, precip, snow fall
triggered management
compare scheduled/triggered operations
plant growth without stress
verify plant maturity, growing period
plant growth, stress active
verify fertilizer amounts, irrigation
simulated point inputs
verify point source/tile flow inputs
water balance
water yield ratio, baseflow fraction
suggest parameter values for
esco, epco, perco, latq_co and cn3_swf
iterative process
crop yields
average annual yields of individual crops
suggest parameter values for
pest_stress, lai_pot, harv_idx, epco
hydrological processes
multi-metric/multi-signature approach
calibrate functional parameter groups for
hydrograph & process signature measures
sediment transport
calibrate RUSLE/in-stream parameters
phosphorus concentrations
calibrate
P-cycle parameters
nitrogen concentrations
calibrate
N-cycle parameters
SCENARIOS & OPTIMIZATION

Figure 6.1: Proposed calibration workflow in OPTAIN.

6.2.2 Model verification

The calibration of model parameters is usually computationally expensive. Thus, it is highly valuable to invest some time in the verification of a SWAT+ model setup before starting with the model calibration. Verification may only require a few model simulations and analyses to identify issues which would cause substantial obstacles in the model calibration. The presence of certain issues in a model setup can in the worst case require fixing the issues and repeat an entire model calibration. It is therefore essential to perform model verification at the beginning of a calibration procedure to identify common issues in a model setup early enough.

In OPTAIN we propose a 5 step procedure for the model setup verification (first column in Figure 6.1) which addresses common issues in SWAT model setups. The first step analyses the overall simulated water balance for a model setup and mainly tries to assess if the weather input data are interpreted correctly and result in a plausible simulation of the climate variables. The steps 2, 3, and 4 focus on the simulation of (farm) management operations and the simulation of plant growth. Plant growth is a central part of a SWAT simulation and controls the Actual Evapotranspiration (ETa), a substantial fraction of the hydrological water balance. Further, plant growth is a complex process which is controlled by multiple parameters and inputs in a SWAT+ model setup. In order to simulate a plausible plant development and in consequence to produce plausible simulations of ETa it is vital to verify the simulation of plant growth related processes in a SWAT+ model setup. The final step 5 analyses the inputs into channels from point sources (e.g. WWTPs, water transfer) and from tile flow of agricultural land objects. The verification of point inputs should mainly assess if the point inputs (units, order of magnitude) and the tile inputs (does tile flow occur?) were parameterised correctly.

Some of the verification steps can be generalised and supported by visual analyses of simulation outputs. To harmonise the SWAT+ model setup verification in OPTAIN we developed the `R` package `SWATdoctR` which provides routines for model diagnostics. The package is still under development and its functionality will be extended and updated throughout the project. The following code examples used `SWATdoctR` in the version 0.1.1, which is available from the UFZ GitLab. To install `SWATdoctR` in `R` the following lines of code should be executed.

```r
# If the package 'remotes' is not installed
install.packages('remotes')

remotes::install_git('https://git.ufz.de/schuerz/swatdoctr')

library(SWATdoctR)
```

6.2.2.1 Running the SWAT+ model and extracting outputs

The model verification requires specific model outputs from a SWAT+ model run to perform analyses and identify potential issues. `SWATdoctR` provides the function `run_swat_verification()` to run a simulation for a SWAT+ model setup to adjust some settings in the model setup and to extract the relevant simulation outputs for any further analyses. The minimum input which must be defined for `run_swat_verification()` is the `project_path` to define where the SWAT+ model setup is located on the local hard drive.

Further, the user can activate/deactivate to read certain outputs. Not all outputs are required for all analyses and some outputs can be too large to fit into the RAM. By default `run_swat_verification()` reads all outputs which are defined with the input argument `outputs = c('wb', 'mgt', 'plt')`. `outputs = 'wb'` defines to read the output files *'basin_wb_day.txt'* and *'basin_pw_day.txt'*. These two files are required to analyse all climate variables at the basin scale. `outputs = 'mgt'` defines to read the output file *'mgt_out.txt'*, which is required to analyse the management operations that were set in the simulation run. `outputs = 'mgt'` further reads input files such as *'landuse.lum'*, or *'hru-data.hru'* as HRU properties are in some cases required to be linked with management operations. `outputs = 'plt'` defines to read the output file *'hru_pw_day.txt'*. This file provides HRU outputs at daily time steps and therefore it can be rather large for large model setups and long simulation periods. Therefore, it may be necessary to exclude *'hru_pw_day.txt'* when reading the outputs. A few plot functions are then not available in the model setup analysis.

With the input arguments `start_date`, `end_date`, and `years_skip` the simulation period and the years that are skipped in printing simulated outputs can be controlled. If these input arguments are set `NULL` the values which are defined in the model input files *'time.sim'* and *'print.prt'* will be used to define the simulation period. These input arguments should not be used if the management of the SWAT model setup were generated with the `SWATfarmR`, as `SWATfarmR` always writes management schedules together with the simulation period to make sure that the used weather input time series and the scheduled crop rotations are in line.

As indicated in Figure 6.1, some analyses must be performed with plant stress factors activated and under unconstrained plant growth conditions. The setting for plant stress can be done with the input argument `nostress`. `nostress = 0` activates all stress factors for plant growth in the simulation, `nostress = 1` deactivates all stress factors, and `nostress = 2` only activates nutrient stresses.

SWAT simulations are never performed in the original project folder, but a copy of the project is generated in the subfolder *'.run_verify'*, which is deleted after the simulation. `keep_folder` is an optional input argument which controls if the simulation folder should be kept and not be deleted after the simulation runs. This option can be useful for debugging and checking if `run_swat_verification()` worked as intended.

In the example below a simulation for model verification was performed for a SWAT+ model setup where all stress factors were deactivated. Deactivating plant stresses can be useful for the first verification steps as with this setting the analysis of climate variables shows ETa values where the main drivers and constraints for plant growth are the climate inputs. Further, if the first check of the climate variables was OK the verification of plant growth without plant stresses can be immediately performed with the same simulation outputs without having to repeat the simulation runs.

```
model_path <- 'Define:/your/path'

sim_nostress <- run_swat_verification(project_path = model_path,
                                      outputs = c('wb', 'mgt', 'plt'),
                                      nostress = 1)
```

6.2.2.2 Step 1: Analysis of simulated climate variables

The climate variables daily precipitation and daily minimum/maximum temperatures are required inputs of a SWAT+ model setup (see more Weather data). Further climate inputs such as solar radiation, relative humidity and wind speed are optional input variables and can be essential for the calculation of the potential evapotranspiration (PET). Climate inputs are grouped to weather stations in a model setup and are assigned to spatial objects (HRUs, channels, reservoirs, etc.) with the nearest neighbour method.

The input of weather data and the assignment of climate variables to spatial objects can be sources for several issues which must be analysed:

- Data structure of the climate input tables, units of the climate variable, no data flag, etc. was wrong and can result in unrealistically small or large values of the climate variables in the simulation.
- The nearest neighbour assignment allocates weather stations to spatial objects where the weather records do not represent the actual weather conditions in a spatial object well. This can for example be an issue in complex terrain.
- The selected method for the calculation of PET results in an under/overestimation of PET when compared to estimates of PET for the region. In such cases other methods for the simulation of PET which are included in SWAT+ should be tested if they better fit the regional conditions and available weather inputs (see more Additional settings).

Large implausibilities in the weather inputs can be identified in analyses of annual basin averages of the simulated climate variables. Simulated annual and average values of climate variables must be comparable to observation data and/or region specific literature values. Any larger deviations of precipitation can indicate errors in the input file or an inappropriate assignment of weather stations to spatial units. If the lapse rate option is active (see more Additional settings), it may be another potential reason for deviations from observations. Over or underestimated PET can indicate errors in the temperature input files (and if provided in the solar radiation, relative humidity and wind speed inputs). Also issues in the assignment of weather stations and the used method for the simulation of PET should be verified in such cases.

`SWATdoctR` provides the function `plot_climate_annual()` to analyse the annual simulated basin averages of climate variables. The function only requires the simulation results from `run_swat_verification()` as an input to plot the climate variables. In `run_swat_verification()` the `outputs` must at least include the basin water balance outputs defined with `outputs = 'wb'`.

```
plot_climate_annual(sim_nostress)
```

Executing the function for an example SWAT+ project resulted in the plot shown in Figure 6.2. The first plot panel shows annual basin PET values as black lines. ETa is split into the three simulated

ET fractions ET from canopy interception storage (`ecanopy`), plant transpiration (`eplant`), and soil evaporation (`esoil`). At the right side of the panel average annual values for all fractions and the total actual evapotranspiration (`et`) are printed. For the simulated study region PET and total ETa are plausible. The proportion of `eplant` is however a bit low and is compensated by a large `esoil`. This finding must be further investigated. In the current version of SWAT+ (rev 60.5.5) this is unfortunately still a common behaviour, that simulated ETa fractions are not always plausible

The second plot panel shows the precipitation fractions rainfall (`rainfall`) and snowfall (`snofall`). In the example both look plausible. Again the summary statistics on the right show the average annual values.

The third panel shows the annual temperature values. The black lines are the annual average temperature values. The red and the blue lines are the maximum daily max temperature and the minimum daily min temperature for each year. Particularly the annual extreme values are a good indicator for wrong inputs. In this example the average, min and max temperatures look plausible.

The lowest panel shows the annual sums of solar radiation. A comparison to literature values of annual solar radiation sums for the region can indicate issues in this input. In the example in Figure 6.2 the simulated basin average is comparable to records for the region.

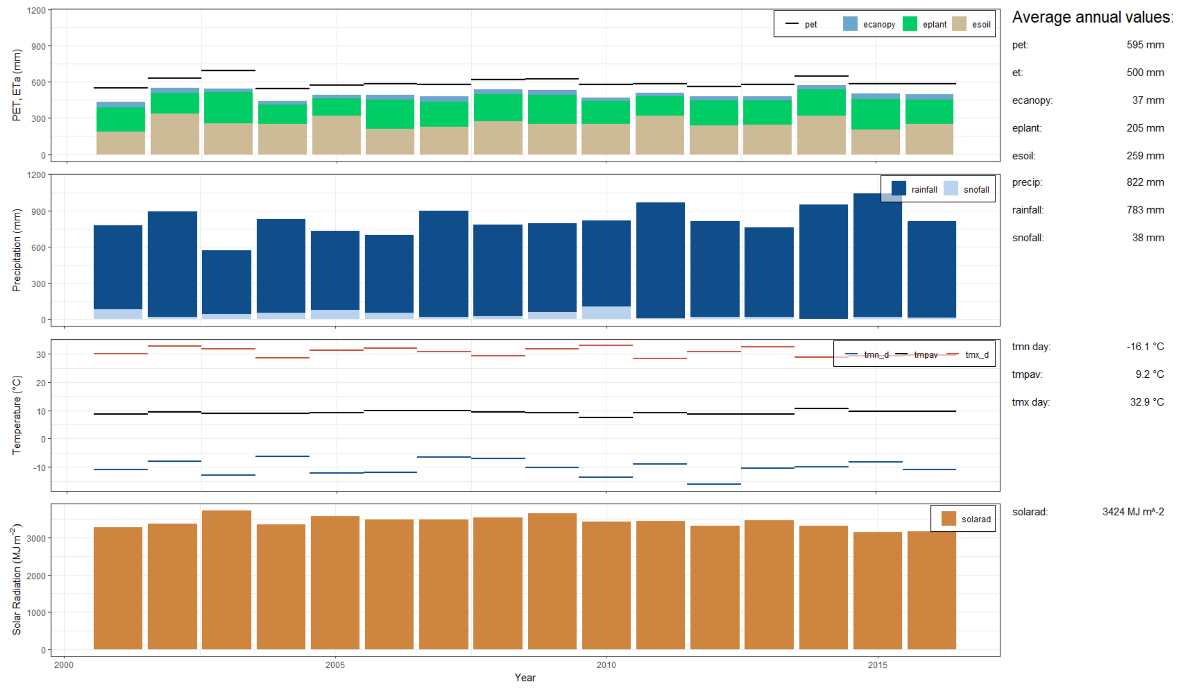


Figure 6.2: Example plot of simulated basin climate variables with the function `plot_climate_annual()`.

The analysis of mean monthly precipitation (output variable *'precip'*), snowfall (output variable *'snofall'*) and snowmelt (output variable *'snomlt'*) sums and their comparison with region specific information (or in the best case observations) provides insight in seasonal dynamics of the precipitation input. Particularly in snow impacted catchments a first verification of snowfall is valuable to see whether precipitation in solid form is simulated, a snow storage can build up and cause increased spring runoff through snow melt. The hydrological cycle of some catchments may be dominated by spring flood events which must be reflected by the simulated processes. Any observed implausibility in such analysis can indicate issues in the weather inputs or require to pay attention in the calibration of model parameters which control the simulation of snow processes (*snofall_tmp*, *snomelt_tmp*, *snomelt_lag*).

`SWATdoctR` provides the function `plot_snow_monthly()` to analyse the simulated average monthly basin values of precipitation, snowfall and snowmelt. The function only requires the simulation results from `run_swat_verification()` as an input to plot the climate variables. In `run_swat_verification()` the `outputs` must at least include the basin water balance outputs defined with `outputs = 'wb'`.

```
plot_snow_monthly(sim_nostress)
```

Figure 6.3 shows the resulting plot for the test catchment. Although this catchment is impacted by spring snowmelt events to a smaller extent, the plot can be helpful to verify the snow processes and provide guidance to adjust the snow parameters if necessary.

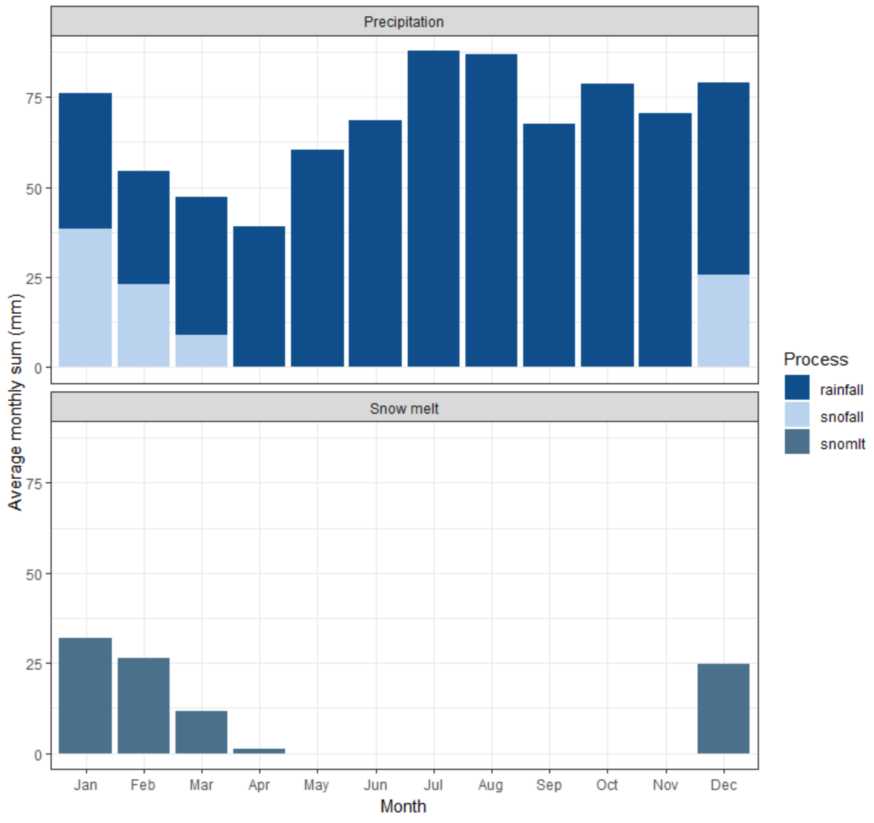


Figure 6.3: Example plot of simulated basin variables precip, snofall, and snomlt with the function `plot_snow_monthly()`.

If inconsistencies in the weather data are identified they should be fixed before continuing with the model verification (see Weather data section). This may require revising the weather input data and to reload them in the SWAT+Editor. PET is a simulated climate variable which employs other input data such as temperature, solar radiation, relative humidity, or the wind speed. In OPTAIN we recommend the PM method for the calculation of PET. Reasons for this choice were outlined in section 3.11.1. In situations where not all required climate inputs are available which are necessary to estimate PET with PM method the estimates will be more uncertain and annual PET sums may differ to regional values. Then the use of a simpler method for the calculation of PET can be a valid solution.

6.2.2.3 Step 2: Simulation of management operations

Management operation inputs in a SWAT+ model setup can be very complex and comprehensive. Just the simulation of a few different crop rotation schemes can already require management input files with several hundred lines. Scheduled operations point to several other input files which define the parameters of operations or inputs such as fertiliser or tillage types. Hence, the development of management schedules is highly error prone. Mistakes in management inputs usually do not stop a simulation or produce warnings in the model diagnostics, but lead to skipping certain operations in a simulation run. These circumstances can impede the validation of simulated management schedules and it can become difficult to identify single erroneous lines in the scheduled management operations.

All operations which are triggered in a SWAT+ simulation run are written into the file *'mgt_out.txt'*. To verify the correct triggering of the scheduled operations in `R`, a tabular comparison between scheduled and simulated operations is the most robust approach. Such a procedure can be cumbersome and only allows to select a few HRUs to perform a comparison. Yet, in most cases it might only be necessary to select a few cases for comparison to see if the scheduled operations work properly.

`SWATdoctR` offers two approaches to investigate management operations in tabular form. `report_mgt()` generates an overview report where the scheduled and triggered operations are matched and compared for each management schedule that was implemented in the simulations. The function prepares the scheduled management operations that were written in the input file *'management.sch'* in tabular form and randomly samples one HRU for each defined schedule from the triggered management operations (from the output file *'mgt_out.txt'*). The comparison is only done for operations that were defined with a fixed date in the management schedule and operations which are triggered by decision tables will be excluded.

Applying the function `report_mgt()` for the model verification simulation outputs returns a table with an overview of the operations which were scheduled but not triggered or operations where *'op_data1'* differs in the scheduled and triggered operations.

```
mgt_report <- report_mgt(sim_nostress)

mgt_report

#> # A tibble: 3 × 3
#>   schedule    op_issue schedule_report
#>   <chr>          <int> <list>
#> 1 agrr_rape          1 <tibble [1 × 8]>
#> 2 agrr_wbar          1 <tibble [1 × 8]>
#> 3 agrr_wwht          1 <tibble [1 × 8]>
```

The example table shows that 3 schedules were identified where issues in the scheduled and triggered operations were identified. Further detail on the reported issues are available from the column `schedule_report`. To access the detailed information the respective entry in the table `mgt_report` can be called. The differing operations for the schedule *'agrr_wwht'* can be accessed the following.

```
mgt_report$schedule_report[3]

#> [[1]]
#> # A tibble: 1 × 8
#>    year   mon   day op_typ op_data1_trig op_data1 op_data2 op_data3
#>   <dbl> <dbl> <dbl> <chr>  <chr>         <chr>    <chr>       <dbl>
#> 1     1     8     1 harv   NA            rape     grain           0
```

The example shows that a harvest operation in the first year of simulation was scheduled but the operation was not triggered (`op_data1_trig = NA`). A reason for that can be that there was no initial land cover with *'rape'* as a crop defined in the file *'plant.ini'* for this land use.

`report_mgt()` is a good starting point to explore the triggered management. But this analysis can be error prone. Still the safest way to analyse the triggered and the scheduled managements is to compare the input and output tables. `SWATdoctR` provides the function `print_triggered_mgt()` to print the triggered managements for individual HRUs. For selecting HRUs e.g. with a specific management the helper function `get_hru_id_by_attribute()` can be useful. In the example below the id for an HRU was selected that uses the management `mgt = 'agrr_wwht'`, which is in this case e.g. HRU 10. `print_triggered_mgt(sim_verify = sim_nostress, hru_id = hru_agr$id[1])` then shows the management which was triggered for the HRU 10. The table is actually longer and only the first 3 years are shown here for demonstration. This table can now be visually compared with the management input table (*'management.sch'*).

```
hru_wwht <- get_hru_id_by_attribute(sim_nostress, mgt = 'agrr_wwht')

print_triggered_mgt(sim_verify = sim_nostress, hru_id = hru_wwht$id[1])

#> Triggered managament for
#>   hru:        10
#>   management: agrr_wwht
#>
#># A tibble: 146 × 7
#>     year   mon   day phuplant operation op_data1 op_data3
#>    <dbl> <dbl> <dbl>    <dbl> <chr>     <chr>       <dbl>
#>  1  2000     3     1  0.167   FERT      elem_n         90
#>  2  2000     4     1  0.323   FERT      elem_n        100
#>  3  2000     8     1  0       KILL      rape            0
#>  4  2000     9    10  0.708   FERT      elem_p          0
#>  5  2000     9    20  0.867   TILLAGE   fallplow        0
#>  6  2000    10     1  0       PLANT     wwht            0
#>  7  2000    10     1  0       TILLAGE   rothoe          0
#>  8  2001     2    25  0.0976  FERT      elem_n         80
#>  9  2001     4    10  0.270   FERT      elem_n         60
#> 10  2001     6    10  0.946   FERT      elem_n         80
#> 11  2001     8    10  2.06    HARVEST   wwht            0
#> 12  2001     8    10  0       KILL      wwht            0
#> 13  2001     9     1  0       FERT      elem_p          0
#> 14  2001     9     5  0       TILLAGE   fallplow        0
#> 15  2001     9    20  0       PLANT     wbar            0
#> 16  2001     9    20  0       TILLAGE   rothoe          0
#> 17  2002     3    20  0       FERT      elem_n         80
#> 18  2002     4    20  0       FERT      elem_n        110
#> 19  2002     7    10  8.76    HARVEST   wbar            0
#> 20  2002     7    10  0       KILL      wbar            0
#> 21  2002     7    20  0       FERT      elem_p          0
#> 22  2002     8    15  0       TILLAGE   fallplow        0
#> 23  2002     8    20  0       PLANT     rape            0
#> 24  2002     8    20  0       TILLAGE   rothoe          0
#> 25  2002    10     1  0       FERT      rind        10000
#> 26  2003     3     1  0       FERT      elem_n         90
#> 27  2003     4     1  0        FERT      elem_n        100
#> 28  2003     8     1  0.651    HARVEST   rape            0
#> 29  2003     8     1  0        KILL      rape            0
#> 30  2003     9    10  0        FERT      elem_p          0
#> 31  2003     9    20  0        TILLAGE   fallplow        0
#> 32  2003    10     1  0        PLANT     wwht            0
#> 33  2003    10     1  0        TILLAGE   rothoe          0
```

Operations which are missing in the simulated management schedules must be checked in the *'management.sch'* input file. By answering the following questions for the scheduled management operations their proper implementation in the model setup can be verified:

- Are the date sequences in the scheduled operations correct and in a right order (mistakes in assigned month and day values)?
- Does the variable *op_data1* point to the correct entry in the respective input data file? Does the label exist in the input file? E.g. does defined *op_data1* exist in *'tillage.til'* for tillage operations, or does defined *op_data1* exist in *'plant.plt'* for plant operations.
- Does the variable *op_data2* point to the correct entry in the respective operations file (*'.ops'*)? E.g. does harvest operation defined with *op_data2* exist in *'harv.ops'*.

6.2.2.4 Step 3: Analysis of unconstrained plant growth

The verification of plant growth is a two-tiered approach. In a first step plant growth is simulated and analysed without simulating any limiting stress factors. Such analysis illustrates the potential biomass or yield a plant can gain given the climatic and soil conditions of the simulated catchment. Moreover, it allows us to verify the duration of the scheduled growing period or if the selected crop parametrizations meet the climatic conditions. The second step includes potential sources for plant stress, such as nutrient stress due to limited fertiliser inputs, or water stress due to limited water availability. An analysis of plant stress factors can show issues in quantities of scheduled operations, such as the amounts of fertiliser inputs, or the definition of irrigation schedules and decision rules. This section addresses the analysis with unconstrained simulated plant growth. Plant growth stresses are covered in the following section below. Plant stress can be activated/deactivated in the simulations by setting the parameter *nostress* in the file *'codes.bsn'* to 0/1/2 (see more Additional settings). For simulations which are analysed in the following section *nostress* is set to 1, meaning that all plant stress is turned off.

SWAT simulations employ the heat units concept (Barnard, 1948) to simulate the stages of plant development. Heat units (HUs) are units of degrees temperature which exceed a certain plant specific base temperature. The daily degrees above this threshold are accumulated over a growing period. A plant has a certain budget of HUs (potential heat units or PHUs) which must be collected in order to reach plant maturity. Whether a plant reaches maturity or not (and is e.g. ready for harvest) depends on plant specific properties (e.g. base temperature or PHUs), but also on meteorological model inputs (air temperature). The time series of the air temperature is site specific, but plant base temperature and PHUs can be for example specific to varieties of a crop.

For a plausible simulation of farm management and in consequence the simulation of variables such as ETa, nutrient cycles, or erosion protection through plant cover, the plants must develop appropriately during their growing period and must reach maturity. Plant maturity is expressed by different variables in a SWAT simulation, for example by the collected heat units of a plant at plant harvest, which is written into the file *'mgt_out.txt'* as *phu_plant*, or the temporal simulation of the leaf area index (LAI) of a crop which is written into the daily simulation outputs *'hru_pw_day.txt'*. The LAI is a good proxy for plant growth and reaches a maximum value and again starts to decrease (drying up of a mature plant) when a crop reaches maturity. Particularly for perennial crop land uses (e.g. forests, pastures) the temporal development of the plant biomass is a relevant variable to look at.

`SWATdoctR` provides two ways to investigate plant growth, where `plot_variable_at_harvkill()` function summarises the state of variables at the time of harvest/kill operations for all crops in a model setup and thus provides a general overview, while the function `plot_hru_pw_day()` allows detailed analyses of the daily time series of HRU related variables, which then can only be performed for a few HRUs of a model setup.

`plot_variable_at_harvkill()` uses the simulation outputs that are written into the file *'mgt_out.txt'* and extracts the values of variables that are written for harvest operations which are followed by a kill operation of that crop. It is important to mention here, that the SWAT+ user must define the harvest/kill operation of a crop as two separate operations harvest and kill in the management schedule to be able to read the variable states at the last harvest operation before the plant is killed.

Heat unit fractions As outlined above, the collected heat units indicate whether a plant reached maturity before it was harvested. A SWAT+ simulation writes heat unit fractions for each operation that was applied to a crop into the variable *'phuplant'* in the file *'mgt_out.txt'*. The heat unit fraction indicates what fraction of the PHUs of a crop were reached when a certain operation was triggered. At harvest a heat unit fraction of 1 must be exceeded and ideally the value is in a range of 1.1 to 1.5. By setting `variable = 'phu'` in `plot_variable_at_harvkill()` the heat unit fractions at the final harvest for all crops is plotted in a box plot.

```
plot_variable_at_harvkill(sim_nostress, variable = 'phu')
```

Figure 6.4 shows the resulting boxplot for the example SWAT+ model setup. The dashed line marks the value 1 for reference. Although this test case was considered a verified model setup, this analysis revealed a surprising model behaviour. The crops in this model simulation reached unusual large values for their heat unit fractions. Although the heat unit fractions should be above 1, values of 2 should not be exceeded, as this may also indicate issues in the plant growth simulation. In this case plants develop too fast. Such behaviour must be further investigated in the model setup. Apart from the unusually large values the crops *'csil'* (Silage corn) and *'rape'* (Winter rape) show heat unit fractions of lower than 1 at harvest for all harvests of these two crops. Therefore, the plant development and the defined growing seasons must be analysed and very likely be adjusted.

A second example from a different model setup shows a better balanced simulation of plant growth (Figure 6.4). In this example only 3 crops *'corn'* (Corn), *'pnut'* (Peanut), and *'cots'* (Cotton) were included in crop rotations in the implemented management schedules. Only cotton did not reach a heat unit fraction of 1 at harvest in some cases, whereas the other crops were in acceptable ranges for all harvest and kill operations.

Yields and biomass The crop yields and plant biomass are also good indicators to evaluate the plant development. With the simulation results where the plant growth was not constrained by stress factors the simulated yields and plant biomass could be compared to literature values of optimum yields of a certain crop. To summarize the plant yields at harvest the input argument `variable` is set to `variable = 'yield'` and for biomass to `variable = 'bioms'`. In the example below the unconstrained yields for the example model setup are plotted that resulted in the large implausible heat unit fraction values.

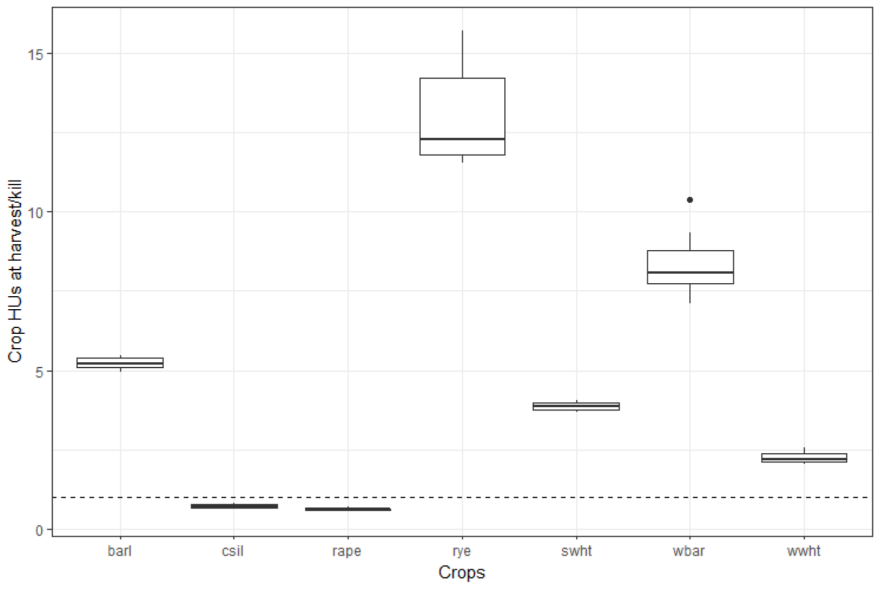


Figure 6.4: Example plot of the crop heat unit fractions at harvest plotted with the function `plot_variable_at_harvkill()`.

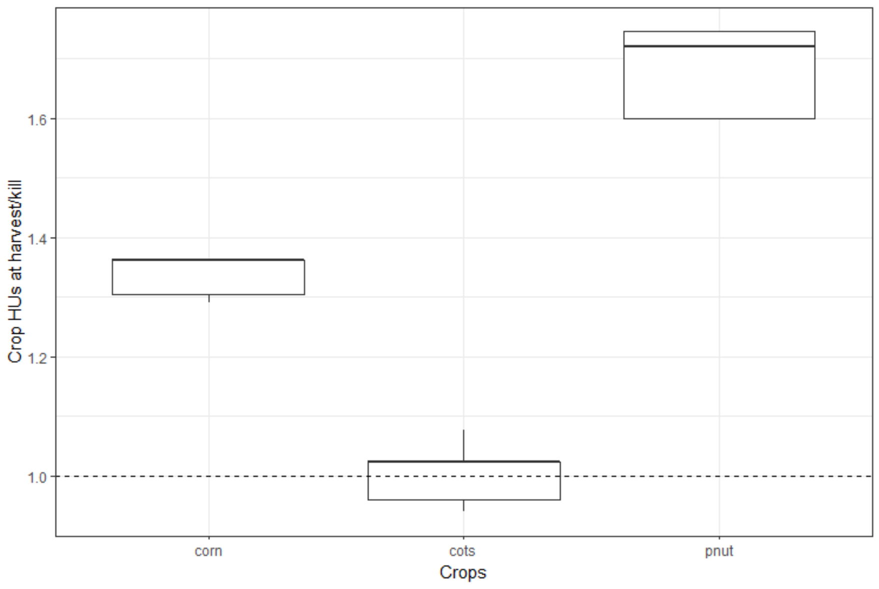


Figure 6.5: A second example plot of the crop heat unit fractions at harvest plotted with the function `plot_variable_at_harvkill()` and a different SWAT+ model setup.

```
plot_variable_at_harvkill(sim_nostress, variable = 'yield')
```

Figure 6.6 shows the resulting yield box plot for the different crops at harvest. The two crops *'csil'* and *'rape'* which showed heat unit fractions slightly lower than 1 result in rather plausible yields when no plant stresses are active. All other crops which showed implausible heat unit fractions also resulted in very low yields. Thus, these issues must be fixed before performing a model calibration for this model setup.

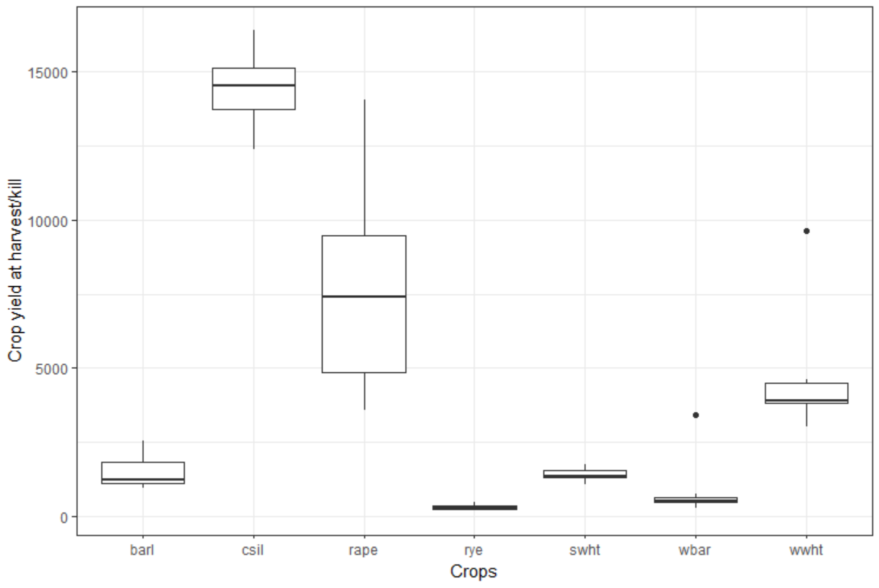


Figure 6.6: Example plot of the crop yields at harvest without simulated plant stress plotted with the function `plot_variable_at_harvkill()`.

Stress factors In the current simulation results all stresses should be deactivated and therefore the fourth option of `plot_variable_at_harvkill()` is just for checking if the settings were done correctly. By setting `variable = 'stress'` box plots for all stress factors and crops are plotted. In the example below all stress values are as expected 0, as the stress factors were deactivated for the simulation.

```
plot_variable_at_harvkill(sim_nostress, variable = 'stress')
```

Daily plant development When issues were identified in the heat unit fractions, yields or biomass of crops it can be useful to have a closer look at the daily development of plant specific variables. `SWATdoctR` provides the function `plot_hru_pw_day()` to plot variables that are written into the simulation output file *'hru_pw_day.txt'*, which saves the daily simulations of plant and weather output variables for all HRUs. To access the file after the model runs with `run_swat_verification()` the input argument `output` must include `output = 'pw'`. This option may have been not used in the simulation run as the output file *'hru_pw_day.txt'* can be rather large and may not fit in the computer's RAM. In this case plots of daily time series for variables at the HRU level are not possible.

For the example SWAT+ model setup it was possible to read the daily HRU outputs from *'hru_pw_day.txt'*. Based on the findings above it is worth having a look into the daily simulations

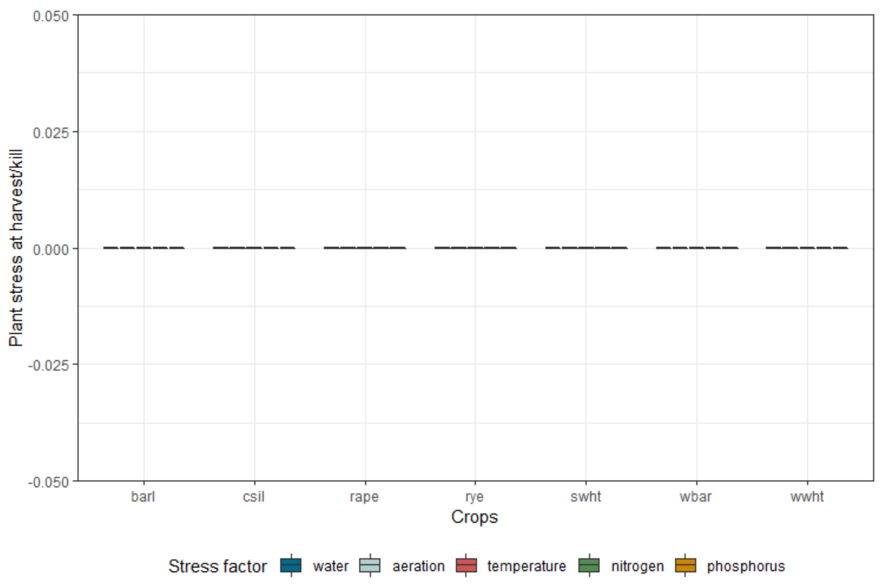


Figure 6.7: Example plot of the plant stress factors per crop for the simulations without simulated plant stress plotted with the function `plot_variable_at_harvkill()`. In this case this is just for checking the simulation settings and all stresses should be 0.

of LAI for some of the crops that showed an unusual behaviour. We again use the function `get_hru_id_by_attribute()` to identify HRUs which use the management `mgt = 'agrr_wwht'`. For HRUs with this land use (`hru_id = sample(hru_agr$id, 5)`) the variables *'lai'* and *'bioms'* are plotted (`var = c('lai', 'bioms')`) for the years 2001 to 2005 (`years = 2001:2005`).

```r
hru_agr <- get_hru_id_by_attribute(sim_nostress, mgt = 'agrr_wwht')

plot_hru_pw_day(sim_verify = sim_nostress,
                hru_id = sample(hru_agr$id, 5),
                var = c('lai', 'bioms'),
                years = 2001:2005)
```

Figure 6.8 shows the daily time series for the variables *'lai'* and *'bioms'* and the 5 selected HRUs. The temporal behaviour in the 5 selected HRUs is identical and therefore the lines overlap. From printing the management schedule with `print_triggered_mgt()` we know that *'wwht'* was harvested in 2001 and then the crop sequence continued with *'wbar'* » *'rape'* » *'wwht'* » *'wbar'* » *'rape'*. The pattern in the LAI and the biomass in Figure 6.8 show that *'wwht'* and *'rape'* develop slowly after planting in autumn, go dormant during the winter and continue to develop in spring until the crop is harvested (first, third, and fourth pattern in plot panels). For *'rape'* (year 2003) the LAI and biomass patterns additionally show that no clear plateau is reached and the plant is not fully mature at harvest. LAI for *'wbar'* in contrast to the other crops quickly peaks after planting and immediately drops back to a low value and the plant does not develop any further after that. Biomass stays low. These patterns are completely in line with the findings from the boxplots above.

The daily plots can also be used for land uses where no harvest/kill operations take place and therefore crops of these land uses do not appear in the plots with the function `plot_variable_at_harvkill()`. It can be however valuable to also check the development of land uses such as forest and grassland land uses. In the example below HRUs were identified which have a forest land use (`lum = 'frst_lum'`).

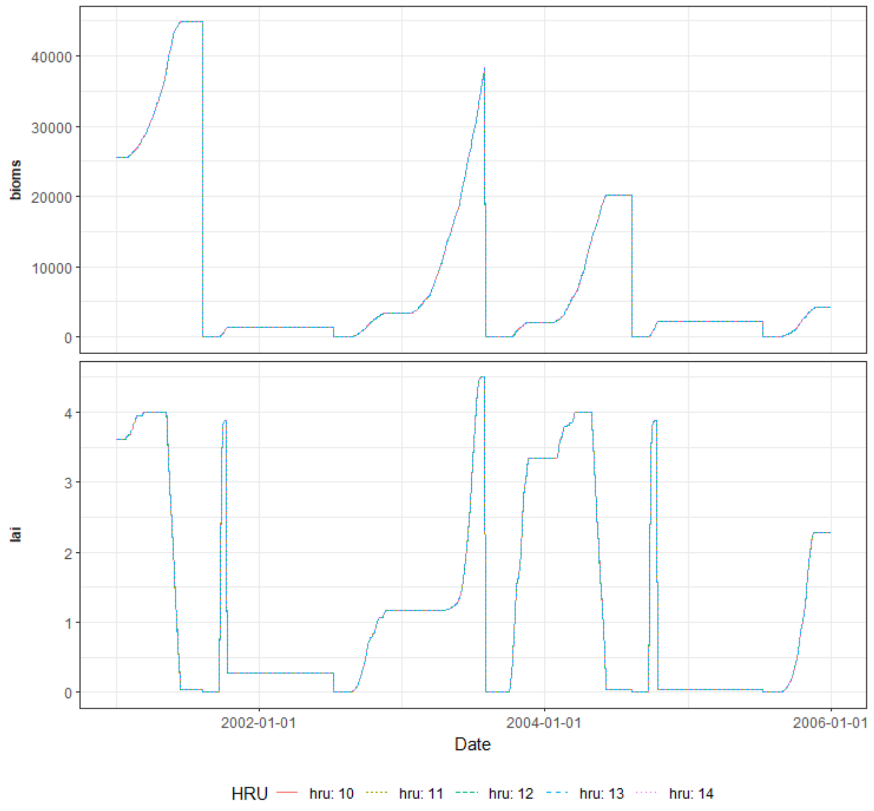


Figure 6.8: Example plot of the daily LAI and biomass development in the years 2001 to 2005 for 5 HRUs that implement the management schedule *'agrr_wwht'*.

For 5 randomly selected HRUs of those forest HRUs, LAI and biomass were plotted for a time period between 2001 and 2015.

```r
hru_agr <- get_hru_id_by_attribute(sim_nostress, lum = 'frst_lum')

plot_hru_pw_day(sim_verify = sim_nostress,
                hru_id = sample(hru_agr$id, 5),
                var = c('lai', 'bioms'),
                years = 2001:2015)
```

Figure 6.9 shows the development of LAI and biomass for the forest land uses. Forests show a very repetitive pattern for LAI, where during the summer months LAI increases to its maximum of 5 and drops in autumn. The biomass shows a continuous build up from its initialised value until it reaches an equilibrium state between biomass build up and decay, where the forest is considered to be a mature forest in the model simulations.

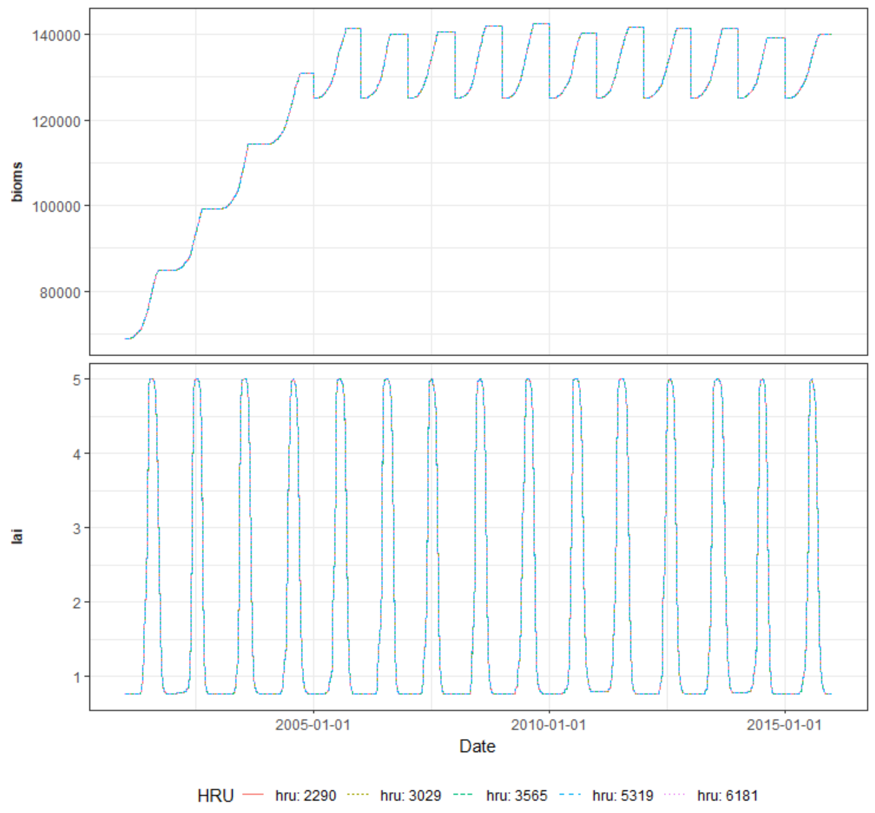


Figure 6.9: Example plot of the daily LAI and biomass development in the years 2001 to 2015 for 5 HRUs that have a forest land use (*'frst_lum'*).

6.2.2.5 Model simulations with plant stress active

The following analysis of specific management operations (fertiliser, irrigation, etc.) require simulations where plant stresses were active during the model run. Therefore, an additional model run with `run_swat_verification()` is necessary, but setting the input argument `nostress = 1` to activate all plant growth stress factors. At this point all above identified issues in the model setup must be fixed, before continuing with the model verification. This may require performing several iterations of model simulations with inactive/active plant growth stress factors.

Although the model simulations are performed now with all stress factors active, turning off the nutrient plant stress only can as well be a useful option for analyses (`nostress = 2`). This is particularly useful for eliminating the fertilisation impact on the plant growth and focusing only on the weather/climate and structural setting of the plant. After running the simulation with *nostress* set to 2, all the above-mentioned outputs can be analysed. Particularly, the aeration, temperature, and water stress, alongside yields are relevant outputs to be analysed. A simulation with inactive nutrient stress will provide a good approximation of possible yields with an optimal fertilisation and ideal plant nutrient supply. All other stresses will indicate the need of irrigation, drainage or plant-specific parameter adjustments for a plant to grow.

```
sim_stress <- run_swat_verification(project_path = model_path,
                                    outputs = c('wb', 'mgt', 'plt'),
                                    nostress = 0)
```

6.2.2.6 Step 4: Analysis of plant growth with plant stress

In a model simulation plant growth is often limited by the stress factors such as: water stress, aeration stress, temperature stress, nitrogen stress or phosphorus stress. If any or several of those stress factors are significant in the simulation of the crop development, the simulated biomass and yields can be strongly reduced. The five different stresses are printed as the variables *strsw*, *strsa*, *strstmp*, *strsn*, *strsp*, into the file *'<scale>_pw_<time>.txt'*. Additionally, these variables are written as the variables *var4* (*strsw*), *var5* (*strsa*), *var3* (*strstmp*), *var1* (*strsn*), and *var2* (*strsp*) for harvest operations in the *'mgt_out.txt'*, respectively.

The plotting of plant growth stress factors was used in the previous section simply for verification that all stress factors were deactivated. In this analysis plotting the stress factors for all crops can indicate reasons for impaired plant growth. The crop specific distributions of all stress variables at harvest can be analysed with the function `plot_variable_at_harvkill()` to identify any unusual large plant stress values. The plotted stress factors can provide guidance to further analyse the scheduled management, particularly the scheduled fertiliser inputs and scheduled irrigation operations or defined decision rules to trigger irrigation.

The example below (Figure 6.10) shows the 5 simulated plant growth stress factors for the planted and harvested crops in the example SWAT+ model setup. As in this small example the previously identified issues were not fixed, the analysis of the crops that showed the unusually high heat unit fractions is not really useful. The stress factors for those crops are rather low. But those plants also do not really develop in the simulations and therefore might not be limited by any of these stress factors. Only the crops with a rather acceptable plant development should be analyzed here, which are *'csil'*, *'rape'*, and *'wwht'*. Overall the three crops show increased temperature stress and also aeration stress. Particularly *'rape'* shows overall large values of temperature stress. This can indicate that the selected variety of *'rape'* does not have a parametrization that meets the regional conditions. Increased aeration stress seems plausible for the used model setup, as the simulated study site is a region with a large fraction of tile drained soils. The large values of aeration stress can indicate areas which should be tile drained.

```
plot_variable_at_harvkill(sim_stress, variable = 'stress')
```

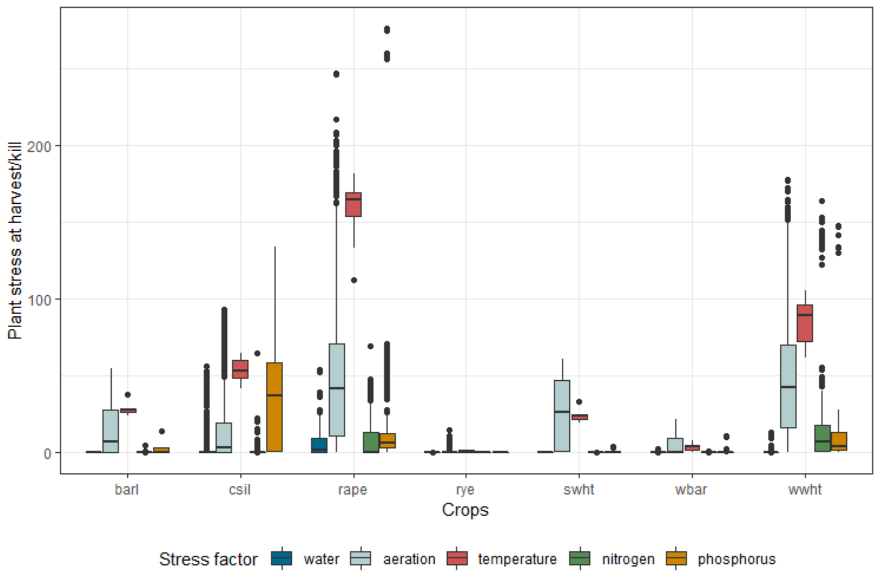


Figure 6.10: Plant stress factors per crop for the simulations with active plant stress plotted with the function `plot_variable_at_harvkill()`.

Only for *'csil'* slightly increased values of phosphorus stress were identified. If any unusual nitrogen or phosphorus stresses are identified, the fertiliser inputs for the respective crop may be revised and adjusted if the inputs are too low. A common issue here is that the fertiliser amounts were input incorrectly in the management schedule (confusion with units, fertiliser weight vs. N or P weight, etc.). Water stress is not an issue in the analysis of the example model setup. If substantial water stress would be identified, missing irrigation or drought periods that actually took place can be two potential explanations. It should be verified if irrigation is implemented in the case study and therefore has to be implemented in the model setup as well. If irrigation was implemented in the model setup the triggering of management operation should be verified in the simulated operations in *'mgt_out.txt'* (see section above).

Plotting the yields for the simulations with plant stress implemented (Figure 6.11) shows that the simulated stress factors impact the crop yields quite substantially. While the lowest yields for *'csil'* without stress factors were clearly above 13 tons, non of the yields simulated with active plant stress were above 10 tons and in some cases almost zero yield was simulated.

```
plot_variable_at_harvkill(sim_stress, 'yield')
```

As a summary the following full procedure can be performed to verify the appropriate functioning of plant growth:

- Make sure all the plant communities, even for a single plant, are initialised in the *'plants.ini'*, and all the plants are defined in the plants database *'plants.plt'*.
- Simulate the plant growth with all the stresses turned off (*nostress* is set to 1) and check if the plant is growing (LAI and biomass are increasing). This will show if the plant is actually set-up in a way that the model is simulating the growth cycle.
- Simulate the plant growth with only fertiliser stress off (*nostress* is set to 2) and check if the plant is growing (LAI and biomass are increasing). If the plant/crop is harvested, at this stage

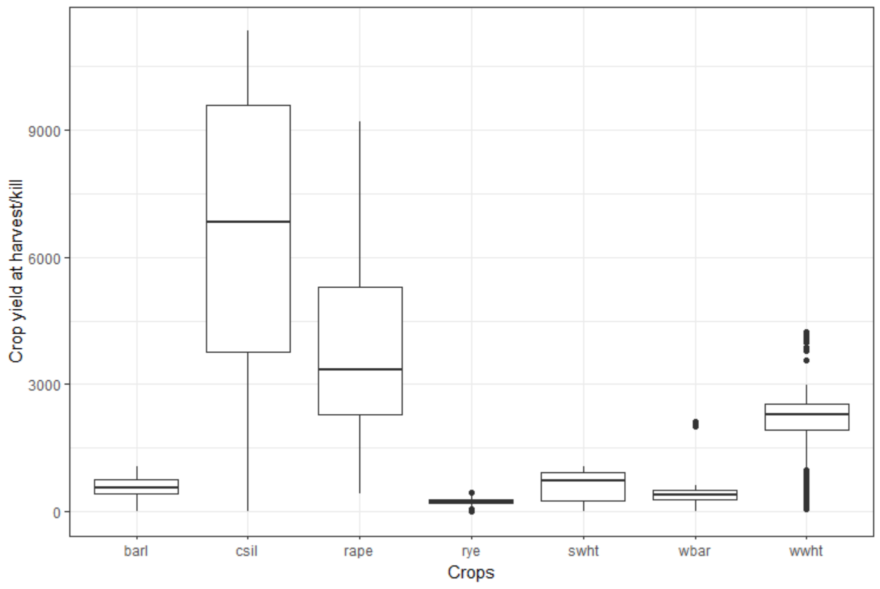


Figure 6.11: Yields for the example model setup per crop for the simulations with active plant stress.

the model should simulate the optimal yield for case-specific climatic conditions. Please note that in SWAT+ the yield is always given as dry weight.

- Simulate the plant growth with enabled stresses (*nostress* is set to 0). If unreasonable values for crop growth (LAI, biomass, yields, stresses) are produced, this is an indicator of possible set-up or parametrization errors in the model and should be investigated and fixed. No amount of later soft or hard calibration can fix errors if the process is simulated incorrectly.

6.2.2.7 Step 5: Simulation of point inputs

Point sources, such as waste water treatment plants or water transfers are defined with the files *'recall.rec'* and *'recall.con'* and corresponding time series records *'.rec'* files in a SWAT+ model setup (see more in the section on point sources). The point source time series inputs define the water, sediment, and nutrient loads which are emitted by a point source into a spatial object. Wrong units of the defined fluxes or wrong time intervals for a certain accumulated flux are common mistakes for point source inputs. Thus, it is good practice to verify the simulated influxes from point sources into the respective spatial objects.

So far no general procedure was implemented in the OPTAIN workflow to verify the point source inputs. Approaches will however be tested and implemented in later versions of `SWATdoctR`.

Flow from tile drainage systems into channels is defined in the file *'rout_unit.con'* by sending a certain flow fraction (*frac*) as tile flow (*hyd_typ* defined as *til*) to a channel (*obj_typ* defined as *sdc*) with the respective *obj_id*. Further, the defined landuse and management (*'landuse.lum'*) of a tile drained land object must point to the parametrization of a tile drainage network (parameter *tile* points to entry in *'tiledrain.str'*).

The verification of tile flow should mainly focus on whether tile flow occurs or not. The occurrence of tile flow can be verified with the output variable *qtile* for the respective land objects in the output file *'hru_wb_aa.txt'*. If no tile flow occurs for an HRU for which tile flow was parameterized the model inputs above have to be checked for any errors.

SWATdoctR provides only a very basic approach to analyse the tile flow from HRUs, by printing average annual tile flows in tabular form. The function `print_avannual_qtile()` selects all HRUs for which the landuse definition uses a tile flow parametrization (the variable *'tile'* in *'landuse.lum'*). These HRUs may also include land uses which are applied on drained soils, where however no tile flow will occur (e.g. urban land uses). With the input argument `exclude_lum` specific land uses can be excluded from the analysis. By default all urban land uses are excluded. Also excluding forest and grassland landuses will ease the analysis. In the example all urban, wetland, rangeland, and grassland landuses were excluded.

```
qtile <- print_avannual_qtile(sim_stress,
                              exclude_lum = c('urbn_lum', 'urbn_lud', 'utrn_lum',
                                              'utrn_lud', 'wetl_lud', 'wetl_lud',
                                              'rnge_lud', 'rnge_lum', 'fesc_lud'))
qtile


#> # A tibble: 1,494 × 5
#>       id qtile lu_mgt   mgt       soil
#>    <int> <dbl> <chr>    <chr>     <chr>
#>  1  4561  3.71 corn_lud agrr_csil HNd
#>  2  4421  8.17 corn_lud agrr_csil HNd
#>  3  5551 14.2  corn_lud agrr_csil HNd
#>  4  4993 15.2  corn_lud agrr_csil HNd
#>  5  4365 15.4  corn_lud agrr_csil HNd
#>  6  4318 19.7  corn_lud agrr_csil HNd
#>  7  2583 19.8  corn_lud agrr_csil HNd
#>  8  5131 19.8  corn_lud agrr_csil HNd
#>  9  4440 19.9  corn_lud agrr_csil HNd
#> 10  5362 20.1  corn_lud agrr_csil HNd
#> # ... with 1,484 more rows
#> # i Use `print(n = ...)` to see more rows
```

The printed table is automatically sorted, starting with the lowest simulated *'qtile'* values. Although the smallest average annual *'qtile'* sums are rather low, tile flow occurs on all tile drained agricultural areas. A quick summary shows that tile flow differs by two orders of magnitude in the example. This must be considered in the calibration, as tile flow can have significant shares.

```
summary(qtile$qtile)

#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>   3.709 154.594 198.518 185.672 214.422 311.408
```

6.2.3 Soft calibration

Model calibration using soft data is called soft calibration. The new SWAT+ model has built-in soft calibration modules to perform model parameter adjustments for water budget and crop yield calibration in a semi-automated way.

6.2.3.1 Water balance soft calibration

There are two options available for Water Balance (WB) Soft Calibration (SC): Adjust the water yield and baseflow ratios (uses the "y" flag in the *'codes.sft'* file under the first column LANDSCAPE_YN); Adjust all the individual water balance components (uses the "a" flag in the *'codes.sft'* file under the first column LANDSCAPE_YN). In most cases only some of the water balance components are known or can be derived. The first method of upland water balance SC was successfully applied in a large-scale calibration study across the contiguous US (White et al. (2022)). In that study the use of SC procedure reduced the prediction error of streamflow at gages by varying degrees and is highly recommended by the SWAT+ development team. The in-built SC procedure can be enabled manually (as of November 2022) and is not available in the SWAT-Editor. Once enabled, SWAT+ will run several iterations, based on your criteria and the current model performance. With each iteration, the model will make adjustments to a parameter and check the performance of the model after those adjustments. Steps, which SWAT+ performs during the SC procedure are shown in Table 6.1:

Table 6.1: Adjustments, which SWAT+ performs during the water balance soft calibration procedure.

Process	Adjusted parameter	Range	Explanation
Evapotranspiration	*esco*	within +/- 1%	adjusts *esco* to calibrate water yield ratio
Potential Evapotranspiration	*harg_pet*	within +/- 1%	adjust *harg_pet* to calibrate water yield ratio
Surface runoff	*cn3_swf*	within +/- 2%	adjust *cn3_swf* to calibrate surface runoff ratio
Lateral flow	*latq_co*	within +/- 10%	adjust *latq_co* to calibrate lateral soil flow ratio
Percolation	*perco*	within +/- 5%	adjust *perco* to calibrate baseflow ratio. Note that tiled hrus don't allow *perco* to change, the values are fixed at 0.1.

To initiate the soft calibration procedure of your SWAT+ model, the following steps must be performed:

1. Check if the *.sft* files are present in your model setup directory.

- If not present, download and add the *.sft* files to your SWAT+ model project folder. Check the SWAT+ source repository for the files. For the WB soft calibration you will need the *'codes.sft'*, *'wb_parms.sft'*, and the *'water_balance.sft'* files.
- If present, continue to step 2.

2. Manually edit your *'file.cio'* and add *.sft* file names, if not present. The files should be added to line 22 (CHG), columns 4-6. The *'file.cio'* is a free-format. The files should be listed in the order: Column 4: *'codes.sft'* Column 5: *'wb_parms.sft'* Column 6: *'water_balance.sft'*

Save the changes to the *'file.cio'*.

1. Edit *'codes.sft'* file by changing the "n" to "y" in the first column. Save the changes.
2. Modify *'water_balance.sft'* with your values for fractions. Only the values under the WYR and BFR columns need to be modified (see Soft data). Make sure to count the columns, as the file is a free-format. Save the edits.
3. Modify *'wb_parms.sft'* file if needed, although the default values are a good starting point. Save the edits if they were performed. At this point, the model is set-up to: i) include your *.sft* files in the simulation, ii) calculate the WB fractions, iii) make the adjustments based on your defined or default criteria.
4. Execute *'swat.exe'*. This process will take a while, because several iterations are performed.
5. Immediately after the execution save the *'hydrology.hyd'* file as a backup.
6. Now, you can inspect the results. Results from each iteration will be saved in the output file *'basin_wb_aa.txt'*, where you can track the changes in the WB components after each iteration.
7. Your new parameters for the HRUs are stored in *'hydrology_cal.hyd'* file. If you decide to use them, rename *'hydrology_cal.hyd'* to *'hydrology.hyd'*. At this point the WB soft calibration procedure is complete.
8. To stop the soft calibration procedure change "y" to "n" in *'codes.sft'*. OPTIONAL: Replace the *.sft* to *null* in the *'file.cio'*.
9. Now the model can be run normally.

If necessary, the WB SC procedure can be performed several times.

Advice: a good practice is to compare the flow duration curves of your initial (uncalibrated) model and the SC model with the observed one. Although flow duration curves are derived from the observed streamflow (hard) data, such analysis at this stage will indicate how much your upland processes affect the streamflow.

6.2.3.2 Crop yield soft calibration

As with the WB SC, the crop yield SC procedure will run the model several times, each time performing adjustments to one variable in the following order: 1) *epco*, 2) *pest_stress*, 3) *lai_pot*, and 4) *hi_pot*. When set up properly, this algorithm will aim to decrease the error between the observed and simulated yields. There is a different number of iterations for each variable. Since *epco* is highly non-linear, the SWAT+ soft calibration procedure will perform a maximum of 5 adjustments. *Pest_stress* is linear so only one iteration will be performed, and *lai_pot* and *hi_pot* adjustment will run for a maximum of 3 iterations. The model will stop iterating when the mean yields are within 3% difference with the observed. The first iteration uses the initial (default) values. The initial change applied to each parameter is a function of the percent difference between the simulated and observed yields. After the initial change, the algorithm uses linear interpolation in subsequent iterations (See table below 6.2).

Table 6.2: Soft calibration parameter sequence adjustment.

Sequence	Parameter	Change Type	Initial change	Number of linear interpolations
1	*epco*	absolute value	if (diff_{pct} >= 10%) chg_init = -0.01 * diff_{pct} + 0.06; if (diff_{pct} < 10%) chg_init = 1.0	4
2	*pest_stress*	absolute value	diff_{pct}	0
3	*lai_pot*	absolute change	0.5 * diff_{pct}	2
4	*hi_pot*	absolute change	0.005 * diff_{pct}	2

The crop yield SC iteration amount and the degree of change will depend on the initial difference between the observed and simulated yields.

The crop yield soft calibration procedure is set up in a similar way, as the WB soft calibration.

1. Initially, add the necessary *.sft* files to your project directory.

- If not present, download and add the .sft files to your SWAT+ model project folder. Check the SWAT+ source repository for the files. For the WB SC you will need the *'codes.sft'*, *'plant_gro.sft'*, and the *'plant_parms.sft'* files.
- If the files are already present, continue to step 2.

2. Manually edit your *'file.cio'* and add *.sft* file names, if not present. The files should be added to line 22 (CHG), columns 4, 9-10. The *'file.cio'* is a free-format. The files should be listed in the order: Column 4: *'codes.sft'* Column 9: *'plant_parms.sft'* Column 10: *'plant_gro.sft'*

Save the changes to the *'file.cio'*.

3. Edit *'codes.sft'* by adding "y" to the third column, under the PLNT_YN. Save the changes.
4. Modify *'plant_gro.sft'* with your values of target yields for each crop (see Soft data). The target is the average annual yield of dry weight crop for the entire basin (model). More than one crop type with yields could be added here. Save the edits.
5. Modify *'plant_parms.sft'* file if needed, although the default values are a good starting point. Save the edits if they were performed.

At this point, the model is set-up to: i) include your *.sft* files in the simulation, ii) calculate the crop yield differences between the simulated and observed, iii) make the adjustments based on your defined or default criteria.

6. Execute *'swat.exe'*. This process will take several iterations to complete.
7. Immediately after the execution save the *'hydrology.hyd'* file as a backup.
8. Now, you can inspect the results. Results from each iteration will be saved in the output file *'basin_crop_yld_aa.txt'*, where you can track the changes in the crop yields after each iteration.
9. Your new parameters for the HRUs are stored in *'hydrology_cal.hyd'* file. If you decide to use them, rename *'hydrology_cal.hyd'* to *'hydrology.hyd'*.

10. Moreover, SWAT+ generates the *'plant_parms.cal'* file, which will print the fitted values, the applied change, the minimum and maximum tried values for each parameter for each crop type. You can use these values to update your *'plants.plt'* file and *epco* for your agricultural HRUs. Although your HRUs are already updated with the new *'hydrology.hyd'* file, if you use the new generated one, just update the plant database. At this point the crop SC procedure is complete.
11. To stop the soft calibration procedure change "y" to "n" in *'codes.sft'*. OPTIONAL: Replace the *.sft* to *null* in the *'file.cio'*.
12. Now the model can be run normally.

This procedure can be repeated several times, if necessary.

Advice: since the crop soft calibration procedure will impact the upland water balance components, it is recommended to revise the water balance. If necessary, repeat the WB SC.

6.2.4 Hard calibration

As briefly mentioned in section 6.2.3, in OPTAIN we differentiate between soft and hard calibration steps for the SWAT+ model setups. The soft calibration step provides estimates for the parameters *esco*, *perco*, *latq_co*, and *cn3_swf* to fit the overall water balance and fits average annual crop yields by adjusting the parameters *pest_stress*, *lai_pot*, *harv_idx*, and *epco*. Typically the parameter adjustments in the soft calibration already result in well balanced model setups. The hard calibration uses the soft calibrated model setup as a starting point to fit the simulations of in-stream discharge, sediment concentrations, or nutrient concentrations such as total phosphorus or nitrate-nitrogen to observed data. Various approaches for hard calibration exist in the SWAT literature which are dependent on the availability of observation data and the purpose of the calibrated model setup.

6.2.4.1 Criteria for hard calibration

Calibration and validation design Model calibration usually includes a validation of the calibrated model setup. The proposed validation procedure for OPTAIN case studies will be covered with more detail in the section 6.3. For the calibration and validation of a model setup the observation data which were collected for the hard calibration (see section 6.1) are split into two sets of data. The most common procedure is to split available time series data into separate time intervals where the calibration is performed for one set of data and the model is validated for different time intervals. The separation should account for comparable climatic conditions and climate variability in both data sets so that both the calibration and the validation periods include dry and wet periods (Arnold et al., 2012c).

In all OPTAIN SWAT+ CSs several variables will be of interest to be investigated in scenario analyses, which are discharge and other water balance components, sediment transport related variables, or different variables of the phosphorus and nitrogen cycles. In the best case, observation data for all variables of interest should be available for both the calibration and validation periods. In such a case the splitting of the available observation data into calibration and validation periods must consider the data availability of all considered variables together with the climatic variability of the selected calibration and validation periods.

Another approach is to split the available hard calibration data spatially into a calibration and a validation data set. The model calibration would then be performed for example for one or several gauged locations in the stream network and would be validated at other locations where gauge data is available. This approach will be further addressed below in the section on multi-site calibration.

Single/Multi-variable calibration In many cases, and this also applies to the SWAT+ model setups in OPTAIN, model setups should be capable simulating multiple variables such as discharge, sediment yields, or nutrient loads. In a review article Arnold et al. (2012c) suggest calibrating a model setup sequentially for the different target variables following for example the calibration protocol proposed by Engel et al. (2007). This sequential calibration approach is frequently found in the literature (e.g. Piniewski et al. (2019), Mehdi et al. (2018), Wallace et al. (2018), Malagó et al. (2017), Bieger et al. (2014)). Fohrer et al. (2022), in contrast, recently proposed a guideline for water quality modelling where they suggest performing a joint multi-metric calibration of discharge and water quality variables to find a better compromise in the simulation of discharge and water quality. Joint calibration of multiple variables can be particularly found for SWAT modelling studies which employ automated calibration strategies (e.g. Schürz et al. (2019), Haas et al. (2016)).

Although the argument of better balanced calibration through a joint multi-variable is valid, we propose to perform a sequential calibration approach for the hard calibration procedure in OPTAIN. In most cases continuous time series for observed in-stream discharge is available, while observation data for other variables of interest is very limited (e.g. monthly, bi-weekly grab samples, or only a few data points from a sampling campaign). Thus, stream flow data are in many cases the most reliable data which are available to tune a SWAT model setup and the informative value of other data is limited. OPTAIN aims to provide a harmonised workflow for hard calibration in all CSs. A sequential approach which performs a thorough hard calibration for stream flow and in the following employs an approach which best meets the available data of e.g. sediment loads or nutrient concentrations provides the highest flexibility in a common calibration procedure. As illustrated in Figure 6.1, the calibration procedure is split into three main sequences, i) a process based calibration of the catchment hydrology, ii) followed by a calibration of sediment transport (if sediment transport is a relevant target variable), iii) and a calibration of the phosphorus and nitrogen cycles.

Single/Multi-site calibration In well monitored and/or large study regions observation data is often available in multiple monitoring locations which enables it to perform a multi-site calibration procedure. Similar to the calibration for multiple variables a multi-site calibration can be performed simultaneously or sequential starting from the head watersheds progressing to the catchment outlet (Leta et al., 2017). Another potential use case of multi-site observation data in model calibration is to calibrate a model setup implementing the observations from one set of sites and validating the model performance in other locations where observation data is available (e.g. Piniewski et al. (2017)).

The OPTAIN CSs are small to medium size watersheds where in most cases only observation data at one location (which is usually the catchment outlet) is available for model calibration. Thus in the majority of cases considerations on different multi-site calibration approaches are not relevant. Yet, in some CSs observations are available for at least two locations. Depending on the data availability in the different locations the different potential approaches for multi-site calibration will be investigated and a common strategy for OPTAIN will be developed.

Model performance evaluation and signature measures Hydrological model calibration typically employs performance metrics (in many cases the Nash Sutcliffe Efficiency (NSE) described in Nash and Sutcliffe (1970)) to evaluate the performance of model simulations with adjusted model parameter values. While a large part of the SWAT modelling literature use only one or a few performance metrics to evaluate a model setup, there is a clear suggestion to use multiple criteria which evaluate different characteristics of model simulations at the same time (e.g. Guse et al. (2020), Schürz et al. (2019), Haas et al. (2016), Pfannerstill et al. (2014), Efstratiadis and Koutsoyiannis (2010)).

Signature measures can describe specific characteristics of the discharge (McMillan, 2021) and can therefore be a link between a hydrological process that contributed to the runoff and the specific characteristic of the discharge time series Shafii and Tolson (2015). The analysis of multiple signatures can improve the process representation and result in a better balanced simulation of e.g. the catchment hydrograph Euser et al. (2013).

A comprehensive collection and analysis of performance metrics and signature measures can be found for example in McMillan (2021) and McMillan et al. (2022). The aim in OPTAIN is to evaluate a wide range of characteristics of the simulated time series of output variables when compared to available observation data. Thus, particularly for in-stream discharge for which observation data will mostly be available as continuous daily records the evaluation of the simulated discharge time series must include a wide range signature measures which are evaluated with performance metrics to achieve a good process representation of the hydrology in the OPTAIN case study catchments.

The selection of signature measures and performance metrics is an ongoing process and will be defined and revised during the hard calibration. Successful implementations in SWAT case studies will be evaluated and our selection will be based on measures and metrics which were found to be relevant in the model calibration in other case studies (e.g. Alemayehu et al. (2022), Fernandez-Palomino et al. (2021), Guse et al. (2020), Haas et al. (2016), Pfannerstill et al. (2014)).

6.2.4.2 Calibration of hydrological processes

Parameter identifiability is a common issue in the calibration of complex models such as SWAT+ where usually large sets of parameters are tuned in the calibration process Efstratiadis and Koutsoyiannis (2010). Several authors (e.g. Guse et al. (2020) or Efstratiadis and Koutsoyiannis (2010)) propose multi-criteria calibration procedures to improve the identifiability of acceptable parameter value ranges for the calibrated model parameters. Other approaches perform separate model calibrations for sections of the hydrograph which are dominated by different processes. Zhang et al. (2011) for example implemented a base flow separation for the observed hydrograph and performed separate calibrations for the base flow and fast runoff dominated sections of the hydrograph.

Most SWAT+ model parameters can be associated to single hydrological processes in the model structure. In many cases changes in model parameters only have a direct impact on a single simulated process and only indirectly impact other processes. Based on identifiable process-parameter relationships we plan to test several approaches in OPTAIN to determine a generalised workflow which can be implemented in all case studies. All potential approaches will include sets of signature measure/performance criteria combinations that are found to be effective proxies for runoff components. One approach to include the defined criteria in the calibration procedure is to adjust all hydrological model parameters at the same time while evaluating all defined criteria simultaneously. By performing several iterations in the model calibration, the parameter ranges will be progressively constrained to identify well performing parameter combinations. This is the most common procedure found in multi-signature/multi-criteria calibration procedures (e.g. Alemayehu et al. (2022), Fernandez-Palomino et al. (2021), Guse et al. (2020), Haas et al. (2016), Pfannerstill et al. (2014)).

A second approach would be to perform a process oriented sequential calibration. In a sequence process related signature/metric sets and functional parameter groups for the same process will be defined and a calibration for fast runoff, lateral runoff, and base flow will be performed in a consecutive sequence. Fast runoff is dominantly controlled by the parameters *surlag*, *cn2*, and *cn3_swf*. The parameter *surlag* can impact the timing of and the recession of fast runoff. Magnitudes of discharge peaks can be adjusted by varying the parameters *cn2* and *cn3_swf*. Increases in the Curve Number parameters result in an increase of immediate runoff, while a decrease leads to more infiltration of water which is then available for the other processes lateral runoff, or an infiltration to the aquifer. Lateral flow is associated with the soil parameters available water capacity *awc*, saturated hydraulic conductivity *k*, and the dry bulk density *bd* which control the water budget that is available for lateral flow, and the lateral travel time *lat_ttime* and lateral slope length *lat_len* which control the lateral transport of water and thus the timing of lateral flow. The water budget that is available for groundwater processes and base flow is mostly controlled by the percolation parameter *perco*. *perco* was already adjusted in the soft calibration step to meet the base flow ratio of the catchment. Therefore, if *perco* is included in the hard calibration as well, the value range of *perco* must not deviate too much from the suggested value of *perco* that resulted from the soft calibration in order to maintain the aimed base flow ratio. Functional base flow parameters are the base flow recession constant *alpha*, the minimum groundwater level at which flow from the aquifer occurs *flo_min*, the revaporation constant *revap_co*, and the minimum groundwater level at which revaporation occurs *revap_min*.

The sequential process based calibration procedure would introduce additional iterations for calibration and may result in a more complex procedure than just calibrating all hydrological parameters at the same time. Yet, a reduced parameter space in the calibration of each process associated group due to a lower number of parameters in each step will substantially reduce the dimensionality issue and therefore can enhance the parameter identifiability. Eventually both proposed approaches have advantages and trade-offs. Tests of both approaches in the hard calibration step will show how the calibration procedure will be implemented by all case studies. Approaches which were found to be robust to be implemented in a harmonised way in OPTAIN will be implemented in `R` functions and `R` script modelling workflows.

6.2.4.3 Calibration of sediment transport

Sediment transport is mostly driven by large surface runoff events. Thus a good model performance with respect to sediment transport processes requires a hydrologically well calibrated model setup where the magnitude and timing of large discharge peaks are met. Automated indirect methods exist to acquire continuous time series of sediment concentrations, by e.g. recording the in-stream turbidity. Nevertheless, in most situations sediment load and concentration data is limited and only grab sample data or accumulated sediment budgets are available.

Consequently, rather simple approaches will be implemented for the calibration of sediment transport. Although the sediment calibration is part of the hard calibration procedure (as it requires a good performance for discharge simulations), depending on the data availability the possible approach to be implemented has more the character of a soft calibration.

A pragmatic approach for sediment (and nutrient) calibration was outlined in a calibration protocol by Engel et al. (2007). The protocol was established for previous versions of the SWAT model, but can be translated to SWAT+. Engel et al. (2007) suggest to adjust the USLE practice factor *usle_p*, the USLE cover factor *usle_c*, the coefficient for sediment routing *spcon* and the channel erodibility factor *ch_erod* to minimise the percent bias between observed and simulated sediment concentrations/loads and to maximise the correlation between observed and simulated sediment concentrations/loads.

6.2.4.4 Calibration of phosphorus and nitrogen concentrations

The same arguments in terms of observation data which were stated for sediment loads and concentrations apply to observation data for phosphorus and nitrogen concentration data. Typically, monthly or bi-weekly grab samples are available, where the assumption is made that this grab sample is representative for the in-stream nutrient concentration on that date. Thus, the calibration approach for nutrient concentrations will be similar to the one for sediment transport.

To improve the calibration of phosphorus concentrations Engel et al. (2007) and Wallace et al. (2018) for example propose to include the phosphorus percolation coefficient *pperco*, the phosphorus soil partitioning coefficient *phoskd*, the phosphorus sorption coefficient *psp*, or the phosphorus uptake distribution factor *p_updis* in the calibration.

For the simulation of the nitrogen cycles Haas et al. (2015) and Wallace et al. (2018) for example propose to include the nitrogen percolation coefficient *nperco*, the nitrogen uptake distribution factor *n_updis*, the denitrification exponential rate coefficient *cdn*, the denitrification threshold water content *sdnco*, or the rate factor for humus mineralization of active organic nitrogen *cmn* in the calibration.

6.3 Validation

Trust and confidence are critical to the success of adapting the results of environmental models. Models that are setup in SWAT+, considered as a good environmental modeling tool, are only as good as the data and assumptions that go into them, and they can be affected by various sources of error and uncertainty. For this reason, it is important to validate the models to ensure that they are producing accurate and reliable results. There are several ways to validate the SWAT+ model which are possible to use within OPTAIN. Individual CSs can use either of the approaches, or a combination of several. In all cases the validation procedure comes down to comparing the model's predictions to actual observations of the environment. This can be done by comparing the model's output to data from sensors, measurements, or other sources of real-world data. If the model's predictions are consistently close to the observations, it is considered to be a good representation of the environment. Several papers have been published (Arnold et al., 2012b; Moriasi et al., 2015) and are considered as a sufficient guidance on how to perform the validation of earlier SWAT model. Same approaches are valid for the new version of the model - SWAT+.

Recommended workflow for a typical validation:

- Consider and prepare the available observation data in your specific CSs (i.e. flow timeseries from 2010 to 2020). Ensure that the model simulates the entire time window.
- Divide your data into calibration and validation periods. It is wise to divide the available periods in such a way, that both calibration and validation timeseries contain various environmental conditions, i.e. "dry" and "wet" years. The periods do not have to be consecutive. For our example, the calibration period might be 2012-2017, and validation 2010-2011 and 2018-2020.
- Extract and compare the model output (*flo_out* in this example) with the flow timeseries for the validation period.
- Use the same methods and statistical indicators as for hard calibration to compute the chosen performance criteria.
- Based on the result for the chosen criteria, the model is considered to be reliable or not. Refer to the 6.2.4 chapter and the above-mentioned publications for guidance on the criteria.
- Repeat the procedure for other available hard data and/or other location in the model.

Workflow for other validation methods:

In some cases, the available timeseries is not sufficient to cover both model calibration and validation. In such cases alternative validation methods can be chosen, i.e. to validate the model at a different point/outlet (i.e. upstream or downstream). In this case, the procedure would remain the same as in the typical workflow, only the source data/timestep and the extraction point would change. - Consider and prepare the available observation data in your specific CSs (i.e. flow timeseries from both locations for the entire available period). Ensure that the model simulates the entire time window. - Extract and compare the model output (*flo_out* in this example) with the flow timeseries for the validation period at the validation point, which is different than the one used for calibration. - Note, that in such a case, the time window can overlap, meaning that the same or partially overlapping time period may be used for validation and calibration. - Use the same methods and statistical indicators as for hard calibration to compute the chosen performance criteria. - Repeat the procedure for other available hard data and/or other location in the model.

Ideally, calibration and validation should be process and spatially based, while considering input, model, and parameter uncertainties. There are many more model validation variations (i.e. validation with data outside the modelled boundary, validation with soft data, expert-based validation), which all have merits and downturns. Overall, the validation of environmental models is important because it helps to ensure that they are producing reliable and accurate results. This is essential for making informed decisions about environmental management and policy, as well as for understanding the impacts of human activities on the environment, which is part of the OPTAIN projects' goals.

Chapter 7

Scenario setup

Calibrated and validated model setups can be used for scenario runs. One of the main goals of WP4 in OPTAIN is to evaluate the effectiveness of various NSWRMs under both current and future climate. To meet this objective, the following workflow of scenario runs is proposed: (1) climate change runs (section 7.1); (2) NSWRM scenario runs (section 7.2); (3) combined climate change & NSWRM scenario runs (section 7.3). The first set of runs will allow to assess projected changes in water balance, sediment and nutrient budgets in two future horizons relative to the baseline period, under different emission scenarios and using ensembles of climate models to represent uncertainty. The results from the second set of runs will illustrate the effectiveness of the maximum implementation of selected NSWRMs under current climate. Finally, the third set of runs will lead to the assessment of NSWRM effectiveness under future climate. This chapter discusses the background for climate scenarios, the assumptions and settings of particular model runs, as well as the general workflow and output naming conventions.

7.1 Climate change effects

7.1.1 Climate forcing

One of the goals of SWAT+ modelling in OPTAIN is assessing the effect of climate change and NSWRMs on water balance, sediment and nutrient fluxes in the case study catchments. To this end, WP3 delivered a climate scenario dataset, stored on ZENODO that will be used as a forcing in calibrated SWAT+ models. The methodology behind is fully described in the OPTAIN deliverable D3.1 “Climate scenarios for integrated modelling” (Honzak and Pogačar, 2022). Here we provide a short overview of the main features that are important from a hydrological modelling point of view.

To address uncertainty in climate projections, a multi-model ensemble approach was applied (Rathjens et al., 2016; Singh, 2016), with multiple emission scenarios and climate models. More specifically, a common climate database - Regional Climate Model (RCM) simulations from the European branch of the Coordinated Regional Climate Downscaling project (EURO-CORDEX)[1] and the Representative Concentration Pathway (RCP) scenarios 2.6, 4.5 and 8.5 were used. The advantage of using CORDEX is its detailed resolution of 0.11 degrees or ~12.5 km. Raw General Circulation Model (GCM) outputs would be incompatible with the small scale of OPTAIN catchments.

The EURO-CORDEX repository provides dozens of RCM simulations, but only some of them contain all variables required by SWAT+ with a daily time step (see Weather data). A final selection contained

[1] Official website https://www.euro-cordex.net/

six simulations, being combinations of three different driving GCMs and five different RCMs (Table 7.1) under each RCP.

Table 7.1: List of RCM simulations selected for SWAT+ model runs in OPTAIN.

Model number	Driving model (GCM)	Ensemble	RCM	End date	Model code
1	EC-EARTH	r12i1p1	CCLM4-8-17	31.12.2100	earthcclm
2	EC-EARTH	r3i1p1	HIRHAM5	31.12.2100	earthhirh
3	HadGEM2-ES	r1i1p1	HIRHAM5	30.12.2099	hadghirh
4	HadGEM2-ES	r1i1p1	RACMO22E	30.12.2099	hadgracmo
5	HadGEM2-ES	r1i1p1	RCA4	30.12.2099	hadgrca
6	MPI-ESM-LR	r2i1p1	REMO2009	31.12.2100	mpiremo

In addition, raw climate model outputs have systematic errors compared to observational data (Sunyer et al., 2015), which translate into even larger biases when processed through impact models (hydrological models in particular). Thus, a popular bias correction method, called quantile mapping, was applied to address this issue. The reference dataset for bias correction was ERA5-Land at 0.1 degree resolution.

The final product consisting of six bias-corrected, daily RCM simulations for three RCPs, covering the period 1981 - 2099/2100 is available in netCDF format on 0.1 degree grid for all case study catchments. In addition, ERA5-Land reanalysis data covering the period 1981-2021 are also available, with the same resolution and time step. The data for each case study are prepared in rectangular domains covering the catchment boundaries.

7.1.2 Climate change runs

Climate change runs in OPTAIN using forcing data described above should be made in a consistent manner to enable fair cross-comparisons of results between case studies. Although it seems straightforward to read new weather data into the model setup and run the model, there are actually more things to consider in order to ensure full consistency. The proposed procedure of climate change runs in OPTAIN includes the following aspects:

1. Using one reference and two future time slots are recommended. In OPTAIN, the reference time slice is 1991-2020. The “near future” and “far future” time slots are 2031-2060, and 2071-2100, respectively. Priority should be given to “near future” runs.

2. A script to convert forcing data from the netCDF to SWAT+ format will be provided. Simulations have to include the warm-up period (3 years are suggested) so the total length of weather time series is 33 years. The total number of runs to make for a given time slice is 18 (6 RCMs times 3 RCPs). All runs for the baseline period and two future time slices equal 54.

3. A script to generate weather generator (*.wgn*) files for each individual climate simulation input will have to be run (see Weather Generator)

4. SWATFarmR has to be rerun for each set of weather data files (i.e. *.tmp* and *.pcp* files) representing a given climate simulation, considering modified time windows for agricultural practices in the warmer climate. In result new *‘management.sch’* file will be generated, specific for each individual run.

5. In order to account for the physiological effect of elevated atmospheric CO_2 on actual ET and plant growth in SWAT+ when using the Penman-Monteith PET method (Gunn et al., 2021), default values of atmospheric CO_2 concentrations should be modified for each RCP and time period. As of November 2022, SWAT+ does not allow to take into account dynamic (annual) changes in CO_2, so average values for the entire simulation period should be used. RCP-specific annual time series of CO_2 concentrations are available at the RCP Database portal.

6. If annual, monthly or daily point sources inputs are used (all options other than constant), then the input files should be adjusted to match the simulation time period of each run (under assumption of no future changes in loadings).

7. If monthly or annual atmospheric deposition time series are used (all options other than constant) then the input files should be adjusted to match the simulation time period of each run (under assumption of no future changes in atmospheric deposition).

8. If decision tables are used for whichever purpose in the model setup, it should be verified if the way they are designed will also be valid under future climate.

9. After all these things have been taken care of, the SWAT+ run can be made with desired output printout settings. Recommendations on the "minimum" options for print settings will be provided. It should be kept in mind that a massive amount of data may be generated for all the runs, especially if options for daily time step output and HRU-level outputs are used.

10. A minimum common set of environmental performance indicators (e.g. changes in low or high flows, soil water retention, nitrate load, crop yields, etc.) based on OPTAINs deliverable D2.2 ("Tailored environmental and socio-economic performance indicators for selected measures") will be provided after consultation with WP2. These indicators will have to be calculated and delivered in a pre-defined format. The climate change effect will be illustrated by comparing the boxplots for each time horizon. These results will also be used during the local meetings with OPTAIN MARG.

11. It is recommended that input and output files of each climate change run are archived in a repository.

Due to the massive amount of model runs (and outputs) it is necessary to follow naming conventions for archiving the scenario results. This will support a more efficient comparison across case studies. Each output name must include the following information, separated by an underscore ('_'):

- Case study code ('cs1', 'cs2', etc.)
- Indicator code (tbd)
- LULC/NSWRM code (in this task just one option: 'statusquo')
- Period code ('ref', 'near', and 'far' for periods 1991-2020, 2031-2060, and 2071-2100, resp.)
- Climate model code (see Table 7.1)
- RCP code ('2p6', '4p5', '8p5' for RCPs 2.6, 4.5, and 8.5, resp.)

An example output could thus be: *'cs4_lowflow_statusquo_ref_hadgracmo_8p5.csv'*.

7.2 NSWRM effectiveness

OPTAIN's overall key question is to explore optimal spatial pattern and combinations of Natural/Small Water Retention Measures (NSWRM) to achieve maximum retention effectiveness and similarly meet multiple other objectives at the catchment-scale. While this task requires tens of thousands of model simulations driven by a multi-objective heuristic search algorithm (in our case NSGA-II), which will be done at a later stage in the project (coordinated by WP5), we can already use our calibrated SWAT+ models to run a pre-defined set of scenarios to evaluate the effectiveness of single NSWRMs at both catchment- and field-scale. It is expected that each case study will run several (4-6) "maximum implementation" scenarios of various NSWRMs that were earlier prioritised with local stakeholders in OPTAINs WP2. "Maximum implementation" means that a given measure is applied in all possible locations that were earlier reserved in the land use map in the case of structural measures (see section on Land input), or on all fields on which it is feasible in the case of management measures.

Each case study is requested to run one scenario for each NSWRM considered in the case study, namely the scenario of maximum implementation. Guidance on how to simulate a specific NSWRM with SWAT+ was provided by OPTAINs deliverable D2.3 (*'Participatory modelling settings and standardised guidelines for parameterisation of measures - SWAT+ and SWAP retention measure implementation handbook'*)(Marval et al. (2022)).

By following the rules for delineating land objects described in section 2.2, none of the NSWRM scenarios requires a new model setup. The existing model files of the calibrated and validated model can be used. However, a few of them must be edited as described in OPTAINs deliverable D2.3.

The process of running NSWRM scenarios will also be automatized, at latest for running the multi-objective optimization. WP5 will work on script-based solutions, also for running the maximum implementation scenarios. For such solutions, it can be already foreseen that each case study needs to prepare *hru_scenario.csv* files as exemplarily shown in Figure 7.1 and Figure 7.2. For each scenario, the new land use must replace the old one in file *'hru-data.hru'* (column *lu_mgt*). It is thus necessary that any potential land use is included in the *'landuse.lum'* file, pointing to the right parameters (e.g. Curve Numbers (*cn2*), USLE P (*cons_prac*), and Manning's n (*ov_mann*)) under scenario condition. Moreover, the 'scenario' parameter sets must be included in the respective parameter files (e.g. *'cntable.lum'*, *'cons_practice.lum'*, *'ovn_table.lum'*). Any new reservoir (e.g. representing a retention pond) must be added to the *'reservoir.res'* file (this implies also adding a new routing unit in the *'rout_unit.con'* file including the channel id into which this reservoir drains). The new reservoir must point to the right parameter values (*'hydrology.res'*, *'sediment.res'*,*'nutrients.res'*) and the hru changing to a reservoir must be deleted in file *'hru-data.hru'* (see OPTAINs deliverable D2.3 for relevant files and parameters of each NSWRM).

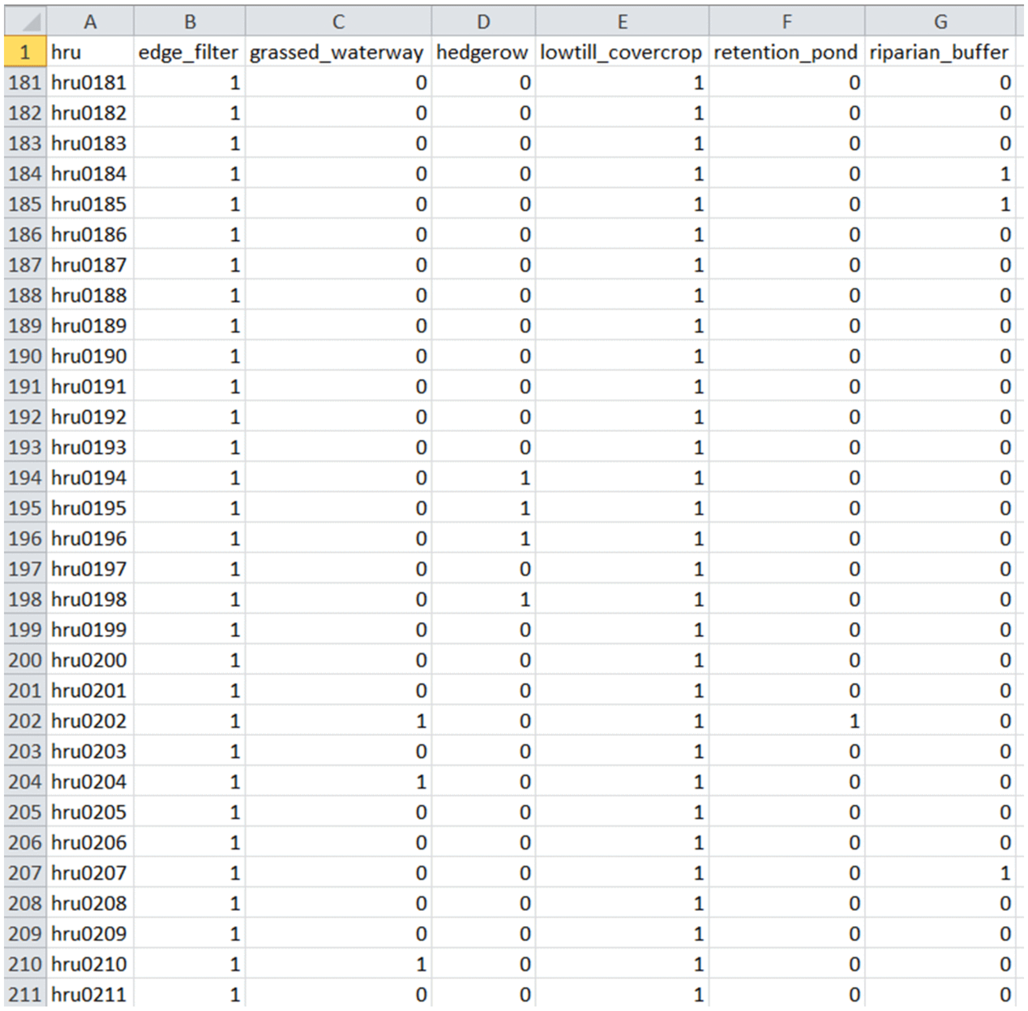
The NSWRM scenarios and the baseline model (representing the status quo without any new NSWRM implementation) should be run under observed climate covering both periods, calibration and validation period (+ 3 years warm up).

Recommendations on the "minimum" options for print settings will be provided. It should be kept in mind that massive amount of data may be generated for all the runs, especially if options for daily time step and HRU-level outputs are used.

A minimum common set of environmental performance indicators (e.g. changes in low or high flows, soil water retention, nitrate load, crop yields, etc.) based on OPTAINS deliverable 2.2 (*'Tailored environmental and socio-economic performance indicators for selected measures'*) (Krzeminska and Monaco (2022)) will be provided after consultation with WP2. These indicators will have to be calculated and delivered in a pre-defined format. The NSWRM effect will be illustrated by comparing boxplots for each NSWRM scenario and the baseline model. It is recommended to carry out comparisons for both the catchment and the field scale (where only selected fields or all fields where a NSWRM has been implemented are compared). These results will also be used during the local meetings with OPTAINs Multi-Actor Reference Groups.

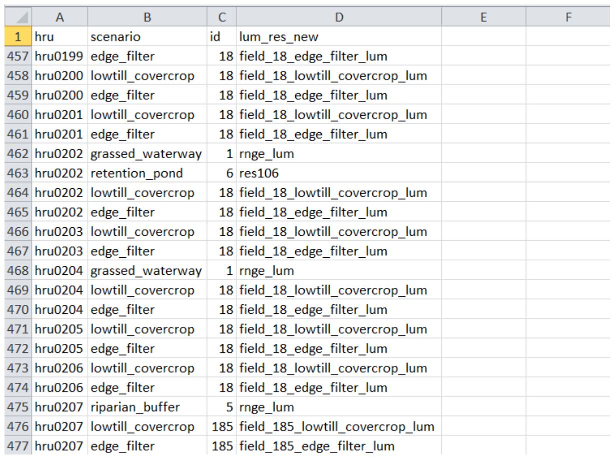
It is recommended that input and output files of each NSWRM scenario run are archived in a repository.

In order to support a more efficient comparison across case studies, each model output name must include the following information, separated by an underscore ('_'):



	A	B	C	D	E	F	G
1	hru	edge_filter	grassed_waterway	hedgerow	lowtill_covercrop	retention_pond	riparian_buffer
181	hru0181	1	0	0	1	0	0
182	hru0182	1	0	0	1	0	0
183	hru0183	1	0	0	1	0	0
184	hru0184	1	0	0	1	0	1
185	hru0185	1	0	0	1	0	1
186	hru0186	1	0	0	1	0	0
187	hru0187	1	0	0	1	0	0
188	hru0188	1	0	0	1	0	0
189	hru0189	1	0	0	1	0	0
190	hru0190	1	0	0	1	0	0
191	hru0191	1	0	0	1	0	0
192	hru0192	1	0	0	1	0	0
193	hru0193	1	0	0	1	0	0
194	hru0194	1	0	1	1	0	0
195	hru0195	1	0	1	1	0	0
196	hru0196	1	0	1	1	0	0
197	hru0197	1	0	0	1	0	0
198	hru0198	1	0	1	1	0	0
199	hru0199	1	0	0	1	0	0
200	hru0200	1	0	0	1	0	0
201	hru0201	1	0	0	1	0	0
202	hru0202	1	1	0	1	1	0
203	hru0203	1	0	0	1	0	0
204	hru0204	1	1	0	1	0	0
205	hru0205	1	0	0	1	0	0
206	hru0206	1	0	0	1	0	0
207	hru0207	1	0	0	1	0	1
208	hru0208	1	0	0	1	0	0
209	hru0209	1	0	0	1	0	0
210	hru0210	1	1	0	1	0	0
211	hru0211	1	0	0	1	0	0

Figure 7.1: Snippet of the *hru_scenario.csv* file for the German case study. The file lists all hrus relevant for at least one NSWRM. If a certain NSWRM can be potentially implemented in a certain HRU, it is indicated by value 1, while 0 excludes the implementation. In the given example, the case study will need to run six maximum implementation NSWRM scenarios (as there are six NSWRM columns), each time by editing the relevant model settings addressing all HRUs with value 1. No changes have to be made for HRUs with value 0. At this stage, we will not run combined NSWRM scenarios, where NSWRM can be implemented simultaneously within the catchment.



	A	B	C	D	E	F
1	hru	scenario	id	lum_res_new		
457	hru0199	edge_filter	18	field_18_edge_filter_lum		
458	hru0200	lowtill_covercrop	18	field_18_lowtill_covercrop_lum		
459	hru0200	edge_filter	18	field_18_edge_filter_lum		
460	hru0201	lowtill_covercrop	18	field_18_lowtill_covercrop_lum		
461	hru0201	edge_filter	18	field_18_edge_filter_lum		
462	hru0202	grassed_waterway	1	rnge_lum		
463	hru0202	retention_pond	6	res106		
464	hru0202	lowtill_covercrop	18	field_18_lowtill_covercrop_lum		
465	hru0202	edge_filter	18	field_18_edge_filter_lum		
466	hru0203	lowtill_covercrop	18	field_18_lowtill_covercrop_lum		
467	hru0203	edge_filter	18	field_18_edge_filter_lum		
468	hru0204	grassed_waterway	1	rnge_lum		
469	hru0204	lowtill_covercrop	18	field_18_lowtill_covercrop_lum		
470	hru0204	edge_filter	18	field_18_edge_filter_lum		
471	hru0205	lowtill_covercrop	18	field_18_lowtill_covercrop_lum		
472	hru0205	edge_filter	18	field_18_edge_filter_lum		
473	hru0206	lowtill_covercrop	18	field_18_lowtill_covercrop_lum		
474	hru0206	edge_filter	18	field_18_edge_filter_lum		
475	hru0207	riparian_buffer	5	rnge_lum		
476	hru0207	lowtill_covercrop	185	field_185_lowtill_covercrop_lum		
477	hru0207	edge_filter	185	field_185_edge_filter_lum		

Figure 7.2: Snippet of the *‘lum_res_new .csv’* file for the German case study. Such a table format can include names of the new land use (indicated by *_lum*) or reservoir (indicated by* res*) to trigger a chain of edits.

- Case study code (‘cs1’, ‘cs2’, etc.)
- Indicator code (tbd)
- LULC/NSWRM code (see Table 7.2)
- Period code (in this task just one option: ‘baseline’, which is referring to the period of calibration + validation, incl. 3 years of warm-up)

Table 7.2: List of NSWRMs and their codes to be used for naming output results in OPTAIN.

LULC/NSWRM	LULC/NSWRM code
No implementation, status quo	statusquo
Riparian buffers	buffers
Edge-of-field filter strips	edgefilter
Hedges/Field division	hedges
Grassland cover on erosive slopes	grassslope
Grassland cover in recharge area	grassrchrg
Retention/detention ponds	ponds
Afforestation	afforest
Floodplain restoration	floodres
Channel restoration	channres
Swales	swales
Constructed wetlands	wetlands
Controlled drainage	cdrain
Terracing	terraces
No-till agriculture	notill
Low-till agriculture	lowtill
Mulching	mulching
Subsoiling	subsoiling
Crop rotation	rotation
Intercropping	intercrop
Green cover/ catch crops	covercrop
Early sowing	earlysow

LULC/NSWRM	LULC/NSWRM code
Drought-resistant plants	droughtplt

An example output could thus be: *'cs1_phosporus_buffers_baseline.csv'*

7.3 Combined scenarios

By combining climate scenarios and the scenarios of maximum NSWRM implementation, we can study the effectiveness of NSWRM under changing climate. This requires repeating the 54 climate scenario runs, described in section 7.1, for all NSWRMs considered in a given case study (each time with only one NSWRM implemented at all possible locations as described in section 7.2). If a case study considers six different NSWRM, this will amount to a total of 324 additional model runs.

Due to the massive amount of model runs (and outputs) it is necessary to follow naming conventions for archiving the scenario results. Each output name must include the following information, separated by an underscore ('_'):

- Case study code ('cs1', 'cs2', etc.)
- Indicator code (tbd)
- LULC/NSWRM code (see section 7.2)
- Period code ('ref', 'near', and 'far' for periods 1991-2020, 2031-2060, and 2071-2100, resp.)
- Climate model code (see section 7.1)
- RCP code ('2p6', '4p5', '8p5' for RCPs 2.6, 4.5, and 8.5, resp.)

An example output could thus be: *'cs6_cropyield_droughtplt_near_mpiremo_2p6.csv'*

7.4 Uncertainty and sensitivity analysis

Uncertainty Analysis (UA) and sensitivity analysis Sensitivity Analysis (SA) are strongly linked and can be considered as two synonymous procedures which show the same thing from two different perspectives. While UA aims to quantify the uncertainties of a systems output with respect to uncertain inputs, SA quantifies the impact of the input uncertainties on the resulting output uncertainties and apportions the output uncertainties to the different uncertain inputs (Saltelli et al., 2008, 2004). UA and SA often implement the same methods and are therefore ideally employed in a combined system assessment to gain a comprehensive understanding of the analysed system (Pianosi et al., 2016).

7.4.1 Sources of uncertainty in the OPTAIN scenario assessment

Impact assessments like the one that is performed in the OPTAIN SWAT+ modeling case studies propagate through rather comprehensive modeling workflows. A wide range of uncertainties are introduced in every step of such a modeling chain which eventually subsume to the uncertainties of the actual impact assessment.

In OPTAIN's SWAT+ modeling workflows for example spatial data are collected from different data sources which are required in the model setup. All input data are uncertain to some degree. Sources of uncertainties can for example be measurement uncertainties (the most natural form of uncertainties), uncertainties that result from aggregation of data (e.g. spatially, temporally, or thematically), or simply because an input is again derived by a model (which may simplify a process and uses uncertain inputs). The SWAT model setup procedure uses the input data and generates a simplified representation of the landscape where it strongly aggregates all of its properties which again introduces uncertainties. The defined model setup is calibrated where different simulated model output variables are compared to uncertain observation data. The evaluation of the model performance is done with metrics that usually aggregate the information of the comparison of simulations and observations into a single value (Clark et al., 2021). Eventually, a model calibration finds several model parametrizations that will simulate the observed data equally well. Thus, all found model setups are equally plausible representations of the analysed system and cannot be rejected (Beven, 2006, 1996). This issue is well known in the hydrological literature with the term equifinality. To account for the uncertainties that result from the model setup procedure ensembles of model configurations and/or acceptable sets parameter combinations can be used in further model applications (see e.g. Schürz et al. (2019), Ficklin and Barnhart (2014)).

The representation of structural and management related NSWRMs is in any case simplified in the SWAT+ model setups, regardless whether the measures will be represented by spatial objects with the novel COCOA approach, or in a parametric way. Due to their simplified model representation the simulated effect of NSWRMs is highly uncertain. Data which stem for example from field experiments which can be used to validate the simulated effects of NSWRMs are limited and may not meet a specific situation in the respective OPTAIN case study. The uncertainties in potential effects of implemented NSWRMs can be expressed by using different equally plausible parametrizations of a certain measure, or by using different model representations. A representation of a measure must then be considered to be plausible when it cannot be rejected, e.g. when simulations contradict observation data (Beven, 2018).

The evaluation of the effectiveness of NSWRMs employs a small selection of metrics to assess whether a certain combination of NSWRMs has a strong positive effect on certain environmental and ecological criteria or not. Similar to the evaluation of the performance of a model setup, the selected metrics cannot provide a full picture of the performance of an NSWRM combination and strongly aggregates the information which is employed in the evaluation. Hence, besides the pareto-optimal characteristics of potentially well performing NSWRM combinations their informative value inherits a certain amount of uncertainty. As with other sources of uncertainties which are outlined here a potential way to account for the inherited uncertainties to some extent is to select sets of solutions in further impact assessments.

The scenario assessment in OPTAIN also accounts for the impact of the future climate development on water and nutrient retention in the analyzed case studies. As outlined in section 7.1 the common practice in climate change impact assessment is the use of climate model ensembles. Climate simulations which are implemented in an impact assessment are impacted by several sources of uncertainty. They may include several socioeconomic scenarios (e.g. the current RCPs Moss et al. (2010) that drive an array of GCMs (Knutti and Sedláček, 2013)). The GCMs also have inherent uncertainty. GCMs provide the boundary conditions for Regional Climate Models (RCMs) (e.g. Jacob et al. (2014)). The downscaling (Wilby et al., 1998) of the RCM simulations and the bias correction (Teutschbein and Seibert, 2012) are associated with their own uncertainty and are standard procedures in climate scenario development. Eventually, this chain of uncertainties should be reflected in the climate model ensemble which will be implemented in the OPTAIN case study simulations.

This rather incomprehensive view of the propagation of uncertainties through the OPTAIN modeling workflow clearly shows that a full consideration of all uncertainties in the final impact assessment is simply unfeasible. While some sources of uncertainties are easier to express (e.g. selection of model ensembles of climate simulations or environmental model parametrizations), other sources of uncertainties would be difficult to implement, simply because they may require substantial modifications in the model code (e.g. different model representations of NSWRMs), or would substantially complicate the modeling workflow (e.g. using separate, different SWAT+ model setups which represent structural model uncertainties).

7.4.2 Potential framework for UA and SA in OPTAIN

To harmonize the assessment of simulation uncertainties under consideration of selected sources of input uncertainties (UA) and to identify the most significant sources of uncertainties (SA) a flexible modeling framework must be implemented. Schürz et al. (2019) outlined such a framework for the assessment of different inputs such as future climate simulations, model parametrizations, future land use scenarios, or different structural model configurations for SWAT model impact assessments. Although this framework was developed for older versions of SWAT this framework can be updated and employed for the SWAT+ modeling studies in OPTAIN.

The conceptual framework in Schürz et al. (2019) is to separate the developed SWAT+ model setups in to building blocks that consist of model inputs which together represent one of the analyzed model inputs. The different future climate simulations for example may be represented by different sets of weather input files and farm managements which correspond to the weather time series, whereas spatial distributions of implemented NSWRMs may be represented by different assignemts of landuse and management to spatial objects in a model setup. The sections 7.1 climate change effects and 7.2 NSWRM effectiveness provided already a coarse outline how inputs and corresponding outputs can be organized to support the UA and SA analysis.

The UA and SA framework must be set up in a way that executable SWAT+ model setups can be built from the building blocks which were developed for the individual inputs. A scripted workflow in the programming language `R` will be developed based on the previous work in Schürz et al. (2019) to combine different realizations of the analyzed inputs, to run the assembled SWAT+ models and to etract the simulation outputs which will be further analyzed.

Based on the definite number of inputs which will be implemented in a combined UA and SA of the OPTAIN impact assessment appropriate methods for UA and SA will be selected. As the OPTAIN SWAT+ model setups are rather complex and computationally expensive a major criterion for the selection of UA and SA is the number of required model iterations. In any case methods will be selected which allow a combined analysis of simulation uncertainties and model input importance. One example is the PAWN SA method (Pianosi and Wagener, 2018, 2015) which allows the use of generic randomly sampled combinations of model inputs, which can be used in a combined analysis of model uncertainties and sensitivities.

An example application for such a combined UA and SA for multiple model inputs can be found in Schürz et al. (2019). The Figures 7.3 and 7.4 show a small part of the analysis in Schürz et al. (2019), but illustrates how the two different perspectives of UA and SA can be employed in the OPTAIN model impact assessments.

Figure 7.3 shows the calculated PAWN sensitivity indices for the 5 model inputs which were analyzed in this study. Larger values of PAWN sensitivity indices indicate that the respective model input had a high relevance for the simulated output uncertainties. The illustrated analysis shows a wide range of analyzed environmental variables. Thus, the plot panels which were separated into the different model inputs provide a good general overview of the importances of the model inputs for a future assessment of water resources and catchment nutrient budgets.

Figure 7.4 in contrast shows the ranges of simulated uncertainties in the analyzed output variables which result from the different combinations of the model inputs. The plots just show exemplary different ways to analyze output variables. The plotted bands show the simulation uncertainties. The uncertainty bands were separated with respect to the used climate scenario inputs. The colors show if a climate scenario simulated an increase or a decrease in future precipitation.

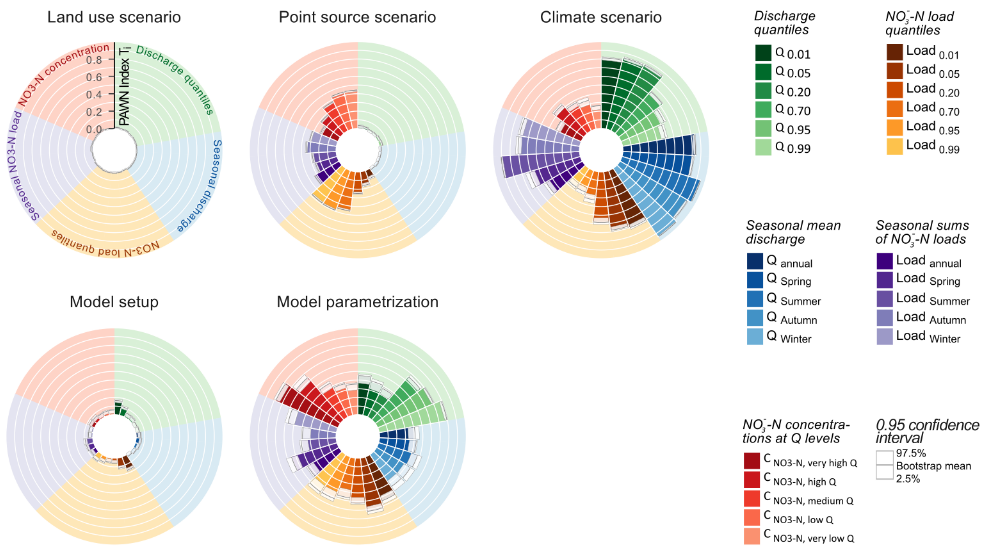


Figure 7.3: Example for a SA of multiple model inputs in an environmental impact assessment with SWAT for a wide range of environmental variables (adapted from Schürz et al. (2019)). The plot panels show the analyzed model inputs. The bars show the parameter importances for the simulation of the respective environmental variables with the PAWN sensitivity index.

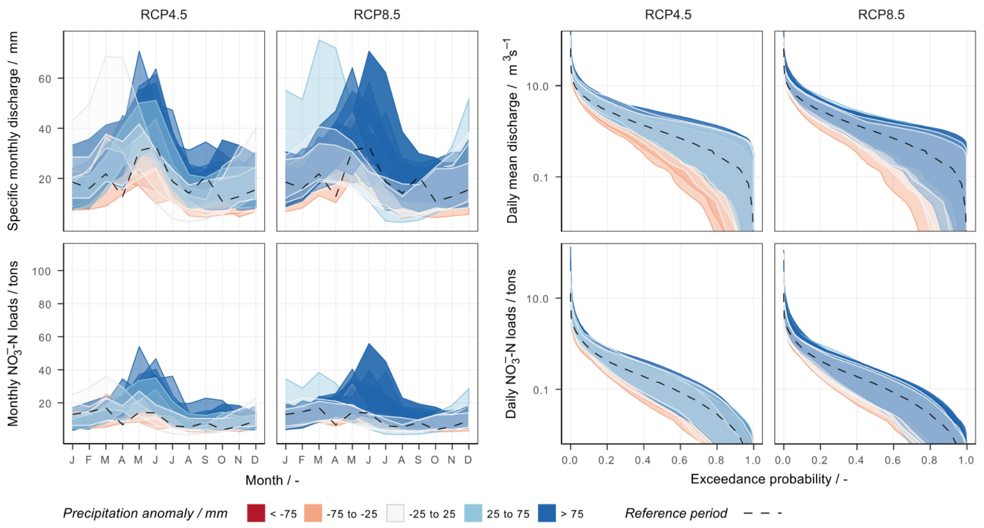


Figure 7.4: Example for a UA of multiple model inputs in an environmental impact assessment with SWAT. The plot panels show long-term monthly averages of discharge an N-loads (left). and their corresponding duration curves. The plotted areas show the simulated uncertainties which result from the analyzed combinations of model inputs. The uncertainty bands in this plot were separated for different climate scenarios, to illustrate the effect of different used climate simulations. Although in OPTAIN not simulated time series of output variables are of major interest but signature measures which express the impacts on water resources and nutrient budgets such a way of visualization could provide relevant insights in the main drivers of the simulated uncertainties in an impact assessment.

7.4.3 Design of the UA and SA in OPTAIN

As outlined in the sections above, the design of the final UA and SA in OPTAIN will be determined by the computational costs/resources, the number of model inputs that we eventually plan to analyze in a combined approach, and the number of realizations that should be considered for each model input in the analysis. A major limitation may be that the number of inputs which are considered in an analysis will exponentially increase the number of combinations of the analyzed inputs (curse of dimensionality). As the computational costs of a single model run will be high in OPTAIN the total number of individual model inputs which will be analyzed should be kept low. A minimum version for the combined UA and SA will include a combination of potentially effective NSWRMs scenarios (including extreme scenarios) and the future climate simulations.

References

Abbaspour, K.C., 2015. SWAT-CUP: SWAT calibration and uncertainty programs - a user manual. Department of Systems Analysis, Integrated Assessment; Modelling (SIAM), Eawag. Swiss Federal Institute of Aquatic Science; Technology,.

Abbaspour, K.C., AshrafVaghefi, S., Yang, H., Srinivasan, R., 2019. Global soil, landuse, evapotranspiration, historical and future weather databases for SWAT Applications. Scientific Data 6:263. https://doi.org/https://doi.org/10.1038/s41597-019-0282-4

Abbaspour, K.C., Genuchten, M.T. van, Schulin, R., Schläppi, E., 1997. A sequential uncertainty domain inverse procedure for estimating subsurface flow and transport parameters. Water Resources Research 33, 1879–1892. https://doi.org/10.1029/97wr01230

Abbaspour, K.C., Johnson, C.A., Genuchten, M.Th. van, 2004. Estimating uncertain flow and transport parameters using a sequential uncertainty fitting procedure. Vadose Zone Journal 3, 1340–1352. https://doi.org/10.2136/vzj2004.1340

Abbaspour, K.C., Rouholahnejad, E., Vaghefi, S., Srinivasan, R., Yang, H., Kløve, B., 2015. A continental-scale hydrology and water quality model for Europe: Calibration and uncertainty of a high-resolution large-scale SWAT model. Journal of Hydrology 524, 733–752. https://doi.org/10.1016/j.jhydrol.2015.03.027

Abbaspour, K.C., Vaghefi, S.A., Srinivasan, R., 2017. A Guideline for Successful Calibration and Uncertainty Analysis for Soil and Water Assessment: A Review of Papers from the 2016 International SWAT Conference. Water 10, 6. https://doi.org/10.3390/w10010006

Ad-hoc-AG Boden, 2005. Bodenkundliche Kartieranleitung., 5th ed. Federal Institute for Geosciences and Natural Resources, Hannover.

Akoko, G., Le, T.H., Gomi, T., Kato, T., 2021. A Review of SWAT Model Application in Africa. Water 13, 1313. https://doi.org/10.3390/w13091313

Alemayehu, T., Gupta, H.V., Griensven, A. van, Bauwens, W., 2022. On the calibration of spatially distributed hydrologic models for poorly gauged basins: Exploiting information from streamflow signatures and remote sensing-based evapotranspiration data. Water 14, 1252. https://doi.org/10.3390/w14081252

Alexander, E.B., 1980. Bulk Densities of California Soils in Relation to Other Soil Properties. Soil Science Society of America Journal 44, 689–692. https://doi.org/10.2136/sssaj1980.03615995004400040005x

Arnold, J.G., Allen, P.M., Muttiah, R., Bernhardt, G., 1995. Automated base flow separation and recession analysis techniques. Groundwater 33, 1010–1018. https://doi.org/10.1111/j.1745-6584.1995.tb00046.x

Arnold, J.G., Bieger, K., White, M.J., Srinivasan, R., Dunbar, J.A., Allen, P.M., 2018. Use of Decision Tables to Simulate Management in SWAT+. Water 10, 713. https://doi.org/10.3390/w10060713

Arnold, J.G., Fohrer, N., 2005. SWAT2000: current capabilities and research opportunities in applied watershed modelling. Hydrological Processes 19, 563–572. https://doi.org/10.1002/hyp.5611

Arnold, J.G., Kiniry, J.R., Srinivasan, R., Williams, J.R., Haney, E.B., Neitsch, S.L., 2012a. Soil & Water Assessment Tool, Input/Output Documentation.

Arnold, J.G., Moriasi, D.N., Gassman, P.W., Abbaspour, K.C., White, M.J., Srinivasan, R., Santhi, C., Harmel, R.D., Griensven, A. van, Liew, M.W.V., Kannan, N., Jha, M.K., 2012b. SWAT: model use, calibration, and validation. Journal Name Changed Transactions of the ASABE 55, 1491–1508.

Arnold, J.G., Moriasi, D.N., Gassman, P.W., Abbaspour, K.C., White, M.J., Srinivasan, R., Santhi, C., Harmel, R.D., Griensven, A. van, Liew, M.W.V., Kannan, N., Jha, M.K., 2012c. SWAT: Model use, calibration, and validation. Transactions of the ASABE 55, 1491–1508. https://doi.org/10.13031/2013.42256

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

Arnold, J., Youssef, M., Yen, H., White, M., Sheshukov, A., Sadeghi, A., Moriasi, D., Steiner, J., Amatya, D.M., Skaggs, R.W., Haney, E., Jeong, J., Arabi, M., Gowda, P., 2016. Hydrological processes and model representation: Impact of soft data on calibration. Transactions of the ASABE (American Society of Agricultural and Biological Engineers) 58. https://doi.org/10.13031/trans.58.10726

Assouline, S., Or, D., 2014. The concept of field capacity revisited: Defining intrinsic static and dynamic criteria for soil internal drainage dynamics. Water Resources Research 50, 4787–4802. https://doi.org/10.1002/2014WR015475

Bailey, R.T., Bieger, K., Flores, L., Tomer, M., 2022. Evaluating the contribution of subsurface drainage to watershed water yield using SWAT+ with groundwater modeling. Science of The Total Environment 802, 149962. https://doi.org/https://doi.org/10.1016/j.scitotenv.2021.149962

Baldan, D., Mehdi, B., Feldbacher, E., Piniewski, M., Hauer, C., Hein, T., 2021. Assessing multi-scale effects of natural water retention measures on in-stream fine bed material deposits with a modeling cascade. Journal of Hydrology 594, 125702. https://doi.org/10.1016/j.jhydrol.2020.125702

Barnard, J., 1948. Heat units as a measure of canning crop maturity. The Canner 106, 28.

Bennett, N.D., Croke, B.F.W., Guariso, G., Guillaume, J.H.A., Hamilton, S.H., Jakeman, A.J., Marsili-Libelli, S., Newham, L.T.H., Norton, J.P., Perrin, C., Pierce, S.A., Robson, B., Seppelt, R., Voinov, A.A., Fath, B.D., Andreassian, V., 2013. Characterising performance of environmental models. Environmental Modelling & Software 40, 1–20. https://doi.org/10.1016/j.envsoft.2012.09.011

Beven, K.J., 2018. On hypothesis testing in hydrology: Why falsification of models is still a really good idea. Wiley Interdisciplinary Reviews: Water 5, e1278. https://doi.org/10.1002/wat2.1278

Beven, K.J., 2006. A manifesto for the equifinality thesis. Journal of Hydrology 320, 18–36. https://doi.org/10.1016/j.jhydrol.2005.07.007

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

Bieger, K., Arnold, J.G., Rathjens, H., White, M.J., Bosch, D.D., Allen, P.M., 2019. Representing the Connectivity of Upland Areas to Floodplains and Streams in SWAT+. JAWRA Journal of the American Water Resources Association 55, 578–590. https://doi.org/10.1111/1752-1688.12728

Bieger, K., Arnold, J.G., Rathjens, H., White, M.J., Bosch, D.D., Allen, P.M., Volk, M., Srinivasan, R., 2017. Introduction to SWAT+, A Completely Restructured Version of the Soil and Water Assessment Tool. JAWRA Journal of the American Water Resources Association 53, 115–130. https://doi.org/10.1111/1752-1688.12482

Bieger, K., Hörmann, G., Fohrer, N., 2014. Simulation of streamflow and sediment with the soil and water assessment tool in a data scarce catchment in the three gorges region, china. Journal of Environmental Quality 43, 37–45. https://doi.org/10.2134/jeq2011.0383

Bieger, K., Rathjens, H., Allen, P.M., Arnold, J.G., 2015. Development and evaluation of bankfull hydraulic geometry relationships for the physiographic regions of the united states. JAWRA Journal of the American Water Resources Association 51, 842–858. https://doi.org/https://doi.org/10.1111/jawr.12282

Čerkasova, N., Nemes, N., Szabó, B., Idzelytė, R, Cüceloğlu, G., Mészáros, J., Kassai, P., Shore, M., Farkas, C., Czelnai, L., 2022. Created data pre-processors successfully applied for input data restructuring. Deliverable D3.3 EU horizon 2020 OPTAIN project, grant agreement no. 862756. https://doi.org/doi:10.5281/zenodo.7052806

Chaubey, I., Migliaccio, K.W., Green, C.H., Arnold, J.G., Srinivasan, R., 2006. Phosphorus modeling in soil and water assessment tool (SWAT) model. Modeling Phosphorus in the Environment 163–187. https://doi.org/10.1201/9781420005417.sec2

Clark, M.P., Vogel, R.M., Lamontagne, J.R., Mizukami, N., Knoben, W.J.M., Tang, G., Gharari, S., Freer, J.E., Whitfield, P.H., Shook, K.R., Papalexiou, S.M., 2021. The abuse of popular performance metrics in hydrologic modeling. Water Resources Research 57. https://doi.org/10.1029/2020wr029001

Costa, M.G., Gama-Rodrigues, A.C., Gonçalves, J.L. de M., Gama-Rodrigues, E.F., Sales, M.V. da S., Aleixo, S., 2016. Labile and non-labile fractions of phosphorus and its transformations in soil under Eucalyptus plantations, Brazil. Forests 7, 1–15. https://doi.org/10.3390/f7010015

D'Andrimont, R., Yordanov, M., Martinez-Sanchez, L., Eiselt, B., Palmieri, A., Dominici, P., Gallego, J., Reuter, H.I., Joebges, C., Lemoine, G., Velde, M. van der, 2020. Harmonised LUCAS in-situ land cover and use database for field surveys from 2006 to 2018 in the European Union. Scientific Data 7, 1–15. https://doi.org/10.1038/s41597-020-00675-z

Efstratiadis, A., Koutsoyiannis, D., 2010. One decade of multi-objective calibration approaches in hydrological modelling: A review. Hydrological Sciences Journal 55, 58–78. https://doi.org/10.1080/02626660903526292

Egnér, H., Riehm, H., Domingo, W.R., 1960. Untersuchungen über die chemische Bodenanalyse als Grundlage für die Beurteilung des Nährstoffzustandes der Böden. II. Chemische Extraktionsmethoden zur Phosphor- und Kaliumbestimmung. Lantbr. Ann. 26, 199–215.

Engel, B., Storm, D., White, M., Arnold, J., Arabi, M., 2007. A hydrologic/water quality model Applicati11. JAWRA Journal of the American Water Resources Association 43, 1223–1236. https://doi.org/10.1111/j.1752-1688.2007.00105.x

EUROSTAT, 2020. Annual crop statistics. Handbook 2020 edition.

Euser, T., Winsemius, H.C., Hrachowitz, M., Fenicia, F., Uhlenbrook, S., Savenije, H.H.G., 2013. A framework to assess the realism of model structures using hydrological signatures. Hydrology and Earth System Sciences 17, 1893–1912. https://doi.org/10.5194/hess-17-1893-2013

Fan, Y., Li, H., Miguez-Macho, G., 2013. Global patterns of groundwater table depth. Science 339, 940–943. https://doi.org/10.1126/science.1229881

Fernandez-Palomino, C.A., Hattermann, F.F., Krysanova, V., Vega-Jácome, F., Bronstert, A., 2021. Towards a more consistent eco-hydrological modelling through multi-objective calibration: A case study in the andean vilcanota river basin, peru. Hydrological Sciences Journal 66, 59–74. https://doi.org/10.1080/02626667.2020.1846740

Ficklin, D.L., Barnhart, B.L., 2014. SWAT hydrologic model parameter uncertainty and its implications for hydroclimatic projections in snowmelt-dependent watersheds. Journal of Hydrology 519, 2081–2090. https://doi.org/10.1016/j.jhydrol.2014.09.082

Fohrer, N., Wagner, P.D., Kiesel, J., Haas, M., Guse, B., 2022. A guideline for spatio-temporal consistency in water quality modelling in rural areas. Hydrological Processes 36, e14711. https://doi.org/10.1002/hyp.14711

Food and Agriculture Organization, 1990. Guidelines for Soil Profile Description., 3rd ed. Food and Agriculture Organization of the United Nations (FAO), Rome, Italy.

Foster, G.R., McCool, D.K., Renard, K.G., Moldenhauer, W.C., 1981. Conversion of the universal soil loss equation to SI metric units. Journal of Soil & Water Conservation 36, 355–359.

Fu, B., Horsburgh, J.S., Jakeman, A.J., Gualtieri, C., Arnold, T., Marshall, L., Green, T.R., Quinn, N.W.T., Volk, M., Hunt, R.J., Vezzaro, L., Croke, B.F.W., Jakeman, J.D., Snow, V., Rashleigh, B., 2020. Modeling Water Quality in Watersheds: From Here to the Next Generation. Water Resources Research 56, 10.1029/2020wr027721. https://doi.org/10.1029/2020wr027721

Fu, B., Merritt, W.S., Croke, B.F.W., Weber, T., Jakeman, A.J., 2018. A review of catchment-scale water quality and erosion models and a synthesis of future prospects. Environmental Modelling & Software 114, 75–97. https://doi.org/10.1016/j.envsoft.2018.12.008

Gascoin, S., Duchame, A., Ribstein, P., Perroy, E., Wagnon, P., 2009. Sensitivity of bare soil albedo to surface soil moisture on the moraine of the Zongo glacier (Bolivia). Geophysical Research Letters 36, 2–6. https://doi.org/10.1029/2008GL036377

Gassman, P.W., Sadeghi, A.M., Srinivasan, R., 2014. Applications of the SWAT Model Special Section: Overview and Insights. Journal of Environmental Quality 43, 1–8. https://doi.org/10.2134/jeq2013.11.0466

Genuchten, M.Th. van, 1980. A closed-form equation for predicting the hydraulic conductivity of unsaturated soils. Soil Science Society of America Journal 44, 892–898.

Gökkaya, K., Budhathoki, M., Christopher, S.F., Hanrahan, B.R., Tank, J.L., 2017. Subsurface tile drained area detection using GIS and remote sensing in an agricultural watershed. Ecological Engineering 108, 370–379. https://doi.org/https://doi.org/10.1016/j.ecoleng.2017.06.048

Grizzetti, B., Bouraoui, F., de Marsily, G., Bidoglio, G., 2005. A statistical method for source apportionment of riverine nitrogen loads. Journal of Hydrology 304, 302–315. https://doi.org/https://doi.org/10.1016/j.jhydrol.2004.07.036

Grochowska, J., Augustyniak, R., Łopata, M., Tandyrak, R., 2020. Is it possible to restore a heavily polluted, shallow, urban lake? Applied Sciences 10. https://doi.org/10.3390/app10113698

Guarracino, L., 2007. Estimation of saturated hydraulic conductivity Ks from the van Genuchten shape parameter α. Water Resources Research 43, 15–18. https://doi.org/10.1029/2006WR005766

Gunn, K.M., Buda, A.R., Preisendanz, H.E., Cibin, R., Kennedy, C.D., Veith, T.L., 2021. Integrating daily CO2 concentrations in SWAT-VSA to examine climate change impacts on hydrology in a karst watershed. Transactions of the ASABE 64, 1303–1318. https://doi.org/10.13031/trans.13711

Guse, B., Kiesel, J., Pfannerstill, M., Fohrer, N., 2020. Assessing parameter identifiability for multiple performance criteria to constrain model parameters. Hydrological Sciences Journal 65, 1158–1172. https://doi.org/10.1080/02626667.2020.1734204

Haas, M.B., Guse, B., Pfannerstill, M., Fohrer, N., 2016. A joined multi-metric calibration of river discharge and nitrate loads with different performance measures. Journal of Hydrology 536, 534–545. https://doi.org/10.1016/j.jhydrol.2016.03.001

Haas, M.B., Guse, B., Pfannerstill, M., Fohrer, N., 2015. Detection of dominant nitrate processes in ecohydrological modeling with temporal parameter sensitivity analysis. Ecological Modelling 314, 62–72. https://doi.org/10.1016/j.ecolmodel.2015.07.009

Harmel, R.D., Smith, P.K., Migliaccio, K.W., Chaubey, I., Douglas-Mankin, K.R., Benham, B., Shukla, S., Muñoz-Carpena, R., Robson, B.J., 2014. Evaluating, interpreting, and communicating performance of hydrologic/water quality models considering intended use: A review and recommendations. Environmental Modelling & Software 57, 40–51. https://doi.org/10.1016/j.envsoft.2014.02.013

Hirsch, R.M., 1979. An evaluation of some record reconstruction techniques. Water Resources Research 15, 1781–1790. https://doi.org/https://doi.org/10.1029/WR015i006p01781

Honzak, L., Pogačar, T., 2022. Climate scenarios for integrated modelling. Deliverable D3.1 EU horizon 2020 OPTAIN project, grant agreement no. 862756. https://doi.org/doi:10.5281/zenodo.7050730

Jacob, D., Petersen, J., Eggert, B., Alias, A., Christensen, O.B., Bouwer, L.M., Braun, A., Colette, A., Déqué, M., Georgievski, G., Georgopoulou, E., Gobiet, A., Menut, L., Nikulin, G., Haensler, A., Hempelmann, N., Jones, C., Keuler, K., Kovats, S., Kröner, N., Kotlarski, S., Kriegsmann, A., Martin, E., Meijgaard, E. van, Moseley, C., Pfeifer, S., Preuschmann, S., Radermacher, C., Radtke, K., Rechid, D., Rounsevell, M., Samuelsson, P., Somot, S., Soussana, J.-F., Teichmann, C., Valentini, R., Vautard, R., Weber, B., Yiou, P., 2014. EURO-CORDEX: new high-resolution climate change projections for European impact research. Regional Environmental Change 14, 563–578. https://doi.org/10.1007/s10113-013-0499-2

Jakeman, A.J., Letcher, R.A., Norton, J.P., 2006. Ten iterative steps in development and evaluation of environmental models. Environmental Modelling & Software 21, 602–614. https://doi.org/10.1016/j.envsoft.2006.01.004

Jalowska, A.M., Yuan, Y., 2019. Evaluation of SWAT impoundment modeling methods in water and sediment simulations. JAWRA Journal of the American Water Resources Association 55, 209–227. https://doi.org/https://doi.org/10.1111/1752-1688.12715

Jeong, J., Santhi, C., Arnold, J.G., Srinivasan, R., Pradhan, S., Flynn, K.F., 2011. Development of algorithms for modeling onsite wastewater systems within SWAT. Transactions of the ASABE 54, 1693–1704.

Jingwen Wu, J.G.A., Haw Yen, 2022. Development of reservoir operation functions in SWAT+ for national environmental assessments. Journal of Hydrology 583. https://doi.org/10.1016/j.jhydrol.2020.124556

Jones, C.A., Sharpley, A.N., Williams, J.R., 1984. A Simplified Soil and Plant Phosphorus Model: III. Testing. Soil Science Society of America Journal 48, 810–813. https://doi.org/10.2136/SSSAJ1984.03615995004800040022X

Jönsson, H., Vinnerås, B., 2004. Adapting the nutrient content of urine and faeces in different countries using FAO and swedish data. Ecosan-closing the Loop, 2nd International Symposium on Ecological Sanitation. Luebeck, Germany 623–626.

Kakoulaki, G., Martinez, A., Florio, P., 2021. Non-commercial Light Detection and Ranging (LiDAR) data in Europe. Publications Office of the European Union, Luxembourg.

Kaste, Ø., Austnes, K., Wit, H. de, 2020. Streamwater responses to reduced nitrogen deposition at four small upland catchments in norway. Ambio 49. https://doi.org/10.1007/s13280-020-01347-3

Knoben, W.J.M., Freer, J.E., Woods, R.A., 2019. Technical note: Inherent benchmark or not? Comparing Nash-Sutcliffe and Kling-Gupta efficiency scores. Hydrology and Earth System Sciences Discussions 1–7. https://doi.org/10.5194/hess-2019-327

Knutti, R., Sedláček, J., 2013. Robustness and uncertainties in the new CMIP5 climate model projections. Nature Climate Change 3, 369–373. https://doi.org/10.1038/nclimate1716

Krzeminska, D., Monaco, F., 2022. Tailored environmental and socio-economic performance indicators for selected measures. Deliverable D2.2 of the EU horizon 2020 project OPTAIN. https://doi.org/doi:10.5281/zenodo.7050652

Lemann, T., Fribourg-Blanc, B., Magnier, J., Eichenberger, J., 2022. Coherent catalogue with a selection of most promising NSWRM including results from MARG exchanges. Deliverable D2.1 EU horizon 2020 OPTAIN project, grant agreement no. 862756.

Leta, O.T., Griensven, A. van, Bauwens, W., 2017. Effect of single and multisite calibration techniques on the parameter estimation, performance, and output of a SWAT model of a spatially heterogeneous catchment. Journal of Hydrologic Engineering 22, 05016036. https://doi.org/10.1061/(ASCE)HE.1943-5584.0001471

Lindsay, J.B., Dhun, K., 2015. Modelling surface drainage patterns in altered landscapes using LiDAR. International Journal of Geographical Information Science 29, 397–411. https://doi.org/10.1080/13658816.2014.975715

Malagó, A., Bouraoui, F., Vigiak, O., Grizzetti, B., Pastori, M., 2017. Modelling water and nutrient fluxes in the danube river basin with SWAT. Science of The Total Environment 603-604, 196–218. https://doi.org/10.1016/j.scitotenv.2017.05.242

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

Marval, Š., Fučík, P., Čerkasova, N., Schürz, C., Farkas, C., Piniewski, M., Strauch, M., Weiland, S., Plunge, S., Krzeminska, D., Lemann, T., Witing, F., 2022. SWAT+ and SWAP retention measure implementation handbook. Deliverable D2.3 EU horizon 2020 OPTAIN project, grant agreement no. 862756.

McMillan, H., 2020. Linking hydrologic signatures to hydrologic processes: A review. Hydrological Processes 34, 1393–1409. https://doi.org/https://doi.org/10.1002/hyp.13632

McMillan, H.K., 2021. A review of hydrologic signatures and their applications. WIREs Water 8, e1499. https://doi.org/10.1002/wat2.1499

McMillan, H.K., Gnann, S.J., Araki, R., 2022. Large scale evaluation of relationships between hydrologic signatures and processes. Water Resources Research 58, e2021WR031751. https:

//doi.org/10.1029/2021WR031751

Mehdi, B., Schulz, K., Ludwig, R., Ferber, F., Lehner, B., 2018. Evaluating the importance of non-unique behavioural parameter sets on surface water quality variables under climate change conditions in a mesoscale agricultural watershed. Water resources management 32, 619–639.

Moeys, J., 2014. The soil texture wizard: R functions for plotting, classifying, transforming and exploring soil texture data. R package version 1.2.13.

Moriasi, D.N., Arnold, J.G., Liew, M.W.V., Bingner, R.L., Harmel, R.D., Veith, T.L., 2007. Model evaluation guidelines for systematic quantification of accuracy in watershed simulations. Transactions of the ASABE 50(3), 885–900. https://doi.org/10.13031/2013.23153

Moriasi, D.N., Gitau, M.W., Pai, N., Daggupati, P., 2015. Hydrologic and Water Quality Models: Performance Measures and Evaluation Criteria. Transactions of the ASABE 58, 1763–1785. https://doi.org/10.13031/trans.58.10715

Moriasi, D.N., Rossi, C.G., Arnold, J.G., Tomer, M.D., 2012. Evaluating hydrology of the soil and water assessment tool (SWAT) with new tile drain equations. Journal of Soil and Water Conservation 67, 513–524. https://doi.org/10.2489/jswc.67.6.513

Moss, R.H., Edmonds, J.A., Hibbard, K.A., Manning, M.R., Rose, S.K., Van Vuuren, D.P., Carter, T.R., Emori, S., Kainuma, M., Kram, T., Meehl, G.A., Mitchell, J.F.B., Nakicenovic, N., Riahi, K., Smith, S.J., Stouffer, R.J., Thomson, A.M., Weyant, J.P., Wilbanks, T.J., 2010. The next generation of scenarios for climate change research and assessment. Nature 463, 747–756. https://doi.org/10.1038/nature08823

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

Nasta, P., Szabó, B., Romano, N., 2021. Evaluation of pedotransfer functions for predicting soil hydraulic properties : A voyage from regional to field scales across Europe. Journal of Hydrology: Regional Studies 37, 100903. https://doi.org/10.1016/j.ejrh.2021.100903

Nemes, A., 2022. Algorithm to harmonize soil particle size data to the FAO/USDA system. https://doi.org/10.5281/zenodo.7353722

Nemes, A., Quebedeaux, B., Timlin, D.J., 2010. Ensemble Approach to Provide Uncertainty Estimates of Soil Bulk Density. Soil Science Society of America Journal 74, 1938–1945. https://doi.org/10.2136/sssaj2009.0370

Nemes, A., Rawls, W.J., 2006. Evaluation of different representations of the particle-size distribution to predict soil water retention. Geoderma 132, 47–58. https://doi.org/10.1016/J.GEODERMA.2005.04.018

Nemes, A., Rawls, W.J., Pachepsky, Y.A., 2006a. Use of the Nonparametric Nearest Neighbor Approach to Estimate Soil Hydraulic Properties. Soil Science Society of America Journal 70, 327–336. https://doi.org/10.2136/SSSAJ2005.0128

Nemes, A., Rawls, W.J., Pachepsky, Ya.A., Genuchten, M.Th. van, 2006b. Sensitivity Analysis of the Nonparametric Nearest Neighbor Technique to Estimate Soil Water Retention. Vadose Zone Journal 5, 1222–1235. https://doi.org/10.2136/vzj2006.0017

Nemes, A., Wösten, J.H.M., Lilly, A., Voshaar, J.O., 1999. Evaluation of different procedures to interpolate particle-size distributions to achieve compatibility within soil databases. Geoderma 90, 187–202.

Nguyen, V.T., Dietrich, J., Uniyal, B., Tran, D.A., 2018. Verification and correction of the hydrologic routing in the soil and water assessment tool. Water 10. https://doi.org/10.3390/w10101419

NRES-21 Hydrology committee of ASABE, 2017. Guidelines for calibrating, validating, and evaluating hydrologic and water quality (h/WQ) models. American Society of Agricultural and Biological Engineers, 621th series.

O'Callaghan, J.F., Mark, D.M., 1984. The extraction of drainage networks from digital elevation data. Computer vision, graphics, and image processing 28, 323–344.

Olsen, R., Cole, C.V., Watanabe, F.S., Dean, L.A., 1954. Estimation of available phosphorus in soils by extraction with sodium bicarbonate. USDA Circular No. 939 USDQ, Government printing office, Washington DC.

Padel, S., 2001. Conversion to organic production software (OrgPlan, OF0159).

Pfannerstill, M., Guse, B., Fohrer, N., 2014. Smart low flow signature metrics for an improved overall performance evaluation of hydrological models. Journal of Hydrology 510, 447–458. https://doi.org/10.1016/j.jhydrol.2013.12.044

Pianosi, F., Beven, K.J., Freer, J., Hall, J.W., Rougier, J., Stephenson, D.B., Wagener, T., 2016. Sensitivity analysis of environmental models: A systematic review with practical workflow. Environmental Modelling & Software 79, 214–232. https://doi.org/10.1016/j.envsoft.2016.02.008

Pianosi, F., Wagener, T., 2018. Distribution-based sensitivity analysis from a generic input-output sample. Environmental Modelling & Software 108, 197–207. https://doi.org/10.1016/j.envsoft.2018.07.019

Pianosi, F., Wagener, T., 2015. A simple and efficient method for global sensitivity analysis based on cumulative distribution functions. Environmental Modelling & Software 67, 1–11. https://doi.org/10.1016/j.envsoft.2015.01.004

Piniewski, M., Marcinkowski, P., Koskiaho, J., Tattari, S., 2019. The effect of sampling frequency and strategy on water quality modelling driven by high-frequency monitoring data in a boreal catchment. Journal of Hydrology 579, 124186. https://doi.org/10.1016/j.jhydrol.2019.124186

Piniewski, M., Szcześniak, M., Kardel, I., Berezowski, T., Okruszko, T., Srinivasan, R., Schuler, D.V., Kundzewicz, Z.W., 2017. Hydrological modelling of the vistula and odra river basins using SWAT. Hydrological Sciences Journal 62, 1266–1289. https://doi.org/10.1080/02626667.2017.1321842

Poggio, L., De Sousa, L.M., Batjes, N.H., Heuvelink, G.B.M., Kempen, B., Ribeiro, E., Rossiter, D., 2021. SoilGrids 2.0: Producing soil information for the globe with quantified spatial uncertainty. SOIL 7, 217–240. https://doi.org/10.5194/SOIL-7-217-2021

Rathjens, H., Bieger, K., Srinivasan, R., Chaubey, I., Arnold, J.G., 2016. CMhyd User Manual. Documentation for preparing simulated climate change data for hydrologic impact studies.

Rathjens, H., Oppelt, N., 2012. SWATgrid: An interface for setting up SWAT in a grid-based discretization scheme. Computers & Geosciences 45, 161–167. https://doi.org/10.1016/j.cageo.2011.11.004

Ross, C.W., Prihodko, L., Anchang, J., Kumar, S., Ji, W., Hanan, N.P., 2018. Global Hydrologic Soil Groups (HYSOGs250m) for Curve Number-Based Runoff Modeling. https://doi.org/https://doi.org/10.3334/ORNLDAAC/1566

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

Saltelli, A., Tarantola, S., Campolongo, F., Ratto, M., 2004. Sensitivity analysis in practice: A guide to assessing scientific models, 1st ed. John Wiley & Sons Ltd, Chichester, West Sussex, UK.

Schürz, C., Hollosi, B., Matulla, C., Pressl, A., Ertl, T., Schulz, K., Mehdi, B., 2019. 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, 1211–1244. https://doi.org/10.5194/hess-23-1211-2019

Seibert, J., McDonnell, J.J., 2003. The quest for an improved dialog between modeler and experimentalist, in: Calibration of Watershed Models. American Geophysical Union (AGU), pp. 301–315. https://doi.org/https://doi.org/10.1002/9781118665671.ch22

Shafii, M., Tolson, B.A., 2015. Optimizing hydrological consistency by incorporating hydrological signatures into model calibration objectives. Water Resources Research 51, 3796–3814. https://doi.org/10.1002/2014WR016520

Sharma, A., McDaniel, R., Mehan, S., Arnold, J., Trooien, T., Sammons, N., Amegbletor, L., 2022. Field scale simulation of drainage water management using swat+: Analysis on the model performance, intra field variability, climate variability, and viability for eastern south dakota. https://doi.org/10.2139/ssrn.4076538

Sharpley, A.N., Jones, C.A., Gray, C., Cole, C.V., 1984. A Simplified Soil and Plant Phosphorus Model: II. Prediction of Labile, Organic, and Sorbed Phosphorus. Soil Science Society of America Journal 48, 805–809. https://doi.org/10.2136/SSSAJ1984.03615995004800040021X

Sharpley, A.N., Williams, J.R., 1990. EPIC — Erosion / Productivity Impact Calculator: 1. Model

Documentation. (No. Technical Bulletin No. 1768.). U.S. Department of Agriculture.

Singh, V.P., 2016. Handbook of Applied Hydrology, Second Edition. N.Y: McGraw-Hill Education, New York.

Skaggs, R.W., 2017. Coefficients for quantifying subsurface drainage rates. Applied Engineering in Agriculture 33, 793–799. https://doi.org/10.13031/aea.12302

Somorowska, U., Łaszewski, M., 2017. Human-influenced streamflow during extreme drought: Identifying driving forces, modifiers, and impacts in an urbanized catchment in central poland. Water and Environment Journal 31, 345–352. https://doi.org/https://doi.org/10.1111/wej.12249

Sunyer, M.A., Gregersen, I.B., Rosbjerg, D., Madsen, H., Luchner, J., Arnbjerg-Nielsen, K., 2015. Comparison of different statistical downscaling methods to estimate changes in hourly extreme precipitation using RCM projections from ENSEMBLES. International Journal of Climatology 35, 2528–2539. https://doi.org/10.1002/joc.4138

Szabó, B., Gyurkó, D., Weynants, M., Weber, T.K.D., 2019. Web interface for European hydraulic pedotransfer functions (euptfv2). https://doi.org/10.34977/euptfv2.01

Szabó, B., Mészáros, János., 2022. Derivation of soil physical and hydraulic properties. https://doi.org/10.5281/zenodo.6684582

Szabó, B., Mészáros, J., Kassai, P., Braun, P., Nemes, A., Farkas, C., Čerkasova, N., Monaco, F., Chiaradia, E.A., Witing, F., 2022. Solutions to overcome data scarcity. Deliverable D3.2 EU horizon 2020 OPTAIN project, grant agreement no. 862756.

Szabó, B., Weynants, M., Weber, T.K.D., 2021. Updated European Hydraulic Pedotransfer Functions with Communicated Uncertainties in the Predicted Variables (euptfv2). Geosci. Model Dev. 14, 151–175. https://doi.org/10.5194/gmd-14-151-2021

Tan, M.L., Gassman, P.W., Srinivasan, R., Arnold, J.G., Yang, X., 2019. A Review of SWAT Studies in Southeast Asia: Applications, Challenges and Future Directions. Water 11, 914. https://doi.org/10.3390/w11050914

Tan, M.L., Gassman, P., Yang, X., Haywood, J., 2020. A Review of SWAT Applications, Performance and Future Needs for Simulation of Hydro-Climatic Extremes. Advances in Water Resources 143, 103662. https://doi.org/10.1016/j.advwatres.2020.103662

Terrier, M., Perrin, C., Lavenne, A. de, Andréassian, V., Lerat, J., Vaze, J., 2021. Streamflow naturalization methods: A review. Hydrological Sciences Journal 66, 12–36. https://doi.org/10.1080/02626667.2020.1839080

Teutschbein, C., Seibert, J., 2012. Bias correction of regional climate model simulations for hydrological climate-change impact studies: Review and evaluation of different methods. Journal of Hydrology 456-457, 12–29. https://doi.org/10.1016/j.jhydrol.2012.05.052

Thompson, A.L., Baffaut, C., Lohani, S., Duriancik, L.F., Norfleet, M.L., Ingram, K., 2020. Purpose, development, and synthesis of the soil vulnerability index for inherent vulnerability classification of cropland soils. Journal of Soil and Water Conservation 75, 1–11. https://doi.org/10.2489/jswc.75.1.1

Tóth, B., Weynants, M., Nemes, A., Makó, A., Bilas, G., Tóth, G., 2015. New generation of hydraulic pedotransfer functions for Europe. European Journal of Soil Science 66, 226–238. https://doi.org/10.1111/ejss.12192

Tóth, G., Guicharnaud, R.-A., Tóth, B., Hermann, T., 2014. Phosphorus levels in croplands of the European Union with implications for P fertilizer use. European Journal of Agronomy 55, 42–52. https://doi.org/10.1016/j.eja.2013.12.008

Tóth, G., Jones, A., Montanarella, L., 2013. LUCAS Topsoil Survey. Methodology, data and results. Publications Office of the European Union, Luxembourg: https://doi.org/10.2788/97922

U.S. Department of Agriculture Natural Resources Conservation Service, 2009. Part 630 Hydrology, Chapter 7 Hydrologic Soil Groups, in: National Engineering Handbook.

USDA, U.S.D. of A., 1951. Soil survey manual, U.S. Dept. Agriculture Handbook No. 18. Washington, DC.

Vigiak, O., Grizzetti, B., Zanni, M., Aloe, A., Dorati, C., Bouraoui, F., Pistocchi, A., 2020. Domestic waste emissions to european waters in the 2010s. Scientific Data 7. https://doi.org/10.1038/s41597-

020-0367-0

Wagner, P.D., Bieger, K., Arnold, J.G., Fohrer, N., 2022. Representation of hydrological processes in a rural lowland catchment in northern germany using SWAT and SWAT+. Hydrological Processes 36, e14589. https://doi.org/https://doi.org/10.1002/hyp.14589

Waidler, D., White, M., Steglich, E., Wang, S., Williams, J., Jones, C.A., Srinivasan, R., 2009. Conservation Practice Modeling Guide for SWAT and APEX.

Wallace, C., Flanagan, D., Engel, B., 2018. Evaluating the effects of watershed size on SWAT calibration. Water 10, 898. https://doi.org/10.3390/w10070898

Wang, L., Liu, H., 2006. An efficient method for identifying and filling surface depressions in digital elevation models for hydrologic analysis and modelling. International Journal of Geographical Information Science 20, 193–213. https://doi.org/10.1080/13658810500433453

Weber, T.K.D., Weynants, M., Szabó, B., 2020. R package of updated European hydraulic pedotransfer functions (euptf2). https://doi.org/10.5281/zenodo.4281045

Wessolek, G., Kaupenjohann, M., Renger, M., 2009. Bodenphysikalische Kennwerte und Berechnungsverfahren für die Praxis. Bodenökologie und Bodengenese 40, 1–80.

Weynants, M., Montanarella, L., Tóth, G., Arnoldussen, A., Anaya Romero, M., Bilas, G., Borresen, T., Cornelis, W., Daroussin, J., Gonçalves, M.D.C., Haugen, L.-E., Hennings, V., Houskova, B., Iovino, M., Javaux, M., Keay, C.A., Kätterer, T., Kvaerno, S., Laktinova, T., Lamorski, K., Lilly, A., Mako, A., Matula, S., Morari, F., Nemes, A., Patyka, N.V., Romano, N., Schindler, U., Shein, E., Slawinski, C., Strauss, P., Tóth, B., Woesten, H., 2013. European HYdropedological Data Inventory (EU-HYDI). European Commission EUR 26053 – Joint Research Centre – Institute for Environment; Sustainability; EUR – Scientific; Technical Research series – ISSN 1831-9424, Luxembourg. https://doi.org/10.2788/5936

White, M.J., Arnold, J.G., Bieger, K., Allen, P.M., Gao, J., Čerkasova, N., Gambone, M., Park, S., Bosch, D.D., Yen, H., Osorio, J.M., 2022. Development of a Field Scale SWAT+ Modeling Framework for the Contiguous U.S. JAWRA Journal of the American Water Resources Association. https://doi.org/10.1111/1752-1688.13056

White, M.J., Storm, D.E., Busteed, P.R., Smolen, M.D., Zhang, H., Fox, G.A., 2010. A quantitative phosphorus loss assessment tool for agricultural fields. Environmental Modelling & Software 25, 1121–1129. https://doi.org/10.1016/j.envsoft.2010.03.017

Wilby, R.L., Wigley, T.M.L., Conway, D., Jones, P.D., Hewitson, B.C., Main, J., Wilks, D.S., 1998. Statistical downscaling of general circulation model output: A comparison of methods. Water Resour. Res. 34, 2995–3008. https://doi.org/10.1029/98wr02577

Williams, Kannan, N., Wang, X.(Susan)., Santhi, C., Arnold, J., 2012. Evolution of the SCS runoff curve number method and its application to continuous runoff simulation. Journal of Hydrologic Engineering 17, 1221–1229. https://doi.org/10.1061/(ASCE)HE.1943-5584.0000529

Withers, P.J., Jordan, P., May, L., Jarvie, H.P., Deal, N.E., 2014. Do septic tank systems pose a hidden threat to water quality? Frontiers in Ecology and the Environment 12, 123–130. https://doi.org/https://doi.org/10.1890/130131

Yen, H., Park, S., Arnold, J.G., Srinivasan, R., Chawanda, C.J., Wang, R., Feng, Q., Wu, J., Miao, C., Bieger, K., Daggupati, P., Griensven, A. van, Kalin, L., Lee, S., Sheshukov, A.Y., White, M.J., Yuan, Y., Yeo, I.-Y., Zhang, M., Zhang, X., 2019. IPEAT+: A Built-In Optimization and Automatic Calibration Tool of SWAT+. Water 11, 1681. https://doi.org/10.3390/w11081681

Yen, H., White, M., Jeong, J., Arabi, M., Arnold, J., 2015. Evaluation of alternative surface runoff accounting procedures using the SWAT model. International Journal of Agricultural and Biological Engineering 8, 1–15. https://doi.org/10.3965/j.ijabe.20150803.833

Yuan, Y., Chiang, L.-C., 2015. Sensitivity analysis of SWAT nitrogen simulations with and without in-stream processes. Archives of Agronomy and Soil Science 61, 969–987. https://doi.org/10.1080/03650340.2014.965694

Zhang, X., Srinivasan, R., Arnold, J., Izaurralde, R.C., Bosch, D., 2011. Simultaneous calibration of surface flow and baseflow simulations: A revisit of the SWAT model calibration framework. Hydrological Processes 25, 2313–2320. https://doi.org/10.1002/hyp.8058

Zhu, Y., Chen, Y., Ali, M.A., Dong, L., Wang, X., Archontoulis, S.V., Schnable, J.C., Castellano, M.J., 2021. Continuous in situ soil nitrate sensors: The importance of high-resolution measurements across time and a comparison with salt extraction-based methods. Soil Science Society of America Journal 85, 677–690. https://doi.org/10.1002/saj2.20226