Manual and Guide of the *River Architect* Tool Kit

Stream Assessment, Analysis and Design

University of California, Davis | January 2019

written by Sebastian Schwindt

Contents

Part I
Getting started

1 Signposts

River Architect serves for the GIS-based planning of habitat enhancing stream restoration features regarding their lifespans, design characteristics, optimum placement in the terrain, and ecological benefit. A main graphical user interface (GUI) provides five modules for generating lifespan and design maps, action (optimum lifespan) maps, terrain modification (terraforming) assessment of digital elevation models (DEM), habitat evaluation, and project cost-benefit analyses.

Lifespan maps indicate the expected longevity of restoration features as a function of terrain change, morphological characteristics, and 2D hydrodynamic modeling results. **Design maps** are a side product of lifespan mapping and indicate required feature dimensions for stability, such as the minimum required size of angular boulders to avoid their mobilization during floods (see Part II and Schwindt et al., 2019).
Action maps result from the comparison of lifespan and design maps of multiple restoration features and assign features with the highest longevity to each pixel of a raster. Thus, the *Action Planner* module assess optimum features as a function of highest lifespans among comparable feature groups such as terraforming or vegetation planting species(see Part III).
The **Terrain Modification** module prepares DEMs of particular reaches for or after the virtual application of stream restoration features. Moreover, this module can compare ”exiting” (pre-project) and ”with implementation” (post-feature application) Rasters to determine required earth movement (terraforming) works (see Part IV).
The **Habitat Evaluation** module applies a user-defined set of discharges for the spatial evaluation of the habitat suitability index (HSI). The hydraulic habitat suitability results from 2D hydrodynamic numerical model outputs of flow depth and velocity. In addition, the option ”cover” can be used to assess cobble, boulder, vegetation and streamwood habitats (see Part V).
The **Project Maker** module evaluates the costs for gained area in usable habitat for target fish species and lifestages. A unit cost workbook provides relevant costs and the gain in usable habitat area results from the *Habitat Evaluation* module. The usage of the *Project Maker* module is explained in Part VI.
A set of **Tools** provides console Python scripts to generate required input files and to support terraforming drawing efforts.

Fig. 1 shows a flowchart for the application of *River Architect*’s modules and external input data for designing habitat enhancement projects. The modules and tool-scripts can also be individually applied for other purposes than suggested in the flowchart.

The procedure of project design following the flowchart involves the following steps:

1. Generate a terrain elevation model (DEM).
2. Determine relevant discharges for 2D hydrodynamic modeling:
   - At least three annual discharges describing the ”most of the time” - situation of the considered river for habitat evaluation assessments. *River Architect*’s *Tools* contain scripts for generating flow duration curves from gaging station data.
   - At least three flood discharges against which potential restoration features have to withstand (determine lifespan intersects).
3. Run a 2D hydrodynamic model (steady) with all determined discharges to generate hydraulic snap-shots of the river.

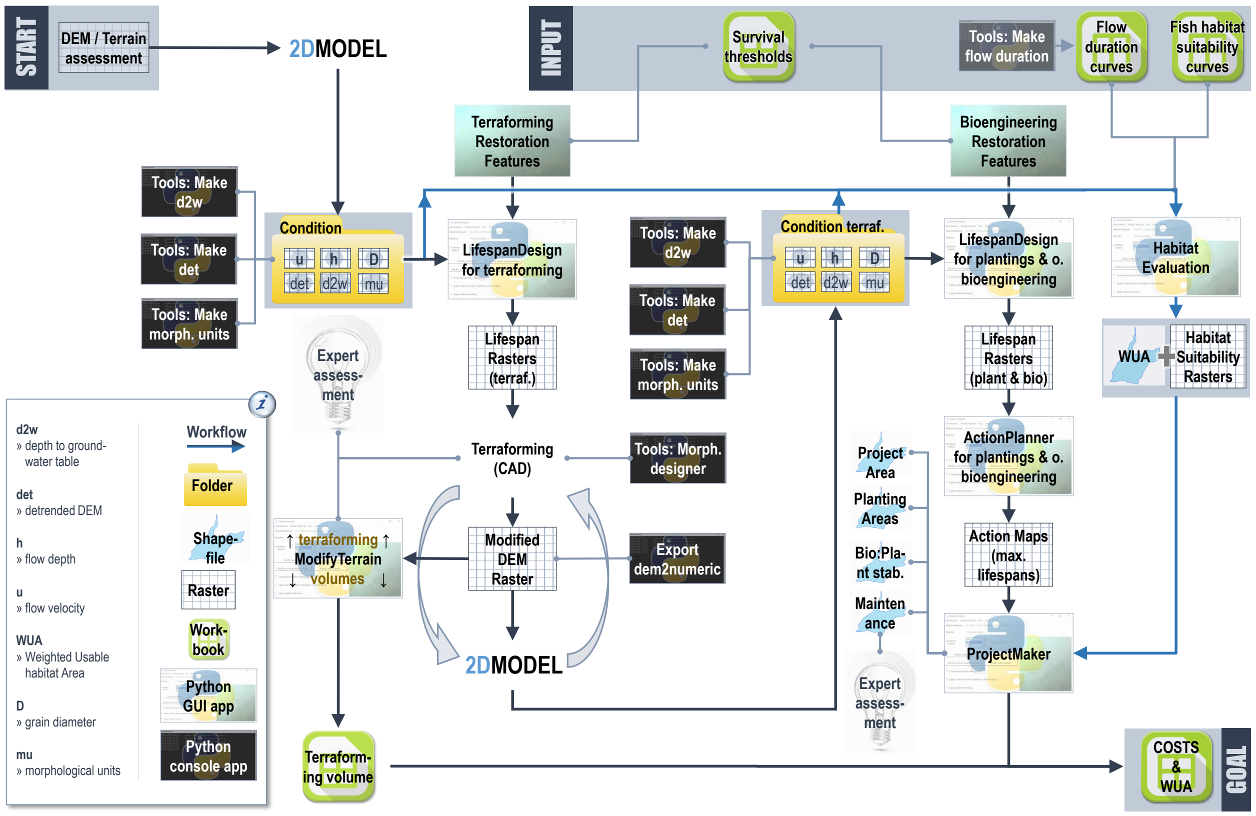


Figure 1: Flowchart for designing habitat enhancing stream restoration projects with the *River Architect*'s modules.

4. Compile a raster database of existing (pre-project) river conditions, including:
   - A detrended digital elevation model (see *River Architect*'s *Tools*);
   - Flow depth and velocity for multiple discharges Rasters from 2D hydrodynamic modeling (see Sec. 5);
   - A substrate map (`dmean` for metric or `dmean_ft` for U.S. customary units); relevant methods are described in Detert et al. (2018); Stähly et al. (2017); Jackson et al. (2013);
   - Datasets that can be used to assess design feature stability, such as side channel design criteria (e.g., Sec. 10.11);
   - Terrain change Rasters (Topographic Change Detection or DEM differencing according to Wyrick and Pasternack, 2016);
   - A depth to groundwater table Raster (see *River Architect*'s *Tools*);
   - A morphological unit Raster (see *River Architect*'s *Tools* applying methods from Wyrick and Pasternack, 2014).
5. Apply *LifespanDesign* module to framework (terraforming) features.
6. *LifespanDesign* maps and expert assessment serve for the identification of relevant terraforming features.
7. Iterative terraforming application (if relevant):
   - Use the *ModifyTerrain* module for systematic terrain grading or broaden the river bed, however, adaptations are required and computer-aided design must be manually applied to modify the existing (pre-project) DEM, where the *Tools* provide assistance for designing self-sustaining pool-riffle channels.

- Re-compile the flow depth an velocity maps (re-run 2D model) with the modified DEM, where the *Tools* provide routines for converting between raster types.
- Verify the suitability of the modified DEM (e.g., barrier height to ensure flood safety); if the verification show weaknesses, adapt the terraforming and re-compile the flow depth and velocity maps until terraforming is satisfactory.
- Use the *ModifyTerrain* module for comparing pre- and post project DEMs to determine required excavation and fill volumes.

8. Apply the *LifespanDesign* module to vegetation plantings and (other) bioengineering features based on the terraformed DEM (or the original DEM if no terraforming applies).
9. Use the *MaxLifespan* module to identify best performing (highest lifespan) vegetation plantings and bioengineering features.
10. If the soils are too coarse, apply the maintenance feature of ”incorporate fine sediment in soils” to ensure that planned vegetation plantings can grow.
11. If gravel augmentation methods are applicable: Consecutively apply the *LifespanDesign* and *MaxLifespan* module to maintenance features to foster self-sustaining, artificially created morphological patterns within the terraforming process.
    If gravel is added in-stream, re-run the numerical model for the assessment of gravel stability with the *LifespanDesign* module and the combined habitat suitability with the *HabitatEvaluation* module to compare the Weighted Usable Area (habitat) before and after stream restoration.
12. Use the *HabitatEvaluation* to assess the ”existing” (pre-project) and ”with implementation” (post-project) habitat suitability in terms of weighted usable habitat area (WUA).
13. Use the *ProjectMaker* to calculate costs, net gain in WUA, and their ratio as a metric defining the project tradeoff.

The working principles of the *LifespanDesign*, *MaxLifespan*, *ModifyTerrain*, *HabitatEvaluation*, and *ProjectMaker* modules are explained in chapters II, III, IV, V, and VI, respectively. The differentiation between terraforming (framework), planting and other bioengineering, and maintenance features is described in Sec. 4. The correct installation of the *River Architect* package and setting the good code environment is explained in the following Sec. 2.

2 Package structure, requirements and logfiles

2.1 File structure

The main directory (`/RiverArchitect/`) contains the documentation file, a `Tools` folder, and a template folder tree named `/NewRiver/`. This template folder contains the program launcher named `LAUNCH_River Architect_WINx64.bat` and the Python 2.7 file `stream_restoration_gui.py` with routines called by the launcher. The *River Architect* modules are located in sub-folders of `/NewRiver/`. Thus, the master folder (`/RiverArchitect/`) includes the following files and directories:

- **.site_packages**
  Contains adapted third-party Python packages and own packages
  - `openpyxl`
    Contains a modified version of the openpyxl (version 2.5.2) package for *River Architect*
  - `riverpy`
    Package-own python scripts with recurring routines and classes that are used in multiple modules.

    * `cDefinitions.py` contains inter-module information of reach and feature keywords.
    * `cGravel.py` contains subfeatures of the Gravel augmentation-feature in `cFeatures`.
    * `cPlants.py` contains subfeatures of the Plantings-feature in `cFeatures` and the `ModifyTerrain` module.
    * `cTerrainIO.py` applies on the openpyxl package the assessment of reach information and for writing calculated volumes to xlsx output.
    * `fGlobal.py` provides functions that are required in this module and the other modules in several classes.

- **00_Documentation**
  Contains this manual.

- ***NewRiver*/01_Conditions**
  This folder contains *`condition`* folders with parameter Rasters. The condition name begins with a 4-digit year number (e.g., `2008`), optionally followed by a 3-characters reach ID (e.g., `xyz`) and a feature layer indicator (e.g., `lyr01` for terraforming features). The syllables are separated by an underscore. The process of defining of reaches is explained in Sec. 6.1 and Sec. 18.3. The setting of feature layers is introduced in Sec. 6.2.

- **Module (folder): *`NewRiver`*`/LifespanDesign`**
  Lifespan and Design analyses of restoration features (see Manual Chapter II).
  - `Output` folder with sub-folders for `Mapping` and `Rasters` from individual module runs.
  - `Products` folder with sub-folders `Layouts`, `Maps` and `Rasters` for manually storing results from relevant module runs.
  - `.cache` folder occurs temporarily when the module is executed.
  - `.templates` folder should not be modified and contains input (`*.inp`) files; if required, the module includes routines for changing the input files.
  - `cFeatureLifespan.py` contains stream restoration features classes with pointers to parameters and threshold values.
  - `cLifespanDesignAnalysis.py` contains GIS-based functional core for processing `Raster` files.
  - `cMapLifespanDesign.py` contains routines creating layout files (`mxd`) and `PDF` maps.
  - `cParameters.py` contains the parameter input core with pointers to `Rasters` and `Raster` names.
  - `cReadInpLifespan.py` contains classes that read input data from `*.inp` files.
  - `cThresholdDirector.py` provides the ThresholdDirector class for reading threshold values from spreadsheet “thresholds”.
  - `feature_analysis.py` coordinates class instantiations and function calls.
  - `lifespan_design_gui.py` is a standalone script that creates the graphical user interface (GUI) for running the *LifespanDesign* module.
  - `LAUNCH_Windows_x64.bat` is a batchfile that runs `lifespan_design_gui.py`.

- **Module (folder): *`NewRiver`*`/MaxLifespan`**
  Action planner in folder `MaxLifespan` (see Manual Chapter III)
  - `Output` folder with sub-folders for `Layouts`, `Maps` and `Rasters` from individual module runs.
  - `Products` folder with sub-folders `Layouts`, `Maps` and `Rasters` for manually storing results from relevant module runs.
  - `.cache` folder occurs temporarily when the module is executed.

- `.templates` folder contains additional Rasters, which are required by this module; other Rasters are loaded from *01_Conditions*.
- `action_gui.py` is a standalone script that creates the graphical user interface (GUI) for running the *MaxLifespan* module.
- `action_planner.py` coordinates class instantiations and function calls.
- `cActionAssessment.py` contains the GIS-based functional core that identifies optimum lifespans and associated features by processing lifespan/design `Raster` and `shape` files.
- `cFeatureActions.py` contains pointers to stream restoration feature data in the *LifespanDesign* module.
- `cMapActions.py` coordinates layout and action map creation.
- `cReadActionInput.py` contains functions for reading `*.inp` files from the `.templates` folder.
- `LAUNCH_Windows_x64.bat` is a batchfile that runs `action_planner_gui.py`.

- **Module (folder): *`NewRiver`*`/ModifyTerrain`**
  Performs half-automated terrain modifications and calculates excavation / fill volumes of terraforming features (see Manual Chapter IV).

  - `Input` folder containing optional modified DEMs for volume difference assessment.
  - `Output` folder with sub-folders `Logfiles` and `Rasters` from individual module runs.
  - `Products` folder with sub-folders `Logfiles` and `Rasters` for manually storing results from relevant module runs.
  - `.cache` folder occurs when the module is executed.
  - `.templates` folder contains additional Rasters, which are required by this module; other Rasters are loaded from `01_Conditions`.
  - `cMapModifyTerrain.py` provides routines for the layout creation and mapping of modified DEMs and volume/terrain elevation differences.
  - `cModifyTerrain.py` contains GIS-based functional core for modifying DEM `Raster` files and calculating volumes using ArcGIS “3D” extension.
  - `modify_terrain_gui.py` is a standalone script that creates the graphical user interface (GUI) for running the *ModifyTerrain* module
  - `LAUNCH_Windows_x64.bat` is a batchfile that runs `modify_terrain_gui.py` on Windows x64.

- **Module (folder): *`NewRiver`*`/HabitatEvaluation`**
  Creates Habitat Suitability Index Rasters / maps and quantifies weighted usable area for target fish species and a user-defined range of discharges (see Manual Chapter V).

  - `CHSI` contains subfolders with with composite habitat suitability index Rasters for pre- and post-project `conditions`.
  - `FlowDurationCurves` contains workbooks with flow duration curves (exceedance probabilities). Refer to the external `Tools` to generate appropriate `Spreadsheets`.
  - `HSI` contains subfolders with with habitat suitability index `Rasters` for pre- and post-project `conditions`.
  - `WUA` contains result workbooks with WUA values for examined `conditions`. The `Rasters` subfolder contains the associated composite habitat suitability Rasters.
  - `.cache` folder occurs temporarily when the module is executed.
  - `.templates` folder contains spreadsheet templates for the quantification of weighted usable area and the definition of fish species, lifestages and associated habitat suitability curves.

  - `cFish.py` contains the Fish class that reads characteristic species and lifestage data from `.templates/Fish.xlsx`.
  - `cHabitatIO.py` uses the openpyxl package to read and write data from and to `xlsx` files, respectively.
  - `cHSI.py` contains the CHSI, HHSI and FlowAssessment classes to calculate composite habitat suitability Rasters, hydraulic habitat suitability Rasters and interpolating the annual flow duration of considered discharges.
  - `habitat_gui.py` contains the MainGui class of this module.
  - `sub_gui_hhsi.py` opens a new GUI window to create hydraulic habitat suitability Rasters and determine associated annual flow duration.
  - `LAUNCH_Windows_x64.bat` is a batchfile that runs `habitat_gui.py` on Windows x64.

- **Module (folder): *`NewRiver`*`/ProjectMaker`**
  Applies on results from *MaxLifespan* and *HabitatEvaluation*, as well as manual inputs to calculate project cost-benefit metrics (see Manual Chapter VI).

  - `.cache` folder occurs temporarily when the module is executed.
  - `.templates` folder contains a template folder tree and template workbooks with unit cost tables, as well as sample application data that illustrate potential results of the module.
  - `cIO.py` uses the openpyxl package to read and write data from and to `xlsx` files, respectively.
  - `cWUA.py` applies on *HabitatEvaluation* results, in particular CHSI Rasters for calculating WUA in the project area.
  - `fFunctions.py` contains module-specific functions.
  - `project_maker_gui.py` contains the MainGui class of this module.
  - `s20_plantings_delineation.py` applies on *MaxLifespan* products for assessing most suitable vegetation plantings within the project area.
  - `s21_plantings_stabilization.py` applies on *MaxLifespan* products and user-defined input parameters for mapping bioengineering futures required in order stabilize vulnerable vegetation plantings.
  - `s40_compare_wua.py` applies on *HabitatEvaluation* CHSI rasters used in `cWUA.py` for assessing the weighted usable habitat for a target fish species and lifestage within the project area.
  - `LAUNCH_Windows_x64.bat` is a batchfile that runs `habitat_gui.py` on Windows x64.

- **Folder: `Tools`**
  Applies on results from *MaxLifespan* and *HabitatEvaluation*, as well as manual inputs to calculate project cost-benefit metrics (see Manual Chapter VI).

  - `.cache` folder occurs temporarily when the module is executed.
  - `.templates` folder contains a template workbooks for multiple purposes.
  - `Products` folder contains results of any script in this folder.
  - `cDepth2Groundwater.py` provides routines for calculating depth to groundwater Rasters.
  - `cDetrendedDEM.py` provides routines for generating detrended DEM Rasters.
  - `cHydraulic.py` contains a Hydraulic class with routines for calculating cross-section-averaged flow characteristics.
  - `cInputOutput.py` contains classes required for reading and writing data, as well as calculation progress logging.

- `cMorphUnit.py` provides routines for calculating instream morphological units (Wyrick and Pasternack, 2014).
- `cPoolRiffle.py` provides routines for designing self-sustaining pool-riffle channels.
- `fTools.py` is a set of functions used by other Python applications within this folder.
- `make_annual_peak.py` prepares required input data for statistic flow analyses and with the U.S. Army Corps of Engineers' `HEC-SPP` software.
- `make_d2w.py` calculates depth to groundwater Rasters (uses `cDepth2Groundwater.py`).
- `make_det.py` calculates detrended DEM Rasters (uses `cDetrendedDEM.py`).
- `make_flow_duration.py` creates flow duration curves (annual averages) for the assessment of WUA.
- `make_mu.py` calculates instream morphological unit Rasters (uses `cMorphUnit.py`).
- `morphology_designer.py` creates design tables for self-sustaining pool-riffle channels (uses `cHydraulic.py` and `cPoolRiffle.py`).
- `run_make_....bat` are a batchfiles that run `make_....py` on Windows x64.
- `run_morphology_designer.bat` are a batchfiles that run `morphology_designer.py` on Windows x64.

2.2 Additional River Architect Tools

Beyond the fully automated generation of many Raster and shapefile types for stream restoration, an additional Toolbox is available that helps to prepare input files such as detrended digital elevation models, depth to groundwater Rasters or flow duration curves. Moreover, routines for the hydraulic design of pool-riffle sequences or flood analysis are available, where the flood analysis applies on the U.S. Army Corps of Engineers' `HEC-SPP` software. The *Tools* routines are located in `RiverArchitect/Tools/`. Using the tool routines (*Python* files) requires basic knowledge of *Python* and manual modifications of particular codes.

2.3 Requirements

The execution of *River Architect* requires the following external packages to be installed, which are part of the standard ArcGIS – `python` installation: arcpy, arcpy.sa, argparse, glob, logging, os, shutil, subprocess (not mandatory, also works without this package), sys, Tkinter, and __future__. Furthermore, the *River Architect* package requires ArcGIS' "Spatial Analyst" and "3D" (ModifyTerrain only) extensions.
The call of the GUIs from the individual module batch files (`LAUNCH_Windows_x64.bat`) is designed for Windows (x64) but can be easily changed to UNIX operating systems given that ArcGIS is installed. However, the *River Architect* package was currently only tested on Windows platforms.
Any folder beginning with a "." as for example `.cache`, `.idea` or `.ReferenceLayouts` must not be modified or assessed by any other program, in particular during the execution of package methods. Files stored in `.templates` folders are directly called by the GUIs if user definitions are admitted.
At the end of an execution, the applied modules have created their output folders, which are indicated in the command prompt.
A spreadsheet editing software such as *Excel* or *OpenOffice* is required for modifications of user definitions.

2.4 Logfiles

Logfiles `.log` are created in the module directories during every run task. These files contain time-stamped terminal messages of program activities, warnings and error messages. Thus, logfiles enable the user to review process duration and to trace back problems. The handling of potential errors and warning messages are listed in Chapter VIII with descriptions of problem sources and solutions.

3 Getting started (GUI)

3.1 Prepare file structure

The first step is to copy the template file structure (`NewRiver` folder) in *River Architect* and to rename the copy corresponding to the name of the analyzed river.

3.2 Program environment setup and batchfile modification

The package is designed for an ArcGIS Python **x64** interpreter (ArcGIS 10.5 or higher – older versions use the standard ArcGIS `python.exe`). The appropriate Windows (x64) `python` interpreter is typically stored in `"C:\Python27\ArcGISx64XX.X\python.exe"`. Please note the importance of using the **x64** version: The 32-bit version will result in `ERROR 999998: Unexpected Error`.

Before launching the *River Architect* package for the first time, the batch files need to be adapted to the system environment. On Windows, set the batch file environment as follows:

1. Right-click on `LAUNCH_River Architect_WIN64.bat` and choose *Edit with Texteditor* or *Open with ...* and choose a *Texteditor* software.
2. Check, and if necessary, replace the path to the good python interpreter:
   Default: `"C:\Python27\ArcGISx64XX.X\python.exe"`
3. The string `%cd%` automatically points to the folder where the GUI is located.
4. Save `LAUNCH_River Architect_WINx64.bat` and close *Texteditor*.
5. Set default application to open input file type documents (`*.inp` files):
   Go to folder `...\River Architect\LifespanDesign\.templates\` *and right-click on* `mapping.inp` *to access the menu* `Open with ....` *Choose any text editor, such as* `Notepad`*,* `Texteditor` *or* `Notepad++` *and click* `OK`.

Apply the procedure repetitively to the `LAUNCH_Windows_x64.bat` stored in the modules sub-folders for setting individual launches. Adapt the directories of the GUI creator according to the corresponding module GUI maker ending on `..._gui.py`.
On UNIX platforms (Apple or Linux), make sure that the python interpreter is version 2.7 and that it can import the arcpy package. Then, open the system terminal, navigate to the directory where the package is installed (location of `.py` files) and type: `./LAUNCH_UNIX.sh`.

After editing the batch files, launch *River Architect* by double-clicking on
`LAUNCH_River Architect_WINx64.bat`.

3.3 Welcome GUI

The *River Architect* package starts in a GUI now (Fig. 2), which contains three buttons to launch one of the package modules. Please note that the main window will close and a new GUI window will open. The options of the module GUIs are described in the corresponding chapters (see “Quick GUI to ...”).
Alternatively, modules can be individually launched by double-clicking on `LAUNCH_Windows_x64.bat` in the corresponding module folders. Moreover, the lifespan and design map module can be executed as a standalone python script, which is described in the module chapter II.

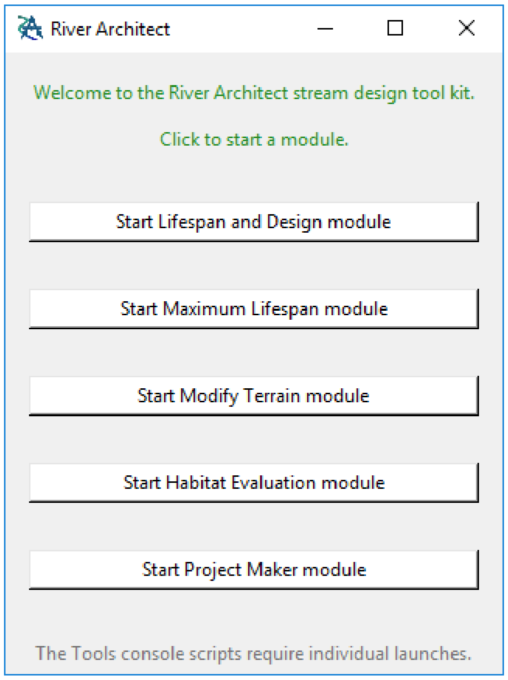

River Architect
Welcome to the River Architect stream design tool kit.
Click to start a module.
Start Lifespan and Design module
Start Maximum Lifespan module
Start Modify Terrain module
Start Habitat Evaluation module
Start Project Maker module
The Tools console scripts require individual launches.

Figure 2: `River Architect` GUI start up window.

4 Restoration features

The *River Architect* package differentiates between feature layers that actively modify the terrain (terraforming features), vegetation plantings features as well as (soil-) bioengineering features that provide direct aid for habitat enhancement or stabilize terrain modifications, and features that maintain artificially created, habitat enhancing morphological units (maintenance features). The features can be modified in the *LifespanDesign* module's thresholds workbook (`.../RiverArchitect/LifespanDesign/.templates/threshold_values.xlsx`), which can be open from the GUI's whenever needed. Changes in this workbook should limit to cells with `INPUT`-type formatting and only `Feature Names` and `FeatureIDs` of vegetation plantings should be modified. Other modifications may cause calculation instabilities or program crashes. The following list provides an overview on default features, where *shortnames* occur in output file names of Rasters, layouts, PDF-maps, and spreadsheets and plantings

- **Terraforming features** modify the terrain elevation:
  - Backwater, representative for swale and slackwater creation (*shortname: backwt*)
  - Berm Setback (Widening, *shortname: widen*)
  - Grading of terrain (Bar and Floodplain Lowering *shortname: grade*)
  - Side Cavities (Bank Scalloping or Groins, *shortname: sideca*)
  - Side Channels, representative for Anabranches, Multithread- or Anastomosed Channels and Flood Runners (*shortname: sidech*)
- **Plantings features** are up to four vegetation plantings that can be defined in the *LifespanDesign* module's `threshold_values.xlsx` workbook. The default plant species are (can be modified, except for the fields that are marked for input in the thresholds workbook):
  - (Fremont) Cottonwood (*Populus Fremontii*, *shortname: cot*)
  - Box Elder (*Acer Negundo*, *shortname: box*)

  - White Alder (*Alnus Rhombifolia*, *shortname: whi*)
  - Willows (*Salix Goodingii / various*, *shortname: wil*)

- **Toolbox features** have a direct effect on habitat suitability and stabilize terrain modifications (framework features). The features are considered:
  - Engineered Log Jams and instream wood placement including rootstocks (*shortname: elj*)
  - Angular boulders (rocks), representative for bolder or rock placements (*shortname: rocks*)
  - Other soil-bioengineering for terrain (slope) stabilization comprise for instance brush layers and / or fascines
- **Complementary features** enhance the stability of artificial river systems that result from framework and toolbox features, such as:
  - Sediment Replenishment (instream, *shortname: gravin*)
  - Stockpiles of gravel or Gravel Augmentation (on banks or floodplain, *shortname: gravou*)
  - Incorporation of Fine Sediment in soils to increase the survivorship of plantings (*shortname: fines*)

In addition, the package provides the option of limiting restoration feature maps to zones of low habitat suitability (see details in the descriptions of the *HabitatEvaluation* module, part V).

5 Conditions, input Rasters and folder management

A *condition* folder filled with Rasters corresponding to the analyzed situation needs to be prepared in `RiverArchitect/01_Conditions/` folder. For example, if feature lifespans need to be assessed based on the situation in the year 2008, the condition folder name is `2008` and the Raster input folder is `/01_Conditions/2008/`. The *`condition`* name may NOT include any SPACE character and the initial condition should correspond to a 4-digit year.
The five modules provide options to process input data according to the defined starting condition year. The modules create output folders beginning with the 4-digit year and automatically append feature layer (cf. Sec. 6.2) and reach (cf. Sec. 18.3) information.

The input Rasters need to be in (ArcGIS) GRID format, notably, a *`Raster_name`*`.aux.xml` file and an *`Raster_name`* folder with *adf* and *xml* files. Depth Raster names must start with `h` and velocity Raster names must start with `u`, followed by a three digit discharge `QQQ`, which is independent of the unit system. If the discharge is larger than 1000 cfs (or 1000 m^3/s), the letter `k` must be appended. For example, a flow depth Raster associated with a discharge of 55 cfs needs to be called `h055` and a velocity Raster associated with a discharge of 11000 cfs needs to be called `u011k`. Likewise, a flow depth Raster associated with a discharge of 55 m^3/s needs to be called `h055`. Thus, the Raster names ignore discharge digits after the decimal point for discharges smaller than 1000 cfs or m^3/s and three digits to the left of the decimal point for discharges larger than 1000 cfs or m^3/s. Moreover, every flow depth Raster requires a matching velocity Raster and vice verse; e.g., `h055` requires a Raster called `u055`.

The arcpy package does not consider pixels with `noData` values and the *River Architect* package has its own routines to handle `noData` during the calculation. To ensure computation stability and pertinence, the hydraulic input Rasters (flow depth and flow velocity) need to be fitted manually to set assign zero values to `noData` pixels, even in the absence of water. This can be achieved with the following formula either in python using the `arcpy.sa` package or in *ArcGIS* Desktop using the `Raster Calculator` (for discharges larger than 1000 cfs or m^3/s): `Con((IsNull("hXXXk")== 1), (IsNull("hXXXk")* 0), Float("hXXXk"))` for flow depth and `Con((IsNull("uXXXk")== 1), (IsNull("uXXXk")* 0), Float("uXXXk"))` for flow velocity. The `XXX` values indicate that the formulae need to be applied to all `h` and `u` Rasters.

Relevant Raster names for calculation are defined in an input file (`.inp`) of the *LifespanDesign* module (see Sec. 11.1 for details and definitions). More Rasters indicating morphological units (e.g. Wyrick and Pasternack, 2014) or topographic change (e.g. Carley et al., 2012) as well as a detrended digital elevation model (DEM), surface grain size estimate and a depth to groundwater Raster are (optionally) required. The input preparation *Tools* `make_d2w.py`, `make_det.py` and `make_mu.py` can be used to generate depth to groundwater, detrended DEM and morpholoical units Rasters, respectively.
A base case is provided with the *River Architect* installation files. The input files of the base case (defined in the *.inp* file) represent a patch of the lower Yuba River in 2008. The base case includes a set of Rasters for flood scenarios corresponding to flood return periods of <1.0 year, 1.2 years, 2.5 years, 4.7 years and 12.7 years, as well as a couple of annual discharges for habitat assessments. The below listed Rasters are available in `01_Conditions/2008/` for the base case *`condition`* = `2008`. Formatted font indicates optional Rasters, which are however recommended to use because they significantly increase the pertinence of lifespan maps; Rasters written in `Courier New` font are mandatory. The Raster names correspond to the above-described naming conventions.

Flow velocity (in fps):			**Flow depth (in ft):**		
-	`u530`	for habitat evaluation	-	`h530`	for habitat evaluation
-	`u700`	for habitat evaluation	-	`h700`	for habitat evaluation
-	`u880`	for habitat evaluation	-	`h880`	for habitat evaluation
-	`u001k`	for habitat evaluation and min. floods	-	`h001k`	for habitat evaluation and min. floods
-	`u005k`	1.2-years flood velocities	-	`h005k`	1.2-years flood depths
-	`u021k`	2.5-years flood velocities	-	`h021k`	2.5-years flood depths
-	`u042k`	4.7-years flood velocities	-	`h042k`	4.7-years flood depths
-	`u084k`	12.7-years flood velocities	-	`h084k`	12.7-years flood depths

Topographic change (in ft):			**Depth two water table (in ft):**		
-	dodfill	2006/2008–2014 deposition heights	-	d2w	referring to base flows of 530–880 cfs
-	dodscour	2006/2008–2014 scour depths			

Morphological Units (string):			**D$_{mean}$ valley (in ft):**		
-	`mu`	generated with `make_mu.py`	-	`dmean_ft`	mean valley grain size

DEM (in ft a.s.l.):			**DEM detrended (in ft):**		
-	`dem`	Digital Elevation Model	-	`dem_detrend`	`make_det.py`

Side channel			**Wildcard**		
-	sidech	Side channel delineation	-	wild	0/nodata (= off) and 1 (= on) values for any purpose to confine analysis

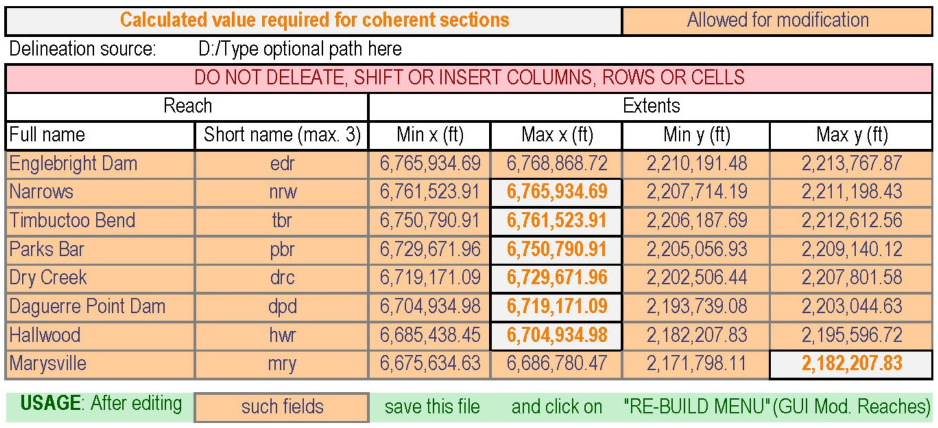
Some parameters, such as the dimensionless bed shear stress or the mobile grain size, can be directly computed from the flow velocity, depth, and present grain size. Additional input Rasters could be used for every parameter to shorten calculation duration, but this approach required large storage capacity on the hard disk and it is less flexible regarding computation methods. Therefore, the *River Architect* uses its own routines for calculating parameters such as the dimensionless bed shear stress or mobile grain sizes.

6 Define Reaches and Features

6.1 Set Reaches

Particular rivers or reaches for the analysis can be defined from the *LifespanDesign* and *ModifyTerrain* GUIs, referring to:
`ModifyTerrain/.templates/computation_extents.xlsx`

The *ModifyTerrain* and *LifespanDesign* modules provide options for reach differentiation and limit calculations to defined particular reaches. These limitations are automatically used by the other modules. This subdivision of the computation domain enables the analysis of up to eight reaches per copy of *River Architect*. Fig. 3 illustrates the Reach Menu of the *ModifyTerrain* GUI. Changes can be effected by clicking on the `Reaches` dropdown menu and then `DEFINE REACHES`, or directly in the folder `ModifyTerrain/.templates/`. Detailed instructions are provided in Sec. 18.3.



Calculated value required for coherent sections				Allowed for modification	
Delineation source:	D:/Type optional path here				
DO NOT DELEATE, SHIFT OR INSERT COLUMNS, ROWS OR CELLS					
Reach		Extents			
Full name	Short name (max. 3)	Min x (ft)	Max x (ft)	Min y (ft)	Max y (ft)
Englebright Dam	edr	6,765,934.69	6,768,868.72	2,210,191.48	2,213,767.87
Narrows	nrw	6,761,523.91	**6,765,934.69**	2,207,714.19	2,211,198.43
Timbuctoo Bend	tbr	6,750,790.91	**6,761,523.91**	2,206,187.69	2,212,612.56
Parks Bar	pbr	6,729,671.96	**6,750,790.91**	2,205,056.93	2,209,140.12
Dry Creek	drc	6,719,171.09	**6,729,671.96**	2,202,506.44	2,207,801.58
Daguerre Point Dam	dpd	6,704,934.98	**6,719,171.09**	2,193,739.08	2,203,044.63
Hallwood	hwr	6,685,438.45	**6,704,934.98**	2,182,207.83	2,195,596.72
Marysville	mry	6,675,634.63	6,686,780.47	2,171,798.11	**2,182,207.83**
USAGE: After editing	such fields	save this file	and click on	"RE-BUILD MENU" (GUI Mod. Reaches)	

Figure 3: Spreadsheet with reach definitions (stored in `ModifyTerrain/.templates/computation_extents.xlsx`).

If the workbook is accidentally deleted or irreparable, incorrect modifications were made, there is a backup copy available:
`ModifyTerrain/.templates/computation_extents - Copy.xlsx`

6.2 Define or modify features

The *LifespanDesign* module uses a spreadsheet to read threshold value for feature failures (cf. Sec. 6.2). This spreadsheet additionally defines feature names and features IDs, which can be modified if needed. The spreadsheet can be accessed either by clicking on the *LifespanDesign* GUI's The "Modify survival threshold values" button or directly from:
`/RiverArchitect/LifespanDesign/.templates/threshold_values.xlsx`

Modifications of feature IDs and names require careful consideration because the packages apply analysis routines as a function of the features *Python* classes. Changing feature names and parameters and IDs only provides the possibility of renaming features and modifying threshold values, as well as the unit system. The feature IDs are internal abbreviations, which also determine the names of output Rasters, shapefiles, and maps. Editing feature evaluations (e.g., adding an analysis routines) requires changes in the *Python* code as explained in Sec. 12.5.
The workbook enables changing vegetation plantings species in columns `J` to `M`. The following columns are associated with distinct feature layers (cf. definitions in Sec. 4) in the workbook:

- Framework features: Columns `"E"`, `"F"`, `"G"`, `"H"`, `"I"`.
- Plant features: Columns `"J"`, `"K"`, `"L"`, `"M"`.
- Other Bioengineering features: Columns `"N"`, `"O"`, `"P"`.
- Maintenance features: Columns `"Q"`, `"R"`, `"S"`.

Detailed instructions for the usage of `threshold_values.xlsx` is provided in Sec. 6.2 and more information on threshold values is provided in Sec. 10. If the spreadsheet is accidentally deleted or irreparable, incorrect modifications were made, there is a backup copy available:
`/RiverArchitect/LifespanDesign/.templates/threshold_values - Copy.xlsx`

Part II
Feature lifespan and design assessment

7 Introduction to lifespan and design mapping

Survival thresholds applied to a sequence of habitat enhancement features, can be spatially compared with hydraulic and sediment data as a result of 2D numerical modeling. Modeled discharges can be associated with flood return periods that determine feature lifespans. The resulting lifespan maps indicate the temporal stability of particular stream design features and techniques. Areas with particularly low or high lifespans help planners optimize the design and positioning of features. Moreover, discharges related to specific flood-return periods enable probabilistic estimates of the longevity of particular features. Following these procedures described by Schwindt et al. (2019), the *LifespanDesign* module creates `rasters`, `mxd`-layouts and `pdf`-maps of the following types:

- **Lifespan maps** qualitatively indicate areas where features make sense and the associated feature lifetime estimate in years.
- **Design maps** indicate dimensional requirements for achieving the success of a feature, e.g., the minimum required block (grain) sizes for angular boulders (rocks) stability.

This chapter explains the usage of the *LifespanDesign* module and it is structured as follows:

Section 8:	Quick Guide to the application of the code using GUI with descriptions of required input rasters and alternative launch options.
Section 9:	Physical explanations of relevant parameters.
Section 10:	Explanations of hypotheses and restoration features.
Section 11:	Detailed explanation of input file usage.
Section 12:	Detailed explanations of coding conventions with descriptions of extension possibilities.

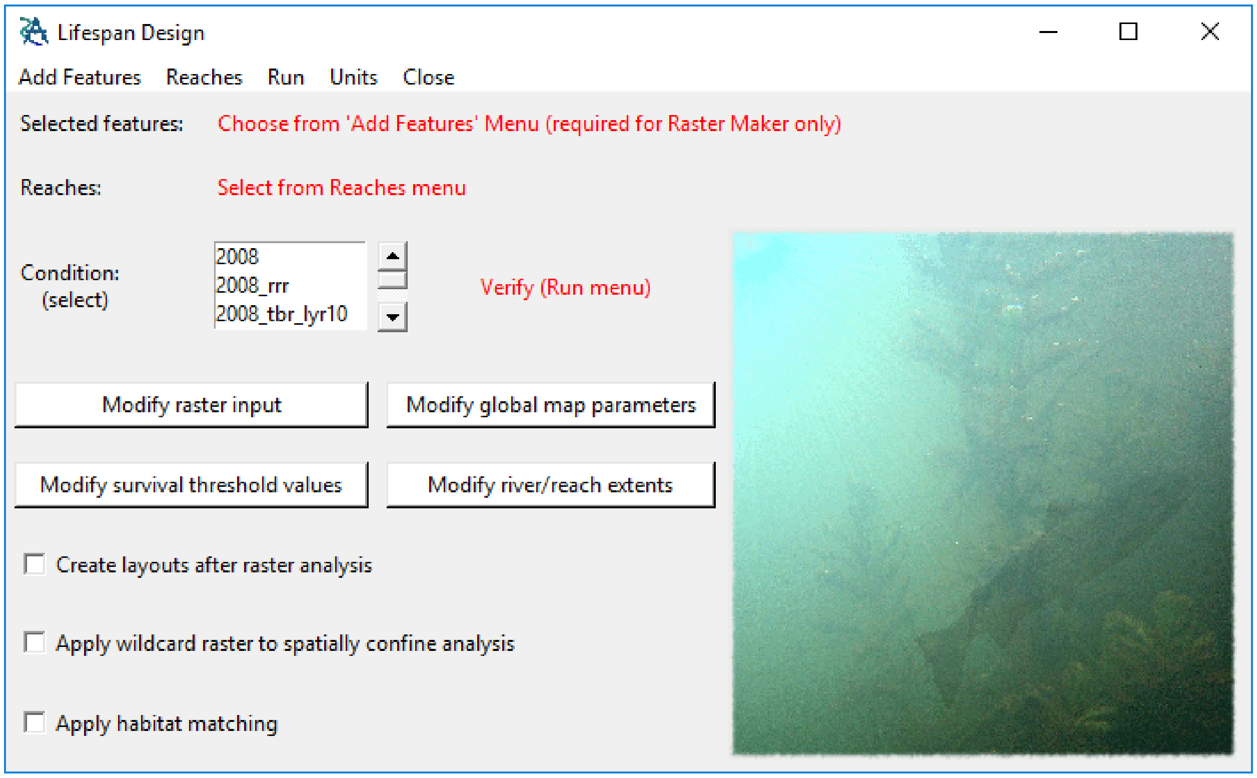
8 Quick GUIde to lifespan and design maps

8.1 Interface and choice of features

The introduction (Sec. 3) explains required modifications of the module batch launcher (`LAUNCH_Windows_x64.bat`) environment.
Figure 4 shows the modules GUI at start-up, which may take a couple of seconds to launch because the module creates some of its menu entries from a spreadsheet. To begin, click on the drop-down menu "Add Features" and select relevant features. Multiple selection is possible and will extend the "Selected features" list. The *LifespanDesign* module enables the selection of the feature groups "terraforming" (framework), "vegetation plantings", "other (soil) bioengineering" and "maintenance" according to the descriptions in Sec. 10. Soil bioengineering considers slope stability in the *MaxLifespan* and *ModifyTerrain* modules.

8.2 Input: Condition and preparation of rasters

The names of input raster files are defined in a proper file format (*.inp*), which can be changed directly from the GUI button "Modify raster input". The *.inp* files indicates where it requires singles rasters only (`STRING`) or lists of rasters (min. two rasters, `LIST`). The maximum number of rasters is unlimited, but it is recommended to use less than ten rasters to limit the calculation duration. The lifespans related to the hydraulic rasters are defined in the *.inp* file. Modifications of map extents (Sec. 11.2) can be made by clicking on the "Modify map extent" button. Sec. 11.1 provides more information on setting up the input file.


Lifespan Design
Add Features
Reaches
Run
Units
Close
Modify raster input
Modify global map parameters
Modify survival threshold values
Modify river/reach extents
Verify (Run menu)

Selected features: Choose from 'Add Features' Menu (required for Raster Maker only)

Reaches: Select from Reaches menu

Condition: (select) 2008 / 2008_rrr / 2008_tbr_lyr10

☐ Create layouts after raster analysis

☐ Apply wildcard raster to spatially confine analysis

☐ Apply habitat matching

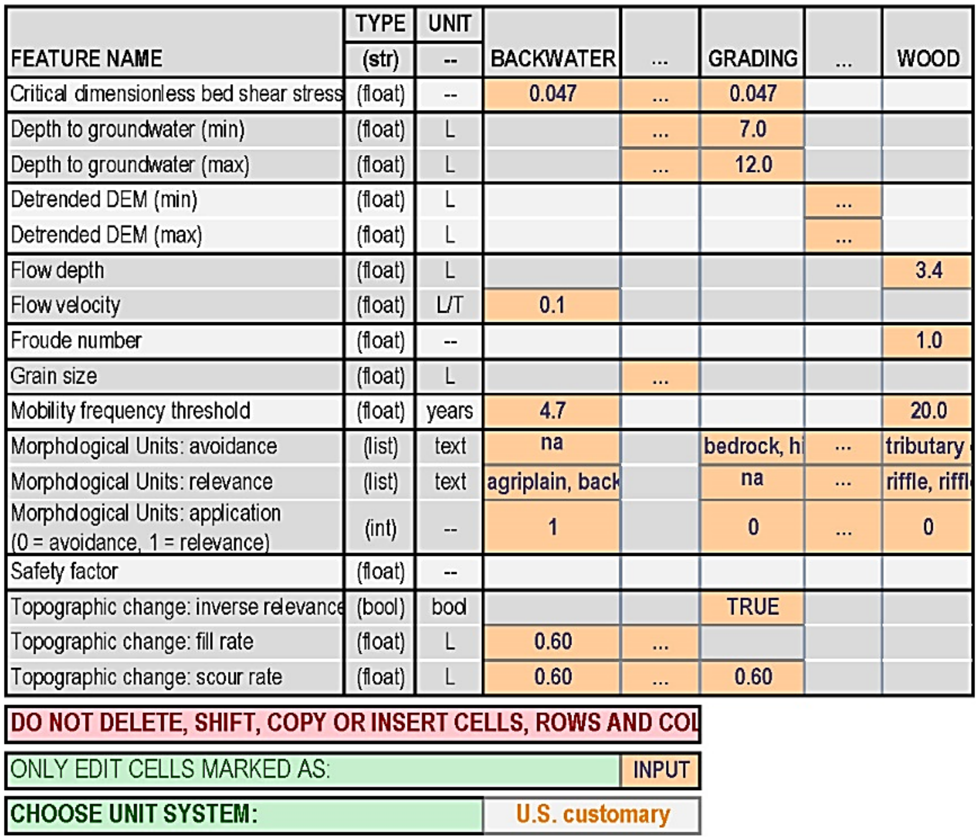
Figure 4: GUI start up window.

8.3 Input: Modify threshold values

The "Modify survival threshold values" button opens a spreadsheet (location: `RiverArchitect/Lifespan Design/.templates/threshold_values.xlsx`), where threshold values and survival identifiers can be modified (cf. Fig. 5) and modifications of the spreadsheet are intuitive. Any modification beyond the "INPUT"-highlighted cells may corrupt the results or cause errors and program crashes. Valid changes limit to the `thresholds` sheet, while the `.templates` sheet must not be modified.
The "Topographic change: inverse relevance" threshold applies when the feature relevance refers to regions where the scour and fill rates below the specific threshold values are relevant. By default, features such as angular boulders (rocks) are relevant where the topographic change rate (scour or fill) exceeds the angular boulders (rocks) threshold value for scour rate. However, features such as grading or side cavities, are relevant where the scour or fill rates do not exceed the threshold rates because these areas are presumably disconnected from the river. Thus, "Topographic change: inverse relevance" is `TRUE` for the grading, side cavity, and side channel features.
The unit system (U.S. customary or SI metric) in the threshold values spreadsheet (Fig. 5) are independent of the GUI settings but they need to be coherent with the input raster files.
More on information on threshold values is provided in Sec. 10, which discusses the identifiers and threshold value of the base case scenario (lower Yuba River in 2008).

8.4 Input: Optional arguments

The checkbox "Include layout creation in raster analysis" provides the optional automated preparation of `.mxd` files for mapping the results (see explanations in Sec. 8.7.2).
The checkbox "Apply wildcard raster to spatially confine analysis" can be checked to use the `wild` raster for spatial limitation of the results. This application makes sense, e.g., if the wildcard raster contains particular land parcels,



FEATURE NAME	TYPE (str)	UNIT --	BACKWATER	...	GRADING	...	WOOD
Critical dimensionless bed shear stress	(float)	--	0.047	...	0.047		
Depth to groundwater (min)	(float)	L		...	7.0		
Depth to groundwater (max)	(float)	L		...	12.0		
Detrended DEM (min)	(float)	L				...	
Detrended DEM (max)	(float)	L				...	
Flow depth	(float)	L					3.4
Flow velocity	(float)	L/T	0.1				
Froude number	(float)	--					1.0
Grain size	(float)	L		...			
Mobility frequency threshold	(float)	years	4.7				20.0
Morphological Units: avoidance	(list)	text	na		bedrock, hi	...	tributary
Morphological Units: relevance	(list)	text	agriplain, back		na	...	riffle, riffl
Morphological Units: application (0 = avoidance, 1 = relevance)	(int)	--	1		0	...	0
Safety factor	(float)	--					
Topographic change: inverse relevance	(bool)	bool			TRUE		
Topographic change: fill rate	(float)	L	0.60	...			
Topographic change: scour rate	(float)	L	0.60	...	0.60		

DO NOT DELETE, SHIFT, COPY OR INSERT CELLS, ROWS AND COL

ONLY EDIT CELLS MARKED AS: INPUT

CHOOSE UNIT SYSTEM: U.S. customary

Figure 5: Spreadsheet with threshold values and survival identifiers.

where the owner wants to foster habitat enhancement.
The checkbox "Apply habitat matching" provides the option of habitat matching to regions where the habitat suitability index is low (<0.4, see explanations in part V).
Switching between unit systems (U.S. customary or SI – metric) is possible via the drop-down menu "Units"; please note that the unit system needs to be consistent with all input raster files.

8.5 Run

Once all inputs are defined, click on "Run" and "Verify settings" to ensure the consistency of the chosen settings (the window will freeze for some seconds). After successful verification, the selected options change to green font.

Three "Run" drop-down menu provides the following routines:

- Raster Maker prepares lifespan and design rasters in the directory `Output/Rasters/`*`condition`*`/`
- Layout Maker prepares `.mxd` layouts in the directory `Output/Mapping/`*`condition`*`/Layouts`;
  by default the layout maker applies on the rasters stored in `Output/Rasters/`*`condition`*`/` but it also accepts other raster input directories as an optional argument when the module is used without GUI (see Alternative Run options in Sec. 8.6).
- Map Maker prepares map assemblies (`pdfs`) in the directory `Output/Mapping/`*`condition`*`/`;
  by default the maps are created based on the layouts stored in `Output/Mapping/`*`condition`*`/` but the

method also accepts other layout input directories as an optional argument when the module is used without GUI (see Alternative Run options in Sec. 8.6)

Either "Run" option causes a run confirmation window popping up and clicking "OK" calls the analysis, which will run in the background python window and it freezes the GUI windows. Running the Raster Maker takes 1 to 10 hours, depending on the feature set and habitat matching. The Layout Maker requires that rasters exist in the `Output/Rasters/`*`condition`*`/` directory. After the Layout creation, manual intervention is required to run Map Maker (see explanations in Sec. 8.7.2).

After the analysis, the GUI unfreezes and a red button will appear, which invites reading the logfiles with information, error and warning messages that occurred during the analysis.

Moreover, the module requires the directory `01_Conditions/`*`condition`*`/` to be located in the same folder as the `.py`-files. Section 5 explains the preparation of this directory.
The directory `Output/Mapping/.ReferenceLayouts` is essential for `class Mapper()`. Section 11.2 illustrates possibilities and procedures for adapting map layouts.

8.6 Alternative run options

The three run options of the GUI call the following methods:

1. Raster Maker calls `feature_analysis.raster_maker` for the preparation of rasters in the directory `Output/Rasters/`*`condition`*`/`
2. Layout Maker calls `feature_analysis.layout_maker` for the preparation of `.mxd` layouts in the directory `Output/Mapping/`*`condition`*`/Layouts`; this method applies on rasters stored in `Output/Rasters/`*`condition`*`/` by default but it also accepts other raster input directories as an optional argument
3. Map maker calls `feature_analysis.map_maker` for the preparation of maps assembled in `pdfs` in the directory `Output/Mapping/`*`condition`*`/`; by default the layouts stored in `Output/Mapping/`*`condition`*`/` underlie the `pdf` creation but the method also accepts other layout input directories as an optional argument
Please not that directories always need to be ***absolute****; relative paths will result in errors.*

The alternative run options are relevant, e.g., for the batch processing of several conditions. Moreover, the alternatives enable running the Layout Maker or Map Maker in another folder than `Output/Rasters/`*`condition`*`/`. The first alternative is importing the module *LifespanDesign* in the ArcGIS Python **x64** interpreter as follows:

1. Prepare input in `.../01_Conditions/`*`condition`*`/` folder
2. Go to ArcGIS Python folder
Example: `C:/Python27/ArcGISx64XX.X`
3. Launch `python.exe`
4. Enter `import os`
5. Navigate to Script direction using the command `os.chdir("ScriptDirectory")`
Example: `os.chdir("D:/Python/RiverArchitect/LifespanDesign/")`
6. Import the module: `import feature_analysis as fa`

Once the module is imported three methods are available and their use is intended in the following order:

1. fa . raster_maker (”condition”, ∗args) for raster (ESRI GRID) creation
2. fa . layout_maker(”condition”, ∗args) for layout (`.mxd`) creation
3. fa . map_maker(”condition”, ∗args) for map (`pdf`) creation

The following steps illustrate the application of fa . raster_maker (”condition”, ∗args) for creating rasters.

- Basic execution: fa . raster_maker (”condition”), for example: fa . raster_maker (”2008”)
- The code is now running (this takes two to four hours) and it will prompt its activities.
- Alternatively, the analysis can be limited to some features only (count 2 to 30 minutes per feature). raster_maker accepts optional arguments. which are `feature_list`, which enables the analysis of any feature listed in Sec. 4, and mapping, which calls layout (`mxd`) creation. Some examples for particular applications:
  → Example 1: fa . raster_maker (”2008”, [”Plantings”]) analyses plantings only.
  → Example 2: fa . raster_maker (”2008”, [”Plantings”, ”Boulders/rocks”], True) analyses plantings and angular boulders (rocks) only with an optional argument True that activates the creation of layouts for plantings and angular boulders (rocks).
  → Example 3: fa . raster_maker (”2008”) analyses all available features (see Sec. 4).
- The complete list of optional arguments of fa . raster_maker (...) is as follows:
  Hint: Respecting the order of optional arguments is crucial to ensure proper application of the desired analysis options.

`args[0]` = feature_list as above described.

`args[1]` = mapping, which can be True or False (default).

`args[2]` = habitat_analysis , which can be True or False (default) for activating or deactivating habitat delineation (limitation) of restoration features to zones with low habitat suitability (`cHSI` = 0.0 to 0.4).

`args[3]` = habitat_radius is a Float number determining in what distance to low habitat suitability zones restoration features should be applied (default = 400.0 ft or m).

`args[4]` = unit_system is either ”us” (default) or ”si”.

`args[5]` = wildcard is either True or False (default).

The code creates a temp folder called `.cache` where it stores temp variables, databases, and rasters. Avoid accessing `.cache` while the code is running and ensure its (manual) deletion in the case that the code crashed.

fa . layout_maker(”condition”, ∗args) creates layout files (`.mxd`) and it can be used as follows.

- With prior creation of rasters (see above Example 2):
  fa . raster_maker (”condition”, [”Featurename”], True or fa . raster_maker (”condition”, [], True; please note that True needs to be given at third place and the default is False (layout creation deactivated).
- Creating layouts only (requires that rasters exist):
  Option 1: fa . layout_maker(”condition” uses the raster input folder `.../Output/Rasters/`*`condition`*`/`
  or
  Option 2: fa . layout_maker(”condition”, ”D:/Any/absolute/path/” uses an alternative raster input folder (must be an absolute path); ensure finishing the path with ”/” or ”\\”

fa . map_maker(”condition”, ∗args) for creating `pdf` map assemblies requires layout files `.mxd` prepared by either fa . raster_maker (”condition”, [”Featurename”], True or fa . layout_maker(”condition”. After either method has created layout files `.mxd`, manual intervention is required because of an `arcpy` deficiency: called outside of *ArcMap Desktop*, `arcpy` works as a background process that cannot actively change layer symbology. The module has an own

`ServerStyle` file stored in `.../Output/Mapping/.ReferenceLayouts`, which defines the legend style. Currently *ArcGIS* can apply the styles of any `.ServerStyle` to the legend only but not to layers, even though the styles are contained in the file. For more information, follow the discussion on GeoNet.

In the meanwhile, manual intervention is required as explained in the Output-Section 8.7. Also `fa.map_maker("condition", *args)` accepts an optional argument defining an alternative layout input path:

- Option 1: `fa.map_maker("condition"` uses the layout input folder `.../Output/Mapping/`*`condition`*`/`
- Option 2: `fa.map_maker("condition", "D:/Any/absolute/path/"` uses an alternative layout input folder (must be an absolute path); ensure finishing the path with `"/"` or `"\\"`

The second alternative is running the module as standalone script (`feature_analysis.py`) from the system command line:

1. Launch terminal
   Windows: Launch `cmd`
   Mac OS: Launch `Terminal.app`
   Linux: Open terminal
2. On Windows: navigate to the place where *ArcGIS* `python.exe` is stored:
   For example: `C:\Python27\ArcGISx64XX.X\` and pay attention using
3. Run `feature_analysis` as script:
   - Windows: `python.exe DriveLetter :\...\ LifespanDesign\ feature_analysis "condition" ["Feature" "name"]`
   - Linux `python .../ feature_analysis "condition" ["Featurename"]`
     Hint: Ensure that python calls the correct version used by arcpy.
4. The code asks for a condition, which needs to be typed case-sensitive and without any apostrophes:
   For example: `Enter the condition (shape: >> XXXX, e.g., >> 2008 )>> 2008`
5. Next, the code asks for a `feature_list` , which is and optional argument (simply hitting enter will work, too); the feature list must be typed as list (in brackets):
   For example: `Enter the condition (no mandatory; do not forget brackets – example: >> ["Featurename1", "Featurename2] >> ["Sidecavity", "Bermsetback"]`
6. The code is now running - this takes time - and it will prompt when it finished.

Calling the module as `.py` script may cause in errors because of differences between path interpretation methods and it is limited to the creation of rasters only. Therefore, the fastest and most consistent way for using the `feature_analysis` module is to import it as above described.

8.7 Output

8.7.1 Rasters

The output rasters are either of the types lifespan (`lf_`*`shortname`*) or design (`ds_`*`shortname`*) and they are created in `.../Output/Rasters/`*`condition`*`/`. The usage of *`shortname`*s (see list in Sec. 4) is necessary because `arcpy` does cannot handle rasters with names longer than 13 characters. The analysis automatically shortens too long raster names on the basis of shortnames and it creates the condition-output directory if it does not yet exist. Existing files in the `Output/Rasters/`*`condition`*`/` folder are overwritten (the code enforces overwriting and tries to delete any existing content, i.e., ensure that the output folder does not contain any important files).

8.7.2 Layouts and Maps

The module provides a half-automated routine for mapping the rasters in `pdfs`. Full automation is not possible because when `arcpy` is called outside of an *ArcMap-Desktop* application, it runs as a background process, which cannot transfer the symbology from any layer or feature to another layer or feature (see above comments in Sec. 8.6). The following workflow can be used to obtain `pdf` maps of all rasters from `Output/Rasters/`*`condition`*`/`.

1. Prepare layouts
   (a) GUI: Either check button before launching "Run: Raster Maker" or directly by clicking on "Run: Layout Maker" from the "Run" menu.
   (b) Alternative `python` console: Either use `fa.raster_maker(condition, 1)` of `fa.layout_maker(condition)`:
      - Calling the `raster_maker` with the optional argument "1", e.g., `fa.raster_maker("2008", 1)` calls the function `fa.layout_maker` based on prepared layouts for lifespan and design maps.
      - Directly call `fa.layout_maker("condition")` for creating layouts only.
      - Directly call `fa.layout_maker("condition"), r"D:/Alternative/Raster/Directory/"` for creating layouts from a directory that differs from `Output/Rasters/`*`condition`*`/`.
2. Python now prepares layout files (`.mxd`) in the folder `Output/Mapping/`*`condition`*`/Layouts/` corresponding to the raster names in `Output/Rasters/`*`condition`*`/`.
3. Open each layout file (`lf_ ....mxd` and `ds_....mxd`) in *ArcMap-Desktop* and use the following procedure to apply the symbology (see illustration in Fig. 6):
   (a) In the Table of Contents, double-click on the gray-scaled `temp` layer for accessing the *Properties* window.
   (b) Go to the *Symbology* tab and click on "Classified" (computing histograms is required, if queried). Click on "Import..." button (folder symbol in the top-right corner) and select `lf_sym_ras` (for lifespan maps) or `ds_sym_ras` (for design maps).
   Hint: Some layouts contain on/off ("NoData+/1) values only. In these cases, "Unique Values" apply instead of "Classified".
   (c) Click OK and the gray layer should adapt its colors.
   (d) Save and exit the `.mxd` file.
4. Run: Map Maker
   (a) GUI: Click on "Run" and "Run: Map Maker".
   (b) Alternative `python` console: type and run `fa.make_maps(condition)`, which produces a `pdf` catalog of each layout.
5. Find the maps in the `Output/Rasters/`*`condition`*`/Layouts/` directory.

The module uses layouts that are placed in the directory `.../Output/Mapping/.ReferenceLayouts/`, which should not be changed unless the `pdf` style requires adaptations. The map extents, scales and focus can be changes in the `mapping.inp` file (see Sec. 11.2).

8.7.3 Interpretation

The success of features corresponds to their ecological sustainability and physical stability, which may positively correlate, i.e., high stability corresponds to high ecological sustainability. However, features such as gravel augmentation or grading have an inverse relationship between ecological sustainability and physical stability. For example, frequently mobile gravel injections create valuable habitat but are, by definition, unstable. In such cases, the lifespan maps need to be considered in the opposite way: Optimum areas for application correspond to regions with low lifespans.

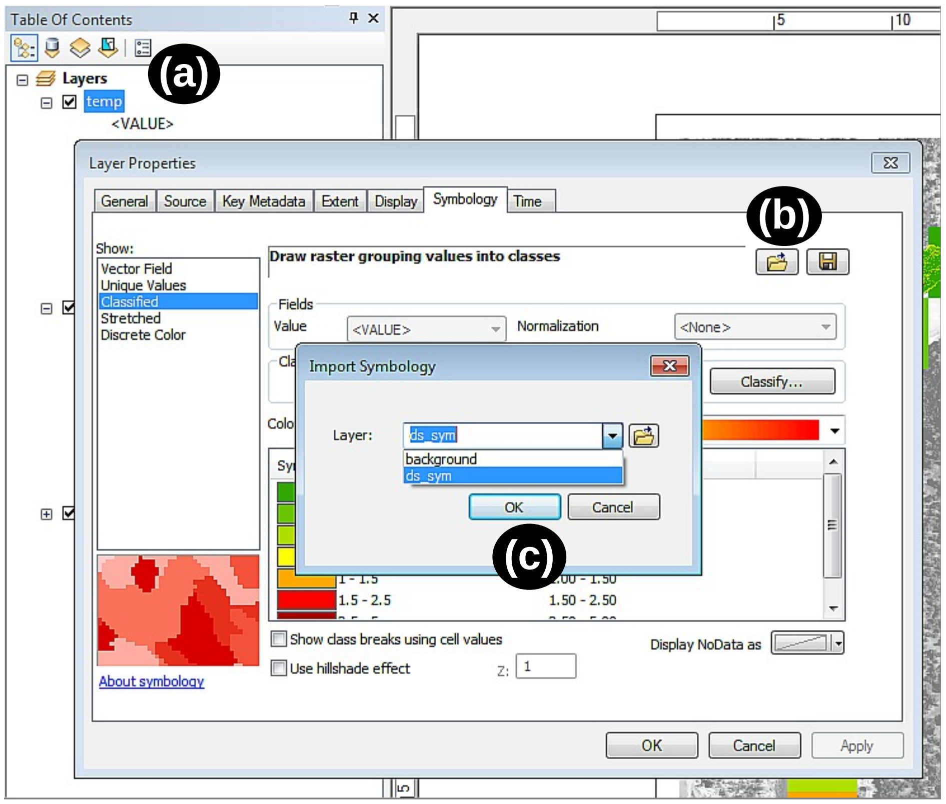


Figure 6: Steps a) to c) for adapting the symbology in *ArcGIS Desktop* according to the descriptions in the text.

8.7.4 Quit module and logfiles

The GUI can be closed via the `Close` dropdown menu if no background processes are going on (see terminal messages). The GUI flashes and rings a system bell when it completed a run task. If layout creation and/or mapping were successfully applied, the target folder automatically opens. After execution of either run task, the GUI disables functionalities, which would overwrite the results and it changes button functionality to open logfiles and quit the program. Logfiles are stored in the `RiverArchitect/LifespanDesign/` folder with names `lifespan_design.log` (Raster Maker) and `mapper.log` (Layout/Map Maker). Logfiles from the previous runs are overwritten.

9 Parameter hypothesis

Combinations of recurring parameters determine the lifespans of features. The code analyses the following parameters, where the application order (hierarchy) differs from the alphabetic order for reasons of map integrity (see coding conventions in Sec. 12.2 for details).

- `chsi` composite Habitat Suitability Index (dimensionless value between 0 and 1)
- `d2w` is the surface depth to the groundwater table (length units)
- `det` is the detrended DEM (length units)
- `Dcr` are mobile or stable grain sizes that are entrained by rare discharges that occur according to a defined return period (see angular boulders (rocks) in Sec. 10.8
- `fill` corresponds to annual sediment deposition rates (length units; see also Wyrick and Pasternack, 2016)

- `Fr` is the Froude number corresponding to u/(h g), where g denotes gravity acceleration (dimensionless hydraulic)
- `h` is the flow depth (length units)
- `mu` are the morphological units (strings; see also Wyrick and Pasternack, 2014)
- `Se` is the energy slope (cf. angular boulders (rocks) in Sec. 10.8 and side channels in Sec. 10.11)
- `scour` corresponds to annual erosion rates (length units, see also Wyrick and Pasternack, 2016)
- `sidech` delineation of priority regions for side channels (van Denderen et al., 2017)
- `taux` (or τ_*) is the dimensionless bed shear stress and its critical value $\tau_{*,cr}$ (–)
- `tcd` combines `scour` and `fill` analysis
- `u` is the flow velocity (length per time, i.e., fps or m/s)
- `wild` wildcard parameter that can only take on/off values (noData, 0 or 1)

The code uses the `mu` raster to identify feature-adequate morphological units that are stored in feature .mu_good and feature-inadequate units that are stored in feature .mu_bad. Thus, two approaches are possible: an inclusive approach that limits relevant areas using the feature .mu_good list and an exclusive approach that excludes non-relevant areas using the feature .mu_bad list. The following morphological units are considered (Wyrick and Pasternack, 2014):

– agriplain	– bank
– bedrock	– chute
– cutbank	– fast glide
– flood runner	– floodplain
– high floodplain	– hillside
– island high floodplain	– island-floodplain
– lateral bar	– levee
– medial bar	– mining pit
– point bar	– pond
– pool	– riffle
– riffle transition	– run
– slackwater	– slow glide
– spur dike	– swale
– tailings	– terrace
– tributary channel	– tributary delta
– in-channel bar (all within-bankfull bars)	

10 Feature hypothesis

The *River Architect* package applies the following hypothesis to habitat enhancement features referring to the base case of the lower Yuba River in its 2008 condition. For the topographic change, scour and fill rates are considered over a six-year observation period (2008 to 2014, see Weber and Pasternack, 2017). The base case stores the below stated threshold parameters in `RiverArchitect/LifespanDesign/.templates/threshold_values.xlsx`.

10.1 Backwater

The creation of artificial backwaters and swales, or more generally calm water zones, makes sense where the stream power is low and the observed topographic changes are small. The following parameters identify relevant areas for backwater creation:

- `u` with a threshold of 0.1 fps (0.03 m/s).
- `mobile_grains` with frequency threshold of 4.7 years and $\tau_{*,cr}$ threshold of 0.047.
- `tcd` with scour and fill thresholds of $\geq$ 0.1 ft·6 years.
- `mu` using the inclusive method with mu_good = [" agriplain ", "backswamp", "mining pit", "pond", "pool", " slackwater ", "swale"].

10.2 Bioengineering

Areas with a 1.0-year lifespan require bioengineering features that are independent of the depth to the groundwater table because plantings likely will not have sufficient water to survive. Such features typically imply the placement of angular boulders.
In the context of river engineering, soil-bioengineering applies living materials (plants) to stabilize terrain and enhance habitat. Alas, dry conditions in arid and semi-arid (Mediterranean) climate zones limits the possibilities of application. The LifespanDesign module maps potential bioengineering areas, as a function of

- `d2w` the maximum depth to groundwater distance indicates where vegetation plantings-based bioengineering applies.
- `dem` is used to compute the percentwise terrain slope `S0`, where modified terrain with slopes of more than a certain percentage is considered to require reinforcement (set `S0` threshold in `RiverArchitect/LifespanDesign/.templates/threshold_values.xlsx`, see Sec. 8.3)

The lifespan maps of bioengineering features can take three values:

`20.0` years (or maximum value as defined in the input definitions file, cf. Sec. 8.2), if the terrain slope is greater than defined in the thresholds workbook and the depth to groundwater is lower than defined in the thresholds workbook (cf. Sec. 8.3);

`1.0` year, if the terrain slope is greater than defined in the thresholds workbook and the depth to groundwater is greater than defined in the thresholds workbook;

`NoData`, if the terrain slope is lower than defined in the thresholds workbook.

10.3 Berm Setback / Widen

Berms are artificial lateral confinements that are represented by human-made bars and dikes. Also, levees represent a lateral confinement but their flood protection-function should not be deleted, and therefore, levees are not considered for setback action. The code replaces the keyword "Bermsetback" with "Widen" because the removal of lateral confinements represents a river widening.

- `mu` using the inclusive method with mu_good = ["bank", " floodplain ", "high floodplain ", " island –" " floodplain ", " island high floodplain ", " lateral bar", "levee", "spur dike", " terrace "].
- `det` detrended DEM with a lower limit of 17 ft (5.18 m) and an upper limit of 25 ft (7.62 m).

The complete detrended DEM range of the morphological unit lateral bar covers values between -1.24 ft (0.38 m) to 29.5 ft (9.0 m) and the morphological unit spur dike covers `det`-values between 1.9 ft (0.58 m) to 25.9 ft (7.89 m). The other morphological units are in similar ranges. However, the detrended DEM limits the application of berm setback and widening to economically reasonable extents. The `det` limits in the code refer to empiric values corresponding to berm setback features according to (USACE and YCWA, 2016).

10.4 Engineered Log Jams / Instream wood

Lifespan maps and design maps are created for engineered log jams (ELJs), where the following parameters apply:

Lifespan maps

- `h` with mobility threshold of 1.7 multiplied with the log diameter of 2 ft (0.6 m Lange and Bezzola, 2006; USACE and YCWA, 2016).
- `Fr` with a threshold of 1 (critical flow conditions).
- `mu` excluding tributary sections (see below descriptions).

Design maps

- `h` is used to computed the minimum required log diameter to avoid motion for a 20-years flood.

Regarding morphological units, riffle-pool and plane bed morphologies are favorable for ELJ placement, where side channel and tributary systems are not convenient for wood placement. ELJs inclusive list is defined as mu_good = [" riffle ", " riffle transition ", "pool", " floodplain ", " island floodplain ", " lateral bar", "medial bar", " run"] and the exclusive list is defined as mu_bad = [" tributary channel", " tributary delta "]. For ELJs, the exclusive approach based on mu_bad applies (see details in the parameter descriptions in Sec. 9).

The design maps for the minimum required log diameter D_w result from (Ruiz-Villanueva et al., 2016)'s interpolation curve as a function of the flow depth. The module applies on the single-thread formula because it returns larger values for the log diameter than the multi-thread formula when the probability of motion is set to zero: `Dw` = 0.32 / 0.18 $*$ `h`. The output map limits to regions where `Dw` is smaller than 300 in (7.6 m).

10.5 Fine sediment

Artificially introduced fine sediment facilitates root growth of new plantings but the flow may easily entrain artificially placed fine sediment. Moreover, spontaneous percolation of fine sediment into the voids of the coarser existing sediment may occur. Therefore, plantings-specific parameters apply to the introduction of fine sediment, as well as filter criteria. The analysis considers fine sediment with a maximum grain diameter of 0.08 in (2 mm sand). The feature analysis module uses the following raster criteria:

Lifespan maps

- `taux` with a threshold of $\tau_{*,cr}$ = 0.030.
- `Dcf` is the maximum admissible size of fine sediment including the ($D_{max,fine}$ $<$ 0.08 in [2 mm]).
- `tcd` with the scour threshold of White Alder (largest for plantings) of 1 ft (0.308 m) multiplied with 6 years and the fill threshold of Cottonwood (highest for plantings) of 0.8·0.2·7 ft [2.13 m]·6 years
- `d2w` with a lower limit of 1 ft and an upper limit of 10 ft corresponding to plantings limits.

Design maps

- `filter` criteria resulting in a design map according to (USACE, 2000):
  $D_{15,fine} > D_{15,coarse}$ / 20;
  $D_{85,fine} > D_{15,coarse}$ / 5;
  $D_{max,fine}$ must be finer than sand, i.e., $<$ 0.08 in (2 mm), to satisfy its "fine" character.

The topographic change and depth to water table thresholds correspond to the largest values that any plantings type (cf. Sec. 10.7) supports because only these areas are of interest for the incorporation of fine sediment in soils.

10.6 Grading

Grading aims at the reconnection of high floodplains and isolated islands by means of floodplain terracing and bar lowering. Its application is from an interest in areas where potential plantings cannot reach the groundwater table or where even high discharges cannot rework the channel. Low dimensionless shear stress, infrequent grain mobilization or low scour rates indicate relevant sites. The following parameter rasters and hypothesis apply to lifespan maps for grading measures (no design maps).

- `mobile_grains` with frequency threshold of 12.7 years and $\tau_{*,cr}$ threshold of 0.047.
- `taux` with mobility threshold of $\tau_{*,cr}$ equal to 0.047.
- `scour` with a threshold value of 0.1 ft multiplied with 6 years and the inverse argument, i.e., areas of interest correspond to regions where the scour threshold is not exceeded.
- `d2w` with a lower limit of 7 ft (2.13 m) and an upper limit of 10 ft (3.05 m).

Further aspects may be considered in addition to the implemented parameters:

- Depth to groundwater
  The YRERFS report (USACE and YCWA, 2016) proposes grading where the depth to groundwater is between 7 (2.13 m) and 10 ft (3.05 m). A visual control of the maps indicates that the upper limit should be increased to 12 ft which corresponds to the tip of several islands.
- Morphological Units
  Currently not applied because every analysis would require the expensive manual assessment of morphological units. This is not necessary for assessing potential grading zones that are primarily determined by the depth to groundwater.

10.7 Plantings

The survival analysis of plantings assumes a general cutting length of min. 7 ft (2.13 m), where approximately 80 % of the cuttings are planted in the ground and 20 % protrude above the ground. The lifespan maps for plantings vary among four indigenous species, which have previously been determined to be relevant for habitat enhancement at lower Yuba River. No design maps are created because the lifespan maps already contain all relevant information.

- Box Elder
  Parameters (extracted from Friedman and Auble, 1999; Kui and Stella, 2016): `h` (exclude all submerged regions for more than 1'000 cfs), `taux` (threshold of 0.047) and `d2w` (lower threshold is 3 ft and upper threshold is 6 ft);
  The maximum submergence duration supported by Box Elder cuttings is 85 days per year. The discharge duration curve from Marysville gaging station (1967–2015) indicate a cumulative annual submergence of 85 days per year for a discharge of 569 cfs, where the Hallwood-study indicates successive 21-submergence when the discharge exceeds 2'000 cfs. The code uses the 1'000-cfs-discharge situation as tradeoff for the 85-days submergence criterion.

- Cottonwood
  Parameters (extracted from Stromberg et al., 1993; Polzin and Rood, 2006; Wilcox and Shafroth, 2013; Bywater-Reyes et al., 2015; Kui and Stella, 2016): `hyd` ($h \geq$ 1.5·0.2·7 ft [2.13 m] and $u \geq$ 3.0 fps), `tcd` (scour$\geq$ 0.1·0.8·7 ft [2.13 m] ·6 years and fill $\geq$ 0.8·0.2·7 ft·6 years) and `d2w` (lower threshold is 5 ft and upper threshold is 10 ft);
  Uses thresholds for combined hydraulics analysis (velocity and depth), scour and fill (tcd) and depth to water table. Given the minimum cutting length of 7 ft (2.13 m), cottonwood plantings have a threshold_scour of 0.1·0.8·7 ft·6 years (2008 to 2014) and a threshold_fill of 0.8·0.2·7 ft·6 years.

- White Alder
  Parameters: `taux` (threshold of 0.047), `scour` ($\geq$ 1 ft·6 years, cf. Jablkowski et al., 2017) and `d2w` (lower threshold is 1 ft and upper threshold is 5 ft);
  In addition to the scour maps, potential scour resulting from a grain mobility frequency analysis provide information on the lifespans of White Alder plantings. threshold_scour is 1 ft·6 years.

- Willow
  Parameters (extracted from Stromberg et al., 1993; Pasquale et al., 2011, 2012, 2014): `h` ($h \geq$ 0.7 ft + 0.2·7 ft), `taux` (threshold of 0.1), `scour` ($\geq$ 0.1·0.8·7 ft·6 years) and `d2w` (lower threshold is 3 ft and upper threshold is 5 ft);
  Willow cuttings have a maximum submergence survival that defines the threshold_h as 0.7 ft + 0.2·7 ft and maximum scour survival of 0.1·0.8·7 ft·6 years.

10.8 Angular boulders (rocks)

The punctual placing of boulders and comprehensive rock cover is referred to as "angular boulders (rocks)" for stabilizing banks or erosion-prone surfaces (e.g., Maynord and Neill, 2008). The mobility of the present terrain indicates the necessity of boulder placement on the basis of lifespan maps. Moreover, the required minimum diameter for boulders results from the spatial evaluation of D_{cr} on angular boulders (rocks) design maps. The following parameters apply to the generation of angular boulders (rocks) maps:

Lifespan maps

- `taux` with mobility threshold of $\tau_{*,cr}$ equal to 0.047.
- `scour` with a threshold value of 1 ft multiplied with 6 years.

Design maps

- `stable_grains` for design maps (see below formulae), with a frequency threshold of 20.0 years and $\tau_{*,cr}$ threshold of 0.047.

The minimum required grain sizes are determined in a two-way analysis, i.e., two minimum angular boulders (rocks) size maps are produced based on the highest discharge where hydraulic data is available (20.0 years):

1. `ds_rocks_Dcr` is a derivative of the Gauckler-Manning-Strickler formula using Manning's n:
   $D_{cr} = SF \cdot u^2 \cdot n^2 / \left[(s-1) \cdot h^{1/3} \cdot \tau_{*,cr}\right]$
2. `ds_rocks_Dcr` is a derivative of the Chézy formula using the energy slope:
   $D_{cr} = SF \cdot h \cdot S_e / [(s-1) \cdot \tau_{*,cr}]$

where:

D_{cr} is the minimum required angular boulders (rocks) size (in INCHES);

h is the flow depth (pixel-wise, in ft);

n is Manning's n (in s/ft$^{1/3}$ – an internal conversion factor of k = 1.49 applies);

s is the dimensionless relative grain density (ratio of sediment and water density, equal to 2.68);

S_e is the energy slope (derived from arcpy's "Slope" function, dimensionless);

SF is a safety factor equal to 1.3 (dimensionless);

u is the flow velocity (pixel-wise, in fps);

$\tau_{*,cr}$ is the threshold value of dimensionless bed shear stress for incipient grain motion, equal to 0.047.

The energy slope maps result from computing the theoretic energy height maps as ras_energy = dem + h. raster_110k + u. raster_110k ^{2}/(2 g), where g denotes gravitational acceleration.

10.9 Sediment replenishment / gravel augmentation

Large dams tend to retain the nearby-totality of the catchment sediment supply. The missing sediment causes channel incision and the morphological depletion of lower Yuba River in the long term. Regular artificial gravel injections can antagonize this artificial sediment scarcity (e.g., Pasternack et al., 2010). Other authors ((Gaeuman, 2008) and (Ock et al., 2013)) distinguish replenishment techniques inside and outside of the main channel. According to this, two types of gravel augmentation are considered:

1. Gravel stockpiles on the floodplain and river banks; and
2. Gravel injections or stockpiles directly in the main channel.

Gravel deposits on floodplains should be erodible by frequent floods, i.e., stockpiles make sense where only larger floods entrain grains. In contrast, gravel injections in the main channel aim at the immediate creation of spawning habitat that should not wash out with the next minor flood event. However, gravel injections with low longevity in the main channel can also serve for an urgent equilibrium of river sediment budget. Therefore, the lifespan maps for gravel replenishment require two different interpretations inside and outside of the main channel: High lifespans are desirable in the main channel for immediate habitat creation and low lifespans are desirable for equilibrating the sediment budget.

- In-channel gravel injections

  Lifespan maps

  - `mobile_grains` analysis with a minimum frequency of 1.0 years and $\tau_{*,cr}$ threshold of 0.047.
  - `mu` uses the inclusive method with mu_good = ["chute", "fast glide", "flood runner", "bedrock", " lateral bar", "medial bar", "pool", " riffle ", " riffle transition ", "run", "slackwater", "slow glide", "swale", " tailings "]

  Design maps

  - `stable_grains` for design maps (see angular boulders (rocks) formulae), with threshold_freq of 1.0 years and $\tau_{*,cr}$–threshold of 0.047.

- Floodplain / overbank gravel stockpiles

  Lifespan maps

  - `mobile_grains` analysis with a minimum frequency of 1.0 years and $\tau_{*,cr}$ threshold of 0.047.
  - `scour` with a threshold value of 1 ft per year.

  - mu uses the inclusive method with mu_good = [" agriplain ", "backswamp", "bank", "cutbank", " flood runner", " floodplain ", "high floodplain ", " hillside ", " island high floodplain ", " island –" " floodplain ", " lateral bar", "levee", "medial bar", "mining pit ", "point bar", "pond", "spur" "dike", " tailings ", " terrace "]

    Design maps

  - stable_grains for design maps (see angular boulders (rocks) formulae), with threshold_freq of 1.0 years and $\tau_{*,cr}$–threshold of 0.047.

10.10 Side cavities

From a parametric point of view, side cavities make sense at lateral channel confinements that represent either preservable habitat or require protection to prevent bank collapses. In the latter case, groin cavities are an adequate protection measure that can additionally improve habitat conditions. The code analyses relevant sites based on the morphological units and important scour rates at banks. It excludes fill zones where artificial side cavities are prone to sedimentation making the measure ecologically inefficient.

- tcd with a fill threshold value of 1 ft multiplied with 6 years and a scour threshold of 100 ft leads to the exclusion of fill-prone sites.
- mu using the inclusive method with mu_good = ["bank", "cutbank", " lateral bar", "spur dike", " tailings "].

10.11 Side channels / anabranches

Any discrete parameters exist for assessing design or lifespan maps for side channels, anabranches, anastomosed or multithread channels. The identification of splays and bank rigidity requires manual and visual proof.

An initial decision support on the basis of design maps was contemplated by comparing the minimum energy slope $S_{e,min}$ with the terrain slopeS_0. In the 1D-theory, the minimum energy slope results from the H-h diagram (Moglen, 2015), based on the assumption that the minimum energy per unit force and pixel H_{min} corresponds to the Froude number Fr = 1 with the critical flow velocity u_c and flow depth h_c. The pixel unitary discharge results from q = u · h, where u and h are pixel values from the `u` and `h` rasters. Thus, the following system of equations can be used:

$$Fr = 1 \quad \leftrightarrow \quad 1 = \frac{u_c}{\sqrt{g\,h_c}} \quad \leftrightarrow \quad u_c = \sqrt{g\,h_c} \tag{1a}$$

$$h_c = \left(\frac{q^2}{g}\right)^{1/3} \tag{1b}$$

$$q = u \cdot h \tag{1c}$$

$$\Rightarrow \quad H_{min} = h_c + \frac{u_c^2}{2\,g} = 1.5 \cdot \left(\frac{q^2}{g}\right)^{1/3} \tag{1d}$$

Thus, the available discharges and related flow velocity `u` / depth `h` rasters could be used for the following calculation (python script sample):

```
S0 = Slope(dem.raster, "PERCENT_RISE", 1.0))/100
for h.ras in h.rasters and u.ras in u.rasters:
            ## compute energetic level
            energy_level[discharge] = dem.raster + 1.5 * Power(Square(h.ras[
                discharge] * u.ras[discharge]) / g, 1/3)})
            ## compute energy slope Se,min
            Se[discharge] = Slope(energy_level[discharge], "PERCENT_RISE", 1.0))
                /100
            ## result = compare Se and S0 (Se / S0)
            Se_S0[discharge] = Se[discharge] / S0)})
```

This sample function uses `arcpy.sa`'s `Slope` function with the arguments `PERCENT_RISE` for obtaining percent values instead of degrees and `zFactor` = 1.0 because the x-y-grid units are the same as in z-direction. `g` denotes gravity acceleration (SI metric: 9.81 m/s^s or U.S. customary: 32.2 ft/s^2).
However, the underlying 2D numerical model uses the critical flow depth as an iteration criterion for stability, which causes that $S_{e,min}$ approximately equals S_0. Thus, the $S_{e,min}$ / S_0 ratio is approximately unity and not meaningful. Otherwise, the $S_{e,min}$ / S_0 indicated pixels with excess energy ($S_{e,min}$ / S_0 > 1) that allegedly caused erosion. In contrast, pixels with energy shortage ($S_{e,min}$ / S_0 < 1) allegedly resulted in sediment deposition. Minor topographic change would be expected where the $S_{e,min}$ / S_0–ratio is close to unity.
Unless this problem is not solved, the package indicates the adequacy of side channel construction on lifespan maps using the following criteria:

- `fill` the fill rate does not exceed the threshold value defined in the thresholds spreadsheet (Sec. 8.3)
- `taux` the critical dimensionless bed shear stress should be smaller than the threshold value defined in the thresholds spreadsheet (Sec. 8.3)
- `sidech` needs to be a manually created *Arc GRID* raster in `01_Conditions/`*`condition`*`/`. The delineation is typically made in a shape file, which is then converted into an *Arc GRID* raster file. The delineation criteria are (van Denderen et al., 2017):

- Side channel intakes are situated at the outer bank, downstream of outer bends or at the inner bank, inside mild inner bends;
- A side channel should be longer than the main channel to avoid cutting off the main channel;
- Structures should be placed in the side channel to control the flow repartitioning and to avoid flow separation in the main channel.

Moreover

11 Input definition files

11.1 Raster data

The file `input_definitions.inp` is stored on the directory `/.templates/` and can be accessed using the link `InputDefinitions.lnk` directly in the code directory. `input_definitions.inp` contains information about lifespan duration and raster names, which link to rasters containing spatial information as described in Sec. 9. The order of definitions and lines must not be changed to ensure the proper functioning of the module. Enter or change information in the corresponding lines, only between the "=" and the "#" signs (the input routines uses these signs as start and end identifiers for relevant information). The following definitions apply line by line:

Lines 1–3	None	Do not change
Line 4	Return periods	Comma-separated list of flood discharge return periods corresponding to the hydraulic rasters; i.e., the first entry after “=” corresponds to the return period of the first velocity and flow depth raster (Lines 11 and 12, respectively)
Lines 5–7	None	Do not change
Line 8	CHSI	One raster name of spatial composite Habitat Suitability Indexes
Line 9	DoD	Comma-separated list of two (first = scour, second = fill) DEM of Differences rasters; if one raster is missing, replace it by double quotation marks, for example scour is missing: ... = ””, dodFill # ...
Line 10	det	One raster name defining the detrended DEM raster
Line 11	u	Comma-separated list defining flow velocity rasters corresponding to discharge return periods (Line 4); replace missing rasters by double quotation marks, for example, when `u` rasters of a return period list of five entries are not available for entries 2 and 4, type ... = u001k, ””, u003k, ””, u005k # ... . However, ensure that at least two `u` rasters are defined. The 00xk identifier relates to the underlying discharge in thousand cfs or m^3/s. Smaller discharges are written without ”`k`”. For example, a velocity raster related to a discharge of 110423 cfs is named u110k, and a velocity raster related to a discharge of 544.4 cfs is named u544.
Line 12	h	Comma-separated list defining flow depth rasters corresponding to discharge return periods (Line 4); replace missing rasters by double quotation marks, for example, when `h` rasters of a return period list of six entries are not available for entries 2, 3 and 5, type ... = h001k, ””, ””, h004k, ””, h006k # ... . Ensure that at least two `h` rasters are defined. The 00xk identifier relates to the underlying discharge in thousand cfs or m^3/s. Smaller discharges are written without ”`k`”. For example, a flow depth raster related to a discharge of 110423 cfs is named h110k, and a flow depth raster related to a discharge of 544.4 cfs is named h544.
Line 13	Grains	One raster name defining the raster containing mean grain diameters (pay attention on raster units: use feet for U.S. customary and m for S.I.)
Line 14	mu	One raster name delineating morphological units according to the definitions in Sec. 9
Line 15	d2w	One raster name defining the depth to groundwater table
Line 16	DEM	One raster name defining the digital elevation model
Line 17	sidech	One raster name delineating appropriate sites for side channels
Line 18	wild	One raster name for the spatial confinement of the feature analysis of 0/nodata (= off) and 1 (= on) values for any purpose (wildcard raster)

The module produces results based on the available information only, where any raster name can be substituted with double quotation marks ””. However, this lack of information reduces the accuracy of final lifespan and design maps. No maps are produced for a feature where the information is insufficient for the analysis. The required information for every feature corresponds to the definitions in Sec. 11.1.

11.2 Mapping

The file `mapping.inp` defines map center points, extents (dx and dy in ft) and scales (scale has no effect currently). `mapping.inp` is stored on the directory `/.templates/` and directly accessible from the code directory via the link `MapLayouts.lnk`.

The extent of the map determines the map scale, where the corresponding dx and dy values define the map width and height in ft, respectively. The layout templates (`.mxd` in the directory `.../Output/Mapping/.Reference Layouts/` define the paper size, which is by default "ANSI E landscape" (width = 44 inches, height = 34 inches).
The map focus is defined page-wise in `mapping.inp` from Line 8 onward. Existing pages can be removed by simply deleting the line. Additional pages can be added by inserting or appending a new line below Line 8, which needs to begin with the keyword "`Page`" and x and y need to be stated in brackets, separated by a comma without any white space ([xxxxxx.xx,yyyyyy.yy]).

Good practice for changing the map layouts starts with opening the `find_center_points.mxd` layout from `. ../Output/Mapping/.ReferenceLayouts/`. Zoom to new focus point using, for example, *ArcGIS* `Go To XY` function from the `Tools` toolbar or freehand to any convenient extent. Use *ArcGIS* `Info` cursor and click in the center of the reticule to obtain the current center point. Write new center point coordinates for the desired page number in `mapping.inp`.
For retrieving the extent, in *ArcGIS* Desktop, go to the `View` menu, click on `Data Frame Properties...` and go to the `Data Frame` tab. In the `Extent` box, click on the scroll-down menu and choose `Fixed Extent`. Subtracting the `Right` value from the `Left` value defines dx (Line 3 in `mapping.inp`) and subtracting the `Top` value from the `Bottom` value defines dy (Line 4 in `mapping.inp`).
The feature_analysis .map_maker() function uses these definitions for zooming to each point defined below Line 8 in `mapping.inp`, cropping the map to the defined extents and exporting each page to a `PDF` map bundle containing as many pages as there are defined in `mapping.inp`.
The program uses the reference coordinate system and projection defined in the layout templates (`.mxd`); i.e., coordinate definitions in `mapping.inp` and `.mxd` files need to refer to the same coordinate system and projection.

12 Code extension and modification

The code can be extended with new parameters, e.g., direct shear stress output from the numerical model, new analyses, e.g., a new shear stress law, and features, e.g., another plant species block ramps.

12.1 Conventions

The rasters creation results from analysis_ and design_ functions that are stored in `cLifespanDesignAnalysis`. analysis_ functions create rasters with lifespan data (0 to 20 years) and design_ functions create rasters with design parameters such as the required stables grain size of angular boulders (rocks).
Class names start with an upper case letter and do not contain any special characters, also excluding dash or underscore signs. Instantiations of classes are all lower case letters. Features, Parameters and Analysis classes are stored in separate files called `cFeatureLifespan.py`, `cParameters.py` and `cLifespanDesignAnalysis.py`, respectively. In addition, Feature classes may inherit subfeature classes from files names `cSubfeature.py`, for example `cPlants.py`.
Function names consist of lower case letters only and the underscore sign "_" separates words.
All class names, variable names and function names are in alphabetic order (a = up, z = down), except the parameter_list s, which determine the run hierarchy (see Sec. 12.2).

12.2 Order of analysis and temp (.cache) raster names

The best position of restoration features and their lifespans depend on multiple parameters in most cases. The output rasters (lifespan maps) are computed in by batch-processing every parameter, i.e., one parameter map is processed after another. This batch processing strictly follows the below-listed hierarchy:

1. Flow depth rasters (dimensional) starting with the lowest discharge to the highest discharge
   Internal raster name: `ras_hXXXk`

2. Flow velocity rasters (dimensional) starting with the lowest discharge to the highest discharge
   Internal raster name: `ras_uXXXk`

3. Hydraulic rasters (dimensionless)
   Internal raster name: `ras_taux` (dimensionless bed shear stress) or `ras_Fr` (Froude number); if needed: the hierarchy among the dimensionless hydraulic numbers is not important

4. Mobile bed, fine sediment and stable grain size raster analysis
   Internal raster name: `ras_Dcr` (mobile or stable grain size)

5. Topographic change rasters
   Internal raster names: `ras_fill` (fill raster only), `ras_scour` (scour only) or `ras_tcd` (combined fill and scour)

6. Detrended DEM raster analysis
   Internal raster name: `ras_det` (relevant, e.g., for berm setback)

7. Morphological Unit rasters
   Internal raster name: `ras_mu`

8. Side channel delineation
   Internal raster name: `ras_sch`

9. Depth to water table
   Internal raster name: `ras_d2w` (relevant, e.g., for plantings and terrain grading)

The dimensional hydraulic maps need to be invoked before any other analysis is performed because the u and h maps are the only ones that entirely cover the area of interest, without "`noData`" pixels.
Every feature has a feature . parameter_list attribute containing a list of parameters that determine the feature lifespan and applicability space. The parameters are ordered in the feature . parameter_list according to the hierarchy. Once the last element of feature . parameter_list is processed and stored in the cache folder, the code exits the loop and copies the last `ras_`*`parameter`* to the `Output/Rasters/`*`condition`*`/` folder. This copy is renamed `lf_`*`shortname`*, where the usage of shortnames (see list in Sec. 4) is necessary because `arcpy` cannot save or copy raster with names exceeding 13 characters.

12.3 Add parameters

The currently implemented parameters are listed in Sec. 9. New parameters require new input rasters in addition to the list in Sec. 5. The rasters need to be saved in the folder `01_Conditions/`*`condition`*`/` using the *Esri Grid* format. Other raster formats such as `.tif` may cause inconsistencies that result in error messages when the code attempts to save the final rasters. The template for creating a new parameter class is shown in the box. Use the following workflow to implement a new parameter in the code:

1. Create *Esri Grid* parameter rasters in the folder `01_Conditions/`*`condition`*`/`.
2. Add a new parameter class in the file `cParameters.py` (cf. box explanations).
3. Add a new function called `analyse_`*`parameter`* to the `ArcPyAnalysis` class in the file `cLifespanDesignAnalysis.py` (see Sec. 12.4) or change existing analysis for using the new parameter.

Add new parameter class to `cParameters.py`

- Replace `EXPRESSIONS` as indicated
- Write function in alphabetic order in `cParameters.py`; e.g., the class `Mypar` should be placed below the existing class `GrainSizes` and `WaterTable`
- Coding convention: the class name begins with a Capital letter, where an instance of the class would begin with a small letter

```
class PARAMETERNAME():
  def __init__(self, condition):
    self.condition=condition  # [str] planning situation, .e.g., "2008"
    self.raster_path='YOUR PATH/01_Conditions/'
    self.raster_names=['RASTER1', 'RASTER2', ..., 'RASTERi', ..., 'RASTERn']
    self.RAS1=arcpy.Raster(self.raster_path+self.condition+'/'+self.
        raster_names[0])
    self.RAS2=arcpy.Raster(self.raster_path+self.condition+'/'+self.
        raster_names[1])
    ...
    self.RASi=arcpy.Raster(self.raster_path+self.condition+'/'+self.
        raster_names[i-1])
    ...
    self.RASn=arcpy.Raster(self.raster_path+self.condition+'/'+self.
        raster_names[n-1])
```

12.4 Add analysis

The analysis routines are differentiated between analyse and design-functions, which are contained in the file `cLifespanDesignAnalysis.py`.
analyse-functions return rasters containing estimated survival times (in years) or on/off values (1/0). An analyse-function will always try to find existing rasters produced from previous analysis functions according to the analysis hierarchy (Sec. 12.2), unless a dimensional hydraulic analysis (`u`, `h` or their combination) is performed. For this reason, analyse-function uses the verify_raster_info ()-function to look up for previous analyses that are stored in raster_dict_lf . At the end of an analyse-function, the raster_dict_lf is updated using raster_dict_lf .update(" ras_current "). This serial map-analysis produces lifespan rasters, which can be regardlessly converted to design rasters by the save_manager()-function when the feature properties are set to self .ds = True while self . lf = False (Sec. 12.5).
design-functions produce rasters containing specific parameter values, such as the critical grain size in inches. A design-function will update the raster_dict_ds -dictionary which is passed to the save_manager()-function when the feature variable self .ds = True.
The major difference between the raster_dict_lf and raster_dict_ds -dictionaries is that the save_manager() saves the only the last hierarchy-based entry of raster_dict_lf to produced lifespan rasters but all entries of raster_dict_ds to produced design rasters. The combination of multiple parameters into one design raster can be achieved anyway by setting self .ds = True while self . lf = False (Sec. 12.5), which converts lifespan rasters to design rasters.

Use the following workflow to implement a new parameter in the code:

1. Ensure that all required parameters are available (Parameter list: Sec. 9; Add parameters: Sec. 12.3).
2. Create an identifier string of 2 to 3 characters; the following explanations refer to a dummy identifier named NEW (replace with lowercase letters).
3. Add a new analyse_NEW or design_NEW-function in the file `cLifespanDesignAnalysis.py` (cf. code example below).
4. In `cFeatureLifespan.py` ensure that concerned features have the following properties:
   - The feature . parameter_list needs to contain the new analysis' identifier (NEW)
   - All required threshold values are defined (feature .threshold_NEW1 = ...)
5. In `feature_analysis.py`, add a call of the new function:

```
if parameter_name == "NEW":
    feature_analysis.analyse_NEW(feature.threshold_NEW1, ... )
```

The template for a new analyse_NEW-function in the file `cLifespanDesignAnalysis.py` starts with the general statement of unit conversion (controlled by user input) and continues as follows (pay attention on indentation):

```
def analyse_NEW(self, threshold_NEW1, ...):
    ## Lines where changes are required are tagged with #--CHANGE--#
    ## Convert length units of threshold values
    threshold_LENGTH = threshold_LENGTH * self.ft2m      #--CHANGE--#
    try:
     arcpy.CheckOutExtension('Spatial')  # check out license
     arcpy.gp.overwriteOutput = True
     arcpy.env.workspace = self.cache
     self.logger.info("      >>> Analyzing NEW.") #--CHANGE--#
     parameter1 = PARAMETER1(self.condition)      #--CHANGE--#
     parameter2 = PARAMETER2(self.condition)      #--CHANGE--#
     ...                                          #--CHANGE--#
     self.ras_NEW = calculation with parameter1, parameter2, ...
        threshold_NEW1, ...    #--CHANGE--#
     self.ras_LF = self.compare_raster_set(parameter_ras, threshold)
     ## verify existing analyses
     if self.verify_raster_info():
         self.logger.info("            based on raster: " + self.
            raster_info_lf)
         ## make temp_ras without noData pixels
         temp_ras_NEW = Con((IsNull(self.ras_NEW) == 1), (IsNull(self.ras_NEW
            ) * some_factor), self.ras_NEW) #--CHANGE--#
         ## compare temp_ras with raster_dict but use self.ras_... values if
            condition is True
         ras_NEW_update = Con((temp_ras_NEW == 1), self.ras_NEW, self.
            raster_dict_lf[self.raster_info_lf])   #--CHANGE--#
         self.ras_NEW = ras_NEW_update    #--CHANGE--#
     ## update lf dictionary
     self.raster_info_lf = "ras_NEW"      #--CHANGE--#
     self.raster_dict_lf.update({self.raster_info_lf: self.raster_info_lf})
     arcpy.CheckInExtension('Spatial')
    except arcpy.ExecuteError:
       self.logger.info("ExecuteERROR: (arcpy) in NEW analysis.")     #--
          CHANGE--#
       self.logger.info(arcpy.GetMessages(2))
       arcpy.AddError(arcpy.GetMessages(2))
    except Exception as e:
       self.logger.info("ExceptionERROR: (arcpy) in NEW analysis.")  #--
          CHANGE--#
       self.logger.info(e.args[0])
       arcpy.AddError(e.args[0])
    except:
       self.logger.info("ERROR: (arcpy) in NEW analysis.")            #--
          CHANGE--#
       self.logger.info(arcpy.GetMessages())
```

The template for a new design_NEW-function in the file `cLifespanDesignAnalysis.py` is as follows (pay attention on indentation):

```
def design_NEW(self, threshold_NEW1, ...):
    ## Lines where changes are required are tagged with #--CHANGE--#
    try:
     arcpy.CheckOutExtension('Spatial')  # check out license
     arcpy.gp.overwriteOutput = True
     arcpy.env.workspace = self.cache
     self.logger.info("      >>> Designing NEW.") #--CHANGE--#
     parameter1 = PARAMETER1(self.condition)       #--CHANGE--#
     parameter2 = PARAMETER2(self.condition)       #--CHANGE--#
     ...                                           #--CHANGE--#
     self.ras_NEW1 = calculation with parameter1, parameter2, ...
        threshold_NEW1, ...  #--CHANGE--#
     ## if required add more design rasters (all need to be added to self.
        raster_dict_ds)
     self.ras_NEWi = another (optional) calculation with parameter1,
        parameter2, ... threshold_NEW1, ... #--CHANGE--#

     ## update ds dictionary
     self.raster_dict_ds.update({self.raster_info_lf: self.ras_NEW1}) #--
        CHANGE--#
     ## if required uncomment:
     # self.raster_dict_ds.update({self.raster_info_lf: self.ras_NEWi}) #--
        CHANGE--#

     arcpy.CheckInExtension('Spatial')

    except arcpy.ExecuteError:
       self.logger.info("ExecuteERROR: (arcpy) in NEW design.")    #--CHANGE
          --#
       self.logger.info(arcpy.GetMessages(2))
       arcpy.AddError(arcpy.GetMessages(2))
    except Exception as e:
       self.logger.info("ExceptionERROR: (arcpy) in NEW design.")  #--CHANGE
          --#
       self.logger.info(e.args[0])
       arcpy.AddError(e.args[0])
    except:
       self.logger.info("ERROR: (arcpy) in NEW design.")           #--CHANGE
          --#
       self.logger.info(arcpy.GetMessages())
```

12.5 Extend features

The currently implemented features are listed in Sec. 4. New features can be implemented in the `cFeatureLifespan.py` file using the following workflow:

1. Ensure that all required parameters are available (Parameter list: Sec. 9; Add parameters: Sec. 12.3).
2. Ensure that all required analysis and / or design functions are available (cf. Sec. 12.4).
3. Choose a name for the new feature beginning with an uppercase letter followed by lowercase letters only; the name Newfeature is subsequently used for illustrative purpose
4. In `cFeatureLifespan.py` modify the class RestorationFeature :
   - Implement the new feature instantiation when called by adding the following to def __init__ (self , feature_name , *sub_feature):

     ```
     if feature_name == "Newfeature" and not(sub_feature):
         self.feature = Newfeature()
         self.sub = False
         self.name = feature_name
     ```

   - Please note: the shortname should not have more than 6 characters; otherwise the code will cutoff the shortname automatically.
   - Both the initiation __ini__ (self) and the instantiation Newfeature() are necessary to facilitate the external access to Methods and Properties.
   - A feature may have subfeatures (as for example the class Plantings). In this case, replace and not(sub_feature) with and sub_feature and set self .sub = True.

   Add a new class to `cFeatureLifespan.py` according to the example below, considering the required hierarchically ordered self . parameter_list , self . threshold_ ... and lifespan (self . lf = True / False) / design (self .ds = True / False) raster analysis properties.
5. Add new column in `LifespanDesign/.templates/threshold_values.xlsx` and add feature name as well as relevant threshold values.
6. Commit changes in `RiverArchitect/ModifyTerrain/cDefinitions.py` class Features:
   - Append shortname to self . id_list = ["backwt", "widen", "grade", "sideca", "sidech", "elj", "fines", "box", "cot", "whi", "wil", "rocks", "gravin", "gravou", "cust"]
   - Append full feature name to self . name_list = ["Backwater", "Bermsetback (Widen)", "Grading", "Sidecavity", "Sidechannel", "ELJ", "Finesediment", "Plantings : Box Elder", "Plantings : Cot−" "tonwood", "Plantings : White Alder", "Plantings : Willows", "Boulders/rocks", "Gravel replenishment", "Gravel stockpile", "Custom DEM"]
   - Append feature threshold column (`threshold_values.xlsx`) name in self . threshold_cols = ["E", "Q", "G", "O", "P", "R", "F", "J", "K", "L", "M", "N", "H", "I", "S"]

The template for a new Newfeature-class in the file `cFeatureLifespan.py` is as follows (pay attention on indentation), given that no subfeatures apply:

```
class Newfeature():
 ## This is the Newfeature class.
 def __init__(self):
     self.ds = False # identify if design map applies
     self.lf = True  # identify if lifespan map applies
     self.parameter_list = ["PAR1", "PAR2", ..., "PARi", , "PARn"] # Respect
        Hierarchy -- example: PAR1 = "hyd"
     self.shortname = "max6ch"
     thresh = ThresholdDirector(self.shortname) # instantiate reader of
        threshold values
     ## uncomment and adapt follow line if PAR = mu applies
     # self.mu_bad = thresh.get_thresh_value("mu_bad")
     # self.mu_good = thresh.get_thresh_value("mu_good")
     # self.mu_method = thresh.get_thresh_value("mu_method")
     self.threshold_1 = thresh.get_thresh_value("ID_1")
     self.threshold_2 = thresh.get_thresh_value("ID_2")
     ...
     self.threshold_i = thresh.get_thresh_value("ID_i")
     ...
     self.threshold_n = thresh.get_thresh_value("ID_n")
     thresh.close_wb() # close threshold workbook
 def __call__(self):
     pass
```

Valid ID_i strings are either string of: "mu_bad", "mu_good", "mu_method","D", "d2w_low", "d2w_up", "det_low", " det_up", " fill ", "Fr", "freq", "h", " inverse_tcd ", "scour", "sf", "taux", "u". The get_thresh_value ("ID_i ") function is a routine of the ThresholdDirector class which is stored in `LifespanDesign/cThresholdDirector.py`. Modifications of the ThresholdDirector class are not recommended and threshold values should be modified in the spreadsheet `LifespanDesign/.templates/threshold_values.xlsx` (see Sec. 8.3).

If the new feature has subfeatures, the following template applies:

```
class NewFeature(Subfeature_1, Subfeature_2, ..., Subfeature_i, ...
    Subfeature_n):
  ## This is the Newfeature class inheriting from Subfeature_1 to Subfeature_n
      .
  def __init__(self, subfeature):
      self.lf = True  # identify if lifespan map applies
      self.ds = False # identify if design map applies
      if subfeature == 'subfeature_1':
          Subfeature_1.__init__(self)
      if subfeature == 'subfeature_2':
          Subfeature_2.__init__(self)
      if subfeature == 'subfeature_i':
          Subfeature_i.__init__(self)
      if subfeature == 'subfeature_n':
          Subfeature_n.__init__(self)
  def __call__(self):
      pass
```

Then, subfeature need to be defined, e.g., in an external file called `cNewSubFeature.py` (if so, add `from cNewSubFeature import *` at the top of `cFeatureLifespan.py`), according to the following class-template and for each subfeature (`Subfeature_1, ..., Subfeature_n`). Please note that the lifespan `self.lf = True / False` and design `self.ds = True / False` properties are already assigned in the inheriting feature class.

```
class NewSubFeature_i():
  ## This is the NewSubFeature_i class.
  def __init__(self):
      self.parameter_list = ["PAR1", "PAR2", ..., "PARi", , "PARn"] # Respect
          Hierarchy!; example: PAR1 = "hyd"
      self.shortname = "max6ch"
      thresh = ThresholdDirector(self.shortname) # instantiate reader of
          threshold values
      ## uncomment and adapt follow line if PAR = mu applies
      # self.mu_bad = thresh.get_thresh_value("mu_bad")
      # self.mu_good = thresh.get_thresh_value("mu_good")
      # self.mu_method = thresh.get_thresh_value("mu_method")
      self.threshold_1 = thresh.get_thresh_value("ID_1")
      self.threshold_2 = thresh.get_thresh_value("ID_2")
      ...
      self.threshold_i = thresh.get_thresh_value("ID_i")
      ...
      self.threshold_n = thresh.get_thresh_value("ID_n")
      thresh.close_wb() # close threshold workbook
  def __call__(self):
      pass
```

Part III
Maximum Lifespan Assessment (MaxLifespan)

13 Introduction to maxium (best) lifespan mapping

The *MaxLifespan* module serves for the GIS – based prioritization of stream restoration features based on lifespan and design maps and it creates `rasters`, `shapefiles`, `mxd`-layouts and `pdf`-maps. This chapter is structured as follows:

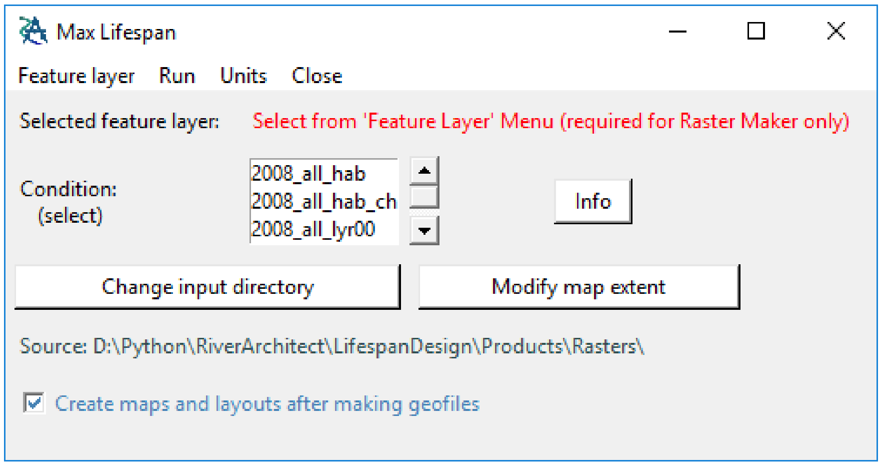
Section 14:	Quick Guide to the application of the GUI with description of required input (rasters), alternative run options and output descriptions.
Section 15:	Descriptions of outputs and procedures for half-automated pdf-map generation.
Section 16:	Detailed explanations of coding conventions with descriptions of extension possibilities.

Maximum lifespan mapping uses lifespan maps produced with the *LifespanDesign* to identify the feature(s) with the highest lifespan for every pixel within the three feature groups. If the maximum pixel lifespan can be obtained by several features, the *MaxLifespan* module overlays polygons indicating the best feature types. For terrain modifications, all relevant features (grading, widening/berm setback, backwater enhancement as well as side channel or side cavity creation) are equally considered. Thus, the planner has to decide and manually manipulate feature polygons which are relevant for the particular project. Regarding toolbox features, the *MaxLifespan* module evaluates plantings against wood (engineered log jams) and angular boulders (rocks) placement to increase habitat suitability and stabilize terrain modifications. Again the planner has to decide, which plantings, wood or angular boulders (rocks) polygons are relevant to keep for the final version. Finally, the *MaxLifespan* module uses complementary feature lifespan and design maps as well as terrain slope analysis to highlight areas where gravel augmentation, the incorporation of fine sediment in the soil and bioengineering features for terrain/slope stabilization are relevant. Also in this last step, the planner needs to decide, which feature polygons to keep. However, if the analysis of complementary features identifies unstable slopes, it is strongly recommended to take action in the concerned areas.

14 Quick GUIde to maximum lifespan maps

14.1 Main window set-up and run

The *MaxLifespan* module requires lifespan and design maps, i.e., the prior run of the *LifespanDesign* module is required. Then, the *MaxLifespan* module can be launches and Fig. 7 shows the *MaxLifespan* GUI after the module start-up.
First, the module requires the choice of a feature set from the dropdown menu. Second, a `condition` needs to be defined analog to the *LifespanDesign* module (exactly four characters, see Sec. 5).
By default, the *MaxLifespan* will look up lifespan and design maps that are stored in the folder `.../RiverArchitect/LifespanDesign/Products/Rasters/`*`condition`*`/`. This input directory can be modified by clicking on the `Change input directory` button. Furthermore, the extents of the maximum lifespan map output can be modified by clicking on the “Modify map extent” button, which opens an input file (`*.inp`) analog to the *LifespanDesign* module (Sec. 11.2).
The *MaxLifespan* will automatically look for raster files beginning with “`lf`” or “`ds`” and containing the shortname of the considered features (see shortname list in Sec. 4. Please note that raster names that do not start with either “`lf`” or “`ds`” and/or that do not contain the complete shortname of the considered features are not recognized by *MaxLifespan*.
The background image of the maximum lifespan maps also refers to lifespan and design maps and corresponds to the raster `.../RiverArchitect/01_Conditions/`*`condition`*`/back`.



Selected feature layer: Select from 'Feature Layer' Menu (required for Raster Maker only)

Condition: (select)
2008_all_hab
2008_all_hab_ch
2008_all_lyr00

Source: D:\Python\RiverArchitect\LifespanDesign\Products\Rasters\

☑ Create maps and layouts after making geofiles

Figure 7: GUI start up window.

The mapping check box provides the optional creation of maps with the creation of geofiles (rasters and shapefiles). If the check box is selected, running the Geofile Maker also includes the successive runs of the Layout Maker and Map Maker. It is recommended to keep this box checked (default) because maximum lifespan mapping is fully automated and the procedure is fast.

Once all inputs are defined, click on "Run" and "Verify settings" to ensure the consistency of the chosen settings. After successful verification, the selected feature list and the verified condition change to green font.

Three "Run" options exist in the drop-down menu:

- Run: Geofile Maker prepares the optimum lifespan raster and associated feature polygons (shapefiles) in the directories `RiverArchitect/MaxLifespan/Output/Rasters/`*`condition`*`/` and `.../Output / Shapefiles/`*`condition`*`/`
- Run: Layout Maker prepares `.mxd` layouts in the directory `RiverArchitect/MaxLifespan/Output/ Layout/`*`condition`*`/` (more information on layouts in Sec. 14.3.2).
- Run: Map Maker prepares maximum lifespan map assemblies (`pdfs`) in the directory `RiverArchitect/ MaxLife span/Output/Maps/`*`condition`*`/` (more information on layouts in Sec. 14.3.2)

14.2 Alternative run options

The three principal run options of the GUI call the following methods:

1. Run: Geofile Maker calls action_planner . geo_file_maker
2. Run: Layout Maker calls action_planner .layout_maker
3. Run: Map Maker calls action_planner .map_maker

In the batch processing of multiple scenarios, it can be useful to call the geo_file_maker from a script as a standalone. This can be done as follows:

1. Go to ArcGIS Python folder
   Example: `C:/Python27/ArcGISx64XX.X`

2. Launch `python.exe`
3. Enter `import os`
4. Navigate to Script direction using the command `os.chdir(" ScriptDirectory ")`
   Example: `os.chdir("D:/Python/ RiverArchitect /MaxLifespan/")`
5. Import the module: `import action_planner as ap`

6. Launch Geofile Maker: `ap.geo_file_maker(condition, feature_type, *args)`, where `args[0]` is a boolean value for activating or deactivation of integrated PDF-mapping (default = False), `args[1]` is a string that indicates the unit system (either "us" or "si"; default = "us") and `args[2]` can be an alternative input path of lifespan maps than the default directory (see above)
   Example: `ap.geo_file_maker(2008, "framework", True, "us", "D:/temp/")`
   This command calls the Geofile Maker for the condition "2008" for framework features, with activated mapping, U.S. customary units and it sets the raster input path to `D:/temp/`.

14.3 Output

14.3.1 Geofiles

The principal output of the module's Geofile Maker is one raster called `max_lf` (stored in `.../MaxLifespan/Output/Rasters/`*`condition`*`/`) and one shapefile per analyzed feature containing polygons of the feature's best performing areas (stored in `.../MaxLifespan/Output/Shapefiles/`*`condition`*`/`). Moreover, the module produces rasters with names corresponding to the lifespan/design raster names and feature shortnames, which essentially contain the same information as the feature shapefiles. These raster files are side products from the production of the feature shapefiles.

14.3.2 Layouts and Maps

The Layout Maker uses `.mxd` layout templates to overlay

- a background raster (`.../RiverArchitect/01_Conditions/`*`condition`*`/back`),
- the best lifespan raster (`.../MaxLifespan/Output/Rasters/`*`condition`*`/max_lf`) and
- shapefiles of best performing feature areas (`.../MaxLifespan/Output/Shapefile/`*`condition`*`/lf_feat...` ords`_feat...`).

The layouts templates are stored in `.../MaxLifespan/.templates/layouts/` and they are named after the feature set type; notably `framework.mxd`, `toolbox.mxd` and `complementary.mxd`. These templates can be changed to modify the maximum lifespan map layout, e.g., the legend, paper size, symbology or background source image. Apart from the background image raster, the shapefile and raster sources in the template `mxd`s refer to the *MaxLifespan*s output folder and the sources should not be modified. The Layout Maker chooses the correct layout as a function of the feature set type and copies this layout to the `.../MaxLifespan/Output/Layouts/`*`condition`*`/` directory.

The *MaxLifespan*'s Map Maker run-routine uses this layout copy (`.mxd`) and the map extent definitions (Sec. 14 and details in Sec. 11.2). Unlike in the *LifespanDesign* module, the production of maximum lifespan map `PDF`s completely automated and they are produced in `.../MaxLifespan/Output/Maps/`*`condition`*`/`.

The module enforces overwriting of existing files in the output folder and it tries to delete any existing content. Therefore, it is recommended to copy relevant outputs to the directory `.../MaxLifespan/Products/.../...`.

14.4 Quit module and logfiles

The GUI can be closed via the `Close` dropdown menu if no background processes are going on (see terminal messages). The GUI flashes and rings a system bell when it completed a run task. If layout creation and/or mapping were successfully applied, the target folder automatically opens. After execution of either run task, the GUI disables functionalities, which would overwrite the results and it changes button functionality to open logfiles and quit the program. Logfiles are stored in the `RiverArchitect/MaxLifespan/` folder and named `action_planner.log`. Logfiles from the previous runs are overwritten.

15 Working principle

The Geofile Maker uses the CellStatistics (with "Max" argument) command of arcpy's Spatial Analyst toolbox to identify the best lifespans of features. In the case of features where only design rasters are available, i.e., raster units are either on/off (1/0) or dimensional indicators (e.g., minimum grain sizes), the Geofile Maker converts any non-zero value of the design raster to 0.8. The value of 0.8 is an arbitrarily chosen identifier with the hypothetical unit of years, where the only importance is that this identifier is larger than zero and smaller than 0.9. Thus, the identifier is smaller than any lifespan value and the CellStatistic 's "Max" corresponds to the lifespan value when lifespan rasters are compared with design rasters. In other words, the Geofile Maker prioritizes lifespan rasters over design rasters. This choice was made because the data quality of lifespan rasters is better (higher data abundance) than the quality of design rasters, considering that the data quality is a function of available layers (DEM, morphological unit, grain size, hydraulic rasters, etc.). Therefore, pixels where no lifespan value but a design value is available to get assigned a value of 0.8. Finally, the 0.8-pixels are converted to lifespans of 20 years based on the assumption that if the feature is constructed corresponding to the design criteria, its lifespan will be high. Note the difference: lifespan values are prioritized because of the better data quality and the 20-years-value of design raster-only pixels applies to a chain of safe constructive assumptions potentially resulting in high costs.

Recall that `Other bioengineering` features can take three values: (1) `20.0` years, if the terrain slope is greater than defined in the thresholds workbook and the depth to groundwater is lower than defined in the thresholds workbook (cf. Sec. 8.3); (2) `1.0` year, if the terrain slope is greater than defined in the thresholds workbook and the depth to groundwater is greater than defined in the thresholds workbook; (3) `NoData`, if the terrain slope is lower than defined in the thresholds workbook. Thus, where maximum lifespan maps indicate a 1.0-year lifespan, bioengineering features that are independent of the depth to the groundwater table are required. Such features typically imply the placement of angular boulders.

16 Code modification: Add feature sets for maximum lifespan maps

The comprehensive *MaxLifespan* module provides flexibility regarding input directories, layout modifications and mapping extents without modifications of the code. However, modification of the feature sets (framework, toolbox and complementary) require code modifications. The relevant python classes are in the file `cFeatureActions.py`, notably class FrameworkFeatures(Director), class ToolboxFeatures(Director) and class ComplementaryFeatures (Director). These classes all inherit from the Director class which identifies and assigns lifespan and design rasters in the input folder. The following code example indicates where single features can be added or removed from feature sets. It is a generalized code sample where "Framework", "Toolbox" and "Complementary" are replaced with "TYPE". The feature FullName_i and shortame_i must comply with the terminology in Sec. 4 because also the *MaxLifespan* module uses a centralized feature identifier class that is stored in `RiverArchitect/ModifyTerrain/cDefinitions.py`.

```
class TYPEFeatures(Director):
    # This class stores all information about TYPE features
    def __init__(self, condition, *args):
        try:
            ## check if args[0] = alternative input path exists
            Director.__init__(self, condition, args[0])
        except:
            Director.__init__(self, condition)
        self.names = ["FullName_1", "FullName_2", ..., "FullName_n"] #---CHANGE
            HERE
        self.shortnames = ["shortname_1", "shortname_2", ..., "shortname_n"] #
           ---CHANGE HERE
        self.ds_rasters = self.append_ds_rasters(self.shortnames)
        self.lf_rasters = self.append_lf_rasters(self.shortnames)
```

Moreover, the choose_ref_layout(self, feature_type) function of the Mapper class in `MaxLifespan/cMap Actions.py` needs to be updated:

```
  def choose_ref_layout(self, feature_type):
    ## type(feature_type) == str
    if type(feature_type) == str:
      if feature_type == "framework":
        ref_layout_name = "framework.mxd"
      if feature_type == "toolbox":
        ref_layout_name = "toolbox.mxd"
      if feature_type == "complementary":
        ref_layout_name = "complementary.mxd"
      if feature_type == "NEW":
        ref_layout_name = "NEW.mxd"
    ...
```

This also requires the creation of the `NEW.mxd` layout in `MaxLifespan/.templates/layouts`.

Part IV
Modification of terrain (terraforming) assessment

17 Introduction to the ModifyTerrain module

The *ModifyTerrain* module can remodel the terrain DEM according to widen (berm setback) and grading threshold values to enable plantings. Moreover, the module quantifies mass movement volumes by comparing an initial DEM with a modified DEM. Modified DEMs can be automatically generated for widen and grading features based on maximum lifepsan maps or manually created for other framework features or any terrain modification. The module produces spreadsheets containing reach-wise volume differences (excavation and fill), modified raster DEMs, `mxd`-layouts and `pdf`-maps. This chapter explains the module application in the following sections:

Section 18:	Quick Guide to the application of the GUI with descriptions of input requirements and output descriptions.
Section 19:	Descriptions of outputs and procedures for half-automated pdf-map generation.
Section 20:	Detailed explanations of coding conventions with descriptions of extension possibilities.

Please note that an *ArcGIS* `3D` extension is required for running this module.

18 Quick GUIde to terrain assessment

18.1 Main window set-up and run

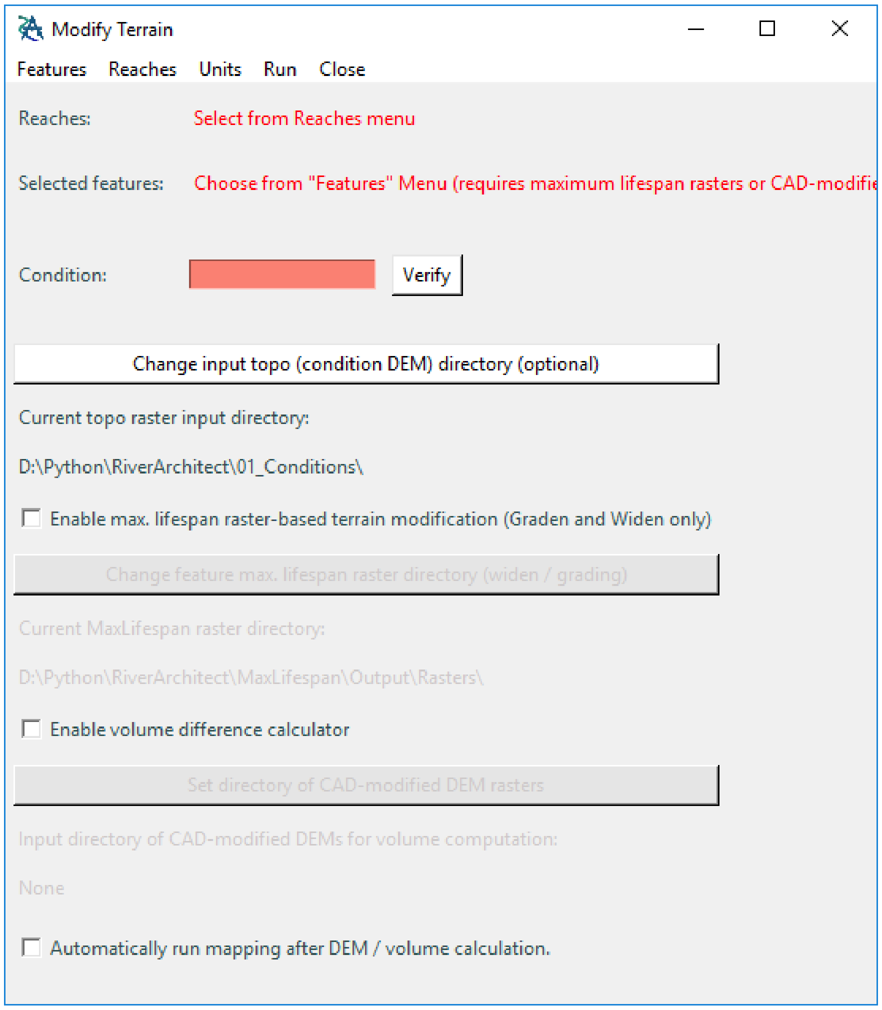
The GUI start-up takes a couple of seconds because the module updates reach information from a spreadsheet. Fig. 8 shows the *ModifyTerrain* GUI at start-up. First, the module requires the choice of a feature set from the dropdown menu, which limits to “CUSTOM”, “Widen” and “Grading”. Second, a `condition` (exactly four characters, corresponding to Sec. 5) needs to be defined, which requires a click on the “Verify” button to update the windows. This behavior is different from the *LifespanDesign* and *MaxLifespan* modules.

18.2 Input: Set initial DEM input folder

For terrain modifications, the module requires an input topo (DEM), which it looks up in the `.../RiverArchitect/LifespanDesign/Input/`*`condition`*`/` directory by default. The input directory can be modified by clicking on the “Change input topo (condition DEM) directory (optional)” button. Note that the input folder needs to contain a GRID-type DEM raster with the name `dem`; other raster names are not recognized and the input `dem` is crucial for any operation of the module.

18.3 Input: Set Reaches

A particularity of this module is that it enables running analysis for specific river reaches, which can be renamed and the reach extents can be modified. By default, the module analyzes all reaches which are defined in a spreadsheet stored in `/ModifyTerrain/.templates/computation_extents.xlsx`. This spreadsheet, shown in Fig. 9, can be opened by clicking on the `Modify Reaches` dropdown menu and then “`DEFINE REACHES`”.
The workbook enables the definition of up to eight reach names and the extents. The extents need to correspond to the input DEM coordinate and unit system types. In the example of Fig. 9, the unit system is `GRS_1980_Lambert_Conformal_Conic` with the linear unit of `Foot_US`. If the reaches 00 to 07 align from the East to the West, the `Max x` value of a reach corresponds to the `Min x` value of the next upstream reach. If a is situated in the south of an upstream reach, its `Max y` value corresponds to the `Min y` value of the upstream reach. These gap-less transitions


Modify Terrain

Features Reaches Units Run Close

Reaches: Select from Reaches menu

Selected features: Choose from "Features" Menu (requires maximum lifespan rasters or CAD-modifie

Condition: Verify

Change input topo (condition DEM) directory (optional)

Current topo raster input directory:

D:\Python\RiverArchitect\01_Conditions\

☐ Enable max. lifespan raster-based terrain modification (Graden and Widen only)

Change feature max. lifespan raster directory (widen / grading)

Current MaxLifespan raster directory:

D:\Python\RiverArchitect\MaxLifespan\Output\Rasters\

☐ Enable volume difference calculator

Set directory of CAD-modified DEM rasters

Input directory of CAD-modified DEMs for volume computation:

None

☐ Automatically run mapping after DEM / volume calculation.

Figure 8: GUI start up window.

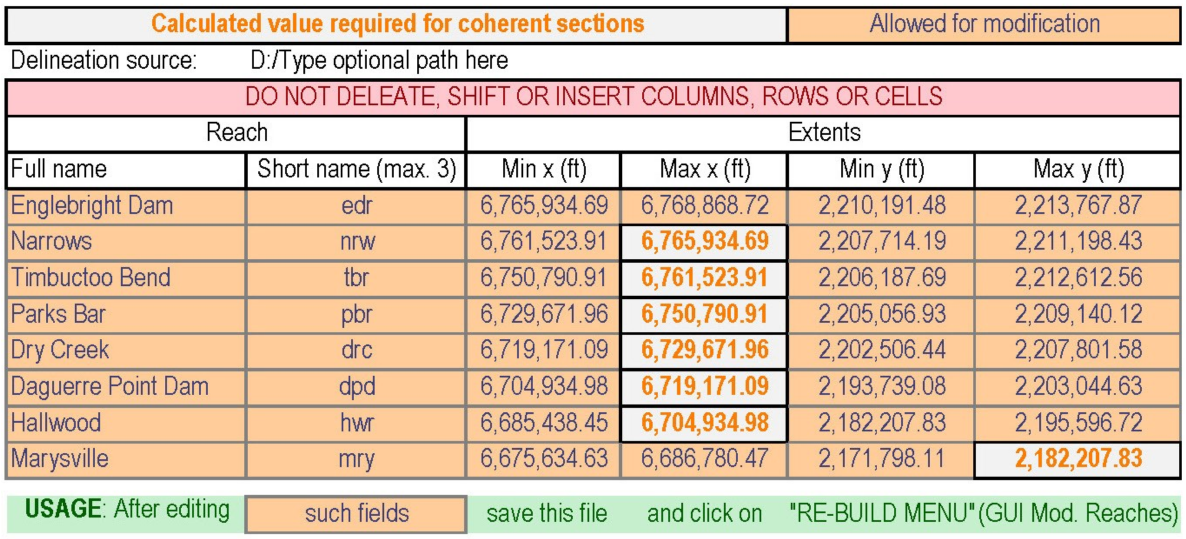
enable consistent mapping of DEM differences and excavation/fill volume calculations.

After editing, saving and closing the spreadsheet, the GUI window can be updated by clicking on the `Modify Reaches` dropdown menu and then "`RE-BUILD MENU`". Whatever name is stored in the spreadsheet, the module uses internal identifiers that point at the rows in the spreadsheet, and therefore, output rasters are enumerated with tags `r00`, `r01`, ... `r07`.

All reaches can be deselected by clicking on "`CLEAR ALL`" to add particular reaches only. If more than five reaches are selected, the GUI truncates the list and displays `Many / All`.

18.4 Input: CUSTOM DEM options

If the "CUSTOM: Use CAD-modified DEM" feature was selected, the `Enable volume difference calculator` check box is auto-selected as required module operation. In addition, the check box `Automatically run mapping after DEM / volume calculation` can be selected to map volume differences between the initial DEM and the customary modified DEM. By default, the module looks for a modified DEM in the folder `ModifyTerrain/Input/DEM/`*`condition`*`/` and the modified DEM needs to have either the string `dem` or a



Calculated value required for coherent sections				Allowed for modification	
Delineation source:	D:/Type optional path here				
DO NOT DELEATE, SHIFT OR INSERT COLUMNS, ROWS OR CELLS					
Reach		Extents			
Full name	Short name (max. 3)	Min x (ft)	Max x (ft)	Min y (ft)	Max y (ft)
Englebright Dam	edr	6,765,934.69	6,768,868.72	2,210,191.48	2,213,767.87
Narrows	nrw	6,761,523.91	6,765,934.69	2,207,714.19	2,211,198.43
Timbuctoo Bend	tbr	6,750,790.91	6,761,523.91	2,206,187.69	2,212,612.56
Parks Bar	pbr	6,729,671.96	6,750,790.91	2,205,056.93	2,209,140.12
Dry Creek	drc	6,719,171.09	6,729,671.96	2,202,506.44	2,207,801.58
Daguerre Point Dam	dpd	6,704,934.98	6,719,171.09	2,193,739.08	2,203,044.63
Hallwood	hwr	6,685,438.45	6,704,934.98	2,182,207.83	2,195,596.72
Marysville	mry	6,675,634.63	6,686,780.47	2,171,798.11	2,182,207.83
USAGE: After editing	such fields	save this file	and click on	"RE-BUILD MENU"(GUI Mod. Reaches)	

Figure 9: Spreadsheet with reach definitions (stored in `ModifyTerrain/.templates/computation_extents.xlsx`).

feature shortname (Sec. 4) in its name for getting recognized. The directory of the modified DEM can be changed by clicking on the `Set directory of CAD-modified DEM rasters` button, but the **name convention (raster DEM name contains `dem` or feature shortname) always needs to be respected**. Refer to Sec. 19.2 for more information on volume difference (fill/excavate) calculation with customary DEMs.

18.5 Input: Widen and Grading options

The “Widen” and “Grading” features use the maximum required distance to the groundwater table, which is admissible for plantings. These threshold values are defined in the *LifespanDesign* modules spreadsheet `RiverArchitect/LifespanDesign/Input/.templates/threshold_values.xlsx` (see explanations in Sec. 8.3). If the `Enable max. lifespan raster-based terrain modification (Grading and Widen only)`
check box is selected, the module provides the option `Run: DEM Modification` to apply the threshold values defined in the cells `J6:M6` for lowering the terrain where maximum lifepsan rasters indicate that widening and grading are most pertinent. Thus, the prior run of the *MaxLifespan* module is required to enable *ModifyTerrain* reading rasters containing the keywords `grade` or `widen` from the folder `RiverArchitect/MaxLifespan/Output/Rasters/`*`condition`*`/`. Moreover, a depth to groundwater table raster (GRID format) with the name `d2w` is required in the directory `RiverArchitect/LifespanDesign/Input/`*`condition`*`/`.
The directory of maximum lifespan and depth to groundwater rasters can be modified by clicking on the `Change feature max. lifespan raster directory (widen/grading)` button, but there need to be rasters in the defined folders which contain the keywords `grade` or `widen` in their name. The DEM modification auto-selects the `Enable volume difference calculator` check box. Please note that the volume calculator is executed after the automated terrain modification, even if the check box is deselected.
Selecting the check box `Automatically run mapping after DEM / volume calculation` enables mapping of volume differences between the initial DEM and the customary modified DEM, as well as mapping of the modified DEMs after widening and grading.

18.6 Input: Prepare mapping layouts

The Run: Map Maker uses layout files (`.mxd`) stored in the directory `ModifyTerrain/Input/Layouts/`*`condition`*`/`. Template layout files are provided in `ModifyTerrain/Input/Layouts/Templates/` and need to be manually copied and adapted to the *`condition`*:

1. Copy relevant layouts from `ModifyTerrain/Input/Layouts/Templates/`.
2. Create new folder `ModifyTerrain/Input/Layouts/`*`condition`*`/` and paste copied layouts in this folder.
3. Open copied layouts in *ArcMap* and adapt links to raster source files, page setup, symbology, legend title, background image source or any other styles.
   Hint 1: The layer names in the templates refer to distinct reaches. Do not remove, add or rename layers, even if the source is missing.
   Hint 2: Ensure that the raster sources in the `_neg.mxd` file point at rasters ending on "_d_neg" in the directory `ModifyTerrain/Output/Rasters/condition/`, and similar for `_pos.mxd` files.
   Hint 3: Therefore, it is recommended to not auto-include mapping in the case of `widen` and/or `grading`.

18.7 Run

All run options in the `Run` dropdown menu are deactivated at the GUI start-up and relevant run options will become available as a function of the selected feature types:

- Run: DEM Modification is available if grading and/or widen are among the chosen features (descriptions in Sec. 18.5); creates modified DEMs and automatically calculates volume differences.
- Run: Volume Calculator becomes available since the selection of any feature; calculates fill and excavation volumes by comparing the input condition DEM with a modified DEM.
- Run: Map Maker prepares map assemblies (`pdfs`) of modified rasters and/or volume differences maps of selected reaches in the directory `RiverArchitect/ModifyTerrain/Output/Maps/`*`condition`*`/` (more information on layouts in Sec. 18.9.2).

18.8 Alternative run options

The *ModifyTerrain* module has no standalone statement and it is recommended to use the GUI for launching the modules routines. If needed, the module can alternatively be imported and used as python package as follows:

1. Go to ArcGIS Python folder
   Example: `C:/Python27/ArcGISx64XX.X`
2. Launch `python.exe`
3. Enter `import os`
4. Navigate to Script direction using the command `os.chdir(" ScriptDirectory ")`
   Example: `os.chdir("D:/Python/ RiverArchitect /ModifyTerrain/")`
5. Import the module: `import cModifyTerrain as cmt`

6. Instantiate a *ModifyTerrain* object:
   `mt = cmt.ModifyTerrain( condition , unit_system , feature_ids , topo_in_dir , feat_in_dir , reach_ids )`
   `unit_system` must be either “us” or “si”
   `feature_ids` is a list of features shortnames (Sec. 4)
   `topo_in_dir` is an input directory for DEM and depth to groundwater table rasters
   `feat_in_dir` is an input directory for feature max. lifespan rasters; for custom DEMs, this can be a dummy directory
   `reach_ids` is a list of reach names to limit the analysis

7. The DEM Modification is launched by calling the *ModifyTerrain* object: `logfile = mt()`

8. The analysis is limited to running the Volume Calculator when the *ModifyTerrain* object is called with arguments:
   `logfile = mt(True, path_to_modified_DEM)`

9. Mapping requires importing the modules mapping class file:
   `import cMapModifiedTerrain as cmat`

10. A map object is instantiated with: `mapper = cmat.Mapper(condition, feature_shortname )`

11. Automatically generated DEMs of adapted terrain after grading or widening can be mapped by looping over relevant reach IDs as defined in the spreadsheet (Sec. 18.3): `for rID in reach_ids :`
    `mapper.map_reach(rID, feature_shortname , volume_type=−1)`
    If the `volume_type` is -1, excavation areas are mapped and if the `volume_type` is 1, fill areas are mapped.

12. Terrain elevation differences between an initial (condition-defined) DEM and a customary modified DEM can be mapped with:
    `mapper.map_custom(self. in_vol , volume_type =...)`

13. IMPORTANT: The final step for drawing maps is entering: `mapper. finalize_map ()`
    The command prompt informs about mapping progress and occasional warning/error messages.

18.9 Output

18.9.1 Rasters

The module creates rasters of modified DEMs for grading and/or widen features and terrain difference rasters for all relevant feature types (grading, widen, custom) in the directory `.../ModifyTerrain/Output/Rasters/`*`condition`*`/`). Raster names contain a reach identifier (`r00`, `r01`, ... `r07` corresponding to spreadsheet rows 6–13), part of the feature shortname and, if it is a terrain **d**ifference raster, “`d`” with either “`neg`” for excavation or “`pos`” for fill.

18.9.2 Layouts and Maps

The Map Maker uses `.mxd` layout templates stored in `.../ModifyTerrain/.templates/layouts/` to overlay

- a background raster (`.../RiverArchitect/LifespanDesign/Input/`*`condition`*`/back`) and
- volume difference rasters stored in (`.../ModifyTerrain/Output/Rasters/`*`condition`*`/`).

18.9.3 Spreadsheets

The resulting volume differences are reach-wise written to a spreadsheet in the directory `.../ModifyTerrain/Output/Spreadsheets/`. This folder contains a template called `volumes_template.xlsx`, which must not be modified. When *ModifyTerrain* is run for the first time on a DEM *`condition`*, it creates a copy of the spreadsheet template, which is called *`condition`*`_volumes.xlsx`. In this spreadsheet, *ModifyTerrain* copies the `template` sheet twice per run. One of the copies is called `excavate_YYYYMMDD_HHhMM` and lists the reach-wise required excavation volumes in the chosen unit system. The other copy is called `fill_YYYYMMDD_HHhMM` and lists the reach-wise required fill volumes in the chosen unit system. The strings `YYYYMMDD` and `HHhMM` indicate the date and time of program execution. Anew runs of *ModifyTerrain* on the same *`condition`* will append two more copies (excavate and fill) of the `template` sheet with the date-time indicator. It is recommended to cut-paste *`condition`*`_volumes.xlsx` in the `.../ModifyTerrain/Products/` directory after every run to keep results well-arranged and to force the module to create a new *`condition`*`_volumes.xlsx` file for every run.

18.10 Quit module and logfiles

The GUI can be closed via the `Close` dropdown menu if no background processes are going on (see terminal messages). The GUI flashes and rings a system bell when it completed a run task. If mapping was successfully applied, the target folder automatically opens. After execution of either run task, the GUI disables functionalities, which would overwrite the results and it changes button functionality to open logfiles and quit the program. Logfiles are primarily stored in the `RiverArchitect/ModifyTerrain/` folder and named `logfile_YYYYMMDD.log`. Logfiles from the same date are overwritten and safe copies of logfiles are made in `RiverArchitect/ModifyTerrain/Output/Logfiles/`. The input and output class produces its own logfiles called `IO_logger.log`. This decoupled logging is necessary to enable problem identification in the reach-defining spreadsheet, which is used on multiple code levels.

19 Working principles

19.1 Modify terrain DEM

The module can lower the terrain for grading and/or widen features to make relevant areas adequate for plantings. It looks up the maximum possible depth to groundwater for the considered planting types in `RiverArchitect/LifespanDesign/Input/.templates/threshold_values.xlsx`, cells `J6:M6`. The required lowering dz results from the minimum depth to groundwater value of the latter cells:
`required_d2w = min([plant1.threshold_d2w_up, plant2.threshold_d2w_up, plant3.threshold_d2w_up, plant4.threshold_d2w_up])`
The *`condition`* DEM (act_dem) is lowered using the arcpys spatial analyst:
`new_dem = Con((d2w > required_d2w), Float(act_dem - (d2w - required_d2w)), act_dem)`

19.2 Volume differences

The *`condition`* DEM (act_dem) is subtracted from the new_dem of grading or widen features, or the mod_dem of customary modifications to obtain a difference DEM diff_dem indicating the dz differences in elevation. The module assumes that a customary modified DEM results from Contour line modifications that were transformed to a raster (mod_dem). This transformation uses interpolations that cause imprecision in the raster DEM leading to virtual surface difference between the *`condition`* DEM and the modified DEM. Therefore, the module uses a level of change detection lod of 0.99 ft (or 0.30 m) to eliminate such virtual differences: `new_dem = Con(ABS(act_dem - mod_dem)>= lod, mod_dem, 0)`.
Then, the difference DEMs result from
`diff_dem_pos = Con(act_dem < new_dem, new_dem - act_dem, 0)` for fill and

diff_dem_neg = Con(act_dem >= new_dem > 0, act_dem − new_dem, 0) for excavations.
The volume of excavation and fill results from arcpys SurfaceVolume_3d function, which requires an *ArcGIS* 3D extension:
volume_fill = arcpy.SurfaceVolume_3d(diff_dem_pos, "", "ABOVE", 0.0, 1.0)
volume_excavation = arcpy.SurfaceVolume_3d(diff_dem_neg, "", "ABOVE", 0.0, 1.0)
The variable volume_fill and volume_excavation are then written to the output spreadsheet (Sec. 18.9.3).

20 Code modification: Feature sets for maximum lifepsan maps

20.1 Change sensitivity threshold (lod) for terrain modification detection

The lod variable serves for the elimination of virtual terrain differences that result from the interpolation of rasters from contour lines (see explanation in Sec. 19.2). The internal variable name for lod is self.volume_threshold and it is defined in the initiator of the *ModifyTerrain* class (file `.../ModifyTerrain/cModifyTerrain`). The assigned values of 0.99 (U.S. customary) or 0.30 (SI metric) can be changed in class ModifyTerrain() → def __init__ (self, condition, ...) : → *## set unit system* paragraph:

```
if self.units == "us":
    self.convert_volume_to_cy = 0.037 #ft3 -> cy: float((1/3)**3)
    self.unit_info = " cubic yard"
    self.volume_threshold = 0.99 # -- CHANGE lod US customary HERE --
else:
    self.convert_volume_to_cy = 1.0
    self.unit_info = " cubic meter"
    self.volume_threshold = 0.30 # -- CHANGE lod SI metric HERE --
```

Ensure that a layout exists in `ModifyTerrain/Input/Layouts/`*`condition`*`/` according to the descriptions in Sec. 18.6.

20.2 Add routine for automated DEM modification

Other routines for the automated generation of modified terrains can be added as follows:

1. Create new function in the *ModifyTerrain* class (file `.../ModifyTerrain/cModifyTerrain`), which contains routines for creating a new DEM, for example:

```
def create_new_dem(self, feat_id, extents):
  self.logger.info("")
  feature_name = self.features.feat_name_dict[feat_id]
  self.logger.info("* *  * *  * * " + feature_name.capitalize() + " *
     *  * *  * *")
  ## set arcpy env
  arcpy.gp.overwriteOutput = True
  arcpy.env.workspace = self.cache
  if not (type(extents) == str):
    try:
      ## XMin, YMin, XMax, YMax
      arcpy.env.extent = arcpy.Extent(extents[0], extents[1], extents
          [2], extents[3])
      except:
          self.logger.info("ERROR: Failed to set reach extents.")
          return (-1)
  else:
      arcpy.env.extent = extents
  # arcpy.CheckOutExtension('Spatial')  # check out license if needed

  ## get feature maximum lifepsan raster (or any other input raster):
  feat_act_ras = self.get_action_raster(feat_id)
  ## set NoData values to 0:
  feat_ras_cor = Con(IsNull(feat_act_ras), self.null_ras, feat_act_ras)
  self.logger.info("  >> Calculating DEM after terrain " + feature_name
      + " ... ")

  ## assign a dem for modification (see descriptions below)
  if self.raster_info.__len__() > 0 and not ("diff" in self.raster_info)
      :
      ## use modified DEM if there was a prior automated modification
      self.logger.info("    ... based on " + str(self.raster_info) + "-DEM
          ... ")
      dem = self.raster_dict[self.raster_info]
      ...add other required rasters
  else:
      ## use condition DEM if there was no prior raster modification
      dem = self.ras_dem
      ...add other required rasters

  ## IMPLEMENT FORMULAE HERE
  new_dem = ...some function...
  ## calculated difference DEM for volume calculation
  new_dem_diff_neg = ...
  new_dem_diff_pos = ...

  ## update class dictionaries (communicate modifications)
  self.raster_dict.update({feat_id[0:3] + "_diffneg": new_dem_diff_neg})
  self.raster_dict.update({feat_id[0:3] + "_diffpos": new_dem_diff_pos})
  self.raster_info = feat_id[0:3]
  self.raster_dict.update({self.raster_info: new_dem})

  # arcpy.CheckInExtension('Spatial')  # release license if necessary
```

Note:

- The `self`, `feat_id`, `extents` arguments are required for the implementation in the call-routine, where `feat_id` is a feature shortname (Sec. 4) and `extents` is an `arcpy.Extent` variable that limits DEM creation to this extent.
- `self.logger.info()` sends messages to the logger, which are also printed in the terminal.
- `dem = self.raster_dict[self.raster_info]` uses the latest DEM version; this is the `condition` DEM if no other terrain modification was applied before. Otherwise, for example if "grading" was used for the

automated terrain modification before this function is used, dem = self . raster_dict [self . raster_info] points at the terrain DEM after grading.

2. Implement the new function in the modification manager:

```
def modification_manager(self, feat_id):
  if not(self.reach_delineation):
    extents = "MAXOF"
  else:
    try:
      extents = self.reader.get_reach_coordinates(self.reaches.
          dict_id_int_id[self.current_reach_id])
    except:
      extents = "MAXOF"
      self.logger.info("ERROR: Reach delineation recognized but not
          identifiable in input
    ## START CHANGE FROM HERE ON
    if ("grad" in feat_id) or ("wide" in feat_id):
      self.lower_dem_for_plants(feat_id, extents)
    if ("feature_shortname" in feat_id):
      self.create_new_dem(feat_id, extents)
```

3. Save edits

4. The adapted code can now be executed using the alternative run options described in Sec. 18.8, where feature_ids = ["shortname of new feature"].
Hint: The new method can also be implemented in the GUI by adding a self .featmenu.add_command(label= "New Feature", command=lambda: self. define_feature ("new ID") to def __ini__ (...) of the FaGui() class in the file `modify_terrain_gui.py`. This requires adding an if not(feature_id == "new ID"): ... else : ... statement in the self . define_feature function according to the function environment.

Part V
Habitat Evaluation

21 Introduction to Habitat Suitability evaluation

The *HabitatEvaluation* module creates habitat suitability index (HSI) rasters for various fish species and combines multiple HSI rasters into a composite habitat suitability index raster (cHSI or CSI). The habitat suitability index ranges between 0.0 and 1.0, according to Bovee (1986). It uses a threshold value for defining valuable habitat, which is initially set to 0.4; i.e., HSI values between 0.0 and 0.4 or `NoData` are considered as ”non-habitat” and values between 0.4 and 1.0 correspond to valuable habitat. Currently, only hydraulic habitat suitability rasters can be calculated based on flow depth and velocity rasters for multiple discharges. A minimum of three normal discharges within the annual flow duration curve should be analyzed, e.g., the Q_{300}, Q_{200} and Q_{100}, which denote the flows that are exceeded during 300, 200 and 100 days per year, respectively. The *River Architect Tools* (Sec. 2.2) provide the make_flow_duration routine to produce flow duration curves if required. *HabitatEvaluation* uses the annual flow exceedance probabilities that are associated with the cHSI rasters for summing up the surface where the cHSI is larger than the threshold value. This surface corresponds to the Weighted Usable habitat Area (WUA) in [yd^2 per year] or [m^2 per year]. The module writes relevant flows, exceedance properties and the WUA to `condition` related spreadsheets in the `HabitatEvaluation/WUA/` directory. The next sections explain the module usage:

Section 22:	Quick Guide to the application of the GUI.
Section 23:	Working principles of the module.
Section 24:	Descriptions of code modification possibilities.

22 Quick GUIde to habitat suitability evaluation

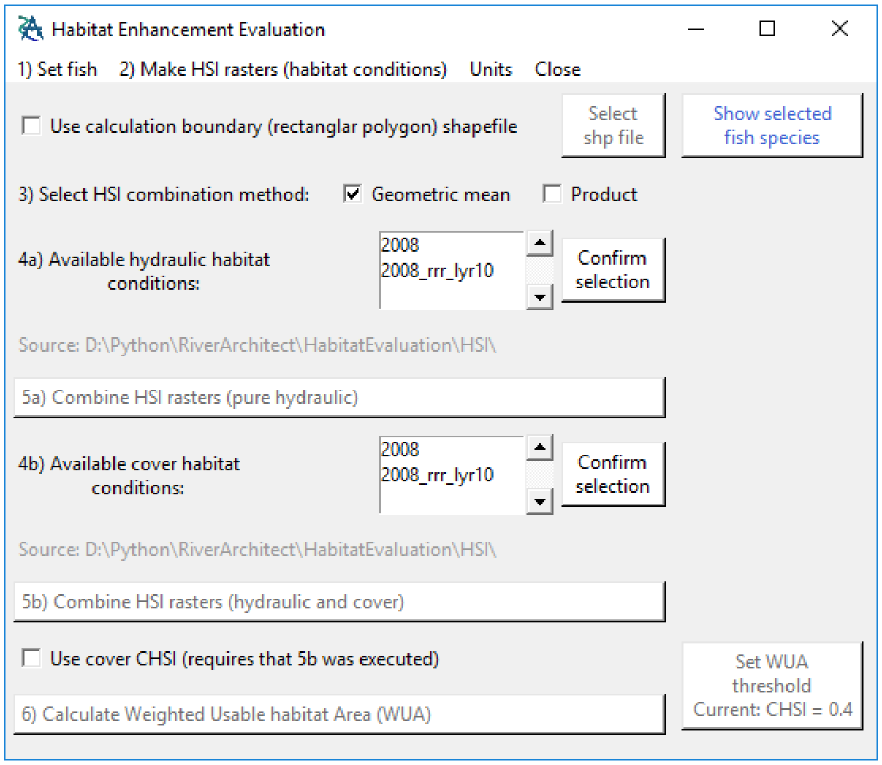
22.1 Main window set-up and run

Fig. 10 shows the *HabitatEvaluation* GUI at start-up. First, the module requires a definition of relevant fish and lifestages that it reads from a workbook (see Sec. 22.2). Second, hydraulic habitat suitability rasters and related discharge exceedance probabilities need to be calculated (Sec. 22.5). This last step creates habitat conditions, which can be selected in the third step. Step four combines flow depth and velocity habitat suitability rasters (Sec. 22.7). Step five computes the WUA (Sec. 22.8).

22.2 Input: Fish

The Set fish menu enables the definition of flow depth and velocity dependent habitat suitability curves. The DEFINE FISH SPECIES menu entry opens the `Fish.xlsx` workbook, which is located in `HabitatEvaluation/.templates/`. The `Fish.xlsx` workbook contains the definition of fish species names (rows 2 to 4) and up to four lifestages per species. For every lifestage, piece-wise linear habitat suitability curves can be entered as a function of the following parameters:

- Flow velocity u in row 7 to 34.
- Flow depth h in row 36 to 68.
- Substrate (grain size) D in row 70 to 77.
- Cover (minerals) in row 79 to 80. `Min%` describes the minimum surface occupation of either `Cobble` or `Boulder` that is required to improve habitat by a HSI value.


Habitat Enhancement Evaluation
1) Set fish
2) Make HSI rasters (habitat conditions)
Units
Close
Select shp file
Show selected fish species
Confirm selection
5a) Combine HSI rasters (pure hydraulic)
Confirm selection
5b) Combine HSI rasters (hydraulic and cover)
Set WUA threshold
6) Calculate Weighted Usable habitat Area (WUA)

☐ Use calculation boundary (rectanglar polygon) shapefile

3) Select HSI combination method: [✓ Geometric mean] ☐ Product

4a) Available hydraulic habitat conditions: 2008, 2008_rrr_lyr10

Source: D:\Python\RiverArchitect\HabitatEvaluation\HSI\

4b) Available cover habitat conditions: 2008, 2008_rrr_lyr10

Source: D:\Python\RiverArchitect\HabitatEvaluation\HSI\

☐ Use cover CHSI (requires that 5b was executed)

Current: CHSI = 0.4

Figure 10: GUI start up window.

- Cover (vegetative) in row 82 to 83. `Rad.` defines the radius around single `Plants` or `Wood` placements, where habitat improves by a HSI value.

Ensure the application of the correct unit system; the drop down menu in the `Fish.xlsx` workbook automatically sets the units of flow velocity u, flow depth h, grain size D, and delineation radius Rad around polygons. The radius Rad describes the "impact" perimeter of boulders, plants and / or wood that is drawn around the delineated polygons.
The base scenario provides habitat suitability curves for four sample fish species. More fish species can easily be appended by copy-pasting the template frame (area in thick borders in the `template` sheet) after the last defined fish species. For example, if another fish species is added to the base scenario, cells `C2` to `J83` from the `template` sheet are copied and pasted at cell `AI2` in the `fish` sheet. However, the number of lifestages per fish species and the above-stated rows need to be respected when entering piece-wise linear habitat suitability functions.
The structure of `Fish.xlsx` must not be modified (inserting or deleting rows or columns), unless the module's source code is also changed (not recommended). If the structure is changed anyway, the module needs to be modified as explained in Sec. 24.
Note that any relevant species-lifestage needs to have at least one entry for the velocity habitat suitability curve, as the module uses this first data cell in every column to verify if it contains data or not. For example, if a substrate habitat suitability curve is given, but the velocity habitat suitability curve is left blank, the concerned lifestage will not be considered relevant.

The module uses the piece-wise linear curves of habitat suitability indices to interpolate the HSI value of raster pixels. For example, if a velocity raster's pixel has a value of 0.51 (fps or m/s), the module looks up the HSI values related to the next smaller provided value (e.g., 0.5 fps or m/s) and the next higher value (e.g., 0.6 fps or m/s) and linearly interpolates the habitat suitability index for 0.51 (fps or m/s).

22.3 Input: Combine methods (habitat suitability rasters

The module provides the options of either using the geometric mean or the product to combine depth and velocity rasters (and eventually cover rasters). The following formulae are implemented to combine a depth HSI raster $DHSI$ with a velocity HSI raster $VHSI$ to a $cHSI$ raster.

$$\text{Geometric mean:} \quad cHSI = \sqrt{DHSI \cdot VHSI} \tag{2}$$

$$\text{Product:} \quad cHSI = DHSI \cdot VHSI \tag{3}$$

If a cover HSI raster $covHSI$ is used, the following formulae apply:

$$\text{Geometric mean:} \quad cHSI = \sqrt[3]{DHSI \cdot VHSI \cdot covHSI} \tag{4}$$

$$\text{Product:} \quad cHSI = DHSI \cdot VHSI \cdot covHSI \tag{5}$$

The cover HSI raster $covHSI$ represents the maximum pixel values of applied cover types (see Sec. 23.6 for details).

22.4 Input: Define computation boundaries

A boundary shapefile (polygon) can be selected to limit the calculation extents and assessment of the Weighted Usable Area. Typically, that shapefile should be stored in `.../01_Conditions/`*`condition`*`/boundary.shp` and it should contain one valid rectangle with an `Id` field value of `1` for that rectangle in the `Attribute table`.

22.5 Input: HHSI

Before habitat suitability rasters can be calculated, at least one fish species/lifestage needs to be selected (multiple selection is possible). Then, HSI rasters can be generated by clicking on the `Generate HSI rasters` menu and `Flow depth and velocity HSI`. A new window opens and first asks for a discharge (or flow) duration curve. Clicking the associated button opens the file explorer in `HabitatEvaluation/FlowDurationCurves/` , where workbooks containing flow duration curves are located. A new flow duration curve can be generated with the make_flow_duration routine of *River Architect*'s *Tools* (Sec. 2.2) and using `flow_duration_template.xlsx`. Any flow duration workbook needs 365 discharges (for 365 days per year) listed in column `B`, starting at row 2 in descending order. The discharges need to be positive float numbers. The associated exceedance durations (days per year) are stated in column `C`.
Second, hydraulic habitat conditions need to be selected. The module looks up available hydraulic habitat conditions in `RiverArchitect/01_Conditions/`. After highlighting (click) one of the available hydraulic conditions, a click on the `Confirm selection` button generates a workbook in `RiverArchitect/Habitat Evaluation/WUA/` with the name *`condition_fil`*`.xlsx` for each previously selected fish. The *`fil`* string abbreviates the selected fish species and lifestage, where *`fi`* represents the first two letters of the fish species and *`l`* the first letter of the fish lifestage. Existing workbooks for the same condition and fish are renamed (`_old` gets appended to the file name). Older `..._old.xlsx` workbooks are overwritten.
The generated *`condition_fil`*`.xlsx` can be opened by clicking on the `Optional: View discharge dependency file` button. If opened, close this workbook before continuing. Until here, only the columns `B` to `E` should contain values, which constitute the plotted flow duration curve.
Finally, a click on `Run (generate habitat condition)` launches the calculation of hydraulic habitat suitability index (HHSI) rasters, which are created in `RiverArchitect/HabitatEvaluation/HSI/`*`con- dition`*`/` . The window starts flashing when the calculation finished. For returning to the main window (it partially freezes while the HHSI window is open), click on the `RETURN` button.

22.6 Input: Cover HSI

As before, at least one fish species/lifestage needs to be selected (multiple selection is possible). The cover HSI raster generation can be limited to a user-defined flow region by selecting one of the `hxxx` raster names in the 2) Define

flow region section. However, the later combination of the cover HSI rasters with the HHSI (hydraulic HSI) rasters will automatically limit the usable habitat area to wetted pixels only. Thus, the most pertinent choice here is selecting `all terrain`. Click on `Confirm selection` to do so.
Relevant cover types can be selected by checking the according checkboxes, where the geofiles are required to be stored in `01_Conditions/`*`condition`* apply the cover types:

- Substrate: A `dmean` (S.I. /metric units) or `dmean_ft` (U.S. customary units) raster is required, see details in Sec. 5.
- Boulders: A `boulders.shp` polygon shapefile is required; the polygons delineating boulders need to have an `Short Integer`-type field called `cover` in the (`Attributes table`) and the `cover` field value of polygons is `1`.
- Cobbles: A `dmean` (S.I. /metric units) or `dmean_ft` (U.S. customary units) raster is required, see details in Sec. 5. Cobble is defined, where the `dmean...` raster indicates grain sizes between 0.064 m and 0.256 m.
- Plants: A `plants.shp` polygon shapefile is required; the polygons delineating boulders need to have an `Short Integer`-type field called `cover` in the (`Attributes table`) and the `cover` field value of polygons is `1`.
- Wood: A `wood.shp` polygon shapefile is required; the polygons delineating boulders need to have an `Short Integer`-type field called `cover` in the (`Attributes table`) and the `cover` field value of polygons is `1`.

The geofiles are used with the habitat suitability (curve) definitions in the `Fish.xlsx` workbook (tab `fish`), which is located in `HabitatEvaluation/.templates/`.

HINT: The applicable cover types are limited to the terms ”Substrate”, ”Boulders”, ”Cobbles”, ”Plants”, and ”Wood”. Bridge piers or other structural turbulence objects may constitute other cover types that are not explicitly implemented in the *HabitatEvaluation* module. However, may cover types can be associated with similar effects as the implemented cover types. Thus, other cover types can be added as polygons in the shapefiles for ”Boulders”, ”Plants”, or ”Wood” cover types.

22.7 Combine habitat suitability rasters

Back in the main window, select one available habitat condition (3) and confirm the selection. The available habitat conditions refer to the conditions created with the `Generate HSI rasters` routines (Sec. 22.5). Confirming the selection activates the `Combine HSI rasters ...` buttons for launching the combination of HSI rasters. The HSI rasters can be combined either using the geometric mean or as their product by (un-)checking one of the checkboxes above the `Combine HSI rasters ...` buttons. The default combine method is `Geometric mean`. For more details, see Sec. 22.3.
Two combination buttons are available: a) `WITHOUT COVER` and b) `WITH COVER`. Additional habitat in terms of turbulent eddies created by cobbles, boulders, submerged plants and streamwood is not well determined by 2D numerical models. `COVER` adds additional habitat as a function of the relative cobble or boulder surface and the proximity of plants or streamwood. This method values artificially placed cobbles, boulders, submerged plants and streamwood in stream restoration projects. However, the gain in WUA by using `COVER` methods is not satisfactory and the automation requires considerable efforts regarding the manual delineation of stream restoration elements. Therefore, the `WITH COVER` routines are currently only implemented as development elements without effective functionality. Currently, use the `Combine HSI rasters WITHOUT COVER` button to create cHSI rasters, which are produced in `RiverArchitect/HabitatEvaluation/CHSI/`*`habitat_condition`*`/no_cover/`.

22.8 Calculate WUA

The `Usable Area Analysis ...` buttons launch the calculation of usable habitat area based on the combined habitat suitability index (cHSI) rasters (Sec. 22.7). Usable (habitat) area is defined as the surface where cHSI (or CSI) pixel values are larger than the `WUA threshold`. By default, this threshold value is 0.4; i.e., the routine sums up the surface of pixels where the cHSI is larger than 0.4. The threshold value can be changed by clicking on the `Set WUA threshold` button.
Launch the WUA calculation by clicking on `Usable Area Analysis ...`. As before, only the `WITHOUT COVER` option effectively calculates the usable habitat area, which is saved as raster in `RiverArchitect/HabitatEvaluation/WUA/Rasters/`*`habitat_condition`*`/no_cover/`. WUA is calculated in the previously created `RiverArchitect/HabitatEvaluation/WUA/`*`condition_fil`*`.xlsx` workbook (Sec. 22.5). The `Usable Area Analysis ...` routine fills column `F` in the workbook, which automatically calculates column `G`: WUA per discharge. The *Total WUA* value in cell `J2` is the sum of column `G`.

22.9 Output and application in stream restoration projects

22.9.1 Rasters

The module creates HHSI rasters for the selected `condition` in the folder `RiverArchitect/Habitat Evaluation/HSI/`*`condition`*`/`, where depth HSI rasters are named `dsi_`*`fil`*`qqqqqq` and velocity HSI rasters are named `vsi_`*`fil`*`qqqqqq`. The `qqqqqq` string refers to the discharge that is derived from the name of flow depth rasters stored in `RiverArchitect/01_Conditions/`*`condition`*`/`. Please note, that the maximum discharge that can be handled is 999999 cfs or 999999 m^3/s because of the maximum length of raster file names.
CSI or cHSI rasters are created in `RiverArchitect/HabitatEvaluation/CHSI/`*`condition`*`/`.
Rasters with relevant information for usable habitat area are created in `RiverArchitect/HabitatEvalu ation/WUA/`*`habitat_condition`*`/`. The raster statistics correspond to the numbers written to column `F` in `RiverArchitect/HabitatEvaluation/WUA/`*`condition_fil`*`.xlsx`.

22.9.2 Workbooks for stream restoration

The `RiverArchitect/HabitatEvaluation/WUA/`*`condition_fil`*`.xlsx` workbook contains the key outputs of this module. The usable areas, related to certain discharges, in column `G` and their sum (WUA) in cell `J2` are important figures for comparing two situations (`conditions`).
For example, a relevant question can be ”What was the weighted usable habitat area for juvenile Chinook salmon in the year 2008 compared with 2014?” Comparing both the WUA in `2008_chj.xlsx` and the WUA in `2014_chj.xlsx` answers the question.
Another relevant question is ”How much did terraforming increase WUA?”. To answer this question, the habitat conditions of a (hydraulic) `condition` need to be evaluated based on 2D hydrodynamic model outputs for multiple discharges within the annual flow duration curve. Then, layer 1 features, as described in Sec. 2 and the *ModifyTerrain* module, need to be implemented into the DEM of the `condition`. The 2D hydrodynamic model needs to be re-run using the modified DEM and the same set of multiple annual flow duration discharges. Based on the sets of hydraulic rasters (flow depth and velocity), the *HabitatEvaluation* module can compute the WUA for both conditions and selected fish species, e.g., WUA of a 2014 DEM for juvenile Chinook salmon is originally calculated in `2014_chj` and the WUA of the modified (terraformed) 2014 DEM will be contained in `2014_lyr10_chj`. Comparing the `J2` cells of both workbooks reveals the net gain in WUA. When multiple restoration variants have to be compared, the net gain in WUA of all variants can be vetted against construction costs to obtain a price in terms of US $ per yd^2 (or m^2) gain in WUA.

22.10 Quit module and logfile

The best option to quit the module is the `Close` dropdown menu if no background processes are going on (see terminal messages), where also the processing `habitat_evaluation.log` logfile can be opened and reviewed for any error messages.

23 Working principles

23.1 Cover HSI: Substrate

A `dmean` raster is needed in the `RiverArchitect/01_Conditions/`*`condition`*`/` folder. If this box is checked, a `substrate_hsi` raster is created in `RiverArchitect/HabitatEvaluation/HSI/`*`condition`*`/`. The applied Habitat Suitability Curves can be adapted by clicking on the `Edit HSCs` button.

23.2 Cover HSI: Boulder

A `boulder` shapefile containing polygons with boulder sizes (diameters) needs to be available in `RiverArchitect/HabitatEvaluation/Cover/`*`condition`*`/`. The polygons need to be manually delineated for the entire region of interest. The module will convert boulder size information into a raster and retain boulders with a size larger than a threshold value. Areas, where the boulder presence covers more than 30 % of the surface get assigned an HSI value of 0.5. Both the 30 % surface ratio and the associated HSI of 0.5 can be changed for every fish species and lifestage (Sec. 22.2).

23.3 Cover HSI: Cobble

A `cobble` raster containing substrate sizes (cobble diameters) needs to be available in `RiverArchitect/HabitatEvaluation/HSI/`*`condition`*`/`. The module will evaluate the percentage of area that is covered with cobble larger than a threshold value (grain size). Areas, where the percentage area covers more than 30 % of the surface get assigned an HSI value of 0.3. Both the 30 % surface ratio and the associated HSI of 0.3 can be changed for every fish species and lifestage (Sec. 22.2).

23.4 Cover HSI: Streamwood

A `streamwood` shapefile containing polygons with single wood elements needs to be available in `RiverArchitect/HabitatEvaluation/HSI/`*`condition`*`/`. The module draws polygons with a user-defined radius around the streamwood polygons and assigns a value of `1` to these polygons. The new polygons are converted into a raster and an HSI value of 0.3 is assigned to `1` pixels. The user-defined radius and the associated HSI of 0.3 can be changed for every fish species and lifestage (Sec. 22.2).

23.5 Cover HSI: Vegetation

A `plantings` shapefile containing polygons with single plants needs to be available in `RiverArchitect/HabitatEvaluation/HSI/`*`condition`*`/`. The module draws polygons with a user-defined radius around the plant polygons and assigns a value of `1` to these polygons. The new polygons are converted into a raster and an HSI value of 0.3 is assigned to `1` pixels. The user-defined radius and the associated HSI of 0.3 can be changed for every fish species and lifestage (Sec. 22.2).

23.6 Cover HSI combination methods

The cover rasters are combined by selecting the maximum value of the superposition of applied cover types: covHSI = arcpy.sa.Float(arcpy.sa. CellStatistics (applied_covers , "MAXIMUM", "DATA")), where applied_covers is a list of arcpy.Raster() of applied cover types.

23.7 Usable habitat area calculation

The usable area is measured by converting cHSI rasters to polygon shapefiles using arcpy.RasterToPolygon_conversion (cHSI, polygon_shp, "NO_SIMPLIFY"). The area of single polygons is calculated by arcpy. CalculateAreas_stats (polygon_shp, self.cache + ... + "wua_eval.shp"). The polygon areas are summed up in a loop over the polygons (with arcpy.da.UpdateCursor(self.cache + ... + "wua_eval.shp", "F_AREA")as cursor: for row in cursor: area += float(row[0])). The area variable contains the usable habitat area for every discharge-related cHSI raster and it is eventually written in column `G` of *`condition_fil`*`.xlsx`.

24 Code modification: Changing the structure of Fish.xlsx

The `Fish.xlsx` workbook is read by the Fish class stored in `cFish.py`. The start rows for reading velocity, depth, substrate, mineral cover and vegetation cover habitat suitability curves from the workbook are hard coded in the self.parameter_rows dictionary of the Fish class. If rows were deleted or inserted, the self.parameter_rows need to be adapted.

- "u": row – row needs to correspond to the row number where the flow velocity related habitat suitability curve starts.
- "h": row – row needs to correspond to the row number where the flow depth related habitat suitability curve starts.
- "substrate": row – row needs to correspond to the row number where the substrate related (D) habitat suitability curve starts.
- "cov_min": row – row needs to correspond to the row number where the mineral cover (cobbles and boulders) related habitat suitability curve starts.
- "cov_veg": row – row needs to correspond to the row number where the vegetation cover (plants and wood) related habitat suitability curve starts.

The insertion or deletion of rows can be easily and robustly adapted by changing the above dictionary items.

However, changing or deleting columns is more complex because the module is coded in a manner that it can theoretically read an infinite number of fish species, but always limited to the same number of lifestages. Preferably omit non-relevant lifestages (do not put any number). Otherwise, change the code where read columns are relative incremental increases of numeric column values, starting at self.species_row = 2. For example, the "spawning" lifestage is at first place, and therefore, its relative column is 1. The "fry" lifestage is at second place but it needs to jump over an extra u column. Therefore, the relative column of "fry" is 3, the relative column of "juvenile" is 5 and the relative column of "adult" is 7. If another lifestage is used for any fish species, it needs to match one of the before mentioned stored in the self.ls_col_add dictionary of the Fish class. For example, the base scenario uses *Lamprey* fish and a "ammocoetes" lifestage instead of "fry". Therefore, the entry "ammocoetes": 3 needs to be added to self.ls_col_add.

Part VI
Project Maker

25 Introduction to the ProjectMaker module

The *ProjectMaker* module guides through the half-automated assessment of cost-relevant quantities and ecological project benefits. A "restoration plan" or project proposal for a restoration plan herein designates an isolated restoration measure in a river *REACH* at a selected site. *version*s of a restoration plan may refer to terraforming options or other planning *condition*s (year). A project proposal is prepared for (preliminarily) definite *version*s of a restoration plan including relevant soil bioengineering restoration features, i.e., vegetation plantings and stabilizing features including the placement of angular boulder and engineered log jams, and it evaluates cost-relevant quantities. A project cost table uses the cost-relevant quantities for a preliminary cost assessment. The habitat utility in terms of net gain in weighted usable habitat (WUA) for target fish species determines the project return in "US $ per [yd^2 or m^2] of newly created WUA".

This chapter explains the module application in the following sections:

Section 26:	Application of the GUI with descriptions of input requirements.
Section 27:	Generating a project plan and running a cost-quantity assessment with *Project Maker*.
Section 28:	Mapping final designs of features and the construction site.
Section 29:	Calculation of WUA and final results.

26 Quick GUIde to a project assessment

26.1 Prerequisites

Ensure that the following steps were executed in order to generate the required geodata for creating a project proposal:

- If terraforming applies:
  - The *REACH_SiteName* restoration terraforming plan was verified with 2D hydrodynamic modeling
  - The *River Architect* package's *ModifyTerrain* module was applied to calculate excavation / fill volumes (for usage, refer to part IV).
- The *LifespanDesign* and *MaxLifespan* modules were executed for plantings and bioengineering features. Thus, the following directories should exist and contain plantings and other bioengineering rasters:
  - Plantings:
    `.../RiverArchitect/MaxLifespan/Products/Rasters/`
    `20XX_REACH_lyr20_plants/`
  - Toolbox and Bioengineering:
    `.../RiverArchitect/MaxLifespan/Products/Rasters/`
    `20XX_REACH_lyr20_toolbox/`
- The *HabitatEvaluation* module was applied to the pre-project (initial) condition and the "with implementation" condition.

26.2 Main window set-up and run

The *ProjectMaker* GUI is shown in Fig. 11. The creation of a cost-benefit assessment requires the step-wise definition of variables and calculation beginning at the top of the GUI and moving forward to the bottom. The following sections provide details regarding input requirements and calculations of every step.

26.3 Input: Variables and automatically generated files

The assessment uses the following parameters and formats, which can be entered in the GUI:

- *version* (or vii) is a "*v*" + 2-digits (*ii*) version number (string), e.g., `v10`
- *condition* is a 4-digits year indicator (int), e.g., `2008`
- *REACH* is a 3-char reach indicator (string), e.g., `TBR`
- *SiteName* is a site name string written in CamelCase, e.g., `BigRavine`
- *stn* is a 3-char short name string of the site name, e.g., `rav`

Click on the `VALIDATE VARIABLES` button to verify that the variables entered are correct. A successful validation opens an info-box, a project assessment workbook, and a layout file that invites to create project specific files. The required actions include:

- WORKBOOK (*REACH_stn_costs_version*.xlsx)
  The workbook contains a spreadsheet named *costs*, where unit costs and quantities are evaluated. The *from_ geodata* spreadsheet will contain quantities such as area (in square meters or acres) of vegetation planting types. The numbers in the *from_geodata* tab are generated by a subset of codes that use geodata, which require manual actions as described in the next steps.
- LAYOUT FILE
  Save a copy by replacing the project parameters: *REACH_SiteName_vii.mxd* and proceed to the creation of required input geodata as described in the next sections.

26.4 Input: Project Area Polygon shapefile

To determine cost-relevant quantities for a site-related restoration plan, a manual delineation of the project site is necessary, e.g., by using the *REACH_SiteName_vii.mxd* layout file.

1) Create a new polygon-shapefile in ... `/Geodata/Shapefiles/` and name it *ProjectArea*.
2) Remove the newly created layer from *mxd* file's *Table of Contents*, double-click on the existing *Project area* layer → *Layer Properties* opens up → go to the *Source* tab → click on *Set Data Source...* → Select the newly created `/Shapefiles/`*ProjectArea.shp* file → click *OK*.
3) In the *mxd* file's *Table of Contents*, right-click on the *Project area* layer, then *Open Attribute Table*. In the *Table*, click on the top-left drop-down menu and *Add field ...*. Add a *Text*–type (*length* = 50) field named *AreaCode* and add another *short integer*–type (*precision* = 0) field named *gridcode*. Close the table.
4) Delineate project area
   a. Optional: Import modified terrain to visualize boundaries of terraforming
   b. In the *mxd* file's *Table of Contents*, right-click on the *Project area* layer, then *Edit Features* → *Start editing*.
   c. In the *Create Features* tab, click (highlight) on *ProjectArea*, then in *Construction Tools* field, click on *Polygon*.

d. Draw a polygon around the designated project area (finish with the *F2*-key).

e. Go to the *Attributes* tab and type *Restoration zone* (*text*) in the *AreaCode* field and *1* (*short integer*) in the *gridcode* field.

f. Save edits and stop editing.

26.5 Input: Delineate Plantings shapefile

The *MaxLifespan* module produces geofiles, i.e., rasters and shapefiles, for complete river reaches. In addition, terraforming may require clearing of existing vegetation in the project area. An overlay of the above created project area polygon over recent satellite image shows, where existing shrubs intersect with projected terraforming surfaces. A *PlantDelineation.shp* shapefile with polygons delineating these intersects needs to be created and drawn as follows in the .../`Geodata/`*`REACH_SiteName_vii.mxd`*layout file:

1) In the Catalog tab, open the folder tree .../`Geodata/Shapefile` (double click on the folder to make it appear in the lower box).

2) Right-click in the lower box, click on *New* → *Shapefile* and name is PlantDelineation (type: *Polygon*); ensure that the coordinate system is coherent with other layers of *REACH_SiteName_vii.mxd*.

3) Remove the newly created layer from *mxd* file's *Table of Contents*, double-click on the existing *Clearing of shrubs* layer → *Layer Properties* opens up → go to the *Source* tab → click on*Set Data Source...* → Select the newly created .../`Shapefiles/`*PlantDelineation.shp* file → click *OK*.

4) In the *mxd* file's *Table of Contents*, right-click on the layer *PlantDelineation*, then *Open Attribute Table*. In the *Table*, click on the top-left drop-down menu and *Add field...*. Add a *Text*–type (*length* = 50) field named *ActionType* and add another *short integer*–type (*precision* = 0) field named *gridcode*. Close the table.

5) Delineate existing plantings area:

   a. Ensure that a valid background image is linked to the *background* layer (*Layer Properties* → *Source* tab).

   b. In the *mxd* file's *Table of Contents*, right-click on the layer *PlantDelineation*, then *Edit Features* → *Start editing*

   c. In the *Create Features* tab, click on *ProjectDelineation*, then in *Construction Tools* window, click on *Polygon*.

   d. Draw polygons around existing plantings that are visible on the background (satellite image) project area, within the zone where the modified DEM rasters indicate terrain modification (finish polygon with the *F2*-key).
   When delineating existing plantings for clearing, remember that in stream restoration and habitat enhancement projects "clearing" should limit to the absolutely required minimum.

   e. Go to the *Attributes* tab and type *Clearing* (*text*) in the *ActionType* field and *1* (*short integer*) in the *gridcode* field.

   f. Once all visible plantings within the terraforming project area are delineated, save the edits and stop editing.

Save and close *REACH_SiteName_vii.mxd*.

27 Cost quantity assessment and the cost master workbook

The `REACH_stn_costs_version.xlsx` is subsequently referred to as the ***cost master workbook***. The workbook is automatically generated as a template-copy and it contains two `cost ...` tabs. **Important:** As a function of the unit system (U.S. Customary or SI metric), **only keep the relevant cost worksheet and delete the other one** (see Fig. 12). **Rename the retained costs tab to `costs`.**
The prices contained in the cost master workbook are in US $ and may be adapted to fit local construction costs. The following sections describe steps and requirements for the assessment of cost-relevant quantities with the cost master workbook.

27.1 Terraforming

The *ModifyTerrain* module evaluated terrain excavation and fill volumes. *ModifyTerrain* created spreadsheets featuring terraforming volumes in cubic meters / yards in the directory `.../RiverArchitect/ModifyTerrain/Output/Spreadsheets/condition_volumes.xlsx` (see also IV). Optionally, these workbooks can be copied to a *condition*_volumes.xlsx spreadsheet in the project folder.

Recall: *condition_volumes.xlsx* has to tabs: (1) *excavate_YYYYMMDD_HHhMM* and (2) *fill_YYYYMMDD_HHhMM*. Copy the terraforming volumes from either of these two spreadsheets to the cost master workbook's (*REACH_stn_costs_vii.xlsx*) *terraforming_volumes* spreadsheet (cells are highlighted, only values).

The template's unit costs of US $ 10.52 per cubic yard include transport and material storage. It is hypothesized that the smaller value, i.e., either the reach's *excavate* or the reach's *fill* volume, reduces the higher value's costs by half of the unit costs. This assumption is made because the smaller volume can be reused on-site, and therefore, material storage and transport costs reduce. The costs for terraforming works are evaluated in cell *G8* of the *cost* tab of *condition*_volumes.xlsx based on the excavate and fill volumes that need to be copied to the *terraforming_volumes* tab of *condition*_volumes.xlsx. The following formula applies (*vol* refers to the *terraforming_volumes* spreadsheet):

$$costs!G8 = \; costs!D8 \cdot min(vol!C5, \;\; vol!C6) \cdot \left[\frac{1}{2} + \left(\frac{max(vol!C5, \; vol!C6)}{min(vol!C5, \; vol!C6)} - 1\right)\right]$$

27.2 Vegetation plantings and supporting features

Before the most reasonable vegetation plantings are implemented into the project plan, the *MaxLifespan* module needs to be run based on anew 2D simulations made with the terraformed DEM. The resulting maximum lifespan rasters should be available in the directory `.../RiverArchitect/MaxLifespan/Products/Rasters/condition_reach_lyr20_plants/` and `.../RiverArchitect/MaxLifespan/Products/Rasters/condition_reach_lyr20_toolbox/`.

27.2.1 Delineation of most suitable plantings based on maximum lifespan maps

The GUI's *Delineate plantings* button launches a python function that picks up these maximum lifespan rasters, limits there extents to the *ProjectDelineation* Polygon and evaluates relevant quantities for construction purposes. When the calculation has successfully finished, the function's log file (*logfile_20.log*) automatically opens. Read the log file carefully and ensure that no error or warning messages occurred. If error messages occurred, check the geodata sources and error messages, ensure that the costs master file (*REACH_stn_costs_vii.xlsx*) is closed and trace back error messages. Re-run *Delineate Plantings* and trace back error messages until no error messages occur anymore.

After a successful run, *Delineate plantings* has written vegetation plantings areas to the cost master workbook's *from_geodata* spreadsheet. The *costs* spreadsheet automatically evaluates plantings in the *Vegetation plantings* frame. Nevertheless, double-check assigned cell links to the *from_geodata* spreadsheet and close the cost master workbook.

Delineate plantings saves the cropped maximum lifespan rasters and shapefiles with area summaries in the /Rasters/ and Shapefiles/ subfolders. If the cell links in the automatically opened cost master workbook's *costs* spreadsheet are correct, save and close the workbook.

27.2.2 Stabilize plantings with low expected lifespan

Even though the vegetation plantings maximum lifespan maps identify the optimum plantings types according to the highest lifespans, the projected vegetation plantings may be associated with low lifespans. Therefore, supporting (stabilizing) features such as engineered log jams (here: single anchored logs or root wads) may be required. The GUI's *Stabilize plantings* button launches a python function that adds stabilizing bioengineering features such as anchored wood logs to planting areas associated with the user defined "*Critical plantings lifespan*" variable. For example, if *Critical plantings lifespan* equals 2.5 years, all plantings that have an equal or smaller expected lifespan of 5 years get assigned the most suitable bioengineering feature. The *Stabilize plantings* function uses the following priorities of stabilizing features:

1) Large wood logs (diameters defined in `RiverArchitect/LifespanDesign/.templates/` *threshold_values.xlsx*) if their lifespan is higher than *Critical plantings lifespan*.

2) Engineered (anchored) wood logs, where maximum lifespan maps indicate convenient applicability.

3) Vegetation-based bioengineering features (pre-defined in cost master workbook: brush layers; alternatively, fascines or geotextile can be linked from *costs!F30:F33* to *from_geodata!C16**. . . , where the depth to the groundwater does not exceed the threshold values defined in `RiverArchitect/LifespanDesign/.templates/`*threshold_values.xlsx*.

4) Mineral-based bioengineering features (rock paving), where the depth to the ground water table is insufficient for vegetative stabilization and where the terrain is steeper than the threshold values defined in `RiverArchitect/LifespanDesign/.templates/`*threshold_values.xlsx*.

5) Angular boulders where high dimensionless bed shear stress predictions prohibit the utilization of any above feature.

As before, a log file (*logfile_21.log*) opens up at the end of the calculation for verification of the calculation process. *Stabilize plantings* writes construction-relevant numbers for vegetation planting stabilization to the cost master workbook's *from_geodata* spreadsheet. The *costs* spreadsheet automatically evaluates stabilizing feature quantities in the *Toolbox stabilization* and *Bioengineering (other)* frames. Nevertheless, check the assigned cell links to the *from_geodata* spreadsheet and adapt feature types if required. Moreover, *Stabilize plantings* creates a shapefile called (*Plant_stab.shp*) in `.../Geodata/Shapefiles/`. After checking the cell links in the automatically opened cost master workbook's *costs* spreadsheet, save and close the workbook.

27.3 Bioengineering features (other

Additional habitat can be created with cover features, i.e., engineering logs jams or root wads, at locations that result from an expert assessment. To implement cover features, open . . . /Geodata/*REACH_SiteName_vii.mxd* in order to do the following:

1) Create a new polygon-shapefile in .../Geodata/Shapefiles/ and name it *StreamWood*.

2) Remove the newly created *StreamWood* layer from *mxd* file's *Table of Contents*, double-click on the existing *ELJs (Cover habitat)* layer → *Layer Properties* opens up → go to the *Source* tab → click on *Set Data Source*. . . → Select the newly created .../Geodata/Shapefiles/*StreamWood.shp* file → click *OK*.

3) Start editing the *ELJs (Cover habitat)* layer.

4) Draw engineered log jams and root wads as 10 ft x 10 ft (3.5 m x 3.5 m) rectangles.
   Design hints:
   Engineered log jams and root wads must not be placed in side channels or anabranched sections of the rivers. However, these features can add "cover" habitat in backwater zones or reconnected ponds.
   A save premise is to keep a distance of at least 100 ft (or approximately 30 m) between individual log jams or root wads. To respect the distances, draw a circle with a diameter of 2·100 ft (or approximately 2·30 m) and place single engineered log jams in the middle of the circles.

5) Save the edits and stop editing.

6) Write the number of drawn streamwood to the cost master workbook's (*REACH_stn_costs_vii.xlsx*) *costs* spreadsheet (*Bioengineering* frame).

27.4 Other civil engineering works

Site access, terrain acquisition or culverts may be required and contribute to the project costs. Satellite images and GIS measurement tools help identifying the required length of new roads or roads that need to be developed.
The length of new roads can be evaluated, e.g., by drawing paths transferring the path length in yd' or m' (length yard or meter) to the cost master workbook's (*REACH_stn_costs_vii.xlsx*) *costs* spreadsheet (*Civil engineering & other* frame). For later revision, export the drawn paths to a newly created folder, e.g., .../Geodata/Shapefiles/ as **.kmz* file.
The resulting costs need to be manually entered in the costs master workbook's (*REACH_stn_costs_vii.xlsx*) *costs* spreadsheet (Civil engineering & other frame).

27.5 Other costs and remarks

The final project costs include site mobilization and demobilization as well as unexpected costs and engineering fees at the bottom of the cost master workbook's (*REACH_stn_costs_vii.xlsx*) *costs* spreadsheet. The total costs for the project proposal are summarized at the top of the *costs* spreadsheet.

28 Mapping of construction elements

Open `.../Geodata/`*`REACH_SiteName_vii.mxd`* and switch to *Layout View* (*ArcMap* → *View* → *Layout View*). Double click on every layer in the *Table of Contents* and define the correct source files (*Source* tab) that result from the above described cost assessment. Relevant shapefile are stored in `.../Geodata/Shapefiles/` and relevant rasters are stored in `.../Geodata/Rasters/`. Export the map (*ArcMap* → *File* → *Export Map...*) to the project folder and name it *REACH_SiteName_vii_lyr2x.pdf* (proposition for consistent file naming).

29 Ecological benefit asessment (Calculate WUA)

The project costs are vetted against the net gain in weighted usable habitat area for target fish species. The GUI's *Calculate Net **W**eighted **U**sable habitat **A**rea* routine calculates usable habitat from rasters that indicate where the Composite Habitat Suitability Index (CHSI) is higher than a selected threshold value.

29.1 Additional input and requirements

Every CHSI raster refers to a steady discharge within a flow duration curve. The expected flow exceedance duration per discharge bin multiplied with the usable habitat area is summed up to the WUA. The comparison of the existing (pre-project) and the "with implementation" (post-project) habitat suitability requires the following:

- Both situations (pre- and post-project) were simulated in the 2D hydrodynamic model.

- Flow duration curves for the project site were established (also refer to Sec. 21):
  - A workbook template for flow duration curves is available in `RiverArchitect/HabitatEvaluation/FlowDurationCurves/flow_duration_templates.xlsx`
  - The `Tools` – folder contains the *make_flow_duration.py* script that analyses discharge series (mean daily discharge) of any length for producing the required format for WUA calculation. An example file of mean daily discharges for creating a flow duration curve with *make_flow_duration.py* is provided.
- The *River Architect*'s *HabitatEvaluation* module was executed for both situations (pre- and post-project) to obtain CHSI maps.
- Example:
  - The pre-project terrain DEM dates from 2008 and terrain modifications were performed based on the 2008 DEM in a reach called `rea`.
  - Both DEMs, original and modified correspond to pre- and post-project conditions, respectively.
  - Both DEMs were simulated in the 2D hydrodynamic model with discharges of 100, 200, 500, 1000, 2000, and 5000 cfs (or m^3/s).
  - The corresponding modelling results (flow depth and velocity) were stored in the directories `RiverArchitect/01_Conditions/2008 and RiverArchitect/01_Conditions/2008_rea_lyr10`, respectively. The string `lyr10` refers to terraforming according to the code naming conventions.
  - The *River Architect*'s *HabitatEvaluation* module applied to both situations with a CHSI threshold value of, e.g., 0.4. This threshold value means that all pixels with a CHSI value lower than 0.4 were considered as being non-habitat and the *HabitatEvaluation* module excludes these pixels from the CHSI rasters. Thus, the *HabitatEvaluation* module produced CHSI rasters that are stored in:
    * `RiverArchitect/HabitatEvaluation/WUA/Rasters/2008` (existing / pre-project)
    * `RiverArchitect/HabitatEvaluation/WUA/Rasters/2008_rea_lyr10` (with implementation / post-project)
  - Te *HabitatEvaluation* module associated (relative) discharge duration and usable habitat areas with the rasters. For example, if the target fish species was Chinook salmon, juvenile lifestage (naming convention chj), the *HabitatEvaluation* module wrote the usable habitat area and discharge duration to the following workbooks:
    * `RiverArchitect/HabitatEvaluation/WUA/2008_chj.xlsx`
    * `RiverArchitect/HabitatEvaluation/WUA/2008_rea_lyr10_chj.xlsx`

29.2 Run WUA calculation

When the above requirements are fulfilled, the *Project Proposal* GUI can assess the difference in usable habitat area between both situations (pre- and post-project), i.e., the net gain in WUA. For starting the calculation, define the above-described input data and confirm the calculation:

- Select a fish species corresponding to the one analyzed with the *HabitatEvaluation* module, e.g., Chinook salmon, juvenile lifestage. The *Select fish* button turns green after selecting a target fish species + lifestage.
- Select an initial condition (pre-project) and confirm the selection (button turns green after selection).
- Select a condition after terraforming (with implementation / post-project) and confirm the selection (button turns green after selection.
- Click on the *Calculate Net gain in Weighted Usable habitat Area (WUA)* button to start the assessment.

The program will run in the background and prompts the calculation progress in the console window.

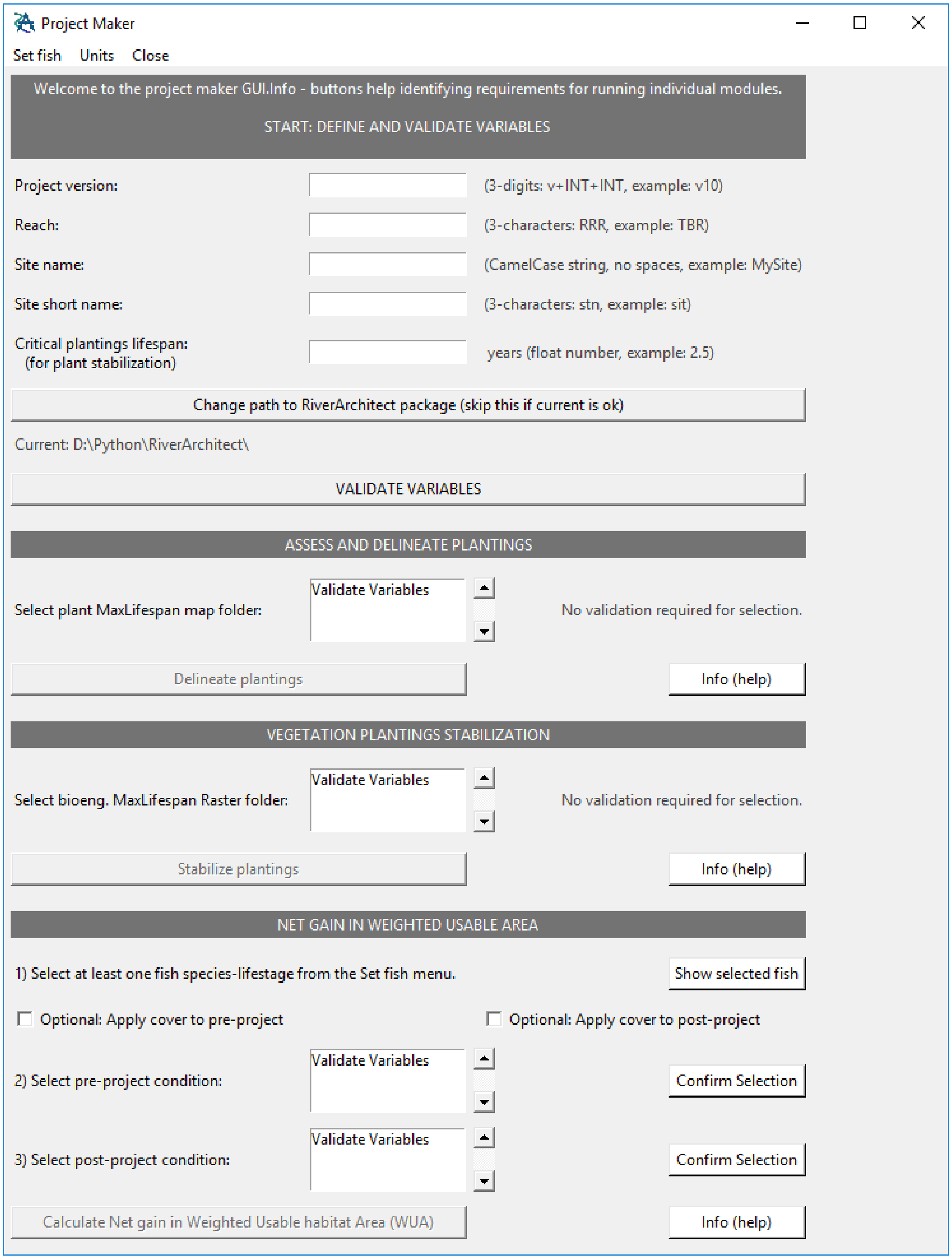
29.3 Output

After a successful run, a copy of the **cost master workbook** with the file name extension corresponding to the target fish automatically opens. For example, if the target fish was Chinook salmon – juvenile, the copy of the workbook is `.../REACH_stn_assessment_vii_chj.xlsx`.

Moreover, the particular **usable areas associated with the available discharges were written** to `/Geodata/WUA_evaluation_chj.xlsx`.

The discharge-related **shapefiles** with polygons of usable habitat area were saved as: `/Geodata/Rasters/condition/no_cover/NUMwua_eval.shp`. NUM is an automatically prefix added by the WUA evaluation routine. The association of the NUM shapefile with the corresponding discharge was logged to: `/Geodata/logfile_40.log`.

The cells *G3* and *I2/3* in `REACH_stn_assessment_vii_fish.xlsx`. state the net gain in WUA and the project return in units of US $ per square yard (or m^2 for any currency defined) net gain in WUA (comparison of pre- and post-project condition), respectively.



Welcome to the project maker GUI.Info - buttons help identifying requirements for running individual modules.

START: DEFINE AND VALIDATE VARIABLES

Project version:		(3-digits: v+INT+INT, example: v10)
Reach:		(3-characters: RRR, example: TBR)
Site name:		(CamelCase string, no spaces, example: MySite)
Site short name:		(3-characters: stn, example: sit)
Critical plantings lifespan: (for plant stabilization)		years (float number, example: 2.5)

Change path to RiverArchitect package (skip this if current is ok)

Current: D:\Python\RiverArchitect\

VALIDATE VARIABLES

ASSESS AND DELINEATE PLANTINGS

Select plant MaxLifespan map folder: Validate Variables — No validation required for selection.

Delineate plantings — Info (help)

VEGETATION PLANTINGS STABILIZATION

Select bioeng. MaxLifespan Raster folder: Validate Variables — No validation required for selection.

Stabilize plantings — Info (help)

NET GAIN IN WEIGHTED USABLE AREA

1) Select at least one fish species-lifestage from the Set fish menu. — Show selected fish

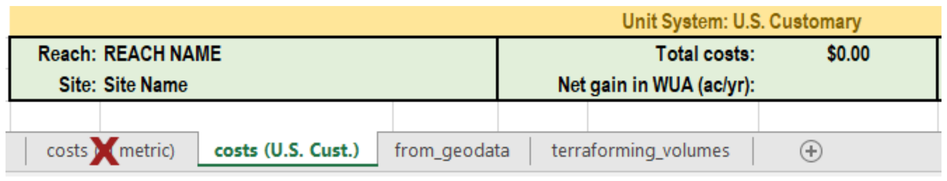
☐ Optional: Apply cover to pre-project ☐ Optional: Apply cover to post-project

2) Select pre-project condition: Validate Variables — Confirm Selection

3) Select post-project condition: Validate Variables — Confirm Selection

Calculate Net gain in Weighted Usable habitat Area (WUA) — Info (help)

Figure 11: GUI start up window.


Unit System: U.S. Customary

Reach: REACH NAME	Total costs: $0.00
Site: Site Name	Net gain in WUA (ac/yr):

Figure 12: Delete the non-applicable unit system tab and rename the tab to `costs`.

Part VII
Frequently Asked Questions (FAQ)

▷ **How can I change map styles?**
Map styles are controlled by settings made in `.mxd` layout files. Template layout files are stored in different locations for each module and pointers to rasters or shapefiles should be modified in these templates before mapping functions are executed.

- `LifespanDesign`:
  – Map layout templates are stored in `/RiverArchitect/LifespanDesign/Output/Mapping/.ReferenceLayouts/`.
  – Mapping functions use the file `legend.ServerStyle`, which is located in the `.ReferenceLayouts` folder. Contrary to `.style` files, the `.ServerStyle` file is required because `arcpy`-Python uses *ArcGIS Engine*, rather than *ArcGIS Desktop*. Own `.style` files can be created using *ArcMap*'s `Customize > Style Manager`. From the `Style Manager`, load the `LifespanDesign legend.style` file from the `.ReferenceLayouts` folder. Go to *LegendItems* and double-click on *LYR_lf_style*. The LifespanDesign module's mapping function accounts for font (size) changes made in the *Label Symbol* or *Description Symbol*. For more guidance on creating styles, click here. Next, save (or export) the `.style` file and convert it to a `.ServerStyle` file using `MakeServerStyleSet.exe`, which is typically located in `C:/Program Files (x86)/ArcGIS/Desktop10.x/bin/`. Note that `MakeServerStyleSet.exe` and the `.style` should to be located in the same folder. Finally, rename the new file to `legend.ServerStyle` and paste it in `/RiverArchitect/LifespanDesign/Output/Mapping/.ReferenceLayouts/`.
  – More descriptions in Sec. 8.6.
- `MaxLifespan`: see Sec. 14.3.2.
- `ModifyTerrain`: see Sec. 18.6.
- `HabitatEvaluation`: No mapping function implemented. For mapping *CHSI* rasters, create own `.mxd` layout files.
- `ProjectMaker`: see Sec. 28.

▷ **What is a *condition*?**
A *condition* refers to a planning state that is typically characterized by a 4-digits year indicator, followed by a layer specifier. *Condition*al Rasters are stored in `RiverArchitect /01_Conditions/`. For more information, refer to Sec. 5.

Part VIII

Error messages and Troubleshooting

30 Error and Warning messages

Most errors occur when the wrong python interpreter is used or when rasters or layouts have bad formats or when the information stated in the input file (see Sec. 11.1) is erroneous. The package writes process errors and descriptions to logfiles. When the GUI encounters problems, it directly provides causes and remedies in pop-up infoboxes. The common error and warning messages, which can be particularly raised by the package (alphabetical order) are listed in the following with detailed descriptions of causes and remedies. Most error messages are written to the logfiles, but some exception errors are only printed to the terminal because they occur before logging could even be started. Such non-logged `ExceptionErrors` are listed at the bottom of Sec. 30.1. Some non-identifiable errors raised by the `arcpy` package disappear after rebooting the system.

30.1 Error messages

▷ **`ERROR 000641: Too few records for analysis.`**

Cause This `arcpy` error message occurs here when arcpy. CalculateAreas_stats tries to compute the area of an empty shapefile.

Remedy If this error occurs within the calculation of WUA (weighted usable area) calculations, it may be ignored because some discharges do not provide any usable habitat area for a target fish species within a defined project area.
Otherwise, trace back files and check the shapefile consistency.

▷ **`ERROR 999998: Unexpected Error.`**
This is an operating system error and it can indicate different error conditions, i.e., the real reasons may have various error sources. Some of the most probable causes are:

Cause Usage of the wrong python interpreter

Remedy – Make sure to use the `ArcGIS`**`x64`**`XX.X` python interpreter (64 bit).
– Make sure that all input rasters are in *(Esri) Grid* format and well placed in the folder `LifespanDesign/Input/`*`condition`*`/`.
– Rebooting the system can help in some cases.

▷ **`ERROR: .cache folder in use.`**

Cause The content in the cache folder is blocked by another software and the output is probably affected.

Remedy Close the software that blocks `.cache`, including `explorer.exe`, other instances of `python` or `ArcGIS` and rerun the code. Also re-logging may be required if the folder cannot be unlocked.

▷ **`ERROR: .cache folder will be removed by package controls.`**

Cause arcpy could not clean up the `.cache` folder and the task is passed to *Python*'s os package. The content in the cache folder is blocked by another process and the output is probably affected.

Remedy Close the software that blocks `.cache`, including `explorer.exe`, other instances of *Python* or *ArcGIS* and rerun the code. Also re-logging may be required if the folder cannot be unlocked.

▷ **`ERROR: (arcpy) in PAR.`**

Cause Similar to ExceptionERROR: (arcpy) ... . The error is raised by the analysis_ ... , design_ ... and other functions when arcpy raster calculations could not be performed. Missing rasters, bad raster assignments, errors in input geodata files, or bad raster calculation expressions are possible reasons. The error can also occur when the `Spatial` license is not available.

Remedy See ExceptionERROR: (arcpy) ...

▷ **`ERROR: Analysis stopped ([...] failed).`**

Cause Raised by analysis (...) function in `LifespanDesign/feature_analysis.py` when it encountered an error.

Remedy Trace back the error message in brackets. If a results raster could not be saved, it means that the analyzed feature has no application, i.e., the results raster is empty, and therefore, it cannot be saved.

▷ **`ERROR: Area calculation failed.`**

Cause Raised by calculate_wua (self) of *HabitatEvaluation*'s CHSI() class in `HabitatEvaluation/cHSI.py` when it could not calculate the usable habitat area (see Sec. 23.7).

Remedy – Ensure that the WUA threshold has a meaningful value between 0.0 and 1.0 (Sec. 21).
– Ensure that neither the directory `HabitatEvaluation/.cache/` nor the directory `HabitatEvaluation/WUA/` or their contents are in use by other program.
– Review the input settings according to Sec. 22.
– Follow up earlier error messages.

▷ **`ERROR: Bad assignment of x/y values in coordinate input file.`**

Cause Raised by the coordinates_read (self) function of the Info () class in either `LifespanDesign/cReadInpLifespan.py` or `MaxLifespan/cReadActionInput.py` when `mapping.inp` has bad assignments of x-y coordinates.

Remedy Ensure that the coordinate definitions in `mapping.inp` (`LifespanDesign/.templates/` or `ActionPlanner/.templates/`) correspond to the definitions in Sec. 11.2.

▷ **`ERROR: Bad call of map centre coordinates. Creating squared-x layouts.`**

Cause Raised by get_map_extent (self , direction) function of the Info () class in either `LifespanDesign/cReadInpLifespan.py` or `MaxLifespan/cReadActionInput.py` when `mapping.inp` has bad assignments of x-y coordinates.

Remedy – *LifespanDesign*: Ensure that the file `mapping.inp` exists in the directory `LifespanDesign/.templates/` corresponding to the definitions in Sec. 11.2.
– *MaxLifespan*: Ensure that the file `mapping.inp` exists in the directory `MaxLifespan/.templates/` corresponding to the definitions in Sec. 11.2.
– General: Replace `mapping.inp` with the original file and re-apply modifications strictly following Sec. 11.2.

▷ **`ERROR: Bad mapping input file.`**

Cause Raised by either get_map_extent (self , direction), coordinates_read (self) or get_map_scale (self) function of the Info () class in either `LifespanDesign/cReadInpLifespan.py` or `MaxLifespan/cReadActionInput.py` when `mapping.inp` has wrong formats or it is missing.

Remedy See `ERROR: Bad call of map centre coordinates [...].`

▷ **`ERROR: Boundary shapefile in arcpy.PolygonToRaster[...].`**

Cause Raised the HabitatEvaluation module's make_boundary_ras(self , shapefile) function (`cHSI.py`) when it could not convert a provided shapefile defining calculation boundaries to a raster and load it as arcpy. Raster (.../ HabitatEvaluation /HSI/condition /bound_ras).

Remedy Verify that a the selected boundary shapefile (Sec. 22.4) has a valid rectangle and an `Id` field value of `1` for that rectangle.

▷ **`ERROR: Boundary shapefile provided but [...].`**

Cause Raised the HabitatEvaluation module's make_chsi(self , fish , boundary_shp) function (`cHSI.py`) when the "To Raster" conversion of the provided shapefile defining calculation boundaries failed.

Remedy See `ERROR: Boundary shapefile in arcpy.PolygonToRaster[...].`

▷ **`ERROR: Calculation of cell statistics failed.`**

Cause Raised by identify_best_features (self) of *MaxLifespan*'s ArcPyContainer() class in `MaxLifes span/cActionAssessment.py` when arcpy.sa. CellStatistics () could not be executed.

Remedy – The latest feature added to the internal best lifespan raster may contain inconsistent data. Manually load the last feature raster (the logfile tells the feature name) into *ArcMap* and trace back the error. If needed, re-run lifespan/design Raster Maker.
– In the case that the error occurs already with the first feature added, the *MaxLifespan*'s `zero` raster may be corrupted. The remedy described for the error message `ExceptionERROR: Unable to create ZERO Raster. Manual intervention required` can be used to manually re-create the `zero` raster.

▷ **`ERROR: Calculation of volume from RASTER failed.`**

Cause The volume_computation(self) function of the ModifyTerrain () class in `ModifyTerrain/cModifyTerrain.py` raises this error when the command arcpy.SurfaceVolume_3d(RASTER, "", "ABOVE", 0.0, 1.0) failed.

Remedy – Ensure that an *ArcGIS* `3D` extension license is available.
– Ensure that manually modified (Customary Feature) raster DEMs contain valid data.
– Ensure that the input directory of manually modified (Customary Feature) raster DEMs is correct (default: `ModifyTerrain/Input/DEM/`*`condition`*`/`).

▷ **`ERROR: Cannot find FEAT max. lifespan raster.`**

Cause The automated terrain modification with grading and/or widen features uses max. lifespan rasters (maps) to identify relevant areas. If the get_action_raster (self, feature_name) function of the ModifyTerrain () class in `ModifyTerrain/cModifyTerrain.py` cannot find max. lifespan rasters in the defined max. lifespan raster directory (default: `MaxLifespan/Output/Rasters/`*`condition`*`/`), it raises this error message.

Remedy Ensure that grading and/or widen max. lifespan rasters exist in the defined input folder (default `MaxLifespan/Output/Rasters/`*`condition`*`/`) and that the names of the rasters contain the feature shortname, i.e., `grade` and/or `widen`.

▷ **`ERROR: Cannot find flow depth raster.`**

Cause Raised by make_chsi(self, fish, boundary_shp) of the *HabitatEvaluation*'s CHSI() class in `HabitatEvaluation/cHSI.py` when it could associate a flow depth raster based on the name of a habitat suitability index (HSI) raster name.

Remedy – Ensure that the flow depth raster names in `RiverArchitect/01_Conditions/`*`condition`*`/` strictly comply with the naming conventions described in Sec. 5.
– Ensure that the HSI rasters are stored in `.../HabitatEvaluation/HSI/`*`condition`*`/`, with the correct raster names including information about the discharge (see Sec. 22.9.1).

▷ **`ERROR: Cannot find modified DEM. Ensure that file names contain 'dem'.`**

Cause The volume difference calculation and mapping of Custom CAD-modified DEM rasters failed because the get_cad_rasters_for_volume (self, feat_id) function of the ModifyTerrain() class in `ModifyTerrain/cModifyTerrain.py` cannot find the raster files.

Remedy Ensure that Custom CAD-modified DEM rasters exist in the defined input folder (default `ModifyTerrain/Input/DEM/`*`condition`*`/`) and that the names of the rasters contain the keyword `dem`, e.g., a valid raster name is `dem14_mod`, or feature shortname, i.e., `cust`.

▷ **`ERROR: Could not access Fish.xlsx (...).`**

Cause The get_hsi_curve (self, species, lifestage, par) function of the Fish () class (`HabitatEvaluation/cFish.py`) or the main() function in `s40_compare_wua.py` raise this error message when it cannot access `Fish.xlsx` or copy read values from the `/HabitatEvaluation/WUA/`*`condition`* directory.

Remedy Ensure that neither `HabitatEvaluation/.templates/Fish.xlsx` nor any file in `/HabitatEvaluation/WUA/`*`condition`* is used by another program.

▷ **`ERROR: Could not add cover HSI.`**

Cause The make_chsi(self, fish) function of the CHSI() class (`HabitatEvluation/cHSI.py`) raises this error message when it failed to add cover HSI rasters.

Remedy Manually verify cover HSI rasters in `HabitatEvaluation / HSI/` and recompile cover HSI rasters if needed (see Sec. 22.6).

▷ **`ERROR: Could not append PDF page XX to map assembly.`**

Cause The make_pdf_maps(self, *args) or map_custom(self, input_ras_dir , *args), map_reach(self , reach_id , feature_id , *args) functions of the Mapper class in `MaxLifespan/cMapper.py` or `ModifyTerrain/cMapModifiedTerrain.py` raise this error when they failed to map the current page (extent).

Remedy – *MaxLifespan*: Ensure that the definitions of `MaxLifespan/.templates/mapping.inp` are correct, analog to the descriptions of the *LifespanDesign* module in Sec. 11.2.
– *ModifyTerrain*: Also refer to error message `ERROR: Could not create PDF`.
– General: Ensure that no other program accesses the `MaxLifespan/.cache/`, `ModifyTerrain/.cache/` or `MaxLifespan/Output/`, `ModifyTerrain/Output/` directories or its contents.

▷ **`ERROR: Could not calculate CellStatistics (raster comparison).`**

Cause Raised by compare_raster_set (self , ...) function of the ArcPyAnalysis() class in `LifespanDesign/cLifespanDesignAnalysis.py` when the provided it failed to combine the lifespan according to the provided input rasters (hydraulic or scour fill or morphological units).

Remedy Manually open the input rasters and ensure that they comply with the requirements stated in Sec. 5.

▷ **`ERROR: Could not create PDF`**

Cause The map_custom(self, input_ras_dir , *args) function of the Mapper() class (`ModifyTerrain/cMapModifiedTerrain.py`) raises this error message when it arcpy.mapping.ExportToPDF(self.mxd, self.output_map_dir + map_name, image_compression="ADAPTIVE", resolution=96) failed.

Remedy Ensure consistent layout template definitions according to Sec. 18.6.

▷ **`ERROR: Could not create Raster of the project area.`**

Cause Raised by set_project_area (self) of *ProjectMakers*'s CWUA() class in `ProjectMaker/cWUA.py` when it failed to convert the project area shapefile to a raster, which it needs for limiting spatial calculations to the project extent.

Remedy Ensure that the project was correctly delineated (Sec. 26.4).

▷ **`ERROR: Could not crop raster to defined flow depth.`**

Cause The crop_input_raster (self , fish_species , fish_lifestage , depth_raster_path) function of the CovHSI (HHSI) class (`HybitatEvluation/cHSI.py`) raises this error message when it failed cropping the raster with the spatial analyst operation Con((Float(h_raster) >= h_min), cover_type_raster).

Remedy Ensure that the provided flow depth file (selected in the GUI) contains valid data and that `Fish.xlsx` contains a minimum flow depth value for the selected fish species and lifestage.

▷ **`ERROR: Could not export PDF page no. XX`**

Cause The make_pdf_maps(self, *args) function of the Mapper class in `MaxLifespan/cMapper.py` raises this error when `MaxLifespan/.templates/mapping.inp` contains invalid xy-coordinates (format).

Remedy Ensure the definitions of `MaxLifespan/.templates/mapping.inp` analog to the descriptions of the *LifespanDesign* module in Sec. 11.2.

▷ **`ERROR: Could not find max. lifespan Rasters.`**

Cause Error raised by the main() function in `ProjectMaker/s20_plantings_delineation.py`) when the defined directory of max. lifespan rasters contains invalid or corrupted raster data.

Remedy – Ensure the correct usage of variables and input definitions (Sec. 26).
– Ensure that max. lifespan Rasters were generated without errors; if necessary, visually control the consistency of max. lifespan rasters in `.../MaxLifespan/Products/Rasters/`*`condition_reach`*`_lyr20_plants/` and `.../MaxLifespan/Products/Rasters/`*`condition_reach`*`_lyr20_plants/` or `...bioengineering`(cf. Sec. 27.2.1).

▷ **`ERROR: Could not find any worksheet.`**

Cause Error raised by the open_wb(self) function of the Read() class in `ProjectMaker/cIO.py`) when the concerned workbook contains errors.

Remedy – Ensure the correct usage of `HabitatEvaluation/.templates/Fish.xlsx` (Sec. 22.2).
– Ensure the correct adaptation of `ProjectMaker/.../REACH_stn_assessment_vii.xlsx` (Sec. 27).

▷ **`ERROR: Could not find sheet.`**

Cause Error raised by the open_wb(self) function of the Read() class in `HabitatEvaluation/cHabitatIO.py`) when the template workbook contains errors.

Remedy Ensure the correct usage of `HabitatEvaluation/.templates/Fish.xlsx` (Sec. 22.2) and the completeness of `HabitatEvaluation/.templates/Q_def_hab_template_si.xlsx` and `HabitatEvaluation/.templates/Q_def_hab_template_us.xlsx`. If either template workbook is corrupted or does not exist, re-install missing files.

▷ **`ERROR: Could not find sheet ``extents'' in computation_extents.xlsx.`**

Cause Error raised by the get_reach_coordinates (self , internal_reach_id) function of the Read() class in `.site_packages/riverpy/cTerrainIO.py`) when the `extents` sheet in the reach coordinate spreadsheet (`ModifyTerrain/.templates/computation_extents.xlsx`) could not be read.

Remedy Ensure the correct setup of `ModifyTerrain/.templates/computation_extents.xlsx` (Sec. 18.3).

▷ **`ERROR: Could not find the cover input geofile [...]`**

Cause Error raised by the __init__ (self , ...) function of the CovHSI(HHSI) class in `HabitatEvaluation/cHSI.py`) when the input cover geofile could not be read or is missing.

Remedy Ensure that a geofile (raster or shapefile) exists in the specified *`condition`* folder for the specified cover type (checkbox activated in the GUI). The `Help` button in the GUI provides more information on required geofiles and Sec. 23.

▷ **`ERROR: Could not interpolate exceedance probability of Q = [...]`**

Cause Raised by interpolate_flow_exceedance (self , Q_value) of *HabitatEvaluation*'s FlowAssessment() class in `HabitatEvaluation/cHSI.py` when the flow duration curve contains invalid data.

Remedy Ensure the correct setup of the used flow duration curve in `HabitatEvaluation/FlowDurationCurves/`. The file structure must correspond to that of the provided template `flow_duration_template.xlsx` and all discharge values need to be positive floats. Review Sec. 22.5 for details.

▷ **`ERROR: Could not open workbook.`**

Cause Error raised by the `__init__(self)` function of the `Read()` class in `ProjectMaker/cIO.py`) when the concerned workbook contains errors.

Remedy Ensure the correct usage of the concerned workbook (Part VI.

▷ **`ERROR: Could not load newly created Raster of the project area.`**

Cause Raised by `set_project_area(self)` of *ProjectMakers*'s `CWUA()` class in `ProjectMaker/cWUA.py` when the converted the project area shapefile is corrupted.

Remedy Ensure that the project was correctly delineated (Sec. 26.4).

▷ **`ERROR: Could not perform spatial radius operations [...].`**

Cause The `spatial_join_analysis(self, rater, curve_data)` function of the `CovHSI(HHSI)` class (`HabitatEvaluation/cHSI.py`) raises this error message when one or several spatial calculations failed, including `arcpy.RasterToPoint_conversion [...]`, `arcpy.SpatialJoin_analysis [...]` and / or `arcpy.PointToRaster_conversion [...]`.

Remedy Ensure that the cover input files and habitat suitability (curve) parameters are properly defined according to Sec. 22.6.

▷ **`ERROR: Could not process information from [...].`**

Cause The `main()` function in `ProjectMaker/s40_compare_wua.py` raises this error message when it could not calculate the weighted usable habitat area for condition or (set of) discharge(s).

Remedy Ensure that the variable (parameters) are properly defined according to Sec. 26 and that the *HabitatEvaluation* module contains the required information.

▷ **`ERROR: Could not read parameter type [...] from Fish.xlsx.`**

Cause The `get_hsi_curve(self, species, lifestage, par)` function of the `Fish()` class (`HabitatEvaluation/cFish.py`) raises this error message when it cannot read a habtiat suitability curve from `Fish.xlsx`.

Remedy – Ensure that`HabitatEvaluation/.templates/Fish.xlsx` is not opened in any other program.
– Ensure that a habitat suitability curve is defined in Fish.xlsx for the considered hydraulic or cover parameter according to Sec. 22.2.

▷ **`ERROR: Could not retrieve reach coordinates.`**

Cause The automated terrain modification with grading and/or widen features in the `modification_manager(self, feat_id)` function of the `ModifyTerrain()` class in `ModifyTerrain/cModifyTerrain.py` raises this error when the reach extents defined in `ModifyTerrain/.templates/computation_extents.xlsx` are not readable. In particular, the command `self.reader.get_reach_coordinates(self.reaches.dict_id_int_id[self.current_reach_id])` caused the error.

Remedy – Follow the instructions in Sec. 18.3 for correct reach definitions.
– If the *ModifyTerrain* module is externally loaded, ensure the correct definition of features and feature shortnames (see Sec. 18.8).

▷ **`ERROR: Could not run WUA analysis.`**

Cause The main() function in `ProjectMaker/s40_compare_wua.py` raises this error message when it could not calculate WUA.

Remedy Trace back warning and other error messages. Ensure the correct definition of parameters, creation of required geodata, and file naming (Part VI)

▷ **`ERROR: Could not save best lifespan raster.`**

Cause Raised by identify_best_features (self) of *MaxLifespan*'s ArcPyContainer() class in `MaxLifespan/cActionAssessment.py` when the calculated internal best lifespan raster is corrupted.

Remedy – Check prior WARNING and ERROR messages.
– Ensure that neither the directory `MaxLifespan/.cache/` nor the directory `MaxLifespan/Output/` or their contents are in use by other programs.

▷ **`ERROR: Could not save CSI raster associated with ...`**

Cause Raised by make_chsi_hydraulic (self , fish) of *HabitatEvaluation*'s CHSI() class in `HabitatEvaluation/cHSI.py` when the calculated cHSI raster is empty or corrupted.

Remedy – Ensure that neither the directory `HabitatEvaluation/.cache/` nor the directory `HabitatEvaluation/WUA/` or their contents are used by another program.
– Review the input settings according to Sec. 22.

▷ **`ERROR: Could not save cover / H HSI [...] raster ...`**

Cause Raised by make_hhsi(self , fish_applied) of *HabitatEvaluation*'s HHSI() class in `HabitatEvaluation/cHSI.py` when the calculated HHSI raster is empty or corrupted.

Remedy – Ensure that no other software uses data from neither the `HabitatEvaluation/` nor the `StreamRestoration/01_Conditions/` directories.
– Review the input flow velocity and depth rasters according to Sec. 5.

▷ **`ERROR: Could not save WORKBOOK.`**

Cause The main() function in `ProjectMaker/s40_compare_wua.py` raises this error message when it could not save `WUA_evaluation_`*`unit`*`.xlsx`.

Remedy Ensure that the workbook exists, has valid contents, and is not opened by another program.

▷ **`ERROR: Could not save WUA-CHSI raster.`**

Cause Raised by calculate_wua (self) of *HabitatEvaluation*'s CHSI() class in `HabitatEvaluation/cHSI.py` when the calculated cHSI raster is empty or corrupted.

Remedy – Ensure that the WUA threshold has a meaningful value between 0.0 and 1.0 (Sec. 21).
– Ensure that neither the directory `HabitatEvaluation/.cache/` nor the directory `HabitatEvaluation/WUA/` or their contents are in use by other programs.
– Review the input settings according to Sec. 22.

▷ **`ERROR: Could not load existing Raster of the project area.`**

Cause Raised by set_project_area (self) of *ProjectMakers*'s CWUA() class in `ProjectMaker/cWUA.py` when it found a raster that delineates the project area, but this raster is corrupted. The function requires the shapefile to raster conversion to limit applicable rasters to the project extent range, which is done with raster calculator operations.

Remedy – Ensure that the project was correctly delineated (Sec. 26.4).
– Manually inspect the project delineation raster.

▷ **`ERROR: Could not transfer net WUA gain.`**

Cause The main() function in `ProjectMaker/s40_compare_wua.py` raises this error message when it could not copy the calculated WUA from `WUA_evaluatio _unit.xlsx` to *`REACH_stn_costs_vii.xlsx`*.

Remedy Open `WUA_evaluation_`*`template_unit`*`.xlsx` and verify the calculated values. Trace back potential error sources in the CHSI rasters `/HabitatEvaluation/` folder and other error messages.

▷ **`ERROR: Could not transfer WUA data for [FISH].`**

Cause The main() function in `ProjectMaker/s40_compare_wua.py` raises this error message when it could not retrieve WUA data from the `/HabitatEvaluation/WUA/` module to `WUA_evaluation_`*`unit`*`.xlsx`.

Remedy Open `WUA_evaluation_`*`template_unit`*`.xlsx` and verify the calculated values. Trace back potential error sources in the CHSI rasters `/HabitatEvaluation/` folder and other error messages.

▷ **`ERROR: Could not write value to CELL [...]`**

Cause Error raised by the write_data_cekk (self , column, row, value) function of the Write() class in `HabitatEvaluation/cHabitatIO.py`) when it cannot write a value to `RiverArchitect/HabitatEvaluation/WUA/`*`condition_fill`*`.xlsx`.

Remedy Close all applications that may use `RiverArchitect/HabitatEvaluation/WUA/`*`condit ion_fill`*`.xlsx`. Detailed information on *HabitatEvaluation* workbook outputs are available in Sec. 22.5.

▷ **`ERROR: Could not write WUA data for [FISH].`**

Cause The main() function in `ProjectMaker/s40_compare_wua.py` raises this error message when it could not write the calculated WUA to when it cannot write a value to `WUA_evaluation_`*`template_`*`unit.xlsx`.

Remedy Ensure that the workbook is not opened by another program and / or visually verify that the concerned `CHSI` rasters contain valid values.

▷ **`ERROR: Cover raster calculation (check input data).`**

Cause Raised by call_analysis (self, curve_data) of *HabitatEvaluation*'s CovHSI(HHSI) class in `HabitatEvaluation/cHSI.py` when the cover HSI raster calculation failed.

Remedy Ensure that the input geofiles (raster or shapefile) are correctly set up according to Sec. 22.6 ff.

▷ **`ERROR: Extent is not FLOAT. Substituting to extent = 7000.00.`**

Cause Raised by the save_design (self, name) or save_lifespan (self, name) functions of the ArcPyAnalysis class in either `LifespanDesign/cLifespanDesignAnalysis.py` when the output folder for rasters (the folder directory is stated in the logfile) contains rasters of the same name which cannot be deleted.

Remedy Ensure that no other program uses the raster output folder and consider moving existing files in that folder to `LifespanDesign/Products/Rasters/`*`condition`*.

▷ **`ERROR: Existing files are locked. Consider deleting [...] file structure.`**

Cause Raised by the get_map_extent (self, direction) function of the Info() class in either `LifespanDesign/cReadInpLifespan.py` or `MaxLifespan/cReadActionInput.py` when `mapping.inp` has bad assignments of x-y coordinates (not a number).

Remedy See `ERROR: Bad call of map centre coordinates ...`

▷ **`ERROR: Failed calling`*`PAR`*`analysis of`*`FEATURE`*`.`**

Cause Special case of **`ERROR: Function analysis`**, which may occur after code modifications.

Remedy – Make sure that the self.parameter_list s of features (Sec. 12.5) has valid entries that also occur in analysis_call (*args) (`LifespanDesign/feature_analysis.py`).
– Make sure that valid function names exist in `LifespanDesign/cLifespanDesignAnalysis.py` (Sec. 12.4).

▷ **`ERROR: Failed to access computation_extents.xlsx.`**

Cause Error raised by the get_reach_coordinates (self, internal_rach_id) function of the Read() class in `ModifyTerrain/cReadTerrainIO.py`) when the reach coordinate spreadsheet (`ModifyTerrain/.templates/computation_extents.xlsx`) could not be read.

Remedy Ensure correct setup of `ModifyTerrain/.templates/computation_extents.xlsx` (Sec. 18.3).

▷ **`ERROR: Failed to access /load Fish.xlsx / Q_def_hab ...`**

Cause Error raised by the open_wb(self) and make_condition_xlsx (self, fish_sn) functions of the Read() class in `HabitatEvaluation/cHabitatIO.py`) when the template workbook contains errors.

Remedy Ensure the correct usage of `HabitatEvaluation/.templates/Fish.xlsx` (Sec. 22.2) and the completeness of `HabitatEvaluation/.templates/Q_def_hab_template_si.xlsx` and `HabitatEvaluation/.templates/Q_def_hab_template_us.xlsx`. If either template workbook is corrupted or does not exist, re-install missing files.

▷ **`ERROR: Failed to access WORKBOOK.`**

Cause Error raised by the write_volumes(self , ...) function of the Writer () class in `.site_packages/riverpy/cTerrainIO.py`) or the __init__ (..) function of *ProjectMakers*'s Read() class in `ProjectMaker/cIO.py` when the WORKBOOK is inaccessible or locked by another program.

Remedy Ensure that the concerned workbook exists and no other program uses the workbook.

▷ **`ERROR: Failed to add raster.`**

Cause Raised by read_hyd_rasters (self) of *HabitatEvaluation*'s HHSI() class in `HabitatEvaluation/cHSI.py` when is could not find hydraulic input rasters.

Remedy – Ensure that no other software uses data from neither the `HabitatEvaluation/` nor the `StreamRestoration/01_Conditions/` directories.
– Review the input flow velocity and depth rasters according to Sec. 5.

▷ **`ERROR: Failed to create WORKBOOK.`**

Cause Error raised by the write_volumes(self , ...) function of the Writer () class in `.site_packages/riverpy/cTerrainIO.py`) when the `template` it could not add new sheets in `ModifyTerrain/Output/Spreadsheets/`*`condition`*`_volumes.xlsx` or write to copies of `ModifyTerrain/Output/Spreadsheets/volume_template.xlsx`.

Remedy Trace back earlier error messages, ensure that no other program locked `ModifyTerrain/Output/Spreadsheets/`*`condition`*`_volumes.xlsx` and ensure that `ModifyTerrain/Output/Spreadsheets/volume_template.xlsx` was not deleted.

▷ **`ERROR: Failed to open Fish.xlsx. Ensure that the workbook is not open.`**

Cause Raised by the edit_xlsx (self) function of the Fish () class in `HabitatEvaluation/cFish.py` when
`HabitatEvaluation/.templates/Fish.xlsx` is opened by another program or non-existent.

Remedy Ensure that the file `HabitatEvaluation/.templates/Fish.xlsx` exists and close any software that may use the workbook.

▷ **`ERROR: Failed to read coordinates from computation_extents.xlsx (return 0).`**

Cause Error raised by the get_reach_coordinates (self , internal_rach_id) function of the Read() class in `.site_packages/riverpy/cTerrainIO.py`) when the reach coordinate spreadsheet (`ModifyTerrain/.templates/computation_extents.xlsx`) contains invalid data.

Remedy Ensure correct setup of `ModifyTerrain/.templates/computation_extents.xlsx` (Sec. 18.3).

▷ **`ERROR: Failed to read maximum depth to water value for [...].`**

Cause Error raised by the lower_dem_for_plants function of the ModifyTerrain class in `ModifyTerrain/cModifyTerrain.py`) when the threshold workbook (`LifespanDesign/.templates/threshold_values.xlsx`) is not accessible or does not contain values for *Depth to groundwater (min) / max* contains invalid data.

Remedy Ensure the correct setup of `LifespanDesign/.templates/threshold_values.xlsx` (Sec. 8.3). Note that ModifyTerrain starts reading depth to ground water values column by column, until it meets a non-numeric value.

▷ **`ERROR: Failed to save PDF map assembly.`**

Cause The make_pdf_maps(self, *args) function of the Mapper class in `MaxLifespan/cMapper.py` or `ModifyTerrain/cMapper.py` raises this error when the map assembly is corrupted.

Remedy Ensure that no other program accesses the `MaxLifespan/.cache/`, `ModifyTerrain/.cache/` or `MaxLifespan/Output/`, `ModifyTerrain/Output/` directories or their contents.

▷ **`ERROR: Failed to save WORKBOOK.`**

Cause Raised by calculate_wua (self) of *HabitatEvaluation*'s CHSI() class in `HabitatEvaluation/cHSI.py` when it could not save `*condition_fill*.xlsx`.

Remedy Ensure that no other software uses `HabitatEvaluation/WUA/*condition_fill*.xlsx`.

▷ **`ERROR: Failed to set reach extents -- output is corrupted.`**

Cause The automated terrain modification with grading and/or widen features in the lower_dem_for_plants (self , feat_id , extents) function of the ModifyTerrain () class in `ModifyTerrain/cModifyTerrain.py` raises this error when the reach extents defined in `ModifyTerrain/.templates/computation_extents.xlsx` are not readable.

Remedy Follow the instructions in Sec. 18.3 for correct reach definitions.

▷ **`ERROR: Feature identification failed.  Using default layout.`**

Cause Raised by choose_ref_layout (self , feature_type) of *MaxLifespan*'s Mapper class in `MaxLifespan/cMapActions.py` when there no layout could be assigned to the feature_type argument. The feature_type argument is not either " terraforming ", " plantings ", " bioengineering ", or "maintenance".

Remedy – If code was modified: Ensure that the new feature set can be recognized by the choose_ref_layout (self , feature_type) function. If needed, expand the if statement by the new feature set.
– Check consistency of suspected lifespan/design rasters, the correctness of lifespan/design input directory definitions (Sec. 14) and if needed re-run lifespan/design Raster Maker.

▷ **`ERROR: FEAT SHORTNAME contains non-valid data or is empty.`**

Cause Raised by get_design_data (self) in `MaxLifespan/cActionAssessment.py` when the feature shortname raster is empty or the shortname itself does not match the code conventions.

Remedy – If code was modified: Review code modifications and ensure to define feature shortnames as listed in Sec. 4. If a new feature was added, it also needs to be appended in the container lists (self . id_list , self . threshold_cols , self . name_list) of the Feature () class in `.site_packages/riverpy/cDefinitions.py`. A new feature also requires modifications of the `RiverArchitect/LifespanDesign/.templates/threshold_values.xlsx` spreadsheet (Sec. 8.3), in line with the column state in the self . threshold_cols list of the Feature () class.
– Check consistency of suspected lifespan/design rasters, the correctness of lifespan/design input directory definitions (Sec. 14) and if needed re-run lifespan/design Raster Maker.

▷ **`ERROR: Function analysis_call received bad arguments.`**

Cause The analysis_call (∗args) method in `LifespanDesign/feature_analysis.py` causes this error when it is not able to assign an analysis function based on the provided parameter_name. It may come along with `ERROR: .cache folder in use.` or after changes have been effected in the code.

Remedy If the `.cache` folder is in use, delete it manually (works sometimes only after logging of and on). If the error occurs after code modifications, make sure that the self . parameter_list s of features (Sec. 12.5) has valid entries that occur in analysis_call (∗args) (`LifespanDesign/feature_analysis.py`) and that valid function names exist in `LifespanDesign/cLifespanDesignAnalysis.py` (Sec. 12.4).

▷ **`ERROR: Incoherent data in RAS (raster comparison).`**

Cause Raised by compare_raster_set (self , ...) function of the ArcPyAnalysis() class in `LifespanDesign/cLifespanDesignAnalysis.py` when the provided the input `RAS` raster (hydraulic or scour fill or morphological units) are invalid.

Remedy Manually open the concerned `RAS` raster and ensure that it complies with the requirements for input rasters stated in Sec. 5.

▷ **`ERROR: Input file not available.`**

Cause Raised by get_line_entries (self , line_no) function of the Info () class in `LifespanDesign/cReadInpLifespan.py` when it cannot access input files.

Remedy – Ensure that the file `LifespanDesign/.templates/input_definitions.inp` exists in the directory `LifespanDesign/.templates/` corresponding to the definitions in Sec. 11.1.
– Ensure that the file `mapping.inp` exists in the directory `LifespanDesign/.templates/` corresponding to the definitions in Sec. 11.2.
– In case of doubts: Replace `LifespanDesign/.templates/input_definitions.inp` and `mapping.inp` with the original files and re-apply modifications strictly following Sec. 11.

▷ **`ERROR: Insufficient data. Check raster consistency and add more flows(?).`**

Cause The compare_raster_set (self , raster_set , threshold) function in `LifespanDesign/cLifespanDesignAnalysis.py` raises this error when insufficient hydraulic rasters are provided or when the provided hydraulic rasters have inconsistent data.

Remedy – Make sure to provide at least two pairs of hydraulic (`u` and `h`) rasters that correspond to two different discharges (one `u` and one `h` raster per discharge).
– As a rule of thumb: the more hydraulic rasters provided, the better are the lifespan maps. However, for reasons of consistency, the maximum number of hydraulic rasters in six per `u` and one `h`, i.e., six lifespans.
– Verify raster and corresponding lifespan definitions in `LifespanDesign/.templates/input_definitions.inp` (Sec. 11.1).

▷ **`ERROR: Invalid cell assignment for discharge / rasters.`**

Cause Error raised by the make_condition_xlsx (self , fish_sn) function of the Write () class in `HabitatEvaluation/cHabitatIO.py`) when it cannot write discharge values to `RiverArchitect/HabitatEvaluation/WUA/`*`condition_fill`*`.xlsx`.

Remedy Ensure that the flow duration curve is well defined (see Sec. 22.5) and that `RiverArchitect/HabitatEvaluation/WUA/`*`condition_fill`*`.xlsx` is not used by any other application.

▷ **`ERROR: Invalid feature names for column headers.`**

Cause Error raised by the write_volumes(self , ...) function of the Writer() class in `.site_packages/riverpy/cTerrainIO.py`, when the `template` sheet in the output (template) workbook (`ModifyTerrain/Output/Spreadsheets/`*`condition`*`_volumes.xlsx` or `...volume_template.xlsx`) has inconsistent feature (short-) names.

Remedy Ensure that `ModifyTerrain/Output/Spreadsheets/`*`condition`*`_volumes.xlsx` or `...volume_template.xlsx` contain consistent header names (Sec. 18.9.3) corresponding to the definitions in Sec. 4.

▷ **`ERROR: Invalid feature ID.`**

Cause Error raised by the __init__ (self , ...) function of the ThresholdDirector () class in `/LifespanDesign/cThresholdDirector.py`, when the feature IDs (shortnames) in `/LifespanDesign/.templates/threshold_values.xlsx` are incorrectly defined.

Remedy – Ensure correct definitions in `/LifespanDesign/.templates/threshold_values.xlsx` (Sec. 8.3).
– Consider replacing corrupted threshold workbooks with the original file.

▷ **`ERROR: Invalid file name or data.`**

Cause Error raised by the save_close_wb(self , *args) function of the Write() class in `HabitatEvaluation/cHabitatIO.py`) or `ProjectMaker/cIO.py`) when it cannot save `RiverArchitect/HabitatEvaluation/WUA/`*`condition_fill`*`.xlsx` or a copy of the cost master workbook.

Remedy – *HabitatEvaluation*: Close all applications that may use *`condition_fill`*`.xlsx` and ensure that its template exists. Detailed information on *HabitatEvaluation* workbook outputs are available in Sec. 22.5.
– *ProjectMaker*: Close all applications that may use the cost master workbook (*`REACH_stn_costs_version`*`.xlsx`) and ensure that it exists. Detailed information are available in Sec. 27.

▷ **`ERROR: Invalid interpolation data type (type(Q flowdur) == ...)`**

Cause Raised by interpolate_flow_exceedance (self , Q_value) of *HabitatEvaluation*'s FlowAssessment() class in `HabitatEvaluation/cHSI.py` when the flow duration curve contains invalid data.

Remedy Ensure the correct setup of the used flow duration curve in `HabitatEvaluation/FlowDurationCurves/`. The file structure must correspond to that of the provided template `flow_duration_template.xlsx`. Review Sec. 22.5 for details.

▷ **`ERROR: Invalid x-y coordinates in mapping.inp`**

Cause The make_pdf_maps(self, *args) function of the Mapper class in `MaxLifespan/cMapActions.py` raises this error when `MaxLifespan/.templates/mapping.inp` contains invalid map definitions (extents).

Remedy Ensure the definitions of `MaxLifespan/.templates/mapping.inp` analog to the descriptions of the *LifespanDesign* module in Sec. 11.2.

▷ **`ERROR: Invalid x-y coordinates in reach spreadsheet.`**

Cause The map_custom(self, input_ras_dir , *args), map_reach(self , reach_id , feature_id , *args) functions of the Mapper class in `ModifyTerrain/cMapModifiedTerrain.py` raises this error when the reach definition spreadsheet (`ModifyTerrain/.templates/computation_extents.xlsx`) contains invalid coordinates.

Remedy Ensure the definitions in `ModifyTerrain/.templates/computation_extents.xlsx` correspond to the descriptions in Sec. 18.3, using consistent coordinate and unit systems.

▷ **`ERROR: Invalid xy-extents.`**

Cause The map_custom(self, input_ras_dir , *args), map_reach(self , reach_id , feature_id , *args) functions of Mapper() class (`ModifyTerrain/cMapModifiedTerrain.py` raises this error message when the customary defined DEM raster is corrupted.

Remedy Ensure that customary defined DEM rasters are non-empty rasters with coherent coordinate and units systems and that rasters are in the stated directory for customary DEMs (default directory: `ModifyTerrain/Input/DEM/`*`condition`*`/`), as described in Sec. 18.4.

▷ **`ERROR: Invalid keyword for feature type.`**

Cause The Manager class in `MaxLifespan/cFeatureActions.py` raises this error when it received a feature_type argument that is not "terraforming", "plantings", "bioengineering", or "maintenance". The error may occur either after code modifications or when geo_file_maker (condition , feature_type , *args) in `MaxLif espan/action_planner.py` was executed as standalone or imported as a package in an external application.

Remedy – Ensure that code extensions comply with coding conventions and instructions in Sec. 16.
– Ensure that external calls of geo_file_maker (condition , feature_type , *args) contain an acceptable feature_type , i.e., feature_type = either "terraforming ", " plantings ", " bioengineering ", or "maintenance ".

▷ **`ERROR: Lifespan data fetch failed.`**

Cause The get_lifespan_data (self) or get_design_data (self) function of the ArcPyContainer class in `MaxLif espan/cActionAssessment.py` raise this error when it could not retrieve lifespan or design maps from the defined lifespan/design input directory.

Remedy – Check lifespan/design folder definitions (review Sec. 14).
– Ensure that lifespan and/or design rasters are in the defined folder.

▷ **`ERROR: Mapping failed.`**

Cause The function make_pdf_maps(self, *args) (`LifespanDesign/cMapLifespanDesign.py` or `Act ionPlanner/cMapActions.py`) or map_custom(self, input_ras_dir , *args), map_reach(self , reach_id , feature_id , *args) (`ModifyTerrain/cMapModifiedTerrain.py`) raise this error message when it could not create `PDF` maps.

Remedy – *LifespanDesign (1)*: The layout files in `LifespanDesign/Output/Mapping/`*`condition`*`/Layou ts/` are either corrupted or non-existent. Re-run Layout Maker or successively re-run Raster Maker and Layout Maker. Follow exactly the instructions for preparing map files (see Sec. 8.7.2).

– *LifespanDesign (2)*: Make sure that the file `legend.ServerStyle` exists in `LifespanDesign/Output/Mapping/.ReferenceLayouts`
– *MaxLifespan*: Ensure consistent layout files in `MaxLifespan/.templates/layouts/` (see Sec. 14.3.2) and trace back earlier warning and error messages.
– *ModifyTerrain*: Ensure consistent layout files in `ModifyTerrain/Input/Layouts/` (see Sec. 18.6) and trace back earlier warning and error messages.

▷ **`ERROR: Map layout preparation failed.`**

Cause The prepare_layout (self) functions of Mapper() classes (`LifespanDesign/cMapLifespanDesign.py`, `MaxLifespan/cMapActions.py` or `ModifyTerrain/cMapModifiedTerrain.py`) raise this error message when they encounter problems with either the provided rasters or layout (`.mxd`) files.

Remedy – *LifespanDesign*: If a layout (`.mxd`) in `LifespanDesign/Output/Mapping/.ReferenceLayouts/` was modified, ensure similar layer structures in the `.mxd` files corresponding to the existing templates (default directory: `LifespanDesign/Output/Rasters/`*`condition`*`/`) or layout templates (`.mxd` files in `LifespanDesign/Output/Mapping/.ReferenceLayouts`).
– *MaxLifespan*: Ensure that all relevant `.mxd` layouts (" terraforming ", " plantings ", " bioengineering ", or "maintenance") are contained in the `MaxLifespan/.templates/layouts/` directory (see also Sec. 14.3.2). If needed, add new layouts after code modifications (Sec. 16).
– *ModifyTerrain*: Ensure that a layout template exists (explanations in Sec. 18.6).

▷ **`ERROR: Mapping could not assign xy-values. Undefined zoom.`**

Cause Error raised by the zoom2map(self, xy) functions of the Mapper() classes (`LifespanDesign/cMapLifespanDesign.py`, `MaxLifespan/cMapActions.py` or `ModifyTerrain/cMapModifi edTerrain.py`) when it receives a bad format of x-y values.

Remedy – Ensure the correct format of `mapping.inp` (LifespanDesign or *MaxLifespan* module) corresponding to the definitions in Sec. 11.2.
– Ensure correct setup of `ModifyTerrain/.templates/computation_extents.xlsx` (Sec. 18.3).

▷ **`ERROR: Missing (or wrong format of) raster input definitions.`**

Cause Raised by get_line_entries (self , line_no) function of the Info () class in `LifespanDesign/cReadInpLifespan.py` when `LifespanDesign/.templates/input_definitions.inp` is corrupted.

Remedy Ensure that the file `LifespanDesign/.templates/input_definitions.inp` exists in the directory `LifespanDesign/.templates/` corresponding to the definitions in Sec. 11.1. In case of doubts: Replace `LifespanDesign/.templates/input_definitions.inp` with the original file and re-apply modifications strictly following Sec. 11.

▷ **`ERROR: Multiple openings of Fish.xlsx. Close all office apps ...`**

Cause Raised by the assign_fish_names (self) function of the Fish () class in `HabitatEvaluation/cFish.py` when `HabitatEvaluation/.templates/Fish.xlsx` is opened by another program or non-existent.

Remedy Ensure that the file `HabitatEvaluation/.templates/Fish.xlsx` exists and close any software that may use the workbook.

▷ **`ERROR: No HSI assigned for parameter type ...`**

Cause Raised by the get_hsi_curve (self, species, lifestage, par) function of the Fish() class in `HabitatEvaluation/cFish.py` when `HabitatEvaluation/.templates/Fish.xlsx` it expected a habitat suitability curve for `par`, but it could not find values..

Remedy Ensure that the file `HabitatEvaluation/.templates/Fish.xlsx` has valid contents according to Sec. 22.2.

▷ **`ERROR: No custom (DEM/feature) raster found.`**

Cause The map_custom(self, input_ras_dir, ∗args) function of Mapper() class (`ModifyTerrain/cMapModifiedTerrain.py` raises this error message when it cannot find customary defined DEM rasters (default directory: `ModifyTerrain/Input/DEM/`*`condition`*`/`).

Remedy Ensure that customary defined DEM rasters are in the stated directory for customary DEMs (default directory: `ModifyTerrain/Input/DEM/`*`condition`*`/`), as described in Sec. 18.4.

▷ **`ERROR: No HSI assigned for parameter type ...`**

Cause Raised by the get_hsi_curve (self, species, lifestage, par) function of the Fish() class in `HabitatEvaluation/cFish.py` when `HabitatEvaluation/.templates/Fish.xlsx` it expected a habitat suitability curve for `par`, but it could not find values..

Remedy Ensure that the file `HabitatEvaluation/.templates/Fish.xlsx` has valid contents according to Sec. 22.2.

▷ **`ERROR: No layout template found (feature ID: FEAT.`**

Cause Error raised by the choose_ref_layout (self, feature_id, volume_type) function of the Mapper() class in `ModifyTerrain/cMapModifiedTerrain.py` when it cannot match layout files in `ModifyTerrain/Input/Layouts/`*`condition`*`/` for the feature shortname FEAT and a neg or pos string.

Remedy Ensure that a layout is available in `ModifyTerrain/Input/Layouts/`*`condition`*`/` according to the descriptions in Sec. 18.6.

▷ **`ERROR:`** ***`PAR`*** **`- raster copy to Output/Rasters folder failed.`**

Cause The `.cache` folder does not exist or does not contain GRID rasters or the output folder is not accessible. This error is likely to occur when other errors occurred previously.

Remedy – Follow trouble shooting of other error messages and re-run.
– Avoid modifications of any folder in the code directory while the program is running, in particular, `.cache`, `01_Conditions/`, `LifespanDesign/Output/Rasters/` and `LifespanDesign/Output/Mapping/`.

▷ **`ERROR: Raster copy to Output folder failed.`**

Cause The save_rasters (self) function of the ModifyTerrain() class in `ModifyTerrain/cModifyTerrain.py` raises this error when saving a terrain differences or new DEM raster failed.

Remedy Refer to the error `ERROR: Raster could not be saved.` message.

▷ **`ERROR: Raster could not be saved.`**

Cause The save_rasters (self) function of the ModifyTerrain() class in `ModifyTerrain/cModifyTerrain.py` raises this error when a terrain differences or new DEM raster is corrupted.

Remedy Potential reasons for corrupted rasters are:
– The computed volume difference or new DEM raster is empty or contains `NoData` pixels only. The design parameters or raster of the concerned feature need to be reviewed.
– The `ModifyTerrain/.cache/` folder is locked by another program. Close potential applications, and if necessary, reboot the system. – If `ModifyTerrain/.cache/` was not empty before the module execution, error may occur. Manually delete `ModifyTerrain/.cache/` if it still exists after a run task.
– The directory `ModifyTerrain/Output/Rasters/`*`condition`*`/` was deleted or it is locked by another program. Ensure that the directory exists and no other program uses `ModifyTerrain/Output/Rasters/`*`condition`*`/` or its contents.

▷ **`ERROR: Raster identification failed. Omitting layout creation of ...`**

Cause Error message raised by the choose_ref_layout (self , raster_name) function in `LifespanDesign/cMapLifespanDesign.py` when it cannot assign a layout template from `LifespanDesign/Output/Mapping/.ReferenceLayouts` to a raster (default storage directory: `LifespanDesign/Out put/Rasters/`*`condition`*`/`).

Remedy – If a layout (`.mxd`) in `LifespanDesign/Output/Mapping/.ReferenceLayouts/` was modified, make sure to implement changes also in the choose_ref_layout (self , raster_name) function (`LifespanDesign/cMapLifespanDesign.py`).
– If a new output raster type results from modifications or extensions of the parameters, analysis or feature methods (Sections 12.3, 12.4 and 12.5, respectively), ensure that the conditional phrases in choose_ref_layout (self , raster_name) (`LifespanDesign/cMapLifespanDesign.py`) can identify it and assign an existing layout (`.mxd`) from `LifespanDesign/Output/Mapping/.ReferenceLayouts/`.

▷ **`ERROR: Received request for volume calculation but not input directory ...`**

Cause The __call__ (self , ∗args) function of the ModifyTerrain() class in `ModifyTerrain/cModifyTerrain.py` raises this error when it received args [0] = True (enable volume calculator only), but no input directory for a modified terrain is given (missing args [1] = DIRECTORY). This error may occur if the code was modified or called externally.

Remedy Ensure that the input directory of manually modified (Customary Feature) raster DEMs (default: `ModifyTerrain/Input/DEM/`*`condition`*`/`) is correctly passed to the *ModifyTerrain* object.

▷ **`ERROR: Scale is not INT. Substituting scale: 2000.`**

Cause Raised by get_map_scale (self) function of the Info () class in either `LifespanDesign/cReadInpLifespan.py` or `MaxLifespan/cReadActionInput.py` when it cannot interpret the value assigned to the map scale.

Remedy Ensure that the file `mapping.inp` (in `LifespanDesign/.templates/` or `MaxLifespan /.templates/`) has a correct assignment of the map scale according to the descriptions in Sec. 11.2.

▷ **`ERROR: Shapefile conversion failed.`**

Cause Raised by calculate_wua (self) of *HabitatEvaluation*'s CHSI() class in `HabitatEvaluation/cHSI.py` when it could not convert the CHSI raster to a shapefile.

Remedy – Ensure that the WUA threshold has a meaningful value between 0.0 and 1.0 (Sec. 21).
– Ensure that neither the directory `HabitatEvaluation/.cache/` nor the directory `HabitatEvaluation/WUA/` or their contents are in use by other programs.
– Review the input settings according to Sec. 22.
– Follow up earlier error messages.

▷ **`ERROR: TEMPLATE sheet does not exist.`**

Cause Error raised by the write_volumes(self , ...) function of the Writer () class in `.site_packages/riverpy/cTerrainIO.py`) or the make_condition_xlsx (self , fish_sn) of the Write () class in `HabitatEvaluation/cHabitatIO.py`) when the `template` sheet in the output (template) workbooks (`ModifyTerrain/Output/Spreadsheets/`*`condition`*`_volumes.xlsx`, `...volume_template.xlsx` or `HabitatEvaluation/.templates/Q_def_hab_template_....xlsx`) are corrupted.

Remedy – *ModifyTerrain*: Ensure that `ModifyTerrain/Output/Spreadsheets/`*`condition`*`_volumes.xlsx` or
`...volume_template.xlsx` contain the `template` sheet (Sec. 18.9.3).
– *HabitatEvaluation*: Ensure that `HabitatEvaluation/.templates/Q_def_hab_template_....xlsx` contains the `summary` sheet; re-install the templates if necessary.

▷ **`ERROR: u/h/hyd--raster analysis does not accept ras_`*`name`*` raster.`**

Cause Internal programming error: A parameter module called a raster which does not match the batch processing hierarchy.

Remedy Move new model downward in the processing hierarchy and avoid calling an u/h/hyd–raster with the optional argument `raster_info`.

▷ **`ERROR: Volume value assignment failed.`**

Cause Error raised by the write_volumes(self , ...) function of the Writer () class in `.site_packages/riverpy/cTerrainIO.py`) when it received invalid volume data.

Remedy Ensure that no other program uses `ModifyTerrain/Output/Spreadsheets/`*`condition`*`_volume.xlsx` and trace back earlier errors (modified DEM rasters may be corrupted).

▷ **`ERROR: Writing failed.`**

Cause Error raised by the write_volumes(self , ...) function of the Writer() class in `.site_packages/riverpy/cTerrainIO.py`) when the `template` it could not add new sheets in `ModifyTerrain/Output/Spreadsheets/`*`condition`*`_volumes.xlsx` or write to copies of `ModifyTerrain/Output/Spreadsheets/volume_template.xlsx`.

Remedy See error message `ERROR: Failed to create WORKBOOK`.

▷ **`ERROR: Wrong format of lifespan list (.inp)`**

Cause Raised by lifespan_read (self) (in `LifespanDesign/cReadInpLifespan.py`) when the lifespan list in `LifespanDesign/.templates/input_definitions.inp` has a wrong format or is empty.

Remedy Ensure that the file `LifespanDesign/.templates/input_definitions.inp` (in `LifespanDesign/.templates/`) contains a lifespan list (return periods list) with not more than six comma-separated entries according to the definitions in Sec. 11.1.

▷ **`ExceptionERROR: (arcpy) [...].`**

Cause The error is raised if any arcpy application of any module encountered problems; e.g., the analysis_ ... and design_ ... functions in `LifespanDesign/cLifespanDesignAnalysis.py` raise this error when raster calculations could not be performed. Missing rasters, bad raster assignments or bad raster calculation expressions are possible reasons. The error can also occur when the `Spatial` license is not available.

Remedy – Make sure that a `Spatial` license is available.
– Trace back previous error and warning messages.
– Verify raster calculation expressions in concerned analysis_ ... and design_ ... functions (`LifespanDesign/cLifespanDesignAnalysis.py`).
– Verify raster definitions in concerned analysis_ ... and design_ ... functions (`LifespanDesign/cLifespanDesignAnalysis.py`).
– Verify raster definitions of used parameters (`cParameters.py` and input files `*.inp` according to Sec. 11).
– If further system errors are stated, trace back error messages.

▷ **`ExceptionERROR: Cannot find package files [...].`**

Cause The program cannot retrieve the listed internal files.

Remedy Check the installation of the package and its file structure according to Sec. 2.

▷ **`ExceptionERROR: Cannot open reference (condition) ...`**

Cause Raised by the ModifyTerrain() class (__init__ (self , condition , feature_type , *args)) in `ModifyTerrain/cModifyTerrain.py` when it cannot find a `...` raster in `01_Conditions/`*`condition`*`/` (or other user defined input directory), where `...` is either a `dem` or a `wt_depth_base` raster. A `wt_depth_base` raster is required for automated terrain modification after grading and/or widen features.

Remedy Ensure that the missing raster (`dem` or a `wt_depth_base`) exists in `01_Conditions/`*`condition`*`/`, or if applies, the user defined input directory. If no `wt_depth_base` raster is available, the terrain modification of grading and/or widen features cannot be automated. In this case, consider adding a new DEM automation function (explained in Sec. 20.2) or modifying the DEM manually.

▷ **`ExceptionERROR: Could not find base raster for assigning lifespans.`**

Cause Raised by *MaxLifespan*'s ArcPyContainer() class (__init__ (self , condition , feature_type , *args)) in `MaxLifespan/cActionAssessment.py` when it cannot find its zero raster template in `MaxLifespan/.templates/rasters/zeros`.

Remedy Follow the instructions for the error message `ExceptionERROR: Unable to create ZERO Raster. Manual intervention required:...` to manually create the `MaxLifespan/.templates/rasters/zeros` raster.

▷ **`ExceptionERROR: Could not retrieve zero raster from` *`MaxLifespan`*`.`**

Cause Raised by the ModifyTerrain() class (__init__ (self , condition , feature_type , *args)) in `ModifyTerrain/cModifyTerrain.py` when it cannot find the zero raster template in `MaxLifespan/.templates/rasters/zeros`.

Remedy Follow the instructions for the error message `ExceptionERROR: Unable to create ZERO Raster. Manual intervention required:...` to manually create the `MaxLifespan/.templates/rasters/zeros` raster.

▷ **`ExceptionERROR: Missing fundamental packages (required: ...).`**

Cause The listed (required) packages are not available.

Remedy Check installation of required packages and code structure files according to Sec. 2.

▷ **`ExceptionERROR: Unable to create ZERO Raster. Manual intervention required`**

Cause *MaxLifespan* failed to create a zero raster covering the computation area.

Remedy The raster creation needs to be manually made in *ArcMap*'s Python interpreter (the external interpreter could not do the job and only the cuckoo from California knows why). Thus, manually create the zeros raster as follows:

1. Launch *ArcMap* and its implemented Python window (`Geoprocessing` dropdown menu: `Python`).
2. Enter the following sequences (replace REPLACE_... according to the local environment):

```
import os
from arcpy.sa import *
zero_ras_str = os.getcwd() + "\\.templates\\rasters\\zeros"
condition = "REPLACE_CONDITION"
base_dem = arcpy.Raster("REPLACE_PATH\\RiverArchitect\\
    LifespanDesign\\Input\\" + condition + "\\dem")
arcpy.gp.overwriteOutput = True
arcpy.env.extent = base_dem.extent
arcpy.env.workspace = "D:\\Python\\RiverArchitect\\LifespanDesign\\
    Input\\" + condition + "\\"
zero_ras = Con(IsNull(base_dem), 0, 0)
zero_ras.save(zero_ras_str)
```

3. Close *ArcMap*

▷ **`ExecuteERROR: (arcpy) [...].`**

Cause Similar to ExceptionERROR: (arcpy) ... . The error is raised by arcpy applications of all modules; e.g., by the analysis_ ... and design_ ... functions in `LifespanDesign/cLifespanDesignAnalysis.py` or when raster calculations could not be performed. Missing rasters, bad raster assignments or bad raster calculation expressions are possible reasons. The error can also occur when the `Spatial` license is not available.

Remedy See `ExceptionERROR: (arcpy) [...]`

▷ **`WindowsError: [Error 32] The process cannot access the file because ...`**

Cause Files in the `.cache`–folder or the `Output`–folder are used by another program.

Remedy – Make sure that `ArcGIS` Desktop is not running.
– Make sure that no other code copy (Python) uses these folders.

30.2 Warning messages

▷ **`WARNING: .cache folder will be removed by package controls.`**

Cause Raised by clear_cache (self) of *HabitatEvaluation*'s CHSI() class in `HabitatEvaluation/cHSI.py` when it could not clear and remove the `.cache/` folder.

Remedy Ensure that no other software uses the temporary rasters stored in `HabitatEvaluation/.cache/`, and if necessary, delete the folder manually after quitting the module.

▷ **`WARNING: Bad value ( ... ).`**

Cause Raised by calculate_wua (self) of *HabitatEvaluation*'s CHSI() class in `HabitatEvaluation/cHSI.py` when a CHSI polygon contains an invalid value.

Remedy Review cHSI rasters `HabitatEvaluation/CHSI/`*`condition`*`/`.

▷ **`WARNING: computation_extents.xls contains too many reach names.`**

Cause Raised by Read(). get_reach_info (self , type) in `.site_packages/riverpy/cTerrainIO.py` when `ModifyTerrain/.templates/computation_extents.xlsx` contains more than eight reach names in columns `B` and/or `C`.

Remedy Ensure that `ModifyTerrain/.templates/computation_extents.xlsx` does not contain more than eight reaches, i.e., only cells `B6:C13` contain reach names and identifiers (cf. Sec. 18.3).

▷ **`WARNING: Conversion to polygon failed (FEAT).`**

Cause Raised by identify_best_features (self) in `MaxLifespan/cActionAssessment.py` when the arcpy . RasterToPolygon_conversion (FEAT raster) failed, e.g., because of an empty FEAT raster.

Remedy An empty FEAT raster of best lifespans occurs when the feature has no spatial relevance. Consider other terrain modifications or maintenance features to increase the features lifespans and start over planning the feature (set).

▷ **`WARNING: Could not clear/remove .cache.`**

Cause All modules may raise this warning message when the content in the `.cache` folder was accessed and locked by another software.

Remedy Make sure that no other software, including *ArcMap* Desktop or `explorer.exe` uses the `MODULE/.cache` folder.

▷ **`WARNING: Could not clean up PDF map temp_pages.`**

Cause The make_pdf_maps(self, ∗args) or finalize_map (self) functions of Mapper() classes in either `LifespanDesign/cMapLifespanDesign.py`, `MaxLifespan/cMapActions.py` or `ModifyTerrain/cMapModifiedTerrain.py` create single `PDF`s of every map image. These single-page `PDF`s are finally combined into one `PDF` map assembly and the single-page `PDF`s are deleted afterward. If the single-page `PDF`s are locked by another process or corrupted, the make_pdf_maps(self, ∗args) function raises this warning message when it cannot remove temporary .

Remedy Ensure that no other program is using the `PDF` files in `MODULE/Output/Maps/`*`condition`*`/` while mapping is in progress.

▷ **`WARNING: Could not clear temp.lyr`**

Cause The function prepare_layout (self) (`LifespanDesign/cMapLifespanDesign.py`) prints this warning message when it cannot remove the `temp` layer from the layout template.

Remedy Ensure that no other program is using the `.mxd` files (layout), which is used for the map preparation, or the `.cache` folder.

▷ **`WARNING: Could not divide [...] by [...]"`**

Cause Raised by calculate_relative_exceedance (self) of *HabitatEvaluation*'s FlowAssessment() class in `HabitatEvaluation/cHSI.py` when the flow duration curve contains invalid data.

Remedy Ensure the correct setup of the used flow duration curve in `HabitatEvaluation/FlowDurationCurves/`. The data types and file structure must correspond to that of the provided template `flow_duration_template.xlsx` and all discharge values need to be positive floats. Review Sec. 22.5 for details.

▷ **`WARNING: Could not get flow depth raster properties. Setting [...]`**

Cause The crop_input_raster (self , ...) function (`HabitatEvaluation/cHSI.py`) prints this warning message when it cannot read the raster properties from the defined input flow depth raster.

Remedy Make sure that the defined flow depth raster exists in `RiverArchitect/01_Conditions/`*`condition`*`/`.

▷ **`WARNING: Could not get minimum flow depth [...]. Setting h min [...]`**

Cause The crop_input_raster (self , ...) function (`HabitatEvaluation/cHSI.py`) prints this warning message when it could not read the minimum flow depth from `Fish.xlsx`. A default value of 0.1 (ft or m) is used to delineate relevant flow regions.

Remedy Make sure that the defined Fish species / lifestage is assigned a cover value and at least one flow depth value in `Fish.xlsx` according to the definitions in Sec. 22.6.

▷ **`WARNING: Could not reset styles.`**

Cause Raised by Write().write_volumes(self, ...) in `.site_packages/riverpy/cTerrainIO.py` when the `template` sheet in the output (template) workbook (`ModifyTerrain/Output/Spreadsheets/`*`condition`*`_volumes.xlsx` or `...volume_template.xlsx`) is either locked or not accessible.

Remedy Ensure that no other program uses `ModifyTerrain/Output/Spreadsheets/`*`condition`*`_volumes.xlsx` or `...volume_template.xlsx` and that both workbooks have not been accidentally deleted.

▷ **`WARNING: Could not read project area extents.`**

Cause Raised by CWUA().get_extents(self, ...) in `/ProjectMaker/cWUA.py` when the function failed to read the project area extents from the `ProjectArea.shp` shapefile.

Remedy Ensure that the texttProjectArea.shp shapefile is correctly created (in particular the *Attributes Table*), according to Sec. 26.4.

▷ **`WARNING: Could not set project area extents ().`**

Cause Raised by CWUA().set_env(self) in `/ProjectMaker/cWUA.py` when the function failed to set project area extents.

Remedy Occurs when the CHSI Raster associated with a certain discharge is empty. Ignore this Warning if the CHSI Raster was correctly identified as being empty, otherwise, revise CHSI Raser creation with the *HabitatEvaluation* module (part V).

▷ **`WARNING: Design map - Could not assign frequency threshold. [...]`**

Cause Design maps, such as stable grain size, refer to hydraulic data related to a defined return period. If design_ ... functions) `LifespanDesign/cLifespanDesignAnalysis.py`) cannot identify a particular threshold_freq value, design_ ... functions automatically try to use hydraulic data related to the first entry of lifespans (`Return periods` entry in `LifespanDesign/.templates/input_definitions.inp`, see Sec. 11.1).

Remedy – Assign a float value to the concerned feature in the `Mobility frequency threshold` row of the `LifespanDesign/.templates/threshold_values.xlsx[thresholds]` spreadsheet (see also Sec. 8.3).
– Make sure that the defined defined `Mobility frequency threshold` float is consistent with the defined `Return periods` in `LifespanDesign/.templates/input_definitions.inp` (see Sec. 11.1).

▷ **`WARNING: Empty design raster [...]`**

Cause The analyzed feature is not applicable in the defined range.

Remedy – If the feature is not intended to be applied anyway, ignore the warning message.
– If the feature is intended to be applied, manual terrain modifications adapting the feature's threshold values may be necessary.

▷ **`WARNING: Empty lifespan raster [...]`**

Cause The analyzed feature is not applicable in the defined range.

Remedy – If the feature is not intended to be applied anyway, ignore the warning message.
– If the feature is intended to be applied, manual terrain modifications adapting the feature's threshold values may be necessary.

▷ **`WARNING: Failed to arrange worksheets.`**

Cause Raised by Write().write_volumes(self, ...) in `.site_packages/riverpy/cTerrainIO.py` when it could not bring to front the latest copy of the `template` sheet in the output (template) workbook (`Modify Terrain/Output/Spreadsheets/`*`condition`*`_volumes.xlsx` or `...volume_template.xlsx`), which contains the calculation results.

Remedy Trace back earlier error and warning messages. Ensure that no other program uses `ModifyTerrain/Output/Spreadsheets/`*`condition`*`_volumes.xlsx` or `...volume_template.xlsx` and that both workbooks have not been accidentally deleted.

▷ **`WARNING: Failed to write unit system to worksheet.`**

Cause Raised by Write().write_volumes(self, ...) in `.site_packages/riverpy/cTerrainIO.py` when it could not write volume (numbers) to a copy of the `template` sheet in the output (template) workbook (`ModifyTerrain/Output/Spreadsheets/`*`condition`*`_volumes.xlsx` or `...volume_template.xlsx`).

Remedy Trace back earlier error and warning messages. Ensure that no other program uses `ModifyTerrain/Output/Spreadsheets/`*`condition`*`_volumes.xlsx` or `...volume_template.xlsx` and that both workbooks have not been accidentally deleted.

▷ **`WARNING: Flow_duration[...].xlsx has different lengths of [...]"`**

Cause Raised by get_flow_data(self, *args) of *HabitatEvaluation*'s FlowAssessment() class in `HabitatEvaluation/cHSI.py` when the flow duration curve contains invalid data.

Remedy Ensure that columns `B` and `C` of the flow duration curve workbook have the same length (in particular the last value / row must be the same) and check for empty cells.

▷ **`WARNING: Identification failed (FEAT).`**

Cause Raised by identify_best_features(self) in `MaxLifespan/cActionAssessment.py` when the analyzed feature cannot matched with the internal best lifespan raster.

Remedy Features with very low lifespan may result in empty rasters. Consider other terrain modifications or maintenance features to increase the features lifespans and start over planning the feature (set).

▷ **`WARNING: Invalid feature names for column headers.`**

Cause Raised by Write().write_volumes(self, ...) in `.site_packages/riverpy/cTerrainIO.py` when it could not write feature names to a copy of the `template` sheet in the output (template) workbook (`ModifyTerrain/Output/Spreadsheets/`*`condition`*`_volumes.xlsx` or `...volume_template.xlsx`).

Remedy – Ensure that `ModifyTerrain/.templates/computation_extents.xlsx` contains valid reach descriptions (Sec. 18.3).
– Ensure that no other program uses `ModifyTerrain/Output/Spreadsheets/`*`condition_`*`volumes.xlsx` or `...volume_template.xlsx` and that both workbooks have not been accidentally deleted.

▷ **`WARNING: Invalid type assignment -- setting reach names to IDs.`**

Cause Raised by Read(). get_reach_info (self , type) in `.site_packages/riverpy/cTerrainIO.py` when the type argument is not full_name or id. In this case, the *ModifyTerrain* module uses column `C` in `ModifyTerrain/.templates/computation_extents.xlsx` for reach names and IDs.

Remedy This warning message only occurs if the GUI application was changed or when the *ModifyTerrain* module is externally called with bad argument order. Review the argument order/assignments in the external call and ensure that the type variable is in the allowed_types = [”full_name”, ”id”] list.

▷ **`WARNING: Invalid unit_system identifier.`**

Cause Raised by ModifyTerrain. __init__ () in `ModifyTerrain/cModifyTerrain.py` when the unit system identifier is not either `us` or `si`. The program will use the default unit system (`U.S. customary`).

Remedy This warning message only occurs if the GUI application was changed or when the *ModifyTerrain* module is externally called with bad argument order. Review the argument order/assignments in the external call var = mt.ModifyTerrain(condition =..., unit_system =..., ...) .

▷ **`WARNING: Old logfile is locked [...].`**

Cause Raised by the logging_start (logfile_name) function (multiple classes) when the logfiles are locked by another process. The parenthesis `[...]` indicate the concerned run task.

Remedy Ensure that the logfiles of the concerned module are not opened in any other process/program.

▷ **`WARNING: Overwriting existing/old ...`**

Cause The concerned directory already contains an output file of the same name, which is overwritten now.

Remedy Ensure to save important layout files in another directory if overwriting is not desired. Cut and paste relevant layouts and maps after every run of Layout Maker or Map Maker to `LifespanDesign/Products/.../`*`condition`*`/` and modify file names.

▷ **`WARNING: Raster / layout identification failed. Using lifespan ...`**

Cause Warning message from the choose_ref_layer (self , feature_type) function (`LifespanDesign/cMapLifespanDesign.py`) if it cannot determine the raster type, i.e., whether it is a lifespan or a design raster. In this case, the layer symbology of lifespan maps is assign by default, which can cause errors later on.

Remedy – Verify the layout templates (`.mxd`) in `LifespanDesign/Output/Mapping/.ReferenceLayouts/` for correct layer names, i.e., ”lf_sym” for lifespan and ”ds_sym” for design map templates.
– Ensure that all layout templates (`.mxd`) in `LifespanDesign/Output/Mapping/.ReferenceLayouts/` names either start with `lf` or `ds` for lifespan and design layouts, respectively.

▷ **`WARNING: Volume value assignment failed.`**

Cause Raised by `Write().write_volumes(self, ...)` in `.site_packages/riverpy/cTerrainIO.py` when it could not write volume (numbers) to a copy of the `template` sheet in the output (template) workbook (`ModifyTerrain/Output/Spreadsheets/`*`condition`*`_volumes.xlsx` or `...volume_template.xlsx`).

Remedy Trace back earlier error and warning messages. Ensure that no other program uses `ModifyTerrain/Output/Spreadsheets/`*`condition`*`_volumes.xlsx` or `...volume_template.xlsx` and that both workbooks have not been accidentally deleted.

References

Bovee, K. D., Sep. 1986. Development and evaluation of Habitat Suitability Criteria for use in the instream flow incremental methodology. Tech. Rep. 21, National Ecology Center, U.S. Fish and Wildlife Service, Fort Collins, CO, USA, Biological Report 86(7).
URL `https://pubs.er.usgs.gov/publication/70121265`

Bywater-Reyes, S., Wilcox, A. C., Stella, J. C., Lightbody, A. F., 2015. Flow and scour constraints on uprooting of pioneer woody seedlings. Water Resources Research 51 (11), 9190–9206.
URL `http://dx.doi.org/10.1002/2014WR016641`

Carley, J. K., Pasternack, G. B., Wyrick, J. R., Barker, J. R., Bratovich, P. M., Massa, D. A., Reedy, G. D., Johnson, T. R., 2012. Significant decadal channel change 58-67 years post-dam accounting for uncertainty in topographic change detection between contour maps and point cloud models. Geomorphology 179 (Supplement C), 71–88.
URL `http://www.sciencedirect.com/science/article/pii/S0169555X12003819`

Detert, M., Kadinski, L., Weitbrecht, V., 2018. On the way to airborne gravelometry based on 3D spatial data derived from images. International Journal of Sediment Research 33 (1), 84–92.
URL `http://www.sciencedirect.com/science/article/pii/S1001627918300350`

Friedman, J. M., Auble, G. T., 1999. Mortality of riparian box elder from sediment mobilization and extended inundation. Regulated Rivers: Research & Management 15 (5), 463–476.
URL `http://dx.doi.org/10.1002/(SICI)1099-1646(199909/10)15:5$<$463::AID-RRR559$>$3.0.CO;2-Z`

Gaeuman, D., 2008. Recommended quantities and gradation for long-term coarse sediment augmentation downstream from Lewiston Dam. Tech. Rep. TM-TRRP-2008-2, Trinity River Restoration Program, Weaverville, CA, USA.

Jablkowski, P., Johnson, E. A., Martin, Y. E., 2017. Role of river flow and sediment mobilization in riparian alder establishment along a bedrock-gravel river, south fork eel river, california. Geomorphology 295 (Supplement C), 28–38.
URL `http://www.sciencedirect.com/science/article/pii/S0169555X1730274X`

Jackson, J. R., Pasternack, G. B., Wyrick, Joshua, R., Mar. 2013. Lower Yuba River Accord Monitoring and Evaluation program: Substrate of the Lower Yuba River. resreport, University of California Davis, Davis, CA, USA, Prepared for the Lower Yuba River Accord River Management Team.
URL `http://www.yubaaccordrmt.com/Annual$%$20Reports/Mapping$%$20and$%$20Modeling/LYRsubstrate20131218.pdf`

Kui, L., Stella, J. C., 2016. Fluvial sediment burial increases mortality of young riparian trees but induces compensatory growth response in survivors. Forest Ecology and Management 366 (Supplement C), 32–40.
URL `http://www.sciencedirect.com/science/article/pii/S0378112716300123`

Lange, D., Bezzola, G. R., 2006. Schwemmholz Probleme und Lösunganſätze [Driftwood problems and approaches for solutions]. Minor, H.-E, ed.: Mitteilung Nr. 188 der Versuchsanstalt für Wasserbau, Hydrologie und Glaziologie an der Eidgenössischen Technischen Hochschule Zürich, Zürich, Switzerland, in German.

Maynord, S., Neill, C., 2008. Sedimentation Engineering. ASCE Manual and Reports on Engineering Practice No. 110. American Society of Civil Engineering, García, M. H. (ed.), Reston, VA, USA, Ch. APPENDIX B - Riprap Design, pp. 1037–1056, EM-1601.

Moglen, G. E., 2015. Fundamentals of Open Channel Flow. CRC Press, Taylor & Francis Group, Virginia Tech, VA, USA, 270 pages.

Ock, G., Sumi, T., Takemon, Y., 2013. Sediment replenishment to downstream reaches below dams: implementation perspectives. Hydrological Research Letters 7 (3), 54–59.
URL `https://www.jstage.jst.go.jp/article/hrl/7/3/7$_$54/$_$article`

Pasquale, N., Perona, P., Francis, R., Burlando, P., 2012. Effects of streamflow variability on the vertical root density distribution of willow cutting experiments. Ecological Engineering 40 (Supplement C), 167–172.
URL `http://www.sciencedirect.com/science/article/pii/S0925857411003697`

Pasquale, N., Perona, P., Francis, R., Burlando, P., 2014. Above-ground and below-ground Salix dynamics in response to river processes. Hydrological Processes 28 (20), 5189–5203.
URL `http://dx.doi.org/10.1002/hyp.9993`

Pasquale, N., Perona, P., Schneider, P., Shrestha, J., Wombacher, A., Burlando, P., 2011. Modern comprehensive approach to monitor the morphodynamic evolution of a restored river corridor. Hydrology and Earth System Sciences 15 (4), 1197–1212.
URL `https://www.hydrol-earth-syst-sci.net/15/1197/2011/`

Pasternack, G. B., Morford, S. L., Fulton, A. A., 2010. Yuba River analysis aims to aid spring-run chinook salmon habitat rehabilitation. California Agriculture 64 (2), 69–77.

Polzin, M. L., Rood, S. B., 2006. Effective disturbance: Seedling safe sites and patch recruitment of riparian cottonwoods after a major flood of a mountain river. Wetlands 26 (4), 965–980.
URL `https://link.springer.com/article/10.1672/0277-5212(2006)26$%$5B965:EDSSSA$%$5D2.0.CO;2`

Ruiz-Villanueva, V., Wyzga, B., Zawiejska, J., Hajdukiewicz, M., Stoffel, M., 2016. Factors controlling large-wood transport in a mountain river. Geomorphology 272 (Supplement C), 21–31.

Schwindt, S., Pasternack, G. B., Bratovich, P. M., Rabone, G., Simodynes, D., 2019. Hydro-morphological parameters generate lifespan maps for stream restoration management. Journal of Environmental Management 232, 475–489.
URL `http://www.sciencedirect.com/science/article/pii/S0301479718312751`

Stähly, S., Friedrich, H., Detert, M., 2017. Size Ratio of Fluvial Grains' Intermediate Axes Assessed by Image Processing and Square-Hole Sieving. Journal of Hydraulic Engineering 143 (6), 06017005.
URL `https://ascelibrary.org/doi/abs/10.1061/%28ASCE%29HY.1943-7900.0001286`

Stromberg, J. C., Richter, B. D., Patten, D. T., Wolden, L. G., 1993. Response of a Sonoran Riparian forest to a 10-year return flood. The Great Basin Naturalist 53 (2), 118–130.
URL `http://www.jstor.org/stable/41712765`

USACE, Apr. 2000. Design and Construction of Levees. In: Engineering and Design. No. EM 1110-2-1913 in Engineer Manual. U.S. Army Corps of Engineers, Washington, DC, USA, p. 164.

USACE, YCWA, 2016. Yuba River Ecosystem Restoration Feasibility Study (YRERFS) - Habitat Measures. Tech. rep., Yuba County Water Agency (YCWA), Sacramento, CA.

van Denderen, R. P., Schielen, R. M. J., Blom, A., Hulscher, S. J. M. H., Kleinhans, M. G., 2017. Morphodynamic assessment of side channel systems using a simple one-dimensional bifurcation model and a comparison with aerial images. Earth Surface Processes and Landforms [in press], 14, esp.4267.
URL `http://dx.doi.org/10.1002/esp.4267`

Weber, M. D., Pasternack, G. B., 2017. Valley-scale morphology drives differences in fluvial sediment budgets and incision rates during contrasting flow regimes. Geomorphology 288, 39–51.
URL `http://www.sciencedirect.com/science/article/pii/S0169555X16309862`

Wilcox, A. C., Shafroth, P. B., 2013. Coupled hydrogeomorphic and woody-seedling responses to controlled flood releases in a dryland river. Water Resources Research 49 (5), 2843–2860.
URL `http://dx.doi.org/10.1002/wrcr.20256`

Wyrick, J. R., Pasternack, G. B., 2014. Geospatial organization of fluvial landforms in a gravel-cobble river: Beyond the riffle-pool couplet. Geomorphology 213 (Supplement C), 48–65.
URL `http://www.sciencedirect.com/science/article/pii/S0169555X14000099`

Wyrick, J. R., Pasternack, G. B., 2016. Revealing the natural complexity of topographic change processes through repeat surveys and decision-tree classification. Earth Surface Processes and Landforms 41 (6), 723–737, ESP-15-0190.R1.