River Architect

Release latest

River Architect Team

Nov 12, 2021

CONTENTS

Please refer to the Installation section to learn how to install, update, and launch River Architect.

River Architect is an `open-access, peer-reviewed (Journal SoftwareX) <https://doi.org/10.1016/j.softx.2020.100438>`__ and Python3-based software for the GIS-based planning of habitat enhancing river design features regarding their lifespans, parametric characteristics, optimum placement in the terrain, and ecological benefits. The main graphical user interface (GUI) provides modules for generating lifespan and design maps, terrain modification (terraforming), assessment of digital elevation models (DEM), habitat assets including fish stranding risk, and project cost-benefits.

River Architect invites to analyses and modifications of the longevity and ecological quality of riverscapes. Different planning bases ("conditions") can be easily created using an introductory module called GetStarted. **Lifespan**, **Morphology (Terraforming)** and **Ecohydraulic** assessments can then be created based on the *Conditions*, including the creation of project plans and cost tables with a **Project Maker** module.

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 and design mapping and indicate required feature dimensions for stability, such as the minimum required size of angular boulders to avoid their mobilization during floods (more information in Schwindt et al.2019). **Best lifespan 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 Max Lifespan module assess optimum features as a function of highest lifespans among comparable feature groups such as terraforming or vegetation planting species.

Morphology (Terraforming) includes routines to Modify Terrain for river restoration purposes. Currently, two terrain modification algorithms are implemented: (1) Threshold value-based terrain modifications in terms of grading or widening / broaden rivers for riparian forest establishment; and (2) River Builder for the creation of synthetic river valley. A Volume Assessment module can compare an original (pre-project or pre-terraforming application) and a modified DEM ("with implementation" or post-feature application) to determine required earth movement (terraforming volumes) works.

Ecohydraulics assessments include the evaluation of the ecohydraulic state and connectivity of riverscapes. The Habitat Area (Seasonal Habitat Area Calculator) module applies user-defined flows (discharges) for the spatial evaluation of the habitat suitability index (*HSI*) in terms of Seasonal Habitat Area (SHArea). The hydraulic habitat suitability results from 2D hydrodynamic numerical model outputs of flow depth and velocity. Also, a "cover" option can be used to assess ecohydraulic effects of cobble, boulder, vegetation, and streamwood. The Stranding Risk module provides insights into the connection of wetted areas on floodplains and how these may be improved to enhance the survivorship of fry / juvenile fish.

The Project Maker module creates preliminary construction plans and evaluates the costs for gain 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 *SHArC* module.

A set of Tools provides *Python* console scripts that are under development and will be implemented in future versions of the *River Architect* GUI.

River Architect represents a comprehensive tool for state-of-the-art planning of ecologic river modifications. The integrated application of *River Architect* to habitat enhancing project planning is illustrated in the following flowchart. 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.

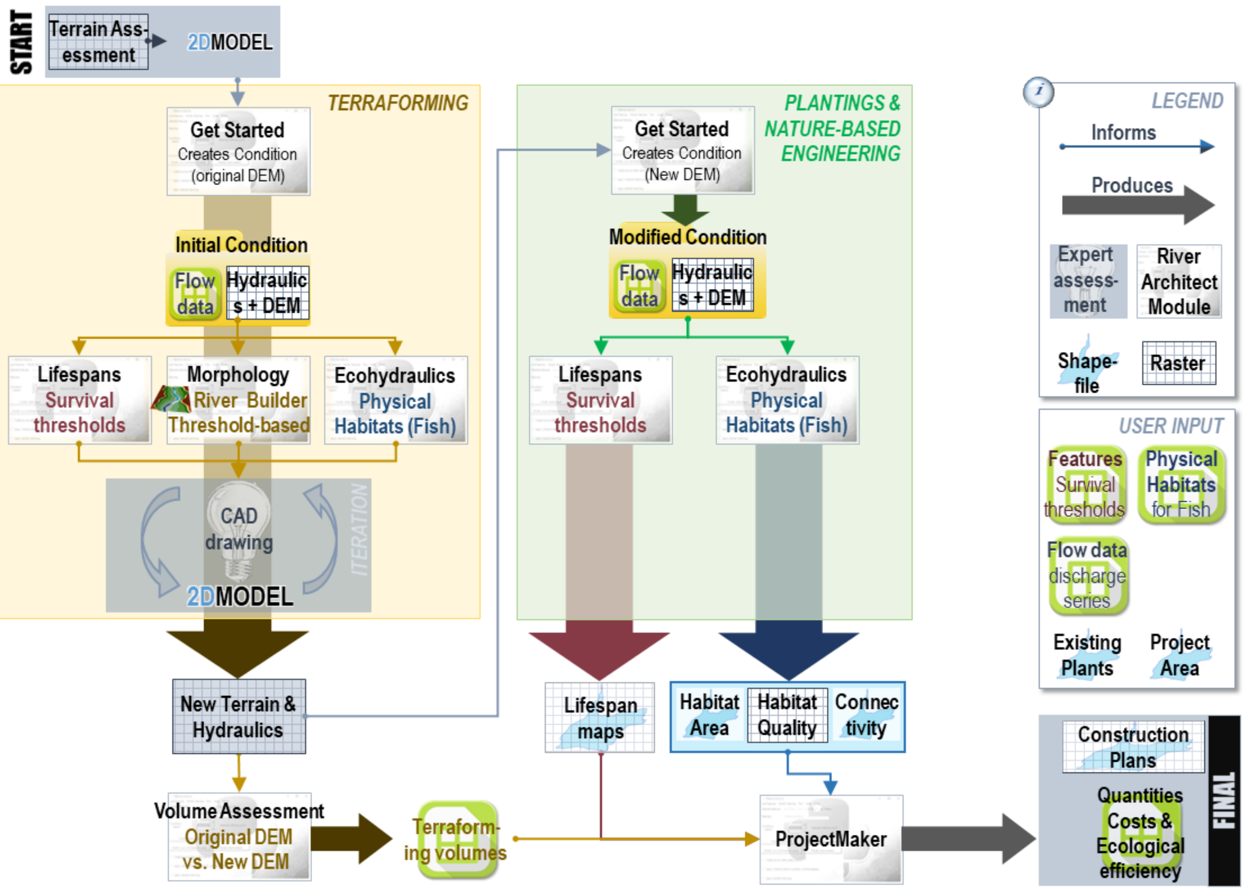


Fig. 1: flowchart

   - 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.
4. Create a Condition using the GetStarted module. The *Condition* should include *GeoTIFFs* of the initial (existing or pre-project) river state:
   - Detrended digital elevation model (DEM);
   - Flow depth and velocity for multiple discharges Rasters from 2D hydrodynamic modeling (see Signposts);
   - 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); and [Jackson et al. (2013)];
   - Datasets that can be used to assess design feature stability, such as side channel design criteria;
   - Topographic change Rasters (Topographic Change Detection or DEM differencing according to Wyrick and Pasternack 2016);
   - Depth to water table Raster (`d2w`);
   - Morphological unit Raster (see Wyrick and Pasternack (2014)).
5. Apply the LifespanDesign module to framework (terraforming) features.
6. Lifespan and Design maps, as well as expert assessment, serve for the identification of relevant framework (terraforming) features.
7. Iterative terraforming (if relevant):
   - Use the ModifyTerrain module for creating synthetic river valley with River Builder; or apply threshold value-based terrain grading or broadening of the river bed. *Please note that both routines require post-processing with computer-aided design (translation into real-world coordinates and/or edge smoothing).*
   - Re-compile the flow depth and velocity maps (re-run 2D model) with the modified DEM.
   - Verify the suitability of the modified DEM (e.g., barrier height to ensure flood safety and habitat suitability with the SHArC module); if the verification show weaknesses adapt the terraforming and re-compile the flow depth and velocity maps until terraforming is satisfactory.
   - Use the *VolumeAssessment* module (Morphology tab) to compare pre- (initial) and post-project (modified) DEMs for determining 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 / initial DEM if no terraforming applies).
9. Use the MaxLifespan module to identify best performing (highest lifespan) vegetation plantings and (other) bioengineering features.
10. If the soils are too coarse (i.e., the capillarity is not high enough to enable plant root growth), apply the connectivity feature "incorporate fine sediment in soils".
11. If gravel augmentation methods are applicable: Consecutively apply the LifespanDesign and MaxLifespan modules to connectivity features to foster self-sustaining, artificially created ecomorphological 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 SHArC module to compare the Seasonal Habitat Area (SHArea) before and after enhancement of Physical Habitats for target fish species (lifestages).
12. Use the SHArC to assess the *"existing"* (pre-project) and *"with implementation"* (post-project) habitat suitability in terms of annually usable habitat area (SHArea).

13. Use the *ProjectMaker* to calculate costs, the net gain in SHArea, and their ratio as a metric defining the project trade-off.

The working principles of the LifespanDesign, MaxLifespan, ModifyTerrain, *VolumeAssessment*, SHArC, and ProjectMaker modules are explained on their own Wiki pages. The differentiation between terraforming (framework), vegetation plantings and other bioengineering, and connectivity features is described within the LifespanDesign Wiki. The Installation Wiki pages describe the proper installation, file organization and environment of *River Architect*.

CHAPTER

ONE

INSTALLATION

- *Install* River Architect
- *Update* River Architect
- *Launch* River Architect
- *Prepare file structure*
- *Program environment setup and batchfile modification*
- *Requirements and dependencies*
- *Logfiles*

River Architect applies on *ArcGIS Pro*'s `arcpy` package and needs to be executed with *ArcGIS Pro*'s conda environment (`C:\Program Files\ArcGIS\Pro\bin\Python\scripts\propy.bat`). The *requirements* section provides more details and installation hints for external packages. *River Architect* is designed as an external application rather than a python package, but its modules can be imported as packages for external use (see the *Alternative Run* sections in the module Wikis).

1.1 Installation with *Git Bash/GUI*

We recommend downloading and updating *River Architect* using Git Bash (or Git GUI). Alternatively, *River Architect* can be downloaded as *zip* file. However, updates are tricky when *River Architect* is downloaded as *zip* file.

Git Bash *GitHub* provides detailed descriptions and standard procedures to work with their repositories (read more). The following "recipe" guides through the first time installation (cloning) of *River Architect*:

1. Open *Git Bash*
2. Type `cd "D:/Target/Directory/"` to change to the target installation directory (recommended: `cd "D:/Python/RiverArchitect/"`). If the directory does not exist, it can be created in the system explorer (right-click in empty space > `New` > > `Folder`).
3. Clone the repository: `git clone https://github.com/RiverArchitect/program`

Done. Close *Git Bash* and start working with *River Architect*.

Git GUI For using *Git GUI*, open *Git GUI* and choose `Clone Existing Repository`. Enter the *Source Location*: `https://github.com/RiverArchitect/program`, and the *Target Directory*: `D:/Python/RiverArchitect` (or any other directory, but ensure that the directory does not exist). Click `Clone`. Done.

1.2 Update with *Git Bash/GUI* (re- pull)

Currently, *Git Bash* (or Git GUI) are the only options to update local copies of *River Architect*. **Before updating *River Architect*, we recommend to save any files and *Conditions* produced with *River Architect* externally (create a save copy).**

Git Bash Open *Git Bash* and do the following:

1. Go to the local *River Architect* installation directory: `cd "D:/Python/RiverArchitect/program/"` (or wherever *River Architect* was cloned).
2. Check the current status: `git status`
3. Pull changes: `git pull`

Please note that merge errors may occur when changes were made in the local copy of *River Architect*. In this case, *Git Bash* will guide through the manual merge process. **If you modified template workbooks (`Fish.xlsx` or `threshold_values.xlsx`): Create a `.gitignore` file** in the `/RiverArchitect/` directory. Open the `.gitignore` file with a text editor and enter the following:

```
/LifespanDesign/.templates/threshold_values.xlsx
/.site_packages/templates/Fish.xlsx
```

Save the `.gitignore` file and close the text editor. Now `git pull` will no longer overwrite `Fish.xlsx` and `threshold_values.xlsx`.

Git GUI Open *Git Bash* and open `D:/Python/RiverArchitect` from the *Open Recent Repository* list (or wherever *River Architect* is installed). If not listed, click on `Open Existing Repository` and select the *River Architect* installation directory. Then:

1. Click on `Rescan`.
2. Click on the `Remote` drop-down menu > `Fetch from` > `origin`.
3. Click on the `Merge` drop-down menu > `Local Merge`.

1.3 Launch *River Architect*

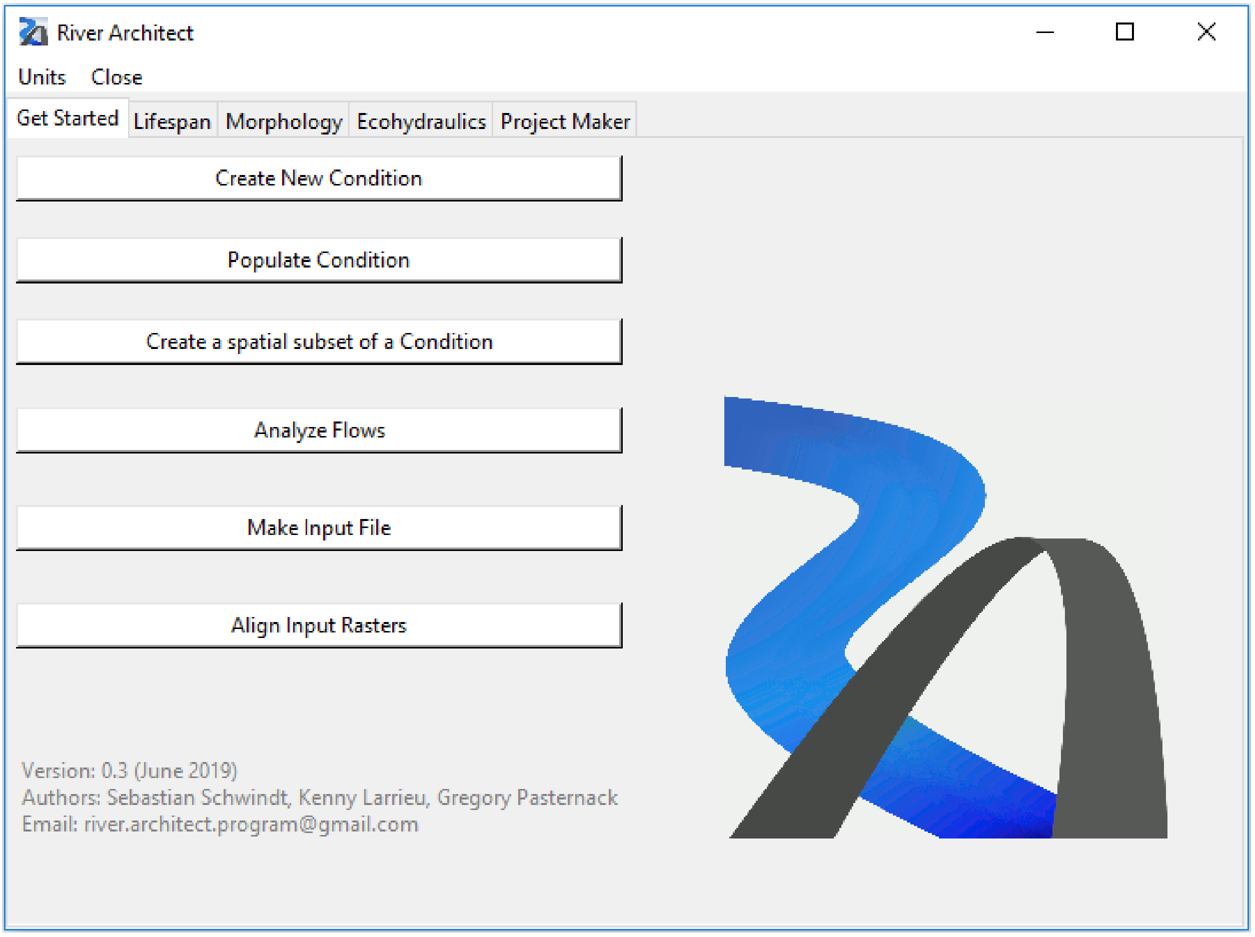
To start using *River Architect*:

1. Right-click on `DRIVE:\path\to\...\RiverArchitect\Start_River_Architect.bat` and click on `Edit`.
    - In the `Start_River_Architect.bat`chfile, ensure that the Python path is correctly defined according to the installed version of *ArcGIS*: `call` **EDIT>>`"%PROGRAMFILES%\ArcGIS\Pro\bin\Python\Scripts\propy"`<<**`"%cd%\master_gui.py"`. For more information about running standalone Python scripts in *ArcGIS Pro*'s conda environment, read their descriptions.

- Save and close `Start_River_Architect.bat`.

2. Double-click on `DRIVE:\path\to\...\RiverArchitect\Start_River_Architect.bat` to start *River Architect*.
3. Create a *Condition* on the Welcome (*Get Started*) tab.
4. Analyze and create ecohydraulic paradises for Physical Habitats prefered by target (fish) species with sustainable river design features using lifespan maps. *Hint: The **Quick GUIde** sections in the module Wikis provide straight-forward application guidance.*

River Architect starts in a GUI that is stored in `master_gui.py`. Alternatively, the modules can be individually launched by double-clicking on the `LAUNCH_Windows_x64.bat` in the module folders (not recommended and may be abandoned in future versions).



Version: 0.3 (June 2019)
Authors: Sebastian Schwindt, Kenny Larrieu, Gregory Pasternack
Email: river.architect.program@gmail.com

CHAPTER

TWO

PACKAGE STRUCTURE, REQUIREMENTS AND LOGFILES

2.1 File structure

The main directory (`/RiverArchitect/`) contains the program launcher named `Start_River_Architect.bat` and the *Python3* file `master_gui.py`. The *River Architect* modules are located in sub-folders. The master folder (`/RiverArchitect/`) includes the following files and directories (the full list of all files for developers is available here):

- **.sitepackages** Contains adapted third-party Python packages and own packages
  - `openpyxl` Contains a modified version of the `openpyxl` package (version 2.5.2) adapted for *River Architect*
  - `riverpy` Particular python scripts with recurring routines and classes that are used in multiple modules.
    - `cDefinitions.py` contains inter-module information of reach and feature keywords.
    - `cFeatures.py` contains river design features classes with pointers to parameters and threshold values.
    - `cFish.py` contains the class that reads characteristic species and lifestage data from `riverpy/templates/Fish.xlsx`.
    - `cFlows.py` contains classes for creating flow duration curves and interpolating the annual/seasonal flow durations of 2D-modeled discharges.
    - `cGravel.py` contains subfeatures of the Gravel augmentation-feature in `cFeatures`.
    - `cInputOutput.py` contains workbook (`.xlsx`) read and write routines.
    - `cInterpolator.py` provides routines for interpolating data between exact data points (x and y).
    - `cLogger.py` contains the `Logger()` class that specifies the appearance of logging messages.
    - `cMakeTables.py` make habitat evaluation workbook tables (`.xlsx` format).
    - `cMapper.py` contains mapping routines (see Mapping Wiki).
    - `cPlants.py` contains subfeatures of the Plantings-feature in `cFeatures` and the `ModifyTerrain` module.
    - `cReachManager.py` processes reach information and reach-wise terrain change (volume differences).
    - `cThresholdDirector.py` provides the `ThresholdDirector()` class for reading threshold values from the "thresholds.xlsx" workbook.
    - `config.py` contains global parameter definitions and pointers to workbooks.

    * `fGlobal.py` provides functions that are required in this module and the other modules in several classes.
    *
  - `templates` contains global template files (content will be enriched in future versions)
- **00_Flows** contains templates for flow (discharge) analyses.
- **01_Conditions** contains `Condition` folders with parameter Rasters. The condition name begins with a 4-digit year number (e.g., `2018`), optionally followed by a 3-characters reach ID (e.g., `xyz`) and a feature layer indicator (e.g., `lyr10` for terraforming features). The syllables are separated by an underscore. The process of defining of reaches is explained in the Signposts and the *ModifyTerrain* Wiki.
- **02_Maps** contains a map `templates` subfolder that is copied by *River Architect*'s modules for processed conditions (e.g., `02_Maps/CONDITION/`) for Mapping of Rasters and Shapefiles. The following templates and folders may be modified:
  - `symbology\` contains a standard symbology layer file (`.lyrx`) for partially logarithmic lifespan classification
  - `river_template.aprx` is the standard *ArcGIS Pro* project file, where developers recommend to adapt background image (layer) connections.
- **Module (folder): `RiverArchitect/StrandingRisk`** for *stranding risk analyses of restoration efforts*.
  - `Output` directory where Stranding Risk module outputs are saved.
  - `connect_gui.py` is a standalone script that creates the graphical user interface (GUI) for running the *Stranding Risk* module.
  - `cConnectivityAnalysis.py` provides methods for calculating stranding risk metrics.
  - `cGraph.py` contains graphic representation routines for habitat connectivity analyses.
- **Module (folder): `RiverArchitect/GetStarted`** prepare input *Conditions* from scratch or existing conditions.
  - `.cache` folder occurs temporarily when the module is executed.
  - `.templates` folder should not be modified and contains the welcome screen imagery.
  - `cConditionCreator.py` contains a managing class for creating and populating *Conditions*.
  - `cDetrendedDEM.py` provides routines for generating detrended DEM Rasters.
  - `cMakeInp.py` provides routines for creating input files (*.inp*) for lifespan mapping.
  - `cMorphUnit.py` provides routines for calculating instream morphological units (Wyrick and Pasternack 2014).
  - `cWaterLevel.py` provides routines for calculating depth to water table Rasters.
  - `fSubCondition.py` contains routines for creating a spatial subset of an existing *Condition*.
  - `popup_ ... .py` contain popup window classes guiding through the creation and population of *Conditions*.
  - `welcome_gui.py` contains the `Get Started` tab source code.
- **Module (folder): `RiverArchitect/LifespanDesign`** for Lifespan and Design analyses of restoration features.
  - `Output` folder with sub-folders for `Rasters` from individual 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.
  - `cLifespanDesignAnalysis.py` contains a GIS-based functional core for processing `Raster` files.
  - `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.
  - `feature_analysis.py` coordinates class and function calls.
  - `lifespan_design_gui.py` is a standalone script that creates the graphical user interface (GUI) for running the *LifespanDesign* module.
- **Module (folder): `RiverArchitect/MaxLifespan`** for maximum lifespan assessment of feature groups (concurring features)
  - `Output` folder with sub-folders for `Layouts`, `Maps` and `Rasters` from individual 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 river restoration feature data in the *LifespanDesign* module.
  - `cReadActionInput.py` contains functions for reading `*.inp` files from the `.templates` folder.
- **Module (folder): `RiverArchitect/ModifyTerrain`** performs half-automated terrain modifications
  - `Input` folder containing optional modified DEMs for volume difference assessment.
  - `Output` folder with sub-folders `Rasters` from individual module runs.
  - `RiverBuilder` folder contains original *River Builder* code (`riverbuilder.r`) and input files, as well as help messages.
  - `.cache` folder occurs when the module is executed.
  - `.templates` folder contains the *reach*-defining workbook `computation_extents.xlsx`.
  - `cModifyTerrain.py` contains GIS-based functional core for modifying DEM `Rasters` with threshold values (grading and widening).
  - `cRiverBuilder.py` contains routines for launching *River Builder*.
  - `cRiverBuilderConstruct.py` contains routines for creating input files for *River Builder*.
  - `modify_terrain_gui.py` is a standalone script that creates the graphical user interface (GUI) for running the *ModifyTerrain* module.
  - `sub_gui_rb.py` contains the GUI for creating *River Builder* input files.
- **Module (folder): `RiverArchitect/SHArC`** creates Habitat Suitability Index Rasters/maps and quantifies annually usable habitat area for target fish species and user-defined ranges of discharges.
  - `CHSI` contains subfolders with composite habitat suitability index Rasters for pre- and post-project `conditions`.

   - HSI contains subfolders with habitat suitability index Rasters for pre- and post-project conditions.
   - SHArea contains result workbooks with SHArea 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 annually usable habitat area and the definition of fish species, lifestages, and associated habitat suitability curves.
   - cHSI.py contains classes to calculate composite habitat suitability Rasters, hydraulic habitat suitability Rasters and interpolating the annual flow duration of considered discharges.
   - sharc_gui.py contains the class of this module.
   - sub_gui_covhhsi.py opens a new GUI window to create cover habitat suitability Rasters.
   - sub_gui_hhsi.py opens a new GUI window to create hydraulic habitat suitability Rasters and determine associated annual flow duration.

- **Module (folder): RiverArchitect/ProjectMaker** applies on results from *MaxLifespan* and *SHArC*, as well as manual inputs to calculate project cost-benefit metrics.
   - .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.
   - cSHArC.py applies *SHArC* results, in particular *cHSI* Rasters for calculating SHArea in the project area.
   - project_maker_gui.py contains the class of this module.
   - s20_plantings_delineation.py applies *LifespanDesign* and *MaxLifespan* for assessing the most suitable vegetation plantings within the project area.
   - s21_plantings_stabilization.py applies *LifespanDesign* and *MaxLifespan* outputs as well as user-defined input parameters for mapping nature-based engineering features required to stabilize vulnerable vegetation plantings.
   - s30_terrain_stabilization.py applies *LifespanDesign* and *MaxLifespan* outputs to stabilize terraforming features.
   - s40_compare_wua.py applies on *SHArC* *cHSI* Rasters used in cSHArC.py for assessing the annually usable habitat area for a target fish species and lifestage within the project area.
   - LAUNCH_Windows_x64.bat is a batchfile that runs project_maker_gui.py on Windows x64.

- **Module (folder):** ***RiverArchitect/VolumeAssessment*** *calculates excavation / fill volumes of terraforming features.*
   - Output folder for calculation results.
   - .cache folder occurs when the module is executed.
   - .templates folder contains volume_template.xlsx.
   - cVolumeAssessment.py contains GIS-based functional core for calculating volumes using an ArcGIS “3D” extension.
   - volume_gui.py is a standalone script that creates the graphical user interface (GUI) for running the VolumeAssessment module

- **Folder: Tools** applies on results from *MaxLifespan* and *SHArC*, as well as manual inputs to calculate project cost-benefit metrics.
   - .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.
- `cHydraulic.py` contains a 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.
- `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_annual_flow_duration.py` - *deprecated (will be removed)* creates flow duration curves (annual averages) for the assessment of SHArea (deprecated). *A more sophisticated version of this function is integrated in the SHArC module with options to limit flow duration assessments to fish-lifestage seasons.*
- `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.
- `run_morphology_designer.bat` is a batchfiles that runs `morphology_designer.py`.

2.2 Program environment setup and batchfile modification

The package is designed for an *ArcGIS Pro*'s Python3 conda environment. Before launching the *River Architect* package for the first time, the batchfiles may require adaptions for the system environment. On **Windows** only (Linux is not yet available because of `arcpy` issues - we are working on it), set the batch file environment as follows:

1. Right-click on `Start_River_Architect.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:* `"%PROGRAMFILES%\ArcGIS\Pro\bin\Python\Scripts\propy"` *Note:* Future versions of *River Architect*'s will require `gdal` and `rasterio`. Therefore, developers recommend to create a new conda environment and to install these packages to that new environment:
   - Open "ArcGIS Pro" and click on the *Project* pane in the upper left corner
   - Click on the *Python* tab
   - Click on the *Manage Environments* button and on *Clone default* (or manually copy the default environment) to, for example, `C:\Python\envs\arcgispro-py3-cust`
   - When cloning was successful, restart *ArcGIS Pro* and go back to the *Project* > *Python* tab and click on *Add Packages*
   - Search for, and install `gdal` (`lib-gdal` also works) and `rasterio`
   - For starting *River Architect* from an *IDE*, set the new conda environment as interpreter (e.g., `C:\Python\envs\arcgispro-py3-cust\python.exe`). The new environment may also be used in the batchfiles.
   - The ArcGIS Pro website provides more information on how to install *Python* packages.

3. Save `LAUNCH_River_ArchitectWINx64.bat` and close *Texteditor*.
4. 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`*.*

There is no standard easy way to import ArcGIS' `arcpy` package in *Python* running on UNIX platforms (Apple or Linux). Future versions of *River Architect* aim at using other packages for geodata processing, which will also run on UNIX platforms (we are currently experimenting with `gdal` and `qgis.core`).After editing the batch files, launch *River Architect* by double-clicking on `Start_River_Architect.bat`.

2.3 Requirements and dependencies

2.3.1 Python packages

The execution of *River Architect* requires the following packages, which are part of the standard *ArcGIS Pro - Python3* distribution:

- `arcpy`
- `argparse`
- `glob`
- `logging`
- `openpyxl`
- `os`
- `shutil`
- `subprocess` (not mandatory, also works without this package)
- `tkinter`

Additional packages:

- `rpy2` and the R language: The *Morphology\ModifyTerrain* module requires an installation of the R language to automate the production of river topographic data via the *RiverBuilder* R package. Note: environmental variables `%R_HOME%` and `%R_USER%` must also be properly defined for Python to access the user installation of R.

River Architect incorporates a modified version of `openpyxl`, but a more proper way is installing `openpyxl` with *ArcGIS Pro*'s package manager. Future versions may additionally require the `gdal` and `rasterio` packages (see *above* instructions).Furthermore, *River Architect* requires *ArcGIS Pro*'s "Spatial Analyst" and "3D" (*Volume Assessment* only) extensions. Another version of *River Architect* based on open-source geodata processing packages is planned for the future.

Any folder beginning with a "." for example `.cache` or `.idea` 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 execution, the applied modules have created their output folders, which are indicated in the command prompt.

2.3.2 Other software and dependencies

A workbook editing software such as *Excel* or *LibreOffice* is required for modifications of user-defined databases (`.xlsx` files).

For the visualization of geodata (`.shp` and `.tif` files), and mapping with project files (`.aprx`), an installation of *ArcGIS Pro* is required. Open-source alternatives are *QGIS* or SAGA (*Please note: There are more GIS alternatives out there.*).

2.4 Logfiles

Logfiles `logfile.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 the Troubleshooting Wiki with descriptions of problem sources (cause) and solutions (remedy).

CHAPTER

THREE

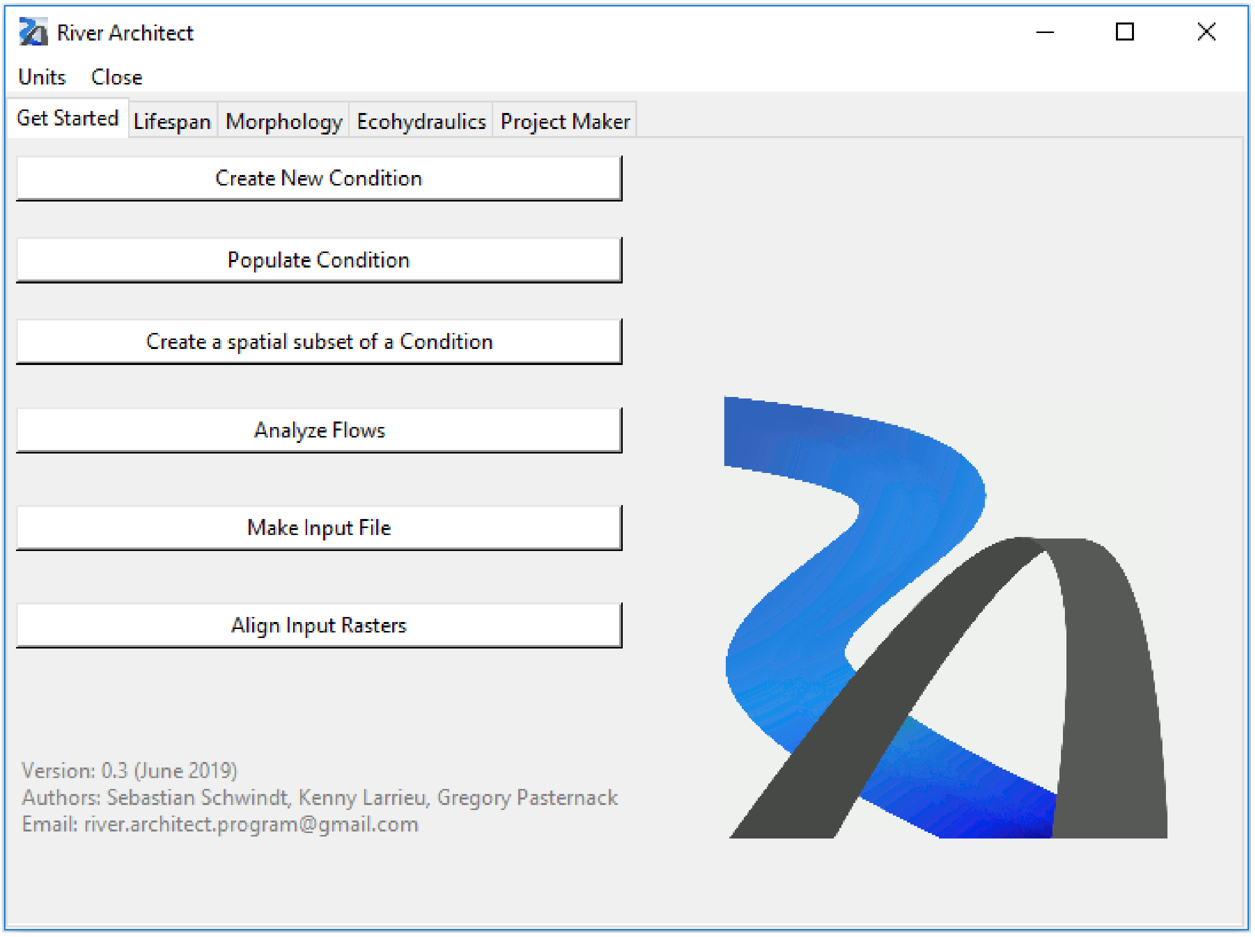
GET STARTED AND SIGNPOSTS

CHAPTER

FOUR

GET STARTED

River Architect's first tab invites the user for the creation of *Conditions*.



Version: 0.3 (June 2019)
Authors: Sebastian Schwindt, Kenny Larrieu, Gregory Pasternack
Email: river.architect.program@gmail.com

A *Condition* is a folder filled with Rasters that represent a temporal snapshot (situation) of a river. *Conditions* are stored in `RiverArchitect/01_Conditions/`. For example, if the goal is to assess feature lifespans based on the situation in the year 2008, the condition folder name may be `2008` and the corresponding folder is `RiverArchitect/01_Conditions/2008/`. The *Condition* name may NOT include any SPACE character and good practice is that the first 4 characters represent a 4-digit year. All other modules build on the data provided in `RiverArchitect/01_Conditions/` and the modules create output folders beginning with the defined *Condition* name and ending with specifiers such as feature layer, reach, or fish (see bottom of HHSI output descriptions) information.

The *Get Started*-tab buttons invite the user to create conditions from scratch, populate (new) conditions, and make spatial subsets. Every button opens up a new window with the following options:

- Create New *Condition*: Create a new *Condition* from scratch
- Populate *Condition*: Add Morphological Unit, Depth to water table, and Detrended DEM Rasters to an existing (or newly created) *Condition*
- Create a spatial subset of a *Condition*: Define a boundary Raster that delineates a spatial frame for creating a subset of an existing *Condition* (useful for comparing project alternatives at different sites).

4.1 Create New Condition

The creation of a new *Condition* requires that a 2D hydrodynamic model was previously run to obtain spatially explicit flow depth and velocity data. Moreover, grain size data (spatially explicit)and a digital terrain elevation model need to be available. A new popup window inquires following inputs for generating a new *Condition* from scratch:

- A name for the new *Condition*
- A folder containing flow velocity Rasters (preferably use *GeoTFF* `.tif` Rasters) corresponding to multiple discharges (*read more on discharge definitions*). A *Raster string* may be defined to select only Rasters that contain certain letters in their name from the defined folder (e.g., enter `u`).
- A folder containing flow depth Rasters (preferably use *GeoTFF* `.tif` Rasters) corresponding to multiple discharges (*read more on discharge definitions*). A *Raster string* may be defined to select only Rasters that contain certain letters in their name from the defined folder (e.g., enter `h`).
- A *DEM* (Digital Elevation Model) or *DTM* (Digital Terrain elevation Model) of the terrain covered by the flow depth and velocity Rasters.
- A *Grain Size* Raster (U.S. customary: in feet or SI metric: in meters).
- An optional folder containing velocity angle Rasters (preferably use *GeoTFF* `.tif` Rasters) corresponding to multiple discharges (*read more on discharge definitions*). Note that velocity angles are defined in degrees from North, i.e. North=0, East=90, West=-90, South=180/-180. A *Raster string* may be defined to select only Rasters that contain certain letters in their name from the defined folder (e.g., enter `va`).
- An optional *Scour* Raster (U.S. customary: in feet or SI metric: in meters) containing annual scour rates, which may result from terrain change detection analyses (Pasternack and Wyrick 2017) or hydro-morphodynamic modeling (tricky).
- An optional *Fill* Raster (U.S. customary: in feet or SI metric: in meters) containing annual fill rates, which may result from terrain change detection analyses (Pasternack and Wyrick 2017) or hydro-morphodynamic modeling (tricky).
- A background Raster, which can optionally be used to *Align Input Rasters*.

Note that the optional inputs are highly recommended to make the subsequent analyses robust.

Once the input is defined, clicking on the `CREATE CONDITION` button will create the new *Condition* as `RiverArchitect/01_Conditions/`*NEW_CONDITION* with the following contents:

- `back.tif` is a background Raster that may be used for limiting lifespan analyses (LifespanDesign) extents and all subsequent analyses. Moreover, the background Raster enables consistent mapping.
- `dem.tif` is a DEM Raster indirectly required by the LifespanDesign, Modify Terrain, Stranding Risk, ProjectMaker and Max Lifespan modules.
- `dmean.tif` is a grain size Raster indirectly required by the LifespanDesign, SHArC, Stranding Risk, ProjectMaker and Max Lifespan modules.
- `fill.tif` is topographic change Raster indicating annual sediment deposition rates, which are required by the LifespanDesign, and indirectly, the Max Lifespan and ProjectMaker modules.

- `hQQQQQQ.tif` are flow depth rasters required by the LifespanDesign, SHArC, Stranding Risk, and (indirectly) the ProjectMaker and Max Lifespan modules. *Read more about file name conventions.*
- `uQQQQQQ.tif` are flow velocity rasters required by the LifespanDesign, SHArC, Stranding Risk, and (indirectly) the ProjectMaker and Max Lifespan modules. *Read more about file name conventions.*
- `scour.tif` is topographic change Raster indicating annual terrain erosion rates, which are required by the LifespanDesign, and indirectly, the Max Lifespan and ProjectMaker modules.
- `vaQQQQQQ.tif` is flow velocity direction rasters required by the *Stranding Risk* module. *Read more about file name conventions.*

The flow depth and velocity Rasters may require manual renaming to adapt to these Raster name conventions. A *River Architect Tools* script facilitates renaming multiple file names. Subsequently, populating the created *Condition* is strongly recommended

4.2 Populate Condition

Condition - wise Morphological Unit, Depth to water table, and Detrended DEM Rasters add consistency to the analyses of all modules. The `Populate Condition` popup window invites the user to define a *Condition* to be populated, and subsequently to create the following Rasters:

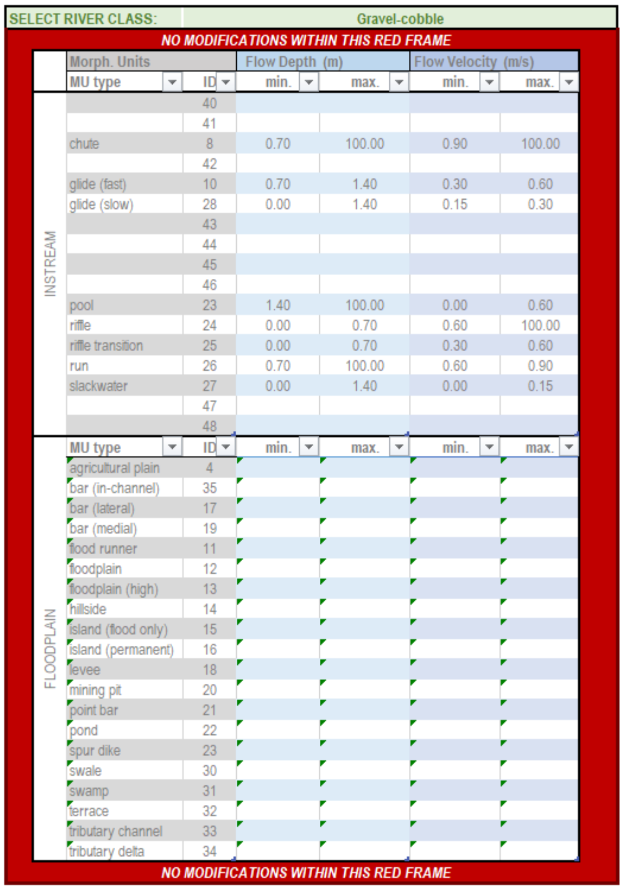
- Morphological Unit,
- Depth to water table, and
- Detrended DEM.

4.2.1 Make Morphological Unit Rasters

Instream morphological unit Rasters according to Wyrick and Pasternack (2014) enable the correct allocation of river design features as defined in the thresholds workbook. For this purpose, the following inputs are needed:

- A flow velocity raster (use baseflow in line with scientific literature);
- A flow depth raster (use baseflow in line with scientific literature).

The delineation of morphological units as a function of flow depth and velocity can be modified in the `morphological_units.xlsx` workbook (`RiverArchitect/.site_packages/templates/` folder). A click on the `Populate Condition` GUI's `View/change MU definitions` opens this workbook.

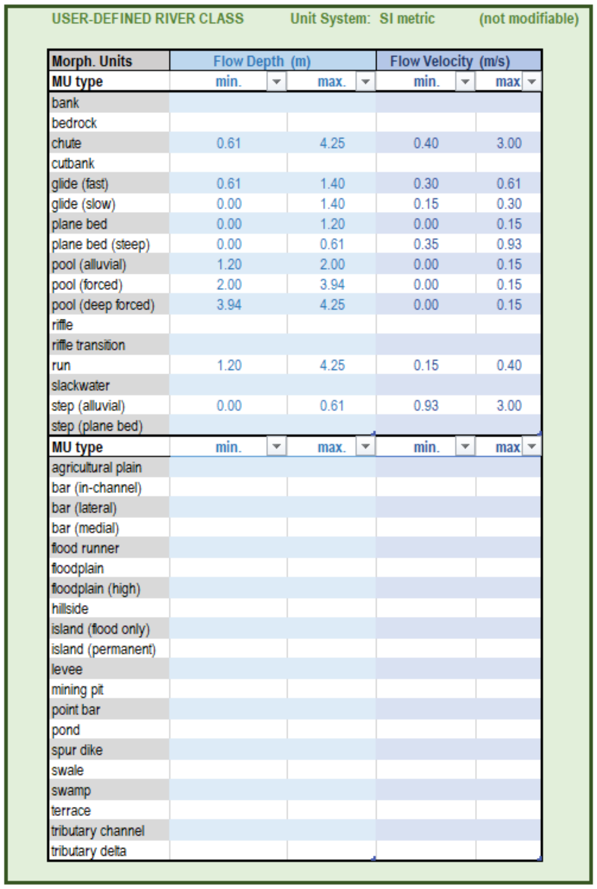


SELECT RIVER CLASS: Gravel-cobble

NO MODIFICATIONS WITHIN THIS RED FRAME

	Morph. Units		Flow Depth (m)		Flow Velocity (m/s)	
	MU type	ID	min.	max.	min.	max.
INSTREAM		40				
		41				
	chute	8	0.70	100.00	0.90	100.00
		42				
	glide (fast)	10	0.70	1.40	0.30	0.60
	glide (slow)	28	0.00	1.40	0.15	0.30
		43				
		44				
		45				
		46				
	pool	23	1.40	100.00	0.00	0.60
	riffle	24	0.00	0.70	0.60	100.00
	riffle transition	25	0.00	0.70	0.30	0.60
	run	26	0.70	100.00	0.60	0.90
	slackwater	27	0.00	1.40	0.00	0.15
		47				
		48				
	MU type	ID	min.	max.	min.	max.
FLOODPLAIN	agricultural plain	4				
	bar (in-channel)	35				
	bar (lateral)	17				
	bar (medial)	19				
	flood runner	11				
	floodplain	12				
	floodplain (high)	13				
	hillside	14				
	island (flood only)	15				
	island (permanent)	16				
	levee	18				
	mining pit	20				
	point bar	21				
	pond	22				
	spur dike	23				
	swale	30				
	swamp	31				
	terrace	32				
	tributary channel	33				
	tributary delta	34				

NO MODIFICATIONS WITHIN THIS RED FRAME



USER-DEFINED RIVER CLASS Unit System: SI metric (not modifiable)

Morph. Units	Flow Depth (m)		Flow Velocity (m/s)	
MU type	min.	max.	min.	max
bank				
bedrock				
chute	0.61	4.25	0.40	3.00
cutbank				
glide (fast)	0.61	1.40	0.30	0.61
glide (slow)	0.00	1.40	0.15	0.30
plane bed	0.00	1.20	0.00	0.15
plane bed (steep)	0.00	0.61	0.35	0.93
pool (alluvial)	1.20	2.00	0.00	0.15
pool (forced)	2.00	3.94	0.00	0.15
pool (deep forced)	3.94	4.25	0.00	0.15
riffle				
riffle transition				
run	1.20	4.25	0.15	0.40
slackwater				
step (alluvial)	0.00	0.61	0.93	3.00
step (plane bed)				
MU type	min.	max.	min.	max
agricultural plain				
bar (in-channel)				
bar (lateral)				
bar (medial)				
flood runner				
floodplain				
floodplain (high)				
hillside				
island (flood only)				
island (permanent)				
levee				
mining pit				
point bar				
pond				
spur dike				
swale				
swamp				
terrace				
tributary channel				
tributary delta				

For making changes in the workbook, choose either one of the pre-defined river classes (`Mountain river` with large roughness elements, `gravel-cobble` bed, or `cobble-boulder` bed) or select `User Defined` from the drop-down menu in cell `E2`. If `User Defined` is selected, morphological units can be defined as a function of upper and lower limits of flow velocity and depth in cells `N6:Q22` (instream) and/or `N24:Q43` (floodplain). Morphological unit names can be modified in cells `M6:M22` (instream) and/or `M24:M43` (floodplain). Please note:

- Do not modify anything outside the green frames, in particular, do not make changes within the red frame (cells `B3:J44` and the `.templates` sheet).
- Use **metric units only**; for the conversion of metric units for their application to Rasters in U.S. customary units, select `Units: U.S. customary` (default) from the `Populate Condition` window. The currently selected unit system is shown at the bottom of the `Populate Condition` window.
- Instream and floodplain units are currently equally applied (future versions of *River Architect* aim at a more precise approach to delineate floodplain morphological units).
- The maximum length of morphological units names (`M6:M22` and/or `M24:M43`) is 50 characters per cell.
- *River Architect* calculates a Raster for each morphological unit defined using the following `arcpy.sa` expression: `Con(h > 0, Con(((u >= u_lower) & (u < u_upper) & (h >= h_lower) & (h < h_upper)), mu_id))`. The final Raster is created by the superimposition of all applicable morphological raster and using the maximum `mu_id` value (pre-defined in column `E`). with `arcpy.sa` cell statistics: `CellStatistics(mu_id_raster_list, "MAXIMUM", "DATA")`.
- The results are
  - the morphological unit's `ID` Raster saved as `RiverArchitect/01_Conditions/CONDITION/mu.tif`; and

  - the morphological unit's name Raster (uses an intermediated Raster to point conversion) saved as `RiverArchitect/01_Conditions/CONDITION/mu_str.tif`.

4.2.2 Make Depth to Water Table Rasters

The depth to the water table is primarily required for identifying relevant regions for target indigenous plant species. For this purpose, the following input Rasters are required.

- A terrain DEM (or DTM) is automatically assigned from the selected *Condition* folder.
- A low-level flow depth Raster (in arid regions) based on the assumption that the groundwater table in the vicinity of the river corresponds to at least this water level, which marks the moment of highest water stress for plants.

RiverArchitect estimates the depth to water table by interpolating the given low flow water surface across the DEM extent. There are three options for the interpolation method used to calculate this surface:

- `IDW`: Inverse distance weighted interpolation. Each null cell in the low flow depth raster is assigned a weighted average of its 12 nearest neighbors. Each weight is inversely proportional to the second power of the distance between the interpolated cell and its neighbor. Thus, the closest neighbors to an interpolated cell receive the largest weights.
- `Kriging`: Ordinary Kriging interpolation. This method also uses a weighted average of the 12 nearest neighbors, but weights are calculated using a semi-variogram, which describes the variance of the input data set as a function of the distance between points. A spherical functional form is also assumed for the fitted semi-variogram. Kriging is the most accurate interpolation method if certain assumptions are met regarding normality and stationarity of error terms. However, it is also the most computationally expensive method.
- `Nearest Neighbor`: The simplest method, which sets the water surface elevation of each null cell to that of the nearest neighboring cell.

All interpolated rasters are saved with a corresponding `.info.txt` file which records the interpolation method and input rasters used in its creation.

4.2.3 Make Detrended DEM Rasters

Automation of grading or the relevance of widening and berm setbacks build on the relative elevation of the terrain over the river water surface elevation. For this purpose, the following input Rasters are required.

- A terrain DEM (or DTM) is automatically assigned from the selected *Condition* folder.
- A flow depth Raster (in arid regions) marks the level for relative elevations. Designers may have different reasons for choosing the relevant flow depth Raster (low flows for habitat enhancement or high flows for flood protection), and therefore, no recommendation is made here.

Please note that step-like artifacts may occur in the detrended DEM. The steps result from variations in the Thalweg elevation of the DEM that attenuate with the selection of flow depth Rasters of higher discharges. Users can decide to select a low-flow depth Raster to be on the safe side for delineating vegetation plantings or a high-flow depth Raster to reduce step-like artifacts in the detrended DEM.

4.3 Create a spatial subset of a Condition

The creation of a spatial subset of a *Condition* requires a Boundary shapefile or Raster (if this is a GRID Raster, select the corresponding .aux.xml file). The boundary file needs to contain On-values (Integer 1) and Off-values (Integer 0) values only as specified in the Project Area Polygon preparation.

If a shapefile is selected:

- River Architect automatically uses the `gridcode` Field in the Shapefile's Attribute Table.
- If the `gridcode` Field cannot be found, the third Field or the `FID` Field is used. (in that order). Note that the enforced usage of the `FID` Field may cause errors later on.
- Ensure that the `gridcode` / third Field contains On-Off Integers only, where 0=Outside and 1=Inside boundary.
- **Restart River Architect to create multiple subsets.** This issue is related to `arcpy`, which will not release the new datasets unless *River Architect* is closed (otherwise there will be an error message referring to registering datasets) We are developing workarounds.

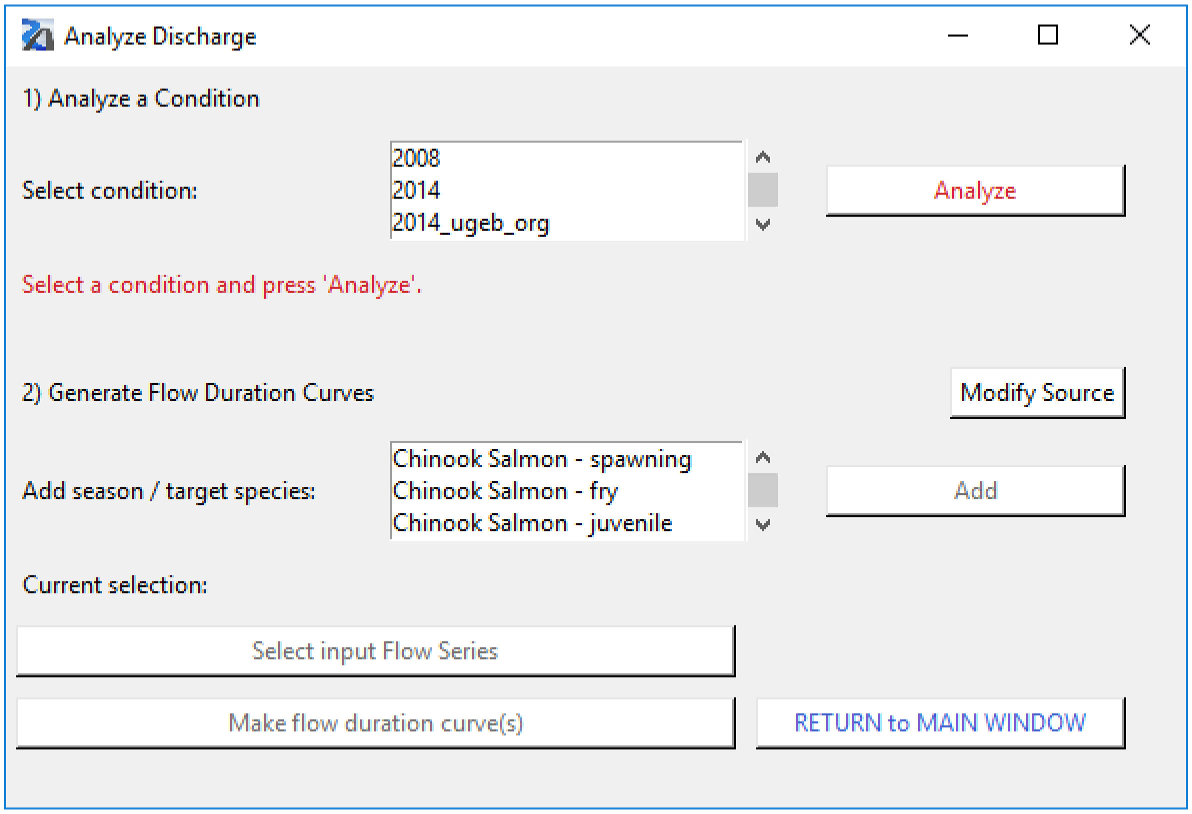
If a Raster is selected:

- Ensure that the Raster contains On-Off Integers only, where 0=Outside and 1=Inside boundary.

The boundary files are of particular interest within the ProjectMaker and SHArC modules.

4.4 Analyze Flows

The sustainability (lifespan) and ecohydraulic analyses require hydraulic data related to discharges (flows) and the return period. The *Analyze Discharge* pop-up window guides through the creation of flow-metadata files that link hydraulic Raster names with flows and return periods for a *Condition*. *Analyze Discharge* looks for flow depth and velocity Rasters in a selected *Condition*, extracts the flow quantity, and creates a template workbook in the *Condition* folder. For this purpose, hydraulic (flow depth and velocity) Rasters must be named according to the *Geofile name convetions* (i.e., flow depth Rasters = "hQQQQQQ.tif" and flow velocity Rasters = "uQQQQQQ.tif").



Analyze Discharge

1) Analyze a Condition

Select condition: 2008 / 2014 / 2014_ugeb_org

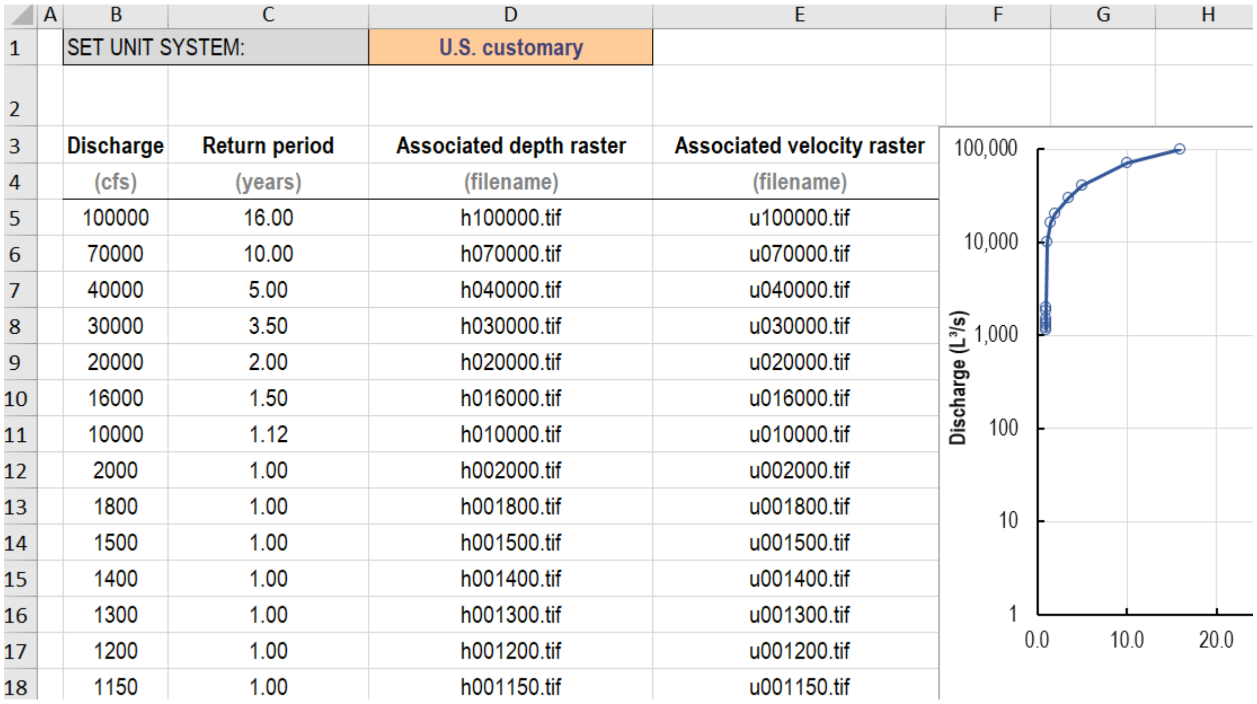
Select a condition and press 'Analyze'.

2) Generate Flow Duration Curves

Add season / target species: Chinook Salmon - spawning / Chinook Salmon - fry / Chinook Salmon - juvenile

Current selection:

Start with selecting a *Condition* from the upper listbox and click the `Analyze` button. This creates a workbook called `RiverArchitect/01_Conditions/flow_definitions.xlsx`, which automatically opens up and asks for the definitions of flow return periods in years. Frequent flows with a return period of one year or less should be assigned `1.0` (column `C`).



	B	C	D	E
1	SET UNIT SYSTEM:		U.S. customary	
2				
3	Discharge	Return period	Associated depth raster	Associated velocity raster
4	(cfs)	(years)	(filename)	(filename)
5	100000	16.00	h100000.tif	u100000.tif
6	70000	10.00	h070000.tif	u070000.tif
7	40000	5.00	h040000.tif	u040000.tif
8	30000	3.50	h030000.tif	u030000.tif
9	20000	2.00	h020000.tif	u020000.tif
10	16000	1.50	h016000.tif	u016000.tif
11	10000	1.12	h010000.tif	u010000.tif
12	2000	1.00	h002000.tif	u002000.tif
13	1800	1.00	h001800.tif	u001800.tif
14	1500	1.00	h001500.tif	u001500.tif
15	1400	1.00	h001400.tif	u001400.tif
16	1300	1.00	h001300.tif	u001300.tif
17	1200	1.00	h001200.tif	u001200.tif
18	1150	1.00	h001150.tif	u001150.tif

Flow duration curves are required for ecohydraulic analyses and can be generated for specific Physical Habitats preferred by target fish species-lifestages. Select at least one *Fish Species - Lifestage* from the lower listbox and use the `Add` button (to add multiple *Fish Species - Lifestage*s, select-add one-by-one). A click on the `Modify Source` button opens the `Fish.xlsx` workbook that contains *Fish Species - Lifestage* definitions. At this point, on particular fish names and seasons start/end dates may be modified. Modifications of *Lifestages* should be avoided. For more details, refer to the SHArC Wiki pages. Before a *Fish Species - Lifestage* flow duration curve can be generated, ensure to `Select input Flow Series` (workbook) with the following characteristics:

- The file ending must be `.xlsx`
- Column `A` must contain dates, where
  - row `1` is the header (i.e., `A1 = "Date"`),
  - row `2` indicates the date units (i.e., `A2 = "(DD-MM-YY)"`), and
  - date series must be entered from row `3` onward (i.e., dates in the format `A3 = 5-Sep-03` or `A4 = 9/5/2003`).
- Column `B` must contain mean daily flows, where
  - row `1` is the header (i.e., `B1 = "Mean daily"`),
  - row `2` indicates the date units (i.e., `B2 = "(CFS)"` or `B2 = "(CMS)"), and
  - mean daily flows must be entered from row `3` onward (i.e., dates in the format `B3 = 59.3` or `B4 = 79`).
- For later application, store the flow series workbook in `RiverArchitect/00_Flows/InputFlowSeries/` (default folder where *River Architect* looks for flow series).

An example workbook is provided with `RiverArchitect/00_Flows/InputFlowSeries/flow_series_example_data.xlsx`.

Click on `Make flow duration curve(s)` (plural applies if multiple *Fish Species - Lifestage*s are selected) to generate an Physical Habitat workbook for target *Fish Species - Lifestage*s. The workbook containing the last *Fish Species - Lifestage* flow duration curve in the selected list will open up automatically when the flow duration curve generation finishes without error messages. All generated workbooks will be saved as `RiverArchitect/00_Flows/CONDITION/flow_duration_FILI.xlsx`, where `FILI` denotes the first two letters of the selected fish species and lifestage. The workbooks contain two tabs that link all observed mean daily flows from the target *Fish Species - Lifestage*'s season (tab 1) with the available 2D model data (tab 2). The result is a flow duration curve that provides a measure of how well the 2D model data may represent the relevant discharges for a *Fish Species - Lifestage*.

4.5 Generate Input File(s) (.inp)

Lifespan mapping uses input files (.inp) to identify relevant Rasters and (flood) return periods. Given that `Analyze Flow` was previously executed for a *Condition*, an input file can be generated with the `Make Input File` tool.

The resulting `input_definitions.inp` is stored in the directory `RiverArchitect/01_Conditions/CONDITION/`. `input_definitions.inp` contains information about lifespan duration and Raster names, which link to Rasters containing spatial information as described in the Parameters Wiki page. The order of definitions and lines must not be changed to ensure the proper functioning of the module. Enter or change the 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). Verify that every `input_definitions.inp` created contains the following definitions (line by line):

The LifespanDesign 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 results are produced for a feature where the information is insufficient for the analysis. The required information for every feature corresponds to the definitions in the input file.

4.6 Map extent input definition files

The file `RiverArchitect/LifespanDesign/.templates/mapping.inp` defines map center points, extents (*dx* and *dy* in ft or m) and scales (scale has no effect currently). The extent of the map determines the map scale, where the corresponding *dx* and *dy* values define the map width and height in ft or m, respectively. The layout templates (in the project file `RiverArchitect/02_Maps/templates/river_template.aprx`) 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. Good practice for changing the map layouts starts with importing the `determine_extents.mxd` layout from `RiverArchitect/ModifyTerrain/templates/` into a map project (`.aprx`). Zoom to new focus point using, for example, using *ArcGIS Pro*'s `Go To XY` function 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 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 `.aprx` file's map layout templates or in `mapping.inp`.

4.7 Align Input Rasters

In order to ensure robustness and accuracy of analyses, it is important that input rasters share a common alignment, cell size, and coordinate system. Input rasters can be aligned in the following ways:

- During creation of a new *Condition*, align input rasters by using a background raster and selecting the "Use to align input rasters" checkbox.
- For an existing condition, the tool from the GetStarted menu allows selection of an alignment raster.

The alignment routine reprojects and/or resamples the input raster data to match the following attributes from the input alignment raster:

- ArcGIS SpatialRefernce (i.e. reference coordinate system)
- lower left corner of raster, modulo cell size
- cell size

Caution should be taken when using the built-in alignment routine, as input data may have already been resampled (e.g. from a mesh, TIN, or point features) and repeated resampling of data may create artifacts in the resultant data. Additionally, other routines in River Architect assume that water surface elevation can be calculated by summation of the DEM and depth rasters, which may not hold true if these data have been subject to different resampling schemes.

CHAPTER

FIVE

GEOFILE (RASTER) CONVENTIONS

The input Rasters need to be in **GeoTIFF** (*.tif*) format, notably, a `raster_name.tif` file. *Note that River Architect is designed to also handle Esri's GRID format, but the primary raster file type should be GeoTIFF*. Depth Raster names must start with `h` and velocity Raster names must start with `u`, followed by a six-digit discharge `QQQQQQ`, which is independent of the unit system. For example, a flow depth Raster associated with a discharge of 55 m^3/s needs to be called `h000055.tif` and a velocity Raster associated with a discharge of 11000 m^3/s needs to be called `u011000.tif`. Likewise, a flow depth Raster associated with a discharge of 55 cfs needs to be called `h000055.tif`. The Raster names ignore discharge value digits after the decimal point. Moreover, every flow depth Raster requires a matching velocity Raster and vice versa (e.g., `h000055.tif` requires a Raster called `u000055.tif`). **Note: `back.tif` may be used to limit calculation extents.**

CHAPTER

SIX

MANUAL INPUT DATA PREPARATION

Relevant Raster names for calculation are defined in an input file (`.inp`) of the *LifespanDesign* module (input section see for details and definitions). Please note that *.inp* files for lifespan mapping are different from the input (*.txt*) files required for *River Builder*. Sample data representing a patch of a Californian gravel-cobble bed river in 2100 can be downloaded here. The input file of the sample case is located in `01_Conditions/2100_sample/input_definitions.inp` file. The sample case includes a set of Rasters for flow scenarios corresponding to return periods of < 1.0 (ignored in the input file), 1.0, … , 2.0, 5.0, 10.0, 15.0, 20.0, 30.0, 40.0, and 50.0 years, as well as a couple of annual discharges for habitat assessments. The according flows are defined in `01_Conditions/2100_sample/flow_definitions.xlsx`, with pre-compiled flow duration curves for whetted area (`alhy`), Chinook salmon juveniles (`chju`), fry (`chfr`), and spawning (`chju`) lifestages that are stored in `00_Flows/2100_sample/flow_duration_FILI.xlsx` (see above *flow definitions*).

The below listed Rasters are available in GeoTIFF format in `01_Conditions/2100_sample/` for the sample case `condition` = `2100_sample`. *Italic font* indicates *optional* Rasters, which are, however, recommended to use because they significantly increase the pertinence of lifespan maps; Rasters written in **`CAPITALIZED ROUGE FONT`** font are **`required`** for *River Architect* to work. The Raster names correspond to the above-described naming conventions.

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 water table Raster are (optionally) required.

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.

`NoData` handling: *River Architect* does not consider pixels with `NoData` values and has routines to handle `NoData` during the calculation.

CHAPTER

SEVEN

FEATURE LIFESPAN AND DESIGN ASSESSMENT

CHAPTER

EIGHT

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 (flows) can be associated with flood return periods that determine feature lifespans. The resulting lifespan maps indicate the temporal stability of particular river 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) (open access factsheet-version), the *LifespanDesign* module creates `Rasters`, projects (`aprx`), and maps (`pdf`) 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 log diameter for the stability of wood / LWM).

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

- A *Quick Guide* to the application of the code using GUI with descriptions of required input Rasters and alternative launch options.
- Physical explanations of relevant parameters.
- Explanations of hypotheses in the design of river modification and restoration features.
- Coding conventions with descriptions of extension possibilities.

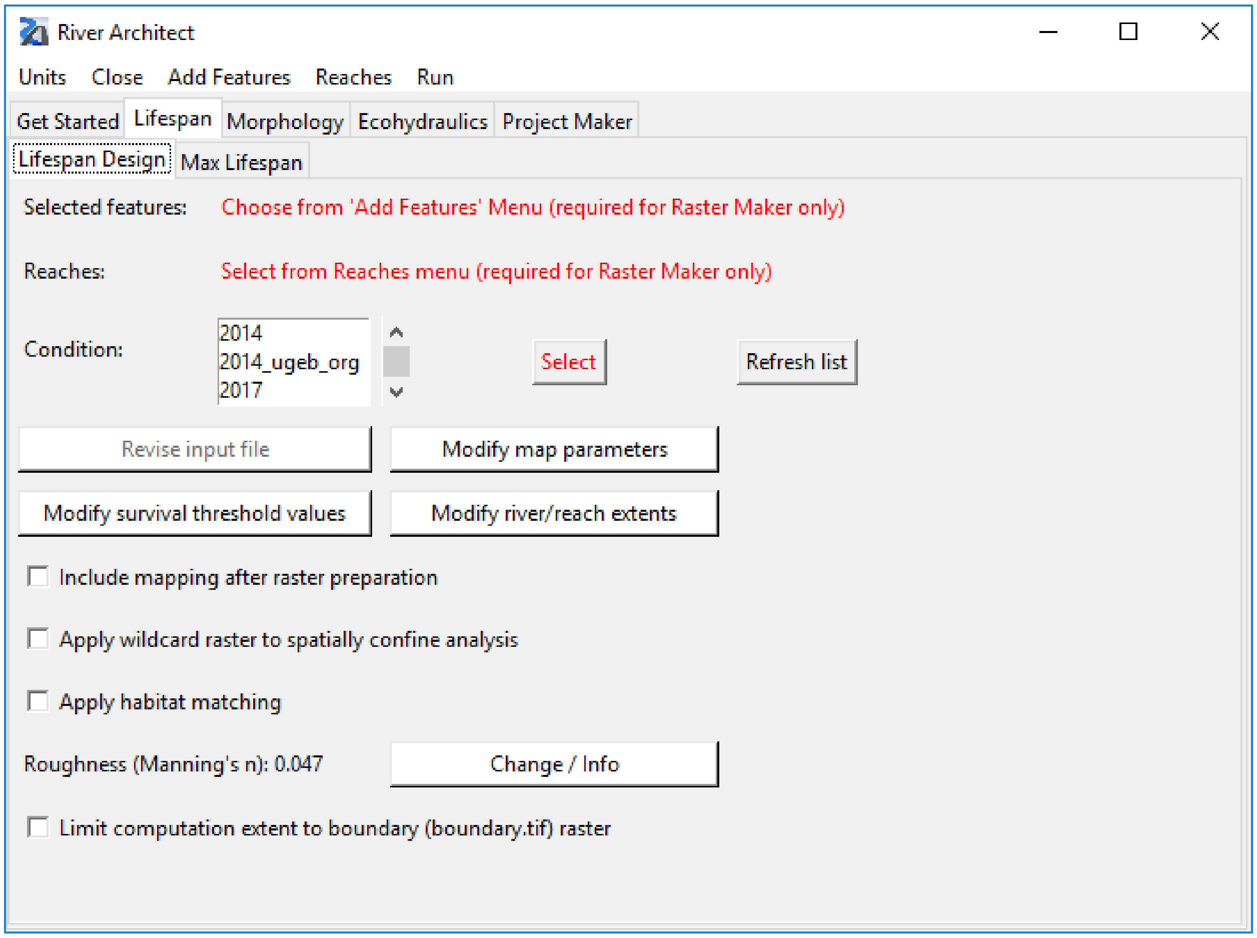
CHAPTER

NINE

QUICK GUIDE TO LIFESPAN AND DESIGN MAPS

9.1 Interface and choice of features

After successful Installation and the creation of at least one *Condition*, lifespan and design maps can be created from the *Lifespan* -> *LifespanDesign* tab.



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

Reaches: Select from Reaches menu (required for Raster Maker only)

Condition: 2014 / 2014_ugeb_org / 2017

☐ Include mapping after raster preparation

☐ Apply wildcard raster to spatially confine analysis

☐ Apply habitat matching

Roughness (Manning's n): 0.047

☐ Limit computation extent to boundary (boundary.tif) raster

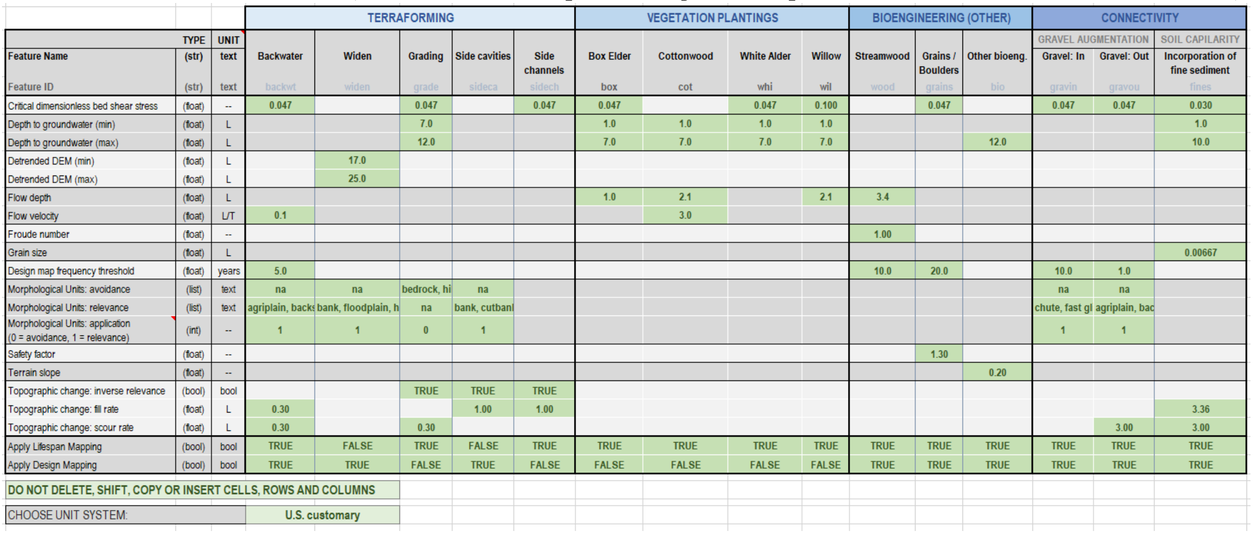
To begin with lifespan/design mapping, click on the drop-down menu "Add Features" and select relevant features. Multiple selection is possible and will extend the "Selected features" list. The *LifespanDesign* module enables the selection of the feature groups of "terraforming" (framework), "vegetation plantings", "other nature-based engineering" and *"connectivity / maintenance"*.

9.2 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 indicate after the # if it requires a single Raster name only (`STRING`) or a list of Rasters (min. two Rasters, `LIST`). *The extension **.tif** is not required for in the Raster names.* The maximum number of hydraulic Rasters is unlimited. The lifespans related to the hydraulic Rasters are also defined in the separate *.inp* file. The Get Started module provides routines for setting up input files that must be stored as `RiverArchitect/01_Conditions/CONDITION/input_definitions.inp` for every *Condition*. Modifications of map extents can be made by clicking on the "Modify map extent" button or (un-)check the `Limit computation extent to background (back.tif) Raster` box.

9.3 Input: The threshold values workbook `threshold_values.xlsx` and calculation hierarchy

The "Modify survival threshold values" button opens the master workbook containing threshold values and survival identifiers (location: `LifespanDesign/.templates/threshold_values.xlsx`).



			TERRAFORMING					VEGETATION PLANTINGS				BIOENGINEERING (OTHER)			CONNECTIVITY		
	TYPE	UNIT													GRAVEL AUGMENTATION		SOIL CAPILARITY
Feature Name	(str)	text	Backwater	Widen	Grading	Side cavities	Side channels	Box Elder	Cottonwood	White Alder	Willow	Streamwood	Grains / Boulders	Other bioeng.	Gravel: In	Gravel: Out	Incorporation of fine sediment
Feature ID	(str)	text	backwt	widen	grade	sideca	sidech	box	cot	whi	wil	wood	grains	bio	gravin	gravou	fines
Critical dimensionless bed shear stress	(float)	--	0.047		0.047		0.047	0.047		0.047	0.100		0.047		0.047	0.047	0.030
Depth to groundwater (min)	(float)	L			7.0			1.0	1.0	1.0	1.0						1.0
Depth to groundwater (max)	(float)	L			12.0			7.0	7.0	7.0	7.0			12.0			10.0
Detrended DEM (min)	(float)	L		17.0													
Detrended DEM (max)	(float)	L		25.0													
Flow depth	(float)	L						1.0	2.1		2.1	3.4					
Flow velocity	(float)	L/T	0.1						3.0								
Froude number	(float)	--										1.00					
Grain size	(float)	L															0.00667
Design map frequency threshold	(float)	years	5.0									10.0	20.0		10.0	1.0	
Morphological Units: avoidance	(list)	text	na	na	bedrock, hi	na									na	na	
Morphological Units: relevance	(list)	text	agriplain, backs	bank, floodplain, h	na	bank, cutbanl									chute, fast gl	agriplain, bac	
Morphological Units: application (0 = avoidance, 1 = relevance)	(int)	--	1	1	0	1									1	1	
Safety factor	(float)	--											1.30				
Terrain slope	(float)	--												0.20			
Topographic change: inverse relevance	(bool)	bool			TRUE	TRUE	TRUE										
Topographic change: fill rate	(float)	L	0.30			1.00	1.00										3.36
Topographic change: scour rate	(float)	L	0.30		0.30											3.00	3.00
Apply Lifespan Mapping	(bool)	bool	TRUE	FALSE	TRUE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
Apply Design Mapping	(bool)	bool	TRUE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE

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

CHOOSE UNIT SYSTEM: U.S. customary

Any threshold value can be changed or defined for any feature, but the workbook structure may not be modified (i.e., cells, columns or rows may not be shifted, moved or deleted). The unit system (U.S. customary or SI metric) in the threshold values spreadsheet are independent of the GUI settings but they need to be coherent with the input Rasters of the *Condition*.

River Architect automatically identifies defined threshold values for selected features. Moreover, *River Architect* automatically applies all defined threshold values to the extent of Rasters available within a *Condition*. The identification and execution of possible analyses follow a strictly hierarchical order. If an analysis cannot be executed because of missing threshold value definitions or Rasters, *River Architect* automatically proceeds to the hierarchically next lower analysis. The analysis hierarchy is defined in order to first created lifespan maps (if row 24 in `threshold_values.xlsx` is set to *TRUE* for a feature), and second to save design maps (if row 25 in `threshold_values.xlsx` is set to *TRUE* for a feature). When defining threshold values in `threshold_values.xlsx` carefully study the following **hierarchy** and parameter application of *River Architect* (read more about **threshold application to features**):

1. **Dimensional hydraulic parameter** analysis:

   - **Flow depth** starting with the lowest discharge to the highest discharge Raster (`hQQQQQQ_QQQ.tif`). A threshold value for the flow depth above which a feature will fail can be defined in row 12 in `threshold_values.xlsx`.

- **Bed shear stress** b calculated as `ras_tb` = {[`uQQQQQQ_QQQ` / (5.75 * Log10(12.2 · `hQQQQQQ_QQQ` / (2 · 2.2 · `dmean`)))]2} where
- A threshold value for mobility according to the bed shear stress \b, cr can be defined in row 6 of `threshold_values.xlsx` (read more for example in Lamb et al. 2008)
- w = water density (1000 kg / m3)
- `uQQQQQQ_QQQ` (m/s or fps), `hQQQQQQ_QQQ` (m or ft), and `d84` = 2.2 · `dmean` (m or ft) are `arcpy.Raster()`s considering that the grain diameter *D84* can be approximated by *D84* = 2.2 · *D50* (Rickenmann and Recking 2011)
- *g* = gravitational acceleration (9.81 m/s2)
- *s* = ratio of sediment grain and water density (2.68)
- Note that the b analysis is **omitted if *SF* is defined**, which enables to run either a b analysis OR a mobile grain analysis.
- **Flow velocity** starting with the lowest discharge to the highest discharge Raster (`uQQQQQQ_QQQ.tif`). A threshold value for the velocity above which a feature will fail can be defined in row 13 in `threshold_values.xlsx`.

2. **Dimensionless hydraulic parameter** analysis:

- **Dimensionless bed shear stress** * calculated as `ras_taux` = {w · [`uQQQQQQ_QQQ` / (5.75 * Log10(12.2 · `hQQQQQQ_QQQ` / (2 · 2.2 · `dmean`)))]2} / [w · *g* (*s* - 1) · `dmean`] where
- A threshold value for mobility according to the critical dimensionless bed shear stress *, cr can be defined in row 7 of `threshold_values.xlsx` (read more for example in Lamb et al. 2008)
- w = water density (1000 kg / m3)
- `uQQQQQQ_QQQ` (m/s or fps), `hQQQQQQ_QQQ` (m or ft), and `d84` = 2.2 · `dmean` (m or ft) are `arcpy.Raster()`s considering that the grain diameter *D84* can be approximated by *D84* = 2.2 · *D50* (Rickenmann and Recking 2011)
- *g* = gravitational acceleration (9.81 m/s2)
- *s* = ratio of sediment grain and water density (2.68)
- Note that the * analysis is **omitted if *SF* is defined**, which enables to run either a * analysis OR a mobile grain analysis.
- **Froude number** *Fr* as `ras_Fr` = `uQQQQQQ_QQQ` / (*g* · `hQQQQQQ_QQQ`)1/2
- A threshold value for mobility according to the Froude number can be defined in row 13 of `threshold_values.xlsx`
- **Mobile grains** (bed mobility) `ras_Dcr`, fine sediment `ras_Dcf` size Rasters as: `ras_Dcx` = = *SF* · `uQQQQQQ_QQQ`2 · *n*2 / [(*s* - 1) · `hQQQQQQ_QQQ`1/3 · *, cr] where
- *, cr is the critical dimensionless bed shear stress (i.e., threshold value) above which sediment is mobile (read more for example in Lamb et al. 2008). *, cr can be defined in row 6 of `threshold_values.xlsx`.
- *n* is Manning's *n* in s/m1/3 (or s/ft1/3 - an internal conversion factor of k = 1.49 applies), which can be changed in the *LifespanDesign* GUI
- *SF* is a dimensionless safety factor that can be defined within *threshold_values.xlsx*; the angular boulders / grains threshold definitions indicate the application of a safety factor of 1.3. The Mobile Grain analysis is **omitted if *SF* is not defined**, which enables to run either a * analysis OR a mobile grain analysis.
- *Note that a Mobile Grain analysis will only work if a safety factor is defined in row 19 of threshold_values.xlsx.*

3. **Topographic change Rasters** `tcd` are applied to limit lifespan Rasters to regions where the `fill` and `scour` threshold values defined in rows 22 and 23 of `threshold_values.xlsx`, respectively, are exceeded. 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 or riprap) are relevant where the topographic change rate (scour) exceeds the angular boulder's threshold value for scour. 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.

4. A **Morphological Unit** Raster as produced with the GetStarted module according to Wyrick and Pasternack 2014 can be used to limit lifespan mapping to morphologically reasonable regions. For example, grading of bedrock units is not reasonable and the default `threshold_values.xlsx` excludes `bedrock` in row 16. *River Architect* enables the application of limit morphological units with an exclusive and an inclusive method:

   - If the exclusive method is chosen (set row 18 in `threshold_values.xlsx` to `0`), *River Architect* will look for morphological units listed in row 16 and it will set pixels with these morphological units to `NoData` in lifespan maps.
   - If the inclusive method is chosen (set row 18 in `threshold_values.xlsx` to `1`), *River Architect* will look for morphological units listed in row 17 and it will set pixels that are not within these morphological units to `NoData` in lifespan maps.
   - Morphological units can be entered as a comma-separated list in row 16 and 17 of `threshold_values.xlsx` corresponding to the settings made during the morphological unit Raster creation (morphological units are defined in `RiverArchitect/.site_packages/templates/morphological_units.xlsx`.

5. **Side channel** delineation work similar to morphological unit delineation and will only be executed if the *Condition* folder contains a Raster called `sidech` (or otherwise named according to the input file in line17). The side channel delineation Raster should only contain Integer values of `1` indicating pixels where a side channel may be drawn. There is no side channel threshold value.

6. The **Depth to the water table** table (see `d2w.tif` Raster creation with the GetStarted module) is particularly important for vegetation plantings or grading (to make terrain suitable for vegetation plantings) features. Threshold values for a lower and an upper limit of the depth to the water table value are defined in row 7 and row 8 of `threshold_values.xlsx`, respectively. *River Architect* will set all areas (pixels) that are not within the value range defined in row 7 and row 8 to `NoData` in lifespan (and design) maps. Leave both rows empty to avoid depth to water table-limited lifespan analyses.

7. The **Detrended DEM** threshold values (see `dem_detrend.tif` Raster creation with the GetStarted module) may be important to limit terraforming features, such as widening/berm setbacks to economically reasonable ranges. Threshold values for a lower and an upper limit of the detrended DEM are defined in row 9 and row 10 of `threshold_values.xlsx`, respectively. *River Architect* will set all areas (pixels) that are not within the value range defined in row 9 and row 10 to `NoData` in lifespan (and design) maps. Leave both rows empty to avoid detrended DEM-limited lifespan analyses.

8. **Nature-based engineering in terms of angular boulders or streamwood are applied to areas (pixels)** where a Frequency threshold (row 15 in `threshold_values.xlsx`) is defined.

9. **Other nature-based engineering** than angular boulders or streamwood is applied where the terrain is steep and as a function of depth-to-groundwater (baseflow level). *River Architect* uses the **Terrain Slope threshold value**s defined in row 20 of `threshold_values.xlsx` to identify areas (pixels) where other nature-based engineering is required. The **Depth to water table** thresholds are used to differentiate between vegetative and mineral nature-based engineering. *River Architect* applies nature-based engineering such as fascines or geotextile between the **(min)** value in row 7 and the **(max)** value in row 8. Regions with at terrain slope above the threshold defined in row 20 and above the maximum Depth to the water table defined in row 8 get mineral nature-based engineering (such as rock paving or riprap) assigned. The differentiation is made because nature-based engineering features may dry out when the water table is too far away (vertically).

10. **Design maps** indicating the stability of grains (sediment) or wood (logs) extracted after the generation of all above-listed lifespan maps (i.e., design mapping is at the bottom of the hierarchy). Thus, design mapping makes sense for features where grain stability is relevant (including angular boulders, backwater zones, grading, gravel/sediment augmentation, and side channel creation) and for the streamwood feature.

    - Set **row 25 to TRUE**
    - For **stable grain size design maps**, make the following definitions for a feature in `threshold_values.xlsx`:
    - critical dimensionless bed shear stress *, cr in row 6
    - frequency threshold in years in row 15 to define the minimum expected lifespan that is required; *River Architect* uses hydraulic parameter rasters (`hQQQQQQ_QQQ` and `uQQQQQQ_QQQ`) corresponding to the return period defined within the GetStarted module (read more on the calculation of stable grain sizes).
    - optional: set a required maximum grain size (e.g., to ensure the filter stability of sand for increasing the soil capilarity for vegetation plantings, the default value for the *Incorporation of fine sediment* feature in column *S* is set to 2.6 cm corresponding to 0.00667 in)
    - For **stable wood log design maps**, make the following definitions for a feature in `threshold_values.xlsx`:
    - flow depth in row 11 (in m or ft)
    - Froude number in row 13 (dimensionless)
    - frequency threshold in years in row 15 to define the minimum expected lifespan that is required; *River Architect* uses hydraulic parameter rasters (`hQQQQQQ_QQQ` and `uQQQQQQ_QQQ` for the flow depth and Froude number calculation) corresponding to the return period defined within the GetStarted module (read more on the calculation of stable wood log sizes).

More information on threshold values is provided in the Feature descriptions with detailed discussions of the identifiers and threshold values.

9.4 Input: Optional arguments

The checkbox "Include mapping after Raster preparation" provides the optional automated *mapping of results*. The checkbox "Apply wildcard Raster to spatially confine analysis" can be checked to use a *Condition*'s `wild.tif` Raster for spatial limitation of the results. This application makes sense for example if the wildcard Raster contains particular land parcels, 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 (<0.5, see explanations in the SHArC module. 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. Manning's *n* (in s/m1/3) is used in the grain mobility analysis (Angular boulder feature descriptions) to determine shear velocity that acts on grains. The default value is 0.0473934 s/m1/3 (the GUI only shows the first three decimal places), which corresponds to a global optimum in the sample case (gravel-cobble bed river). Even if the unit system is set to U.S. customary, Manning's *n* is defined in the GUI in SI-metric units (an internal conversion factor of k = 1.49 (i.e., n / k is automatically applied for U.S. customary unit settings). The logfiles (produced during the program execution) will state the applied Manning's *n* value if used.

9.5 Run

Once all inputs are defined, click on "Run" (drop-down menu) and "Verify settings" to ensure the consistency of the settings (the window will freeze for some seconds). After successful verification, the selected options change to green font. The "Run" drop-down menu provides the following functions:

- `Raster Maker` prepares lifespan and design Rasters in the directory `RiverArchitect/LifespanDesign/Output/Rasters/CONDITION/`
- `Map Maker` prepares project maps and *PDF* assemblies in the directory `RiverArchitect/02_Maps/CONDITION/`; Before running `Map Maker`, ensure that the correct background image (`01_Conditions/CONDITION/back.tif`) is linked in the project template file `RiverArchitect/02_Maps/templates/river_template.aprx`. The mapping extents can be modified using the `mapping.inp` input file and the Mapping Wiki explains how the symbology, layouts, and other extent settings may be modified.

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` may take 0.1 to 10 hours, depending on the input Raster size, feature set and habitat matching options. After the analysis, the GUI unfreezes and a new button invites to read the logfiles with run information, as well as error and warning messages that may have occurred during the analysis. Alternatively, the lifespan mapping functions can be imported in other *Python* applications (without the GUI) as described in the following *Alternative Run options*.

9.6 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. `Map maker` calls `feature_analysis.map_maker` for the preparation of an *ArcGIS Pro* project (*aprx*) file and maps assembled in `pdfs` in the directory `RiverArchitect/02_Maps/CONDITION/`; by default the layouts stored in `RiverArchitect/02_Maps/CONDITION/` underlie the `pdf` creation but the method also accepts other input directories as an optional argument. *Please note that directories always need to be* ***absolute****; relative paths will result in errors.*

The alternative run options are relevant for example to batch process several *Conditions*. The first alternative run option is to import the *LifespanDesign* module in the *ArcGIS Pro*s Python environment as follows:

1. Prepare input data in `.../01_Conditions/CONDITION/` and adapt the background layer image datasets in `RiverArchitect/02_Maps/templates/river_template.aprx`
2. Go to *ArcGIS Pro*s Python folder and double-click one of the following: `C:\Program Files\ArcGIS\Pro\bin\Python\scripts\propy.bat` or `C:\Program Files\ArcGIS\Pro\bin\Python\envs\arcgispro-py3\python.exe`
3. Enter `import os`
4. Navigate to the script direction using the command `os.chdir("ScriptDirectory")` Example: `os.chdir("D:/Python/RiverArchitect/LifespanDesign/")`
5. 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 (*GeoTIFF*) creation
2. `fa.map_maker("condition", args)` for map (`pdf`) creation

The following steps illustrate the application for creating Rasters.

- Basic execution: `fa.raster_maker("condition")`, for example: `fa.Raster_maker("2008_rrr")`
- 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 60 minutes per feature for input Raster sizes of 5MB and 1GB, respectively). `fa.raster_maker("condition", args)` accepts optional arguments that are `feature_list`, which enables the analysis of any feature listed in the *Features* section. Some examples for particular applications: -> Example 1: `fa.raster_maker("condition", ["Plantings"])` analyzes plantings only. -> Example 2: `fa.raster_maker("condition", ["Plantings", "Boulders/rocks"], True)` analyses plantings and angular boulders (rocks / riprap) only with an optional argument that activates the creation of layouts for plantings and angular boulders (rocks). -> Example 3: `fa.raster_maker("2008_rrr")` analyses all available *features*.
- The complete list of optional arguments of 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 (e.g., `cHSI` = 0.0 to 0.5).
  - `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 m or ft).
  - `args[4] = feature_list` is either `us` (default) or `si`.
  - `args[5] = feature_list`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.map_maker(args, **kwargs)` creates `pdf` map assemblies based on the layout files stored in `RiverArchitect/02_Maps/CONDITION/map_CONDITION_design.aprx` and Rasters prepared with `fa.Raster_maker("condition", ["Featurename"], True)`. `fa.map_maker(args, **kwargs)` accepts an optional argument defining a `CONDITION` and a keyworded argument defining map-able *reach IDs*:

- Option 1: `fa.map_maker("2018_mod")` uses the map folder `RiverArchitect/02_Maps/2018_mod/` and Rasters stored in `RiverArchitect/Output/Rasters/2018_mod/`
- Option 2: `fa.map_maker("2018_mod", "rrr")` uses the map folder `RiverArchitect/02_Maps/2018_mod/`, Rasters stored in `RiverArchitect/Output/Rasters/2018_mod/`, and limits mapping to the extents related to a reach `"rrr"` with its extents defined in `RiverArchitect/ModifyTerrain/.templates/computation_extents.xlsx`

The second alternative run option for the *LifespanDesign* module is to run it as a standalone script (`feature_analysis.py`) from the system command line:

1. Launch terminal (Start -> type `cmd`)
2. Navigate to the place where *ArcGIS* `python.exe` is stored: For example: `C:\Program Files\ArcGIS\Pro\bin\Python\envs\arcgispro-py3\`
3. Run as script: `python.exe DriveLetter:\...\LifespanDesign\feature_analysis("condition", ["Featurename"])`

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 works though); the feature list must be typed as list (in brackets): `Enter the condition (no mandatory; do not forget brackets  example:  >> ["Featurename1", "Featurename2"] >> ["Sidecavity", "Bermsetback"]`
6. The program is now running (this takes time) and it will prompt when it finished.

Calling the module as `.py` script may cause 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` script is to import it as above described.

9.7 Output

9.7.1 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 `shortnames` (see the list on the Features page) is necessary because `arcpy` cannot handle Rasters with names longer than 13 characters when the GRID format is used (even though the standard Raster type is *GeoTIFF*). The analysis automatically shortens too long Raster names based on 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).

9.7.2 Mapping

The module produces an *ArcGIS Pro* project file (`.aprx`) and `.pdfs` in `RiverArchitect/02_Maps/CONDITION/map_CONDITION_design.aprx`. The map extents can be modified in the `mapping.inp` input file. For map modifications, read the *above* descriptions or the Mapping Wiki.

9.7.3 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 oppositely: Optimum areas for application correspond to regions with low lifespans (see more in Schwindt et al. 2019).

9.7.4 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/` folder with the name `logfile.log`. Logfiles from previous runs are overwritten.

CHAPTER

TEN

MAXIMUM LIFESPAN ASSESSMENT (MAXLIFESPAN)

CHAPTER

ELEVEN

INTRODUCTION TO MAXIUM (BEST) LIFESPAN MAPPING

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

- *Quick Guide* to the application of the GUI with a description of required input (Rasters), alternative run options and output descriptions.
- *Output and procedures for* `pdf`*-map generation*.
- *Code conventions and 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 predefined 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. The user 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.

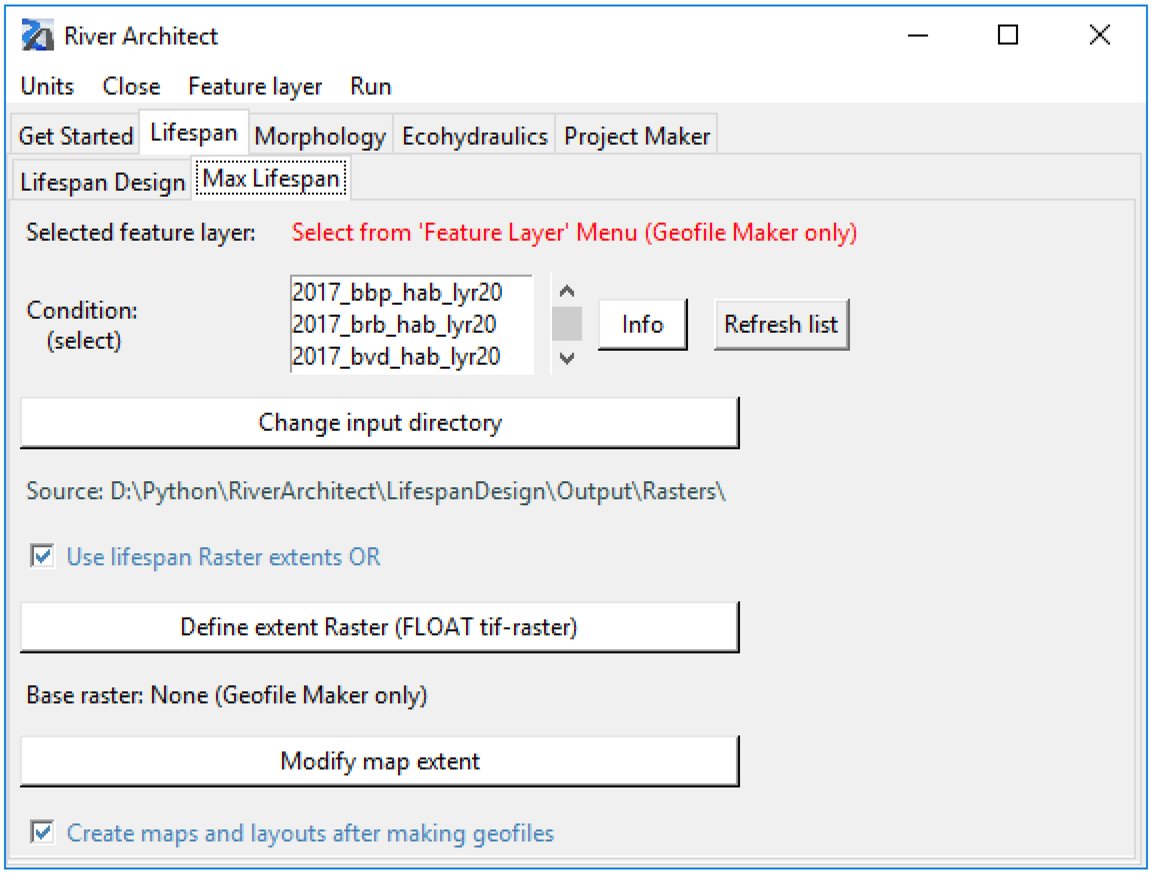
CHAPTER

TWELVE

QUICK GUIDE TO MAXIMUM LIFESPAN MAPS

12.1 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 launched and the following window opens up.



River Architect
Units Close Feature layer Run
Get Started Lifespan Morphology Ecohydraulics Project Maker
Lifespan Design Max Lifespan
Info
Refresh list
Change input directory
Define extent Raster (FLOAT tif-raster)
Modify map extent

Selected feature layer: Select from 'Feature Layer' Menu (Geofile Maker only)

Condition: (select)
2017_bbp_hab_lyr20
2017_brb_hab_lyr20
2017_bvd_hab_lyr20

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

☑ Use lifespan Raster extents OR

Base raster: None (Geofile Maker only)

☑ Create maps and layouts after making geofiles

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.

By default, the *MaxLifespan* will look up lifespan and design maps which are stored in the folder `RiverArchitect/LifespanDesign/Output/Rasters/CONDITION/`. This input directory can be modified by clicking on the `Change input directory` button. 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 the feature group overview). 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.tif`. The mapping checkbox provides the optional creation of maps with the creation of geofiles (Rasters and shapefiles). If the checkbox 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: Map Maker` prepares maximum lifespan map assemblies (`pdfs`) in the directory `RiverArchitect/02_Maps/CONDITION/` (*more information on mapping*).

12.2 Alternative run options

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

1. Run: Geofile Maker calls `action_planner.geo_file_maker`
2. Run: Map Maker calls `action_planner.map_maker`

For batch-processing of multiple scenarios, it can be useful to run the `geo_file_maker` function in a standalone script as follows:

1. Go to *ArcGIS Pro*s Python folder and double-click one of the following: `C:\Program Files\ArcGIS\Pro\bin\Python\scripts\propy.bat` or `C:\Program Files\ArcGIS\Pro\bin\Python\envs\arcgispro-py3\python.exe`
2. Enter `import os`
3. Navigate to script directory using the command `os.chdir("ScriptDirectory")` Example: `os.chdir("D:/Python/RiverArchitect/MaxLifespan/")`
4. Import the *MaxLifespan* module: `import action_planner as ap`
5. 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/`.

12.3 Output

12.3.1 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.

12.3.2 Mapping

The `Map Maker` uses predefined layout templates to overlay

- a background Raster (`RiverArchitect/01_Conditions/CONDITION/back.tif`),
- the best lifespan Raster (`/MaxLifespan/Output/Rasters/CONDITION/max_lf.tif`), and
- shapefiles of best-performing feature areas (`/MaxLifespan/Output/Shapefiles/CONDITION/lf_feat...` or `ds_feat...`).

The module enforces overwriting of existing Raster and shapefiles in the output folder (`/MaxLifespan/Output/`) and it tries to delete any existing content. Before running `Map Maker`, the layout templates are stored in `RiverArchitect/02_Maps/templates/river_template.aprx` and they are named after the feature set type. The mapping wiki pages provide a list of layout names and explain possibilities for modifying the symbology, legend, or map extents. After running `Map Maker`, the map project file and `PDF`s are stored in `RiverArchitect/02_Maps/CONDITION/map_CONDITION_design.aprx`. Anew modifications should be made within this project file (`.aprx`), which is not overwritten, if *River Architect* identifies an existing `CONDITION`-project file.

12.4 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 `logfile.log`. Logfiles from previous runs are overwritten.

CHAPTER

THIRTEEN

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, for example, 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 `CellStatistics`'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 the highest defined lifespan (see *LifespanDesign* module) 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 *max*-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) `max` 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. *LifespanDesign*); (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.

CHAPTER

FOURTEEN

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, modifications 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", "Plants", "Other Bioengineering" and "Longitudinal connectivity" are replaced with "TYPE". The feature `FullName_i` and `shortname_i` must comply with the feature terminology because also the *MaxLifespan* module uses the 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)
```

In addition, the `choose_ref_layout(self, feature_type)` function of the `Mapper` class in `RiverArchitext/.site_packages/riverpy/cMapper.py` needs to be updated:

```
  def choose_ref_layout(self, map_name):
        # type(map_name) == str
        map_name = str(map_name)
        if "lf" in map_name:
            if not ("mlf" in map_name):
                return "layout_lf"
            else:
                if "bio" in map_name.lower():
                    return "layout_mlf_bioeng"
                if "maint" in map_name.lower():
                    return "layout_mlf_maintenance"
                if "plant" in map_name.lower():
```

(continues on next page)

(continued from previous page)

```
                    return "layout_mlf_plants"
                if "terra" in map_name.lower():
                    return "layout_mlf_terraforming"
    ...
```

Moreover, the `choose_ref_map(self, feature_type)` function of the `Mapper` class in `RiverArchitext/.site_packages/riverpy/cMapper.py` needs to be updated:

```
  def choose_ref_layout(self, map_name):
        # type(map_name) == str
        map_name = str(map_name)
                if "lf" in map_name:
            if not ("mlf" in map_name):
                return "layer_lf"
            else:
                if "bio" in map_name.lower():
                    return "layer_mlf_bioeng"
                if "maint" in map_name.lower():
                    return "layer_mlf_maintenance"
                if "plant" in map_name.lower():
                    return "layer_mlf_plants"
                if "terra" in map_name.lower():
                    return "layer_mlf_terraforming"
    ...
```

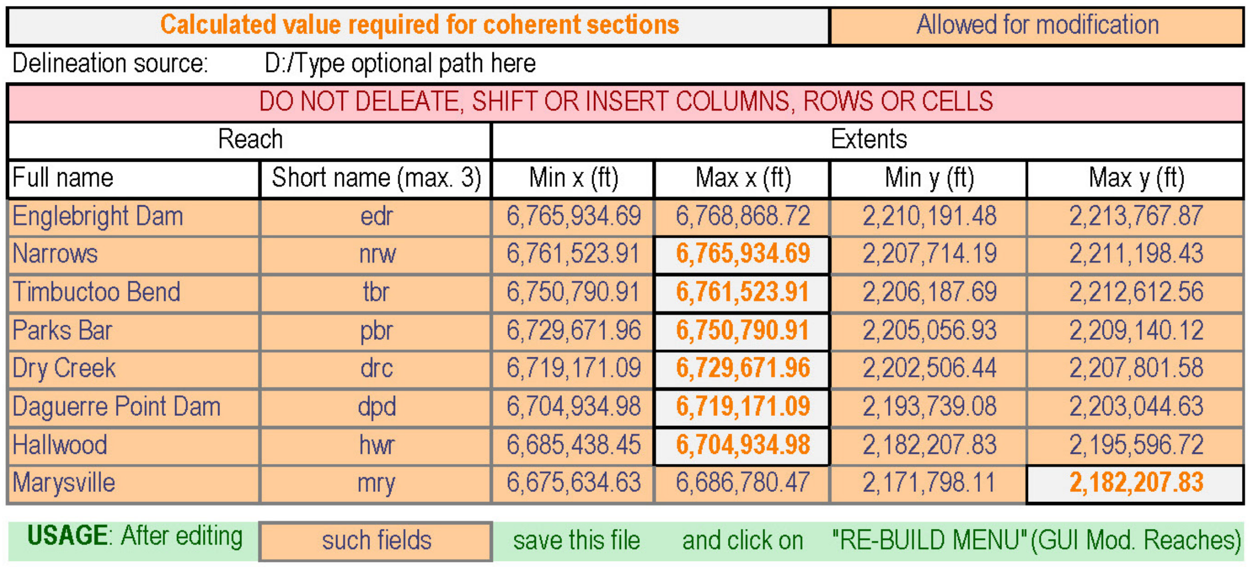
This also requires the creation of the layout and layer in `RiverArchitect/02_Maps/templates/river_template.aprx` and/or `RiverArchitect/02_Maps/CONDITION/map_CONDITION_design.aprx`.

CHAPTER

FIFTEEN

RIVER REACH DEFINITIONS

Particular rivers or reaches for the analysis can be defined from the *LifespanDesign*, *VolumeAssessment* and *ModifyTerrain* modules, referring to: `ModifyTerrain/.templates/computation_extents.xlsx` The concerned 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*. The below figure illustrates the workbook spreadsheet for reach delineation. Changes can be effected by clicking on the `Reaches` dropdown menu and then `DEFINE REACHES`, or directly in the folder `ModifyTerrain/.templates/`. **In order to ignore any reach definitions, select `IGNORE (use Raster extents)`.**



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)

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

`ModifyTerrain/.templates/computation_extents.xlsx` can be opened, for example, by clicking on the `Reaches` drop-down menu (*ModifyTerrain* module) and then "`DEFINE REACHES`". The extents need to correspond to the input DEM coordinate and unit system types (e.g., a georeference may be *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 reach is situated in the south of its upstream reach, its `Max y` value corresponds to the `Min y` value of the upstream reach. Such gapless transitions 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 `Reaches` drop-down menu and then "`RE-BUILD MENU`". Whatever name is stored in the workbook, *River Architect* 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`.

The code handlers for reaches are part of `riverpy` (`RiverArchitect/.site_packages/riverpy/`). The `Reaches` class in `cDefinitions.py` flexibly sends reach information to all modules and reads reach information using the `Read` class in `cReachManager.py`. The dictionaries and lists of the `Reaches` class include an additional entry called `"none": "none"` / `"Raster extents"` to enable reach-independent analyses.

CHAPTER

SIXTEEN

MODIFY TERRAIN (TERRAFORMING)

CHAPTER SEVENTEEN

INTRODUCTION

The *ModifyTerrain* module can (re)model existing terrain DEMs based on threshold values or using the *River Builder* R package. Threshold value-based terraforming uses DEMs of existing rivers and applies either widening (berm setback) or grading, primarily to enable plant survival (for phreatophytes). While the input DEMs are not necessarily required to be part of a *River Architect* *Condition*, the definition of *Condition*s is still required because the module uses maximum lifespan maps and depth to groundwater Rasters to identify candidate pixels for terraforming.

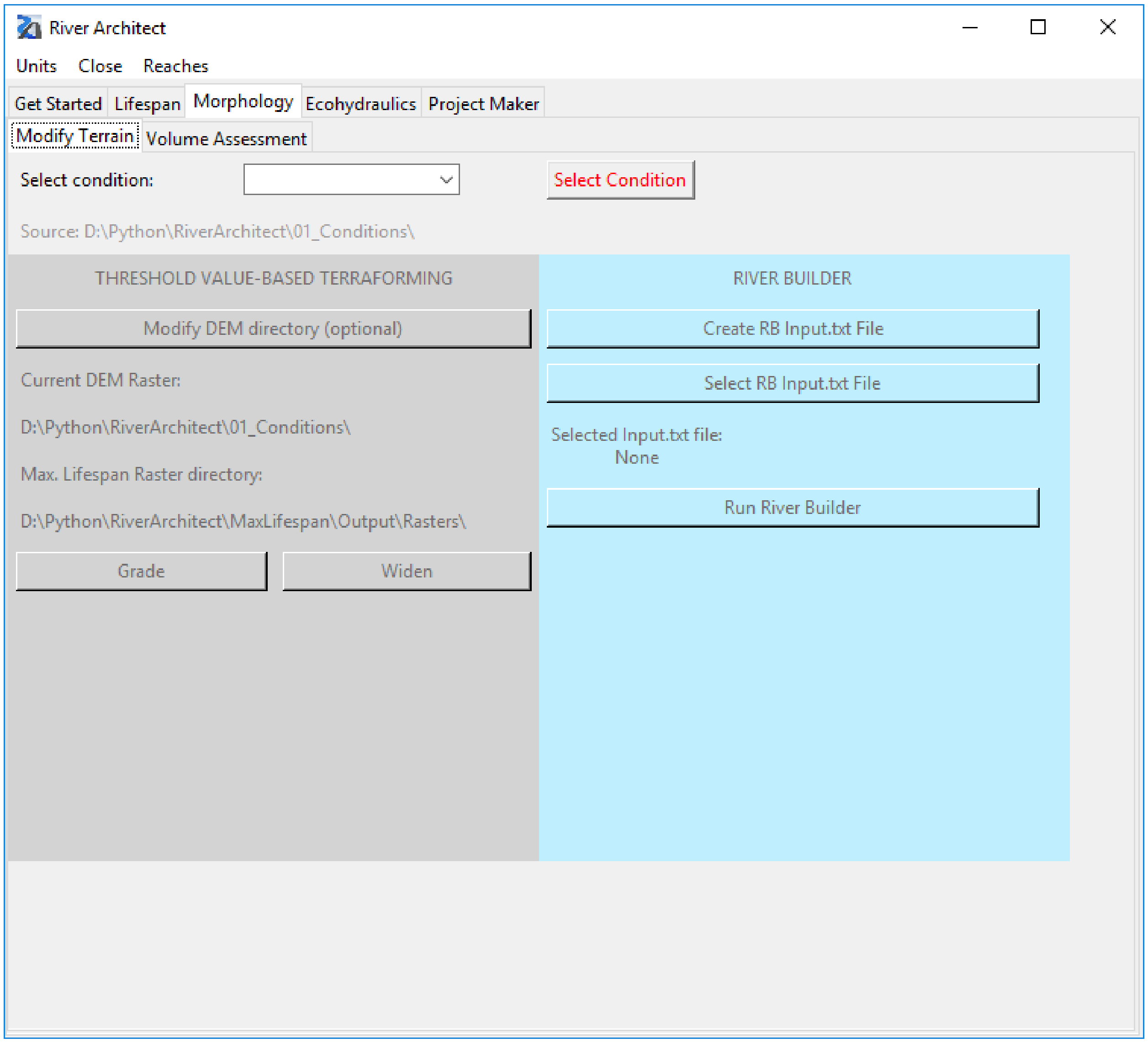
CHAPTER

EIGHTEEN

QUICK GUIDE TO TERRAIN MODIFICATIONS

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 until the following window opens.


River Architect
Units Close Reaches
Get Started Lifespan Morphology Ecohydraulics Project Maker
Modify Terrain Volume Assessment
Select condition:
Select Condition
Source: D:\Python\RiverArchitect\01_Conditions\
THRESHOLD VALUE-BASED TERRAFORMING
Modify DEM directory (optional)
Current DEM Raster:
D:\Python\RiverArchitect\01_Conditions\
Max. Lifespan Raster directory:
D:\Python\RiverArchitect\MaxLifespan\Output\Rasters\
Grade
Widen
RIVER BUILDER
Create RB Input.txt File
Select RB Input.txt File
Selected Input.txt file:
None
Run River Builder

To start with the *ModifyTerrain* module, first a feature set must be chosen from the drop-down menu, which is currently limited to "Widen" and "Grading". Second, a *Condition* needs to be entered, which requires a click on the `Verify` button to update the GUI. This behavior differs from the *LifespanDesign* and *MaxLifespan* modules and an ample revision of the *ModifyTerrain* module will align to the lifespan modules in the future.

18.2 Input: Set Reaches

This module enables analysis for specific *river reaches*, which can be renamed and the reach extents can be modified. The module analyzes all reaches which are defined in a spreadsheet stored in `ModifyTerrain/.templates/computation_extents.xlsx`. For omitting the reach fragmentation, select `IGNORE` from the `Reaches` drop-down menu.

18.3 Input: Set Condition

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 DEM directory (optional)" button. Note that the input folder needs to contain a *GeoTIFF* DEM Raster with the name `dem`; other Raster names are not recognized and the input `dem` is crucial for any operation of the module.

CHAPTER

NINETEEN

THRESHOLD VALUE-BASED TERRAFORMING: WIDENING 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* module's workbook `RiverArchitect/LifespanDesign/Input/.templates/threshold_values.xlsx`. 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 (*GeoTIFF* format) with the name `d2w.tif` is required in the directory `RiverArchitect/01_Conditions/CONDITION/` (use the Get Started's `Populate Condition` function to create a depth to the groundwater Raster). The directory of maximum lifespan and depth to groundwater Rasters can be modified by clicking on the `Change feature max. lifespan Raster directory (optional)` button. This directory needs to contain *GeoTIFF*-Rasters, which have the keywords `grade` or `widen` in their filename.

19.1 Run

All run options in the `Run` dropdown menu enables the `Threshold-based DEM modification` when `Reaches`, `Features`, and a `CONDITION` were defined.

19.2 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 Pro*s Python folder and double-click one of the following: `C:\Program Files\ArcGIS\Pro\bin\Python\scripts\propy.bat` or `C:\Program Files\ArcGIS\Pro\bin\Python\envs\arcgispro-py3\python.exe`
2. Enter `import os`
3. Navigate to Script direction using the command `os.chdir("ScriptDirectory")` Example: `os.chdir("D:/Python/RiverArchitect/ModifyTerrain/")`
4. Import the module: `import cModifyTerrain as cmt`
5. 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 `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 `feat_in_dir` can be a dummy directory `reach_ids` is a list of reach names to limit the analysis

6. The DEM Modification is launched by calling the *ModifyTerrain* object that outputs the calculation logfile with volume change information: `logfile = mt()`
7. The analysis is limited to running the Volume Calculator when the *ModifyTerrain* object is called with arguments: `logfile = mt(True, path_to_modified_DEM)`

19.3 Raster Output

The module creates Rasters of modified DEMs and terrain difference Rasters for grading and/or widen features 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 features shortnames. In addition, terrain **d**ifference Raster, `"d"` with either `"neg"` for excavation or `"pos"` for fill.

19.4 Mapping

Please note that automated mapping is currently deactivated for the *ModifyTerrain* module. For mapping graded or widened terrain, use the VolumeAssessment module and link the *ModifyTerrain* output Rasters to the *ArcGIS Pro* project file `RiverArchitect/02_Maps/CONDITION/map_CONDITION_design.aprx`. The relevant layout names for the *ModifyTerrain* module are:

- `volumes_grade_neg` for mapping the terrain differences of parametrically (threshold-based) floodplain grading within the Modify Terrain module (only excavation `neg` is meaningful)
- `volumes_widen_neg` for mapping the terrain differences of parametrically (threshold-based) river widening within the Modify Terrain module (only excavation `neg` is meaningful)

19.5 Working principles

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 `arcpy`'s spatial analyst: `new_dem = Con((d2w > required_d2w), Float(act_dem - (d2w - required_d2w)), act_dem)`

19.6 Code modification: Add routines for automated DEM modification

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

- Create new function in the `ModifyTerrain` class (file `ModifyTerrain/cModifyTerrain.py`), 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 lifespan 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})
```

(continues on next page)

(continued from previous page)

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

Note:

```
 +  The `self , feat_id , extents` arguments are required for the implementation in␣
↪the call-routine, where is a [features shortnames](River-design-features
↪#introduction-and-feature-groups) `extent` and is an `arcpy.Extent` variable that␣
↪limits DEM creation to this extent.

 +  `self.logger.info()` sends messages to the logger, which are also printed on the␣
↪*Python* 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 to the␣
↪terrain DEM after grading.
```

- 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 CHANGES 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)
```

- Save edits
- The adapted code can now be executed using the *alternative run options*, where `feature_ids = ["shortname of new feature"]`. *Hint:* The new method can also be implemented in the GUI by adding `self.featmenu.add_command(label="New Feature", command=lambda: self.define_feature("new ID")` to `def __init__(...)` 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.

CHAPTER

TWENTY

RIVER BUILDER

Continue reading on River Builder*'s own Wiki page*.

CHAPTER

TWENTYONE

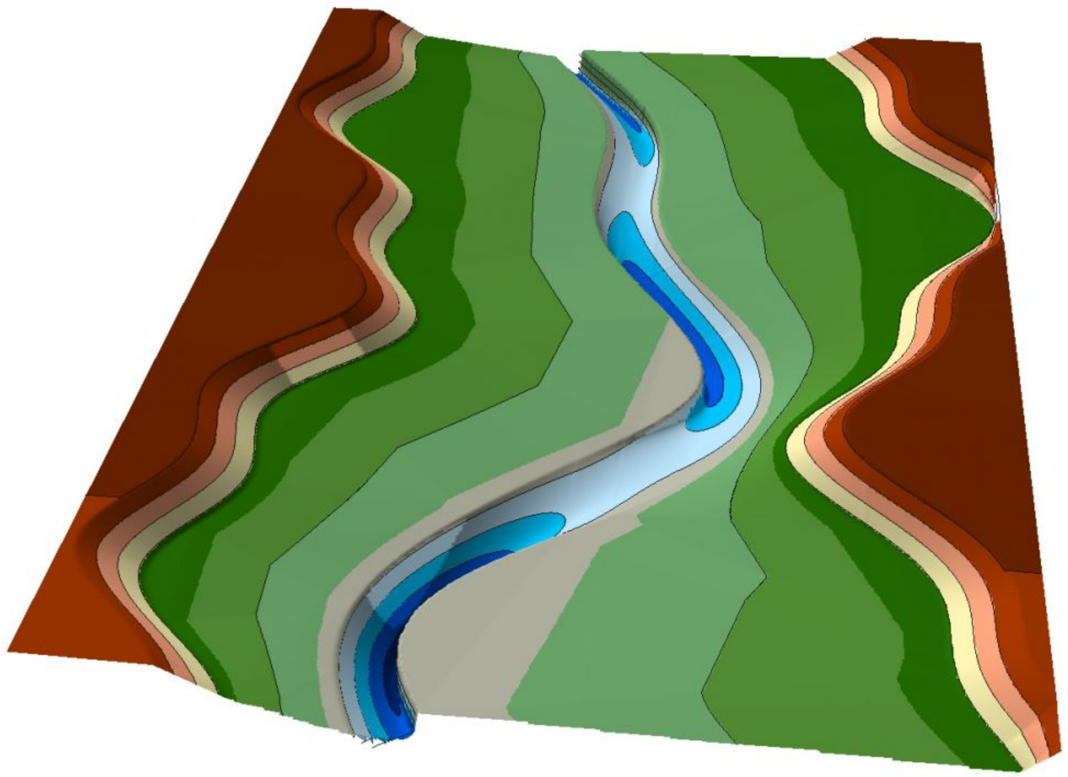
RIVER BUILDER

CHAPTER

TWENTYTWO

INTRODUCTION TO RIVER BUILDER

River Builder is an R package, which is available on CRAN. This R package emulates near-census natural rivers as a function of site-specific, reach-averaged, geometric parameters. Pasternack et al. (2018a, 2018b) introduce the ecomorphological and river classification principles implemented in *River Builder*. The geometric parameters constitute synthetic river valleys, which represent nature-like alluvial channel beds and floodplains. The conceptual approach for designing synthetic river valleys with *River Builder* is described by Brown and Pasternack (2019) and Brown et al. (2014).

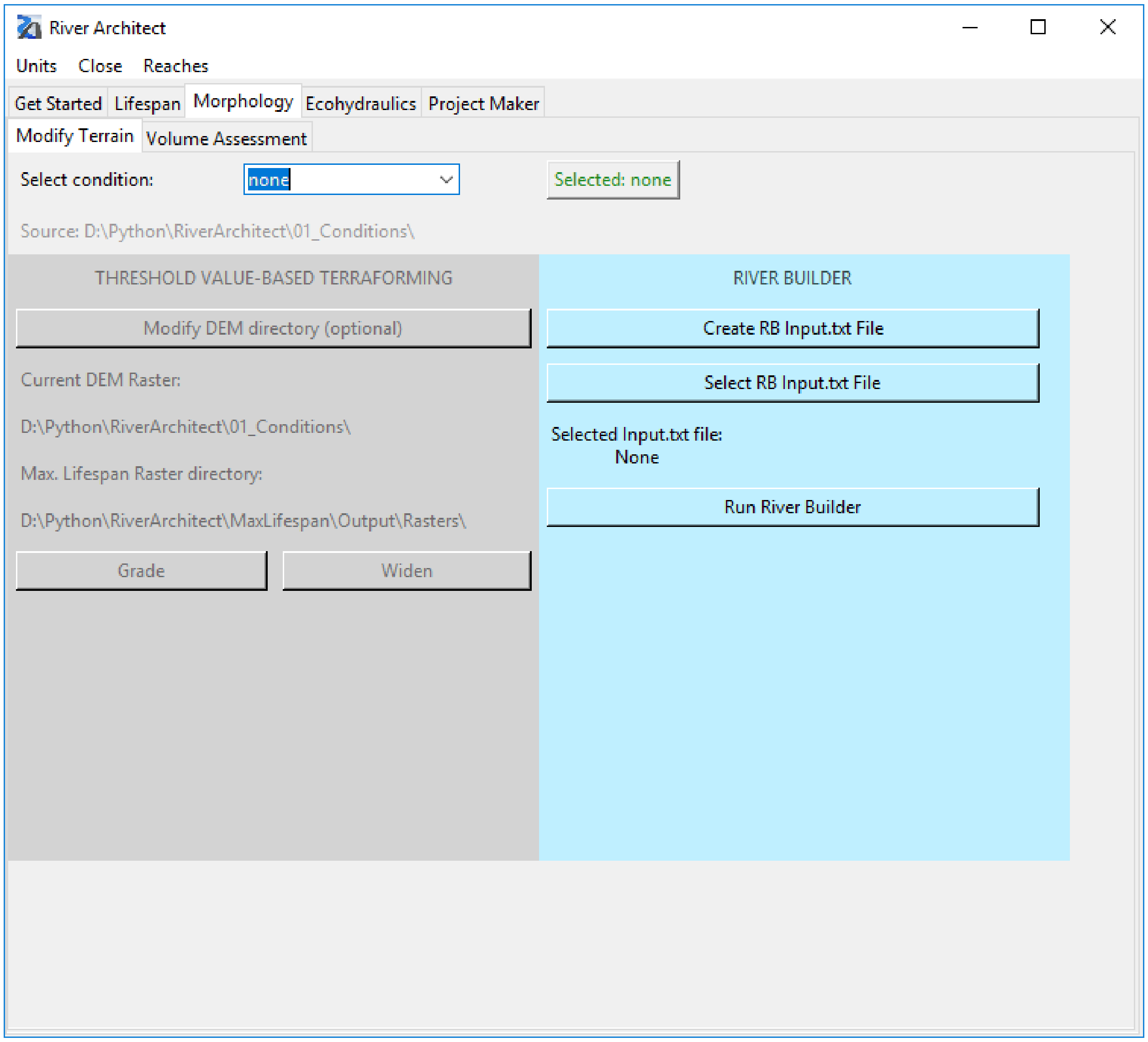


Synthetic river valley created with River Builder (credits: Gregory B. Pasternack).

The synthetic river valley concept produces geodata that can be imported in GIS software (see above illustration) but without reference to real-world coordinates and hydraulic boundary conditions. The boundary conditions for *River Builder* are defined by a user-defined input file and *River Architect* enables the creation of such input files with graphical support. As a result, *River Builder* can be run within *River Architect* (therefore `rpy2` is required). We are working on automating the creation of input files in *River Architect* to the extent that synthetic valleys can be translated into real-world coordinates and reflect hydraulic conditions of real rivers.

Please note that *River Builder* has an own detailed user manual. This wiki primarily describes the usage of *River Builder* within *River Architect*.

To use the `River Builder` interface within *River Architect*, click on the `Morphology` tab and go to the `ModifyTerrain` sub-tab (default). Selecting the `none` *Condition* (confirm with a click on `Select Condition`) enables (unfreezes) the `RIVER BUILDER` frame.


River Architect
Units Close Reaches
Get Started Lifespan Morphology Ecohydraulics Project Maker
Modify Terrain Volume Assessment
THRESHOLD VALUE-BASED TERRAFORMING
Modify DEM directory (optional)
Grade
Widen
RIVER BUILDER
Create RB Input.txt File
Select RB Input.txt File
Run River Builder

Select condition: none

Selected: none

Source: D:\Python\RiverArchitect\01_Conditions\

Current DEM Raster:

D:\Python\RiverArchitect\01_Conditions\

Max. Lifespan Raster directory:

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

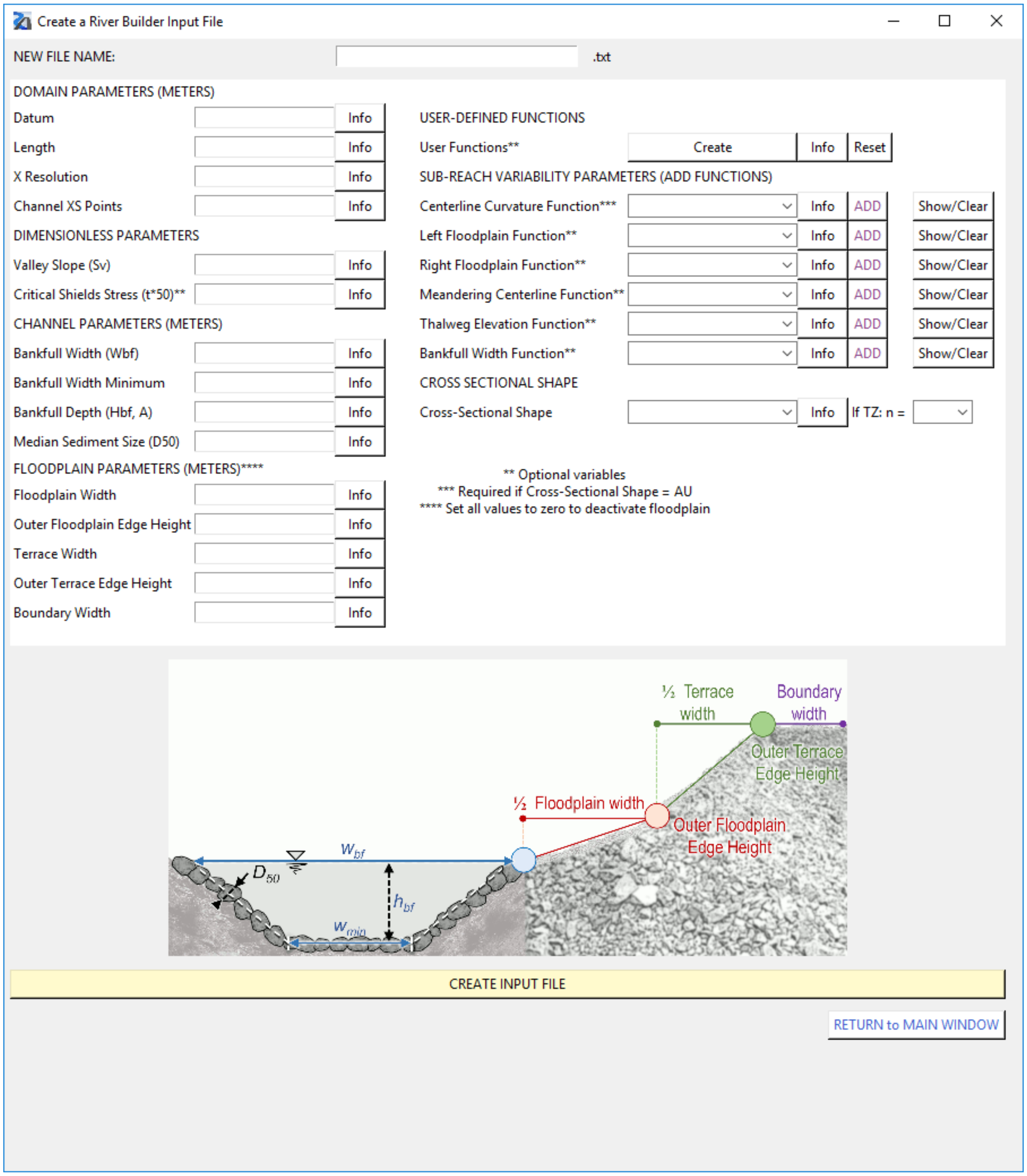
Selected Input.txt file:
None

CHAPTER

TWENTYTHREE

CREATE INPUT FILES

Click on the `Create RB Input.txt File` button to create a new input file for *River Builder*. The following window will pop up.



Create a River Builder Input File

NEW FILE NAME: .txt

DOMAIN PARAMETERS (METERS)

Datum — Info
Length — Info
X Resolution — Info
Channel XS Points — Info

DIMENSIONLESS PARAMETERS

Valley Slope (Sv) — Info
Critical Shields Stress (t*50)** — Info

CHANNEL PARAMETERS (METERS)

Bankfull Width (Wbf) — Info
Bankfull Width Minimum — Info
Bankfull Depth (Hbf, A) — Info
Median Sediment Size (D50) — Info

FLOODPLAIN PARAMETERS (METERS)****

Floodplain Width — Info
Outer Floodplain Edge Height — Info
Terrace Width — Info
Outer Terrace Edge Height — Info
Boundary Width — Info

USER-DEFINED FUNCTIONS

User Functions** — Create — Info — Reset

SUB-REACH VARIABILITY PARAMETERS (ADD FUNCTIONS)

Centerline Curvature Function*** — Info — ADD — Show/Clear
Left Floodplain Function** — Info — ADD — Show/Clear
Right Floodplain Function** — Info — ADD — Show/Clear
Meandering Centerline Function** — Info — ADD — Show/Clear
Thalweg Elevation Function** — Info — ADD — Show/Clear
Bankfull Width Function** — Info — ADD — Show/Clear

CROSS SECTIONAL SHAPE

Cross-Sectional Shape — Info — If TZ: n =

** Optional variables
*** Required if Cross-Sectional Shape = AU
**** Set all values to zero to deactivate floodplain

CREATE INPUT FILE

RETURN to MAIN WINDOW

Make sure to define an input file name (top of the popup window) without entering a file ending. Define all required parameters (i.e., all parameters without any asterisk* in the left column of the popup window). A click on the `Info` buttons opens the parameter descriptions (adapted from *River Builder's user manual*).

The following list summarizes the required parameters and their default values (original descriptions from *River Builder's user manual*):

- DOMAIN PARAMETERS (IN METERS)
  - Datum: FLOAT [1.0=default] - a fixed starting point of operation to measure changes of a specific property.

Used for thalweg elevation. This datum is located at the downstream end of the synthetic river valley, and then the river goes up from there. Thus, if the datum is set as any number > 0, then it is guaranteed that all elevations in the valley will also be > 0. You may set any arbitrary number you wish. If you want to estimate the total elevation range of the thalweg along the river, then simply multiply your Valley Slope input by the Length input and that will give you the overall elevation range of the thalweg. For example, for a 1000 m long channel and a Valley slope of 0.01, then thalweg elevation range is going to span 10 m.

- Length: FLOAT [100.0=default] - the length of the x-axis of the river valley. This is only the channel length if the river is straight. If the river is sinuous, then this length is the valley length and the actual channel length will depend on the sinuosity. Sinuosity is computed by the model and reported in the Data.csv file, so you can compute the final channel length if you want to know it.
- X Resolution: FLOAT > 0.0 (e.g., 1.0 [default] = 1-m resolution) - the resolution of equally spaced data points spaced along the X-axis in units of meters. There is no limit to how fine the point spacing can be, other than computation time for your computer. The smaller this number, the longer the code will take to run. Use your judgment to decide how much detail you need given the scale of your river and the resolution of the sub-reach variability functions you are applying. There is no need for extra detail if the river is easily interpolated with fewer data.
- Channel XS Points: INTEGER > 1 [23=default] - the number of equally spaced points defining the channel's cross-section. This is not applied outside the channel, only in it. Beyond the channel, River Builder provides you with points along the major breaklines, and then you can interpolate the terrain between those on your own later using a TIN approach. For field data collection in a U-shaped channel, people commonly use approx. 15-30 points. An odd number is wise if you want to have a line of output points exactly in the center of the channel.

• DIMENSIONLESS PARAMETERS (-)

  - Valley Slope Sv: FLOAT [0.001=default] - the slope of the valley floor along the X-Z plane. It is used to calculate the overall channel slope with a given sinuosity and govern the incline/decline of the surrounding valley.
  - Critical Dimensionless Bed Shear (aka "Shields) Stress* *, cr, D50: FLOAT [0.047=default] - If you are going to specify a bankfull depth, then set this value to zero. If you want to use the classic equation to establish a bankfull depth to just yield the shear stress needed to mobilize the bed for a specified bed material grain size, then you can use this. Common values used for this parameter are 0.03, 0.045, 0.047, or 0.06, depending on what size fraction you want to insure would be mobilized at bankfull flow. The equation that uses this input is Eq. (4) in Brown et al. (2014). Read more on the dimensionless critical bed shear stress for example in Lamb et al. 2008.

• CHANNEL PARAMETERS (METERS)

  - Bankfull Width Wbf: FLOAT > 0.0 [10.0=default] - the distance between bank tops on the X-Y plane, not counting any sub-reach variability functions.
  - Bankfull Width Minimum: FLOAT >= 0.0 [5.0=default] - the minimum distance between the bank tops on the X-Y plane. This variable is specified to protect the user from making a mistake with sub-reach variability functions in which the banks would cross over, yielding an impossible outcome. This ensures that the river never has a zero or negative width. If you are confident in what you are doing, then you can set this to zero.
  - Bankfull Depth Hbf, A: FLOAT [1.0=default] - the height between the thalweg elevation and then bank top and thalweg on the X-Z plane, not counting any sub-reach variability functions. Depending on channel slope and variability functions, the local bankfull depth can differ from this value.
  - Median Sediment Size D50: FLOAT >= 0.0 [0.01=default] - the particle size of the material comprising the river bed. This is only used if one has left Hbf = 0 and wants to compute the bankfull depth value associated with the initialization of bed material transport for this specified grain size. This value is used in conjunction with the critical Shields stress value as explained earlier.

- FLOODPLAIN PARAMETERS (METERS)*:
  - Floodplain Width*: FLOAT >= 0.0 [default] - double the horizontal distance from the bank top on one side of the river to the outer edge of the floodplain on the same side of the river. Note that if the floodplain has a lateral slope, then this value is still just the horizontal distance, not the slope distance on the floodplain surface. This value is the sum of horizontal floodplain distances for both sizes of the river, so keep that in mind when designing your floodplain. If this number is zero, then there is no floodplain, which yields a confined valley/canyon scenario. If it is a big number, then there is a very wide floodplain. This applies symmetrically to both sides of the river. If you want asymmetric floodplains, then that is handled with independent Left/Right Floodplain sub-reach variability functions.
  - Outer Floodplain Edge Height*: FLOAT >= 0.0 [default] - the incremental increase in elevation from the bank top to the outer edge of the floodplain on the Y-Z plane. If this number is close to zero, then the floodplain is essentially flat. If it is high, then the floodplain is sloped.
  - Terrace Width*: FLOAT >= 0.0 [default] - double the horizontal distance from the outer edge of the floodplain on one side of the river to the outer edge of the terrace on the same side of the river, making it terrace width. This value is the sum of horizontal terrace distances for both sizes of the river, so keep that in mind when designing your terraces. This applies symmetrically to both sides of the river.
  - Outer Terrace Edge Height*: FLOAT >= 0.0 [default] - the incremental increase in elevation from the outer edge of the floodplain to the outer edge of the terrace on the Y-Z plane. If this number is close to zero, then the terrace is essentially flat. If it is high, then the terrace is sloped.
  - Boundary Width*: FLOAT >= 0.0 [default] - double the horizontal distance from the outer edge of the terrace on one side of the river to an arbitrary point out on the valley floor on the same side of the river. This allows for a flat, wide domain that could be another terrace or just the open landscape beyond the alluvial river valley.
- USER-DEFINED FUNCTIONS*
  - User Functions*: OWN WINDOW (multiple assignments possible [no default]) - Three types of functions (Sine, Cosine, and SineSQ) provide periodic oscillations. As simple as those are individually, remember that you can add as many of them as you want into one overall function, using addition, subtraction, etc. As shown in Brown et al. (2014), it is possible to get some very interesting and meaningful variability patterns with 2-4 SIN() functions added together, using different values for frequency, amplitude, and phase. One way to ascertain what those parameter value could be would be to perform harmonic analysis on real series of river variables, such as bed elevation, width, meander centerline, etc. You can extract all the periodic functions you want from that kind of analysis and use those in your design to whatever degree you wish. Because rivers often do exhibit linear trends in variables, such as width and depth, we have added the linear function for users to create that effect. Sometimes you want to have a function that is deterministic and oscillating, but non-periodic. The best application for this type of function would be for the meander centerline because channel alignment is not going to be truly periodic. To achieve this, we provide a Perlin noise function. To find out what sinuosity you get from a particular set of Perlin parameters, you have to run River Builder and look at the Data.csv file. That is a bit clunky, but it works.
- SUB-REACH VARIABILITY PARAMETERS (REQUIRES USER-DEFINED FUNCTIONS)*
  - Centerline Curvature Function*: USER FUNCTION [no default] (multiple assignments possible) - Normally the sub-reach variability functions for centerline curvature would be set equal to those governing the meandering centerline. However, one application of the curvature value is with controlling the asymmetry of a channel's cross-sectional shape. To provide greater flexibility with how a river's XS shape changes along the river, we have created a decoupled centerline curvature variable. This variable does not affect the channels meandering. It only affects how channel asymmetry varies down the river. For example, let's say that you did not want the deepest part of the cross-section to be at the outer bend of a meander, then you could use this to put the deepest part anywhere you want governed by the available variability functions. This is a highly complex tool to control compared to others, so only use it if you know what you are doing with your river design concept.

  - Left Floodplain Function*: USER FUNCTION [no default] (multiple assignments possible) - whereas the channel is treated as one entity having a width, the floodplains have been segregated into independent units. This function governs variability in the location of the outer edge of the river left (i.e., left looking downstream) floodplain.
  - Right Floodplain Function*: USER FUNCTION [no default] (multiple assignments possible) - governs variability in the location of the outer edge of the river left (i.e., left looking downstream) floodplain.
  - Meandering Centerline Function*: USER FUNCTION [no default] (multiple assignments possible) - governs the shape of the river's tortuous flow path (i.e., meandering or sinuosity) on the X-Y plane. At this time, it is not possible to make gooseneck loops, but the available functions offer a good range of flexibility to mimic diverse sinuosity values. If you do not care about the details of the pattern, but only want to match a sinuosity value, then it is recommended that you use the Perlin function and iteratively adjust the parameters until the sinuosity value in the Data.csv output file matches your target value.
  - Thalweg Elevation Function*: USER FUNCTION [no default] (multiple assignments possible) - governs undulations in the bed elevation along the Thalweg on the X-Z plane. Used to create riffle-pool couplets.
  - Bankfull Width Function*: USER FUNCTION [no default] (multiple assignments possible) - governs the downstream variability in the distance between bank tops on the X-Y plane. If you create an undulating function, be very careful that the banks do not cross over to create negative channel widths. We have implemented a minimum bankfull width variable to help avoid this, but it is wise to reason out the amplitude of your width function and not have to rely on that safety net. For longer river segments, try a linear function that increases the width in the downstream direction, consistent with geomorphic theory. You can find parameters for the downstream width increase in the geomorphic literature quite easily.

- CROSS SECTIONAL SHAPE
  - Cross-Sectional Shape: STRING ("AU" [default], "SU" or "TZ") - The shape of the cross-section on the Y-Z plane. Input must be AU, SU, or TZ(*n*), where n is the number of edges that defines the length of the trapezoidal base such that 0 < *n* < (Y-Z Increments). It is recommended that unless you really know what you are doing with AU or are starting from a good AU template file, then only use SU or TZ. If AU is selected, specify a Centerline Curvature. If TZ is selected, specify *n*: - Triangle: *n* = 0 - Trapezoid: 1 <= *n* <= (Y-Z Increments-2) - Rectangle: (Y-Z Increments-3) <= *n* <= (Y-Z Increments)

**All parameters with an asterisk are optional.*

Please note: Each undefined parameter is assigned a default value and all parameters are METRIC. River Architect uses an internal conversion factor when U.S. customary units are selected in the main GUI's `Units` dropdown menu.

A click on `CREATE INPUT FILE` writes the input file to `RiverArchitect/ModifyTerrain/RiverBuilder/INPUT_FILE_NAME.txt`.

CHAPTER

TWENTYFOUR

SELECT INPUT FILES

Once an input file is created, it can be selected by clicking on the `Select RB Input.txt File` button. Also, input files can be used when they were not created with the *above described GUI*, but it is important that these input files are located in `RiverArchitect/ModifyTerrain/RiverBuilder/INPUT_FILE_NAME.txt`. This restriction is necessary because *River Builder* can only read input files that are located in the same directory as the *River Builder* R script (`riverbuilder.r`).

CHAPTER

TWENTYFIVE

RUN RIVER BUILDER WITHIN RIVER ARCHITECT

River Architect uses the `rpy2` package to load *River Builder* (also refer to the *River Architect* installation instructions).

With an input file selected, a click on the `Run River Builder` button launches *River Builder*. Running *River Builder* may take a couple of minutes. Pay attention to messages on the console, which may indicate if *River Builder* had issues with the provided inputs.

CHAPTER

TWENTYSIX

OUTPUT

River Builder writes output files to `RiverArchitect/ModifyTerrain/Output/RiverBuilder/INPUT_FILE_NAME/` (note that the output is stored in the `Output` folder, which is one level above the `RiverBuilder` folder). The following output files are produced (original descriptions from *River Builder's user manual*):

- BoundaryPoints.csv Contains keys that map to specific points in CartesianCoordinates.csv that comprise the boundary around a river's floodplain. This is an essential input file for ArcGIS.
- CartesianCoordinates.csv Contains comma-separated XYZ coordinates for the synthetic river valley. This is an essential input file for ArcGIS.
- Data.csv Contains coefficients of variation, averages, standard deviations, channel slope, and other important information.
- ChannelElevation.png Displays the elevation of the river's channel meander.
- CrossSection.png Displays the river's cross-section at its midway point. The cross-section is bounded above by the river's bank top and below by the thalweg.
- GCS.png Displays the geometric covariance structures of bankfull width and thalweg elevation; thalweg elevation and the channel meander.
- LongitudinalProfile.png Displays the side view of the river which consists of valley top, valley floor, bank top, and thalweg elevation.
- Planform.png Displays the bird's eye view of the river which consists of the channel meander, channel bank, valley floor, and valley top.

For rendering the output into 3D graphs, follow the following instructions from the *River Builder's user manual* (the steps refer to the usage of *ArcMap* and a Python script that is provided along with *River Builder* and included in the *River Architect* standard installation `RiverArchitect/ModifyTerrain/RiverBuilder/RiverBuilderRenderer.py`):

1. Open *ArcMap*. Look for the `Catalog` tab on the far right of the interface. If it does not show, go to `Windows` -> `Catalog` from the top menu to have it opened.
2. In `Catalog`, locate the directory in which the *River Builder* toolbox is stored. If the directory is not visible, select *Connect to Folder* from the `Catalog` toolbar, and select the desired directory.
3. Once the desired directory is visible in `Catalog`, expand the toolbox it contains, right-click the *River Builder* tool, and go to Properties. Under the Source tab, enter the location of where the Python script `RiverArchitect/ModifyTerrain/RiverBuilder/RiverBuilderRenderer.py` is stored in your local machine. Click `OK`.
4. The *River Builder* toolbox is now available for use on *ArcMap*. Select it, and a window will appear. Enter values for the following four inputs:

- Name: The label applied to each output file, i.e. `<Name>_Layer.shp`, `<Name>_Boundary.shp`, etc.

- Output Folder: The location in which an output folder named `<Name>` will be produced. That folder will contain all the output files generated by the script.
- XYZ File: The Cartesian coordinates file produced by the *R* code.
- Boundary Points: The boundary points file produced by the *R* code.

1. When all the inputs are filled, click `OK`, and the script will run. After it is finished running, go to the `<Name>` folder in the output folder that you specified to access the newly generated shape and raster files.

CHAPTER TWENTYSEVEN

ABOUT

River Builder is maintained by Prof. Gregory B. Pasternack, his co-workers, supporters, and friends. Please visit their website to learn more about the general concepts and updates of *River Builder*.

CHAPTER TWENTYEIGHT

VOLUME CHANGE ASSESSMENT

CHAPTER

TWENTYNINE

INTRODUCTION

The *VolumeAssessment* module compares two input DEM Rasters and calculates the volumetric change between both Rasters. The assessment produces workbooks containing reach-wise or project-wise volume differences (excavation and fill), terrain change DEMs, and `pdf`-maps. This chapter explains the module application in the following sections:

- *Quick Guide* to the application of the GUI with descriptions of input requirements and output descriptions.
- *Outputs and procedures for pdf-map generation*.
- *Code conventions and extension*.

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

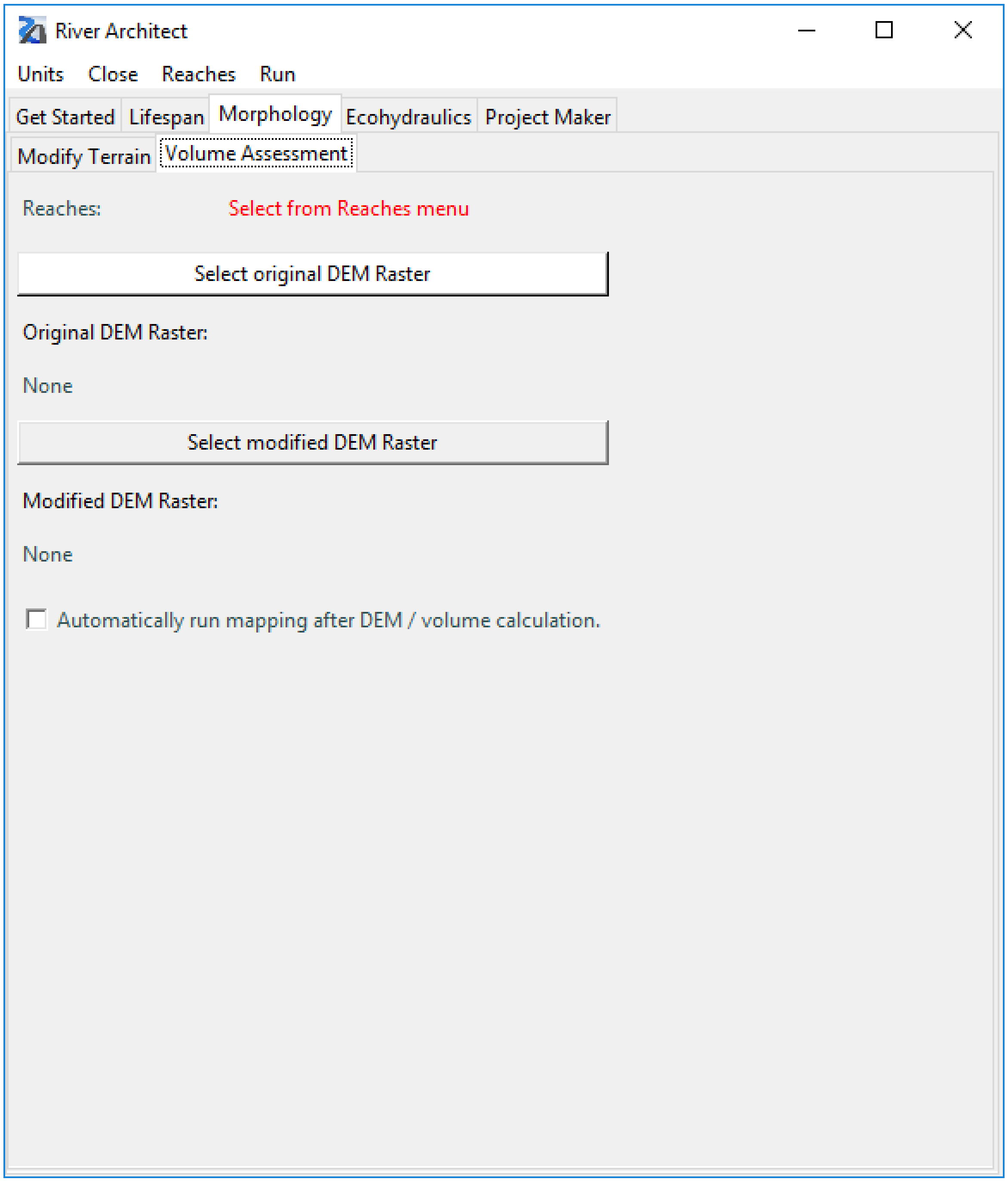
CHAPTER

THIRTY

QUICK GUIDE TO VOLUME ASSESSMENT

30.1 Main window set-up and run

The GUI start-up takes a couple of seconds because the module updates reach information from an external workbook.



Reaches: Select from Reaches menu

Original DEM Raster:

None

Modified DEM Raster:

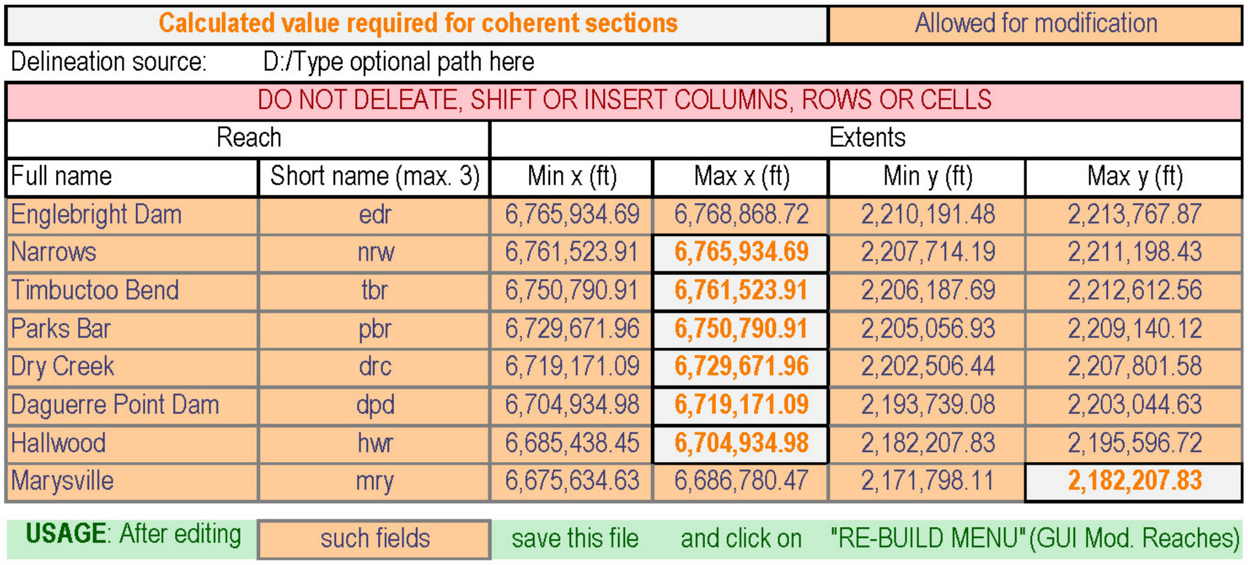
None

☐ Automatically run mapping after DEM / volume calculation.

To start with the *VolumeAssessment* module, first, select a *reach*. The subdivision of the subsequently selected Rasters can be omitted by select `IGNORE` from the `Reaches` drop-down menu. Second, select an input DEM Raster (*GeoTIFF*). Third, select a modified DEM Raster (*GeoTIFF*) that is to be compared with the input DEM Raster.

30.2 Input: Set Reaches

For changing *reach definitions*, please refer to the *reach wiki pages*. Keep in mind that changing *reach definitions* also affects the LifespanDesign and the Modify Terrain modules. 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 `/VolumeAssessment/.templates/computation_extents.xlsx`.



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)

30.3 Mapping

The Run: `Map Maker` uses layout files stored in the *ArcGIS Pro* project file `RiverArchitect/02_Maps/templates/river_template.aprx`. Running `Map Maker` for the first time for a `CONDITION` will produce a copy of the template project file (`.aprx`) that is stored in `RiverArchitect/02_Maps/CONDITION/map_CONDITION_design.aprx`. Before the running `Map Maker` for the first time for a `CONDITION` , ensure that the background layer points at the good background raster (typically this is `RiverArchitect/01_Conditions/CONDITION/back.tif`) The relevant layout names for the *VolumeAssessment* module are:

- `volumes_cust_neg` for mapping required excavation (or scour / erosion) of the input DEM to achieve the modified DEM.
- `volumes_cust_pos` for mapping required fill (or deposition) in the input DEM to achieve the modified DEM.

The *VolumeAssessment* module produces maps with the extents of the input raster by default. Refer to the Mapping wiki for more information on map extents, symbology, and legend modifications.

30.4 Run

The `Run` drop-down menu includes the `Volume Calculator` and the `Map Maker` options. Please note that the `Map Maker` is activated only after the `Volume Calculator` was executed in the current *River Architect* session. Optionally, the bottom checkbox can be activated for automatically running the map creation with the `Volume Calculator`.

30.5 Output

Because the initial state and modified DEM Rasters may be freely selected and not necessarily correspond to a *River Architect Condition*, the *VolumeAssessment* uses **`PSEUDO_CONDITIONS`** that are derived from the directory and name of the modified DEM Raster.

30.5.1 Rasters

The module creates terrain change (volume difference) Rasters in the directory `VolumeAssessment/Output/PSEUDO_CONDITION/`). Raster names contain a reach identifier (`r00`, `r01`, ... `r07` corresponding to spreadsheet rows 6–13) when one or several *reaches* were selected. Otherwise, the Raster names start with a `"ras"` string. Moreover, the Raster names include an `"exc"` or `"fill"` string to indicate excavation (or scour / erosion) and fill, respectively.

30.5.2 Maps

The `Map Maker` (or activated checkbox) produces `pdf` maps of volume difference Rasters in `RiverArchitect/02_Maps/PSEUDO_CONDITION/` (map preparation steps are *above described* and in the [[Mapping] wiki]).

30.5.3 Workbook (spreadsheets)

The volumetric differences in m3 or cubic yards are reach-wise or Raster-wise written to a workbook in the directory `VolumeAssessment/Output/PSEUDO_CONDITION/`, where also the *output Rasters* are located. The workbook template (`volumes_template.xlsx`) is located in `VolumeAssessment/.templates/` and must not be modified. When *VolumeAssessment* is run for the first time for a `PSEUDO_CONDITION`, it creates a copy of the workbook template, which is called `PSEUDO_CONDITION_volumes.xlsx`. *VolumeAssessment* makes two copies of the `template` sheet for excavation and fill, respectively. Thus, one of the spreadsheet copies is called `excavate_YYYYMMDDHHhMM` and lists the reach-wise / Raster-wise excavation volumes in the chosen unit system. The other spreadsheet copy is called `fill_YYYYMMDDHHhMM` and lists the reach-wise / Raster-wise fill volumes in the chosen unit system. The strings `YYYYMMDD` and `HHhMM` indicate the date and time of the program execution. Repeating runs of *VolumeAssessment* on the same modified DEM will append two more copies (excavate and fill) of the `template` sheet with the date-time indicator. It is recommended to cut-paste `PSEUDO_CONDITION_volumes.xlsx` in a `VolumeAssessment/Products/` directory after every run to keep results well-arranged and to force the module to create a new `PSEUDO_CONDITION_volumes.xlsx` file for every run.

CHAPTER

THIRTYONE

WORKING PRINCIPLES

The modified DEM (`PSEUDO_CONDITION`) is subtracted from the input DEM to obtain difference DEMs indicating the *dz* differences in elevation. The module uses a level of change detection `lod` of 0.99 ft (or 0.305 m) to avoid the consideration of DEM imprecision because of pixel size-averaging (excavation): `diff_dem_neg = Con(Abs(input_dem - mod_dem)>= lod, Abs(input_dem - mod_dem), 0.0)` (and vice versa for fill). The excavation and fill volumes result from `arcpy`'s `SurfaceVolume_3d` function, which requires an *ArcGIS* `3D` extension: `volume_excavation = arcpy.SurfaceVolume_3d(diff_dem_neg, "", "ABOVE", 0.0, 1.0)` `volume_fill = arcpy.SurfaceVolume_3d(diff_dem_pos, "", "ABOVE", 0.0, 1.0)` The `volume_...` variables are stored in a list that is finally written to the *output workbook*.

CHAPTER

THIRTYTWO

CODE MODIFICATION: CHANGE SENSITIVITY THRESHOLD (LOD) FOR TERRAIN MODIFICATION DETECTION

The `lod` variable serves for the elimination of virtual terrain differences that may result from pixel sizes and / or Raster transformations (see *above explanations*). The internal variable name for `lod` is `self.volume_threshold` and it is defined in the initiator of the `VolumeAssessment` class (`VolumeAssessment/cVolumeAssessment`). The assigned values of 0.99 ft (U.S. customary) or 0.305 m (SI metric) can be changed in `class VolumeAssessment()` -> `def __init__(self, ...):` -> `# 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  # ft -- CHANGE lod US customary HERE --
else:
    self.convert_volume_to_cy = 1.0  # m3
    self.unit_info = " cubic meter"
    self.volume_threshold = 0.30  # m -- CHANGE lod SI metric HERE --
```

CHAPTER

THIRTYTHREE

STRANDING RISK ASSESSMENT

CHAPTER

THIRTYFOUR

INTRODUCTION TO STRANDING RISK MODULE

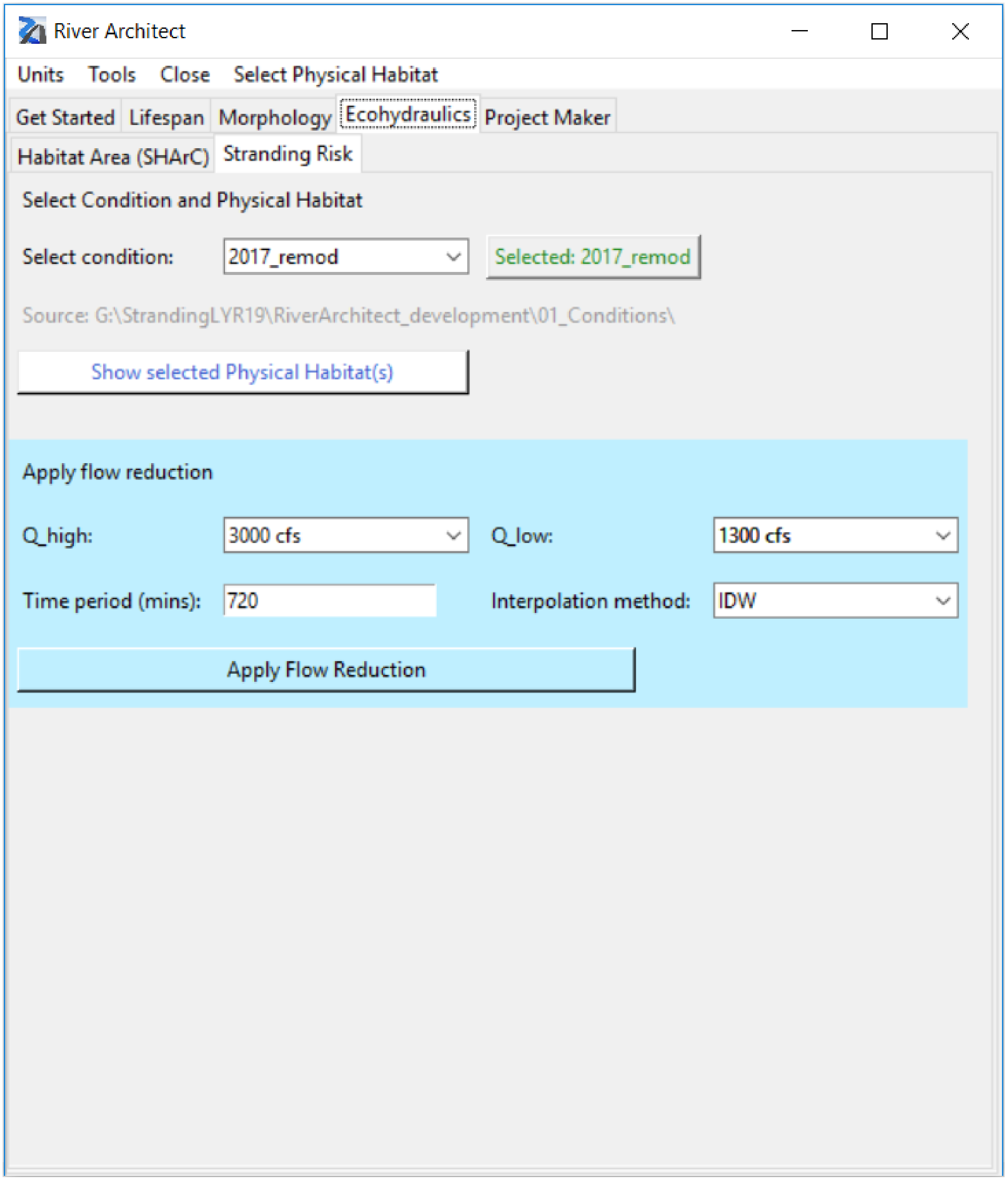
The *Stranding Risk* module can be used to map areas susceptible to stranding and calculate stranding risk metrics for a given fish species, lifestage, and flow reduction scenario. Output rasters include:

- wetted areas that become disconnected from the river mainstem
- discharge at which each area becomes disconnected
- average frequency at which each area disconnects
- habitat suitability of each area prior to disconnection
- estimated ramping rate

CHAPTER

THIRTYFIVE

QUICK GUIDE TO STRANDING RISK ASSESSMENT


River Architect
Units Tools Close Select Physical Habitat
Get Started Lifespan Morphology Ecohydraulics Project Maker
Habitat Area (SHArC) Stranding Risk
Show selected Physical Habitat(s)
Apply Flow Reduction

Select Condition and Physical Habitat

Select condition: 2017_remod — Selected: 2017_remod

Source: G:\StrandingLYR19\RiverArchitect_development\01_Conditions\

Apply flow reduction

Q_high: 3000 cfs

Q_low: 1300 cfs

Time period (mins): 720

Interpolation method: IDW

To begin using the Stranding Risk module, first select a hydraulic Condition.

Note: In order to determine where velocity is a barrier to fish passage, the selected condition must include velocity angle rasters. Otherwise, velocity barriers will not be considered. In order to create the `disconnected_habitat` output raster, cHSI rasters must have already been calculated for the applied Condition and Physical Habitat using the SHArC module. In order to create the `disc_freq` output raster, flows must have already been analyzed (Make flow duration curves) for the applied Condition and Physical Habitat.

Next, select at least one Physical Habitat (fish species/lifestage) from the dropdown menu. The Physical Habitat contains data specific to the fish species/lifestage. Physical Habitat data is used by the Stranding Risk module to determine if fish are able to traverse wetted areas by accounting for the organism's minimum swimming depth and maximum swimming speed (Physical Habitat data are also used by SHArC to determine habitat suitability). These data can be viewed/edited via the drop-down menu: `Select Physical Habitat` –> `DEFINE FISH SPECIES` (scroll to the "Travel Thresholds" section of the workbook).

Once the desired condition and Physical Habitat(s) are selected, choose model discharges Qhigh and Qlow. This defines the range of discharges over which to apply the stranding risk analysis, simulating the changes in habitat connectivity for a flow reduction from Qhigh to Qlow.

Next, input a time period for the downramping. This is the amount of time (in minutes) over which the downramping is expected to occur. This parameter is used to estimate ramping rates as described in *Estimating Ramping Rates*.

Lastly, select an interpolation method. See *Interpolating Hydraulic Rasters* for more information.

For further explanation of the methodology used in the analysis, see Methodology.

Outputs are stored in `StrandingRisk\Output\Condition_name\`. These outputs include:

- interpolated rasters (`h_interp`, `u_interp`, `va_interp`): interpolated depth, velocity (magnitude), and velocity angle rasters. See Interpolating Hydraulic Rasters for more information.'

Outputs specific to the applied flow reduction are stored in the subdirectory `StrandingRisk\Output\Condition_name\flow_red_Qhigh_Qlow`. These outputs include:

- `shortest_paths\`: directory containing a raster for each model discharge in the range Qlow-Qhigh, indicating the minimum distance/least cost required to escape to the river mainstem at Qlow, subject to constraints imposed by the travel thresholds for the selected Physical Habitat. See Escape Route Calculations for more.
- `disc_areas\`: directory containing a shapefile for each model discharge indicating wetted areas which are effectively disconnected at that discharge. See Calculating Disconnected Habitat Area for more.
- `disconnected_area.xlsx`: a spreadsheet containing plotted data of discharge vs disconnected area.
- `disconnected_habitat_specieslifestage.tif`: a raster showing all wetted areas which become disconnected in the applied flow reduction scenario, weighted by the combined habitat suitability index (cHSI) at Qhigh (before the flow reduction occurs). See Calculating Disconnected Habitat Area for more.
- `Q_disconnect.tif`: a raster showing the highest model discharge for which areas are disconnected from the mainstem at Qlow. Locations that are not disconnected at any modeled discharge are assigned a value of zero. Thus, this map indicates locations and discharges below which stranding risks may occur. See Determining Qdisconnect for more.
- `disc_freq.tif`: a raster showing the historical frequency for which each area becomes disconnected, in number of times per year, confined to the season of interest for the analyzed species/lifestage. See *Disconnection Frequencies* for more.
- `ramping_rate_time.tif`: a raster showing the estimated ramping rate before disconnection occurs. See *Estimating Ramping Rates* for more.

CHAPTER

THIRTYSIX

DEFINING TRAVEL THRESHOLDS

Whether or not areas are considered to be connected/navigable for a given fish species/lifestage is dependent upon travel thresholds that are defined in the `Fish.xlsx` workbook (see SHArC for more details). These include a minimum swimming depth and maximum swimming speed. To view/modify the species/lifestage specific travel thresholds, use the drop-down menu: “Select Physical Habitat” –> “DEFINE FISH SPECIES”.

CHAPTER

THIRTYSEVEN

METHODOLOGY

37.1 Interpolating Hydraulic Rasters

The analysis begins by performing an interpolation of the water surface elevation across the extent of the DEM. This step is taken in order to approximate how the water surface extends across floodplain areas, thus identifying the presence of disconnected pools that would not be captured by steady-state hydrodynamic model outputs. The interpolation is performed for each discharge as follows:

- The water depth raster is added to the DEM elevation to produce a water surface elevation (WSE) raster.
- the WSE is interpolated across the DEM extent using an Inverse Distance Weighted (IDW) interpolation scheme on the 12 nearest neighbors.
- The DEM is subtracted from the interpolated WSE raster to produce an interpolated depth raster. Only positive values are saved (negative values indicate an estimated depth to groundwater).
- Interpolated velocity and velocity angle rasters are created, where velocity is set to zero in the newly interpolated areas.
- Interpolated rasters are stored in `StrandingRisk\Condition_name\h_interp`, `StrandingRisk\Condition_name\u_interp`, `StrandingRisk\Condition_name\va_interp`. All interpolated rasters are saved with a corresponding .info.txt file which records the interpolation method and input rasters used in its creation.

The available interpolation methods are:

- `IDW`: Inverse distance weighted interpolation. Each null cell in the low flow depth raster is assigned a weighted average of its 12 nearest neighbors. Each weight is inversely proportional to the second power of the distance between the interpolated cell and its neighbor. Thus, the closest neighbors to an interpolated cell receive the largest weights. This method was found to have the best balance of computational speed and accuracy when tested via cross-validation.
- `Kriging`: Ordinary Kriging interpolation. This method also uses a weighted average of the 12 nearest neighbors, but weights are calculated using a semi-variogram, which describes the variance of the input data set as a function of the distance between points. A spherical functional form is also assumed for the fitted semi-variogram. Kriging is the most accurate interpolation method if certain assumptions are met regarding normality and stationarity of error terms. However, it is also the most computationally expensive method. This method was found to have comparable accuracy compared to IDW, but with significantly higher computational costs.
- `Nearest Neighbor`: The simplest method, which sets the water surface elevation of each null cell to that of the nearest neighboring cell.

37.2 Escape Route Calculations

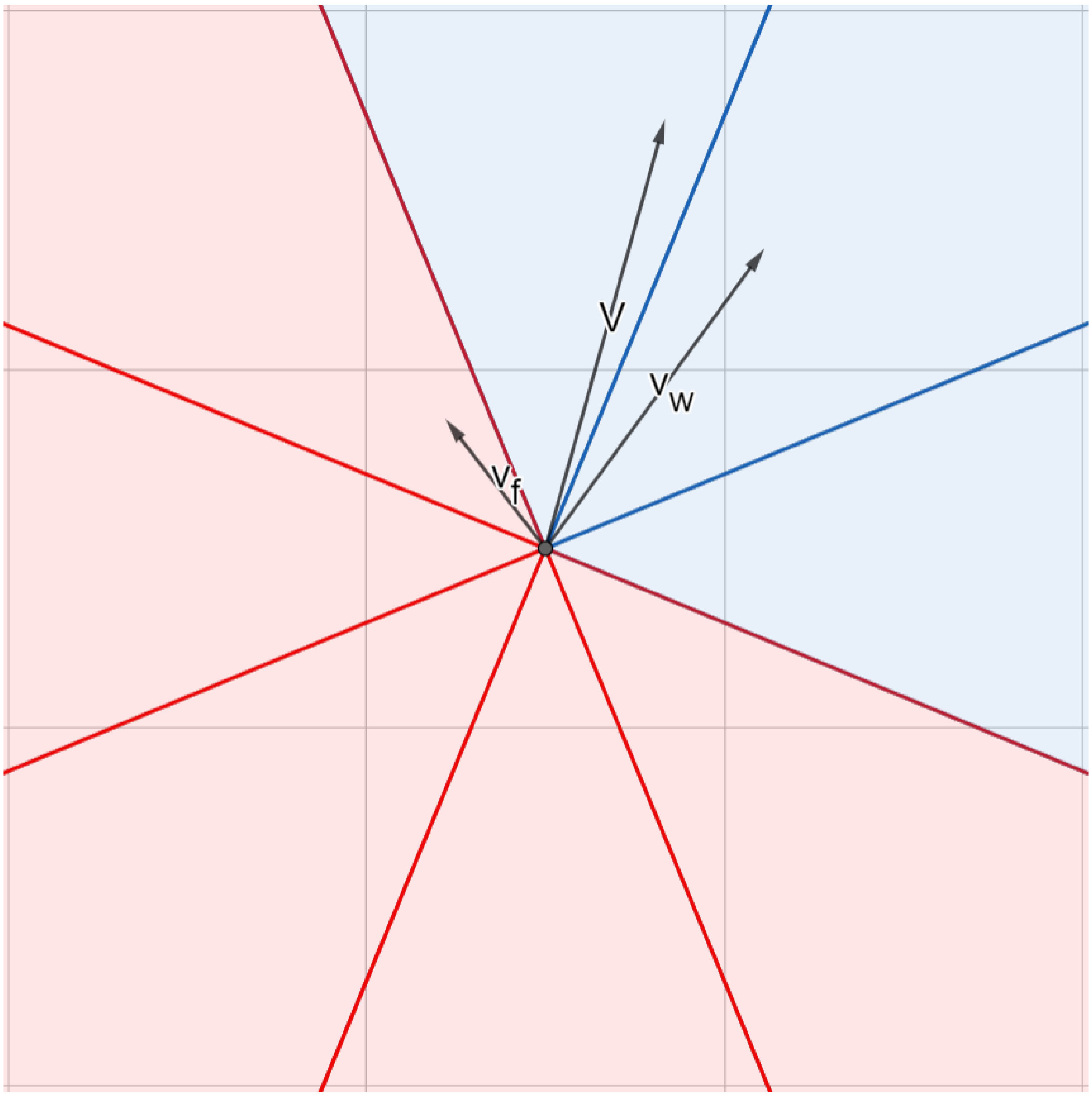
The "shortest escape route" output maps (stored in the `shortest_paths\` output directory) show the length/cost of the shortest/least-cost path from each pixel back into the mainstem of the river channel at Qlow, accounting for both depth and velocity travel thresholds. In this context, the mainstem is defined as the largest continuous portion of interpolated wetted area deeper than the minimum swimming depth at Qlow. Conceptually, the shortest escape route to the mainstem is found from each starting pixel as follows:

- If the starting pixel is already in the mainstem, it assigned a default value of zero. Otherwise, look at all wetted pixels adjacent to the starting pixel.
- Apply depth and velocity travel criteria (see below).
- If both the depth and velocity travel criteria are satisfied, the fish is considered to be able to move from the current pixel to the neighbor. An associated cost of traveling to the neighboring cell can also be computed.
- Apply this method iteratively to each reachable neighbor. The path which reaches a cell in the mainstem with the smallest total cost is the least-cost path to the mainstem.

37.2.1 Applying Depth and Velocity Travel Criteria

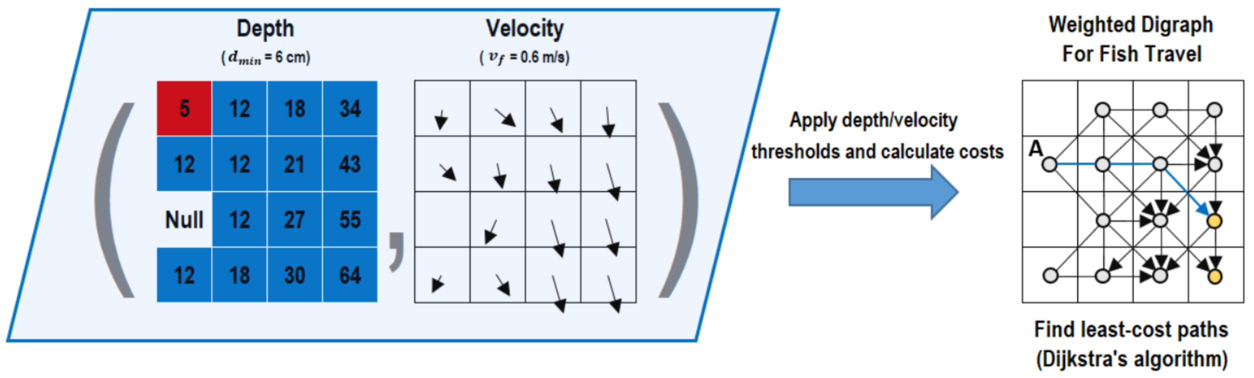
The applied depth and velocity thresholds parameterize the ability of the target fish to travel throughout the river corridor. The criteria to satisfy for travel from cell A to adjacent cell B are defined by three criteria:

- Domain criterion: both cells A and B are wetted.
- Depth criterion: depth at cell B is > d_{min}.
- Velocity criterion: the fish can overcome the current at cell A to reach cell B traveling at v_f



From the center of each cell, the area is divided into 8 octants corresponding to the 8 neighboring cells, with each octant centered on the direction to its neighboring cell and spanning 45°. If it is possible to add the water velocity vector () to a vector with the magnitude of the maximum swimming speed () to yield a vector () falling within an octant, then the velocity threshold criteria is satisfied for travel to the corresponding neighboring cell. A specific example is shown in this diagram where the resultant velocity vector points into the upper quadrant, thus the velocity threshold criteria is satisfied for travel to the upper neighbor. Octants are colored blue/red based on whether the criteria is/is not satisfied for travel in that direction.

Least cost path calculations are implemented in a computationally efficient way by constructing a weighted, directed adjacency graph (stored as a dictionary) and then dynamically traversing the graph working outwards from the mainstem to find shortest path lengths using Dijkstra's algorithm (see explanation below):


Depth
(d_{min} = 6 cm)
Velocity
(v_f = 0.6 m/s)
5
12
18
34
12
12
21
43
Null
12
27
55
12
18
30
64
Apply depth/velocity thresholds and calculate costs
Weighted Digraph For Fish Travel
A
Find least-cost paths (Dijkstra's algorithm)

Applying the travel criteria at each cell yields a set of neighboring cells for which travel is possible. Each cell which can reach other cells or be reached is represented as a vertex of the graph. Reachability of neighboring vertices is represented by edges connecting the vertices, with an arrow symbol indicating the direction of possible travel (edges without arrows indicate travel is possible in both directions). For each edge, an associated cost of traveling along the edge is calculated, yielding a weighted digraph to represent possible fish travel and the associated costs. The least-cost path leading from each vertex to any of the vertices in the target area (corresponding to river mainstem at a lower discharge) is then computed and the path length/cost is stored as a value in the output raster at the location of the starting vertex. Here the target vertices are shown in gold and the least-cost path from point A is shown in blue (where the cost function applied is Euclidean distance).

37.3 Calculating Disconnected Habitat Area

Even if there is wetted area connecting two locations, they may not be considered connected in the context of fish passage. This is because low water depths or high velocities may effectively act as "hydraulic barriers", limiting fish mobility. The applied Physical Habitat contains threshold values for the minimum swimming depth and maximum swimming speed. After escape routes are calculated for a given discharge, the resultant path length raster shows which areas are able to reach the river mainstem for the given hydraulic conditions and biological limitations. Wetted areas in the corresponding interpolated depth raster for which a least-cost path cannot be calculated are considered to be disconnected areas. These disconnected areas are then cropped within the wetted area at Qhigh (not interpolated), as floodplains outside the Qhigh wetted area are assumed to not be active for the simulated flow reduction. Resultant disconnected areas are saved to a shapefile for each discharge in the `disc_areas` output directory.

Once disconnected areas have been calculated, they are weighted by the combined habitat suitability index (cHSI) at Qhigh for the target Physical Habitat. The resultant raster is stored as `disconnected_habitat` in the output directory. Since cHSI serves as a proxy for fish density in habitat-limited environments, the cHSI at Qhigh in areas that become disconnected by the flow reduction serves as a spatially-distributed stranding risk metric.

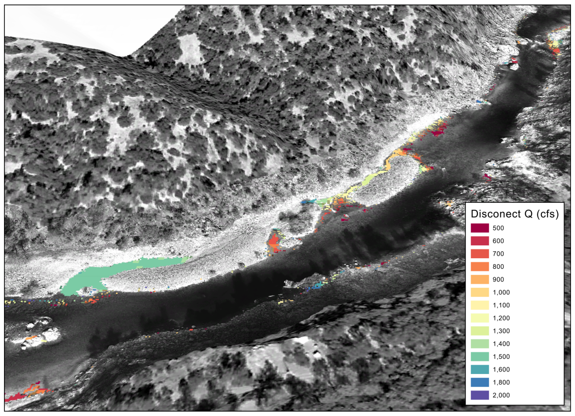
37.4 Determining Qdisconnect

The `Q_disconnect` map outputs provide estimates of the discharges at which wetted areas become disconnected from the mainstem of the river channel. Estimating these discharges is done as follows:

- Assign all pixels wetted at the highest discharge a default value of zero.
- Iterating over discharges in ascending order, pixels within disconnected area polygons are assigned a value of the corresponding discharge.

- The resultant raster is saved as the `Q_disconnect.tif` map output.

Note that the default value of zero indicates pixels wetted at the highest discharge but not present within any of the disconnected area polygons. Other values indicate the highest modeled discharge for which the pixel is disconnected from the channel mainstem.



37.5 Disconnection Frequencies

For each disconnected area, `Q_disconnect` can be combined with the hydrologic flow record in order to calculate the average number of potential stranding events occurring per year. This is generated by computing the total number of times mean daily flow drops below `Q_disconnect` during the season of interest for the applied species/lifestage (Physical Habitat) and dividing by the number of years on record.

A `disc_freq` output map will be produced provided that flows have already been analyzed (Make flow duration curves) for the applied Condition and Physical Habitat.

37.6 Estimating Ramping Rates

Ramping rates are estimated by applying the following assumptions:

- the flow reduction scenario is characterized by a linearly downramping hydrograph from Qhigh to Qlow over the input time period
- Downstream attenuation of the hydrograph is not considered

Ramping rates are estimated via linear approximation, using the difference in depths between model discharges prior to disconnection.

The ramping rate output raster is stored as `ramping_rate_###min.tif`, where `###` corresponds to the input time period over which the downramping occurs.

CHAPTER THIRTYEIGHT

REFERENCES

Travel thresholds used to define Physical Habitats in `Fish.xlsx` can be estimated from the following sources:

Bell, M. C. (1991). Fisheries Handbook of Engineering Requirements and Biological Criteria (3 ed.). Portland, Oregon.

CDFG. (2012). Standard Operating Procedure for Critical Riffle Analysis for Fish Passage in California DFG-IFP-001, updated February 2013. California Department of Fish and Game, DFG-IFP-00(September), 1–25.

Katopodis, C., & Gervais, R. (2016). Fish swimming performance database and analyses. DFO Can. Sci. Advis. Sec. Res. Doc. 2016/002.

U.S. Forest Service FishXing Data Table (various sources)

CHAPTER

THIRTYNINE

PROJECT MAKER

CHAPTER FORTY

INTRODUCTION TO THE PROJECT MAKER 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 that can be delineated with an own `ProjectArea.shp` shapefile. *version*s of a restoration plan may refer to terraforming options or other planning *Condition*s. A project proposal is prepared for (preliminarily) *version*s of a restoration plan including relevant nature-based engineering features (i.e., vegetation plantings, stabilizing features such as the placement of angular boulders, and *anchored* streamwood) and it evaluates cost-relevant quantities. A project cost table uses the cost-relevant quantities for a preliminary cost estimate. The habitat utility in terms of net gain in **S**easonal **H**abitat **Area** (**SHArea**) for target fish species determines the project return in *"US$ per [acre or m2] of newly created SHArea"*. This Wiki page is organized as follows:

- *ProjectMaker GUI usage*
- *Generate a project plan and run a cost-quantity assessment*
- *Map final designs*
- *Calculate gain in SHArea*

CHAPTER

FORTYONE

QUICK GUIDE TO A PROJECT ASSESSMENT

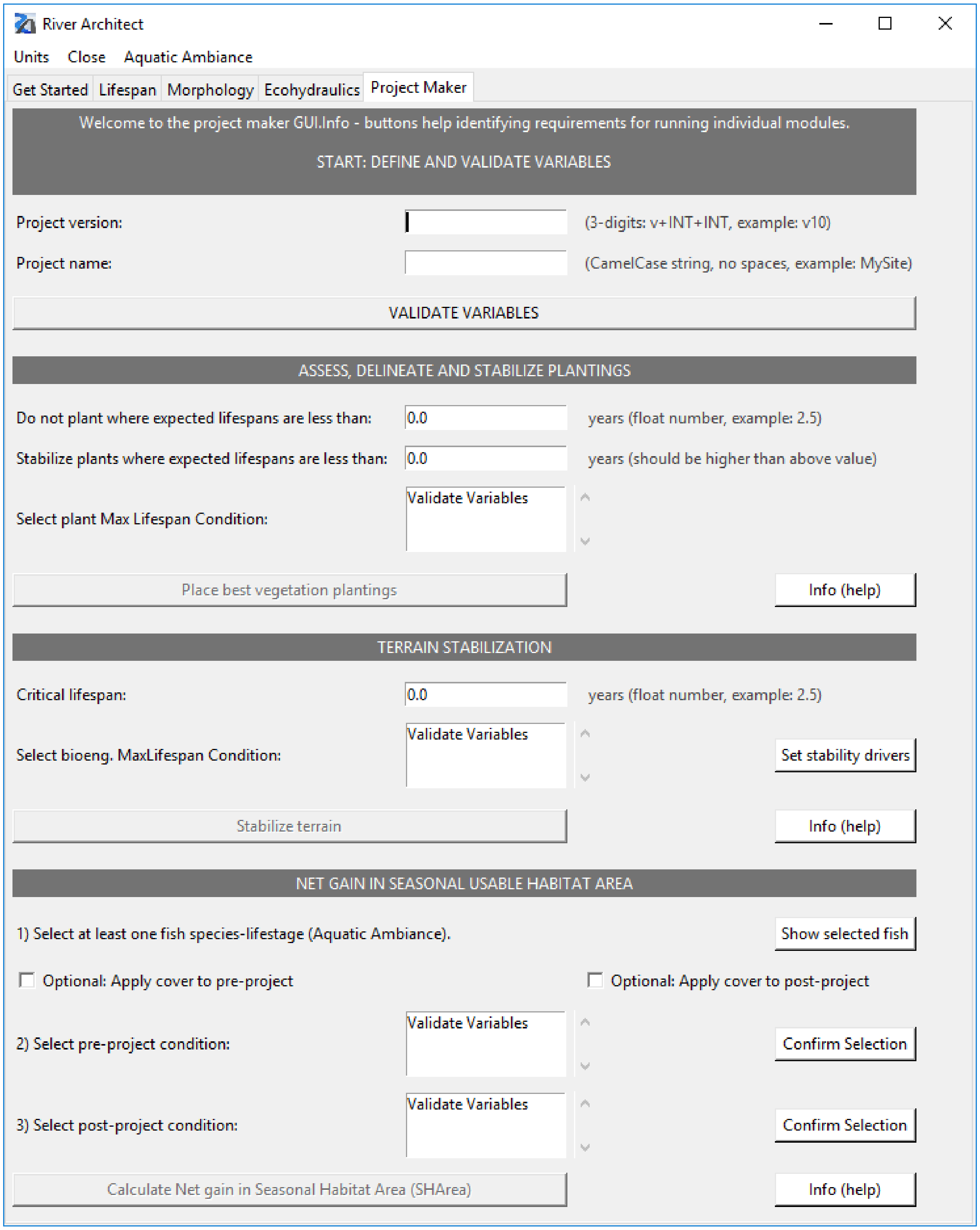
41.1 Prerequisites

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

- If terraforming applies:
  - The *SiteName* restoration terraforming plan was verified with 2D hydrodynamic modeling
  - The *River Architect*'s VolumeAssessment module was applied to calculate excavation / fill volumes.
- The *LifespanDesign* and *MaxLifespan* modules were executed for plantings and other nature-based engineering features. Thus, the following directories should exist and contain plantings and other nature-based engineering feature rasters:
  - Plantings:
    * `RiverArchitect/LifespanDesign/Output/Rasters/CONDITION_lyr20/`
    * `RiverArchitect/MaxLifespan/Output/Rasters/CONDITION_lyr20/`
  - Other nature-based engineering features:
    * `RiverArchitect/LifespanDesign/Output/Rasters/CONDITION_lyr20/`
    * `RiverArchitect/MaxLifespan/Output/Rasters/CONDITION_lyr20/`
- The *SHArC* module was applied to the pre-project (initial) condition and the "with implementation" ("as-built") condition.

41.2 Main window set-up and run

The below figures shows the *ProjectMaker* GUI at startup.



River Architect

Units Close Aquatic Ambiance

Get Started | Lifespan | Morphology | Ecohydraulics | Project Maker

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)

Project name: (CamelCase string, no spaces, example: MySite)

VALIDATE VARIABLES

ASSESS, DELINEATE AND STABILIZE PLANTINGS

Do not plant where expected lifespans are less than: 0.0 years (float number, example: 2.5)

Stabilize plants where expected lifespans are less than: 0.0 years (should be higher than above value)

Select plant Max Lifespan Condition: Validate Variables

Place best vegetation plantings | Info (help)

TERRAIN STABILIZATION

Critical lifespan: 0.0 years (float number, example: 2.5)

Select bioeng. MaxLifespan Condition: Validate Variables | Set stability drivers

Stabilize terrain | Info (help)

NET GAIN IN SEASONAL USABLE HABITAT AREA

1) Select at least one fish species-lifestage (Aquatic Ambiance). | 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 Seasonal Habitat Area (SHArea) | Info (help)

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.

41.3 Input: Variables and automatically generated files

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

- *Project version* (or *vii*) is a "*v*" + 2-digits (*ii*) version number (string), for example, `v10` (other 3-digit strings are also allowed)
- *Project name* is a string (e.g., of a particular site) written in *CamelCase*, for example, `RavineConfluence`

Click on the `VALIDATE VARIABLES` button to verify that the variables entered are correct. A successful validation creates a copy of the template project structure (`RiverArchitect/ProjectMaker/.templates/Project_vii_TEMPLATE/`), which is typically saved as `RiverArchitect/ProjectMaker/ProjectName_vii/`. The *variable validation* opens an info-box, a project assessment workbook (`.xlsx`), and a mapping project (`.aprx` in *ArcGIS Pro*) that invites to create project-specific files. The required actions include:

- WORKBOOK (`ProjectName/ProjectName_assessment_version.xlsx`) The workbook contains a spreadsheet named *costs*, where unit costs and quantities are evaluated. The *from_geodata* sheet 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. After running the cost-quantity assessment, verify that the cells containing automatically calculated costs in the *costs* sheet link to the correct cells in the *from_geodata* sheet.
- MAPPING (`ProjectName/ProjectMaps.aprx`) The copy of *ProjectMaker*'s template project structure contains an *ArcGIS Pro* project file (`ProjectName/ProjectMaps.aprx`) with predefined layouts and layers that may require updates to site-specific geofiles (shapefiles and rasters). Project-specific geofiles (shapefiles and rasters) need to be created as described in the following sections.

41.4 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 template layouts provided in `ProjectName/ProjectMaps.aprx`).

1. Create a new polygon-shapefile in `ProjectName/Geodata/Shapefiles/` (read more on arcgis.com), name it `ProjectArea`, and select the project *Coordinate System* (`Current Map`). Click on `Run`.
2. Remove the newly created layer from layout's *Contents* tab, 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 layout's `Contents` tab, right-click on the `ProjectArea` layer, then `Attribute Table`. In the `Attribute Table`, click on the top-left *Field:* `Add` button. Name the field `AreaCode`, select a `Text` in the *Data Type* column and `50` in the *Length* column. Click below the new field to and add another field (*Click here to add a new field* label) with the following properties: *Field Name* = `gridcode`, *Data Type* = `Short` data type, *Number Format* = `Numeric`, *Precision* = 0, and *Scale* = 0 field named `gridcode`. Go to *ArcGIS Pro*'s *Fields* ribbon (top of the window) and click `Save`. Close the *Fields: Project area (. . .)* tab (the one where the fields were previously added).
4. Delineate project area
    1. Optional: Import modified terrain to visualize boundaries of terraforming.

2. In *ArcGIS Pro*'s *Edit* ribbon (top of the window), click on `Create` (ensure that the `ProjectArea` layer is selected in the *Contents* tab) > A *Create Features* opens.
3. In the *Create Features* tab, click (highlight) on `ProjectArea`, then on `Polygon`.
4. Draw a polygon around the designated project area (finish with the `F2`-key).
5. Go to the `Attribute Table` tab and type *`Restoration zone`* in the `AreaCode` field and *`1`* in the `gridcode` field.
6. `Save` edits.

41.5 Input: Plantings shapefiles

Project Maker will place plants only where it makes sense and removes existing plants where necessary. An overlay of the above-created project area polygon over recent satellite image shows, where existing plants intersect with projected actions and where these plants may need to be cleared (removed). A *PlantExisting.shp* shapefile with polygons delineating these intersects needs to be created and drawn as follows in the `ProjectName/ProjectMaps.aprx`-file:

1. In the Catalog tab, open the folder tree `ProjectName/Geodata/Shapefiles/` (double click on the folder to make it appear in the lower box).
2. Create a new polygon-shapefile in `ProjectName/Geodata/Shapefiles/` (read more on arcgis.com), name it `PlantExisting`, and select the project *Coordinate System* (`Current Map`). Click on `Run`.
3. Remove the newly created layer from layout's *Contents* tab, double-click on the existing `Existing plants (all)` layer -> `Layer Properties` opens up -> go to the `Source` tab -> click on `Set Data Source ...` -> Select the newly created `.../Shapefiles/PlantExisting.shp` file -> click `OK`.
4. In the layout's `Contents` tab, right-click on the `PlantExisting` layer, then `Attribute Table`. In the `Attribute Table`, click on the top-left *Field:* `Add` button. Name the field `ActionType`, select a `Text` in the *Data Type* column and `50` in the *Length* column. Click below the new field to and add another field (*Click here to add a new field* label) with the following properties: *Field Name* = `gridcode`, *Data Type* = `Short` data type, *Number Format* = `Numeric`, *Precision* = 0, and *Scale* = 0 field named `gridcode`. Go to *ArcGIS Pro*'s *Fields* ribbon (top of the window) and click `Save`. Close the *Fields: Project area (. . .)* tab (the one where the fields were previously added).
5. Delineate existing plantings area:
   1. Ensure that a valid background image is linked to the `background` layer (`Properties` -> `Source` tab).
   2. In *ArcGIS Pro*'s *Edit* ribbon (top of the window), click on `Create` (ensure that the `PlantExisting` layer is selected in the *Contents* tab) > A *Create Features* opens.
   3. In the *Create Features* tab, click (highlight) on `PlantExisting`, then on `Polygon`.
   4. Draw polygons around existing plantings that are visible in the background (satellite image) project area, within the zone where the modified DEM rasters indicate terrain modification (finish polygons with the `F2`-key).
   5. Go to the `Attribute Table` tab and type `Existing` (*text*) in the `ActionType` field and `1` (*short integer*) in the `gridcode` field.
   6. Once all visible plantings within the project area are delineated, save the edits.

Terraforming may require clearing of existing vegetation in the project area. In this case, use the above created *PlantExisting.shp* as template to delineate plants to remove (to be cleared):

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

2. Make a copy of a new polygon-shapefile in `ProjectName/Geodata/Shapefiles/PlantExisting.shp` and name it `PlantClearing.shp` in the same folder.
3. In the map *Contents* tab, 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/PlantClearing.shp` file -> click `OK`.
4. Delineate existing plantings to be removed (clearing):
    1. In the layout's `Contents` tab, right-click on the `PlantClearing` layer, then `Attribute Table`.
    2. Highlight all polygons of existing plants that do not need to be removed. Press `Delete` to remove these plant polygons from the clearing list. *When highliting existing plantings for clearing, remember that in river restoration and habitat enhancement projects "clearing" should limit to the absolutely required minimum. That means: Delete as many polygons of existing plants as possble from `PlantClearing.shp`.*
    3. Once all non-clearing plants are removed, save the edits.
5. *Project Maker* will not place any plant where existing plants are, even though when these are highlighted in the the `PlantClearing` shapefile. Therefore, consider to remove the polygons contained in `PlantClearing.shp` from `PlantExisting.shp`.

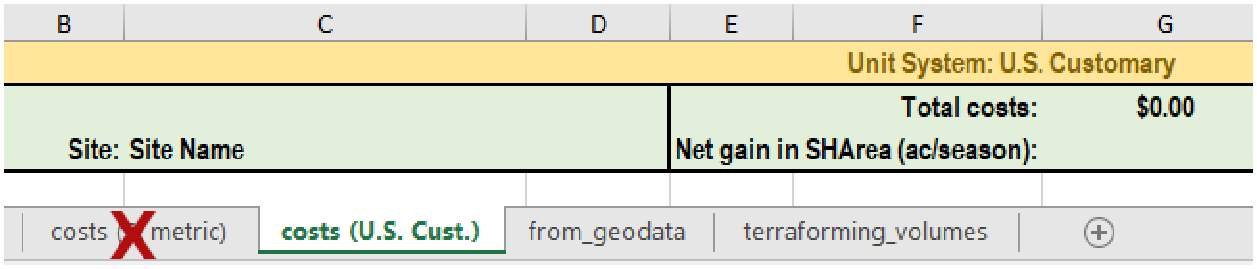
Save and close `ProjectName/ProjectMaps.aprx`. **Note: Both `PlantExisting` and `PlantClearing` shapefiles are not mandatory for running *Project Maker*, but recommended.**

CHAPTER

FORTYTWO

COST QUANTITY ASSESSMENT AND THE COST MASTER WORKBOOK

The `ProjectName_assessment_vii.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 below figure). **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.

42.1 Terraforming

The VolumeAssessment module evaluated terrain excavation and fill volumes. *VolumeAssessment* created workbooks featuring terraforming volumes in cubic meters/yards in the directory `RiverArchitect/VolumeAssessment/Output/PSEUDO_CONDITION_volumes.xlsx`. Optionally, these workbooks can be copied to a `PSEUDO_CONDITION_volumes.xlsx` workbook in the project folder. Recall: *`PSEUDO_CONDITION_volumes.xlsx`* has to tabs: (1) *excavate_YYYYMMDDHHhMM* and (2) *fill_YYYYMMDDHHhMM*. Copy the terraforming volumes from either of these two spreadsheets to the cost master workbook's (`ProjectName_assessment_vii`) *terraforming_volumes* spreadsheet (cells are highlighted, only values). The template's unit costs of US$ 23.00 per cubic yard (or EUR 23.00 per m3) include short transport distances (< 1 km) and material storage. It is hypothesized that the smaller value (i.e., either the *excavate* or the *fill* volume) is incorporated in the costs of the higher value because the smaller volume can be reused on-site. The costs for terraforming 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 · max(vol!C5, vol!C6)

42.2 Vegetation plantings and supporting features

Before the most reasonable vegetation plantings are implemented into the project plan, the LifespanDesign and *MaxLifespan* modules need to be run based on anew 2D simulations made with the terraformed DEM. The resulting (maximum) lifespan rasters need to be available in the directories `RiverArchitect/LifespanDesign/Output/Rasters/CONDITION_lyr20/` and `RiverArchitect/MaxLifespan/Output/Rasters/CONDITION_lyr20/`.

Define critical lifespans Before plantings can be placed, *River Architect* wants to know the critical threshold for lifespans that a plant species needs have to be applicable (2.5 may be a reasonable estimate here). For example, if the field *Do not plant where expected lifespans are less than* = `2.5`, *River Architect* will only place plantings with an expected lifespan of 2.5 years or more. In addition, *River Architect* requires the definition of a critical lifespan of vegetation plantings, which require additional nature-based engineering support. For example, if the field *Stabilize plants where expected lifespans are less than* = `10.0`, *River Architect* will add supporting nature-based engineering features where the best performing plant species lifespan is less than 10.0 years. Note that the second value needs to be higher than the first value to reasonable results.

Place plantings The GUI's *Place best vegetation 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. Read the logfile 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 (`ProjectName_assessment_vii.xlsx`) is closed and traceback error messages. Re-run *Delineate Plantings* and traceback 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. **Note: *Project Maker* places best plants in alphabetic order.** In the case of the pre-defined plant species, that means, first a pixel is tested for its suitability for *Box Elder*, then for *Cottonwood*, then *White Alder*, and then *Willows*. If a pixel already got assigned a plant species, it will not be tested again for other plant species. For example, if a pixel got assigned *Box Elder*, it will not be considered for all other plant species.

Stabilize plantings Even though the vegetation plantings maximum lifespan maps identify the optimum plant species 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 nature-based engineering features such as anchored wood logs to planting areas associated with the user-defined *minimum plantings lifespan*. 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 the *Critical plantings lifespan*.
2. Engineered (anchored) wood logs, where maximum lifespan maps indicate convenient applicability.
3. Vegetative nature-based engineering 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 water table does not exceed the threshold values defined in `RiverArchitect/LifespanDesign/.templates/`*threshold_values.xlsx*.
4. Mineralic nature-based engineering features (rock paving), where the depth to the 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.

Place best vegetation 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 *nature-based engineering (stabilization)* and *nature-based engineering (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 `ProjectName/Geodata/Shapefiles/`. **Check the cell links in the automatically opened cost master workbook's *costs* spreadsheet** (cell links to the *from_geodata* spreadsheet). Finally, save and close the workbook.

42.3 Stabilize terrain

The `Terrain Stabilization` frame enables the identification of areas that require additional support with nature-based engineering features to yield a target lifespan that can be defined in the field *Critical lifespan*. For example, if new terraforms are intended to persist at least 20 years, set *Critical lifespan*=20. A click on the `Stabilize terrain` button launches the calculations, where nature-based engineering features are placed in the same hierarchical order as before. Besides, the terrain stabilization calculates a stable grain size Raster for the provided *Critical lifespan*. The control variables of **,cr* (default 0.047) and Manning's *n* can be defined by clicking on the `Set stability drivers` button. To learn more about the stable grain size computation, please refer to the parameter calculation and stable grain size Raster creation.

The `Terrain Stabilization` produces writes relevant surfaces to the *from_geodata* spreadsheet in costs master file (`ProjectName_assessment_vii.xlsx`) and produces the following geofiles:

- Raster with stable grain (boulder) sizes `ProjectMaker/ProjectName_vii/Geodata/Rasters/terrain_boulder_stab.tif`
- Shapefile with relevant nature-based engineering features (see above definitions) `ProjectMaker/ProjectName_vii/Geodata/Shapefiles/Terrain_stab.shp`

Check the cell links in the automatically opened cost master workbook's *costs* spreadsheet (cell links to the *from_geodata* spreadsheet). Finally, save and close the workbook.

42.4 Manual placement of nature-based engineering features

Additional habitat can be created with cover features (i.e., engineered logs jams or root wads) at locations that result from an expert assessment. To implement cover features, open `ProjectName/Geodata/ProjectName/ProjectMaps.aprx` to do the following:

1. Create a new polygon-shapefile in `ProjectName/Geodata/Shapefiles/` and name it `StreamWood`.
2. Remove the newly created `StreamWood` layer from layout'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 `ProjectName/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 (3.1 m x 3.1 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 elements to the cost master workbook's (`ProjectName_assessment_vii`) *costs* spreadsheet (*nature-based engineering* frame).

42.5 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 to identify 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' [length yard] or m' [length meter] to the cost master workbook's (`ProjectName_assessment_vii`) *costs* spreadsheet; cf. *Civil engineering & other* frame). For later revision, export the drawn paths to a newly created folder (e.g., `ProjectName/Geodata/Shapefiles/`) as polyline shapefile or **.kmz* file. The resulting costs need to be manually entered in the costs master workbook's (`ProjectName_assessment_vii`) *costs* spreadsheet (Civil engineering & other frame).

42.6 Markups, permitting, and other costs

The final project costs include site mobilization and demobilization as well as unexpected costs. Moreover, permitting, markups (such as overhead, profit, and insurance) and engineering fees are added as percentages of costs for construction works at the bottom of the cost master workbook's (`ProjectName_assessment_vii`) *costs* spreadsheet. The total costs for the project proposal are summarized at the top of the *costs* spreadsheet (cell *G2*).

Please note:

- **There is no warranty for the calculated costs (*see disclaimer*).**
- **All workbook-internal cell links must be verified manually, in particular, the `cost (UNITS)` tab.**

CHAPTER

FORTYTHREE

MAPPING OF CONSTRUCTION ELEMENTS

Open `ProjectName/ProjectMaps.aprx` and go to the `Catalog` tab (typically on the right). Click on `> Layouts` and double-click on `ProjectName_assessment_vii`. In the layout's *Contents* tab (typically on the left), select `Layers Map Frame` and double-click on every layer to define the correct dataset source files (`Source` tab) that result from the above-described cost assessment. *Note: Layer groups do not have a* `Source` *tab*. Relevant shapefiles are stored in `ProjectName/Geodata/Shapefiles/` and relevant rasters are stored in `ProjectName/Geodata/Rasters/`. Export the map (*ArcGIS Pro*s `Catalog` -> right-click on `Layouts/ProjectName_assessment_vii` -> select `Export to File...`) to the project folder and name it `ProjectName_assessment_[...].pdf` (proposition for consistent file naming).

CHAPTER

FORTYFOUR

ECOLOGICAL BENEFIT ASESSMENT (CALCULATE SHAREA)

The project costs are vetted against the net gain in annually usable habitat area for target fish species. The GUI's *Calculate Net* ***S****easonal* ***H****abitat* ***Area*** routine calculates usable habitat from rasters that indicate where the composite Habitat Suitability Index (*cHSI*) is higher than a selected threshold value.

44.1 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 SHArea. 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:
  - A workbook template for flow duration curves is available in `RiverArchitect/SHArC/FlowDurationCurves/flow_duration_templates.xlsx`
  - The `GetStarted` tab contains the `Analyze Flow` tool for producing the required format for SHArea calculation in `00_Flows/CONDITION/`.
- The *River Architect*'s *SHArC* module was executed for both situations (pre- and post-project) to obtain *cHSI* rasters.
- 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 m3/s).
  - The corresponding modelling results (flow depth and velocity) were stored in the directories `RiverArchitect/01_Conditions/CONDITION/` and `RiverArchitect/01_Conditions/CONDITION_rea_lyr10/`, respectively. The string `lyr10` refers to terraforming according to the code naming conventions.
  - The *River Architect*'s *SHArC* module applied to both situations with a *cHSI* threshold value of, for example, = 0.5. This threshold value means that all pixels with a *cHSI* value lower than 0.5 were considered as being non-habitat and the *SHArC* module excludes these pixels from the *cHSI* rasters. Thus, the *SHArC* module produced *cHSI* rasters that are stored in:
    * `RiverArchitect/SHArC/SHArea/Rasters/CONDITION/` (existing / pre-project)

    * `RiverArchitect/SHArC/SHArea/Rasters/CONDITION_rea_lyr10/` (with implementation / post-project)

  - The *SHArC* 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 `chju`), the *SHArC* module wrote the usable habitat area and discharge duration to the following workbook: `RiverArchitect/SHArC/SHArea/CONDITION_chju.xlsx`

44.2 Run SHArea 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 SHArea). 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 *SHArC* module (e.g., *Chinook salmon*, *juvenile*). 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 SHArea* button to start the assessment.

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

After running the program, discharge-specific usable habitat area was written to a spreadsheet located in `ProjectName/Geodata/SHArea_evaluation_UNIT.xlsx` (see output section below).

44.3 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 `.../ProjectName_assessment_vii_chju.xlsx`.

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

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

The cells *G3* and *I2/3* in `ProjectName_assessment_vii_FILI.xlsx` state the net gain in SHArea and the project return in units of US$ per acre (or m2 per EUR or any other currency defined) net gain in SHArea (comparison of pre-and post-project condition), respectively.

CHAPTER

FORTYFIVE

RIVER ARCHITECT TOOLS

Tool scripts can be considered as beta versions of future functionalities that will be implemented in *River Architect*. Typically, *Tools* are basic commandline scripts for creative design support. Currently, routines for the hydraulic design of pool-riffle sequences or flood analysis are available, where the flood analysis applies to the U.S. Army Corps of Engineers' `HEC-SPP` software. The *Tools* routines are located in `RiverArchitect/Tools/`.

CHAPTER

FORTYSIX

MAIN TOOL SCRIPTS

Run the following scripts with Esri's *Python* environment (read more)

- `make_annualpeak.py` prepares required input data for statistic flow analyses and with the U.S. Army Corps of Engineers' `HEC-SPP` software.
- `make_flowduration.py` creates flow duration curves (annual averages) for the assessment of AUA.
- `morphologydesigner.py` creates design tables for self-sustaining pool-riffle channels (uses `cHydraulic.py` and `cPoolRiffle.py`).
- `rename_files.py` facilitates renaming input files for conditions according to raster file name conventions by adding, removing or replacing a file name prefix and suffix.
- `run_make_....bat` are a batchfiles that run `make_....py` on Windows x64.
- `run_morphology_designer.bat` is a batchfiles that runs `morphology_designer.py` on Windows x64.

CHAPTER

FORTYSEVEN

CLASS AND FUNCTION FILES

The code execution depends on the following folders and scripts:

- `.templates` folder contains a template workbooks for multiple purposes.
- `Products` folder contains results of any script in this folder.
- `cHydraulic.py` contains a 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.
- `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.

CHAPTER

FORTYEIGHT

ERROR MESSAGES AND TROUBLESHOOTING

CHAPTER

FORTYNINE

KNOWN ISSUES

We do our very best to program *River Architect* as robust and flexible as possible. *River Architect*'s stepwise development nevertheless led to a few restrictions in its freedom of use, which are caused by a couple of hard-code sequences. The following limitations because of hard-coding are known and will be fixed in future versions:

- SHArC: Physical Habitats / fish lifestages must be named either `spawning, fry, ammocoetes, juvenile, adult, hydrologic year, season, depth > x,` or `velocity > x`. The lifestage names are currently hard-coded in the `Fish` class (`RiverArchitect/.site_packages/riverpy/cFish.py`) dictionary `self.ls_col_add = {"spawning": 1, "fry": 3, "ammocoetes": 3, "juvenile": 5, "adult": 7, "hydrologic year": 1, "season": 3, "depth > x": 5, "velocity > x": 7}`. This dictionary can be changed for using other lifestage names and we are working on an improvement in later versions. *Note: This dictionary defines the relative column numbers that is added to the column where a fish name is defined.*
- **ArcGIS Pro 2.4** `Exception Error: (arcpy) - expected 1 arguments, got 2` The latest version of *ArcGIS Pro 2.4* may cause problems when *RiverArchitect* calls the conda environment. Remedy (1): Use `"C:\Program Files\ArcGIS\Pro\bin\Python\envs\arcgispro-py3\python.exe"` instead of `"%PROGRAMFILES%\ArcGIS\Pro\bin\Python\Scripts\propy"` in the `RiverArchitect/Start_River_Architect.bat` (see context in the installation instructions). Remedy (2): In *ArcGIS Pro*, go to *Settings* (bottom left), then *Python*, click on *Manage Environments* and *Clone Default*. Check the clone (`arcgispro-py3-clone`) and close *ArcGIS Pro*.
- **Maximum data size (*River Architect*'s Lifespan & Design mapping hangs without prompting any message)**: *River Architect*'s data processing competence depends on the available Random-Access Memory (RAM). As a general rule, the maximum amount of data should be smaller than the available RAM. The size of processed data can be approximated as:RAM >= *Single Raster size · Number of features · Number of parameters per feature · Number of discharges*The data size limit applies to lifespan mapping only, where *River Architect* loads nearby all available data in the `01_Conditions/CONDITION/` folder for each selected feature to run `arcpy.sa.CellStatistics`. To avoid issues, consider the following rules of thumb:
  - Limit the project area to reasonable extents (single Raster size <= 350 MB).
  - Limit the number of discharges to less than 10 (set relevant discharges defined in rows 4, 11, and 12 of *input files* to less than 10); manual modification of automatically generated input files may be required (rows 4, 11, and 12 of `01_Conditions/CONDITION/input_definitions.inp`).
  - Analyze only one feature at a time rather than feature groups (such as "Plantings"). In this case, the single Raster size may be up to 1 GB.
- **Grain size raster name coherence**: The `LifespanDesign` module will read grain size raster names from `01_Conditions/CONDITION/input_definitions.inp`. However, the `Ecohydraulics` module absolutely requires that a grain size raster in SI units is named `dmean.tif` (units: meters) and a grain size raster in U.S. customary units must be named `dmean_ft.tif` (units: feet).
- **PDF maps export of lifespan** maps fails under some circumstances (*read more in the error message*).

CHAPTER

FIFTY

HOW TO TRACE ERROR AND WARNING MESSAGES

If *River Architect* encounters problems, the code writes *WARNING* and *ERROR* messages to the *Python* command line and to the same logfile where all other computation progress information is logged. The *WARNING* and *ERROR* messages provide information on the error cause and this Wiki contains all possible *WARNING* and *ERROR* messages of *River Architect*.

Thus, for troubleshooting, press the `CTRL` + `F` key, enter (part of) the *WARNING* or *ERROR* message that occurred, and press enter. *Causes* and *Remedies* are provided for every *WARNING* and *ERROR* message. Do not include specific parameters, file names, or expressions in parentheses or brackets.

Otherwise, the *WARNING* and *ERROR* messages are listed in alphabetic order (starting with **E**rror messages, then **E**xceptionalError message, then **W**arning messages . . .).

CHAPTER

FIFTYONE

ERROR AND WARNING MESSAGE GENERATION

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 files is erroneous. The *River Architect* 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 the Error messages. Some non-identifiable errors raised by the `arcpy` package disappear after rebooting the system.

CHAPTER

FIFTYTWO

ERROR MESSAGES

- **`arcpy ERROR 130051: Input feature class is not registered as versioned.`**
  - *Cause:* `arcpy` failed to register datasets, which basically means that the dataset does not exist. Moreover, this arbitrary error may occur if, for example, multiple subsets of a *Condition* were created without closing *River Architect* between each creation.
  - *Remedy:*
    * Make sure that the dataset exists (read error message information).
    * *GetStarted*: When creating *Condition*s, ensure to restart River Architect after every single creation of a *Condition* or *Condition* subset. This issue is related to `arcpy`, which will not register datasets unless *River Architect* is completely closed. We are developing work-arounds.
    * ProjectMaker: Ensure that the project polygon and the selected habitat condition (from SHArC) overlap (otherwise, the intersecting dataset is empty).
- **`ERROR 000641: Too few records for analysis.`**
  - *Cause:* This `arcpy` error message occurs here when `arcpy.CalculateGeometryAttributes_management` tries to compute the area of an empty shapefile.
  - *Remedy:* If this error occurs within the calculation of SHArea (Seasonal Habitat 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, traceback 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 correct *Python* environment (see installation guide)
    * Make sure that all input Rasters are in *GeoTIFF* format and well placed in the folder `RiverArchitect/01_Conditions/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 `SpatialAnalyst` 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_sha(self) of` *SHArC*'s `CHSI()` class in `SHArC/cHSI.py` when it could not calculate the usable habitat area (see SHArea Methods).
  - *Remedy:*
    * Ensure that the SHArea threshold has a meaningful value between 0.0 and 1.0 (SHArC Intro).
    * Ensure that neither the directory `SHArC/.cache/` nor the directory `SHArC/SHArea/` or their contents are in use by other program.
    * Review the input settings according to the SHArC Quick GUIde.
    * Follow up earlier error messages.

- **`ERROR: Bad assignment of x/y values in the coordinate input file.`**
  - *Cause:* Raised by the `coordinates_read(self)` function of the `Info()` class in either `LifespanDesign/cRead InpLifespan.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 `MaxLifespan/.templates/`) correspond to the definitions in *LifespanDesign* Output maps.

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

- **`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: MU calculation failed.`**
  - *Cause:* Error raised by the `MU` class (`GetStarted/cMorphUnits.py`) when the defined flow depth and/or velocity Rasters are not usable.
  - *Remedy:*
    * Ensure that no other program uses the selected in flow depth and velocity Rasters, and *Condition* folder.
    * Manually / visually verify the consistency of the provided input Rasters (*Float* - type).
    * See also next error message (*Baseflow MU update failed.*)
- **`ERROR: MU update failed.`**
  - *Cause:* Error raised by the `MU` class (`GetStarted/cMorphUnits.py`) when processing the provided flow depth and/or velocity Rasters lead to non-meaningful results.
  - *Remedy:*
    * Ensure that the flow depth and velocity Rasters have consistent units.
    * ISSUE: The current code uses U.S. customary units for MU delineation according to Wyrick and Pasternack (2014). If required, manual changes of the delineation functions can be made in `GetStarted/cMorphUnits.py` (`calculate_mu_baseflow` function approximately from line 75 onward). We are working on improving the Morphological Unit generator.
- **`ERROR: Boundary shapefile in arcpy.PolygonToRaster[...].`**
  - *Cause:* Raised the SHArC 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(.../SHArC/HSI/CONDITION/bound_ras)`.
  - *Remedy:* Verify that a the selected boundary shapefile (SHArC computation boundaries) has a valid rectangle and an `Id` field value of `1` for that rectangle.
- **`ERROR: Boundary shapefile provided but [...].`**
  - *Cause:* Raised the SHArC module'`smake_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 `MaxLifespan/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 *ArcGIS Pro* 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 `VolumeAssessment()` class in `VolumeAssessment/cVolumeAssessment.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 the modified DEM Rasters contain valid data and are not used in any other application.
- **`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 *SHArC*'s `CHSI()` class in `SHArC/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 the Conditions / Input Rasters section.
    * Ensure that the HSI Rasters are stored in `.../SHArC/HSI/CONDITION/`, with the correct Raster names including information about the discharge (see the SHArC output section).
- **`ERROR: Cannot load modified DEM.`**
  - *Cause:* The volume difference calculation and mapping of a (CAD-) modified DEM Rasters failed because the `__init__(self, ...)` function of the `VolumeAssessment()` class in `VolumeAssessment/cVolumeAssessment.py` cannot access the Raster.
  - *Remedy:* Ensure that the selected (CAD-) modified DEM Raster exists and that it is not opened in any other program (close *ArcGIS Pro* / *QGIS*).
- **`ERROR: Cannot load original DEM.`**
  - *Cause:* The volume difference calculation and mapping of the selected DEM Rasters failed because the `__init__(self, ...)` function of the `VolumeAssessment()` class in `VolumeAssessment/cVolumeAssessment.py` cannot access the Raster.
  - *Remedy:* Ensure that the selected DEM Raster exists and that it is not opened in any other program (close *ArcGIS Pro* / *QGIS*).
- **`ERROR: Cannot open project: DIR/module_name.aprx.`**

  - *Cause:* Mapping of `module_name` raises this error message when the stated project file (*.aprx*) cannot be opened.
  - *Remedy:* Ensure that the stated is not opened in any other program (*ArcGIS Pro Desktop*) and that the stated project file exists.

- **`ERROR: Could not access Fish.xlsx (...).`**
  - *Cause:* The `get_hsi_curve(self, species, lifestage, par)` function of the `Fish()` class (`.site_packages/riverpy/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 `/SHArC/SHArea/`condition` directory.
  - *Remedy:* Ensure that neither `.site_packages/templates/Fish.xlsx` nor any file in `/SHArC/SHArea/`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 `SHArC/HSI/` and recompile cover HSI Rasters if needed (see CoverHSI).

- **`ERROR: Could not append PDF page XX to map assembly.`**
  - *Cause:* The `make_pdf_maps(self, ...)` function of the `Mapper()` class (`.site_packages/riverpy/cMapper.py`) raises this error when it failed to map the current page (extent).
  - *Remedy:*
    * LifespanDesign / Max Lifespan: Ensure that the definitions in `LifespanDesign/.templates/mapping.inp` are correct.
    * VolumeAssessment: Verify the definitions in the *reach coordinates workbook* and also refer to error message `ERROR: Could not create new project.`
    * General: Ensure that no other program accesses the `LifespanDesign/.cache/`, `MaxLifespan/.cache/`, `MaxLifespan/Output/`, `VolumeAssessment/Output/`, or `02_Maps/*` directories or their contents (close *ArcGIS Pro*).

- **`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 the Conditions / Input Rasters section.

- **`ERROR: Could not calculate stable grain size Raster for ... .`**
  - *Cause:* Error raised by the `ProjectMaker/s30_terrain_stabilization/`'s `main()` function when it cannot calculate the stable grain size for a user-defined minimum (critical) lifespan and *Condition*.
  - *Remedy:* Open `RiverArchitect/logfile.log` and verify that the stated source data exist and are not empty (if necessary, open geofiles in *ArcGIS Pro* and verify contents).

- **`ERROR: Could not create new project (...)`**
  - *Cause:* A module cannot find a project for mapping when it tries to open `aprx_origin = arcpy.mp.ArcGISProject(self.ref_lyt_dir + "module_name.aprx")` or when it tries to save a copy of the source project (`aprx_origin.saveACopy(self.output_mxd_dir + "module_name.aprx")`).

  - *Remedy:* Ensure that the source project `aprx_origin` exists (quoted in ERROR-message brackets) and that the target directory (`to:`) can be accessed (writing rights / close *ArcGIS Pro*). If the source project is missing, re-download it from the source code.

- **`ERROR: Could not create Raster of the project area.`**
  - *Cause:* Raised by `set_project_area(self)` of *ProjectMakers*'s `SHArC()` class in `ProjectMaker/cSHArC.py` or the `main()` function in `ProjectMaker/s20_plantings_delineation.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 (Project Area Polygon preparation).
    * If the Error message occurred within the terrain stabilization process, either restart the Project creation while properly implementing the Project Area Polygon preparation OR manually create a project area Raster (save as `ProjectMaker/ProjectName_vXX/Geodata/Rasters/ProjectArea.tif`, where in-values=`Int(1)` and out-values=`NoData`).

- **`ERROR: Could not create sub-condition folder: ... .`**
  - *Cause:* Raised by `make_sub_condition(...)` (`GetStarted/fSubCondition`) when the defined sub-*Condition* name is invalid.
  - *Remedy:* Ensure that the target sub-*Condition* folder is not in use by any other program and that there are no invalid characters for folder names (such as `$` or `?!`).

- **`ERROR: Could not crop RASTER.`**
  - *Cause:* The `make_sub_condition(...)` (`GetStarted/fSubCondition.py`) function raises this error message when it failed cropping the Raster with the spatial analyst operation `Con(~IsNull(boundary_ras), Float(source_Raster))`.
  - *Remedy:* Verify the consistency of the Rasters in the source condition and make sure that no other program is using data from the source condition or the new sub-*Condition* folder.

- **`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 (`HabitatEvluation/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, ...)` function of the `Mapper()` class (`.site_packages/riverpy/cMapper.py`) raises this error when `LifespanDesign/.templates/mapping.inp` or `MaxLifespan/.templates/mapping.inp` contain invalid xy-coordinates (format).
  - *Remedy:* Ensure the definitions in `LifespanDesign/.templates/mapping.inp` and `MaxLifespan/.templates/mapping.inp` are correct.

- **`ERROR: Could not find / access input Rasters.`**
  - *Cause:* Error raised by the `D2W` class (`GetStarted/cDepth2Groundwater.py`) or the `DET` class (`GetStarted/cDetrendedDEM.py`) or the `MU` class (`GetStarted/cMorphUnits.py`) when the defined DEM, flow depth, or velocity *GeoTIFF*s are not usable.
  - *Remedy:*

    * Ensure that no other program uses the selected in DEM/flow depth Raster and *Condition* folder.
    * Manually / visually verify the consistency of the provided input Rasters.
    * Avoid running `d2w.tif`, `mu.tif` and `det.tif` creation simultaneously and for multiple conditions (the *.cache* folders are locked by individual instantiations).

- **`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 (ProjectMaker Quick GUIde).
    * Ensure that max. lifespan Rasters were generated without errors; if necessary, visually control the consistency of max. lifespan Rasters in `.../MaxLifespan/Output/Rasters/CONDITION_reach_lyr20_plants/` and `.../MaxLifespan/Products/Rasters/CONDITION_reach_lyr20_plants/` or `...nature-based engineering` (cf. Project Maker vegetation plantings and supporting nature-based engineering features).

- **`ERROR: Could not find any worksheet.`**
  - *Cause:* Error raised by the `open_wb(self)` function of the `Read()` class in `RiverArchitect/.site_packages/riverpy/cInputOutput.py` or the `MakeFlowTable` class (`RiverArchitect/.site_packages/riverpy/cMakeTable.py`) when the concerned workbook contains errors.
  - *Remedy:*
    * Ensure the correct usage of `.site_packages/templates/Fish.xlsx` (SHArC Fish section).
    * Ensure the correct adaptation of `ProjectMaker/.../Project_assessment_vii.xlsx` (ProjectMaker's Cost quantity workbook).
    * Open `SHArC/SHArea/CONDITION_sharea_FILI.xlsx`, verify the data contents and make sure that the file is not opened in any other program.

- **`ERROR: Could not find hydraulic Rasters for CONDITION.`**
  - *Cause:* Error raised by the `ProjectMaker/s30_terrain_stabilization/`'s `main()` function when it cannot find hydraulic Rasters for the selected lifespan-design condition.
  - *Remedy:* Ensure that the hydraulic Rasters used for generating Lifespan and Design Rasters are available in *RiverArchitect/*`01_Conditions/CONDITION/` (also: close all other programs that potentially use these Rasters).

- **`ERROR: Could not find Lifespan Raster (...).`**
  - *Cause:* Error raised by the `ProjectMaker/s30_terrain_stabilization/`'s `main()` function when it cannot find lifespan Rasters.
  - *Remedy:* Go to the LifespanDesign tab for the selected *Condition*. Create lifespan maps for the `Grains / Boulders` feature (creates `lf_grains.tif`). Terrain stabilization absolutely requires `Grains / Boulders` (`lf_grains.tif`), and optionally requires `Streamwood` (`lf_wood.tif`) and `Nature-based engineering features (other)` (`lf_bio.tif`).

- **`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 `RiverArchitect/.site_packages/riverpy/cReachManager.py`) when the `extents` sheet in the reach coordinate workbook (`ModifyTerrain/.templates/computation_extents.xlsx`) could not be read.

- *Remedy:* Ensure the correct setup of `ModifyTerrain/.templates/computation_extents.xlsx` (*reach preparation within the* ModifyTerrain *module*).

- **`ERROR: Could not find the cover input geofile [...]`**
  - *Cause:* Error raised by the `__init__(self, ...)` function of the `CovHSI(HHSI)` class in `SHArC/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 the SHArC Working Principles.

- **`ERROR: Could not interpolate exceedance probability of Q = [...]`**
  - *Cause:* Raised by `interpolate_flow_exceedance(self, Q_value)` of the `FlowAssessment()` class in `.site_packages/riverpy/cFlows.py` when the flow duration curve contains invalid data.
  - *Remedy:* Ensure the correct setup of the used flow duration curve in `00_Flows/CONDITION/flow_duration ... .xlsx`. 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 the HHSI input preparation for details.

- **`ERROR: Could not load ... .`**
  - *Cause:* Raised by several functions, when they cannot find a user-defined or built-in template file.
  - *Remedy:* Ensure that the stated file exists, can be opened (is not corrupted) and no other program uses the stated file.

- **`ERROR: Could not load boundary Raster: ... .`**
  - *Cause:* Raised by `make_sub_condition(...)` (`GetStarted/fSubCondition`) when the defined boundary Raster contains invalid data.
  - *Remedy:* Ensure that the spatial boundary was correctly delineated (similar as described in Project Area Polygon preparation) and converted to an Integer Raster: On-values have to be Integer 1 and Off-values have to be Integer 0. The sub-*Condition* creation only considers Pixels with On-values (Integer 1).

- **`ERROR: Could not load existing Raster of the project area.`**
  - *Cause:* Raised by `set_project_area(self)` of *ProjectMakers*'s `SHArC()` class in `ProjectMaker/cSHArC.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 (Project Area Polygon preparation).
    * Manually inspect the project delineation Raster.

- **`ERROR: Could not load newly created Raster of the project area.`**
  - *Cause:* Raised by `set_project_area(self)` of the *ProjectMaker*'s `SHArC()` class in `ProjectMaker/cSHArC.py` when the converted the project area shapefile is corrupted.
  - *Remedy:* Ensure that the project was correctly delineated (Project Area Polygon preparation).

- **`ERROR: Could not open workbook (...).`**
  - *Cause:* Error raised by the `__init__(self)` function of the `Read()` class in `riverpy/cInputOutput.py` or the `write_flow_duration2xlsx(self, ...)` function of the `SeasonalFlowProcessor()` class (`riverpy/cFlows.py`) when the concerned workbook contains errors or cannot be opened for other reasons.

  - *Remedy:*
    - ProjectMaker: Ensure the correct usage of the concerned workbook (*ProjectMaker* Wiki).
    - Flow generator: Ensure that the template workbook in the parentheses is not opened in any other program and that the template workbook was not modified.

- **`ERROR: Could not perform spatial radius operations [...].`**
  - *Cause:* The `spatial_join_analysis(self, rater, curve_data)` function of the `CovHSI(HHSI)` class (`SHArC/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 CoverHSI.

- **`ERROR: Could not process flow series.`**
  - *Cause:* Error raised by the `build_flow_duration_data(self.)` (`SeasonalFlowProcessor()` in `RiverArchitect/.site_packages/riverpy/cFlows.py`) when it could not read the provided flow series data file.
  - *Remedy:* Ensure that the format of the provided flow series workbook corresponds to the provided example data file (`00_Flows/InputFlowSeries/flow_series_example_data.xlsx`).

- **`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 annually usable habitat area for a condition or (set of) discharge(s).
  - *Remedy:* Ensure that the variable (parameters) are properly defined according to ProjectMaker Quick GUIde and that the *SHArC* module contains the required information.

- **`ERROR: Could not read flow duration curve data.`**
  - *Cause:* Raised by `get_flow_data(self, ...)` of the `FlowAssessment()` class in `.site_packages/riverpy/cFlows.py` when the flow duration curve contains invalid data.
  - *Remedy:*
    - Used within `SHArC`: Ensure that a flow duration curve for all selected *Physical Habitats* and the select hydraulic *Condition* was generated. The flow duration curves are typically saved as `00_Flows/CONDITION/flow_duration_FILI.xlsx`, where `FILI` is a placeholder of the first two letters of the selected species and lifestage (*Physical Habitat*), respectively. For example for *Chinook Salmon - Juvenile*, `FILI` becomes `chju`, or for *All Aquatic - hydrologic year*, `FILI` becomes `alhy`.
    - Ensure that the provided discharge series contains valid data formats as indicated in the example workbook (`00_Flows/InputFlowSeries/flow_series_example_data.xlsx`).

- **`ERROR: Could not read parameter type [...] from Fish.xlsx.`**
  - *Cause:* The `get_hsi_curve(self, species, lifestage, par)` function of the `Fish()` class (`.site_packages/riverpy/cFish.py`) and the `SeasonalFlowProcessor` class (`riverpy/cFlows.py`) raise this error message when it cannot read a habitat suitability curve from `Fish.xlsx`.
  - *Remedy:*
    - Ensure that `.site_packages/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 SHArC Fish section.

    * Ensure that the lifestages defined in `.site_packages/templates/Fish.xlsx` are part of the following dictionary: `self.ls_col_add = {"spawning": 1, "fry": 3, "ammocoetes": 3, "juvenile": 5, "adult": 7, "hydrologic year": 1, "season": 3, "depth > x": 5, "velocity > x": 7, "rearing": 7}`. **All lifestages must be defined in `self.ls_col_add`, as relative column number to the 0-column of every Fish species.** For example, for adding a `"rearing"` lifestage for `Chinook salmon` (one of the default species), `"rearing"` must replace `"adult"`, which is in the relative 7th (8-1 for the 0-column) of the Chinook salmon species. If lifestages were otherwise modified, either consider using one of the lifestages defined in the dictionary or modify the dictionary (`self.ls_col_add`) in the `Fish` class (`.site_packages/riverpy/cFish.py`).

- **`ERROR: Could not read source file.`**
  - *Cause:* The `Read` class (`RiverArchitect/.site_packages/riverpy/cInputOutput.py`) or `make_sub_condition` (`GetStarted/fSubCondition`) raise this error when the data from the provided workbook or Raster, respectively, cannot be processed.
  - *Remedy:*
    * `riverpy`: Ensure that the provided workbook respects the input requirements described in the instructions of the applied module.
    * `GetStarted/fSubCondition`: Verify the consistency of the Rasters in the source condition and make sure that no other program is using data from the source condition.
- **`ERROR: Could not read source project (...)`**
  - *Cause:* A module cannot find a project for mapping when it tries to open `aprx_origin = arcpy.mp.ArcGISProject(self.ref_lyt_dir + "module_name.aprx")`.
  - *Remedy:* Ensure that the source project `aprx_origin` exists (quoted in ERROR-message brackets) and not opened in any other application (e.g., close *ArcGIS Pro*). If the source (origin) project is missing, re-download it from the source code.
- **`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 (`ModifyTerrain/cModifyTerrain.py`) or the `make_volume_diff_rasters(self)` function of the `VolumeAssessment()` class (`VolumeAssessment/cVolumeAssessment.py`) raise 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 *reach preparation within the* ModifyTerrain *module* for correct reach definitions.
    * If the *ModifyTerrain* module is externally loaded, ensure the correct definition of features and feature shortnames (see the alternative run options for the *ModifyTerrain* module).
- **`ERROR: Could not run SHArea analysis.`**
  - *Cause:* The `main()` function in `ProjectMaker/s40_compare_wua.py` raises this error message when it could not calculate SHArea.
  - *Remedy:* Traceback warning and other error messages. Ensure the correct definition of parameters, creation of required geodata, and file naming (*ProjectMaker* Wiki))
- **`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 *SHArC*'s `CHSI()` class in `SHArC/cHSI.py` when the calculated cHSI Raster is empty or corrupted.
  - *Remedy:*
    * Ensure that neither the directory `SHArC/.cache/` nor the directory `SHArC/SHArea/` or their contents are used by another program.
    * Review the input settings according to SHArC Quick GUIde.

- **`ERROR: Could not save cover / H HSI [...] Raster ...`**
  - *Cause:* Raised by `make_hhsi(self, fish_applied)` of *SHArC*'s `HHSI()` class in `SHArC/cHSI.py` when the calculated HHSI Raster is empty or corrupted.
  - *Remedy:*
    * Ensure that no other software uses data from neither the `SHArC/` nor the `RiverArchitect/01_Conditions/` directories.
    * Review the input flow velocity and depth Rasters according to the Conditions / Input Rasters section.

- **`ERROR: Could not save RASTER .`**
  - *Cause:* Raised by `save_tif(self, ...)` of [*GetStarted*][11]'s `ConditionCreator()` class in `GetStarted/cConditionCreator.py` or `make_sub_condition` (`GetStarted/fSubCondition.py`) when a Raster file could not be saved to a New or Sub *Condition*, respectively.
  - *Remedy:*
    * Ensure that no other program uses the new `RiverArchitect/01_Conditions/` directory.
    * Visually verify the consistency of provided input Rasters.

- **`ERROR: Could not save WORKBOOK.`**
  - *Cause:* The `main()` function in `ProjectMaker/s40_compare_wua.py` or the `write_flow_duration2xlsx(self, ...)` function of the `SeasonalFlowProcessor()` class (`riverpy/cFlows.py`) raise this error message when they could not save the `WORKBOOK`.
  - *Remedy:* Ensure that the target workbook can be overwritten, has valid contents, and is not opened by another program. Traceback earlier warning and error messages.

- **`ERROR: Could not save SHArea-CHSI Raster.`**
  - *Cause:* Raised by `calculate_sha(self)` of *SHArC*'s `CHSI()` class in `SHArC/cHSI.py` when the calculated cHSI Raster is empty or corrupted.
  - *Remedy:*
    * Ensure that the SHArea threshold has a meaningful value between 0.0 and 1.0 (SHArC Intro).
    * Ensure that neither the directory `SHArC/.cache/` nor the directory `SHArC/SHArea/` or their contents are in use by other programs.

    * Review the input settings according to SHArC Quick GUIde.

- **ERROR: Could not set arcpy environment (permissions and licenses?).**
  - *Cause:* A script using `arcpy` and/or `arcpy.sa` could not set up the work environment.
  - *Remedy:* Either the wrong *Python* interpreter was used or `arcpy` and/or a *Spatial Analyst* extension are not available. Read more on setting up the Python environment.
- **ERROR: Could not transfer net SHArea gain.**
  - *Cause:* The `main()` function in `ProjectMaker/s40_compare_wua.py` raises this error message when it could not copy the calculated SHArea from `SHArea_evaluation_unit.xlsx` to **`REACH_stn_costs_vii.xlsx`**.
  - *Remedy:* Open `SHArea_evaluation_template_unit.xlsx` and verify the calculated values. Traceback potential error sources in the CHSI Rasters `/SHArC/` folder and other error messages.
- **ERROR: Could not transfer SHArea data for [FISH].**
  - *Cause:* The `main()` function in `ProjectMaker/s40_compare_wua.py` raises this error message when it could not retrieve SHArea data from the `/SHArC/SHArea/` module to `SHArea_evaluation_UNIT.xlsx`.
  - *Remedy:* Open `SHArea_evaluation_TEMPLATE_UNIT.xlsx` and verify the calculated values. Traceback potential error sources in the CHSI Rasters `/SHArC/` folder and other error messages.
- **ERROR: Could not write flow data set.**
  - *Cause:* Error raised by the `write_flow_duration2xlsx(self, ...)` function of the `SeasonalFlowProcessor()` class (`riverpy/cFlows.py`) when it could not write flow duration data set to `RiverArchitect/00_Flows/CONDITION/flow_duration_FILI.xlsx`.
  - *Remedy:* Close all applications that may use `00_Flows/CONDITION/flow_duration_FILI.xlsx` and traceback earlier warning and error messages.
- **ERROR: Could not write flow duration curve data (...).**
  - *Cause:* Error raised by the `make_flow_duration(self, ...)` or `make_condition_flow_duration(self, ...)` functions (`SeasonalFlowProcessor()` in `RiverArchitect/.site_packages/riverpy/cFlows.py`) when it could not write flow duration curve data to `RiverArchitect/00_Flows/CONDITION/flow_duration_FILI.xlsx`.
  - *Remedy:* Close all applications that may use `00_Flows/CONDITION/flow_duration_FILI.xlsx`.
- **ERROR: Could not write value to CELL [...]**
  - *Cause:* Error raised by the `save_close_wb(self, ...)` (`Write()` in `RiverArchitect/.site_packages/riverpy/cInputOutput.py`) or `write_data_cell(self, column, row, value)` (`MakeTable` in `.site_packages/riverpy/cMakeTable.py`) when it cannot write a value to `RiverArchitect/SHArC/SHArea/`condition_FILI.xlsx`.
  - *Remedy:* Close all applications that may use `RiverArchitect/SHArC/SHArea/CONDITION_FILI.xlsx`. Detailed information on *SHArC* workbook outputs are available in the HHSI input preparation section.
- **ERROR: Could not write SHArea data for [FISH].**
  - *Cause:* The `main()` function in `ProjectMaker/s40_compare_wua.py` raises this error message when it could not write the calculated SHArea to when it cannot write a value to `SHArea_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 *SHArC*'s`CovHSI(HHSI)` class in `SHArC/cHSI.py` when the cover HSI Raster calculation failed.
  - *Remedy:* Ensure that the input geofiles (Raster or shapefile) are correctly set up according to CoverHSI.

- **`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 `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 checking *PAR* of *FEATURE*.`**
  - *Cause:* Special case of **`ERROR: Function analysis`**, which may occur after code modifications.
  - *Remedy:*
    - Make sure that the `self.parameter_lists` of features (Code Extension / Extend Features within the *LifespanDesign* module) 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` (Add analysis within the *LifespanDesign* module).

- **`ERROR: Failed to access [...].`**
  - *Cause:* Error raised by the `open_wb(self, ...)` function of the `Read()` (or the inheriting `Write()`)class in `RiverArchitect/.site_packages/riverpy/cInputOutput.py` or the `MakeFlowTable` class in `RiverArchitect/.site_packages/riverpy/cMakeTable.py`) when a workbook could not be opened.
  - *Remedy:*
    - Ensure that the listed workbook is not opened in any other program and that the workbook is correctly installed according to the module descriptions.
    - Used within `SHArC`:
      - Ensure that a flow duration curve for all selected *Physical Habitats* and the select hydraulic *Condition* was generated. The flow duration curves are typically saved as `00_Flows/CONDITION/flow_duration_FILI.xlsx`, where `FILI` is a placeholder of the first two letters of the selected species and lifestage (*Physical Habitat*), respectively. For example for *Chinook Salmon - Juvenile*, `FILI` becomes `chju`, or for *All Aquatic - hydrologic year*, `FILI` becomes `alhy`.
      - If the module is executed repeatedly for the same condition and fish species-lifestage, and the workbook (`SHArC/SHArea/CONDITION_sharea_FILI.xlsx`) has been manually deleted, restore it by clicking on the `Make HSI Raster (...)` dropdown menu > Select

“Hydraulic” or “Cover” options to open the popup window > Select a *Condition* > Wait until the green button `View discharge dependency workbook` becomes active > `RETURN to MAIN WINDOW` and repeat habitat raster, raster combination and/or *SHArea* analysis.

· Open `SHArC/SHArea/CONDITION_sharea_FILI.xlsx`, verify the data contents and make sure that the file is not opened in any other program.

- **`ERROR: Failed to access computation_extents.xlsx.`**
  - *Cause:* Error raised by the `get_reach_coordinates(self, internal_reach_id)` function of the `Read()` class in `RiverArchitect/.site_packages/riverpy/cReachManager.py`) when the reach coordinate spreadsheet (`ModifyTerrain/.templates/computation_extents.xlsx`) could not be read.
  - *Remedy:* Ensure the correct setup of `ModifyTerrain/.templates/computation_extents.xlsx` (*reach preparation within the* ModifyTerrain *module*).
- **`ERROR: Failed to access / Failed to load [...]`**
  - *Cause:* (1) Error raised by the `open_wb(self)` and `make_condition_xlsx(self, fish_sn)` functions of the `MakeTable()` class in `.site_packages/riverpy/cMakeTable.py`) when the template workbook contains errors. (2) Error raised by the *GetStarted* module’s `ConditionCreator` (`cConditionCreator.py`).
  - *Remedy:*
    * Ensure the correct usage of `.site_packages/templates/Fish.xlsx` (SHArC Fish section) and the completeness of `SHArC/.templates/Q_sharea_template_si.xlsx` and `SHArC/.templates/Q_sharea_template_us.xlsx`. If either template workbook is corrupted or does not exist, re-install missing files.
    * *GetStarted:* Ensure that the provided Raster exists.
- **`ERROR: Failed to access WORKBOOK.`**
  - *Cause:* Error raised by the `write_volumes(self, ...)` function of the `Writer()` class in `RiverArchitect/.site_packages/riverpy/cReachManager.py`) or the `__init__(..)` function of the `Read()` class in `RiverArchitect/.site_packages/riverpy/cInputOutput.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 *SHArC*’s `HHSI()` class in `SHArC/cHSI.py` when is could not find hydraulic input Rasters.
  - *Remedy:*
    * Ensure that no other software uses data from neither the `SHArC/` nor the `RiverArchitect/01_Conditions/` directories.
    * Review the input flow velocity and depth Rasters according to the Conditions / Input Rasters section.
- **`ERROR: Failed to create WORKBOOK.`**
  - *Cause:* Error raised by the `write_volumes(self, ...)` function of the `Writer()` class in `.site_packages/riverpy/cReachManager.py`) when the `template` it could not add new sheets in `VolumeAssessment/Output/CONDITION_volumes.xlsx` or write to copies of `VolumeAssessment/templates/volume_template.xlsx`.
  - *Remedy:* Traceback earlier error messages, ensure that no other program locked `VolumeAssessment/Output/CONDITION_volumes.xlsx` and ensure that `VolumeAssessment/templates/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 `.site_packages/riverpy/cFish.py` when `.site_packages/templates/Fish.xlsx` is opened by another program or non-existent.
  - *Remedy:* Ensure that the file `.site_packages/templates/Fish.xlsx` exists and close any software that may use the workbook.
- **`ERROR: Failed to load [...].`**
  - *Cause:* Raised by the `make_condition_xlsx(self, ...)` function of the `MakeTable()` class in `.site_packages/riverpy/cMakeTable.py` when it could not copy a template workbook for creating fish-lifestage-specific flow tables (copy of `empty.xlsx`).
  - *Remedy:* Ensure that all template files in `SHArC/.templates/` exist as provided with the installation files (in particular: `empty.xlsx`, `CONDITION_sharea_template_si.xlsx`, `CONDITION_sharea_template_us.xlsx`, and `flow_duration_template.xlsx`).
- **`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/cReachManager.py`) when the reach coordinate spreadsheet (`ModifyTerrain/.templates/computation_extents.xlsx`) contains invalid data.
  - *Remedy:* Ensure correct setup of `ModifyTerrain/.templates/computation_extents.xlsx` (*reach preparation within the* ModifyTerrain *module*).
- **`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/thresh old_values.xlsx`) is not accessible or does not contain values for *Depth to water table (min) / max* contains invalid data.
  - *Remedy:* Ensure the correct setup of `LifespanDesign/.templates/threshold_values.xlsx` (Modify Threshold Values within the *LifespanDesign* module). Note that *ModifyTerrain* starts reading depth to water table values column by column, until it meets a non-numeric value.
- **`ERROR: Failed to save PDF map assembly.`**
  - *Cause:* The `make_pdf_maps(self, ...)` function of the `Mapper()` class (`.site_packages/riverpy/cMapper.py`) raises this error when the map assembly is corrupted.
  - *Remedy:*
    * PDF map export of Lifespan maps has troubles at the moment. We recommend to open the automatically created project `RiverArchitect/02_Maps/CONDITION/maps_CONDITION_lyrXY.aprx` and to export PDF maps in *ArcGIS Pro*: (1) Go to desired layout tab; (2) Ensure map layout fits desired export; (3) Go to the `Share` ribbon and click on the green arrow `Layout` (export layout).
    * Ensure that no other program accesses the `LifespanDesign/.cache/`, `MaxLifespan/.cache/`, `ModifyTerrain/.cache/`, `LifespanDesign/Output/`, `MaxLifespan/Output/`, or `ModifyTerrain/Output/` directories or their contents (close *ArcGIS Pro* and verify read/write rights for `RiverArchitect/02_Maps/CONDITION/`).
- **`ERROR: Failed to save WORKBOOK.`**
  - *Cause:* Raised by `calculate_sha(self)` of *SHArC*'s `CHSI()` class in `SHArC/cHSI.py` when it could not save `CONDITION_FILI.xlsx`.

   - *Remedy:* Ensure that no other software uses `SHArC/SHArea/CONDITION_FILI.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 (`ModifyTerrain/cModifyTerrain.py`) or the creation of volume difference Rasters in the `make_volume_diff_rasters(self)` function of the `VolumeAssessment()` class (`VolumeAssessment/cVolumeAssessment.py`) raise this error when the reach extents defined in `ModifyTerrain/.templates/computation_extents.xlsx` are not readable.
   - *Remedy:* Follow the instructions in *reach preparation within the* ModifyTerrain *module* for correct reach definitions.

- **`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 the River Design and Restoration Features pages. 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/cDefinit ions.py`. A new feature also requires modifications of the `RiverArchitect/LifespanDesign/.templates/threshold_values.xlsx` spreadsheet (Modify Threshold Values within the *LifespanDesign* module), 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 (MaxLifespan Quick GUIde) and if needed re-run lifespan/design Raster Maker.

- **`ERROR: Flow series analysis failed.`**
   - *Cause:* The `make_flow_duration(self, ...)` function in `riverpy/cFlows.py` (`SeasonalFlowAssessment` class) raises this error message when executing the `self.build_flow_duration_data(self)` function failed.
   - *Remedy:* The provided flow series workbook may be erroneous. Ensure that the workbook follows the formats provided in `00_Flows7InputFlowSeries/flow_series_example_data.xlsx` and follow up previous warning and error messages.

- **`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 input Raster `RAS` (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 the Conditions / Input Rasters section.
      * Verify that the Rasters defined in `01_Conditions/CONDITION/input_definitions.inp` (lines 8 to 18) correspond to the GRID Raster names in the select conditions folder in `01_Conditions/`.

- **`ERROR: Input file not available.`**
   - *Cause:* Raised by `get_line_entries(self, line_no)` function of the `Info()` class in `LifespanDesign/cRead InpLifespan.py` when it cannot access input files.

  - *Remedy:*
    - Ensure that the file `01_Conditions/CONDITION/input_definitions.inp` exists in the directory `LifespanDesign/.templates/` corresponding to the definitions in the Input definitions files of the *LifespanDesign* module.
    - Ensure that the file `mapping.inp` exists in the directory `LifespanDesign/.templates/` corresponding to the definitions in *LifespanDesign* Output maps.
    - In the case of doubts: Replace `01_Conditions/CONDITION/input_definitions.inp` and `mapping.inp` with the original files and re-apply modifications strictly following the Conditions / Input Rasters section.

- **`ERROR: Input Rasters contain invalid data.`**
  - *Cause:* Error raised by the `D2W` class (`GetStarted/cDepth2Groundwater.py`) or the `DET` class (`GetStarted/cDetrendedDEM.py`) when the defined DEM or flow depth *GeoTIFF* are not usable.
  - *Remedy:*
    - Ensure that no other program uses the selected in DEM/flow depth Raster and *Condition* folder.
    - Manually / visually verify the consistency of the provided DEM / flow depth Rasters.
    - Avoid running both routines simultaneously and for multiple *Conditions* (there is only one *.cache* folder that is locked by every process call).

- **`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.
    - Verify Raster and corresponding lifespan definitions in `01_Conditions/CONDITION/input_definitions.inp` (Input definitions files of the *LifespanDesign* module).

- **`ERROR: Invalid cell assignment for discharge / Rasters.`**
  - *Cause:* Error raised by the `make_condition_xlsx(self, fish_sn)` function of the `MakeTable()` class in `.site_packages/riverpy/cMakeTable.py`) when it cannot write discharge values to `RiverArchitect/SHArC/SHArea/CONDITION_FILI.xlsx`.
  - *Remedy:* Ensure that the flow duration curve is well defined (see the HHSI input preparation) and that `RiverArchitect/SHArC/SHArea/CONDITION_FILI.xlsx` is not used by any other application.
  -

- **`ERROR: Invalid date and / or flow ranges in discharge series.`**
  - *Cause:* Error raised by the `make_flow_season_data(self, ...)` function (`SeasonalFlowProcessor` class in `.site_packages/riverpy/cFlows.py`) when the data provided in the flow series workbook cannot be interpreted.
  - *Remedy:* Ensure that the provided flow series workbook corresponds to the formats defined in the example data set (`00_Flows/InputFlowSeries/flow_series_example_data.xlsx`). In particular, the date and discharge columns and start rows are hard-coded (we will work on the issue in future . . .) and need to correspond those in the sample data set.

- **`ERROR: Invalid date assignment (Fish.xlsx).`**
  - *Cause:* Error raised by the `get_season_dates(self, ...)` function of the `Fish()` class in `.site_packages/riverpy/cFish.py` ot the `build_flow_duration_data(self)` function (`SeasonalFlowProcessor` class in `.site_packages/riverpy/cFlows.py`) when either function cannot read the dates defined in `.site_packages/templates/Fish.xlsx`.
  - *Remedy:* Ensure that the date assignment in `.site_packages/templates/Fish.xlsx` corresponds to template conditions.
- **`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` (Modify Threshold Values within the *LifespanDesign* module).
    * 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 `MakeTable()` or `Write()` class in `.site_packages/riverpy/cMakeTable.py`) or `RiverArchitect/.site_packages/riverpy/cInputOutput.py`), respectively, when it cannot save a workbook (typically `RiverArchitect/SHArC/SHArea/CONDITION_FILI.xlsx` or a copy of the cost master workbook).
  - *Remedy:*
    * *SHArC*: Close all applications that may use `CONDITION_FILI.xlsx` and ensure that its template exists. Detailed information on *SHArC* workbook outputs are available in the HHSI input preparation.
    * *ProjectMaker*: Close all applications that may use the cost master workbook (`REACH_stn_costs_VERSION.xlsx`) and ensure that it exists. Detailed information is available in ProjectMaker Cost quantity section.
- **`ERROR: Invalid interpolation data type (type(Q flowdur) == ...)`**
  - *Cause:* Raised by `interpolate_flow_exceedance(self, Q_value)` of *SHArC*'s `FlowAssessment()` class in `SHArC/cFlows.py` when the flow duration curve contains invalid data.
  - *Remedy:* Ensure the correct setup of the used flow duration curve in `SHArC/FlowDurationCurves/`. The file structure must correspond to that of the provided template `flow_duration_template.xlsx`. Review the HHSI input preparation for details.
- **`ERROR: Invalid x-y coordinates in extent source [mapping.inp / reaches].`**
  - *Cause:* The `make_pdf_maps(self, ...)` function of the `Mapper` class in `.site_packages/riverpy/cMapper.py` raises this error when either the `LifespanDesign/.templates/mapping.inp`, `MaxLifespan/.templates/mapping.inp`, or `ModifyTerrain/.templates/computation_extents.xlsx` contains invalid map definitions (extents).
  - *Remedy:* Ensure that the extent definitions in `LifespanDesign/.templates/mapping.inp` and/or *`ModifyTerrain/.templates/computation_extents.xlsx`* are valid numeric values according to the module descriptions.
- **`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"`, `"nature-based engineering"`, or `"maintenance"`. The error may occur either after code modifications or when`geo_file_maker(condition, feature_type, *args)` in `MaxLifespan/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 the MaxLifespan module code modification possibilities.
    * Ensure that external calls of `geo_file_maker(condition, feature_type, *args)` contain an acceptable `feature_type` (i.e., `feature_type=` either `"terraforming"`, `"plantings"`, `"nature-based engineering"`, or `"maintenance"`).
- **`ERROR: Invalid reference map for [...] (skipping).`**
  - *Cause:* Error raised by mapping routines when the reference map is not present in the mapping project template or the *Condition* project file (*.aprx*).
  - *Remedy:* Verify that the project file contains all default layouts and maps (layout and map names should be identic).
- **`ERROR: Invalid volume name.`**
  - *Cause:* Error raised by the `write_volumes(self, ...)` function of the `Writer()` class in `.site_packages/riverpy/cReachManager.py`, when the `template` sheet in the output (or template) workbook (`VolumeAssessment/Output/CONDITION_volumes.xlsx` or `VolumeAssessment/.templates/volume_template.xlsx`) are inaccessible.
  - *Remedy:* Ensure that `VolumeAssessment/Output/CONDITION_volumes.xlsx` or `VolumeAssessment/.templates/volume_template.xlsx` are not opened in any other program.
- **`ERROR: Lifespan data fetch failed.`**
  - *Cause:* The `get_lifespan_data(self)` or`get_design_data(self)` function of the `ArcpyContainer` class in `MaxLifespan/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 MaxLifespan Quick GUIde).
    * Ensure that lifespan and/or design Rasters are in the defined folder.
- **`ERROR: Mapping could not assign xy-values. Undefined zoom.`**
  - *Cause:* Error raised by the `zoom2map(self, xy)` functions of the `Mapper()` class (`.site_packages/riverpy/cMapper.py`) when it receives a bad format of *x-y* values.
  - *Remedy:*
    * *LifespanDesign* or *MaxLifespan*: Ensure the correct format of `mapping.inp` is correct.
    * *ModifyTerrain*: Ensure correct setup of `ModifyTerrain/.templates/computation_extents.xlsx` (*reach preparation within the* ModifyTerrain *module*).
    * For more settings, review the Mapping wiki.
- **`ERROR: Mapping failed.`**
  - *Cause:* The `make_pdf_maps(self, ...)` function of the `Mapper()` class (`.site_packages/riverpy/cMapper.py`) raises this error when it could not create `PDF` maps.

  - *Remedy:* Ensure that all layouts (see module-specific layout names) are consistently defined in `RiverArchitect/02_Maps/templates/river_template.aprx` and/or `RiverArchitect/02_Maps/CONDITION/map_CONDITION_design.aprx`. Note that modifying layout and/or layer names may cause this error. Therefore, the template layer and layout names must not be changed. Read more about modifications of the smybology, legend, map extents, or code modifications in the Mapping wiki.

- **`ERROR: Map layout preparation failed.`**
  - *Cause:* The `prepare_layout(self, ...)` function of the `Mapper()` class (`.site_packages/riverpy/cMapper.py`) raises this error when it encounters problems with either the provided Rasters or the layouts in the *ArcGIS* project file (`.aprx`).
  - *Remedy:*
    - *LifespanDesign*: If a layout was modified (see default list), ensure that links to databases and datasets are correct.
    - *MaxLifespan*: Ensure that all relevant layouts (see default list) are contained in the *ArcGIS* project file (see mapping descriptions). If needed, add new layouts after code modifications (the MaxLifespan module code modification possibilities).
    - VolumeAssessment: Ensure that all relevant layouts (see default list) are contained in the *ArcGIS* project file (see mapping descriptions).

- **`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 `01_Conditions/CONDITION/input_definitions.inp` is corrupted.
  - *Remedy:* Ensure that the file `01_Conditions/CONDITION/input_definitions.inp` exists in the directory `LifespanDesign/.templates/` corresponding to the definitions in the input definitions files. In case of doubts: Replace `01_Conditions/CONDITION/input_definitions.inp` with the original file and re-apply modifications strictly following the Conditions / Input Rasters section.

- **`ERROR: Multiple openings of Fish.xlsx. Close all office apps ...`**
  - *Cause:* Raised by the `assign_fish_names(self)` function of the `Fish()` class in `.site_packages/riverpy/cFish.py` when `.site_packages/templates/Fish.xlsx` is opened by another program or non-existent.
  - *Remedy:* Ensure that the file `.site_packages/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 `SHArC/cFish.py` when `.site_packages/templates/Fish.xlsx` it expected a habitat suitability curve for `par*, but it could not find values..
  - *Remedy:* Ensure that the file `.site_packages/templates/Fish.xlsx` has valid contents according to SHArC Fish section.

- **`ERROR: No project/layout available (...).`**
  - *Cause:* Error raised by mapping routines when map making without project file (*.aprx*) is invoked.
  - *Remedy:* Verify that the project file contains all default layouts.

- **`ERROR: No reference layout found for [...] (skipping).`**
  - *Cause:* Error raised by mapping routines when the geofile to be mapped is not associated with any layout in the template or *Condition* project file (*.aprx*).

   - *Remedy:* Verify that the project file contains all default layouts.

- **ERROR: No reference map found for [...] (skipping).**
   - *Cause:* Error raised by mapping routines when the geofile to be mapped is not associated with any map in the template or *Condition* project file (*.aprx*).
   - *Remedy:* Verify that the project file contains all default layouts (the layout and map names should be identic).
- **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 troubleshooting 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 `02_Maps/CONDITION/`.
- **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 difference 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()` (`ModifyTerrain/cModifyTerrain.py`) or the `VolumeAssessment` class (`VolumeAssessment/cVolumeAssessment.py`) raise 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/` or `VolumeAssessment/Output/PSEUDO_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 `VolumeAssessment/Output/PSEUDO_CONDITION/` or their contents.
- **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 *LifespanDesign* Output maps.
- **ERROR: Shapefile conversion failed.**

  - *Cause:* Raised by `calculate_sha(self)` of *SHArC*'s `CHSI()` class in `SHArC/cHSI.py` when it could not convert the CHSI Raster to a shapefile.
  - *Remedy:*
    * Ensure that the SHArea threshold has a meaningful value between 0.0 and 1.0 (SHArC Intro).
    * Ensure that neither the directory `SHArC/.cache/` nor the directory `SHArC/SHArea/` or their contents are in use by other programs.
    * Review the input settings according to SHArC Quick GUIde.
    * Follow up earlier error messages.

- **`ERROR: TEMPLATE sheet does not exist.`**
  - *Cause:* Error raised by the `make_condition_xlsx(self, fish_sn)` of the `MakeTable()` class in `.site_packages/riverpy/cMakeTable.py`) when the `template` sheet in the output (template) workbooks `SHArC/.templates/CONDITION_sharea_template_[...].xlsx`) are corrupted.
  - *Remedy:* Ensure that `SHArC/.templates/CONDITION_sharea_template_[...].xlsx` contains the `summary` sheet and that `SHArC/.templates/empty.xlsx` is available; re-install the templates if necessary.

- **`ERROR: The provided boundary file (%s) is invalid.`**
  - *Cause:* Raised the GetStarted module's sub-condition creator function (`make_sub_condition` in `RiverArchitect/GetStarted/fSubCondition.py`), when the defined shapefile or Raster contains invalid data.
  - *Remedy:* Ensure that the provided shapefile or Raster fullfils the requirements explained in the GetStarted - Create Sub-Condition section.

- **`ERROR: The Raster name is not coherent with the name conventions. Name correction needed.`**
  - *Cause:* Error raised by the `MakeFlowTable` class (`riverpy/cMakeTable.py`) when a Raster was wrongly identified as being a flow depth Raster because of a violation of file name conventions.
  - *Remedy:* Ensure that all file names in the `01_Conditions/CONDITION/` folder follow the file name conventions. If the *Condition* folder contains other files than flow depth and velocity *GeoTiff*s starting with the letter `h` or `u`, rename these files or remove them from the *Condition* folder.

- **`ERROR: The selected folder does not contain any depth/velocity Raster containing the defined string.`**
  - *Cause:* Error raised by the `ConditionCreator` class (`GetStarted/cConditionCreator.py`) when the defined folder does not contain any valid depth or velocity Raster.
  - *Remedy:*
    * Ensure that no other program uses the selected folder.
    * Manually / visually verify the selected folder and provided settings for creating the *Condition*.

- **`ERROR: The source discharge file contains non-detectable formats (...).`**
  - *Cause:* The date format provided in the hydraulic habitat suitability index flow duration curve preparation file contains invalid date formats.
  - *Remedy:* Ensure that the select flow series file corresponds to the file requirements. Use the provided example data file (`RiverArchitect/SHArC/FlowDurationCurves/flow_series_example_data.xlsx`) as template workbook for preparing flow series. The

flow series workbook must be an `.xlsx` file and the dates must be contained in col. `A`, starting from row `3`. The default format is `DD-MMM-YY` (e.g., 14-June-88 for June 14th, 1988).

- **`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/cReachManager.py`) when it received invalid volume data.
  - *Remedy:* Ensure that no other program uses `VolumeAssessment/Output/CONDITION_volume.xlsx` and traceback 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/cReachManager.py`) when the `template` it could not add new sheets in `VolumeAssessment/Output/CONDITION_volumes.xlsx` or write to copies of `VolumeAssessment/templates/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 `01_Conditions/CONDITION/input_definitions.inp` has a wrong format or is empty.
  - *Remedy:* Ensure that the file `01_Conditions/CONDITION/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 the Input definitions files of the *LifespanDesign* module.
- **`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 `SpatialAnalyst` license is not available.
  - *Remedy:*
    * Make sure that a `SpatialAnalyst` 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 the Conditions / Input Rasters section).
    * 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 the Requirements stated in the *Introduction and Installation*.

* **`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.tif` or a `wt_depth_base.tif` Raster. A `wt_depth_base.tif` 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.tif` 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 the Add Routine section in the *ModifyTerrain* module) 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.tif`.
  - *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.tif` 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 the Requirements stated in the *Introduction and Installation*.
* **`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 *ArcGIS Pro*'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:
    * Launch *ArcGIS Pro* and its implemented Python window (`Geoprocessing` dropdown menu: `Python`).
    * 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.tif" condition = "REPLACE_CONDITION" base_dem = arcpy.Raster("REPLACE_PATHRiverArchitectLifespanDesignInput" + condition + "dem.tif")<br/>` arcpy.gp.overwriteOutput = True`<br/>` arcpy.env.extent = base_dem.extent`<br/>` arcpy.env.workspace = "D:\PythonRiver\Architect\LifespanDesign\Input" + condition + ""`<br/>` zero_ras = Con(IsNull(base_dem), 0, 0)`<br/>` zero_ras.save(zero_ras_str)`
    * Close *ArcGIS Pro*
* **`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 `SpatialAnalyst` license is not available.
  - *Remedy:* See `ExceptionERROR: (arcpy) [...]`

- **`Internal Error: ... .`**
  - *Cause:* Internal Error messages are raised when the code communicates erroneous data between class functions.
  - *Remedy:*
    * If the code was modified, revise or revert changes.
    * If Internal Errors occur that are not known issues as stated in other error messages, please provide us with feedback (use the bug report template or send us an email: river.architect.program [at] gmail.com)

- **`OSError: ....`**
  - *Cause:* `arcpy` raises this error for various reasons; in particular, if code was modified or the code environment is not set up properly.
  - *Remedy:* Check the correct installation of *River Architect* and input files (refer to the Installation and Signposts pages and use the correct Python environment). Make sure that no other program uses files required by *River Architect*.

- **`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 Pro* is not running.
    * Make sure that no other code copy (*Python*) uses these folders.

CHAPTER

FIFTYTHREE

WARNING MESSAGES

- **`WARNING: .cache folder will be removed by package controls.`**
  - *Cause:* Raised by `clear_cache(self)` of *SHArC*'s `CHSI()` class in `SHArC/cHSI.py` when it could not clear and remove the `.cache/` folder.
  - *Remedy:* Ensure that no other software uses the temporary Rasters stored in `SHArC/.cache/`, and if necessary, delete the folder manually after quitting the module.
- **`WARNING: ### is not a number.`**
  - *Cause:* Functions raise this warning message when a `float`-number type was expected, but another, non-convertible format was provided.
  - *Remedy:* Verify if the results are consistent. If problems occur, make sure that the provided input data contain numbers or valid `nan` formats (`NoData` for Rasters and empty cells for workbooks) only.
- **`WARNING: Bad value ( ... ).`**
  - *Cause:* Raised by `calculate_sha(self)` of *SHArC*'s `CHSI()` class in `SHArC/cHSI.py` when a CHSI polygon contains an invalid value.
  - *Remedy:* Review cHSI Rasters `SHArC/CHSI/CONDITION/`.
- **`WARNING: Cannot identify reaches.`**
  - *Cause:* Raised by `VolumeAssessment().__init__(...)` in `VolumeAssessment/cVolumeAssessment.py` when the provided reach names are inconsistent with those listed in `ModifyTerrain/.templates/computation_extents.xlsx`.
  - *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, also see *reach preparation within the* ModifyTerrain *module*).
- **`WARNING: Cannot load ...\LifespanRasterSymbology.lyrx.`**
  - *Cause:* Raised by `prepare_layout(...)` of the `Mapper()` (`.site_packages/riverpy/cMapper.py`) when it cannot load `/RiverArchitect/02_Maps/templates/symbology/LifespanRasterSymbology.lyrx`.
  - *Remedy:* Ensure that `/RiverArchitect/02_Maps/templates/symbology/LifespanRasterSymbology.lyrx` exists and that the layer file is not broken (e.g., by opening the file in *ArcGIS Pro*).
- **`WARNING: Cannot zoom to defined extent.`**
  - *Cause:* Raised by the `zoom2map(...)` function of the `Mapper()` class (`.site_packages/riverpy/cMapper.py`) when it cannot interpret the provided coordinates.
  - *Remedy:*

    * All modules: Ensure that the `mapping.inp` and the *`reach definitions`* files contain valid data.
    * LifespanDesign: Ensure that either a correct background Raster dataset was defined prior to mapping.

- **`WARNING: computation_extents.xls contains too many reach names.`**
  - *Cause:* Raised by `Read().get_reach_info(self, type)` in `.site_packages/riverpy/cReachManager.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, also see *reach preparation within the* ModifyTerrain *module*).
- **`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 access hydraulic Raster extents. Using MAXOF instead.`**
  - *Cause:* Raised by the `D2W` class (`GetStarted/cDepth2Groundwater.py`) when the defined flow depth *GeoTIFF* has no valid extents.
  - *Remedy:* Manually / visually verify the provided flow depth Raster and make corrections / overwrite it if necessary.
- **`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 *ArcGIS Pro* or `explorer.exe`, uses the `MODULE/.cache` folder.
- **`WARNING: Could not clean up PDF map temp_pages.`**
  - *Cause:* The `make_pdf_maps(self, map_name, *args, **kwargs)` functions of the `Mapper()` class (`.site_packages/riverpy/cMapper.py`) creates single `PDF`s of every map image (extents defined in mapping.inp or the *reach coordinates workbook*). These single-page `PDF`s are finally combined into one `PDF` map assembly and the single-page `PDF`s are deleted afterward.
  - *Remedy:* Ensure that no other program is using the `PDF` files in `RiverArchitect/02_Maps/CONDITION/` while mapping is in progress (close *ArcGIS Pro*).
- **`WARNING: Could not crop lifespan Raster extents to wetted area.`**
  - *Cause:* Raised by the `ArcPyAnalysis` class (`LifespanDesign/cLifespanDesignAnalysis.py`) when it failed to crop lifespan maps (Rasters) to the extents of the depth Raster associated with the highest discharge analyzed.
  - *Remedy:*
    * Manually / visually verify the provided flow depth Rasters and make corrections / overwrite it if necessary.
    * If cropping the lifespan Raster to the wetted area of the highest discharge is not desired, apply the following Raster calculation to the highest discharge's flow depth Raster to eliminate `NoData` values

(otherwise, all `NoData` pixels are excluded from the lifespan Raster): `Con((IsNull(hQQQQQQ) == 1), (IsNull(hQQQQQQ) * 0), Float(hQQQQQQ))`

- **`WARNING: Could not divide [...] by [...]`**
  - *Cause:* Raised by `calculate_relative_exceedance(self)` of *SHArC*'s `FlowAssessment()` class in `SHArC/cFlows.py` when the flow duration curve contains invalid data.
  - *Remedy:* Ensure the correct setup of the used flow duration curve in `SHArC/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 the HHSI input preparation for details.
- **`WARNING: Could not find Lifespan Raster (...).`**
  - *Cause:* Raised by the `ProjectMaker/s30_terrain_stabilization/`'s or `ProjectMaker/s21_plantings_stabilization/`'s `main()` function when it cannot find lifespan Rasters.
  - *Remedy:* Go to the LifespanDesign tab for the selected *Condition*. Create lifespan maps for the goup layers plantings, nature-based engineering, and connectivity. ProjectMaker absolutely requires `Grains / Boulders` (`lf_grains.tif`), and optionally requires `Streamwood` (`lf_wood.tif`) and `nature-based engineering (other)` (`lf_bio.tif`).
- **`WARNING: Could not get flow depth Raster properties. Setting [...]`**
  - *Cause:* The `crop_input_Raster(self, ...)` function (`SHArC/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 (`SHArC/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 CoverHSI.
- **`WARNING: Could not identify maximum flow in flow duration curve.`**
  - *Cause:* The `get_flow_duration_data(self, ...)` function (`riverpy/cFlows.py`) prints this warning message when it could not identify a maximum flow for interpolating flow exceedance probabilities.
  - *Remedy:* Open `00_Flows/CONDITION/flow_duration_fili` and verify that the discharges in the first spreadsheet are correct.
- **`WARNING: Could not load /01_Conditions/CONDITION/back.tif - ...`**
  - *Cause:* The `sect_extent()` function (`LifespanDesign/cLifespanDesignAnalysis.py`) prints this warning message when it could not load the `back.tif` Raster in the `CONDITION` folder. This warning message can only occur when the checkbox `Limit computation extent to background (back.tif) Raster` is activated.
  - *Remedy:* Make sure that there is a `back.tif` Raster in the `CONDITION` folder. Otherwise, uncheck the checkbox, but using a background Raster `back.tif` is strongly recommended.
- **`WARNING: Could not reset styles.`**
  - *Cause:* Raised by `Write().write_volumes(self, ...)` in `.site_packages/riverpy/cReachManager.py` when the `template` sheet in the output (template) workbook

(`VolumeAssessment/Output/CONDITION_volumes.xlsx` or `VolumeAssessment/.templates/volume_template.xlsx`) is either locked or not accessible.

- *Remedy:* Ensure that no other program uses `VolumeAssessment/Output/CONDITION_volumes.xlsx` or `VolumeAssessment/.templates/volume_template.xlsx` and that both workbooks have not been accidentally deleted.

- **`WARNING: Could not read project area extents.`**
  - *Cause:* Raised by `SHArC.get_extents(self, ...)` in `ProjectMaker/cSHArC.py` when the function failed to read the project area extents from the `ProjectArea.shp` shapefile.
  - *Remedy:* Ensure that the `ProjectArea.shp` shapefile is correctly created (in particular the *Attributes Table*), according to Project Area Polygon preparation.
- **`WARNING: Could not remove .cache folder.`**
  - *Cause:* Raised by multiple modules, if deleting the temporary `.cacheXXXXXXX` folder failed.
  - *Remedy:* Ensure that no other program uses the cache folder and delete exiting `.cache` folders in the concerned modules manually.
- **`WARNING: Could not set project area extents ().`**
  - *Cause:* Raised by `SHArC().set_env(self)` in `/ProjectMaker/cSHArC.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 Raster creation with the *SHArC* module.
- **`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`01_Conditions/CONDITION/input_definitions.inp`, see the Input definitions files of the *LifespanDesign* module).
  - *Remedy:*
    - Assign a float value to the concerned feature in the `Mobility frequency threshold` row of the `LifespanDesign/.templates/threshold_values.xlsx` spreadsheet (see also Modify Threshold Values within the *LifespanDesign* module).
    - Make sure that the defined defined `Mobility frequency threshold* float is consistent with the defined` Return periods`in`01_Conditions/CONDITION/input_definitions.inp` (see the Input definitions files of the *LifespanDesign* module).
- **`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 align input raster [...].`**
  - *Cause:* Raised by `ConditionCreator().fix_alignment` in `GetStarted/cConditionCreator.py` when it could not align the stated raster.
  - *Remedy:* Verify if the stated Raster contains invalid values (exists). Complex filenames that do not comply with the *River Architect* filename convention may also cause this error. Rename all files with the `Tools/rename_files.py` script (see inline code comments for instructions) to adapt all files to the *River Architect* filename convention and re-run `Align Rasters` (if required).

- **`WARNING: Failed to arrange worksheets.`**
  - *Cause:* Raised by `Write().write_volumes(self, ...)` in `.site_packages/riverpy/cReachManager.py` when it could not bring to the front the latest copy of the `template` sheet in the output (template) workbook (`VolumeAssessment/Output/CONDITION_volumes.xlsx` or `VolumeAssessment/.templates/volume_template.xlsx`), which contains the calculation results.
  - *Remedy:* Traceback earlier error and warning messages. Ensure that no other program uses `VolumeAssessment/Output/CONDITION_volumes.xlsx` or `VolumeAssessment/.templates/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/cReachManager.py` when it could not write volume (numbers) to a copy of the `template` sheet in the output (template) workbook (`VolumeAssessment/Output/CONDITION_volumes.xlsx` or `VolumeAssessment/.templates/volume_template.xlsx`).
  - *Remedy:* Traceback earlier error and warning messages. Ensure that no other program uses `VolumeAssessment/Output/CONDITION_volumes.xlsx` or `VolumeAssessment/.templates/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 *SHArC*'s `FlowAssessment()` class in `SHArC/cFlows.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 be 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 curve data (Fish.xlsx):  PAR= ... , HSI= ... .`**
  - *Cause:* Raised by `HHSI().nested_con_raster_calc(self, ...)` in `SHArC/cHSI.py` when the *HSI* curve data provided in `.site_packages/templates/Fish.xlsx` are non-numeric or otherwise invalid.
  - *Remedy:*

    * Ensure that *Physical Habitats* in `Fish.xlsx` are correctly defined according to the definitions.
    * Ensure that the curve data in `Fish.xlsx` are suitable numeric values for linear interpolation.

- **`WARNING: Invalid feature names for column headers.`**
  - *Cause:* Raised by `Write().write_volumes(self, ...)` in `.site_packages/riverpy/cReachManager.py` when it could not write feature names to a copy of the `template` sheet in the output (template) workbook (`VolumeAssessment/Output/CONDITION_volumes.xlsx` or `VolumeAssessment/.templates/volume_template.xlsx`).
  - *Remedy:*
    * Ensure that `ModifyTerrain/.templates/computation_extents.xlsx` contains valid reach descriptions (*reach preparation within the* ModifyTerrain *module*).
    * Ensure that no other program uses `VolumeAssessment/Output/CONDITION_ volumes.xlsx` or `VolumeAssessment/.templates/volume_template.xlsx` and that both workbooks have not been accidentally deleted.
- **`WARNING: Invalid alignment Raster name ([...]).`**
  - *Cause:* Raised by `ConditionCreator().fix_alignment(self, ...)` in `GetStarted/cConditionCreator.py` when it could not open the stated raster.
  - *Remedy:* Complex filenames that do not comply with the *River Architect* filename convention may also cause this error. Rename all files with the `Tools/rename_files.py` script (see inline code comments for instructions) to adapt all Raster filenames to the *River Architect* filename convention and re-run `Align Rasters` (if required).
- **`WARNING: Invalid type assignment -- setting reach names to IDs.`**
  - *Cause:* Raised by `Read().get_reach_info(self, type)` in `.site_packages/riverpy/cReachManager.py` when The `type` argument is not`full_name*` orId`. In this case, the [*ModifyTerrain*][5] 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__()` (`ModifyTerrain/cModifyTerrain.py`) or the `VolumeAssessment.__init__()` (`VolumeAssessment/cVolumeAssessment.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: Merge operation failed (empty shapefiles?).`**
  - *Cause:* Raised by the `ProjectMaker/s30_terrain_stabilization/`'s `main()` function when the nature-based engineering and Angular Boulder shapefiles for stabilizing the terrain failed.
  - *Remedy:* One or both stabilization shapefiles (`ProjectMaker/ProjectName/Geodata/Shapefiles/Terrain_stab.shp` or `.../Terrain_boulder_stab.shp`) may be empty. Check source data or consider to change the user-defined requried critical lifespan.
- **`WARNING: No Lifespan / Design Raster found (...).`**

   - *Cause:* Raised by the `MaxLifespan`'s `Director` class (`MaxLifespan/cFeatureActions.py`) when it cannot find any valid lifespan or design Raster for a condition.
   - *Remedy:* Make sure to run the *LifespanDesign* module for at least one feature within the selected group for a condition. If necessary, verify the contents of lifespan Rasters in `LifespanDesign/Output/Rasters/CONDITION/` (e.g., empty Rasters and Raster names must contain `lf` or `ds` to be recognized).

- **`WARNING: Non-numeric values in data.`**
   - *Cause:* Raised by the interpolation functions of the `Interpolator` class (`.site_packages/riverpy/cInterpolator.py`) when the x data points that should get an interpolated y-value are not numeric.
   - *Remedy:* Check the condition flow duration and/or time duration curve and remove any non-numeric value from the discharge columns.

- **`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 `[...]` indicates the concerned run task.
   - *Remedy:* Ensure that the logfiles of the concerned module are not opened in any other process or program.

- **`WARNING: PlantExisting.shp is corrupted or non-existent.`**
   - *Cause:* Raised by the `ProjectMaker/s20_plantings_delineation/`'s `main()` function when it could not process the existing plants shapefile.
   - *Remedy:* Ensure that existing plants (`ProjectMaker/ProjectName/Geodata/Shapefiles/PlantExisting.shp`) is correctly set up according to the descriptions in the wiki. However, this raster is not absolutely required.

- **`WARNING: Substituting user-defined crit. lifespan (...) with ....`**
   - *Cause:* Error raised by the `ProjectMaker/s30_terrain_stabilization/`'s `main()` function when the user-defined critical lifespan for terrain stabilization does not exaclty correspond to the available hydraulic Rasters defined for the provided *Condition*.
   - *Remedy:* Ensure to only use lifespans defined for the provided *Condition* (cf. definition of flow duration periods).

- **`WARNING: The provided path to Rasters for mapping is invalid. Using templates instead.`**
   - *Cause:* Raised by the `Mapper()` class in `.site_packages/riverpy/cMapper.py` when it could not process the provided input Raster path.
   - *Remedy:* If code changes were made, ensure that `Mapper()` received correct `CONDITION` and `map_type` arguments. Otherwise, `Mapper()` will use the template Raster files located in `RiverArchitect/02_Maps/templates/Rasters/`.

- **`WARNING: Volume value assignment failed.`**
   - *Cause:* Raised by `Write().write_volumes(self, ...)` in `.site_packages/riverpy/cReachManager.py` when it could not write volume (numbers) to a copy of the `template` sheet in the output (template) workbook (`VolumeAssessment/Output/CONDITION_volumes.xlsx` or `VolumeAssessment/.templates/volume_template.xlsx`).
   - *Remedy:* Traceback earlier error and warning messages. Ensure that no other program uses `VolumeAssessment/Output/CONDITION_volumes.xlsx` or `VolumeAssessment/.templates/volume_template.xlsx` and that both workbooks have not been accidentally deleted.

River Architect was developed in the Pasternack Lab for Watershed Hydrology, Geomorphology, and Ecohydraulics at University of California at Davis, Department of Land, Air, and Water Resources.

River Architect is the result of research projects funded by

- The Yuba Water Agency (Marysville, California, USA) under Award #201016094 and Award #10446)
- The USDA National Institute of Food and Agriculture, Hatch project number CA-D-LAW-7034-H.

The scientific procedures integrated into *River Architect* are the result of a dedicated development team and many researchers who contributed useful hints. Therefore, special thanks goes to Lukas Schmocker, Isabella Schalko, Virginia Ruiz-Villanueva, Carmelo Juez, Fabian Friedl and John Stella for their inputs regarding wood introduction, side cavity dynamics, sediment replenishment and riparian vegetation plantings in riverine environments.

CHAPTER

FIFTYFOUR

DISCLAIMER

No warranty is expressed or implied regarding the usefulness or completeness of the information provided by *River Architect* and its documentation (wiki). References to commercial products do not imply endorsement by the Authors of *River Architect*. The concepts, materials, and methods used in the algorithms and described in the wiki are for informational purposes only. The Authors have made substantial effort to ensure the accuracy of the algorithms and the wiki, but science of prediction is uncertain and the Authors shall not be held liable, nor their employers or funding sponsors, for calculations and/or decisions made on the basis of application of *River Architect*. The information is provided "as is" and anyone who chooses to use the information is responsible for her or his own choices as to what to do with the data and the individual is responsible for the results the follow from their decisions.

CHAPTER

FIFTYFIVE

DEVELOPMENT

The development of *River Architect* follows the *Zen of Python*, also known as PEP20.

```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

The *River Architect Wiki* for developers assumes a basic understanding of *Python3*, object-oriented programming, the *markdown* language and basics in *HTML*. In order to ensure quality, we test and debug new code before pushing it, and we run spell checkers before updating the Wiki (albeit we cannot guarantee that the writing is 100 percent correct, we do our best).

CHAPTER

FIFTYSIX

ADD A MODULE TO *RIVER ARCHITECT*

Overview

- *Use Python templates*
- *Bind the new module to the GUI*
- *Full folder and file structure*

New modules or code modifications should be pushed to the `program` repository https://github.com/RiverArchitect/program (read more about using `git`).

River Architect has currently five **master** tabs (or modules) and three of them have **slave**-tabs (sub-modules): *Lifespan*, *Morphology*, and Ecohydraulics. The development of a new module or sub-module follows the same standards and varies only in the way how the tab is finally bound in the *River Architect* master GUI. For creating a new (sub) module, do the following:

1. Have a look at *River Architect* (see folder and file structure) and think about the capacities that the new module should have, and how it can be fitted in the existing framework (think twice before adding a new master tab).
2. Clone the template repository: `git clone https://github.com/RiverArchitect/development.git`.
3. Frome the cloned repository, copy the folder `RiverArchitect/development/moduleTEMPLATE/` to `RiverArchitect/moduleTEMPLATE/`.
4. Rename `RiverArchitect/moduleTEMPLATE/` to `RiverArchitect/NEW_MODULE_NAME/` (obviously, replace `NEW_MODULE_NAME` with the name of the new module).
5. Edit and add the module files `RiverArchitect/NEW_MODULE_NAME/cTEMPLATE.py` and `--TEMPLATE_gui.py` (see below descriptions of the *module template*).
6. Bind the new module to the *River Architect* master GUI (see below descriptions for *binding a new module to the* River Architect *master GUI*).

56.1 Working with the module template

The *River Architect* development repository provides template file for creating a new module in the `moduleTEMPLATE/` directory:

- *`TEMPLATE_gui.py`* is a template of the actual GUI that will be shown in the new tab.
- *`cTEMPLATE.py`* is a template for writing new classes using geospatial analysis with `arcpy.sa`.

Inline comments guide through the scripts and their dependencies. Both scripts load all functions and load most of the required packages from `RiverArchitect/.site_packages/riverpy/fGlobal.py`.

56.1.1 Using `TEMPLATE_gui.py`

The script contains `tkinter` classes to build the tab GUI. The first class is a `PopUpWindow` that may be useful for inquiring complex user input (e.g., calculate a value). A good example of using complex user input within *River Architect* is the River Builder input file creator.

```
class PopUpWindow(object):
    def __init__(self, master):
                ...
```

The most relevant template class here is `class MainGui(sg.RaModuleGui)`, which inherits the tab-slave class `RaModuleGui` from `RiverArchitect/slave_gui.py`. **All tabs must inherit `RiverArchitect/slave_gui.RaModuleGui`**, no matter if this will be a master or a slave tab. As such, all tabs inherit the following class variables (relevant variables only are listed here):

- `self.condition` is a `STR` of a condition contained in `RiverArchitect/01_Conditions/`
- `self.features` is a `cDef.FeatureDefinitions()` object resulting from `RiverArchitect/LifespanDesign/.templates/threshold_values.xlsx` (read more about features)
- `self.reaches` is a `cDef.ReachDefinitions()` object resulting from `RiverArchitect/ModifyTerrain/.templates/computation_extents.xlsx` (*read more about river reaches*)
- `self.units` is a `STR` variable, which is either `us` (US customary) or `si` (SI Metric)
- `self.ww` is an `INT` variable defining the tab window width
- `self.wh` is an `INT` variable defining the tab window height
- `self.wx` is an `INT` variable defining the tab window x-position
- `self.wy` is an `INT` variable defining the tab window y-position
- `self.xd` is an `INT` variable defining the x-pad (pixel space around `tkinter` objects)
- `self.yd` is an `INT` variable defining the y-pad (pixel space around `tkinter` objects)

Moreover, the `RaModuleGui` class automatically creates a `"Units"` and a `"Close"` dropdown menu.

The `MainGui(sg.RaModuleGui)` class header (`__init__()` function) looks like this:

```
class MainGui(sg.RaModuleGui):
    def __init__(self, from_master):
        sg.RaModuleGui.__init__(self, from_master)
        self.ww = 800  # window width
        self.wh = 840  # window height
```

(continues on next page)

(continued from previous page)

```
        self.title = "TEMPLATE"          # <---------------------- SET MODULE NAME␣
↪HERE, BUT DO NOT CHANGE CLASS NAME
        self.set_geometry(self.ww, self.wh, self.title)
                # ... tkinter examples
```

We recommend modifying the `__init__()` function of new modules starting with the `self.ww` and `self.wh` variables, which define the tab window width and height in pixels, respectively (try to avoid values larger than 1000 pixels). Most important and **required** edits are:

- the **file name** (rename `TEMPLATE_gui.py` to something like `new_module_gui.py`), and
- the window title defined with the `self.title` variable (see above code snippet, where `"TEMPLATE"` should be renamed to something like `"New Module"`).

Below `self.title` variable, the template provides some examples for using `tkinter` objects such as labels, listboxes with scroll bars, buttons, checkbuttons, and drop-down menus. For `tkinter` beginners, we recommend to carefully explore different `tkinter` objects (read more).

The `MainGui.run_calculation()` function illustrates the implementation of a geospatial calculator class (*see* `cTEMPLATE.py`):

```
    def run_calculation(self):
        """
        Trigger a calculation with cTEMPLATE.TEMPLATE()
        """
        geo_calc_object = cTEMPLATE.TEMPLATE()

        # inquire input raster name
        dir2raster = askopenfilename(initialdir=".", title="Select input GeoTIFF␣
↪raster",
                                     filetypes=[('GeoTIFF', '*.tif;*.tiff')])
        result = geo_calc_object.use_spatial_analyst_function(dir2raster)
        geo_calc_object.clear_cache()  # remove temporary calculation files

        if not(result == -1):
            showinfo("SUCCESS", "The result raster %s has been created." % result)
        else:
            showinfo("ERROR", "Something went wrong. Check the logfile and River␣
↪Architect Troubleshooting Wiki.")
```

`geo_calc_object` is a `cTEMPLATE.TEMPLATE()` object that is locally created in the `run_calculation()` function. It has a `use_spatial_analyst_function` that requires an input raster path to apply a (random) *ArcGIS Spatial Analyst* map algebra expression to that input raster. For this purpose, the `MainGui.run_calculation()` function asks the user to define an input raster using `tkinters`'s `askopenfilename()` function (imported from `tkinter.filedialog`). The line `result = geo_calc_object.use_spatial_analyst_function(dir2raster)` passes the selected raster file path (`dir2raster`) to the geospatial calculation function, which returns a string of the output raster if the calculation was successful (otherwise, it returns -1). After the calculation, an info window pops up (`showinfo(WINDOW_NAME, MESSAGE)` imported from `tkinter.messagebox`) and informs about the calculation success.

The `MainGui.set_value()` function illustrates the usage of the above mentioned `PopUpWindow` class (in the same file `TEMPLATE_gui.py`) to modify a value. The usage in the template is somewhat meaningless; a more reasonable value modification can be found in the *Lifespan Design* module where the user can modify the *Manning's n* roughness value (see applicaton).

```
    def set_value(self):
        sub_frame = PopUpWindow(self.master)
```

(continues on next page)

(continued from previous page)

```
        self.master.wait_window(sub_frame.top)
        value = float(sub_frame.value)
        showinfo("INFO", "You entered %s (whatever...)." % str(value))
```

The `MainGui.set_value()` function creates a sub-frame by instantiating a `PopUpWindow` object with `MainGui` as master. While the `PopUpWindow` is open, the main window waits for user input in the `PopUpWindow` (`self.master.wait_window(sub_frame.top)`). After closing the window, the function retrieves the user-defined value with `value = float(sub_frame.value)`. Finally, an info window pops up (`showinfo(WINDOW_NAME, MESSAGE)` imported from `tkinter.messagebox`) and recalls the user-defined value (well, this is trivial here . . .).

56.1.2 Using `cTEMPLATE.py`

The `cTEMPLATE.TEMPLATE()` class provides a useful frame for programming geospatial calculations, but none of the template routines is obligatory (in contrast to the *above GUI template*. For using the template class, make the following adaptations to the code block:

```
class TEMPLATE:
    def __init__(self, *args, **kwargs):
        """
        ...
        """

        # Create a temporary cache folder for, e.g., geospatial analyses; use self.
↪clear_cache() function to delete
        self.cache = os.path.dirname(__file__) + "\\.cache%s\\" % str(random.
↪randint(10000, 99999))
        chk_dir(self.cache)  # from fGlobal

        # Enable logger
        self.logger = logging.getLogger("logfile")
```

- rename the class from `TEMPLATE` to something meaningful;
- modify the input arguments of the `__init__(self, *args, **kwargs)` function.

The `cTEMPLATE.TEMPLATE()` class initiates a `self.logger` variable to write calculation progress to a logfile that is created as `RiverArchitect/logfile.log` when *River Architect* is called from the main GUI. This logefile should be used at every critical step where erroneous user input may yield wrong or no result. For example, if a user selects a raster file without valid data, the code should detect and tell the user about the problem using precise `try-except` statements. The `cTEMPLATE.TEMPLATE.use_spatial_analyst_function(input_raster_path)` function provides an example for the implementation of such behavior:

```
        try:
                self.logger.info("  >> loading raster %s ..." % str(input_raster_
↪path))
                in_ras = arcpy.Raster(input_raster_path)
        except:
                # go here if the input_raster_path is invalid and write a USEFUL␣
↪error message
                self.logger.info("ERROR: Game over ... add this error message to RA_
↪wiki/Troubleshooting")
                return -1
```

In this exemplary code snippet, the command `self.logger.info(" >> loading raster %s ... " % str(input_raster_path))` writes the raster input path to the console window and to the log-file using `self.logger.info()`. If the next step fails (i.e., `arcpy.Raster()` cannot load the provided `input_raster_path` as raster), the user should be informed. This information is passed recorded in the `except:` statement with `self.logger.info("ERROR: Game over ... add this error message to RA_wiki/Troubleshooting")`, which ends with returning `-1` (exits the function because the raster is necessary for all other steps). It would be better to use more precise exception rules such as `except ValueError:`, but we kept the broad "shotgun" approach with `except:` only because we encountered unexpected exception types using `arcpy`. A better and future way of error message logging in *River Architect* will be to use `self.logger.error()` in the exception statement rather than `self.logger.info()`, which should exclusively be used to log calculation progress. **Important is to add error messages, as well as warning messages, to the *Troubleshooting Wiki*** (read more about extending the Wiki). Use warning messages when a function can still work even though a variable assignment error occurred such as when an optional, additional raster is missing.

Finally, every class should close with a call function, where only the `TEMPLATE` string should be adapted to the new class name:

```
    def __call__(self, *args, **kwargs):
        # Object call should return class name, file location and class attributes␣
↪(dir)
        print("Class Info: <type> = TEMPLATE (%s)" % os.path.dirname(__file__))
        print(dir(self))
```

More features of the template are provided with the inline comments in `cTEMPLATE.py`.

56.2 Binding a new tab (module) to the *River Architect* master GUI

New modules need to be bound to the master GUI, either as master-tab or as slave-tab. Both need to added to the `RiverArchitect/master_gui.py` script in the `RA_gui` class's `__init__()` function.

56.2.1 Binding a New Module with or without submodules

Assuming a new master tab with the name ***NEW MODULE, optionally with *New Sub Modules***, should to be added, which is located in `RiverArchitect/NewModule/new_module_gui.py` (or submodules in `RiverArchitect/NewSubModule1/new_sub1_gui.py` and `RiverArchitect/NewSubModule2/new_sub2_gui.py`, respectively), the following variables need to be complemented (**attention: the list order is important** - new modules should only be appended at the end of lists):

- `self.tab_names = ['Get Started', 'Lifespan', 'Morphology', 'Ecohydraulics', 'Project Maker', 'NEW MODULE']`
- For adding slave (sub) tabs with names `NEW SUB1` and `NEW SUB2`: + `self.sub_tab_parents = ['Lifespan', 'Morphology', 'Ecohydraulics', 'NEW MODULE']` + `self.sub_tab_names = [['Lifespan Design', 'Max Lifespan'], ['Modify Terrain', 'Volume Assessment'], ['Habitat Area (SHArC)', 'Stranding Risk'], ['NEW SUB1', 'NEW SUB2', ... and so on]]`
- `self.tab_dir_names` + No sub tabs: `self.tab_dir_names= ['\\GetStarted\\', ['\\LifespanDesign\\', '\\MaxLifespan\\'], ['\\ModifyTerrain\\', '\\VolumeAssessment\\'], ['\\SHArC\\', '\\StrandingRisk\\'], '\\ProjectMaker\\', '\\NewModule\\']` + With sub tabs located in `RiverArchitect/NewSubModule1/` and `RiverArchitect/NewSubModule2/`, respectively: `self.tab_dir_names`

`= ['\\GetStarted\\', ['\\LifespanDesign\\', '\\MaxLifespan\\'], ['\\ModifyTerrain\\', '\\VolumeAssessment\\'], ['\\SHArC\\', '\\StrandingRisk\\'], '\\ProjectMaker\\', ['\\NewSubModule1\\', '\\NewSubModule2\\']]`

- `self.tab_list` + No sub tabs: `self.tab_list= [GetStarted.welcome_gui.MainGui(self.tab_container), ttk.Notebook(self.tab_container), ttk.Notebook(self.tab_container), ttk.Notebook(self.tab_container), ProjectMaker.project_maker_gui.MainGui(self.tab_container), NewModule.new_module_gui.MainGui(self.tab_container)]` + With sub tabs: `self.tab_list= [GetStarted.welcome_gui.MainGui(self.tab_container), ttk.Notebook(self.tab_container), ttk.Notebook(self.tab_container), ttk.Notebook(self.tab_container), ProjectMaker.project_maker_gui.MainGui(self.tab_container), ttk.Notebook(self.tab_container)]`
- With sub tabs: `self.sub_tab_list = [[LifespanDesign.lifespan_design_gui.FaGui(self.tabs['Lifespan']), MaxLifespan.action_gui.ActionGui(self.tabs['Lifespan'])], [ModifyTerrain.modify_terrain_gui.MainGui(self.tabs['Morphology']), VolumeAssessment.volume_gui.MainGui(self.tabs['Morphology'])], [SHArC.sharc_gui.MainGui(self.tabs['Ecohydraulics']), StrandingRisk.connect_gui.MainGui(self.tabs['Ecohydraulics'])], [NewSubModule1.new_sub1_gui.MainGui(self.tabs['NEW SUB1']), NewSubModule2.new_sub2_gui.MainGui(self.tabs['NEW SUB2'])]]`

56.2.2 Binding a New Sub-Module to an existing module

The following list indicates variables to be modified when a new sub-module (slave or sub-tab) is added to an existing module. The example assumes the creation of a sub-tab called `RiverCreator` to the `Morphology` module.

- Verify that the `Morphology` tab is mentioned in `self.sub_tab_parents = ['Lifespan', 'Morphology', 'Ecohydraulics', 'NEW MODULE']`
- Add `RIVER CREATOR` to the second nested list in `self.sub_tab_names = [['Lifespan Design', 'Max Lifespan'], ['Modify Terrain', 'Volume Assessment', 'RIVER CREATOR'], ['Habitat Area (SHArC)', 'Stranding Risk']]`; the second nested list must be chosen because `Morphology` is the second item in `self.sub_tab_parents`
- Assuming that the new module is located in `RiverArchitect/RiverCreator`, modify `self.tab_dir_names = ['\\GetStarted\\', ['\\LifespanDesign\\', '\\MaxLifespan\\'], ['\\ModifyTerrain\\', '\\VolumeAssessment\\', '\\RiverCreator\\'], ['\\SHArC\\', '\\StrandingRisk\\'], '\\ProjectMaker\\', ['\\NewSubModule1\\', '\\NewSubModule2\\']]`
- Withe the asumption that the new *River Creator* GUI is located in `RiverArchitect/RiverCreator/rc_gui.py`, modify `self.sub_tab_list = [[LifespanDesign.lifespan_design_gui.FaGui(self.tabs['Lifespan']), MaxLifespan.action_gui.ActionGui(self.tabs['Lifespan'])], [ModifyTerrain.modify_terrain_gui.MainGui(self.tabs['Morphology']), VolumeAssessment.volume_gui.MainGui(self.tabs['Morphology']), RiverCreator.rc_gui.MainGui(self.tabs['River Creator'])], [SHArC.sharc_gui.MainGui(self.tabs['Ecohydraulics']), StrandingRisk.connect_gui.MainGui(self.tabs['Ecohydraulics'])]]`

Please do not forget to update the wiki and test new code!

CHAPTER

FIFTYSEVEN

FULL LIST OF FOLDERS AND FILES

This is the full list of files and folders in the `RiverArchitect/program` repository. Please update after adding or deleting files. The file list can be generated (in Windows) with *PowerShell*: Enter `Get-ChildItem -Path D:\path-to-local-copy-of-RiverArchitect\program -Recurse` and copy-paste the comand line output.

```

RiverArchitect\program
Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----        7/26/2019  11:38 AM                .github
d-----        7/26/2019  11:39 AM                .site_packages
d-----        7/26/2019  11:39 AM                00_Flows
d-----        7/26/2019  11:39 AM                01_Conditions
d-----        7/26/2019  11:39 AM                02_Maps
d-----       11/22/2019  10:17 AM                StrandingRisk
d-----        7/26/2019  11:39 AM                docs
d-----       11/19/2019  12:40 PM                GetStarted
d-----        7/26/2019  11:39 AM                LifespanDesign
d-----        7/26/2019  11:39 AM                MaxLifespan
d-----        10/2/2019   9:57 AM                ModifyTerrain
d-----        11/5/2019   1:57 PM                ProjectMaker
d-----        10/2/2019   9:57 AM                SHArC
d-----       11/25/2019   1:04 PM                Tools
d-----        7/26/2019  11:39 AM                VolumeAssessment
-a----       11/13/2019   8:47 AM             22 .gitignore
-a----         9/4/2019   5:04 PM           1540 LICENSE
-a----       11/22/2019  10:17 AM           6458 master_gui.py
-a----        7/26/2019  11:39 AM           5856 README.md
-a----        11/4/2019  12:33 PM          11594 slave_gui.py
-a----        7/26/2019  11:39 AM            619 Start_River_Architect.bat


RiverArchitect\program\.github
Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----        7/26/2019  11:38 AM                ISSUE_TEMPLATE


RiverArchitect\program\.github\ISSUE_TEMPLATE
Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        7/26/2019  11:38 AM            834 bug_report.md
```

(continues on next page)

(continued from previous page)

```
-a----        7/26/2019  11:38 AM            126 custom.md
-a----        7/26/2019  11:38 AM            595 feature_request.md


RiverArchitect\program\.site_packages
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        7/26/2019  11:39 AM                openpyxl
d-----       11/19/2019  12:40 PM                riverpy
d-----        7/26/2019  11:39 AM                templates
-a----        7/26/2019  11:38 AM             23 info.md


RiverArchitect\program\.site_packages\openpyxl
---- TRUNCATED ---- DEPRECATED MODULE BINDINGS

RiverArchitect\program\.site_packages\riverpy
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        7/26/2019  11:39 AM           7131 cDefinitions.py
-a----        7/26/2019  11:39 AM           4141 cFeatures.py
-a----        7/26/2019  11:39 AM           5840 cFish.py
-a----       11/19/2019  12:40 PM          19563 cFlows.py
-a----        7/26/2019  11:39 AM          12334 cInputOutput.py
-a----        7/26/2019  11:39 AM           2550 cInterpolator.py
-a----        7/26/2019  11:39 AM           2470 cLogger.py
-a----        7/26/2019  11:39 AM          10583 cMakeTable.py
-a----       11/18/2019   2:11 PM          23510 cMapper.py
-a----         8/6/2019   8:03 AM           2899 config.py
-a----        7/26/2019  11:39 AM           9290 cReachManager.py
-a----        7/26/2019  11:39 AM           4245 cThresholdDirector.py
-a----        9/17/2019  10:03 AM          21388 fGlobal.py
-a----        7/26/2019  11:39 AM            120 __init__.py


RiverArchitect\program\.site_packages\templates
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        7/26/2019  11:39 AM          70862 code_icon.ico
-a----       11/19/2019  11:35 AM            134 credits.txt
-a----        7/26/2019  11:39 AM           6172 empty.xlsx
-a----        7/26/2019  11:39 AM          34208 Fish.xlsx
-a----        7/26/2019  11:39 AM          40751 morphological_units.xlsx
-a----        7/26/2019  11:39 AM            164 oups.txt


RiverArchitect\program\00_Flows
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        7/26/2019  11:39 AM                InputFlowSeries
d-----       11/21/2019   6:10 PM                templates
-a----        7/26/2019  11:39 AM             38 info.md


RiverArchitect\program\00_Flows\InputFlowSeries
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
```

(continues on next page)

(continued from previous page)

```
-a----        7/26/2019  11:39 AM         252121 flow_series_example_data.xlsx


RiverArchitect\program\00_Flows\templates
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----       11/21/2019   6:10 PM         188246 disc_freq_template.xlsx
-a----        7/26/2019  11:39 AM         205163 flow_duration_template.xlsx
-a----        7/26/2019  11:39 AM          15163 flow_return_period_template.xlsx


RiverArchitect\program\01_Conditions
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        7/26/2019  11:39 AM                none
-a----        7/26/2019  11:39 AM       13258925 20XX_sample.zip
-a----        7/26/2019  11:39 AM            208 README.md


RiverArchitect\program\01_Conditions\none
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        7/26/2019  11:39 AM            111 info.txt
-a----        7/26/2019  11:39 AM             60 __init__.py


RiverArchitect\program\02_Maps
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        7/26/2019  11:39 AM                templates


RiverArchitect\program\02_Maps\templates
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        7/26/2019  11:39 AM                Index
d-----        7/26/2019  11:39 AM                legend
d-----        7/26/2019  11:39 AM                rasters
d-----        7/29/2019   5:48 PM                river_template.gdb
d-----        7/29/2019   5:48 PM                scratch.gdb
d-----        7/26/2019  11:39 AM                symbology
-a----        7/26/2019  11:39 AM         475672 river_template.aprx
-a----        7/26/2019  11:39 AM           4096 river_template.tbx


RiverArchitect\program\02_Maps\templates\Index
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        7/26/2019  11:39 AM                river_map_template
d-----        7/26/2019  11:39 AM                river_template
d-----        7/26/2019  11:39 AM                Thumbnail


RiverArchitect\program\02_Maps\templates\Index\river_map_template
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        7/26/2019  11:39 AM             75 log.txt
```

(continues on next page)

(continued from previous page)

```
-a----        7/26/2019  11:39 AM             20 segments.gen
-a----        7/26/2019  11:39 AM            200 segments_2
-a----        7/26/2019  11:39 AM           1509 _0.cfs


RiverArchitect\program\02_Maps\templates\Index\river_template
Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        7/26/2019  11:39 AM             71 log.txt
-a----        7/26/2019  11:39 AM             20 segments.gen
-a----        7/26/2019  11:39 AM           2580 segments_12
-a----        7/26/2019  11:39 AM           2075 segments_z
-a----        7/26/2019  11:39 AM            977 _10.cfs
-a----        7/26/2019  11:39 AM           6183 _m.cfs
-a----        7/26/2019  11:39 AM            673 _n.cfs
-a----        7/26/2019  11:39 AM              9 _n_1.del
-a----        7/26/2019  11:39 AM            807 _o.cfs
-a----        7/26/2019  11:39 AM            688 _p.cfs
-a----        7/26/2019  11:39 AM              9 _p_1.del
-a----        7/26/2019  11:39 AM            837 _q.cfs
-a----        7/26/2019  11:39 AM            688 _r.cfs
-a----        7/26/2019  11:39 AM              9 _r_1.del
-a----        7/26/2019  11:39 AM            805 _s.cfs
-a----        7/26/2019  11:39 AM            688 _t.cfs
-a----        7/26/2019  11:39 AM              9 _t_1.del
-a----        7/26/2019  11:39 AM            849 _u.cfs
-a----        7/26/2019  11:39 AM            789 _v.cfs
-a----        7/26/2019  11:39 AM            805 _w.cfs
-a----        7/26/2019  11:39 AM            804 _x.cfs
-a----        7/26/2019  11:39 AM            737 _y.cfs
-a----        7/26/2019  11:39 AM              9 _y_1.del
-a----        7/26/2019  11:39 AM            973 _z.cfs
-a----        7/26/2019  11:39 AM              9 _z_1.del


RiverArchitect\program\02_Maps\templates\Index\Thumbnail
Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        7/26/2019  11:39 AM              0 indx
-a----        7/26/2019  11:39 AM           4208 layers.jpg
-a----        7/26/2019  11:39 AM           5833 layers12.jpg
-a----        7/26/2019  11:39 AM           5611 layers2.jpg
-a----        7/26/2019  11:39 AM           6006 layers22.jpg
-a----        7/26/2019  11:39 AM           5851 layers32.jpg
-a----        7/26/2019  11:39 AM           4823 layers5.jpg
-a----        7/26/2019  11:39 AM           5789 layers7.jpg
-a----        7/26/2019  11:39 AM           4346 map.jpg


RiverArchitect\program\02_Maps\templates\legend
Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        7/26/2019  11:39 AM        1120768 legend.ServerStyle
-a----        7/26/2019  11:39 AM        1622016 legend.style


RiverArchitect\program\02_Maps\templates\rasters
```

(continues on next page)

(continued from previous page)

```
Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----        7/26/2019  11:39 AM                ds_bio_s0
d-----        7/26/2019  11:39 AM                ds_elj_dwc
d-----        7/26/2019  11:39 AM                ds_finese_d15
d-----        7/26/2019  11:39 AM                ds_finese_d85
d-----        7/26/2019  11:39 AM                ds_sidec
d-----        7/26/2019  11:39 AM                ds_sidech110
d-----        7/26/2019  11:39 AM                ds_sym_d
d-----        7/26/2019  11:39 AM                ds_widen
d-----        7/26/2019  11:39 AM                lf_sym
-a----        7/29/2019   5:48 PM           1400 ds_bio_s0.aux.xml
-a----        7/29/2019   5:48 PM           1383 ds_elj_dwc.aux.xml
-a----        7/29/2019   5:48 PM           1413 ds_finese_d15.aux.xml
-a----        7/29/2019   5:48 PM           1387 ds_finese_d85.aux.xml
-a----        7/29/2019   5:48 PM           1487 ds_sidec.aux.xml
-a----        7/29/2019   5:48 PM           1407 ds_sidech110.aux.xml
-a----        7/29/2019   5:48 PM           2335 ds_sym_D.aux.xml
-a----        7/29/2019   5:48 PM            504 ds_widen.aux.xml
-a----        7/26/2019  11:39 AM        1274552 layout_lf.xml
-a----        7/29/2019   5:48 PM           2316 lf_sym.aux.xml


RiverArchitect\program\02_Maps\templates\rasters\ds_bio_s0
Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        7/26/2019  11:39 AM             32 dblbnd.adf
-a----        7/26/2019  11:39 AM            308 hdr.adf
-a----        7/26/2019  11:39 AM            358 prj.adf
-a----        7/26/2019  11:39 AM             32 sta.adf
-a----        7/26/2019  11:39 AM          98486 w001001.adf
-a----        7/26/2019  11:39 AM            428 w001001x.adf


RiverArchitect\program\02_Maps\templates\rasters\ds_elj_dwc
Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        7/26/2019  11:39 AM             32 dblbnd.adf
-a----        7/26/2019  11:39 AM            308 hdr.adf
-a----        7/26/2019  11:39 AM            170 prj.adf
-a----        7/26/2019  11:39 AM             32 sta.adf
-a----        7/26/2019  11:39 AM          49286 w001001.adf
-a----        7/26/2019  11:39 AM            236 w001001x.adf


RiverArchitect\program\02_Maps\templates\rasters\ds_finese_d15
Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        7/26/2019  11:39 AM             32 dblbnd.adf
-a----        7/26/2019  11:39 AM            308 hdr.adf
-a----        7/26/2019  11:39 AM            170 prj.adf
-a----        7/26/2019  11:39 AM             32 sta.adf
-a----        7/26/2019  11:39 AM          98486 w001001.adf
-a----        7/26/2019  11:39 AM            428 w001001x.adf


RiverArchitect\program\02_Maps\templates\rasters\ds_finese_d85
```

(continues on next page)

(continued from previous page)

```
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         7/26/2019  11:39 AM             32 dblbnd.adf
-a----         7/26/2019  11:39 AM            308 hdr.adf
-a----         7/26/2019  11:39 AM            170 prj.adf
-a----         7/26/2019  11:39 AM             32 sta.adf
-a----         7/26/2019  11:39 AM          49286 w001001.adf
-a----         7/26/2019  11:39 AM            236 w001001x.adf


RiverArchitect\program\02_Maps\templates\rasters\ds_sidec
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         7/26/2019  11:39 AM             32 dblbnd.adf
-a----         7/26/2019  11:39 AM            308 hdr.adf
-a----         7/29/2019   5:48 PM            358 prj.adf
-a----         7/26/2019  11:39 AM             32 sta.adf
-a----         7/26/2019  11:39 AM              8 vat.adf
-a----         7/26/2019  11:39 AM           1662 w001001.adf
-a----         7/26/2019  11:39 AM            428 w001001x.adf


RiverArchitect\program\02_Maps\templates\rasters\ds_sidech110
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         7/26/2019  11:39 AM             32 dblbnd.adf
-a----         7/26/2019  11:39 AM            308 hdr.adf
-a----         7/26/2019  11:39 AM            358 prj.adf
-a----         7/26/2019  11:39 AM             32 sta.adf
-a----         7/26/2019  11:39 AM          49286 w001001.adf
-a----         7/26/2019  11:39 AM            236 w001001x.adf


RiverArchitect\program\02_Maps\templates\rasters\ds_sym_d
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         7/26/2019  11:39 AM             32 dblbnd.adf
-a----         7/26/2019  11:39 AM            308 hdr.adf
-a----         7/29/2019   5:48 PM            358 prj.adf
-a----         7/26/2019  11:39 AM             32 sta.adf
-a----         7/26/2019  11:39 AM          32886 w001001.adf
-a----         7/26/2019  11:39 AM            172 w001001x.adf


RiverArchitect\program\02_Maps\templates\rasters\ds_widen
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         7/26/2019  11:39 AM             32 dblbnd.adf
-a----         7/26/2019  11:39 AM            308 hdr.adf
-a----         7/26/2019  11:39 AM            358 prj.adf
-a----         7/26/2019  11:39 AM             32 sta.adf
-a----         7/26/2019  11:39 AM              8 vat.adf
-a----         7/26/2019  11:39 AM          98486 w001001.adf
-a----         7/26/2019  11:39 AM            428 w001001x.adf


RiverArchitect\program\02_Maps\templates\rasters\lf_sym
```

(continues on next page)

(continued from previous page)

```
Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        7/26/2019  11:39 AM             32 dblbnd.adf
-a----        7/26/2019  11:39 AM            308 hdr.adf
-a----        7/29/2019   5:48 PM            358 prj.adf
-a----        7/26/2019  11:39 AM             32 sta.adf
-a----        7/26/2019  11:39 AM          98486 w001001.adf
-a----        7/26/2019  11:39 AM            428 w001001x.adf


RiverArchitect\program\02_Maps\templates\river_template.gdb
Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        7/26/2019  11:39 AM            110 a00000001.gdbindexes
-a----        7/26/2019  11:39 AM            302 a00000001.gdbtable
-a----        7/26/2019  11:39 AM           5152 a00000001.gdbtablx
-a----        7/26/2019  11:39 AM           4118 a00000001.TablesByName.atx
-a----        7/26/2019  11:39 AM           2055 a00000002.gdbtable
-a----        7/26/2019  11:39 AM           5152 a00000002.gdbtablx
-a----        7/26/2019  11:39 AM             42 a00000003.gdbindexes
-a----        7/26/2019  11:39 AM            525 a00000003.gdbtable
-a----        7/26/2019  11:39 AM           5152 a00000003.gdbtablx
-a----        7/26/2019  11:39 AM           4118 a00000004.CatItemsByPhysicalName.atx
-a----        7/26/2019  11:39 AM           4118 a00000004.CatItemsByType.atx
-a----        7/26/2019  11:39 AM           4118 a00000004.FDO_UUID.atx
-a----        7/26/2019  11:39 AM            310 a00000004.gdbindexes
-a----        7/26/2019  11:39 AM           1712 a00000004.gdbtable
-a----        7/26/2019  11:39 AM           5152 a00000004.gdbtablx
-a----        7/26/2019  11:39 AM           4118 a00000004.spx
-a----        7/26/2019  11:39 AM          12310 a00000005.CatItemTypesByName.atx
-a----        7/26/2019  11:39 AM           4118 a00000005.CatItemTypesByParentTypeID.
↪atx
-a----        7/26/2019  11:39 AM           4118 a00000005.CatItemTypesByUUID.atx
-a----        7/26/2019  11:39 AM            296 a00000005.gdbindexes
-a----        7/26/2019  11:39 AM           2074 a00000005.gdbtable
-a----        7/26/2019  11:39 AM           5152 a00000005.gdbtablx
-a----        7/26/2019  11:39 AM           4118 a00000006.CatRelsByDestinationID.atx
-a----        7/26/2019  11:39 AM           4118 a00000006.CatRelsByOriginID.atx
-a----        7/26/2019  11:39 AM           4118 a00000006.CatRelsByType.atx
-a----        7/26/2019  11:39 AM           4118 a00000006.FDO_UUID.atx
-a----        7/26/2019  11:39 AM            318 a00000006.gdbindexes
-a----        7/26/2019  11:39 AM            190 a00000006.gdbtable
-a----        7/26/2019  11:39 AM             32 a00000006.gdbtablx
-a----        7/26/2019  11:39 AM          12310 a00000007.CatRelTypesByBackwardLabel.
↪atx
-a----        7/26/2019  11:39 AM           4118 a00000007.
↪CatRelTypesByDestItemTypeID.atx
-a----        7/26/2019  11:39 AM          12310 a00000007.CatRelTypesByForwardLabel.
↪atx
-a----        7/26/2019  11:39 AM          12310 a00000007.CatRelTypesByName.atx
-a----        7/26/2019  11:39 AM           4118 a00000007.
↪CatRelTypesByOriginItemTypeID.atx
-a----        7/26/2019  11:39 AM           4118 a00000007.CatRelTypesByUUID.atx
-a----        7/26/2019  11:39 AM            602 a00000007.gdbindexes
-a----        7/26/2019  11:39 AM           3479 a00000007.gdbtable
-a----        7/26/2019  11:39 AM           5152 a00000007.gdbtablx
-a----        7/26/2019  11:39 AM              4 gdb
```

(continues on next page)

(continued from previous page)

```
-a----        7/26/2019  11:39 AM            400 timestamps


RiverArchitect\program\02_Maps\templates\scratch.gdb
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        7/26/2019  11:39 AM            110 a00000001.gdbindexes
-a----        7/26/2019  11:39 AM            302 a00000001.gdbtable
-a----        7/26/2019  11:39 AM           5152 a00000001.gdbtablx
-a----        7/26/2019  11:39 AM           4118 a00000001.TablesByName.atx
-a----        7/26/2019  11:39 AM           2055 a00000002.gdbtable
-a----        7/26/2019  11:39 AM           5152 a00000002.gdbtablx
-a----        7/26/2019  11:39 AM             42 a00000003.gdbindexes
-a----        7/26/2019  11:39 AM            525 a00000003.gdbtable
-a----        7/26/2019  11:39 AM           5152 a00000003.gdbtablx
-a----        7/26/2019  11:39 AM           4118 a00000004.CatItemsByPhysicalName.atx
-a----        7/26/2019  11:39 AM           4118 a00000004.CatItemsByType.atx
-a----        7/26/2019  11:39 AM           4118 a00000004.FDO_UUID.atx
-a----        7/26/2019  11:39 AM            310 a00000004.gdbindexes
-a----        7/26/2019  11:39 AM           1712 a00000004.gdbtable
-a----        7/26/2019  11:39 AM           5152 a00000004.gdbtablx
-a----        7/26/2019  11:39 AM           4118 a00000004.spx
-a----        7/26/2019  11:39 AM          12310 a00000005.CatItemTypesByName.atx
-a----        7/26/2019  11:39 AM           4118 a00000005.CatItemTypesByParentTypeID.
↪atx
-a----        7/26/2019  11:39 AM           4118 a00000005.CatItemTypesByUUID.atx
-a----        7/26/2019  11:39 AM            296 a00000005.gdbindexes
-a----        7/26/2019  11:39 AM           2074 a00000005.gdbtable
-a----        7/26/2019  11:39 AM           5152 a00000005.gdbtablx
-a----        7/26/2019  11:39 AM           4118 a00000006.CatRelsByDestinationID.atx
-a----        7/26/2019  11:39 AM           4118 a00000006.CatRelsByOriginID.atx
-a----        7/26/2019  11:39 AM           4118 a00000006.CatRelsByType.atx
-a----        7/26/2019  11:39 AM           4118 a00000006.FDO_UUID.atx
-a----        7/26/2019  11:39 AM            318 a00000006.gdbindexes
-a----        7/26/2019  11:39 AM            190 a00000006.gdbtable
-a----        7/26/2019  11:39 AM             32 a00000006.gdbtablx
-a----        7/26/2019  11:39 AM          12310 a00000007.CatRelTypesByBackwardLabel.
↪atx
-a----        7/26/2019  11:39 AM           4118 a00000007.
↪CatRelTypesByDestItemTypeID.atx
-a----        7/26/2019  11:39 AM          12310 a00000007.CatRelTypesByForwardLabel.
↪atx
-a----        7/26/2019  11:39 AM          12310 a00000007.CatRelTypesByName.atx
-a----        7/26/2019  11:39 AM           4118 a00000007.
↪CatRelTypesByOriginItemTypeID.atx
-a----        7/26/2019  11:39 AM           4118 a00000007.CatRelTypesByUUID.atx
-a----        7/26/2019  11:39 AM            602 a00000007.gdbindexes
-a----        7/26/2019  11:39 AM           3479 a00000007.gdbtable
-a----        7/26/2019  11:39 AM           5152 a00000007.gdbtablx
-a----        7/26/2019  11:39 AM              4 gdb
-a----        7/26/2019  11:39 AM            400 timestamps


RiverArchitect\program\02_Maps\templates\symbology
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        7/26/2019  11:39 AM           7666 LifespanRasterSymbology.lyrx
```

(continues on next page)

(continued from previous page)

```
RiverArchitect\program\StrandingRisk
Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----        8/20/2019  12:22 PM                .templates
-a----       11/21/2019   6:10 PM          28031 cConnectivityAnalysis.py
-a----       11/22/2019  10:17 AM          16505 cGraph.py
-a----       11/19/2019  12:40 PM          12408 connect_gui.py
-a----       11/19/2019  12:40 PM           4199 cRatingCurves.py
-a----        7/26/2019  11:39 AM             84 __init__.py


RiverArchitect\program\StrandingRisk\.templates
Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        8/20/2019  12:22 PM         191729 disconnected_area_template.xlsx


RiverArchitect\program\docs
Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        7/26/2019  11:39 AM           3358 CODE_OF_CONDUCT.md
-a----        7/26/2019  11:39 AM           7504 CONTRIBUTING.md
-a----        7/26/2019  11:39 AM            802 PULL_REQUEST_TEMPLATE.md


RiverArchitect\program\GetStarted
Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----        7/26/2019  11:39 AM                .templates
-a----       11/19/2019  12:40 PM          18968 cConditionCreator.py
-a----        7/26/2019  11:39 AM           7321 cDetrendedDEM.py
-a----        7/26/2019  11:39 AM           4387 cMakeInp.py
-a----        7/26/2019  11:39 AM           8722 cMorphUnits.py
-a----        8/20/2019  12:22 PM          19556 cWaterLevel.py
-a----        7/26/2019  11:39 AM           3542 fSubCondition.py
-a----       10/30/2019   4:27 PM           4851 popup_align_rasters.py
-a----        11/4/2019  10:43 AM          10762 popup_analyze_q.py
-a----       10/30/2019   4:27 PM          18197 popup_create_c.py
-a----        9/17/2019  10:19 AM           7493 popup_create_c_sub.py
-a----        7/26/2019  11:39 AM           4329 popup_make_inp.py
-a----        9/17/2019  10:19 AM          13449 popup_populate_c.py
-a----       10/30/2019   4:27 PM           6226 welcome_gui.py
-a----        7/26/2019  11:39 AM             80 __init__.py


RiverArchitect\program\GetStarted\.templates
Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        7/26/2019  11:39 AM         143253 welcome.gif


RiverArchitect\program\LifespanDesign
Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----       11/25/2019   6:43 PM                .templates
```

(continues on next page)

(continued from previous page)

```
d-----        7/26/2019  11:39 AM                Output
-a----       10/24/2019   2:47 PM          53673 cLifespanDesignAnalysis.py
-a----       10/29/2019   6:17 PM          10816 cParameters.py
-a----        7/26/2019  11:39 AM           5776 cReadInpLifespan.py
-a----       10/23/2019   3:48 PM          16584 feature_analysis.py
-a----       11/25/2019   4:10 PM          21688 lifespan_design_gui.py
-a----        7/26/2019  11:39 AM            113 __init__.py


RiverArchitect\program\LifespanDesign\.templates
Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        7/26/2019  11:39 AM            703 mapping.inp
-a----        7/26/2019  11:39 AM            773 mapping_full.inp
-a----        7/26/2019  11:39 AM          11323 prepare_map_coordinates.xlsx
-a----       11/25/2019   4:10 PM          23190 threshold_values.xlsx


RiverArchitect\program\LifespanDesign\Output
Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----        7/26/2019  11:39 AM                Rasters


RiverArchitect\program\LifespanDesign\Output\Rasters
Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        7/26/2019  11:39 AM             58 __init__.py


RiverArchitect\program\MaxLifespan
Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----        7/26/2019  11:39 AM                .templates
d-----        7/26/2019  11:39 AM                Output
-a----       11/25/2019   4:11 PM          14301 action_gui.py
-a----        7/26/2019  11:39 AM           2934 action_planner.py
-a----        7/26/2019  10:50 AM          11193 cActionAssessment.py
-a----        7/26/2019  11:39 AM           3558 cFeatureActions.py
-a----        7/26/2019  11:39 AM            102 __init__.py


RiverArchitect\program\MaxLifespan\.templates
Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----        7/26/2019  11:39 AM                rasters
-a----        7/26/2019  11:39 AM            704 mapping.inp
-a----        7/26/2019  11:39 AM          11323 prepare_map_coordinates.xlsx


RiverArchitect\program\MaxLifespan\.templates\rasters
Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        7/26/2019  11:39 AM             91 zeros.tfw
-a----        7/26/2019  11:39 AM           9445 zeros.tif
-a----        7/26/2019  11:39 AM            364 zeros.tif.aux.xml
```

(continues on next page)

(continued from previous page)

```
RiverArchitect\program\MaxLifespan\Output
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         7/26/2019  11:39 AM                Rasters


RiverArchitect\program\MaxLifespan\Output\Rasters
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         7/26/2019  11:39 AM             62 __init__.py


RiverArchitect\program\ModifyTerrain
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         7/26/2019  11:39 AM                .templates
d-----         7/26/2019  11:39 AM                Output
d-----         7/26/2019  11:39 AM                RiverBuilder
-a----         7/26/2019  10:39 AM          12435 cModifyTerrain.py
-a----         7/26/2019  11:39 AM           1941 cRiverBuilder.py
-a----         7/26/2019  11:39 AM          12125 cRiverBuilderConstruct.py
-a----          8/6/2019   8:03 AM          13861 modify_terrain_gui.py
-a----         7/26/2019  11:39 AM           3777 riverbuilder_input_example.txt
-a----         9/17/2019  10:27 AM          15650 sub_gui_rb.py
-a----         7/26/2019  11:39 AM             87 __init__.py


RiverArchitect\program\ModifyTerrain\.templates
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         7/26/2019  11:39 AM         382805 chnl_xs_parameters.gif
-a----         7/26/2019  11:39 AM          10681 computation_extents.xlsx
-a----         7/26/2019  11:39 AM        1024000 determine_extents.mxd
-a----         7/26/2019  11:39 AM            593 determine_extents.xml
-a----         7/26/2019  11:39 AM            704 mapping.inp
-a----         7/26/2019  11:39 AM          11323 prepare_map_coordinates.xlsx


RiverArchitect\program\ModifyTerrain\Output
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         7/26/2019  11:39 AM                Grade
d-----         7/26/2019  11:39 AM                RiverBuilder
d-----         7/26/2019  11:39 AM                Widen


RiverArchitect\program\ModifyTerrain\Output\Grade
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         7/26/2019  11:39 AM             62 __init__.py


RiverArchitect\program\ModifyTerrain\Output\RiverBuilder
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         7/26/2019  11:39 AM             62 __init__.py
```

(continues on next page)

(continued from previous page)

```
RiverArchitect\program\ModifyTerrain\Output\Widen
Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----         7/26/2019  11:39 AM             62 __init__.py


RiverArchitect\program\ModifyTerrain\RiverBuilder
Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----         7/26/2019  11:39 AM                messages
-a----         7/26/2019  11:39 AM          45428 riverbuilder.r
-a----         7/26/2019  11:39 AM           3117 RiverBuilderRenderer.py


RiverArchitect\program\ModifyTerrain\RiverBuilder\messages
Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----         7/26/2019  11:39 AM            537 chnl_xs_pts.txt
-a----         7/26/2019  11:39 AM            391 d50.txt
-a----         7/26/2019  11:39 AM            730 datum.txt
-a----         7/26/2019  11:39 AM             81 dy.txt
-a----         7/26/2019  11:39 AM            292 floodplain_outer_height.txt
-a----         7/26/2019  11:39 AM            270 h_bf.txt
-a----         7/26/2019  11:39 AM            381 length.txt
-a----         7/26/2019  11:39 AM            935 sub_curvature.txt
-a----         7/26/2019  11:39 AM            285 sub_floodplain_l.txt
-a----         7/26/2019  11:39 AM            150 sub_floodplain_r.txt
-a----         7/26/2019  11:39 AM            568 sub_meander.txt
-a----         7/26/2019  11:39 AM            149 sub_thalweg.txt
-a----         7/26/2019  11:39 AM            667 sub_w_bf.txt
-a----         7/26/2019  11:39 AM            204 s_valley.txt
-a----         7/26/2019  11:39 AM            572 taux.txt
-a----         7/26/2019  11:39 AM            300 terrace_outer_height.txt
-a----         7/26/2019  11:39 AM           1391 user_functions.txt
-a----         7/26/2019  11:39 AM            150 w_bf.txt
-a----         7/26/2019  11:39 AM            431 w_bf_min.txt
-a----         7/26/2019  11:39 AM            343 w_boundary.txt
-a----         7/26/2019  11:39 AM            831 w_floodplain.txt
-a----         7/26/2019  11:39 AM            419 w_terrace.txt
-a----         7/26/2019  11:39 AM            585 xs_shape.txt
-a----         7/26/2019  11:39 AM            553 x_res.txt


RiverArchitect\program\ProjectMaker
Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----         7/29/2019   5:48 PM                .templates
-a----         10/2/2019   9:57 AM          10092 cSHArC.py
-a----         7/26/2019  11:39 AM           4245 fFunctions.py
-a----        11/25/2019   4:13 PM          34810 project_maker_gui.py
-a----        11/19/2019   9:32 AM          11784 s20_plantings_delineation.py
-a----        11/19/2019   9:43 AM           8778 s21_plantings_stabilization.py
-a----        10/24/2019   2:19 PM          11928 s30_terrain_stabilization.py
-a----        11/12/2019   2:21 PM           8488 s40_compare_sharea.py
-a----         7/26/2019  11:39 AM            217 __init__.py
```

(continues on next page)

(continued from previous page)

```
RiverArchitect\program\ProjectMaker\.templates
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        11/25/2019   4:08 PM                Project_vii_TEMPLATE
-a----         7/26/2019  11:39 AM              5 area_dummy.cpg
-a----         7/26/2019  11:39 AM            118 area_dummy.dbf
-a----         7/26/2019  11:39 AM            132 area_dummy.sbn
-a----         7/26/2019  11:39 AM            116 area_dummy.sbx
-a----         7/26/2019  11:39 AM            220 area_dummy.shp
-a----         7/26/2019  11:39 AM            108 area_dummy.shx


RiverArchitect\program\ProjectMaker\.templates\Project_vii_TEMPLATE
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        11/19/2019  10:27 AM                Geodata
d-----         7/26/2019  11:39 AM                Index
d-----        10/30/2019   5:57 PM                ProjectMaps.gdb
d-----        11/19/2019   9:41 AM                Quantities
-a----         11/7/2019   5:51 PM         203109 ProjectMaps.aprx
-a----         4/26/2019   9:43 AM           4096 ProjectMaps.tbx
-a----        11/25/2019   4:07 PM          35894 Project_assessment_vii.xlsx


RiverArchitect\program\ProjectMaker\.templates\Project_vii_TEMPLATE\Geodata
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        11/19/2019   9:12 AM                Rasters
d-----        11/19/2019   9:12 AM                Shapefiles
-a----         7/26/2019  11:39 AM          20814 SHArea_evaluation_template_si.xlsx
-a----         7/26/2019  11:39 AM          20797 SHArea_evaluation_template_us.xlsx


RiverArchitect\program\ProjectMaker\.templates\Project_vii_TEMPLATE\Geodata\Rasters
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        11/19/2019   9:12 AM             10 __init__.py


RiverArchitect\program\ProjectMaker\.templates\Project_vii_TEMPLATE\Geodata\Shapefiles
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        11/19/2019   9:12 AM             10 __init__.py


RiverArchitect\program\ProjectMaker\.templates\Project_vii_TEMPLATE\Index
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         11/5/2019   1:57 PM                ProjectMaps
d-----         7/26/2019  11:39 AM                Thumbnail


RiverArchitect\program\ProjectMaker\.templates\Project_vii_TEMPLATE\Index\ProjectMaps
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
```

(continues on next page)

(continued from previous page)

```
-a----         4/26/2019  10:18 AM             20 segments.gen
-a----         4/26/2019  10:18 AM           1067 segments_9
-a----         4/26/2019   9:43 AM           1122 _1.cfs
-a----         4/26/2019   9:43 AM            633 _2.cfs
-a----         4/26/2019  10:17 AM            669 _3.cfs
-a----         4/26/2019  10:18 AM              9 _3_1.del
-a----         4/26/2019  10:17 AM            812 _4.cfs
-a----         4/26/2019  10:18 AM            708 _5.cfs
-a----         4/26/2019  10:18 AM              9 _5_1.del
-a----         4/26/2019  10:18 AM            806 _6.cfs


RiverArchitect\program\ProjectMaker\.templates\Project_vii_TEMPLATE\Index\Thumbnail
Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----         4/26/2019  10:17 AM              0 indx
-a----         4/26/2019  10:18 AM           1827 layers.jpg


RiverArchitect\program\ProjectMaker\.templates\Project_vii_TEMPLATE\ProjectMaps.gdb
Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----         4/26/2019   9:43 AM            110 a00000001.gdbindexes
-a----        10/25/2019   8:54 AM            338 a00000001.gdbtable
-a----        10/25/2019   8:54 AM           5152 a00000001.gdbtablx
-a----        10/25/2019   8:54 AM           4118 a00000001.TablesByName.atx
-a----         4/26/2019   9:43 AM           2055 a00000002.gdbtable
-a----         4/26/2019   9:43 AM           5152 a00000002.gdbtablx
-a----         4/26/2019   9:43 AM             42 a00000003.gdbindexes
-a----         4/26/2019   9:43 AM            525 a00000003.gdbtable
-a----         4/26/2019   9:43 AM           5152 a00000003.gdbtablx
-a----        10/25/2019   8:54 AM           4118 a00000004.CatItemsByPhysicalName.atx
-a----        10/25/2019   8:54 AM           4118 a00000004.CatItemsByType.atx
-a----        10/25/2019   8:54 AM           4118 a00000004.FDO_UUID.atx
-a----        10/25/2019   8:54 AM           8536 a00000004.freelist
-a----         4/26/2019   9:43 AM            310 a00000004.gdbindexes
-a----        10/25/2019   8:54 AM          24787 a00000004.gdbtable
-a----        10/25/2019   8:54 AM           5152 a00000004.gdbtablx
-a----        10/25/2019   8:54 AM           4118 a00000004.spx
-a----         4/26/2019   9:43 AM          12310 a00000005.CatItemTypesByName.atx
-a----         4/26/2019   9:43 AM           4118 a00000005.CatItemTypesByParentTypeID.
↪atx
-a----         4/26/2019   9:43 AM           4118 a00000005.CatItemTypesByUUID.atx
-a----         4/26/2019   9:43 AM            296 a00000005.gdbindexes
-a----         4/26/2019   9:43 AM           2074 a00000005.gdbtable
-a----         4/26/2019   9:43 AM           5152 a00000005.gdbtablx
-a----        10/25/2019   8:54 AM           4118 a00000006.CatRelsByDestinationID.atx
-a----        10/25/2019   8:54 AM           4118 a00000006.CatRelsByOriginID.atx
-a----        10/25/2019   8:54 AM           4118 a00000006.CatRelsByType.atx
-a----        10/25/2019   8:54 AM           4118 a00000006.FDO_UUID.atx
-a----         4/26/2019   9:43 AM            318 a00000006.gdbindexes
-a----        10/25/2019   8:54 AM            263 a00000006.gdbtable
-a----        10/25/2019   8:54 AM           5152 a00000006.gdbtablx
-a----         4/26/2019   9:43 AM          12310 a00000007.CatRelTypesByBackwardLabel.
↪atx
-a----         4/26/2019   9:43 AM           4118 a00000007.
↪CatRelTypesByDestItemTypeID.atx
```

(continues on next page)

(continued from previous page)

```
-a----        4/26/2019   9:43 AM          12310 a00000007.CatRelTypesByForwardLabel.
↪atx
-a----        4/26/2019   9:43 AM          12310 a00000007.CatRelTypesByName.atx
-a----        4/26/2019   9:43 AM           4118 a00000007.
↪CatRelTypesByOriginItemTypeID.atx
-a----        4/26/2019   9:43 AM           4118 a00000007.CatRelTypesByUUID.atx
-a----        4/26/2019   9:43 AM            602 a00000007.gdbindexes
-a----        4/26/2019   9:43 AM           3479 a00000007.gdbtable
-a----        4/26/2019   9:43 AM           5152 a00000007.gdbtablx
-a----       10/25/2019   8:54 AM             66 a00000009.gdbindexes
-a----       10/25/2019   8:54 AM            168 a00000009.gdbtable
-a----       10/25/2019   8:54 AM           5152 a00000009.gdbtablx
-a----        4/26/2019   9:43 AM              4 gdb
-a----       10/25/2019   8:54 AM            400 timestamps


RiverArchitect\program\ProjectMaker\.templates\Project_vii_TEMPLATE\Quantities
Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----       11/19/2019   9:12 AM             10 __init__.py


RiverArchitect\program\SHArC
Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----        7/26/2019  11:39 AM                .templates
d-----        7/26/2019  11:39 AM                CHSI
d-----        7/26/2019  11:39 AM                HSI
d-----        7/26/2019  11:39 AM                SHArea
-a----       11/21/2019   6:14 PM          38809 cHSI.py
-a----       10/22/2019   2:31 PM          32273 sharc_gui.py
-a----        7/26/2019  11:12 AM          17490 sub_gui_covhsi.py
-a----       10/30/2019   5:10 PM           9689 sub_gui_hhsi.py
-a----        7/26/2019  11:39 AM            122 __init__.py


RiverArchitect\program\SHArC\.templates
Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        7/26/2019  11:39 AM         242416 CONDITION_QvsA_template_si.xlsx
-a----        7/26/2019  11:39 AM         242653 CONDITION_QvsA_template_us.xlsx
-a----        7/26/2019  11:39 AM          17940 CONDITION_sharea_template_si.xlsx
-a----        7/26/2019  11:39 AM          17923 CONDITION_sharea_template_us.xlsx
-a----        7/26/2019  11:39 AM           6172 empty.xlsx


RiverArchitect\program\SHArC\CHSI
Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        7/26/2019  11:39 AM             62 __init__.py


RiverArchitect\program\SHArC\HSI
Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        7/26/2019  11:39 AM             62 __init__.py
```

(continues on next page)

(continued from previous page)

```
RiverArchitect\program\SHArC\SHArea
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         7/26/2019  11:39 AM             62 __init__.py


RiverArchitect\program\Tools
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         7/26/2019  11:39 AM                .templates
-a----        11/25/2019   1:05 PM           6547 cHydraulic.py
-a----         7/26/2019  11:39 AM           8790 cInputOutput.py
-a----        11/25/2019   1:05 PM           6126 cPoolRiffle.py
-a----        11/25/2019   1:05 PM           8129 fTools.py
-a----        11/25/2019   1:05 PM           3112 make_annual_flow_duration.py
-a----        11/25/2019   1:05 PM           1687 make_annual_peak.py
-a----         7/26/2019  11:39 AM           1554 morphology_designer.py
-a----        10/30/2019  12:49 PM           1218 rename_files.py
-a----         7/26/2019  11:39 AM            137 run_make_annual_flow_duration.bat
-a----         7/26/2019  11:39 AM            132 run_make_d2w.bat
-a----         7/26/2019  11:39 AM            124 run_make_det.bat
-a----         7/26/2019  11:39 AM            124 run_make_det_gen.bat
-a----         7/26/2019  11:39 AM            126 run_make_mu.bat
-a----         7/26/2019  11:39 AM            147 run_morphology_designer.bat


RiverArchitect\program\Tools\.templates
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         7/26/2019  11:39 AM          11618 annual_peak_template.xlsx
-a----         7/26/2019  11:39 AM          20867 flow_duration_template.xlsx
-a----         7/26/2019  11:39 AM          11763 input.xlsx
-a----         7/26/2019  11:39 AM          13784 output.xlsx


RiverArchitect\program\VolumeAssessment
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         7/26/2019  11:39 AM                .templates
d-----         7/26/2019  11:39 AM                Output
-a----        11/21/2019   6:07 PM          10217 cVolumeAssessment.py
-a----         7/26/2019  11:35 AM           7013 volume_gui.py
-a----         7/26/2019  11:39 AM             79 __init__.py


RiverArchitect\program\VolumeAssessment\.templates
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         7/26/2019  11:39 AM          10125 volumes_template.xlsx


RiverArchitect\program\VolumeAssessment\Output
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         7/26/2019  11:39 AM             62 __init__.py
```