Lower Yuba River CEQA Best Available Science

Habitat Enhancement Design And Analysis

Python based creation of Lifespan Maps for Stream Restoration

Manual of the feature_analysis package and GUI

UC Davis | March 2018

written by Sebastian Schwindt

Contents

1 Introduction

1.1 Signposts

The feature_analysis package serves for the GIS – based planning enhancement of stream restoration features regarding their lifespans, design characteristics and optimum placement in the terrain. This manual is structured as follows:

Section 1:	Outline of the package structure, requirements and capabilities.
Section 2:	Description of required input rasters.
Section 3:	Quick Guide to the application of the code using GUI.
Section 4:	Alternatives to the GUI.
Section 5:	Descriptions of outputs and procedures for half-automated pdf-map generation.
Section 6:	Physical explanations of parameters, hypotheses and restoration features.
Section 8:	Troubleshooting – Explanations of error and warning messages, including possible causes and remedies.
Section 9:	Detailed explanations of coding conventions with descriptions of extension possibilities.

1.2 Code structure and requirements

The feature_analysis package includes methods executing the GIS modules that require ArcGIS with the "Spatial Analyst" extension and its Python distribution installed. The appropriate `python` interpreter is typically stored in `C:/Python27/ArcGISXX.X`. The package applies on the following `python` packages, which are part of the standard ArcGIS – `python` installation: arcpy, arcpy.sa, argparse, logging, os, subprocess (not mandatory, also works without this package) and sys.

The main file (feature_analysis) imports the following proper sub-modules:

- `classes_analysis.py`	GIS-based functional core providing class ArcPyAnalysis() that produces `raster` files
- `classes_features.py`	Contains features objects with relevant parameters and threshold values
- `classes_gravel.py`	Similar to `classes_features.py` and contains subfeatures of the Gravel augmentation-feature
- `classes_mapper.py`	Provides class Mapper() that contains functions for creating layouts (`.mxd`) and map assemblies (`.pdf`)
- `classes_make_gui.py`	Contains classes for GUI controls
- `classes_parameters.py`	Parameter input core with pointers to `rasters` and `raster` names.
- `classes_plants.py`	Similar to `classes_features.py` and contains subfeatures of the Plantings-feature
- `functions_global.py`	Assembly of compact functions with multi-purpose use

Moreover, the package requires the directory `Input/`*`condition`* to be located in the same folder as the `.py`-files. Section 2 explains the preparation of this directory.
The directory `OutputMaps/.ReferenceLayouts` is essential for class Mapper(). Section 7.2 illustrates possibilities and procedures for adapting map layouts.
Any folder beginning with a "." as for example `.cache`, `.idea` or `.ReferenceLayouts` must not be modified or assessed by any other program, in particular during the execution of package methods.
At the end of an execution, the applied modules have created their output folders, which are indicated in the command prompt. In addition, `logfiles.log` is created in the code directory.

1.3 Assessment of restoration features and habitat delineation

Restoration features are parameter-wise processed for obtaining lifespan and / or design maps, which also indicate where the application of features makes sense. Accordingly, the package can create `rasters`, `mxd`-layouts and `pdf`-maps of the following types:

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

The following list provides an overview on implemented features where the "shortname" is used for the raster, layout and map output. The list indicates the feature-related map types (lifespan and / or design, cf. detailed descriptions in Sec. 5).

The program provides the option of limiting restoration feature maps to zones of low habitat suitability (see details in Sec. 6.2). A raster containing pixel-wise habitat suitability indices is required to determine low habitat quality associated to values ranging between 0.0 and 0.4. A spatial join limits relevant areas for stream restoration features within a radius that can be specified by the user.

Implemented Restoration Features

- Backwater, representative for swale and slackwater creation
  shortname: backwtr; Lifespan maps = True; Design maps = False;
- Berm Setback (Widening)
  shortname: widen; Lifespan maps = False; Design maps = True;
- Engineered Log Jams, representative for all wood-like placements such as rootstock placement
  shortname: egl; Lifespan maps = True; Design maps = True;
- Grading of terrain (Bar and Floodplain Lowering)
  shortname: grade; Lifespan maps = True; Design maps = False;
- Incorporation of Fine Sediment in Soils where plantings are foreseen
  shortname: finesed; Lifespan maps = True; Design maps = True;
- Plantings include:
  - (Fremont) Cottonwood (*Populus Fremontii*)
    shortname: plants_cot; Lifespan maps = True; Design maps = False;
  - Box Elder (*Acer Negundo*)
    shortname: plants_box ; Lifespan maps = True; Design maps = False;
  - White Alder (*Alnus rhombifolia*)
    shortname: plants_whi; Lifespan maps = True; Design maps = False;
  - Willows (*Salix Goodingii / Divers*)

  shortname: Plants_wil; Lifespan maps = True; Design maps = False;
- Riprap, representative for bolder or rock placements
  shortname: riprap; Lifespan maps = True; Design maps = True;
- Sediment Replenishment / Gravel Augmentation or Stockpiles
  shortname: gravelaug; Lifespan maps = True; Design maps = True;
- Side Cavities (Bank Scalloping or Groins)
  shortname: sidecav; Lifespan maps = False; Design maps = False;
- Side Channels, representative for Anabranches, Multithread- or Anastomosed Channels and Flood Runners
  shortname: sidechan; Lifespan maps = False; Design maps = True;

2 Preparation of input rasters

A `condition` parameter defines the situation to analyze and the required raster files need to be stored in the folder `Input/condition/`. For example, if the habitat lifespans need to be assessed based on the 2008-situation, the condition is "2008" and the raster input folder is `Input/2008/`. The rasters require the ArcGIS GRID format, i.e., `raster_name.aux.xml` + `raster_name` folder that contains the required *adf* and *xml* files.
The names of relevant raster files are defined in a proper file format (*.inp*), which can be changed using any text editor by clicking on "InputDefinitions.lnk" (link to `Input/.templates/input_definitions.inp`) or directly from the GUI. The *.inp* files indicates where it requires singles rasters only (`STRING`) or list of rasters (min. two

rasters, `LIST`). The maximum number of accepted rasters per line (parameter) is six; i.e., the package cannot compare more than 6 flow scenarios. This limit is because of computation time and robustness. Moreover, the lifespans related to the hydraulic rasters are defined in the *.inp* file. Section 7.1 provides more detailed information on input definitions. The base case's *.inp* file provides an example for flood scenarios of lower Yuba River corresponding to flood return periods of $<$ 1.0 year, 1.2 years, 2.5 years, 4.7 years, 12.7 years and 20.0 years. Accordingly, the following rasters need to be available in `/Input/condition/` for the feature analysis at lower Yuba River.

Flow velocity (in fps):

- `u001k` substitute for 85-day submergence
- `u005k` 1.2-years flood velocities
- `u021k` 2.5-years flood velocities
- `u042k` 4.7-years flood velocities
- `u084k` 12.7-years flood velocities
- `u110k` 20.0-years flood velocities

Flow depth (in ft):

- `h001k` substitute for 85-day subm.
- `h005k` 1.2-years flood depths
- `h021k` 2.5-years flood depths
- `h042k` 4.7-years flood depths
- `h084k` 12.7-years flood depths
- `h110k` 20.0-years flood depths

Topographic change (in ft):

- `dodfill` 2006/2008–2014 deposition heights
- `dodscour` 2006/2008–2014 scour depths

Depth two water table (in ft):

- `wt_depth_base` referring to base flows of 530–880 cfs

Morphologic Units (string):

- `mu` from Polygons (cell size: 3)

D_{mean} valley (in ft):

- `dmean_ft` mean valley grain size

DEM (in ft a.s.l.):

- `dem` Digital Elevation Model

DEM detrended (in ft):

- `dem_detrend` Detrended Digital Elevation Model

Habitat suitability (cHSI):

- `max_chsi` Major fish species habitat suitability (COVER)

Background image

- `back` Black & White image of LYR from 2009 (NAD83)

Side channel

- `sidech` Side channel delineation

Wildcard

- `wild` 0/nodata (= off) and 1 (= on) values for any purpose to confine analysis

Some parameters, such as the dimensionless bed shear stress or the mobile grain size, can be directly computed from the flow velocity and / or depth. Prepared input rasters could be used for every parameter but this approach requires large storage capacity on the hard disk and it is less flexible regarding computation methods, e.g., the application of roughness laws or derivation of the energy slope. Therefore, the code has its own routines for calculating parameters such as the dimensionless bed shear stress or mobile grain sizes.

The arcpy package will not handle pixels with `noData` values as desired, and therefore, the code uses its own routines to handle this problem. However, the hydraulic input rasters (flow depth and flow velocity) need to entirely cover the region of interest, even though there are basically `noData` pixels: The `h`*`XXX`*`k` and `u`*`XXX`*`k` input rasters may not have such `noData` pixels. This can be fixed by applying the following formula either in python using the `arcpy.sa` package or in the ArcGIC Desktop application using the `Raster Calculator`: ***Con((IsNull("hXXXk")== 1) , (IsNull("hXXXk")∗ 0), "hXXXk")*** for flow depth and ***Con((IsNull("uXXXk")== 1), (IsNull("uXXXk")∗ 0), "uXXXk")*** for flow velocity. The `XXX` values indicate that the formulae need to be applied to all `h` and `u` rasters.

3 GUI Quick Guide

The package is designed for an ArcGIS Python **x64** interpreter, which is typically stored in the directory `"C:\Python27\ArcGISx64XX.X\python.exe"`. Please note the importance of using the **x64** version: The 32-bit version will result in `ERROR 999998: Unexpected Error`. The fastest and most consistent way is using the graphical user interface (GUI) as follows (WINDOWS):

1. Right-click on `LAUNCH_Windows_x64.bat` and choose *Edit with Texteditor* or *Open with ...* and choose a *Texteditor* software
   Hint: It may be necessary to rename `LAUNCH_Windows_x64.bat` to `LAUNCH_Windows_x64.TXT` to have access to the Open with ... menu.
2. Check and if necessary replace the path to the good python interpreter:
   Default: `"C:\Python27\ArcGISx6410.5\python.exe"`
3. Check and if necessary adapt the directory to `make_gui.py`:
   Default: `"D:\Python\FeatureAnalysis\run_gui.py"`
4. Save `LAUNCH_Windows_x64.bat` and close *Texteditor*
5. Double-click on `LAUNCH_Windows_x64.bat`
   The Feature Analysis package starts in a GUI now (Fig. 1).

On UNIX platforms (Apple or Linux), make sure that the python interpreter is version 2.7 and that it can import the arcpy package. Then, open the system terminal, navigate to the directory where the package is installed (location of `.py` files) and type: `./LAUNCH_UNIX.sh`

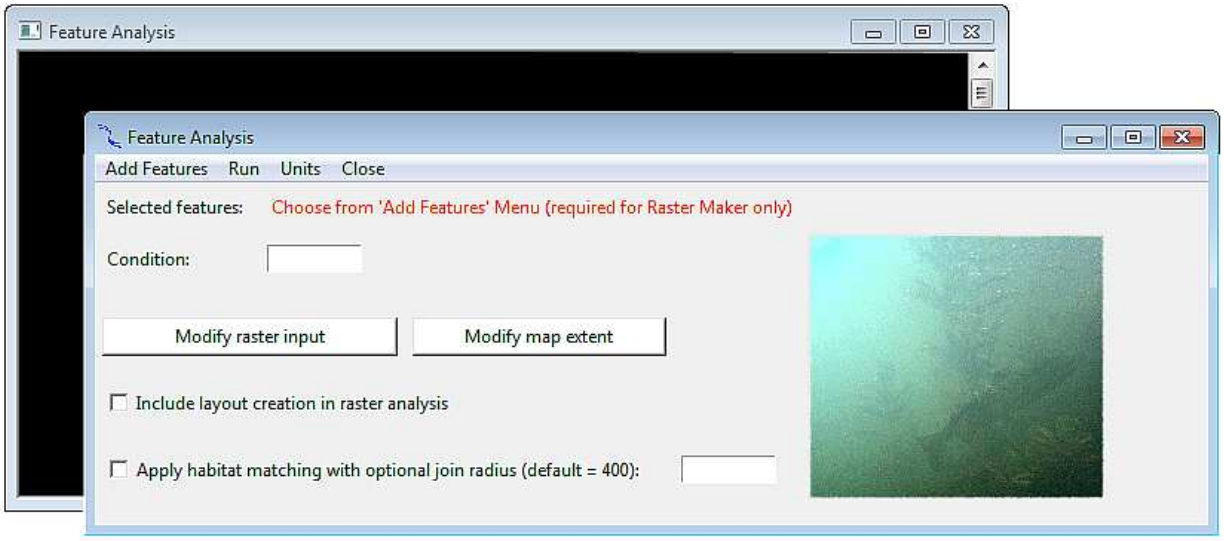

Figure 1: GUI start up window.

Next, click on the drop-down menu “Add Features” and select relevant features. Multiple selection is possible and will extend the list shown after “Selected features:”. Then, a *condition* corresponding to the analyzed situation needs to be typed (see Sec. 2). Modifications of relevant raster names (Sec. 7.1) and map extents (Sec. 7.2) can be made by clicking on the buttons “Modify raster input” and “Modify map extent”, respectively.
The check box “Include layout creation in raster analysis” provides the optional automated preparation of `.mxd` files

for mapping the results (see explanations in Sec. 5.2).
The check box "Apply habitat matching with optional join radius (default = 400)" provides the optional automated preparation of `.mxd` files for mapping the results (see explanations in Sec. 5.2). The radius for the spatial join analysis can be specified in the entry-field behind the habitat matching check box. If habitat matching is checked without specifying a radius, the default value of 400.0 is assigned to the radius (automated units in ft or m if the unit system is U.S. customary or SI, respectively). 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 raster files.

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:

- Raster Maker prepares lifespan and design rasters in the directory `OutputRasters/`*`condition`*`/`
- Layout Maker prepares `.mxd` layouts in the directory `OutputMaps/`*`condition`*`/Layouts`;
  by default the layout maker applies on the rasters stored in `OutputRasters/`*`condition`*`/` but it also accepts other raster input directories as an optional argument when the package is used without GUI (see Alternative Run options in Sec. 4).
- Map Maker prepares map assemblies (`pdfs`) in the directory `OutputMaps/`*`condition`*`/`;
  by default the maps are created based on the layouts stored in `OutputMaps/`*`condition`*`/` but the method also accepts other layout input directories as an optional argument when the package is used without GUI (see Alternative Run options in Sec. 4)

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. The Raster Maker takes 2 to 240 minutes, depending on the feature set. The Layout Maker requires that rasters exist in the `OutputRasters/`*`condition`*`/` directory. After the Layout creation, manual intervention is required to run Map Maker (see explanations in Sec. 5.2).

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

The following sections describe alternative run options and post processing options for map creation.

4 Run options

The package has three principal methods that can be individually called from the GUI and the design of the package is made for the consecutive call of the methods:

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

When the Layout Maker or Map Maker should be applied to another folder than `OutputRasters/condition/` it makes sense to opt for one of the following alternatives to the GUI. The first alternative is importing the package feature_analysis in the ArcGIS Python **x64** interpreter as follows:

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

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

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

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

- Basic execution: fa.raster_maker("condition"), for example: fa.raster_maker("2008")
- The code is now running (this takes two to four hours) and it will prompt its activities.
- Alternatively, the analysis can be limited to some features only (count 2 to 30 minutes per feature). raster_maker accepts optional arguments. which are `feature_list`, which enables the analysis of any feature listed in Sec. 1.3, and mapping, which calls layout (`mxd`) creation. Some examples for particular applications:
  → Example 1: fa.raster_maker("2008", ["Plantings"]) analyses plantings only. → Example 2: fa.raster_maker("2008", ["Plantings", "Riprap"], True) analyses plantings and riprap only with an optional argument True that activates the creation of layouts for plantings and riprap. → Example 3: fa.raster_maker("2008") analyses all available features (see Sec. 1.3).
- The complete list of optional arguments of fa.raster_maker(...) is as follows:
  Hint: Respecting the order of optional arguments is crucial to ensure proper application of the desired analysis options.

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

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

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

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

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

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

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

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

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

fa.map_maker("condition", *args) for creating `pdf` map assemblies requires layout files `.mxd` prepared by either fa.raster_maker("condition", ["Featurename"], True or fa.layout_maker("condition". After either method has created layout files `.mxd`, manual intervention is required because of an `arcpy` deficiency: called outside of *ArcMap Desktop*, `arcpy` works as a background process that cannot actively change layer symbology. The package has an own `ServerStyle` file stored in `.../OutputMaps/.ReferenceLayouts`, which defines the legend style. Currently *ArcGIS* can apply the styles of any `.ServerStyle` to the legend only but not to layers, even though the styles are contained in the file. For more information, follow the discussion on GeoNet.

In the meanwhile, manual intervention is required as explained in the Output-Section 5. Also fa.map_maker("condition", *args) accepts an optional argument defining an alternative layout input path:

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

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

1. Launch terminal
   Windows: Launch `cmd`
   Mac OS: Launch `Terminal.app`
   Linux: Open terminal
2. On Windows: navigate to the place where *ArcGIS* `python.exe` is stored:
   For example: C:\Python27\ArcGISx64XX.X\ and pay attention using
3. Run feature_analysis as script:
   - Windows: python.exe DriveLetter :\...\ FeatureAnalysis\ feature_analysis "condition" ["Featurename"]
   - Linux python .../ feature_analysis "condition" ["Featurename"]
     Hint: Ensure that python calls the correct version used by arcpy.
4. The code asks for a condition, which needs to be typed case-sensitive and without any apostrophes:
   For example: Enter the condition (shape: >> XXXX, e.g., >> 2008)>> 2008

5. Next, the code asks for a feature_list , which is and optional argument (simply hitting enter will work, too); the feature list must be typed as list (in brackets):
   For example: Enter the condition (no mandatory; do not forget brackets – example: >> ["Featurename1", "Featurename2] >> ["Sidecavity", "Bermsetback"]

6. The code is now running - this takes time - and it will prompt when it finished.

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

5 Output

5.1 Rasters

The output rasters are either of the types lifespan (`lf_`*`shortname`*) or design (`ds_`*`shortn`*) and they are created in `.../OutputRasters/`*`condition`*`/`. The usage of *`shortname`*s (see list in Sec. 1.3) is necessary because `arcpy` does not accept saving rasters with names that exceed 13 characters. Shortnames of lifespan rasters may not exceed 10 characters and those of design rasters need to be shorter than 6 characters because the code writes additional information to the name regarding the design parameter. The design parameter names correspond to cached raster names, which are explained in the package details (Sec. 9.2). The analysis automatically shortens too long shortnames and it creates the output directory if it does not yet exist. Existing files in the `OutputRasters/`*`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).

5.2 Layouts and Maps

The package provides a half-automated routine for mapping the rasters in `pdf`s. Full automation is not possible because when `arcpy` is called outside of an *ArcMap-Desktop* application, it runs as a background process, which cannot transfer the symbology from any layer or feature to another layer or feature (see above comments in Sec. 4). The following workflow can be used to obtain `pdf` maps of all rasters from `OutputRasters/`*`condition`*.

1. Prepare layouts
   (a) GUI: Either check button before running raster analysis or click on "Run" and "Run: Layout Maker"
   (b) Alternative `python` console: Either use fa.raster_maker(condition, 1) of fa.layout_maker(condition):
   - Calling the raster_maker with the optional argument "1", e.g., fa.raster_maker("2008", 1) calls the function fa.layout_maker based on prepared layouts for lifespan and design maps.
   - Directly call fa.layout_maker("condition") for creating layouts only
   - Directly call fa.layout_maker("condition"), r"D:/Alternative/Raster/Directory/" for creating layouts from a directory that differs from `OutputRasters/`*`condition`*

2. Python now prepares layout files (`.mxd`) in the folder `OutputMaps/`*`condition`*`/Layouts/` corresponding to the raster names in `OutputRasters/`*`condition`*.

3. Open each layout file (`lf_ ....mxd` and `ds_....mxd`) in *ArcMap-Desktop* and use the following procedure to apply the symbology:
   (a) In the Table of Contents, double-click on the gray-scaled `temp` layer for accessing the *Properties* window.

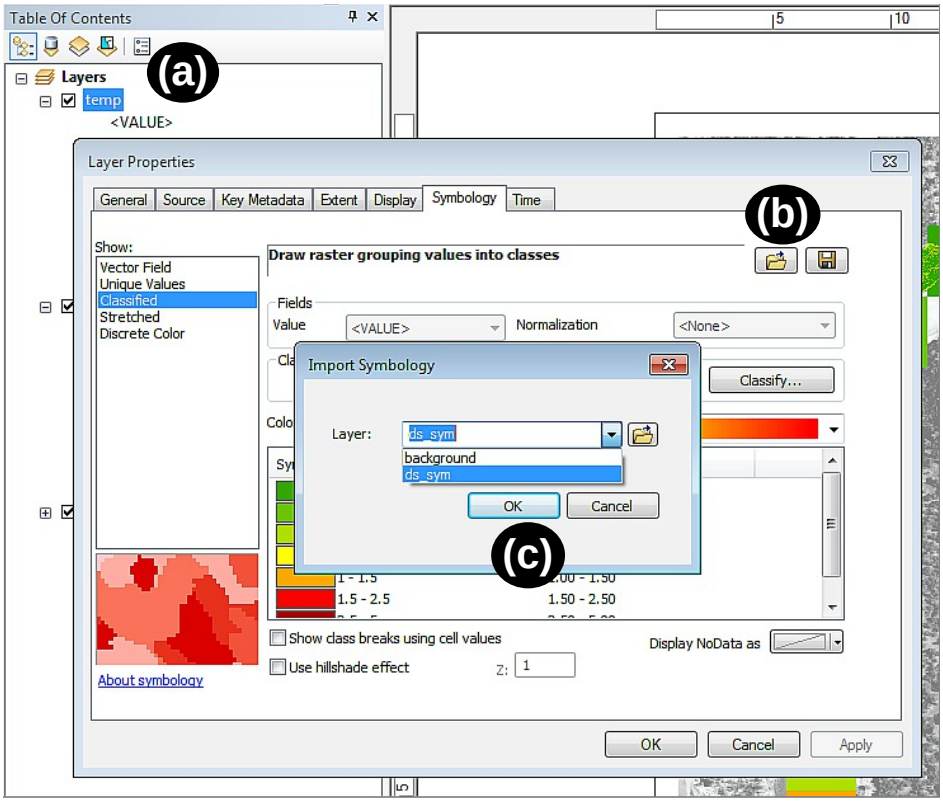

Figure 2: Steps a) to c) for adapting the symbology in *ArcGIS Desktop* according to the descriptions in the text.

(b) Go to the *Symbology* tab and click on "Classified" (computing histograms is required, if queried). Click on "Import..." button (folder symbol in the top-right corner) and select `lf_sym_ras` (for lifespan maps) or `ds_sym_ras` (for design maps).
Hint: Some layouts contain on/off ("NoData+/1) values only. In these cases, "Unique Values" apply instead of "Classified".

(c) Click OK and the gray layer should adapt its colors.

(d) Save and exit the `.mxd` file.

4. Run Map Maker

   (a) GUI: Click on "Run" and "Run: Map Maker"

   (b) Alternative `python` console: type and run `fa.make_maps(condition)`, which produces a `pdf` catalog of each layout.

5. Find the maps in the `OutputRasters/`*`condition`*`/Layouts/` directory.

The package uses layouts that are placed in the directory `.../OutputMaps/.ReferenceLayouts/`, which should not be changed unless the `pdf` style requires adaptations. The map extents, scales and focus can be changes in the `mapping.inp` file (see Sec. 7.2).

5.3 Interpretation

The success of features corresponds to their ecological sustainability and physical stability, which positively correlate in most of the cases, i.e., high stability corresponds to high ecological sustainability. However, features such as gravel

augmentation or grading have an inverse relationship of ecological sustainability and physical stability. For example in the case of gravel augmentation, frequently mobile gravel is important for valuable habitat. In such cases, the lifespan maps need to be considered in the opposite way: Optimum areas for application correspond to regions with low lifespans.

6 Feature and parameter hypothesis

6.1 List of parameters

Combinations of recurring parameters determine the lifespans of features. The code analyses the following parameters, where the application order (hierarchy) differs from the alphabetic order for reasons of map integrity (see coding conventions in Sec. 9.2 for details).

- `chsi` composite Habitat Suitability Index (dimensionless value between 0 and 1)
- `d2w` is the surface depth to the groundwater table (in ft)
- `det` is the detrended DEM (in ft)
- `Dcr` are mobile or stable grain sizes that are entrained by rare discharges that occur according to a defined return period (see Riprap in Sec. 6.9
- `fill` corresponds to annual sediment deposition rates [Wyrick and Pasternack, 2016, in ft]
- `Fr` is the Froude number corresponding to `u`/(`h` g), where g denotes gravity acceleration (dimensionless hydraulic)
- `h` is the flow depth (in ft)
- `mu` are the morphological units [Wyrick and Pasternack, 2014, strings]
- `Se` is the energy slope (cf. Riprap in Sec. 6.9 and Side channels in Sec. 6.12)
- `scour` corresponds to annual erosion rates [Wyrick and Pasternack, 2016, in ft]
- `sidech` delineation of priority regions for side channels [van Denderen et al., 2017]
- `taux` (or τ_*) is the dimensionless bed shear stress and its critical value $\tau_{*,cr}$ (–)
- `tcd` combines `scour` and `fill` analysis
- `u` is the flow velocity (in fps)
- `wild` wildcard parameter that can only take on/off values (noData, 0 or 1)

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

using the feature .mu_bad list. The following morphological units are considered [Wyrick and Pasternack, 2014]:

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

6.2 Habitat suitability assessment

The application of "habitat matching" limits the application of all analyzed features to regions where the composite habitat suitability index (CHSI) is smaller than 0.4. The CHSI values are stored in the input raster `max_chsi` and they represent the maximum CHSI value of the following species and life stages:

- Chinook Salmon: Spawning (CHSI5c), Fry (CHSI) and Juvenile (CHSI);
- Lamprey: Adult (CHSI12)
- Oncorhynchus mykiss (rainbow trout): Fry (CHSI) and Juvenile (CHSI);
- Steelhead trout (CHSI12)
- Sturgeon: Spawning (CSI)

Moreover, the fish species-related values correspond to the maximum CHSI of the following list of discharges in cubic feet per second (hint: not all discharge bins are available for all species and lifestages):

- 300
- 350
- 400
- 450
- 530
- 600
- 622
- 700
- 800
- 880
- 930
- 1,000
- 2,000
- 3,000
- 4,000
- 5,000
- 7,500
- 10,000
- 15,000
- 21,100
- 30,000
- 42,200
- 84,400
- 110,400

6.3 Backwater

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

- `u` with a threshold of 0.1 fps.

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

6.4 Berm Setback

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

- `mu` using the inclusive method with mu_good = [" lateral bar", "spur dike"].
- `det` detrended DEM with a lower limit of 20 ft and an upper limit of 75 ft.

The complete detrended DEM range of the morphological unit lateral bar covers values between -1.24 ft to 29.5 ft and the morphological unit spur dike covers values between 1.9 ft to 25.9 ft. The `det` limits in the code refer to empiric values corresponding to berm setback features according to USACE and HDR [2016].

6.5 Engineered Log Jams

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

Lifespan maps

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

Design maps

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

Regarding morphological units, riffle-pool and plane bed morphologies are favorable for ELJ placement, where side channel and tributary systems are not convenient for wood placement. ELJs inclusive list is defined as mu_good = [" riffle ", " riffle transition ", "pool", " floodplain ", "island floodplain ", " lateral bar", "medial bar", "run "] and the exclusive list is defined as mu_bad = [" tributary channel", " tributary delta "]. For ELJS, the exclusive approach based on mu_bad applies (see details in the parameter descriptions in Sec. 6.1).

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

6.6 Fine sediment

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

Lifespan maps

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

Design maps

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

The topographic change and depth to water table thresholds correspond to the largest values that any plantings type (cf. Sec. 6.8) supports because only these areas are of interest for the incorporation of fine sediment in soils.

6.7 Grading

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

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

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

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

6.8 Plantings

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

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

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

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

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

6.9 Riprap

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

Lifespan maps

- `taux` with mobility threshold of $\tau_{*,cr}$ equal to 0.047.

- `scour` with a threshold value of 1 ft multiplied with 6 years.

Design maps

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

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

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

where:

D_{cr} is the minimum required riprap size (in INCHES);

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

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

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

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

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

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

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

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

6.10 Sediment replenishment / gravel augmentation

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

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

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

- In-channel gravel injections

  Lifespan maps

  – `mobile_grains` analysis with a minimum frequency of 1.0 years and $\tau_{*,cr}$ threshold of 0.047.

  - `mu` uses the inclusive method with mu_good = ["chute", " fast glide", "flood runner", "bedrock", " lateral bar", "medial bar", "pool", " riffle ", " riffle transition ", "run", "slackwater", "slow glide", "swale", " tailings "]

    Design maps

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

- Floodplain / overbank gravel stockpiles

  Lifespan maps

  - `mobile_grains` analysis with a minimum frequency of 1.0 years and $\tau_{*,cr}$ threshold of 0.047.
  - `scour` with a threshold value of 1 ft per year.
  - `mu` uses the inclusive method with mu_good = [" agriplain ", "backswamp", "bank", "cutbank", "flood runner", " floodplain ", "high floodplain ", " hillside ", "island high floodplain ", "island – floodplain ", " lateral bar", "levee", "medial bar", "mining pit", "point bar", "pond", "spur dike", " tailings ", " terrace "]

    Design maps

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

6.11 Side cavities

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

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

6.12 Side channels / anabranches

Any discrete parameters exist for assessing design or lifespan maps for side channels, anabranches, anastomosed or multithread channels. The identification of splays and bank rigidity requires manual and visual proof. As a support, the code generates discharge-dependent design maps that compare the minimum energy slope `Se,min` with the terrain slope `S0`. The minimum energy slope results from the H-h diagram [Moglen, 2015], based on the assumption that the minimum energy per unit force and pixel H_{min} corresponds to the Froude number Fr = 1 with the critical flow velocity u_c and flow depth h_c. The pixel unitary discharge results from $q = u \cdot h$, where u and h are pixel values from the `u` and `h` rasters. Thus, the following system of equations applies:

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

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

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

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

The package routines use the available discharges and related flow velocity `u` / depth `h` rasters (see Sec. 2) as follows (simplified expressions):

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

This function uses `arcpy.sa`'s `Slope` function with the arguments `PERCENT_RISE` for obtaining percent values instead of degrees and `zFactor` = 1.0 because the x-y-grid units are the same as in z-direction. `g` denotes gravity acceleration of 32.2 ft/s^2.
the resulting design maps are named `ds_sidechn_XXX` where `XXX` represents the three-digits discharge in thousand cfs. The values in these maps indicate pixels with excess energy (S_e / S_0 $>$ 1) that allegedly lead causes erosion. In contrast, pixels with energy shortage (S_e / S_0 $<$ 1) allegedly result in sediment deposition. However, minor topographic change can be expected where the S_e / S_0–ratio is close to unity.

7 Input definition files

7.1 Raster data

The file `input_definitions.inp` is stored on the directory `/Input/.templates/` and can be accessed using the link `InputDefinitions.lnk` directly in the code directory. `input_definitions.inp` contains information about lifespan durations and raster names, which link to rasters containing spatial information as described in Sec. 6.1. The order of definitions and lines must not be changed to ensure the proper functioning of the package. Enter or change information in the corresponding lines, only between the "=" and the "#" signs (the input routines uses these signs as start and end identifiers for relevant information). The package can handle a maximum of six

flood discharge scenarios. This upper limit ensures the robustness of computations even when a raster is corrupted or missing. The following definitions apply line by line:

Lines 1–3	None	Do not change
Line 4	Return periods	Comma-separated list of flood discharge return periods corresponding to the hydraulic rasters; i.e., the first entry after "=" corresponds to the return period of the first velocity and flow depth raster (Lines 11 and 12, respectively)
Lines 5–7	None	Do not change
Line 8	CHSI	One raster name of spatial composite Habitat Suitability Indexes
Line 9	DoD	Comma-separated list of two (first = scour, second = fill) DEM of Differences rasters; if one raster is missing, replace it by double quotation marks, for example scour is missing: ... = "", dodFill # ...
Line 10	Det	One raster name defining the detrended DEM raster
Line 11	u	Comma-separated list defining flow velocity rasters corresponding to discharge return periods (Line 4); replace missing rasters by double quotation marks, for example, when `u` rasters of a return period list of five entries are not available for entries 2 and 4, type ... = u001, "", u003, "", u005 # ... . However, ensure that at least two `u` rasters are defined.
Line 12	h	Comma-separated list defining flow depth rasters corresponding to discharge return periods (Line 4); replace missing rasters by double quotation marks, for example, when `h` rasters of a return period list of six entries are not available for entries 2, 3 and 5, type ... = h001, "", "", h004, "", h006 # ... . Ensure that at least two `h` rasters are defined.
Line 13	Grains	One raster name defining the raster containing mean grain diameters (pay attention on raster units: use feet for U.S. customary)
Line 14	mu	One raster name delineating morphological units according to the definitions in Sec. 6.1
Line 15	d2w	One raster name defining the depth to groundwater table
Line 16	DEM	One raster name defining the digital elevation model
Line 17	sidech	One raster name delineating appropriate sites for side channels
Line 18	wild	One raster name for the spatial confinement of the feature analysis of 0/nodata (= off) and 1 (= on) values for any purpose (wildcard raster)

The package produces results based on the available information only, where any raster name can be substituted with double quotation marks "". However, this lack of information reduces the accuracy of final lifespan and design maps. No maps are produced for feature where the information is insufficient for the analysis. The required information for every feature corresponds to the definitions in Sec. 7.1.

7.2 Map layouts

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

The extent of the map determines the map scale, where the corresponding dx and dy values define the map width and height in ft, respectively. The layout templates (`.mxd` in the directory `.../OutputMaps/.ReferenceLayouts/`

define the paper size, which is by default "ANSI E landscape" (width = 44 inches, height = 34 inches).
The map focus is defined page-wise in `mapping.inp` from Line 8 onward. Existing pages can be removed by simply deleting the line. Additional pages can be added by inserting or appending a new line below Line 8, which needs to begin with the keyword "`Page`" and x and y need to be stated in brackets, separated by a comma without any white space ([xxxxxx.xx,yyyyyy.yy]).

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

8 Error messages and Troubleshooting

Most errors occur when the wrong python interpreter is used or when rasters or layouts have bad formats or when the information stated in the input file (see Sec. 7.1) is erroneous. The package writes process errors and descriptions to logfiles. When the GUI encounters problems, it directly provides causes and remedies in pop-up info boxes. 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.

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

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

Remedy If the `.cache` folder is in use, delete it manually (works sometimes only after logging of and on). If the error occurs after code modifications, make sure that the self . parameter_list s of features (Sec. 9.5) has valid entries that occur in analysis_call (∗args) (`feature_analysis.py`) and that valid function names exist in `classes_analysis.py` (Sec. 9.4).

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

Cause Usage of the wrong python interpreter

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

▷ **`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 analysis_ ... and design_ ... functions in `classes_analysis.py` when raster calculations could not be performed. Missing rasters, bad raster assignments or bad raster calculation expressions are possible reasons. The error can also occur when the `Spatial` license is not available.

Remedy See ExceptionERROR: (arcpy) ...

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

Cause Raised by coordinates_read (self) (in `classes_read_inp.py`) when `mapping.inp` has bad assignments of x-y coordinates.

Remedy Ensure that the coordinate definitions in `mapping.inp` (`Input/.templates`) correspond to the definitions in Sec. 7.2.

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

Cause Raised by get_map_extent (self , direction) (in `classes_read_inp.py`) when `mapping.inp` has bad assignments of x-y coordinates.

Remedy Ensure that the file `mapping.inp` exists in the directory `Input/.templates` corresponding to the definitions in Sec. 7.2. In case of doubts: Replace `mapping.inp` with the original file and re-apply modifications strictly following Sec. 7.

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

Cause Raised by get_map_extent (self , direction) (in `classes_read_inp.py`) when `mapping.inp` has wrong formats or it is missing..

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

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

Cause Raised by get_map_extent (self , direction) (in `classes_read_inp.py`) when `mapping.inp` has bad assignments of x-y coordinates (not a number).

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

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

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

Remedy – Make sure that the self . parameter_list s of features (Sec. 9.5) has valid entries that also occur in analysis_call (∗args) (`feature_analysis.py`).
– Make sure that valid function names exist in `classes_analysis.py` (Sec. 9.4).

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

Cause Raised by get_line_entries (self , line_no) (in `classes_read_inp.py`) when it cannot access input files.

Remedy – Ensure that the file `input_definitions.inp` exists in the directory `Input/.templates` corresponding to the definitions in Sec. 7.1.
– Ensure that the file `mapping.inp` exists in the directory `Input/.templates` corresponding to the definitions in Sec. 7.2.
– In case of doubts: Replace `input_definitions.inp` and `mapping.inp` with the original files and re-apply modifications strictly following Sec. 7.

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

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

Remedy – Make sure to provide at least two pairs of hydraulic (`u` and `h`) rasters that correspond to two different discharges (one `u` and one `h` raster per discharge).
– As a rule of thumb: the more hydraulic rasters provided, the better are the lifespan maps. However, for reasons of consistency, the maximum number of hydraulic rasters in six per `u` and one `h`, i.e., six lifespans.
– Verify raster and corresponding lifespan definitions in `input_definitions.inp` (Sec. 7.1).

▷ **ERROR: Mapping failed.**

Cause The function make_pdf_maps(self, ∗args) (`feature_mapper.py`) raises this error message when the layout files in `OutputMaps/`*`condition`*`/Layouts` are either corrupted or non-existent.

Remedy – Re-run Layout Maker or successively re-run Raster Maker and Layout Maker.
– Follow exactly the instructions for preparing map files (see Sec. 5.2).

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

Cause The function prepare_layout (self) (`feature_mapper.py`) raises this error message when it encounters problems with either the provided rasters (default directory: `OutputRasters/`*`condition`*) or layout templates (`.mxd` files in `OutputMaps/.ReferenceLayouts`).

Remedy If a layout (`.mxd`) in `OutputMaps/.ReferenceLayouts` was modified, ensure similar layer structures in the `.mxd` files corresponding to the existing templates.

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

Cause Error raised by the zoom2map(self, xy) function (`feature_mapper.py`) when it receives a bad format of x-y values.

Remedy Ensure the correct format of `mapping.inp` corresponding to the definitions in Sec. 7.2

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

Cause Raised by `get_line_entries ( self , line_no )` (in `classes_read_inp.py`) when `input_definitions.inp` is corrupted.

Remedy Ensure that the file `input_definitions.inp` exists in the directory `Input/.templates` corresponding to the definitions in Sec. 7.1. In case of doubts: Replace `input_definitions.inp` with the original file and re-apply modifications strictly following Sec. 7.

▷ **ERROR: *PAR* – raster copy to OutputRasters folder failed.**

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

Remedy – Follow trouble shooting of other error messages and re-run.
– Avoid modifications of any folder in the code directory while the program is running, in particular, `.cache`, `Input`, `OutputRasters` and `OutputMaps`.

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

Cause Error message raised by the `choose_ref_layout ( self , raster_name )` function in `feature_mapper.py` when it cannot assign a layout template from `OutputMaps/.ReferenceLayouts` to a raster (default storage directory: `OutputRasters/`*`condition`*).

Remedy – If a layout (`.mxd`) in `OutputMaps/.ReferenceLayouts` was modified, make sure to implement changes also in the `choose_ref_layout ( self , raster_name )` function (`feature_mapper.py`).
– If a new output raster type results from modifications or extensions of the parameters, analysis or feature methods (Sections 9.3, 9.4 and 9.5, respectively), ensure that the conditional phrases in `choose_ref_layout ( self , raster_name )` (`feature_mapper.py`) can identify it and assign an existing layout (`.mxd`) from `OutputMaps/.ReferenceLayouts`.

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

Cause Raised by `get_map_scale ( self )` (in `classes_read_inp.py`) when it cannot interpret the value assigned to the map scale.

Remedy Ensure that the file `mapping.inp` (in `Input/.templates`) has a correct assignment of the map scale according to the descriptions in Sec. 7.2.

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

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

Remedy Ensure that the file `input_definitions.inp` (in `Input/.templates`) contains a lifespan list (return periods list) with not more than six comma-separated entries according to the definitions in Sec. 7.1.

▷ **ExceptionERROR: (arcpy) in *PAR*.**

Cause The error is raised by analysis_ ... and design_ ... functions in `classes_analysis.py` when raster calculations could not be performed. Missing rasters, bad raster assignments or bad raster calculation expressions are possible reasons. The error can also occur when the `Spatial` license is not available.

Remedy – Make sure that a `Spatial` license is available.
– Verify raster calculation expressions in concerned analysis_ ... and design_ ... functions (`classes_analysis.py`).
– Verify raster definitions in concerned analysis_ ... and design_ ... functions (`classes_analysis.py`).
– Verify raster definitions of used parameters (`classes_parameters.py` and input files `*.inp` according to Sec. 7).
– If further system errors are stated, trace back error messages.

▷ **ExecuteERROR: (arcpy) in *PAR*.**

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

Remedy See ExceptionERROR: (arcpy) ...

▷ **WARNING: Could not clear cache.**

Cause The content in the cache folder was accessed and blocked by another software.

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

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

Cause The function prepare_layout (self) (`feature_mapper.py`) print this warning message when it cannot remove the `temp` layer from the layout template.

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

▷ **WARNING: Design map - Could not assign frequency threshold. Using default.**

Cause Design maps, such as stable grain size, refer to hydraulic data related to a defined return period. If design_ ... functions) `classes_analysis.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 `input_definitions.inp`, see Sec. 7.1).

Remedy Go to `classes_features.py`) and define a float value for self . threshold_freq .

▷ **WARNING: Overwriting existing version of *LAYOUT*.mxd**

Cause The directory `OutputMaps/`*condition*`/Layouts` already contains a file named *LAYOUT*`.mxd`, which is overwritten by the program.

Remedy Ensure to save important layout files in another directory if overwriting is not desired.

▷ **WARNING: Raster / layout identification failed. Using lifespan ...**

Cause Warning message from the choose_ref_layer (self , feature_type) function (`feature_mapper.py`) if it cannot determine the raster type, i.e., whether it is a lifespan or a design raster. In this case, the layer symbology of lifespan maps is assign by default, which can cause errors later on.

Remedy – Verify the layout templates (`.mxd`) in `OutputMaps/.ReferenceLayouts` for correct layer names, i.e., "lf_sym" for lifespan and "ds_sym" for design map templates.
– Ensure that all layout templates (`.mxd`) in `OutputMaps/.ReferenceLayouts` names either start with `lf` or `ds` for lifespan and design layouts, respectively.

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

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

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

9 Code extension and modification

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

9.1 Conventions

The rasters creation results from analysis_ and design_ functions that are stored in `classes_analysis`. analysis_ functions create rasters with lifespan data (0 to 20 years) and design_ functions create rasters with design parameters such as the required stables grain size of riprap.
Class names start with an upper case letter and do not contain any special characters, also excluding dash or underscore signs. Instantiations of classes are all lower case letters. Features, Parameters and Analysis classes are stored in separate files called `classes_features.py`, `classes_parameters.py` and `classes_analysis.py`, respectively. In addition, Feature classes may inherit subfeature classes from files names `classes_`*subfeature*`.py`, for example `classes_`*plants*`.py`.
Function names consist of lower case letters only and the underscore sign "_" separates words.
All class names, variable names and function names are in alphabetic order (a = up, z = down), except the parameter_list s, which determine the run hierarchy (see Sec. 9.2).

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

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

1. Flow depth rasters (dimensional) starting with the lowest discharge (5'000 cfs) to the highest discharge (110'400 cfs)
   Internal raster name: `ras_hXXXk`
2. Flow velocity rasters (dimensional) starting with the lowest discharge (5'000 cfs) to the highest discharge (110'400 cfs)
   Internal raster name: `ras_uXXXk`
3. Hydraulic rasters (dimensionless)
   Internal raster name: `ras_taux` (dimensionless bed shear stress) or `ras_Fr` (Froude number); if needed: the hierarchy among the dimensionless hydraulic numbers is not important
4. Mobile bed, fine sediment and stable grain size raster analysis
   Internal raster name: `ras_Dcr` (mobile or stable grain size)
5. Topographic change rasters
   Internal raster names: `ras_fill` (fill raster only), `ras_scour` (scour only) or `ras_tcd` (combined fill and scour)
6. Detrended DEM raster analysis
   Internal raster name: `ras_det` (relevant, e.g., for berm setback)
7. Morphological Unit rasters
   Internal raster name: `ras_mu`
8. Side channel delineation
   Internal raster name: `ras_sch`
9. Depth to water table
   Internal raster name: `ras_d2w` (relevant, e.g., for plantings and terrain grading)

The dimensional hydraulic maps need to be invoked before any other analysis is performed because the u and h maps are the only ones that entirely cover the area of interest, without "`noData`" pixels.
Every feature has a feature . parameter_list attribute containing a list of parameters that determine the feature lifespan and applicability space. The parameters are ordered in the feature . parameter_list according to the hierarchy. Once the last element of feature . parameter_list is processed and stored in the cache folder, the code exits the loop and copies the last `ras_`*`parameter`* to the `OutputRasters/`*`condition`*`/` folder. This copy is renamed `lf_`*`shortname`*, where the usage of shortnames (see list in Sec. 1) is necessary because `arcpy` cannot save or copy raster with names exceeding 13 characters.

9.3 Add parameters

The currently implemented parameters are listed in Sec. 6.1. New parameters require new input rasters in addition to the list in Sec. 2. The rasters need to be saved in the folder `/Input/condition/` using the *Esri Grid* format. Other raster formats such as `.tif` may cause inconsistencies that result in error messages when the code attempts to save the final rasters. The template for creating a new parameter class is shown in the box. Use the following workflow to implement a new parameter in the code:

1. Create *Esri Grid* parameter rasters in the folder `/Input/condition/`.
2. Add a new parameter class in the file `classes_parameters.py` (cf. box explanations).
3. Add a new function called `analyse_parameter` to the `ArcPyAnalysis` class in the file `classes_analysis.py` (see Sec. 9.4) or change existing analysis for using the new parameter.

Add new parameter class to `classes_parameters.py`

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

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

9.4 Add analysis

The analysis routines are differentiated between analyse and design-functions, which are contained in the file `classes_analysis.py`.
analyse-functions return rasters containing estimated survival times (in years) or on/off values (1/0). An analyse-function will always try to find existing rasters produced from previous analysis functions according to the analysis hierarchy (Sec. 9.2), unless a dimensional hydraulic analysis (`u`, `h` or their combination) is performed. For this reason, analyse-function uses the verify_raster_info ()-function to look up for previous analyses that are stored in raster_dict_lf . At the end of an analyse-function, the raster_dict_lf is updated using raster_dict_lf .update(" ras_current "). This serial map-analysis produces lifespan rasters, which can be regardlessly converted to design rasters by the save_manager()-function when the feature properties are set to self .ds = True while self . lf = False (Sec. 9.5).
design-functions produce rasters containing specific parameter values, such as the critical grain size in inches. A design-function will update the raster_dict_ds -dictionary which is passed to the save_manager()-function when the feature variable self .ds = True.
The major difference between the raster_dict_lf and raster_dict_ds -dictionaries is that the save_manager() saves the only the last hierarchy-based entry of raster_dict_lf to produced lifespan rasters but all entries of raster_dict_ds to produced design rasters. The combination of multiple parameters into one design raster can be achieved anyway by setting self .ds = True while self . lf = False (Sec. 9.5), which converts lifespan rasters to design rasters.

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

1. Ensure that all required parameters are available (Parameter list: Sec. 6.1; Add parameters: Sec. 9.3).
2. Create an identifier string of 2 to 3 characters; the following explanations refer to a dummy identifier named NEW (replace with lowercase letters).
3. Add a new analyse_NEW or design_NEW-function in the file `classes_analysis.py` (cf. code example below).
4. In `classes_features.py` ensure that concerned features have the following properties:
   - The feature . parameter_list needs to contain the new analysis' identifier (NEW)
   - All required threshold values are defined (feature .threshold_NEW1 = ...)
5. In `feature_analysis.py`, add a call of the new function:

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

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

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

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

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

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

     arcpy.CheckInExtension('Spatial')

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

9.5 Add features

The currently implemented features are listed in Sec. 1. New features can be implemented in the `classes_features.py` file using the following workflow:

1. Ensure that all required parameters are available (Parameter list: Sec. 6.1; Add parameters: Sec. 9.3).
2. Ensure that all required analysis and / or design functions are available (cf. Sec. 9.4).
3. Choose a name for the new feature beginning with an uppercase letter followed by lowercase letters only; the name Newfeature is subsequently used for illustrative purpose
4. In `classes_features.py` modify the class RestorationFeature :

   - Add inheritance of the new feature (consider alphabetic order):
     class RestorationFeature (Backwater, ELJ, Finesediment, Grading, Gravel, Newfeature, Plantings , Riprap, Sidecavity , Sidechannel, Widen):
   - Implement the new feature instantiation when called by adding the following to def __init__ (self , feature_name , *sub_feature):

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

   - Please note: the shortname should not have more than 6 characters; otherwise the code will cutoff the shortname automatically.

   - Both the initiation __ini__ (self) and the instantiation Newfeature() are necessary to facilitate the external access to Methods and Properties.
   - A feature may have subfeatures (as for example the class Plantings). In this case, replace and not(sub_feature) with and sub_feature and set self .sub = True.

   Add a new class to `classes_features.py` according to the example below, considering the required hierarchically ordered self . parameter_list , self . threshold_ ... and lifespan (self . lf = True / False) / design (self .ds = True / False) raster analysis properties.

5. Additional changes in `feature_analysis.py` are recommended but not required:

   - In def raster_maker (condition , *args) complement feature_list = ["Backwater", "Bermsetback", "ELJ", "Finesediment", "Grading", "Gravel", "Newfeature", "Plantings", "Riprap", "Sidecavity", "Sidechannel"]
   - In if __name__ == "__main__" complement feature_list = ["Backwater", "Bermsetback", "ELJ","Finesediment", "Grading", "Gravel", "Newfeature", "Plantings", "Riprap", "Sidecavity", "Sidechannel"]

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

```
class Newfeature():
  ## This is the Newfeature class.
  def __init__(self):
      self.ds = False # identify if design map applies
      self.lf = True  # identify if lifespan map applies
      self.parameter_list = ["PAR1", "PAR2", ..., "PARi", , "PARn"] # Respect
          Hierarchy -- example: PAR1 = "hyd"
      ## uncomment and adapt follow line if PAR = mu applies
      # self.mu_bad = []
      # self.mu_good = []        # self.mu_method = 1 # 0 = apply mu_bad list
          and 1 = apply mu_good list
      self.threshold_1 =... # (unit) description
      self.threshold_2 = ... # (unit) description
      ...
      self.threshold_i = ...
      ...
      self.threshold_n = ...
  def __call__(self):
      pass
```

All threshold values need to have metric units, independent of the unit system of raster files. The conversion of units is reserved to the analysis classes in `classes_analysis.py` only.

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

```
class Newfeature(Subfeature_1, Subfeature_2, ..., Subfeature_i, ...
    Subfeature_n):
  ## This is the Newfeature class inheriting from Subfeature_1 to Subfeature_n
      .
  def __init__(self, subfeature):
      logger = logging.getLogger("feature_analysis")
      logger.info("* * * * * * * * * * * * * * * * * * * * *")
      logger.info("> Subfeature: " + subfeature)
      logger.info("* * * * * * * * * * * * * * * * * * * * *")
      self.lf = True  # identify if lifespan map applies
      self.ds = False # identify if design map applies
      if subfeature == 'subfeature_1':
          Subfeature_1.__init__(self)
      if subfeature == 'subfeature_2':
          Subfeature_2.__init__(self)
      if subfeature == 'subfeature_i':
          Subfeature_i.__init__(self)
      if subfeature == 'subfeature_n':
          Subfeature_n.__init__(self)
  def __call__(self):
      pass
```

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

```
class Subfeature_i():
  ## This is the Newsubfeature class.
  def __init__(self):
      self.parameter_list = ["PAR1", "PAR2", ..., "PARi", , "PARn"] # Respect
          Hierarchy!; example: PAR1 = "hyd"
      ## uncomment and adapt follow line if PAR = mu applies
      # self.mu_bad = []
      # self.mu_good = []
      # self.mu_method = 1 # 0 = apply mu_bad list and 1 = apply mu_good list
      self.threshold_1 = ... # (unit) description
      self.threshold_2 = ... # (unit) description
      ...
      self.threshold_i = ...
      ...
      self.threshold_n = ...
  def __call__(self):
      pass
```

Again, all threshold values need to have metric units, independent of the unit system of raster files. The conversion of units is reserved to the analysis classes in `classes_analysis.py` only.

References

Sharon Bywater-Reyes, Andrew C. Wilcox, John C. Stella, and Anne F. Lightbody. Flow and scour constraints on uprooting of pioneer woody seedlings. *Water Resources Research*, 51(11):9190–9206, 2015. ISSN 1944-7973. doi: 10.1002/2014WR016641. URL http://dx.doi.org/10.1002/2014WR016641.

Jonathan M. Friedman and Gregor T. Auble. Mortality of riparian box elder from sediment mobilization and extended inundation. *Regulated Rivers: Research & Management*, 15(5):463–476, 1999. ISSN 1099-1646. doi: 10.1002/(SICI)1099-1646(199909/10)15:5$\langle$$463::AID-RRR559$\rangle$3.0.CO;2-Z. URL http://dx.doi.org/10.1002/(SICI)1099-1646(199909/10)15:5$<$463::AID-RRR559$>$3.0.CO;2-Z.

David Gaeuman. Recommended quantities and gradation for long-term coarse sediment augmentation downstream from Lewiston Dam. Technical Report TM-TRRP-2008-2, Trinity River Restoration Program, Weaverville, CA, USA, 2008.

P. Jablkowski, E. A. Johnson, and Y. E. Martin. Role of river flow and sediment mobilization in riparian alder establishment along a bedrock-gravel river, south fork eel river, california. *Geomorphology*, 295(Supplement C):28–38, 2017. ISSN 0169-555X. doi: 10.1016/j.geomorph.2017.06.010. URL http://www.sciencedirect.com/science/article/pii/S0169555X1730274X.

Li Kui and John C. Stella. Fluvial sediment burial increases mortality of young riparian trees but induces compensatory growth response in survivors. *Forest Ecology and Management*, 366(Supplement C):32–40, 2016. ISSN 0378-1127. doi: 10.1016/j.foreco.2016.02.001. URL http://www.sciencedirect.com/science/article/pii/S0378112716300123.

Daniela Lange and Gian Reto Bezzola. *Schwemmholz Probleme und Lösungansätze [Driftwood problems and approaches for solutions]*. Minor, H.-E, ed.: Mitteilung Nr. 188 der Versuchsanstalt für Wasserbau, Hydrologie und Glaziologie an der Eidgenössischen Technischen Hochschule Zürich, Zürich, Switzerland, 2006. in German.

Steve Maynord and Charles Neill. *Sedimentation Engineering*, chapter APPENDIX B - Riprap Design, pages 1037–1056. ASCE Manual an Reports on Engineering Practice No. 110. American Society of Civil Engineering, García, M. H. (ed.), Reston, VA, USA, 2008. doi: 10.1061/9780784408148. EM-1601.

Glenn E. Moglen. *Fundamentals of Open Channel Flow*. CRC Press, Taylor & Francis Group, Virginia Tech, VA, USA, 2015. ISBN 978-1466580060. 270 pages.

Giyoung Ock, Tetsuya Sumi, and Yasuhiro Takemon. Sediment replenishment to downstream reaches below dams: implementation perspectives. *Hydrological Research Letters*, 7(3):54–59, 2013. doi: 10.3178/hrl.7.54. URL https://www.jstage.jst.go.jp/article/hrl/7/3/7$_$54/$_$article.

N. Pasquale, P. Perona, P. Schneider, J. Shrestha, A. Wombacher, and P. Burlando. Modern comprehensive approach to monitor the morphodynamic evolution of a restored river corridor. *Hydrology and Earth System Sciences*, 15 (4):1197–1212, 2011. doi: 10.5194/hess-15-1197-2011. URL https://www.hydrol-earth-syst-sci.net/15/1197/2011/.

N. Pasquale, P. Perona, R. Francis, and P. Burlando. Effects of streamflow variability on the vertical root density distribution of willow cutting experiments. *Ecological Engineering*, 40(Supplement C):167–172, 2012. ISSN 0925-8574. doi: 10.1016/j.ecoleng.2011.12.002. URL http://www.sciencedirect.com/science/article/pii/S0925857411003697.

N. Pasquale, P. Perona, R. Francis, and P. Burlando. Above-ground and below-ground Salix dynamics in response to river processes. *Hydrological Processes*, 28(20):5189–5203, 2014. ISSN 1099-1085. doi: 10.1002/hyp.9993. URL http://dx.doi.org/10.1002/hyp.9993.

Gregory B. Pasternack, Scott L. Morford, and Aaron A. Fulton. Yuba River analysis aims to aid spring-run chinook salmon habitat rehabilitation. *California Agriculture*, 64(2):69–77, 2010. ISSN 0008-0845. doi: 10.3733/ca.v064n02p69.

Mary Louise Polzin and Stewart B. Rood. Effective disturbance: Seedling safe sites and patch recruitment of riparian cottonwoods after a major flood of a mountain river. *Wetlands*, 26(4):965–980, 2006. doi: 10.1672/0277-5212(2006)26$\%$5B965:EDSSSA$\%$5D2.0.CO;2. URL https://link.springer.com/article/10.1672/0277-5212(2006)26$%$5B965:EDSSSA$%$5D2.0.CO;2.

Virginia Ruiz-Villanueva, Barllomiej Wyzga, Joanna Zawiejska, Maciej Hajdukiewicz, and Markus Stoffel. Factors controlling large-wood transport in a mountain river. *Geomorphology*, 272(Supplement C):21–31, 2016. doi: 10.1016/j.geomorph.2015.04.004.

J. C. Stromberg, B. D. Richter, D. T. Patten, and L. G. Wolden. Response of a Sonoran Riparian forest to a 10-year return flood. *The Great Basin Naturalist*, 53(2):118–130, 1993. ISSN 0017-3614. URL http://www.jstor.org/stable/41712765.

USACE. Design and Construction of Levees. In *Engineering and Design*, number EM 1110-2-1913 in Engineer Manual, page 164. U.S. Army Corps of Engineers, Washington, DC, USA, April 2000.

USACE and HDR. Yuba River Ecosystem Restoration Feasibility Study (YRERFS) - Habitat Measures. Technical report, Yuba County Water Agency (YCWA), Sacramento, CA, 2016.

R. Pepijn van Denderen, Ralph M. J. Schielen, Astrid Blom, Suzanne J. M. H. Hulscher, and Maarten G. Kleinhans. Morphodynamic assessment of side channel systems using a simple one-dimensional bifurcation model and a comparison with aerial images. *Earth Surface Processes and Landforms*, [in press]:14, 2017. ISSN 1096-9837. doi: 10.1002/esp.4267. URL http://dx.doi.org/10.1002/esp.4267. esp.4267.

Andrew C. Wilcox and Patrick B. Shafroth. Coupled hydrogeomorphic and woody-seedling responses to controlled flood releases in a dryland river. *Water Resources Research*, 49(5):2843–2860, 2013. ISSN 1944-7973. doi: 10.1002/wrcr.20256. URL http://dx.doi.org/10.1002/wrcr.20256.

J. R. Wyrick and G. B. Pasternack. Geospatial organization of fluvial landforms in a gravel-cobble river: Beyond the riffle-pool couplet. *Geomorphology*, 213(Supplement C):48–65, 2014. ISSN 0169-555X. doi: 10.1016/j.geomorph.2013.12.040. URL http://www.sciencedirect.com/science/article/pii/S0169555X14000099.

Joshua R. Wyrick and Gregory B. Pasternack. Revealing the natural complexity of topographic change processes through repeat surveys and decision-tree classification. *Earth Surface Processes and Landforms*, 41(6):723–737, 2016. ISSN 1095-9837. doi: 10.1002/esp.3854. ESP-15-0190.R1.