

COASTAL DYNAMICS ANALYZER (CDA): USER GUIDE

A QGIS Plugin for Transect Based Analysis of Coastal Erosion.



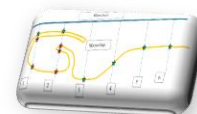
UNIVERSITÀ DEGLI STUDI DI PALERMO

d*i* dipartimento
di ingegneria
unipa



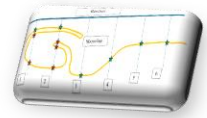
1 GIUGNO 2024

UNIVERSITY OF PALERMO, MARINE AND COASTAL ENGINEERING LAB
Department of Engineering, Palermo, Viale delle Scienze, Bd. 8, 90128 Palermo (Italy)



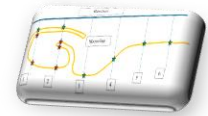
Summary

1) INTRODUCTION.....	3
2) QGIS BUILDING VERSION.....	4
3) PLUGIN STARTING	7
Creation of the Baseline:	8
Rules of Passages:	8
Usage:	8
Additional Information:	8
Credits:.....	9
Control Buttons:.....	9
4) Step 0.....	9
5) Step 1	12
6) Step 2.....	14
7) Step 3.....	17
8) Step 4.....	17
9) Step 5.....	18
10) PSEUDOCODE FRAMEWORK EXPLANATION.....	21



Listo of Figures

Figure 1. QGIS version details (recommended version).....	4
Figure 2. CDA main GUI.....	7
Figure 3. Window to select the folder in which there is ONLY and EXCLUSIVELY the initial baseline in shapefile format (Step 0).....	10
Figure 4. Window to select the folder where the shorelines are in shapefile format (Step 0).	10
Figure 5. Window to select the folder in which to save the baseline in shapefile format (Step 0).	11
Figure 6. Window to select a folder in which to save temporary processing files (TEMP) (Step 0).	11
Figure 7. Window to select values for Distance between transects in meters and Length of transects in meters (Step 0).	12
Figure 8. Step 1 window. Calc_Transect algorithm.	13
Figure 9. Step 2 window.	15
Figure 10. The diagram illustrates the process of selecting transects for repair in shoreline analysis. It features: 1) Baseline: The starting line for the transects and 2) Shoreline: The line where the transects intersect. Transects: Lines labeled 1 through 6 extending from the baseline to the shoreline. The Green Stars: Indicate correct transect-shoreline intersections while Red Stars: Indicate incorrect transect-shoreline intersections.	16
Figure 11. Example of step 2 result, selected transect.	16
Figure 12. Step 3 CDA window. Function "Save_Selected".	17
Figure 13. Step 4 CDA window.	18
Figure 14. Step 5 CDA window. Algorithm TRANSECT_Clip.	19
Figure 15. Example of batch process setup to execute step 5 CDA in series for 7 different shorelines.	20



1) INTRODUCTION

Studies of shoreline evolution are crucial for assessing beach accretion or retreat, influencing erosion management strategies and coastal hazard adaptation plans. With the advent of GIS technologies and satellite imagery, analyses of coastal dynamics have become faster and more accurate. However, there are still no plugins for QGIS that fully automate these analyses based on Transect-Based Analysis (TBA) methods.

The Coastal Dynamics Analyzer (CDA) is a new plugin for QGIS, developed to provide an automatic method for Shoreline Change Analysis (SCA), improving the accuracy and speed of the analysis. This tool was developed starting by the study carried out by Manno et al. 2022.

The CDA plugin is written in PyQGIS (version v.1.0.0). It allows you to define a piecewise polynomial baseline and generate transects for shoreline change analysis. CDA allows commonly used rate-of-change such as End Point Rate (EPR), Net Shoreline Movement (NSM), Shoreline Change Envelope (SCE), and Linear Regression Rate (LRR) to be calculated, providing reports in .csv and shapefile formats.

Easily installed using the QGIS plugin manager or as a Python script, the plugin is based on QGIS algorithms and scientific libraries. It provides an intuitive environment for analysing coastal transects and, due to its speed and accuracy, make it suitable for regional studies and for providing parameters needed for erosion management strategies. In addition, CDA is distinguished by its ability to perform detailed and automated analyses, supporting the scientific community, industry professionals, and coastal managers in monitoring coastal changes, identifying erosion-prone areas, and evaluating the effectiveness of mitigation measures.



2) QGIS BUILDING VERSION

The CDA plugin was developed and tested on QGIS version 3.34.2-Prizren. To ensure proper functioning of the plugin, it is recommended to use this version of QGIS or a later one. Below are details of the version of QGIS used to develop the plugin.

QGIS Version Details.

Image 1 shows the specifications of the version of QGIS on which the CDA plugin was built:


			
Versione di QGIS	3.34.2-Prizren	Revisione codice QGIS	7d199797fc
Versione Qt	5.15.3		
Versione Python	3.9.5		
Versione GDAL/OGR	3.8.2		
Versione PROJ	9.3.1		
Versione database del Registro EPSG	v10.098 (2023-11-24)		
Versione GEOS	3.12.1-CAPI-1.18.1		
Versione SQLite	3.41.1		
Versione PDAL	2.6.0		
Versione client PostgreSQL	15.2		
Versione Spatialite	5.1.0		
Versione QWT	6.1.6		
Versione QScintilla2	2.13.4		
Versione SO	Windows 10 Version 2009		
Plugins Python attivi			
joinmultiplelines	Version 0.4.1		
pluginbuilder3	3.2.1		

Figure 1. QGIS version details (recommended version).

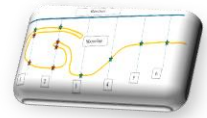
Active Python Plugins

For the CDA plugin to work properly, it is essential to have the following Python plugins active:

- joinmultiplelines: Version 0.4.1
- pluginbuilder3: Version 3.2.1

Recommendations

To install and use the CDA plugin, make sure that your QGIS configuration matches the one above. In particular, check that the versions of the various components and libraries match or are higher than those shown. Using this configuration will ensure that the CDA plugin will work properly without compatibility problems.




3) Installation and Starting

1. Installation

1. Download the CDA plugin ZIP file.
2. Open QGIS.
3. Go to the Plugins menu and select Manage and Install Plugins.
4. Click on the Install from ZIP tab.
5. Browse and select the downloaded ZIP file.
6. Click Install Plugin.

2. Running the Plugin

To run the CDA plugin, you will need to follow these steps:

1. **Open the Processing Toolbox:**
 - Go to the Processing menu.
 - Select Toolbox to open the Processing Toolbox panel.
2. **Locate the Plugin Script:**
 - In the Processing Toolbox, find and expand the **Scripts** section and click on it.
 - Click on “Open existing Script”.
 - Look for **CDA MAIN .py** python script.
3. **Run the Script:**
 - Double-click on the Coastal Dynamics Analyzer script to open it.
 - A script editor window will appear with the plugin's Python code.
4. **Execute the Script:**
 - Click on the green Run button () in the script editor toolbar.
 - This will execute the script and display the main GUI of the CDA plugin.



4) Input data

Overview

The CDA plugin is a tool used for coastal analysis that requires specific input data formats. The primary inputs needed are:

1. A linestring shapefile for the baseline.
2. Individual shapefiles for each shoreline to be analyzed.

Baselines and shorelines must be in different folders!

Baseline Shapefile

The baseline shapefile is a critical component that connects Ground Control Points (GCPs), ensuring the baseline passes through specific, crucial points. This baseline is used to construct the PCHIP (Piecewise Cubic Hermite Interpolating Polynomial) baseline, which is essential for subsequent analyses.

Requirements:

- **Linestring Shapefile:** This should connect the GCPs and ensure the baseline passes through key points.
- **Attribute Table:** The attribute table of the baseline shapefile must include a field named "length" containing the length value. This is particularly important if you want to skip Step 0 (creation of the baseline) and use a pre-existing baseline.

Shoreline Shapefiles

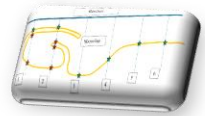
For shoreline analysis, each shoreline must be provided as a separate shapefile. The plugin does not support having all shorelines in a single shapefile.

Requirements:

- **Individual Shapefiles:** Each desired shoreline for analysis must be a separate shapefile. It is acceptable for these shorelines to be undissolved.
- **Attribute Table:** The attribute table for each shoreline shapefile can be empty, as the plugin will populate it with the necessary information during processing.

Important Notes

- **Baseline Creation (Step 0):** If you have a pre-existing baseline and want to skip Step 0, ensure that the baseline shapefile includes a "length" field in the attribute table with the corresponding length value.
- **Shoreline Intersections:** The plugin will search for intersections of transects with each segment of the shoreline, even if the shorelines are undissolved.



By following these guidelines, you can ensure that the CDA plugin works correctly and efficiently for your coastal analysis projects.

5) PLUGIN STARTING

- Main GUI of the CDA plugin. This is the window that is displayed by the user as soon as he starts the plug-in (Figure 2).

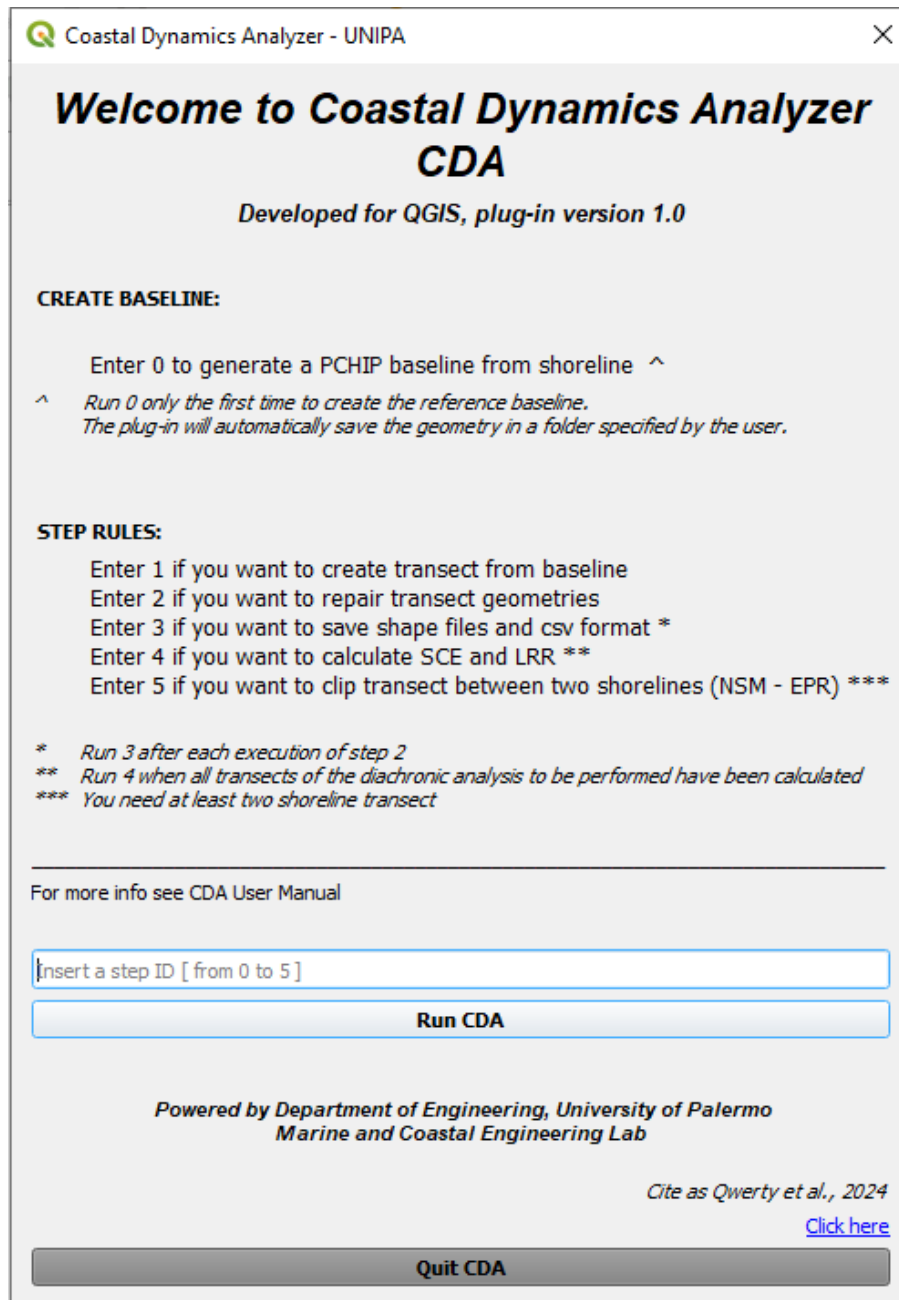
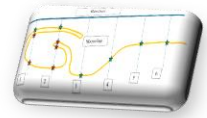


Figure 2. CDA main GUI.



This is the graphical user interface of the "Coastal Dynamics Analyzer (CDA)" plug-in developed for QGIS, version 1.0.0. The plug-in is designed for coastal dynamics analysis and is supported by the Department of Engineering, University of Palermo.

Below is an explanation of the main components and functionality of the interface:

Creation of the Baseline:

- **Enter 0 to generate a PCHIP baseline from shoreline:**
 - You use the "0" option to create the reference baseline from the coastline. This step is needed only the first time to create the reference baseline. The plug-in automatically saves the geometry in a user-specified folder.

Rules of Passages:

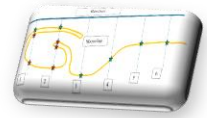
- **Enter 1 if you want to create transect from baseline:**
 - You enter "1" if you want to create transects from the baseline.
- **Enter 2 if you want to repair transect geometries:**
 - You enter "2" if you want to repair transect geometries.
- **Enter 3 if you want to save shape files and csv format:**
 - You enter "3" if you want to save shape files and csv format. You must perform this step after each execution of step 2.
- **Enter 4 if you want to calculate SCE and LRR:**
 - "4" is entered if the Shoreline Change Envelope (SCE) and Linear Regression Rate (LRR) are to be calculated. This step should be performed when all transects of the diachronic analysis to be performed have been calculated.
- **Enter 5 if you want to clip transect between two shorelines (NSM - EPR):**
 - "5" is entered if you want to trim the transect between two shorelines (Net Shoreline Movement (NSM) and End Point Rate (EPR)). At least two shoreline transects are required for this step.

Usage:

- **Insert a step ID [from 0 to 5]:**
 - You enter the ID of the desired step (0 to 5) in the text box.
- **Run CDA:**
 - After entering the passage ID, you click on "Run CDA" to execute the command.

Additional Information:

- **For more information, see the CDA user manual:**
 - More information can be obtained by referring to the CDA user manual.



Credits:

- The plug-in is developed by the University of Palermo Department of Engineering, Marine and Coastal Engineering Lab.
- Cite work such as Scala et al., 2024.

Control Buttons:

- **Quit CDA:**
 - Closes the plug-in interface.

This interface allows various operations for the analysis of coastal dynamics to be performed in a structured and sequential manner using QGIS.

6) Step 0 - Create Baseline

- **Description:** Enter "0" to generate a reference baseline (PCHIP) from the shoreline.
- **Usage:** This step is performed only the first time to create the reference baseline. The plugin will automatically save the geometry to a folder specified by the user.

Below are the 5 windows that CDA will automatically open once step 0 is selected and Run CD is clicked in the main GUI (Fig 1).

N.B. Once you have selected what is required from the first window click on "Select folder." The window will be closed, and the second window will open. Follow this progression until the fifth and final window where you will be asked to enter the distance between transects that you want to use for creating the baseline (this distance represents a kind of average resolution for constructing the PCHIP baseline) and the length of the transects.

In the upper left corner of all folder selection windows, it is indicated which folder is correct to select.

- Window to select the folder in which there is ONLY and EXCLUSIVELY the initial baseline in shapefile format (Figure 3). The initial baseline can simply be a shapefile in "linestring" format of joining control points or even the outline (left or right) of a shoreline buffer at a certain distance from the shorelines.

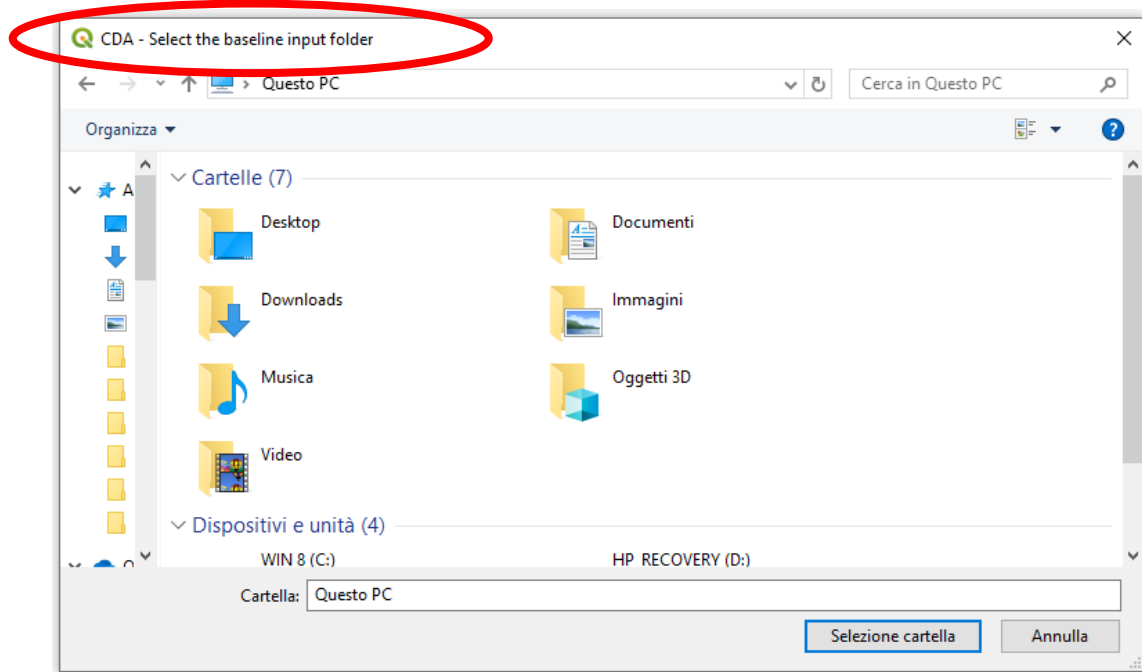
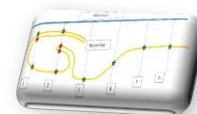


Figure 3. Window to select the folder in which there is ONLY and EXCLUSIVELY the initial baseline in shapefile format (Step 0).

- Window to select the folder where the shorelines are in shapefile format (Figure 4).

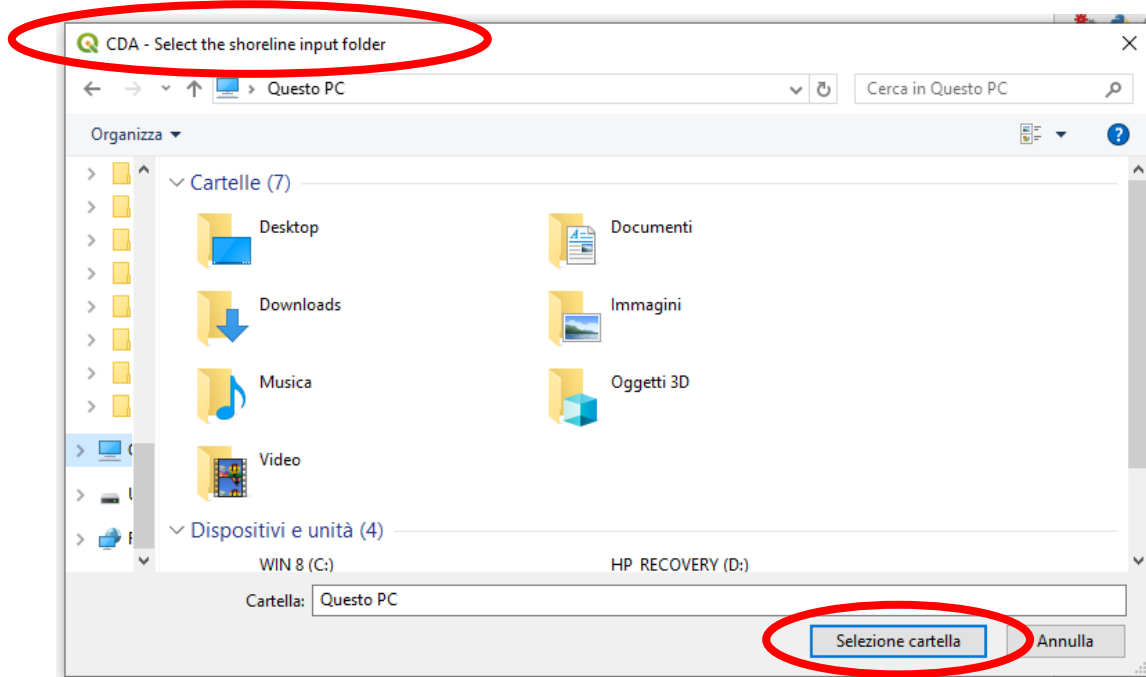
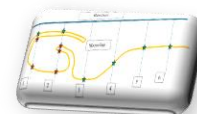


Figure 4. Window to select the folder where the shorelines are in shapefile format (Step 0).



- Window to select the folder in which to save the baseline in shapefile format (Figure 5).

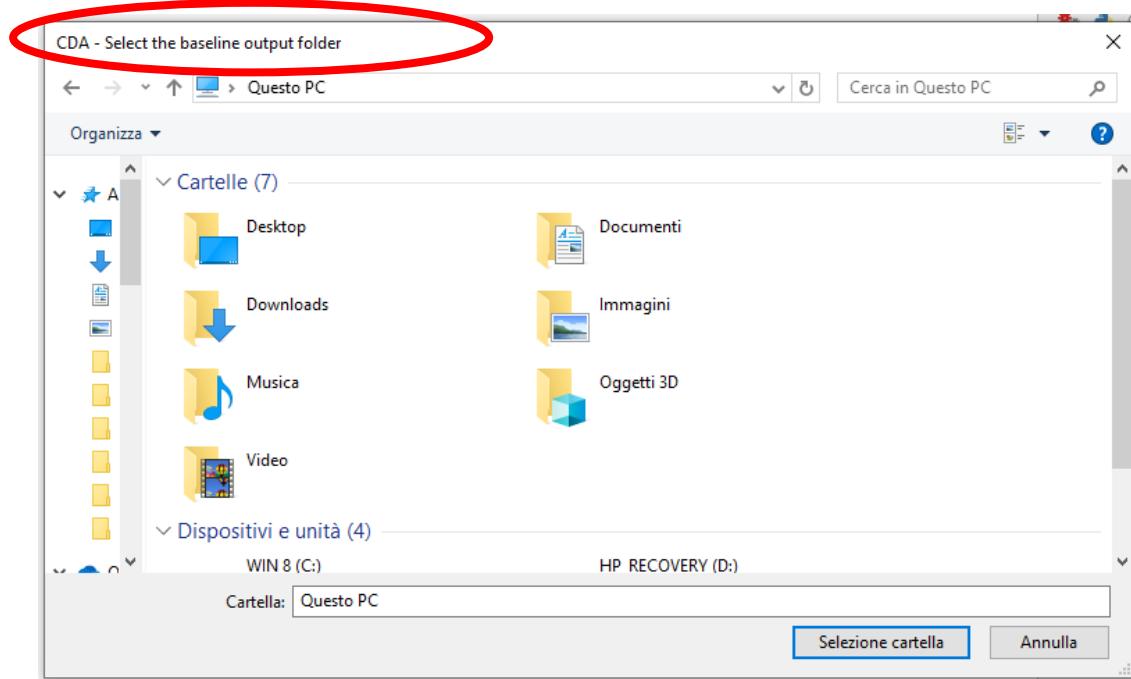


Figure 5. Window to select the folder in which to save the baseline in shapefile format (Step 0).

- Window to select a folder in which to save temporary processing files (TEMP) (Figure 6).

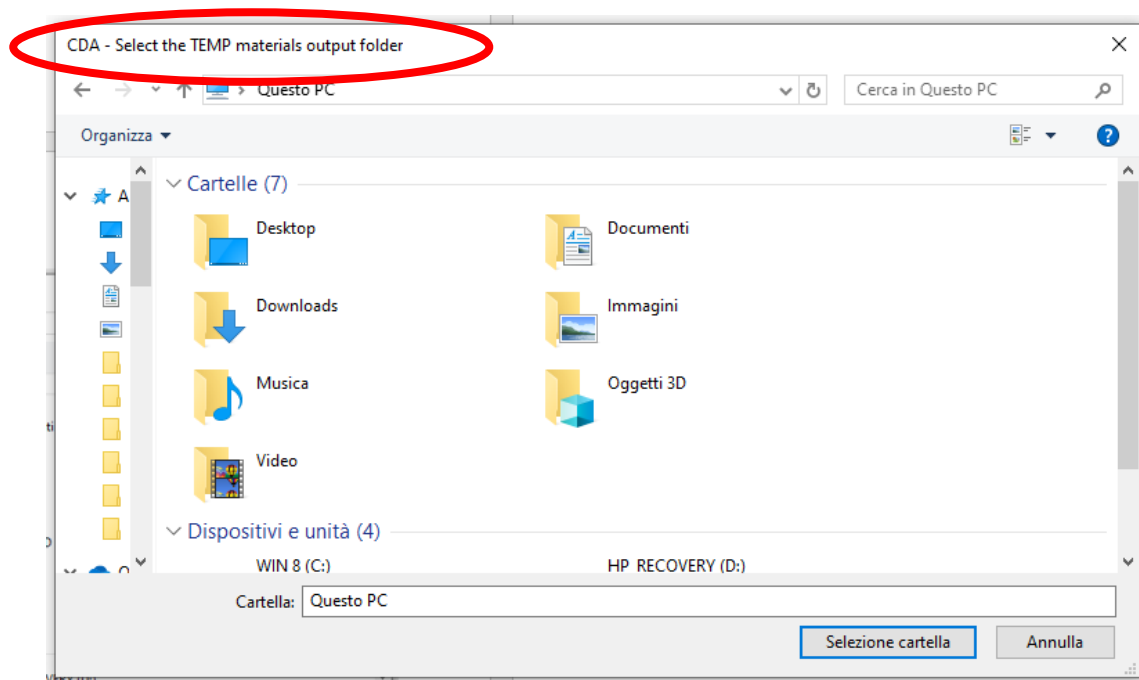
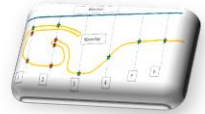


Figure 6. Window to select a folder in which to save temporary processing files (TEMP) (Step 0).



- Window to select values for 1) Distance between transects in meters, 2) Length of transects in meters (Figure 7). Once the numerical values are entered click on INPUT DONE and the processing algorithm will generate the PCHIP baseline.

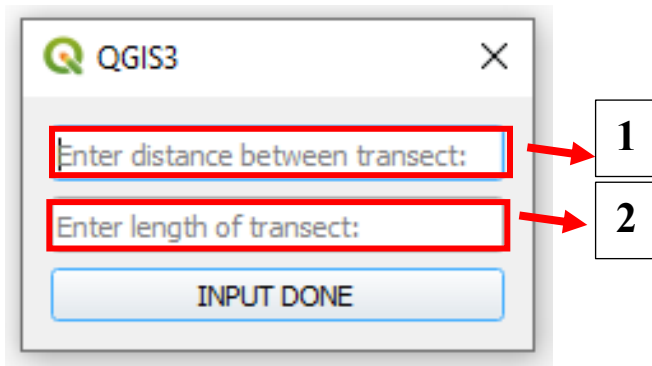


Figure 7. Window to select values for Distance between transects in meters and Length of transects in meters (Step 0).

7) Step 1 - Create Transect from Baseline

- **Description:** Enter "1" to create transects from the baseline using the *Calc_Trans* algorithm.
- **Usage:** Used to generate orthogonal transects from the baseline. These transects are needed for shoreline advancement or retreat analysis.

Below (Figure 8) is the window that CDA will automatically open once step 1 is selected and Run CD is clicked in the main GUI (Fig 2).

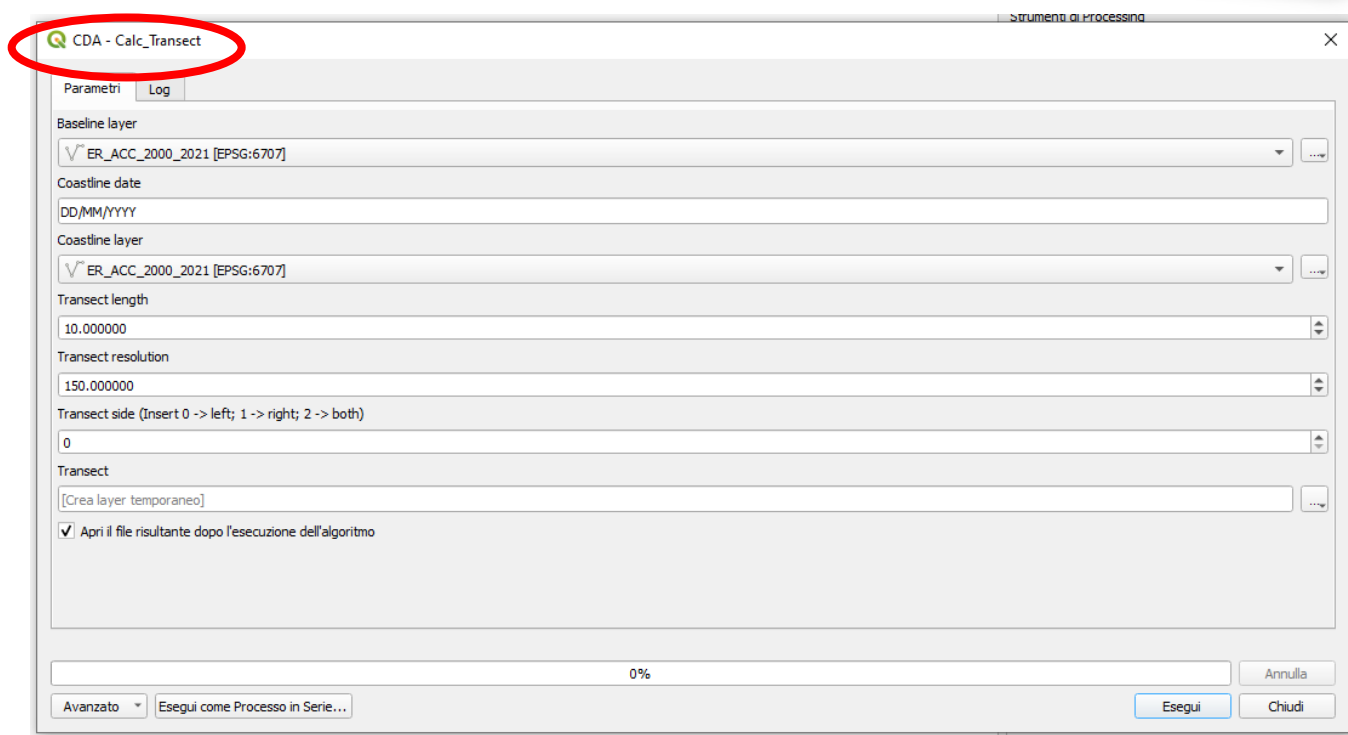
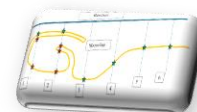
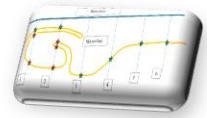


Figure 8. Step 1 window. Calc_Transect algorithm.

NB. In the window you will need to select, either individually or in batch (native QGIS functionality) by clicking on "Run as Batch Process."

- 1) The baseline shapefile layer (which can be the one created in step 0 or a ready-made baseline available to the user;
- 2) The date of acquisition of the coastline with the format indicated in the appropriate box to be filled in;
- 3) The shoreline layer in shapefile format for which to calculate and clip transects;
- 4) The length in meters of the transects (NB. Always enter a length greater than the maximum distance between the selected shoreline and the baseline);
- 5) Transect resolution (i.e., transect-transect interdistance);
- 6) The side of the baseline on which you want the transects to be generated:
 - 6.1) 0 = left
 - 6.2) 1 = right
 - 6.3) 2 = both

N.B. If you batch run the algorithm it will be possible to obtain for all shorelines under consideration transects from the same baseline while keeping the baseline constant and varying only the shorelines layer by layer. It is also possible to vary the resolution and length of transects on a case-by-case basis.



8) Step 2 - Repair Transect Geometries

- **Description:** Enter "2" to select transect geometries to be repaired.
- **Usage:** After transects are created, this step selects geometries to correct any geometric errors that could compromise the analysis.

Figure 9 shows the window that opens automatically when step 2 is selected and Run CD is clicked in the main GUI (Figure 2).

In the window select the transect layers that need adjustment. Again, the indication of what to select is shown in the upper left corner of the window.

It is advisable to perform step 2 only if within the shorelines to be analysed are shorelines folded back on themselves (parallel shorelines), or harbor arms, or piers, etc. See the example below for more information. If the shorelines to be analysed are small in extent and fairly linear (not very complex in shape) performing step 2 may be unnecessary.

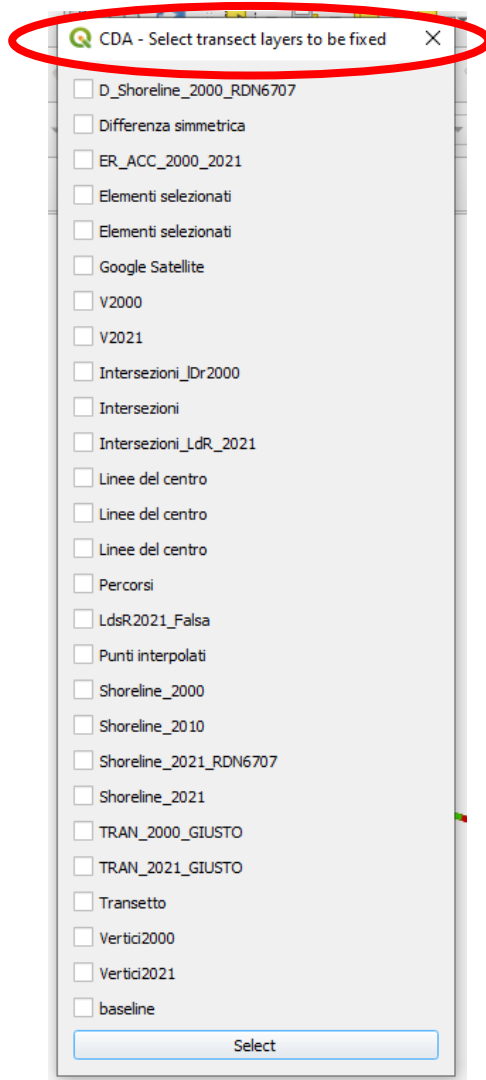


Figure 9. Step 2 window.

- Example of transect need to be fixed (TR.1 and TR.2) red stars represent wrong intersections (Figure 10).

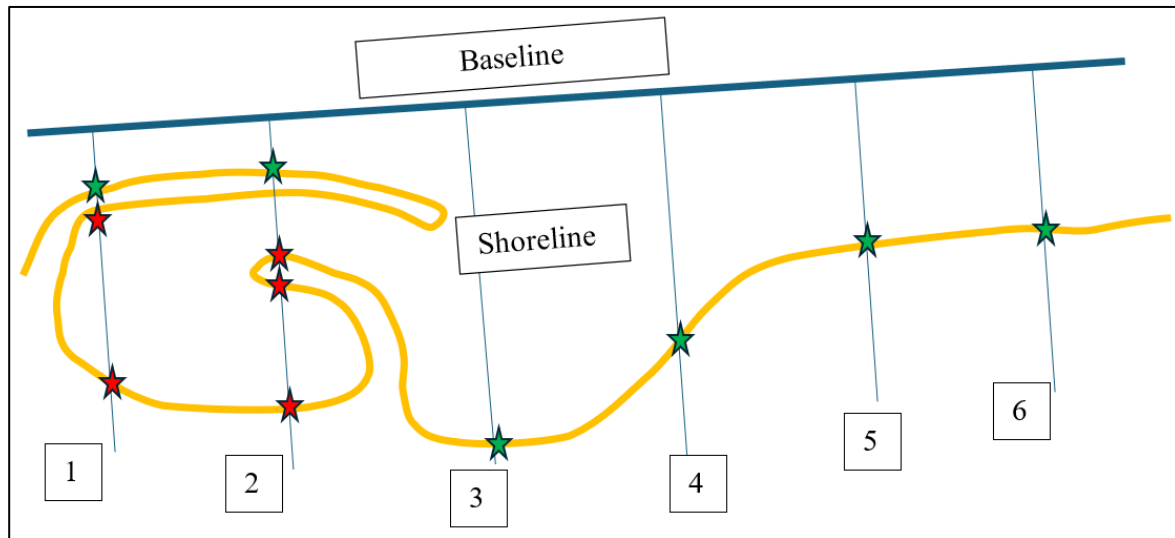
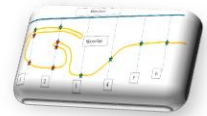


Figure 10. The diagram illustrates the process of selecting transects for repair in shoreline analysis. It features: 1) Baseline: The starting line for the transects and 2) Shoreline: The line where the transects intersect. Transects: Lines labeled 1 through 6 extending from the baseline to the shoreline. The Green Stars: Indicate correct transect-shoreline intersections while Red Stars: Indicate incorrect transect-shoreline intersections.

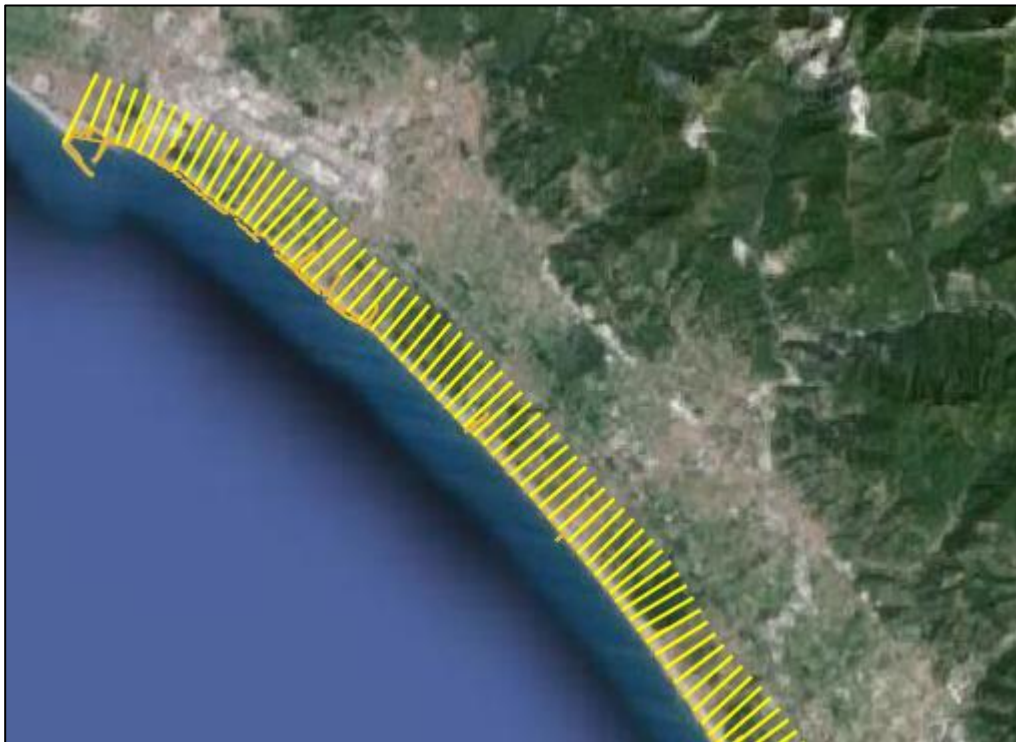
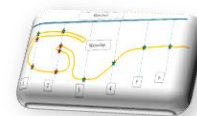


Figure 11. Example of step 2 result, selected transect.

In yellow (Figure 11) the repaired and selected transect.

N.B. Once you have run the algorithm the correct transects will be automatically selected. The next step will allow to save the shape file and .csv format the correct transects (selected in step 2).



9) Step 3 - Save Shape Files and CSV Format

- **Description:** Enter "3" to save shapefiles and CSV format.
- **Usage:** Save transects and associated data in formats that can be easily used and shared (.csv). It is recommended to perform this step after each transect repair (Step 2).

N.B. To speed up the process, it is recommended to run the algorithm serially for each layer processed in step 2. In each case, the correct transects will be selected for export and storage.

The user must select (either individually or in batch) the transect layer(s) to be repaired (saved) and select the filenames/destinations to be saved and written to memory.

Below (Figure 12) is the window that CDA will automatically open when step 3 is selected and Run CD is clicked in the main GUI (Fig 1).

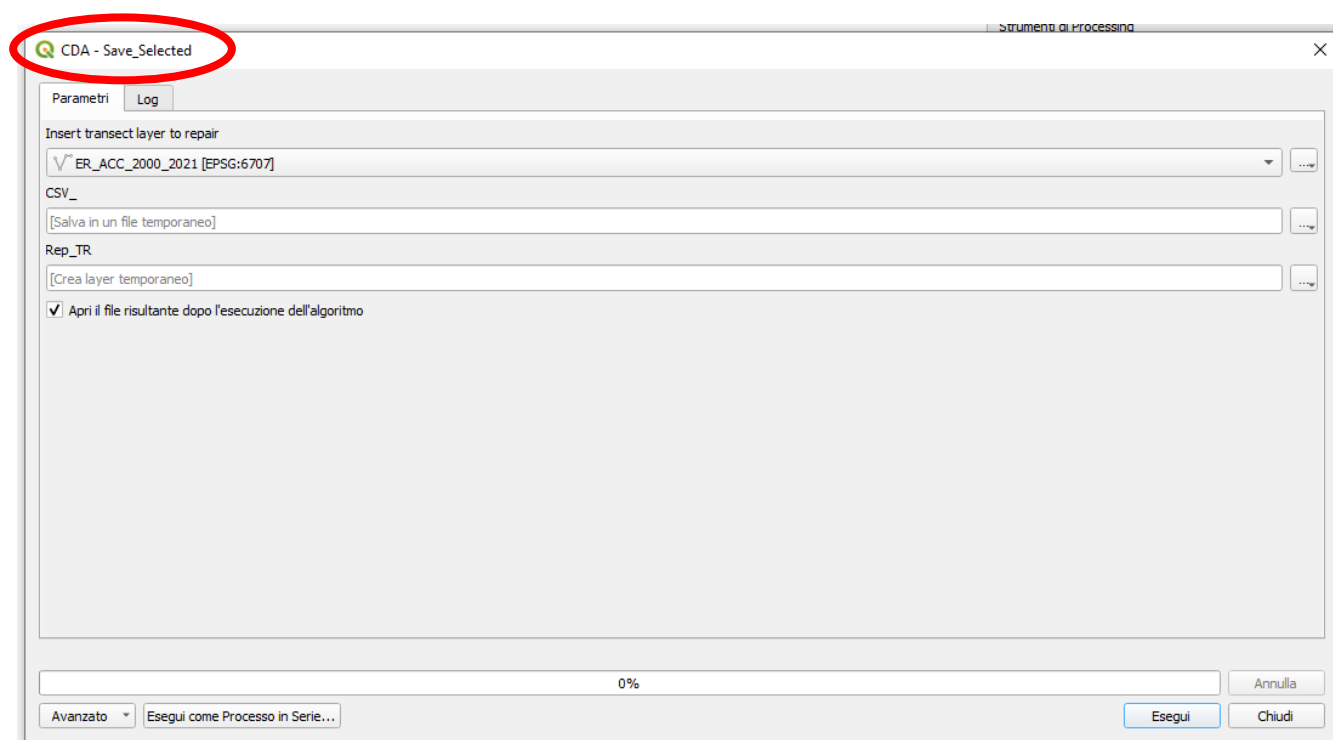
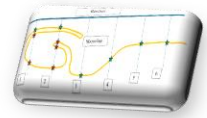


Figure 12. Step 3 CDA window. Function "Save_Selected".

10) Step 4 - Calculate SCE and LRR

- **Description:** Enter "4" to calculate SCE (Shoreline Change Envelope) and LRR (Linear Regression Rate).
- **Usage:** Calculates shoreline change metrics. This step is performed after all shoreline transects have been calculated and/or corrected.



In this case, the user only needs to select the folder where the csv files were saved. The algorithm will automatically calculate on the csv and shape files the SCE and LRR for each transect.

N.B. Again, the upper left corner shows what is required by the window displayed by the user.

IMPORTANT: Do not carry out step 4 until all the transects of the historical analysis to be carried out have been calculated.

Below (Figure 13) is the window that CDA will automatically open once step 4 is selected and Run CD is clicked in the main GUI (Fig 1).

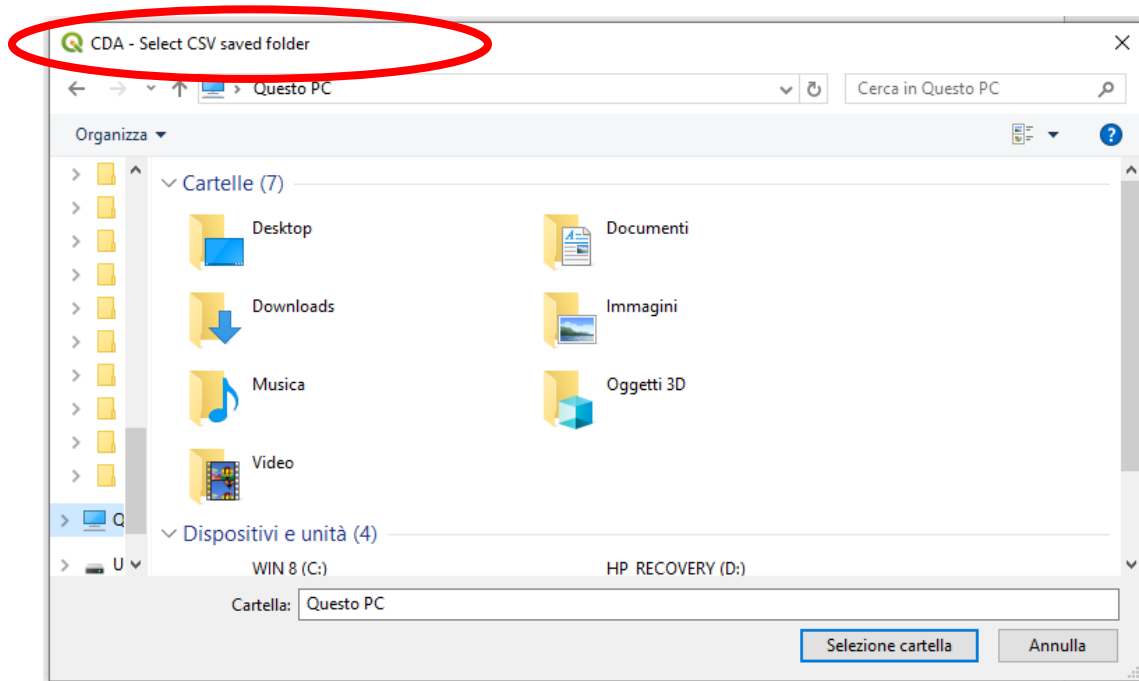
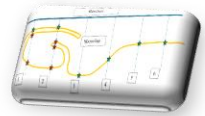


Figure 13. Step 4 CDA window.

11) Step 5 - Clip Transect between Two Shorelines (NSM - EPR)

- **Description:** Enter "5" to cut transects between two shorelines (NSM - Net Shoreline Movement and EPR - End Point Rate).
- **Usage:** This step requires at least two shoreline transects and allows comparison and analysis of the variation between two different shorelines.

Below (Figure 14) is the window that CDA will automatically open once step 5 is selected and Run CD is clicked in the main GUI (Fig 1).



N.B. The algorithm that will be started "CDA - TRANSECT_Clip" will require the input either a single time or in series of:

- 1) Year 1 transects.
- 2) The transects of the following year (year 2)
- 3) The intervening years between the two transects (thus the intervening years between the two reference shorelines).

N.B if transects refer to months or days difference then express the time-lag in months or days. The statistics output from the algorithm will have as units what the user enters (e.g., m/year - m/day - m/month etc.). It is important that the user chooses a unit of measurement and keeps it consistent for all analyses to ensure that the final data is read correctly. The unit of measurement output is therefore a function of the number entered by the user. To avoid any doubt, it is recommended that the variable always be entered in years and then converted once the results are obtained to the quantity desired.

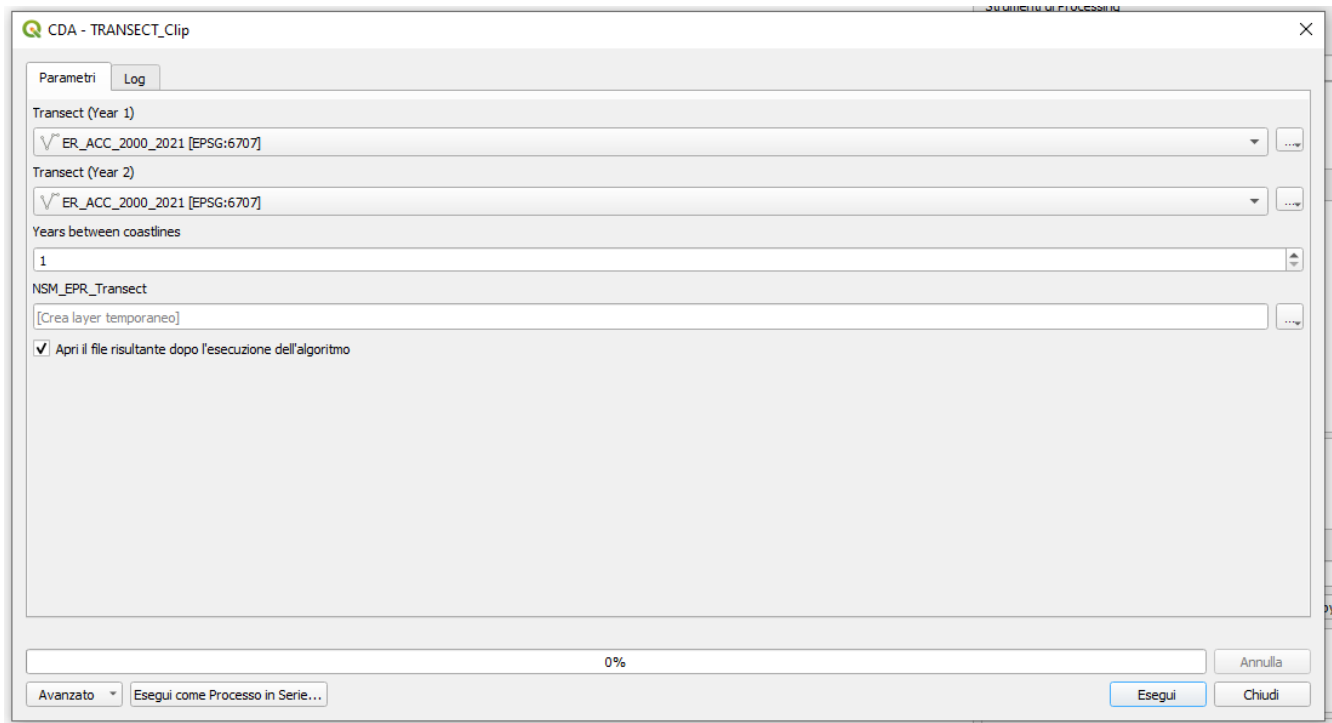


Figure 14. Step 5 CDA window. Algorithm TRANSECT_Clip.

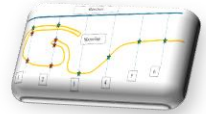
IN BATCH execution:

Example of serial execution.

- ☐ **Description:** Steps can be run in batch mode to automate the process.

Coastal Dynamics Analyzer - CDA

Powered by Department of Engineering, University of Palermo - Marine and Coastal Engineering Lab



□ **Usage:** Allows a series of operations to be performed sequentially without manual intervention for each step, useful for large-scale analysis or when many shorelines need to be processed. Batch execution can be performed for all steps that allow it (Figure 15).

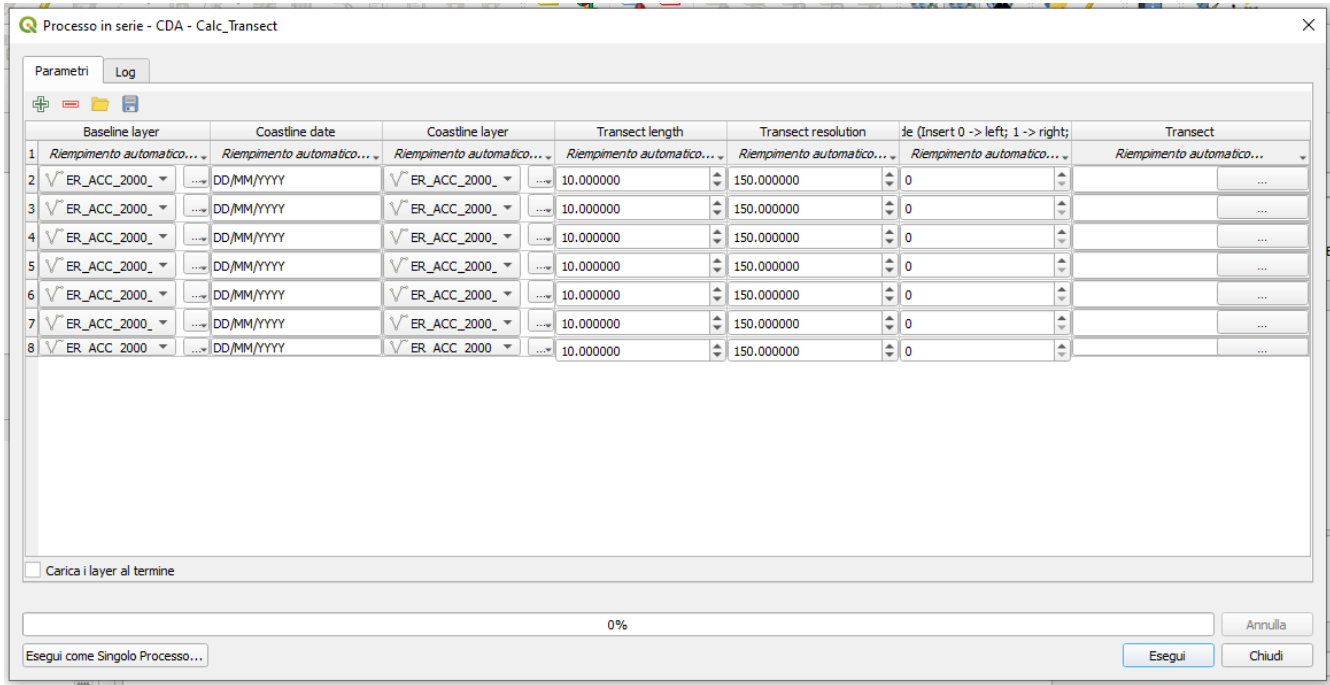
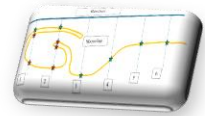


Figure 15. Example of batch process setup to execute step 5 CDA in series for 7 different shorelines.



12) PSEUDOCODE FRAMEWORK EXPLANATION

INCIPIT CODE.

Code analysis of the QGIS plug-in "Coastal Dynamics Analyzer - UNIPA"

This Python code creates a dialog box for the QGIS plug-in called "Coastal Dynamics Analyzer - UNIPA." The dialog guides the user through the steps of coastal dynamics analysis.

1. Imported libraries:

The code starts by importing Python libraries needed to create the GUI and interact with QGIS.

1 Imported libraries

- a. The code starts by importing Python libraries needed to create the GUI and interact with QGIS.

2 Class NumberInputDialog

- a. The NumberInputDialog class inherits from QDialog and defines the dialog box
- b. The `__init__` constructor configures the window:
 - a.1 Set the title of the window ("Coastal Dynamics Analyzer - UNIPA").
 - a.2 Create a vertical layout (QVBoxLayout) to arrange the interface elements.

Disabled comments (sections with #)

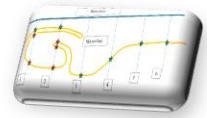
- a.3 The code comments section that adds a background image (presumably the QGIS icon). You can reactivate it if you want to display the icon.
- a.4 Create and configure text labels (QLabels) to display:
 - a.4.1 A welcome message with font formatting (bold, italics, size).
 - a.4.2 Plug-in version.

2. Class NumberInputDialog:

- The NumberInputDialog class inherits from QDialog and defines the dialog box.
- The `__init__` constructor configures the window:
 - Set the title of the window ("Coastal Dynamics Analyzer - UNIPA").
 - Create a vertical layout (QVBoxLayout) to arrange the interface elements.

Disabled comments (sections with #):

- The code comments section that adds a background image (presumably the QGIS icon). You can reactivate it if you want to display the icon.



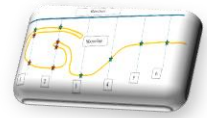
- Create and configure text labels (QLabels) to display:
 - A welcome message with font formatting (bold, italics, size).
 - Plug-in version.
 - Instructions for creating the baseline (with formatting to distinguish steps).
 - Instructions for performing other steps (numbered and with informative notes).
 - About the author (University of Palermo).
 - Plug-in reference citation.
 - Link to the University of Palermo website (made clickable by opening the link in the external browser).
- Adds a text field (QLineEdit) where the user will enter the identifying number of the step to be performed (1 to 5).
- Create and configure a button ("Run CDA") that activates the on_ok_clicked function when clicked.
- Sets the layout of the dialog box.
- Adds a red button ("Quit CDA") to close the window (using the reject method).

3. On_ok_clicked function:

- This function is called when the user clicks on "Run CDA."
- Attempts to convert user-entered text to an integer (int) number.
- If the number entered is valid (between 0 and 5):
 - Saves the value of the step in a global variable AA.
 - Closes the dialog box (self.accept).
- Otherwise, it displays an error message if the user does not enter a valid number.

4. Creating and displaying the dialog box:

- An instance of the NumberInputDialog class is created.
- The dialog box is displayed (dialog.exec_).
- If the user closes the window by clicking "Run CDA" (QDialog.Accepted):
 - The value of the selected step is printed (print("Insert Number:", AA)).
- Otherwise, if the user closes the window otherwise (print("Quit")):
 - A closing message is printed and the window is closed (dialog.close).



Summary:

This code creates an informative and interactive dialog box that guides the user through the steps of the coastal dynamics analysis. Depending on the step selected, presumably additional actions will be performed in your plug-in to process the data.

AA = 0

Analysis of the code for creating the baseline (AA==0)

This code deals with the creation of the baseline, which is the baseline for subsequent analysis of coastal dynamics. Let's look at it step by step:

1. Folder selection:

- Dialog boxes are used to allow the user to select folders:
 - infolder_baseline: Folder containing the shapefile of the initial coastline (baseline).
 - infolder_shoreline: Folder containing shapefiles of successive shorelines for diachronic analysis.
 - outfolder_baseline: Output folder to save the interpolated and dissolved baseline.
 - outfolder_transects: Temporary folder to store intermediate files (used later).
- The code checks whether the user has canceled the selection of a folder and terminates execution if necessary.

2. Interpolation parameters:

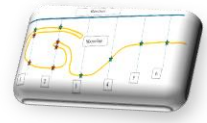
- A dialog box is created with two text fields for the user to enter:
 - dist_trans: Distance between transects.
 - length_trans: Length of transects (presumably used later).

3. Initial baseline reading:

- The code iterates through files with the .shp extension in the infolder_baseline folder and assumes that there is a single shapefile containing the initial baseline.
- Recover the geometry of the initial coastline.

4. Baseline interpolation:

- The total length of the initial baseline is calculated.
- The desired number of equidistant points on the interpolated baseline (sampling) is defined.
- The vertices of the initial geometry are extracted and duplicate points on the X axis are removed to avoid interpolation errors.



- The remaining coordinates are sorted according to the X axis.
- A spline of type CubicSpline is used to interpolate the baseline and generate new equidistant points along the coastline.

5. Creation of the new baseline layer:

- A new vector layer is created in memory called baseline of type LineString with the reference system of the initial baseline.
- Three fields are defined for the layer:
 - ID: Unique numerical identifier for each feature (point).
 - X: X coordinate of the point.
 - Y: Y coordinate of the point.
- You add the fields to the layer and update the properties.
- An empty feature list is created.

6. Feature creation:

- The code iterates over the interpolated points and for each:
 - Create a new feature.
 - Set feature attributes with ID, X and Y coordinates.
 - Sets the feature geometry as a line composed of all interpolated points.
 - Adds the feature to the list.

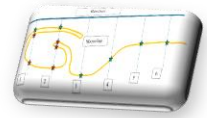
7. Baseline dissolution:

- You dissolve the baseline layer using the JOIN field set to 1 for all features (this step may require additional information to understand its purpose).
- The result of the dissolution is saved in a temporary memory layer called dissolved_baseline.

8. Writing shapefiles:

- You remove any escape characters from the output folder path.
- You save the geometry of the dissolved layer (dissolved_baseline) as a shapefile in outfolder_baseline/dissolved_baseline.shp.
- You save the interpolated baseline layer as a shapefile in outfolder_baseline/baseline.shp.
- The projection files (baseline.prj) are copied from the infolder_baseline folder to the outfolder_baseline folder for both saved shapefiles.

9. Loading the result into the QGIS TOC:



- The instance of the current QGIS project is obtained.
- A vector layer is created from the interpolated baseline shapefile (output_shapefile).
- You add the layer you just created to the TOC panel of QGIS.

This code automates the process of interpolating the initial baseline and creating the shapefiles needed for subsequent analysis.

AA = 1

This code defines a processing algorithm in QGIS called "CDA - Calc_Transect" (Calc_Transect stands for Calculating Transect). The algorithm performs several steps to calculate transects from a baseline and coastline:

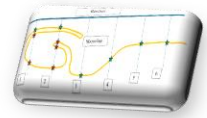
1. Input parameters:

- baseline_layer: Vector layer of the interpolated baseline (created earlier with AA==0).
- coastline_date: Date associated with the coastline used (entered as a string by the user).
- coastline_layer: Coastline vector layer for transect analysis.
- transect_length: Length of transects in meters (user-defined).
- transect_resolution: Transect resolution in meters (distance between points along the baseline to generate transects).
- transect_side_insert_0__left_1__right_2__both: Side of transect insertion (left, right or both).

2. Process:

The algorithm uses multi-step feedback to show the progress of each step:

1. **Points along the geometry:** Equidistant points along the baseline are generated using QGIS's native:pointsalonglines algorithm. The distance between points is defined by the transect_resolution parameter.
2. **Points to path:** Previously generated points are merged to form separate lines in the output of QGIS's native:pointstopath algorithm.
3. **Transect:** Transect lines are created perpendicular to the baseline with the length specified in transect_length using QGIS's native:transect algorithm. The side parameter defines the side of transect insertion with respect to the baseline.
4. **Line intersection:** Intersections between transects and the coastline are computed using QGIS's native:lineintersections algorithm.
5. **Intersection of path lines:** Intersections between transects and lines generated by points on the baseline (presumably used to calculate transect lengths later) are calculated.



6. **Connects by lines (hublines):** Intersections between transects and the coastline are connected with lines using QGIS's native:hublines algorithm.
7. **Fieldcalculator:** A field named "length" is added to the resulting shapefile containing the length of each transect calculated with QGIS's native:fieldcalculator algorithm using the \$length formula.
8. **Years_fieldcalculator:** A field named "date" is added to the final shapefile containing the date associated with the coastline specified by the user with QGIS's native:fieldcalculator algorithm.

3. Output:

The algorithm returns a vector layer containing transects with length and date attributes of the coastline used.

This code makes it possible to automate the process of calculating transects from the baseline and shoreline by integrating several functions of QGIS processing tools.

AA = 2

Code analysis for transect selection and correction (AA==2)

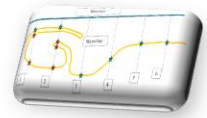
This code deals with the selection and correction of transect layers in QGIS. Let's look at it step by step:

1. LayerSelectionDialog class:

- A LayerSelectionDialog class is defined that inherits from QDialog to create a custom dialog box.
- The dialog box is titled "CDA - Select transect layers to be fixed."
- A vertical layout (QVBoxLayout) is created to arrange the window elements.
- An empty self.layer_check_boxes list is created to store layer checkboxes.
- You call the populate_layer_check_boxes function to populate the layout with checkboxes.
- A loop is created to add each checkbox to the layout.
- You create a "Select" button (select_button) and link its click to the on_select_button_clicked event.
- You add the button to the layout.
- You initialize an empty list self.selected_layer_names to store the names of the selected layers.
- You set the layout as the main layout of the dialog box and define the initial dimensions.

2. Populate_layer_check_boxes function:

- Retrieve all layers in the QGIS TOC using QgsProject.instance().mapLayers().values().
- Iterate on each layer and create a checkbox (QCheckBox) setting the text with the layer name.



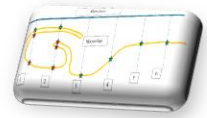
- Adds the checkbox to the self.layer_check_boxes list.

3. On_select_button_clicked function:

- It is called when the user clicks on the "Select" button.
- Cleans up the list self.selected_layer_names.
- Iterate on the check boxes and check if they are selected.
- If a box is selected, it adds its text (layer name) to the self.selected_layer_names list.
- Print the names of the selected layers at the console.
- Closes the dialog box.

4. Layer selection and correction:

- The LayerSelectionDialog is created and displayed.
- After the user has made the selection and closed the window, the list of selected layer names (selected_layer_names) is obtained.
- A loop is run on the selected layer names:
 - We retrieve the reference to the selected layer from the TOC using `QgsProject.instance().mapLayersByName(layer_name)[0]`.
 - The names of the fields used are defined:
 - field_tr_id: Name of the unique identifier field for transects (presumably "TR_ID").
 - field_length: Name of the field containing the length of the transect.
 - An empty min_length_dict dictionary is created to store the minimum length for each value of field_tr_id.
 - You iterate over the features on the selected layer:
 - The value of the field_tr_id field for the current feature (tr_id) is retrieved.
 - The value of the field_length field for the current feature (length) is retrieved.
 - It checks whether the value of tr_id already exists in the min_length_dict dictionary.
 - If it does not exist, tr_id is added to the dictionary with the current length (length).
 - If it exists and the current length is less than the length already in the dictionary, it updates the value in the dictionary with the lesser length.
 - An empty selection list is created to store the IDs of selected features.



- It iterates over the `min_length_dict` dictionary:
 - We extract the value of `tr_id` (`tr_id`) and the minimum length (`min_length`) for each element of the dictionary.
 - A `QgsExpression` is created to select features with a value of `field_tr_id` equal to `tr_id` and a value of `field_length` equal to `min_length`.
 - You retrieve the features that satisfy the expression using `selected_layer.getFeatures(QgsFeatureRequest(exp))`.
 - You extract a list of IDs from these features and add them to the selection list

AA = 3

Code analysis for saving selected transects (AA==3)

This code defines a processing algorithm in QGIS called "CDA - Save_Selected". The algorithm allows selected transects from a layer to be exported to a CSV file.

1. Input parameters:

- `insert_transect_layer_to_repair`: Vector layer of transects from which to export selected features.
- `Csv_`: Path and name of output file in CSV format (user-defined).

2. Process:

The algorithm uses multi-step feedback to show the progress of each step:

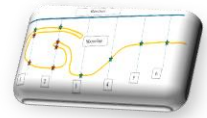
1. **Extract selected features:** Selected features are extracted from the input layer using QGIS's `native:savesselectedfeatures` algorithm. The output is a temporary layer containing only the selected features.
2. **Save vector features to file:** Selected features extracted earlier are saved as CSV files using QGIS's `native:savefeatures` algorithm. The `ACTION_ON_EXISTING_FILE` parameter is set to 0 to create or overwrite an existing file.

3. Output:

The algorithm returns the path to the CSV file containing the selected transects.

This code makes it easy to export user-selected transects of interest into a format readable by other software (CSV) for further analysis.

AA = 4



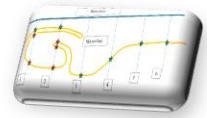
This code analyzes CSV files containing previously saved transect data (presumably with the code analysed above for AA==3). The analysis focuses on calculating the rate of change (Slope Coast Evolution - SCE) and other values for each transect ID (TR_ID).

1. Folder selection and CSV file import:

- You use the PyQt5.QtWidgets library to open a dialog box that allows the user to select a folder.
- It is verified that the user has selected a folder, otherwise the code terminates.
- An all_info dictionary is created to store information from all CSV files.
- An all_tr_ids set is created to store all unique values found for the "TR_ID" column in CSV files.
- It iterates over all files with a ".csv" extension in the selected folder:
 - It opens the CSV file with "latin-1" encoding.
 - You use csv.DictReader to read the CSV file as a dictionary of rows.
 - It iterates over each line of the CSV file:
 - You extract the value of column "TR_ID" and store it in tr_id.
 - You extract the value of the "date" column and convert it to a datetime object.
 - You convert the datetime object to a representative integer (ordinal) in order to perform subsequent calculations.
 - It extracts the value of the "length" column and converts it to a numeric value of type float.
 - You store the extracted information (date converted to ordinal number and length) for the current tr_id in the all_info dictionary.
 - You add the current tr_id to the set all_tr_ids.

2. Calculation of linear regression for each TR_ID:

- A lrr_values dictionary is created to store the angular coefficients of the linear regression for each tr_id.
- It iterates over all unique tr_ids:
 - Two empty lists x and y are created to store the date (converted to ordinal numbers) and length values for the current tr_id.
 - It iterates over each CSV file and its information:
 - If the current tr_id is present in the CSV file, the date value (ordinal number) and length are added to the corresponding list (x and y).
 - You convert the x and y lists to NumPy arrays in order to perform matrix calculations.



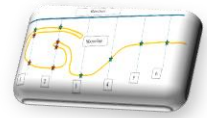
- The angular coefficient of the linear regression is calculated using the formula num / den where:
 - num is the sum of the product of the difference between each date value and the mean and the difference between each length value and the mean.
 - den is the sum of the square of the difference between each date value and the mean.
- You store the calculated angular coefficient (num / den) in the `lrr_values` dictionary for the current `tr_id`.

3. Calculation of the maximum length value for each row:

- A `max_length_per_row` dictionary is created to store the maximum length value found for each `tr_id` in all CSV files.
- A `max_length_date_per_row` dictionary is created to store the date associated with the maximum length value for each `tr_id`.
- It iterates over all unique `tr_ids`:
 - We use the `max` function with a list comprehension expression to find the tuple (ordinal date, length) with the maximum length value for the current `tr_id` in all CSV files.
 - You extract the maximum length value (`max_length_info[0]`) and store it in the `max_length_per_row` dictionary for the current `tr_id`.
 - We extract the date associated with the maximum length value (`max_length_info[1]`) and store it in the `max_length_date_per_row` dictionary for the current `tr_id`.

4. Writing the information into a single CSV file:

- A CSV file named "info_all.csv" is created in the selected folder.
- You define the column headers of the CSV file:
 - "ID": Contains the value of `TR_ID`.
 - For each CSV file in the folder, two columns are added:
 - "{filename}_Date": Contains the date (formatted DD/MM/YYYYYY) for the current `TR_ID` in the CSV file.
 - "{filename}_Length": Contains the length value for the current `TR_ID` in the CSV file.
 - The columns "SCE," "date_SCE," and "LRR" are added.
- It opens the CSV file for writing with "utf-8" encoding.
- You create a `csv.DictWriter` object to write the rows in the CSV file.



- You write the header of the CSV file.
- It iterates over all ordered TR_IDs:
 - A row_data dictionary is created to store the information for the current row.
 - Date and length information for the current TR_ID from each CSV file is added to the row_data dictionary.
 - You convert ordinal dates to datetime objects and format them in DD/MM/YYYY format.
 - The value of SCE (maximum length value) for the current TR_ID is calculated and added to the row_data dictionary.
 - It calculates and adds the date associated with the maximum length value (date_SCE) for the current TR_ID to the row_data dictionary.
 - You check if the linear regression angular coefficient for the current TR_ID is available and add it to the row_data dictionary.
 - You write the row into the CSV file using the row_data dictionary.
- You close the CSV file.
- A warning message is printed to indicate that the operation has been completed and the CSV file has been created.

Code summary:

This code allows the analysis of CSV files containing previously saved transect data. The analysis focuses on calculating the rate of change (Slope Coast Evolution - SCE) and (Linear Regression Rate - LRR) for each transect ID (TR_ID). The code generates a CSV file with unified information on all transects, including the rate of change, the date associated with the maximum length value, and the linear regression angular coefficient.

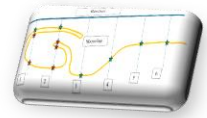
Additional Notes:

- The code assumes that the CSV files contain the columns "TR_ID," "date," and "length."
- The date format in the "date" column can be changed to suit your needs.
- The code can be adapted to include additional calculations or analysis on transect data.

AA = 5

Code analysis for transect clipping (AA==5)

This code defines a processing algorithm in QGIS called "CDA - TRANSECT_Clip". The algorithm allows to calculate Erosion Rate Potential (ERP) by analyzing two layers of transects related to different years.



1. Input parameters:

- `transect_year_1`: Vector layer containing the transects of the first year.
- `transect_year_2`: Vector layer containing the transects of the second year.
- `years_between_coastlines`: Positive integer number representing the years between the two transect surveys.
- `Nsm_epr_transect`: Output of the algorithm (vector layer containing transects with new NSM and EPR fields).

2. Process:

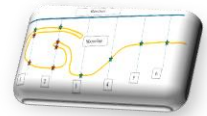
The algorithm uses multi-step feedback to show the progress of each step:

1. **Extract vertices A1 and A2:** The initial vertices (ends) of each transect are extracted from the two input layers using QGIS's native:extractvertices algorithm. The output is two temporary layers containing only the initial vertices.
2. **Symmetric difference:** Symmetric difference is performed between the initial vertices of the two layers using QGIS's native:symmetricaldifference algorithm. The symmetric difference identifies areas that are in only one of the two vertex sets. The output is a temporary layer containing the geometries resulting from the difference.
3. **Intersection Y1 and Y2:** The intersection between the symmetric difference layer and each original transect layer is performed using QGIS's native:intersection algorithm. The output is two temporary layers containing the geometries resulting from the intersection.
4. **Connects via lines (hub lines):** Intersection geometries are connected by lines using QGIS's native:hublines algorithm. The algorithm uses intersection vertices as "hubs" and creates lines that connect them. The output is a temporary layer containing the connecting lines.
5. **Field Calculator (NSM):** A new field called "NSM" is created in the link line layer using QGIS's native:fieldcalculator algorithm. The NSM field is calculated as the difference between the length of a line and the length of its intersection with the first-year transect.
6. **Field Calculator (EPR):** Another new field called "EPR" is created in the link lines layer again using QGIS's native:fieldcalculator algorithm. The EPR field is calculated by dividing the value of NSM by the number of years elapsed between transect surveys (specified by the `years_between_coastlines` parameter).

3. Output:

The algorithm returns the final link line layer containing the "NSM" (Erosion Rate Potential) and "EPR" (Erosion Rate) fields.

This code allows the EPR to be calculated by analyzing the displacement of the coastline between two surveys taken in different years. The value of NSM represents the potential linear erosion, while EPR represents the erosion rate per year.



13) Reference

Giorgio Manno, Carlo Lo Re, Mirko Basile, Giuseppe Ciralo, A new shoreline change assessment approach for erosion management strategies, Ocean & Coastal Management, Volume 225, 2022, 106226, ISSN 0964-5691, <https://doi.org/10.1016/j.ocecoaman.2022.106226>.
(<https://www.sciencedirect.com/science/article/pii/S0964569122002022>)

Please when cite CDA plugin, cite both methodology work (Manno et al., 2022) and SoftwareX plugin paper (Scala et al., 2024).

14) Contacts

For more information about this report, contact:

Pietro Scala^a, pietro.scala@unipa.it

Giorgio Manno^a, giorgio.manno@unipa.it

^a Department of Engineering, University of Palermo (UNIPA) Sicily, Italy.