
ThermoLiDAR Tutorial and Documentation Documentation

Release 0.0.1

Roberto Antolin

September 15, 2014

1	The software	3
1.1	The plugin	3
2	Installation	5
2.1	Requirements	5
2.2	Installing and Configuring ThermoLiDAR software	7
3	Tutorial	13
3.1	LiDAR Processing	13
4	User Manual	35
4.1	Introduction	35
4.2	Thermal Processing	36
4.3	LiDAR Processing	40
4.4	Statistics	67
4.5	Forest Health Assessment	69
4.6	Data Analysis	71
5	Examples	87
5.1	Case Study 1 - Huelva (Spain)	87
5.2	Case Study 2 - Almería (Spain)	116
6	Indices and tables	135
	Bibliography	137

Summary

Release 0.0.1

Date August 13, 2014

Authors Roberto Antolin; FJ Romero,

Target users

status some mature, some in progress

The software

ThermoLiDAR software is designed to provide the required tools to integrate Thermal and LiDAR information for the assessment of forest health and production. The software includes different modules and several settings offering different levels of processing to suit each needs. The main functionalities of the software are:

1. Tools for raw LiDAR and Thermal data processing, data exploring, quality assessment and visualisation.
2. Tools to integrate both data sources using suitable data fusion techniques.
3. Spatial statistical tools to analyse biophysical variables of the vegetation based on the integration of LiDAR and Thermal data integration.
4. Tools for mapping forest health condition and forest production dynamics and producing accuracy assessment.

1.1 The plugin

ThermoLiDAR has been developed as a QGIS plugin. It has been tested successfully with QGIS 1.8.0-Lisboa and it is now been updated to QGIS 2.0. In order to use ThermoLiDAR, it is necessary to [download](#) and install QGIS 1.8.0-Lisboa or QGIS 2.0.0-Dufour <<http://www.qgis>.

The plug-in incorporates tools for LiDAR data management and processing including a graphical user interface for the SPDLib tools.

Installation

2.1 Requirements

2.1.1 Hardware Requirements

Minimum hardware requirements implies

- x86–64 compatible CPU
- 1 GB RAM
- 20 GB available hard disk space

For a fully operational system, the recommended hardware requirements are:

- x86–64 or compatible CPU
- 4 GB RAM
- 50 GB available hard disk space

2.1.2 Software Requirements

THERMOLIDAR software has been developed as a QGIS plug-in so the final user gets access to lots of GIS capabilities freely. This implies QGIS and its Processing plug-in to be installed. The oldest QGIS version supported by the THERMOLIDAR plug-in is **QGIS 2.0.1 Dufour**, and the oldest version for Processing plug-in is **2.0**.

For the THERMOLIDAR software to run, the plug-in requires some external libraries that need to be installed previously:

- R
- SPDLib
- RSGISLib

R is a free programming language and software environment for statistical computing, **SPDLib** is a set of open source software tools for processing laser scanning data, and **RSGISLib** is a collection of tools for processing remote sensing and GIS data. **R** is used by the Data Analysis modules, while **SPDLib** and **RSGISLib** are used by the Data Processing modules.

Installing SPDlib

Warning: This has to be improve and add some Windows info

The notes below hopefully provide some useful details on the process for installing SPDLib. These notes are intended for people compiling the software on a UNIX platform such as Mac OSX, Linux or Solaris (these are the platforms on which the software has been tested).

To compile the software (and the pre-requisites) you will need a C++ compiler, we use the [GNU GCC](#) compilers but the software has also been tested and compiles without a problem using the SunPro compiler on Solaris and the Intel x86 compilers.

You will also need to have [mercurial](#) installed to download the latest version of the SPDLib source code, [cmake](#) to configure the source code before compilation.

Getting the SPDlib Source Code

The SPDLib source code is hosted within a Mercurial repository on [bitbucket](#). To clone the source code into a folder `spdlib` run the following command:

```
hg clone https://bitbucket.org/petebunting/spdlib spdlib
```

Compiling SPDlib

If libraries are not installed within `/usr/local` then the path needs to be specified using the variables available on CMake listed below.

```
$ cmake -D CMAKE_INSTALL_PREFIX=/usr/local \  
-D HDF5_INCLUDE_DIR=/usr/local/include \  
-D HDF5_LIB_PATH=/usr/local/lib \  
-D LIBLAS_INCLUDE_DIR=/usr/local/include \  
-D LIBLAS_LIB_PATH=/usr/local/lib \  
-D GSL_INCLUDE_DIR=/usr/local/include \  
-D GSL_LIB_PATH=/usr/local/lib \  
-D CGAL_INCLUDE_DIR=/usr/local/include \  
-D CGAL_LIB_PATH=/usr/local/lib \  
-D BOOST_INCLUDE_DIR=/usr/local/include \  
-D BOOST_LIB_PATH=/usr/local/lib \  
-D GDAL_INCLUDE_DIR=/usr/local/include \  
-D GDAL_LIB_PATH=/usr/local/lib \  
-D XERCESC_INCLUDE_DIR=/usr/local/include \  
-D XERCESC_LIB_PATH=/usr/local/lib \  
-D GMP_INCLUDE_DIR=/usr/local/include \  
-D GMP_LIB_PATH=/usr/local/lib \  
-D MPFR_INCLUDE_DIR=/usr/local/include \  
-D MPFR_LIB_PATH=/usr/local/lib \  
-D CMAKE_VERBOSE_MAKEFILE=ON \  
  
$ make  
$ make install
```

Pre-requisites

The SPDLib software library has a number of software prerequisites, which are required to build the software.

During the development process, to date, the following libraries have been included:

1. Boost (<http://www.boost.org>) (oldest Version 1.49)
2. HDF5 (<http://www.hdfgroup.org>) (oldest Version 1.8.2)
3. GNU Scientific Library (GSL; <http://www.gnu.org/software/gsl>) (oldest Version 1.14)
4. Xerces-C (<http://xerces.apache.org/xerces-c>) (oldest Version 3.1.1)
5. GDAL/OGR (<http://www.gdal.org>) (oldest Version 1.7)
6. LibLAS (<http://www.liblas.org>) (oldest Version 1.6)
7. CGAL (<http://www.cgal.org>) (oldest Version 3.8)

2.2 Installing and Configuring ThermoLiDAR software

2.2.1 Installation

The source code of the ThermoLiDAR software development is located in a private [bitbucket](#) repository. The easiest to get ThermoLiDAR plug-in is to download it from the official repository (<https://bitbucket.org/thermolidar/thermolidar/downloads>). Once the file is downloaded just unzip it into your system QGIS plug-ins folder. This plug-ins folder should be located in:

```
%HOMEDRIVE%\%HOMEPATH%\qgis2\python\plugins
```

in Windows systems and, for Unix-based systems:

```
~/qgis2/python/plugins
```

Finally, the unzip folder has to be renamed to *thermolidar*.

Alternatively, the software can be directly clone from the repository to our local QGIS plug-in folder using `git`:

```
$ git clone http://bitbucket.org/thermolidar/thermolidar.git ~/qgis2/python/plugins/thermolidar
```

2.2.2 Configuration

Start QGIS and make sure the *Processing Toolbox* is enable and the *Advanced interface* is selected (at the bottom). The Processing Toolbox should look like this:

Manage and Install Plugins is placed into the *Plugins* menu in QGIS

Now the plugin must be enabled in the Installed tab within the *Plugins > Manager and Install Plugin* menu.

From the *Manage and Install Plugins...* within the *Plugins* menu, select the **ThermoLiDAR** plug-in:

Once QGIS load ThermoLiDAR, it appears within the Processing Toolbox

However, to use the plug-in SPDLib and R software have to be enabled and visible from QGIS. Otherwise the user will get an error message that denies running any tool:

Warning: SPDTools folder is not configured. Please, consider to configure it before running SPDTools algorithms.

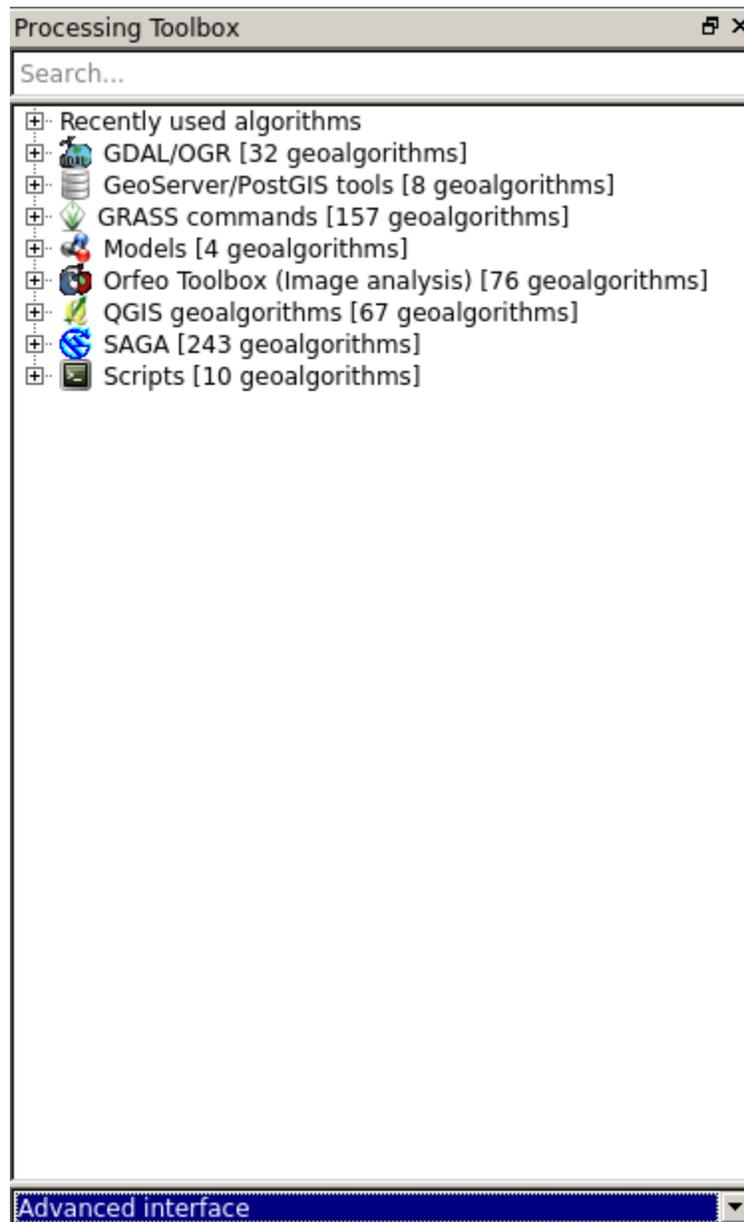


Figure 2.1: Processing Toolbox

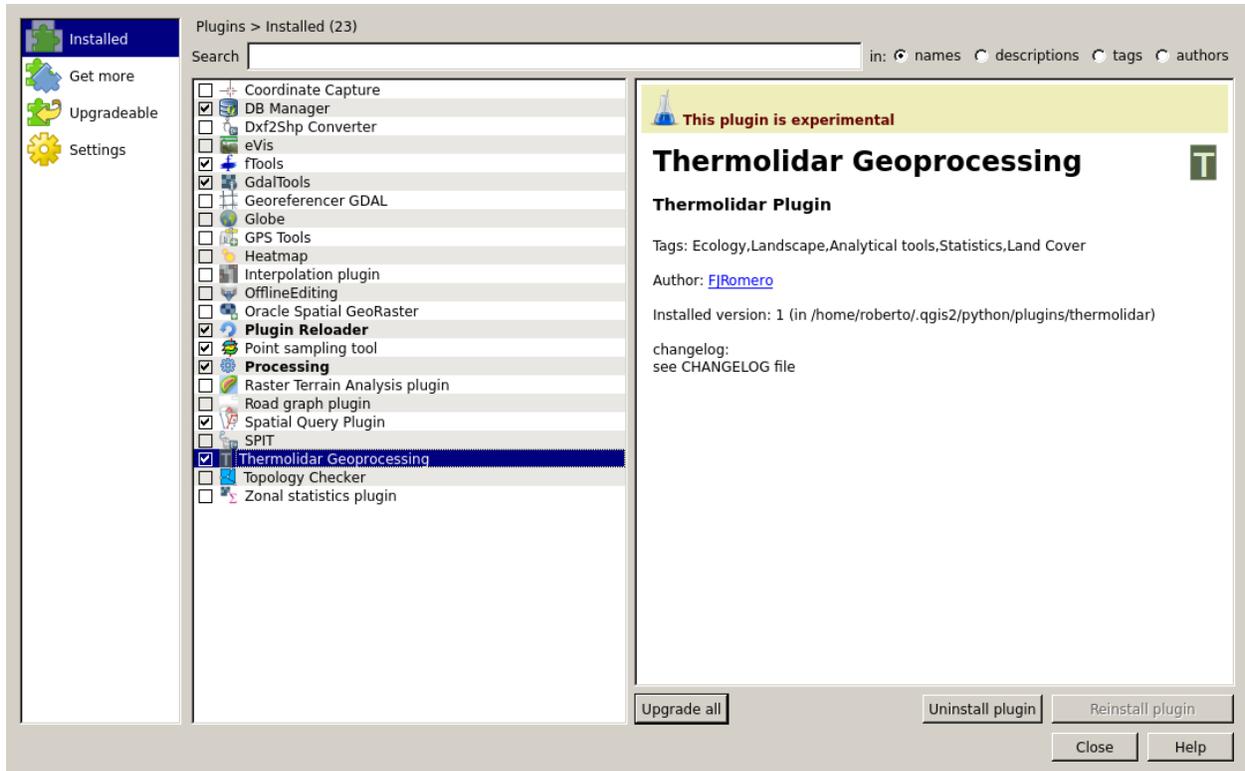


Figure 2.2: ThermoLiDAR plug-in is enabled in the *Installed* tab within the *Manage and Install Plugins...*

To activate them, open the *Processing > Options and configuration > Providers* menu. First, enable the *R statistical package* and provide the *R Scripts Folder* and the *R folder*. *R Scripts Folder* specifies where the R scripts are located, `C:\Users\admin\.qgis2\processing\scripts`, and *R folder* specifies where R is installed, `C:\Program Files\R\R-XXX (XXX stands for the current R version)`.

Finally, activate the enable the ThermoLiDAR plug-in and supplies the folder of the SPDLib binaries, commonly `C:\SPDLib\bin` in Windows systems and `/usr/local/bin`

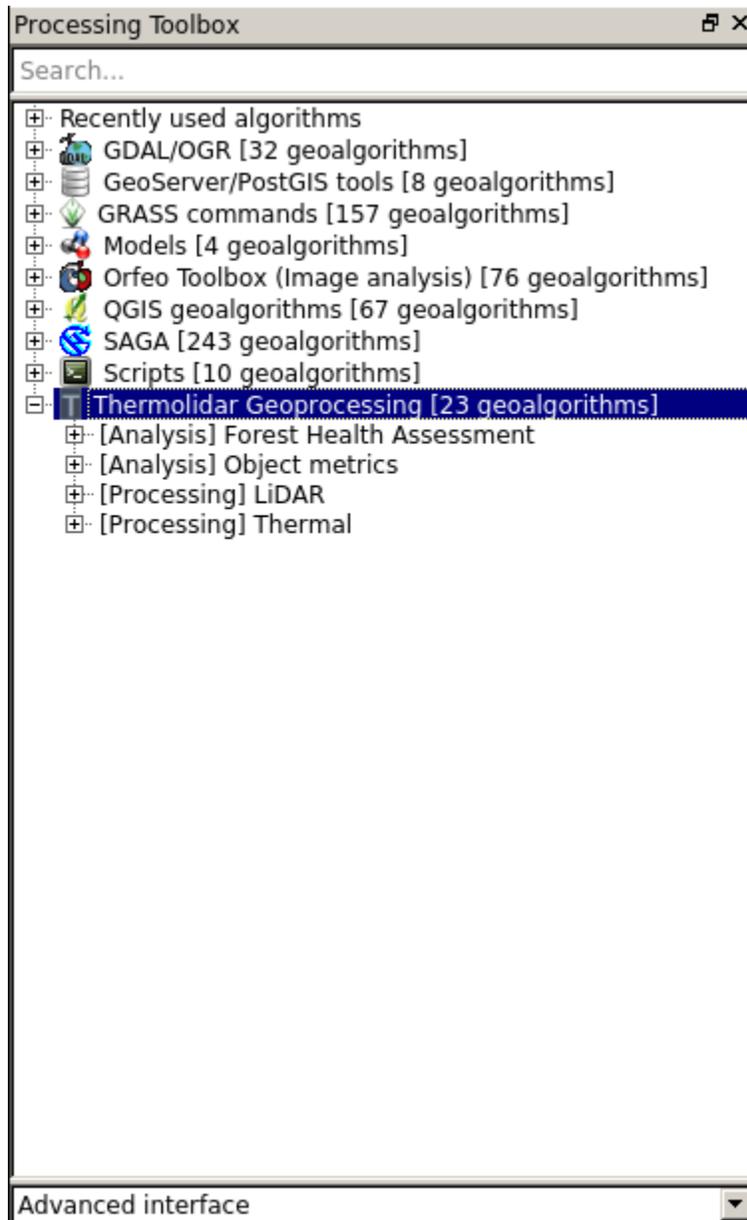
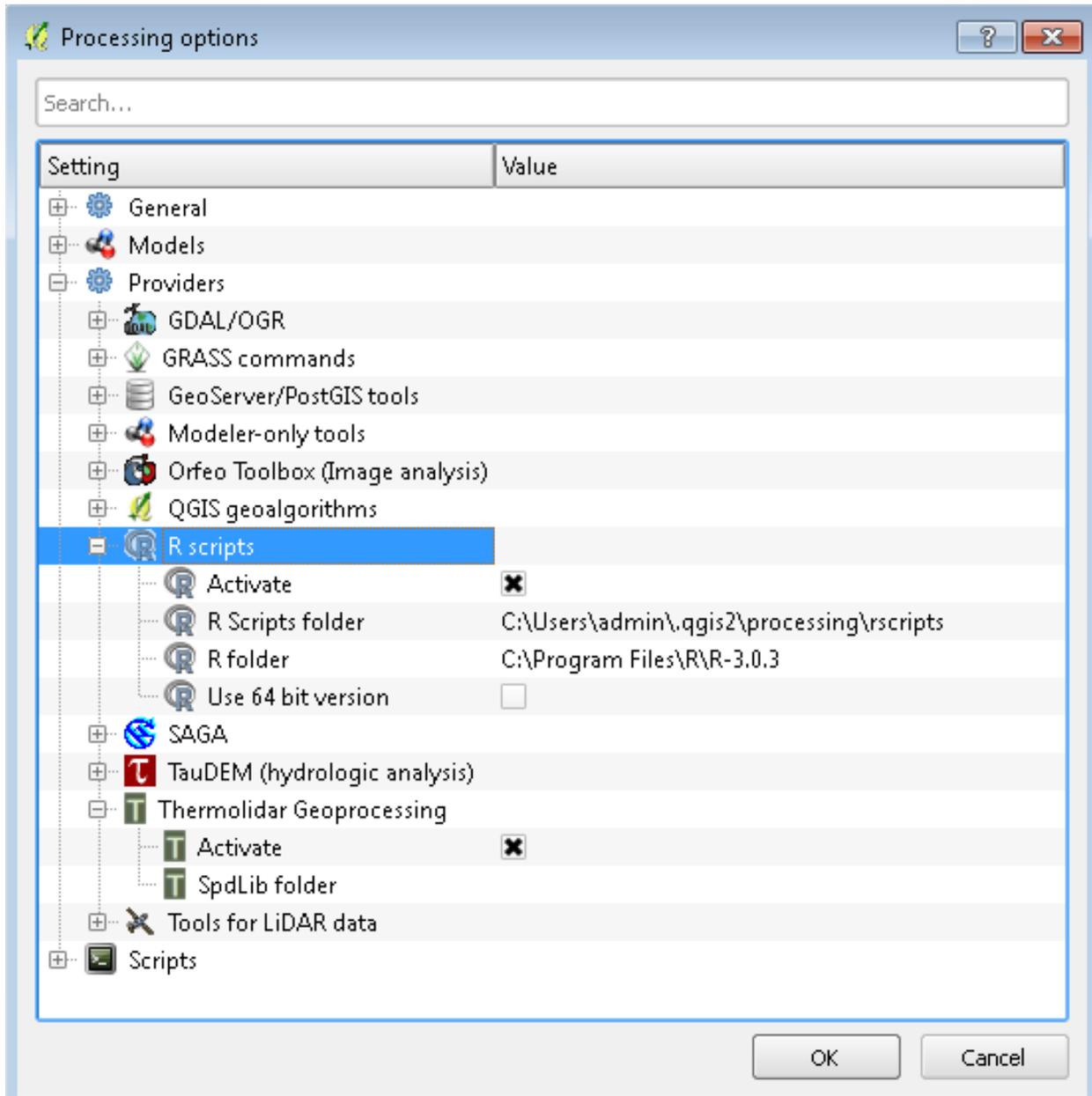


Figure 2.3: ThermoLiDAR fully integrated into Processing Toolbox

Figure 2.4: Visualization of *Processing > Options and configuration* window

3.1 LiDAR Processing

3.1.1 Convert File Formats - *spdtranslate*

The ***spdtranslate*** command is one of the key commands associated with SPDLib as it allows for the conversion between the various supported file formats, while it also supports coordinate system conversion. ..and the definition of the data origin (mainly for TLS data).

Although, the most common use of this tool is converting data to the SPD format. The simplest command for converting to UPD (SPD with a spatial index) is shown below, where the input and output file formats have been specified alongside the field used to attribute each pulse with a location to be used if the pulses where later index (i.e., into an SPD file):

```
spdtranslate --if LAS --of UPD -x FIRST_RETURN -i QueenElisabeth_example.las -o QueenElisabeth_examp
```

If you wish to explicitly define the projection of the SPD file then use the `--input_proj` and `--output_proj` switch to specify a text file containing the OGC WKT string representing the projection. The following command provides an example where the coordinate system (UK Ordnance Survey national grid) has been specified when converting data to an SPD file:

```
spdtranslate --if LAS --of UPD -x FIRST_RETURN --input_proj ./OSGB1936.wkt -i QueenElisabeth_example
```

While the following command converts from WGS84 to UK Ordnance Survey national grid while reading the file and converting to SPD:

```
spdtranslate --if LAS --of SPD --convert_proj --input_proj WGS84.wkt --output_proj OSGB1936.wkt -xFI
```

To convert data to the SPD format, where data is indexed on to a 10 m grid the following command is the simplest form.:

```
spdtranslate --if LAS --of SPD -x FIRST_RETURN -b 10 -i QueenElisabeth_example.las -o QueenElisabeth
```

Within SEXTANTE Toolbox a double click on *spdtranslate* opens its interface that looks like this:

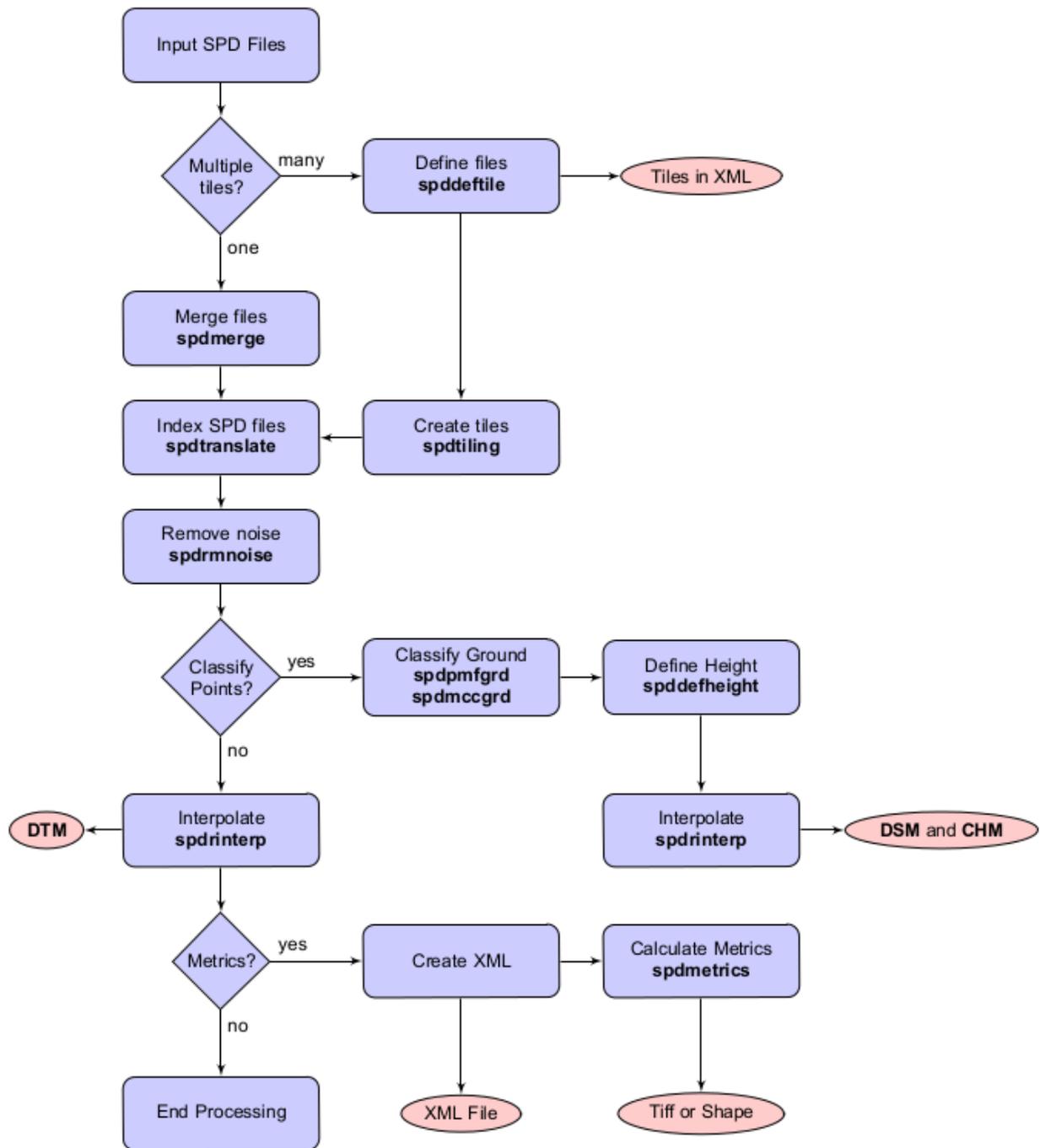


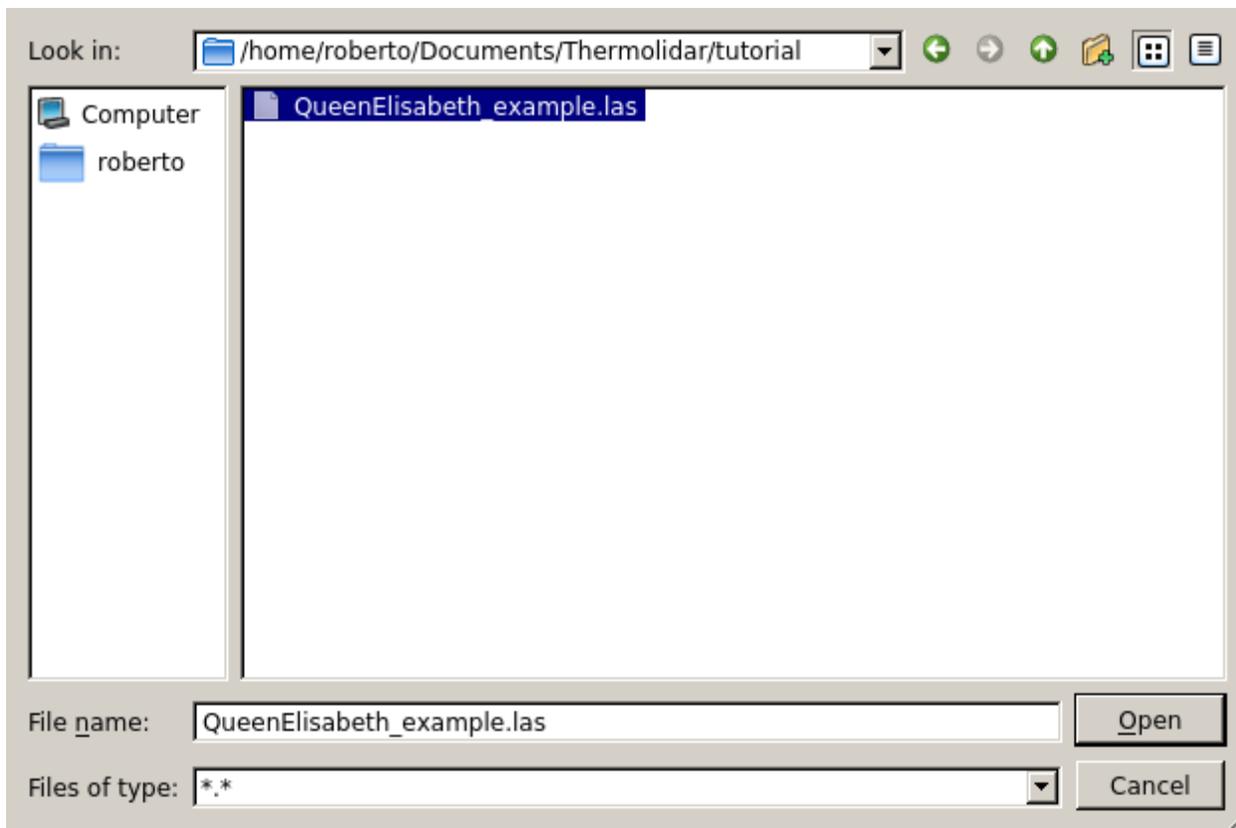
Figure 3.1: [Bunting2013b]

The screenshot shows a 'Parameters' dialog box with the following fields and options:

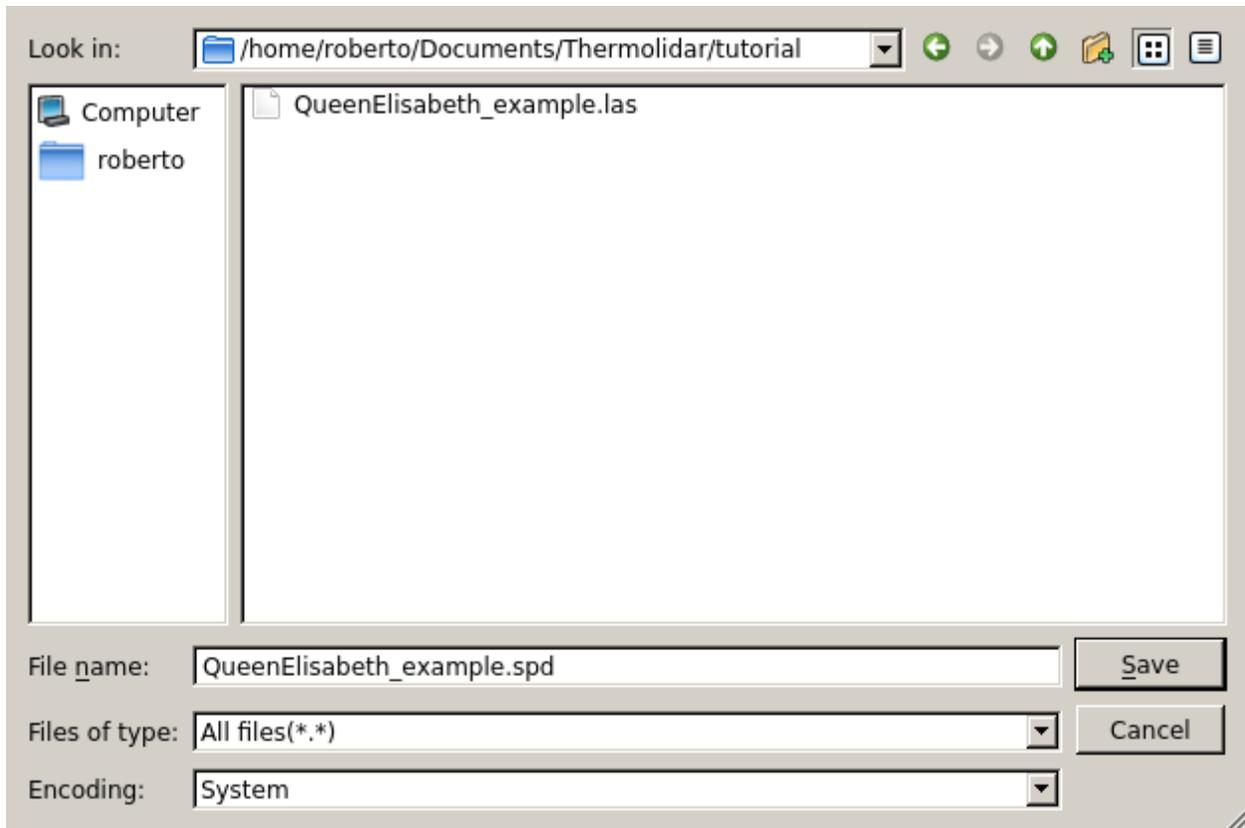
- Input file:** A text input field with a browse button (...).
- Format of the input file:** A dropdown menu set to 'LAS'.
- Format of the output file:** A dropdown menu set to 'SPD'.
- The location used to index the pulses:** A dropdown menu set to 'FIRST_RETURN'.
- Path where temporary file can be written to:** An empty text input field.
- File with a WKT string representing the projection of the input file:** A text input field with a browse button (...).
- File with a WKT string representing the projection of the output file:** A text input field with a browse button (...).
- A schema for the format of the file being imported (Only for ASCII format):** A text input field with a browse button (...).
- Bin size for SPD file index:** A spin box set to '1'.
- Number of columns within a tile, using this option generates a non-sequential SPD files:** A spin box set to '0'.
- Number of rows within a tile (Default 25):** A spin box set to '25'.
- Output file:** A text input field containing '[Save to temporary file]' with a browse button (...).

At the bottom of the dialog, there is a progress bar showing 0% and three buttons: 'Run', 'Close', and 'Cancel'.

Some options are set by default as the format of the input (LAS) and output (SPD) files, and the location used to index pulses (FIRST_RETURN) and the user has to set the other options manually. The two only required options are the input and output file. If the absolute path of the input file is known it can be written directly, otherwise there exists the possibility of searching for it by clicking the button on the right. This will open another interface.



The output file path is specify in the same way by clicking the button on the right of the output file field. In this case, the user search for the path where the output file should be saved into and writes the name of the file. It is worth to notice that the file extension has to be set and it should agree with the output format.



Once both files have been set the **spdtranslate** interface should look like this

Parameters | Log | Help

Input file
 ...

Format of the input file

Format of the output file

The location used to index the pulses

Path were temporary file can be written to

File with a WKT string representing the projection of the input file
 ...

File with a WKT string representing the projection of the output file
 ...

A schema for the format of the file being imported (Only for ASCII format)
 ...

Bin size for SPD file index
 ...

Number of columns within a tile, using this option generats a non-sequential SPD files
 ...

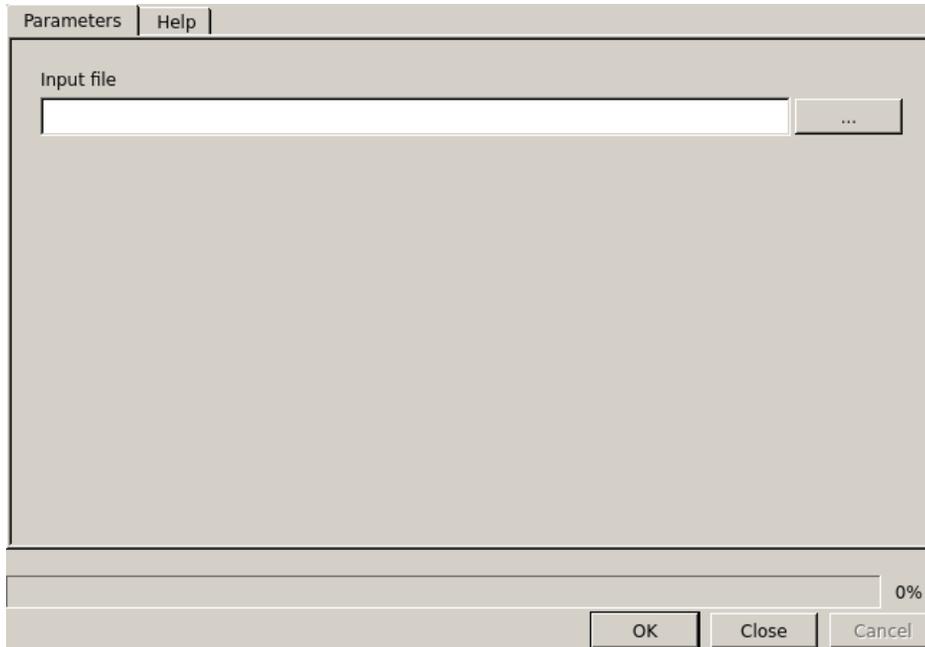
Number of rows within a tile (Default 25)
 ...

Output file
 ...

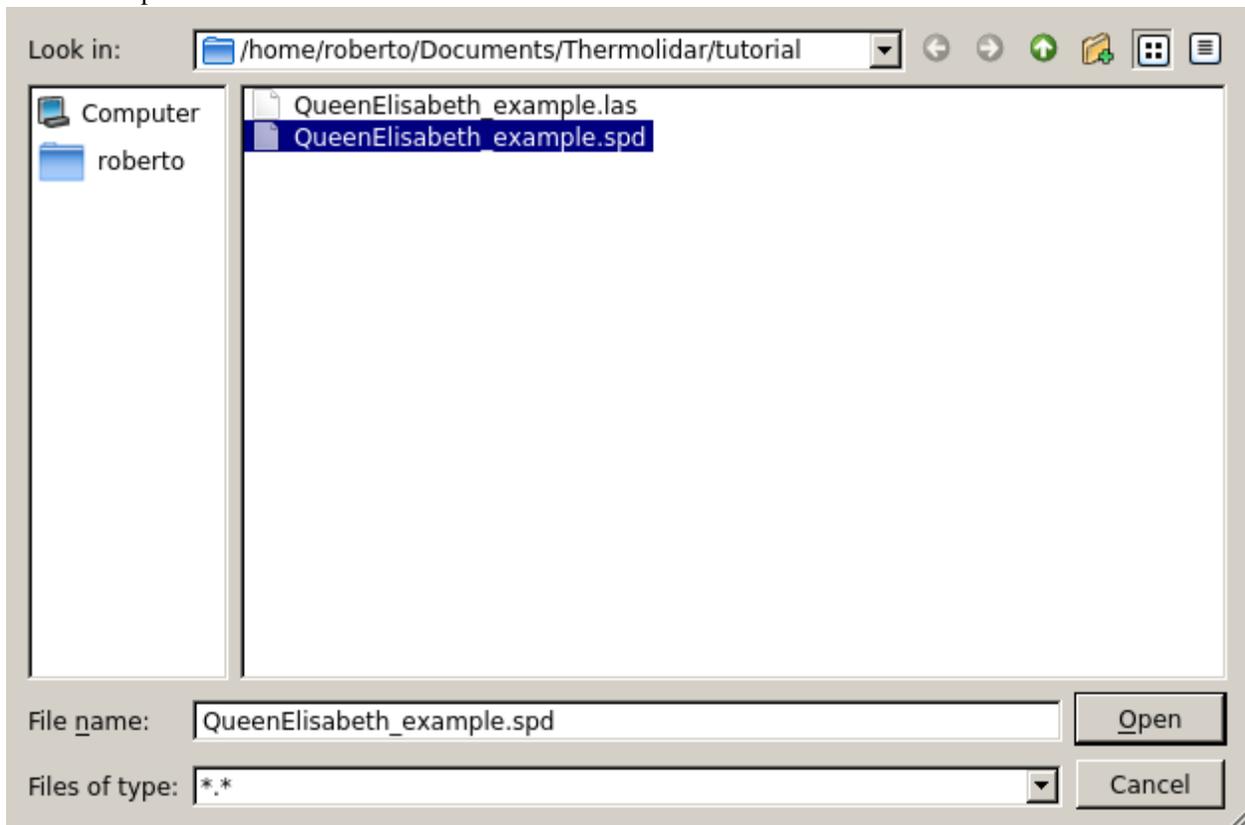
0%

Run Close Cancel

and it should be ready to be executed by clicking **OK**. To be sure **spdtranslate** has worked properly, the user can visualise the new **spd** file into **QGIS** by means of the **spdimport** tool.

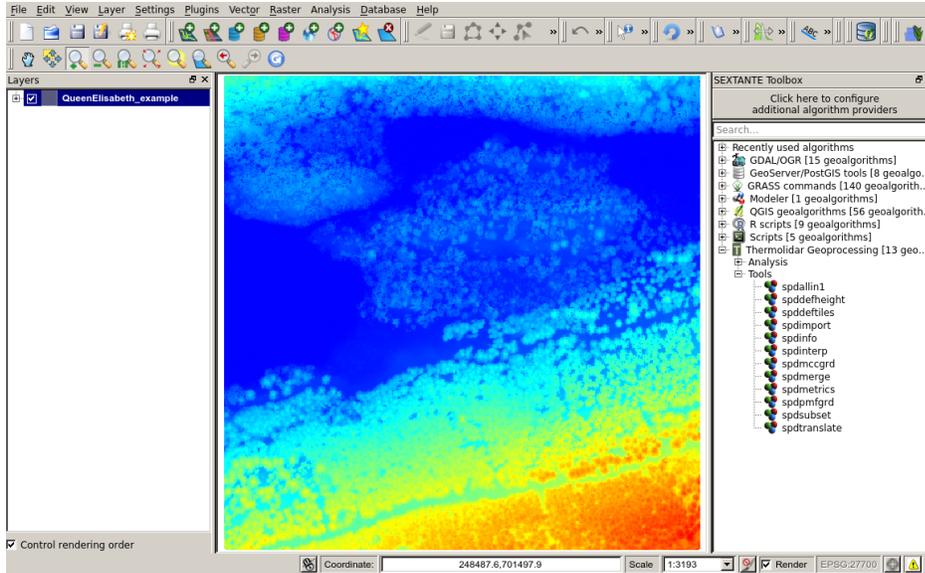


To load the spd file click on the  button and the user will be asked to select a file.



Warning: Be sure to select the spd file, otherwise the **spdimport** module will not work

Once the spd file has been loaded, QGIS will show the result in the canvas:



For more info look at the `spdtranslate` help file.

Temporary files

In the examples given above the whole input files are read into memory and sorted into the spatial grid before being written output the file. This requires that you have sufficient memory to store the whole dataset and index data structure in memory. If you do not have sufficient memory to complete this operation the file needs to be tiled, into blocks which are small enough to fit into memory. Allowing the SPD file to be built in stages, naturally this is slower but once completed it is very fast to make spatial selections within the file and other processing steps can be applied to the whole file with only a relatively small memory footprint. The option to select tiling the file to disk while building the SPD file is `--tempath` which is path and base file name while the tiles will be written. In the QGIS graphical user interface this option is called '*Path where temporary file can be written to*'.

Supported File Formats

The file formats supported are the ones which we have so far required for our research therefore the current level of support is not intended to encompass all the available formats:

- SPD
- UPD
- LAS
- ASCII

SPD The *Sorted Pulse Data* (SPD) file format [Bunting2013b] has been designed specifically for the storage of LiDAR waveform and discrete return data acquired by **TLS**, **ALS** and space borne systems, and includes support for multiple wavelengths within a single file. The format uses a pulse-based structure as opposed to a solely point-based structure, where pulses contain all the information associated with a transmitted pulse from the sensor. The SPD format also supports 2D spatial indexing of the pulses, where pulses can be referenced using cartesian, spherical or polar coordinate systems and projections

UPD The *Unsorted Pulse Data* (UPD) files are SPD files without a spatial index.

LAS The LAS reader is via `LibLAS` and therefore only supports the discrete return (LAS 1.2) data. LAS 1.2 files are exported through the `libLAS` library so as with the importer only discrete return data are supported.

ASCII The ASCII format requires a schema, written in XML, to be supplied. The parser expects a single return per line and the resulting pulses will only contain a single return. The following are some examples of schema's for common ASCII formats.

A schema for the PTS format:

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2     <line delimiter=" " comment="#" ignorelines="0" >
3         <field name="X" type="spd_double" index="0" />
4         <field name="Y" type="spd_double" index="1" />
5         <field name="Z" type="spd_float" index="2" />
6         <field name="AMPLITUDE_RETURN" type="spd_uint" index="3" />
7         <field name="RED" type="spd_uint" index="4" />
8         <field name="GREEN" type="spd_uint" index="5" />
9         <field name="BLUE" type="spd_uint" index="6" />
10 </line>

```

A schema for the XYZ format:

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2 <line delimiter="," comment="#" ignorelines="0" >
3     <field name="X" type="spd_double" index="0" />
4     <field name="Y" type="spd_double" index="1" />
5     <field name="Z" type="spd_float" index="2" />
6     <field name="AMPLITUDE_RETURN" type="spd_uint" index="3" />
7 </line>

```

3.1.2 Classify Ground Returns

Progressive Morphology Filter - `spdpmfgrd`

The `spdpmfgrd` command is an implementation of the progressive morphological filter algorithm of [Zhang2003]. The algorithm works by generating an initial minimum return raster surface at the bin resolution of the SPD file. Circulate morphological operators of a range of scales. At each scale a morphological closing (erosion + dilation) operation is performed. The new height value from the morphological operator is kept if it is above the elevation difference threshold where the elevation difference threshold is increased between threshold using a slope value.

Under most circumstances the default parameters for the algorithm will be fit for purpose, but be careful that the bin size used within SPD is not too large as the processing will be at this resolution:

```
spdpmfgrd -i QueenElisabeth_example.spd -o QueenElisabeth_example_pmfgrd.spd
```

The `spdpmfgrd` QGIS interface has only three options to be set. Apart from the input and output files, you can select the class the filter will be applied to.

Parameters	Log	Help
Input file	/media/DATA/Thermolidar/tutorial/data/QueenElisabeth_example.spd	
Only use points of particular class	0	
Generated an image rather than classifying points (useful for parameter selection)	[Not selected]	
Bin size for SPD file index	0	
Size (in bins) of the overlap between processing blocks	10	
Number of columns within a tile (Generates a non-sequential SPD file)	0	
Number of rows within a tile	100	
Size of the median filter (half size i.e., 3x3 is; 5x5 is 2...)	2	
Do not run a median filter on generated surface (before classifying ground point or export)	No	
Threshold for deviation from identified ground surface for classifying ground returns	0.300000	
Maximum elevation difference threshold	5	
Initial elevation difference threshold	0.300000	
Slope parameter related to terrain	0.300000	
Maximum size of the filter	7	
Initial size of the filter (half size i.e., 3x3 is; 5x5 is 2...)	1	
Output file	/media/DATA/Thermolidar/tutorial/data/QueenElisabeth_example_pmfgrd.spd	
0%		
<input type="button" value="Run"/> <input type="button" value="Close"/> <input type="button" value="Cancel"/>		

For more info look at the `spdpmfgrd` help file.

Filter points depending on class

The `--class` option allows the filter to be applied to returns of a particular class (i.e., if you have ground returns classified but it needs tidying up etc). It's useful for TLS as it can take a thick slice with `mcc` algorithm and then use the PMF algorithm to tidy that result up to get a good overall ground classification.

Multi-Curvature Classifier – `spdmccgrd`

The `spdmccgrd` command is an implementation of the multi-scale curvature algorithm [EvansHudak2007]. This algorithm was created at the US Forest Service and does a good job at classifying ground returns under a forest canopy while retaining the terrain but it does not differentiate the buildings. Under most circumstances the default parameters for the algorithm will be fit for purpose and it is recommend that you try these first with the following command.:

```
$ spdmccgrd -i QueenElisabeth_example.spd -o QueenElisabeth_example_mccgrd.spd
```

The `spdmccgrd` QGIS interface has only three options to be set. Apart from the input and output files, you can select the class the filter will be applied to.

Parameters | Log | Help

Input file
/media/DATA/Thermolidar/tutorial/data/QueenElisabeth_example.spd

Only use points of particular class
0

Bin size for SPD file index
0

Size (in bins) of the overlap between processing blocks
10

Number of columns within a tile (Generates a non-sequential SPD file)
0

Number of rows within a tile
100

Use only multiple return pulses to calculate the amount of change between iterations
No

Median filter to smooth the generated raster instead of an average filter
No

The threshold for the
0.100000

The size of the smoothing filter (half size i.e., 3x3 is 1)
1

The number of points used for the TPS interpolation
16

Maximum search radius for the TPS interpolation
20

Iteration step curveture tolerance parameter
0.500000

Minimum curveture tolerance parameter
0.100000

Initial curveture tolerance paramete
1

Gap between increments in scale
0.500000

The number of scales below the init scale to be used
1

The number of scales above the init scale to be used
1

Initial processing scale (Usually the native resolution of the data)
0

Output file
/media/DATA/Thermolidar/tutorial/data/QueenElisabeth_example_mccgrd.spd

0%

Run Close Cancel

For more info look at the `spdmccgrd` help file.

Filter points depending on class

As in the `spdpmfgrd` case, the `--class` option allows the filter to be applied to returns of a particular class. It's useful as it can take a thick slice with PMF algorithm and then use the MCC algorithm to tidy that result up to get a good overall ground classification.

Combining filters

Another option which can improve the ground return classification is to combine more than one filtering algorithm to take advantage of their particular strengths and weaknesses. A particularly useful combination is to first run the PMF algorithm where a ‘thick’ slice is taken (e.g., 1 or 2 metres above the raster surface) and then the MCC is applied to find the ground returns (using the `--class 3` option):

```
spdmccgrd -i --class 3 QueenElisabeth_example_pmfgrd.spd -o QueenElisabeth_example_mccgrd.spd
```

The screenshot shows the GUI for the `spdmccgrd` command. It features a 'Parameters' tab and a progress bar at the bottom showing 0% completion. The parameters are as follows:

Parameter	Value
Input file	/media/DATA/Thermolidar/tutorial/data/QueenElisabeth_example_pmfgrd.spd
Only use points of particular class	0
Bin size for SPD file index	0
Size (in bins) of the overlap between processing blocks	10
Number of columns within a tile (Generates a non-sequential SPD file)	0
Number of rows within a tile	100
Use only multiple return pulses to calculate the amount of change between iterations	No
Median filter to smooth the generated raster instead of an average filter	No
The threshold for the	0.100000
The size of the smoothing filter (half size i.e., 3x3 is 1)	1
The number of points used for the TPS interpolation	16
Maximum search radius for the TPS interpolation	20
Iteration step curvature tolerance parameter	0.500000
Minimum curvature tolerance parameter	0.100000
Initial curvature tolerance parameter	1
Gap between increments in scale	0.500000
The number of scales below the init scale to be used	1
The number of scales above the init scale to be used	1
Initial processing scale (Usually the native resolution of the data)	0
Output file	/media/DATA/Thermolidar/tutorial/data/QueenElisabeth_example_mccgrd.spd

3.1.3 Define Height Field - `spddefheight`

The `spddefheight` command is used to define the height field within both the pulse and point fields of the SPD data file. This can be done in two ways, the simplest is the use of a DTM of the same resolution as the SPD file bin size. The disadvantage of using a DTM is that it is in effect using a series of spot heights and this can introduce artefacts. Therefore, interpolating a value for each point/pulse generates a continuous surface reducing any artefacts. The recommended approach is to use Natural Neighbour interpolation, as demonstrated in the paper of [BaterCoops2009].

Without Interpolation, using DTM

Using the DTM method the only parameters are the input file and output files. The raster DTM needs to the same resolution as the SPD grid and it can be any raster format supported by the [GDAL](#) library.

Interpolation Mode

The interpolators are the same as those defined within the `spdinterp` command so look to the [Interpolate DTM and DSM](#) section for details on their use. The recommend command using the *natural neighbour* interpolation algorithm, along with the default parameters is:

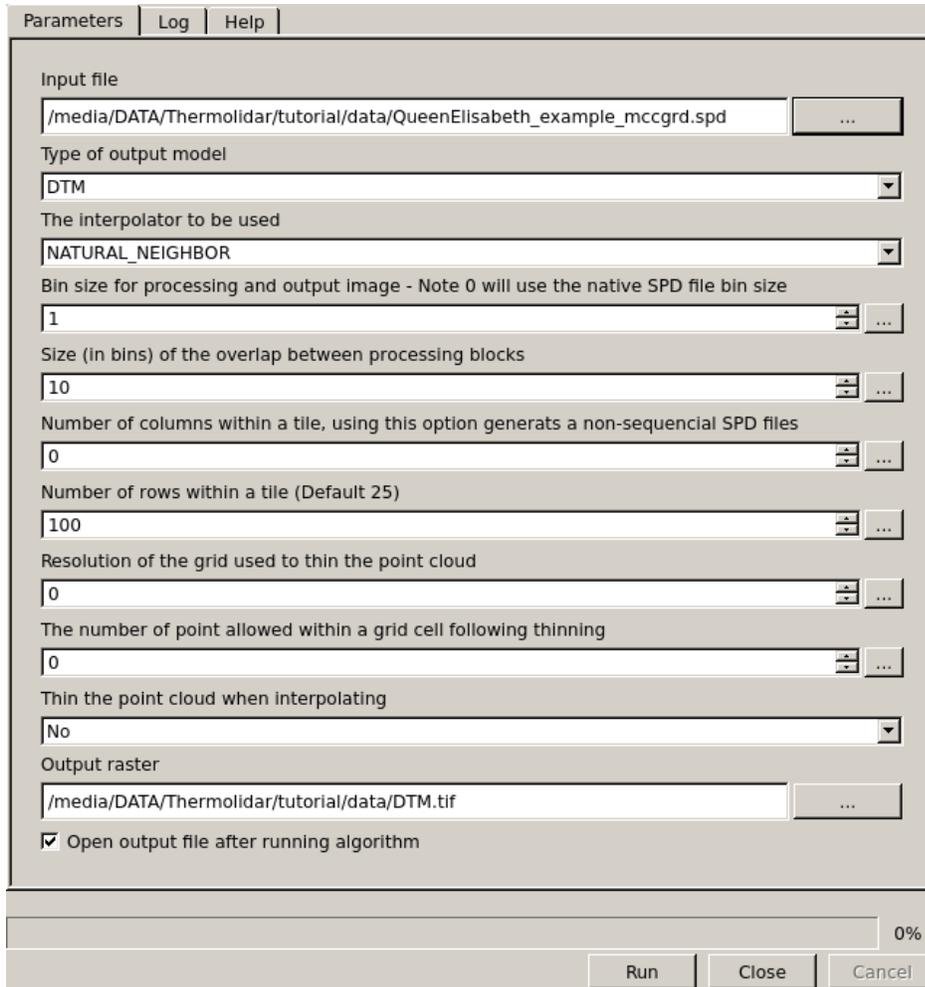
```
spddefheight --interp --in NATURAL_NEIGHBOR -i QueenElisabeth_example_mccgrd.spd -o QueenElisabeth_ex
```

3.1.4 Interpolate DTM and DSM

The most common product to be created from a LiDAR SPD dataset are Digital Terrain Model (DTM), Digital Surface Model (DSM) and Canopy Height Models (CHM). To produce those products you need to interpolate a raster surface from the classified ground returns and top surface points. A key parameter is the resolution of the raster which is generated, within SPDLib the resolution of the raster needs to be a whole number multiple of the SPD index, for example, if the SPD file has a bin size of 10 m then the the output raster file resolution can be 1, 2 or 5 m but not 3 m.

The same module **spdinterp** will permit to generate DTMs, DSMs and CHMs by choosing the output model option and the type of point elevation. To interpolate topographic point elevations generates a DTM in the next way:

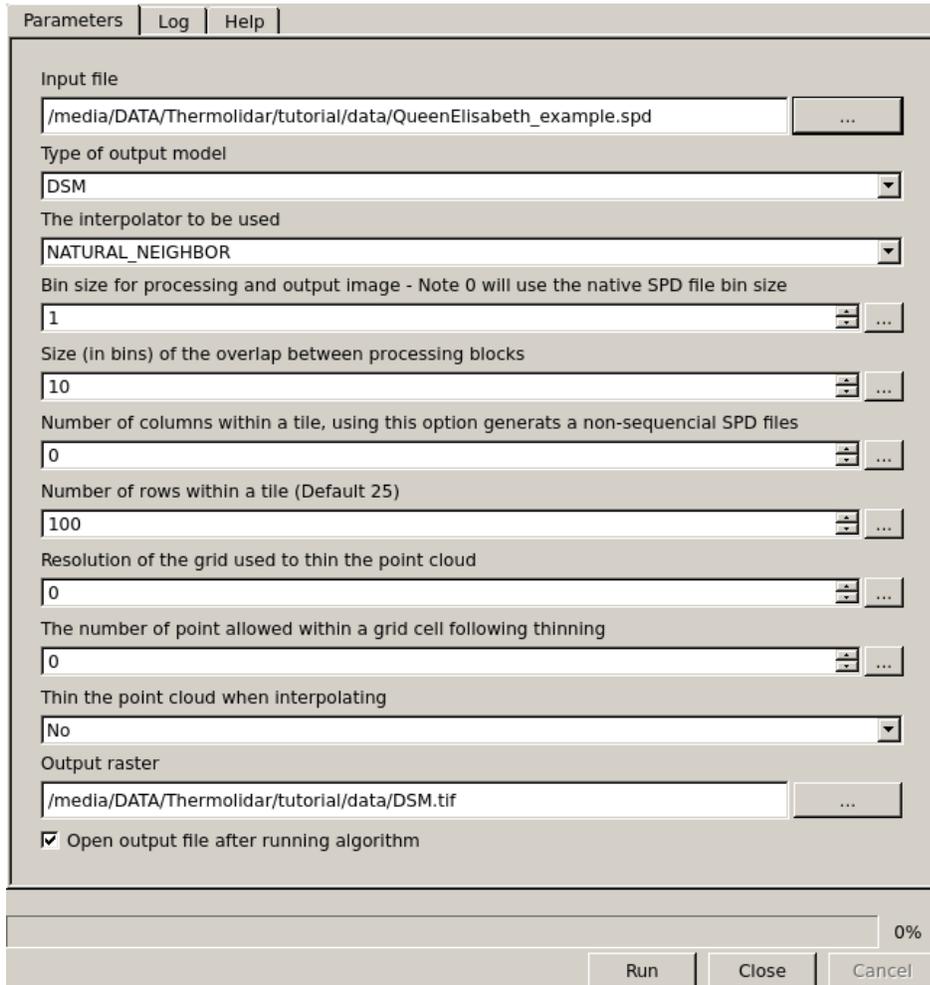
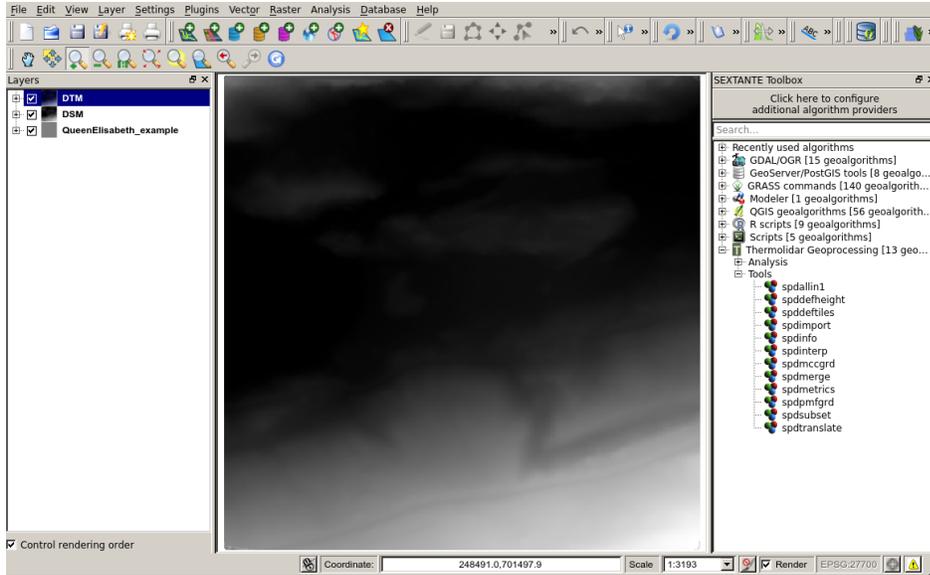
```
$ spdinterp --dsm --topo --in NATURAL_NEIGHBOR -f GTiff -b 1 -i QueenElisabeth_example.spd -o DSM.tif
```

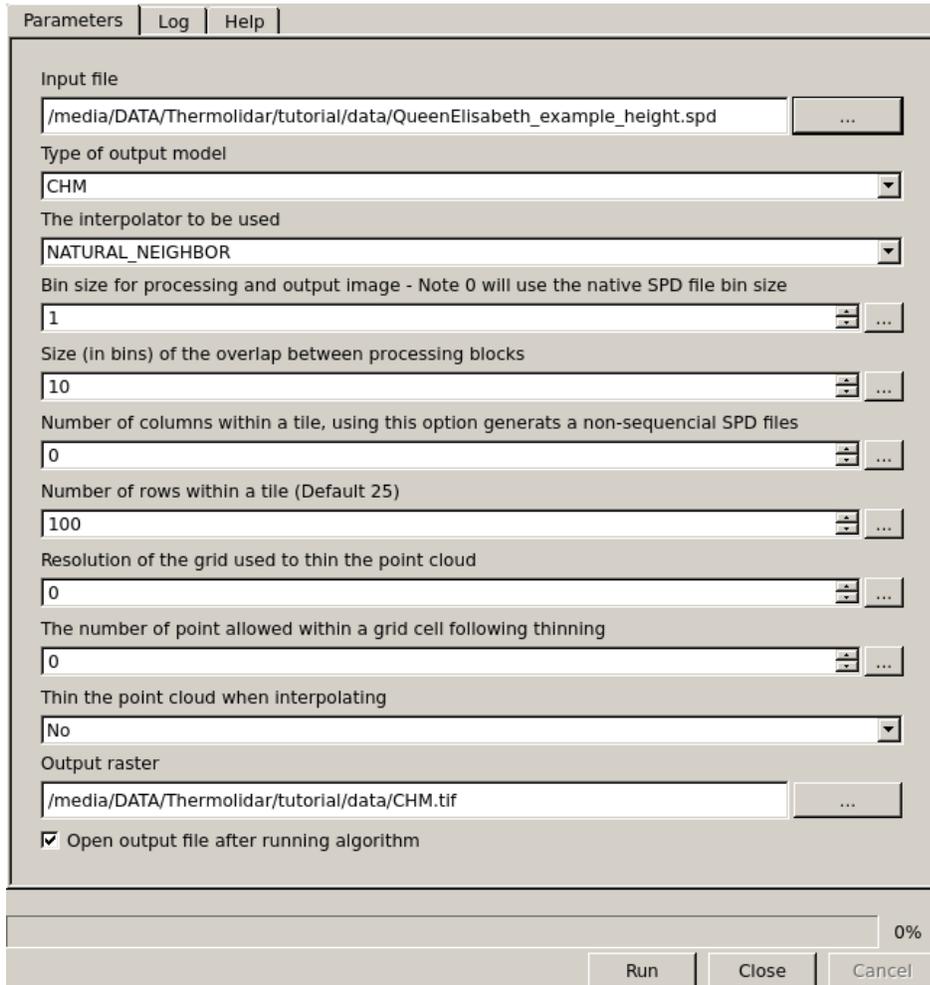
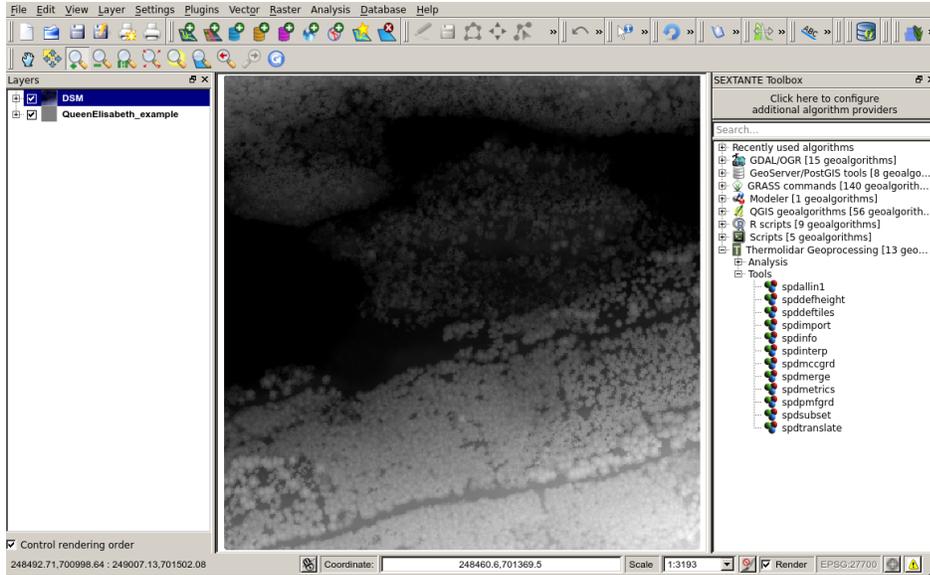


Moreover, importing the spd files generated so far would not show any different result from that imported at the beginning of this tutorial:

```
$ spdinterp --dtm --topo --in NATURAL_NEIGHBOR -f GTiff -b 1 -i QueenElisabeth_example_mccgrd.spd -o
```

```
$ spdinterp --dsm --height --in NATURAL_NEIGHBOR -f GTiff -b 1 -i QueenElisabeth_example_mccgrd.spd -o
```





Parameters | Log | Help

Input SPD file

XML input file

Shapefile vector input file

Output format

Bin size for SPD file index

Number of columns within a tile (Generates a non-sequential SPD file)

Number of rows within a tile

Output file

Open output file after running algorithm

0%

Run Close Cancel

Multiple metrics can be calculated at the same time if listed within an XML. This XML file has to be defined *a priori* with a hierarchical list of metrics and operators. Within the **metrics** tags a list of metrics can be provided by the **metric** tag. Within each metric the **{field}** attribute is used to name the raster band or vector attribute. Here is an example of an SPD metrics XML file template containing the maximum, the average, the median, the number of pulses, the canopy cover and the percentile 95th.

Parameters | Help

Input SPD file
/home/roberto/Documents/Thermolidar/tutorial/QueenElisabeth_example_height.spd ...

XML input file
/home/roberto/Documents/Thermolidar/tutorial/metrics.xml ...

Shapefile vector input file
... 

Run metrics with...
Image

Bin size for processing and output image (Default 0) - Note 0 will use the native SPD file bin size.
1 ...

Output file
/home/roberto/Documents/Thermolidar/tutorial/maxheight.tif ...

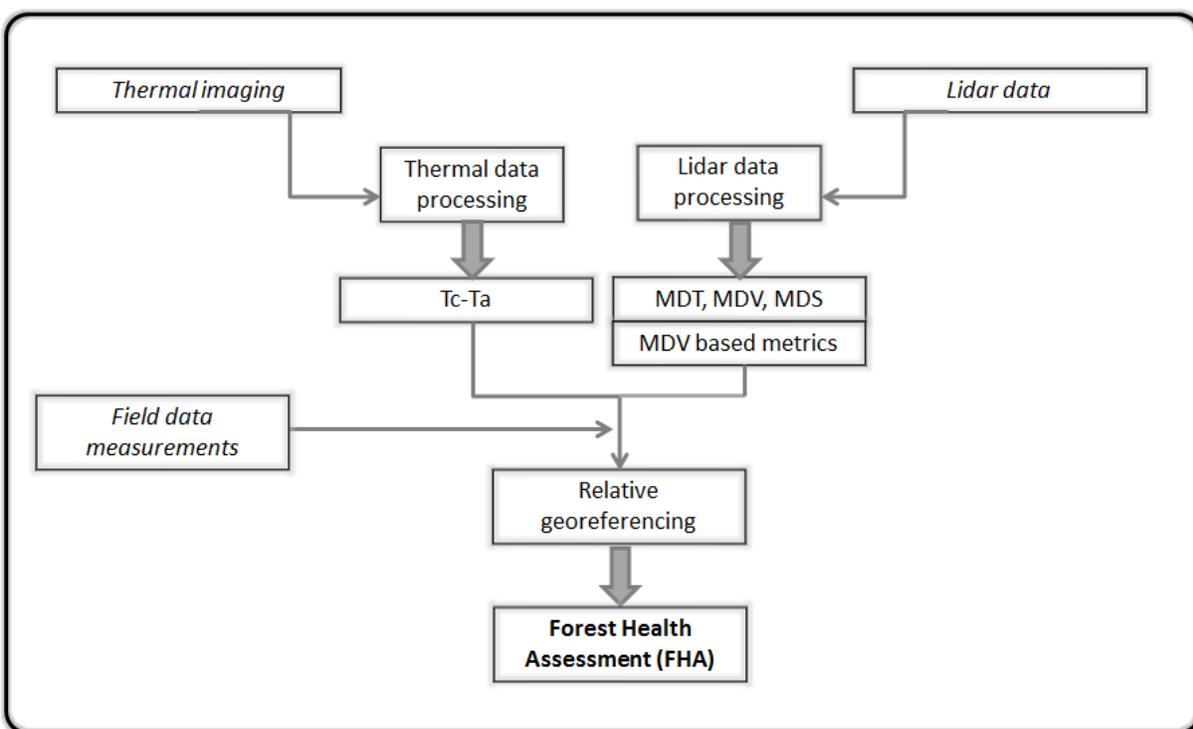
0%

OK Close Cancel

4.1 Introduction

This section summarizes the description of the software and the practical application of the tools implemented using thermal and lidar data collected in the framework of THREMOLIDAR project.

The main structure of the software is summarized in the following figure. Following is a detail of the main processes carried out by the software.



4.1.1 Software structure

Processing

This package includes tools for conducting the processing of raw Thermal and LiDAR data in order to obtain the products required to achieve the parametric analysis of forest health assessment.

- **A.1. Lidar processing.** LiDAR data tool set for the generation of DSMs, DTMs, DVMs and vegetation statistical derivatives thereof.
- **A.2. Thermal image processing.** Thermal data tool set for the calibration of RAW airborne thermal imaging. In addition, tools provide the possibility of calculating a derivate indicator using the difference of the air temperature minus the crown temperature.
- **A.3. Ortorectification and Lidar/Thermal integration.** Geo-reference the thermal data obtained through LiDAR pulses.

Data Analysis

This package includes tools for conducting the simultaneous analysis of thermal and lidar data information linked to field data measurements to evaluate the state and trends of forest health. A detailed description of data inputs and processes applied is included in Fig. 2

- **B.1. Forest Stand Segmentation (FSS).** The processed image is decomposed into regions or objects. Object based delineation algorithms are applied with this tool to define forest stands unis for further study.
- **B.2. Health condition levels (HCL).** Different physiological indicators from field data measurements are processed with this tool to define the ground truth condition of forest status. Health condition levels are statistically generated based in clustering and subsequently validated by ANOVA.
- **B.3. Structurally homogeneous forest units (SHFU).** This option provides the tools required for the classification of forest stands structurally different. The data applied to perform this classification is defined by the user. In this report the main average height of the trees estimated based on Lidar data has been applied as input data. Alternatively, user can provide external forest maps in a .shp format type with an attribute of the number of class.
- **B.3. Forest Health Monitoring (FHM).** The final result of this toolkit is the classification of forest health condition levels. This process requires the training sample plots analyzed and classified in step 1, the forest stand classification calculated in step 2 (or externally provided) and the thermal image data. Then, the user can apply a supervised classification of thermal data using the training sample plots classified in different health condition levels and based on a specific structural stand level. Optionally, users can apply a non-supervised classification in case the absence of field data measurements.

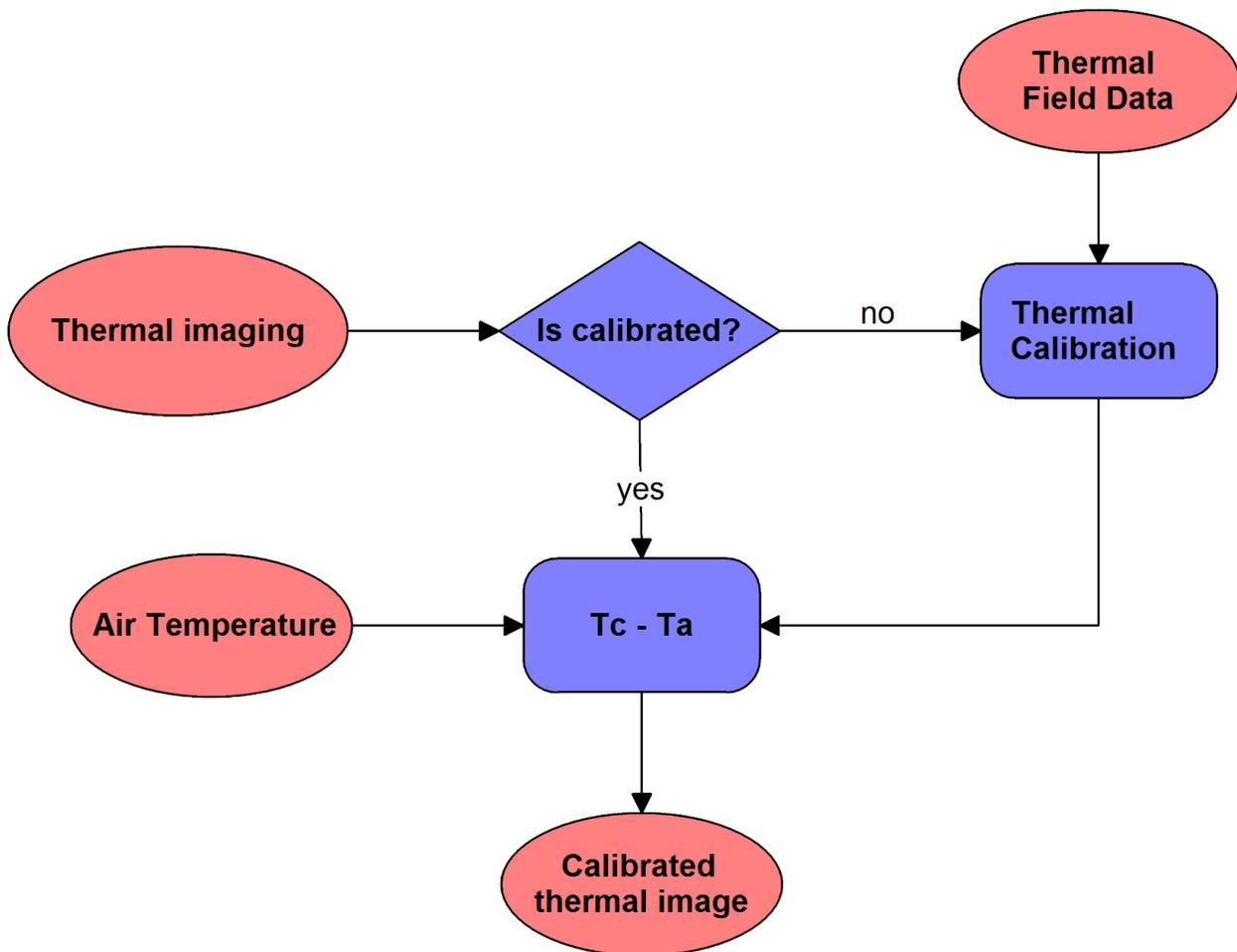
4.2 Thermal Processing

Thermal processing tools data allows a user to perform thermal imaging calibration using the Emissive Empirical Line Method (EELM) which is a common method for airborne thermal data processing based on the ‘In-scene atmospheric correction methods. These approaches were developed to remove atmospheric effects from hyperspectral imaging data allowing the user to utilize similar conditions to the atmosphere state. The advantage of using this type of methods over model-based methods based in radiative transfer theory is that they capture the true state of the atmosphere at the time of data collection and the relative low computational efforts required to perform the corrections (in comparison with radiative transfer models approaches). The main difficulty for in-scene method is getting correctly the field measurements parameters required for the correction algorithm.

The Emissive Empirical Line Method (EEELM) is the infrared extension of the widely known Empirical Line Method (ELM) atmospheric correction. EEELM employs a linear regression for each band to relate at-sensor radiance with the ground leaving radiance (GLR) via target emissivity and temperature by generating atmospheric transmission, upwelling radiance, and downwelling radiance terms. EEELM requires at least one bright target, one dark target and it is also recommended to measure any intermediate target.

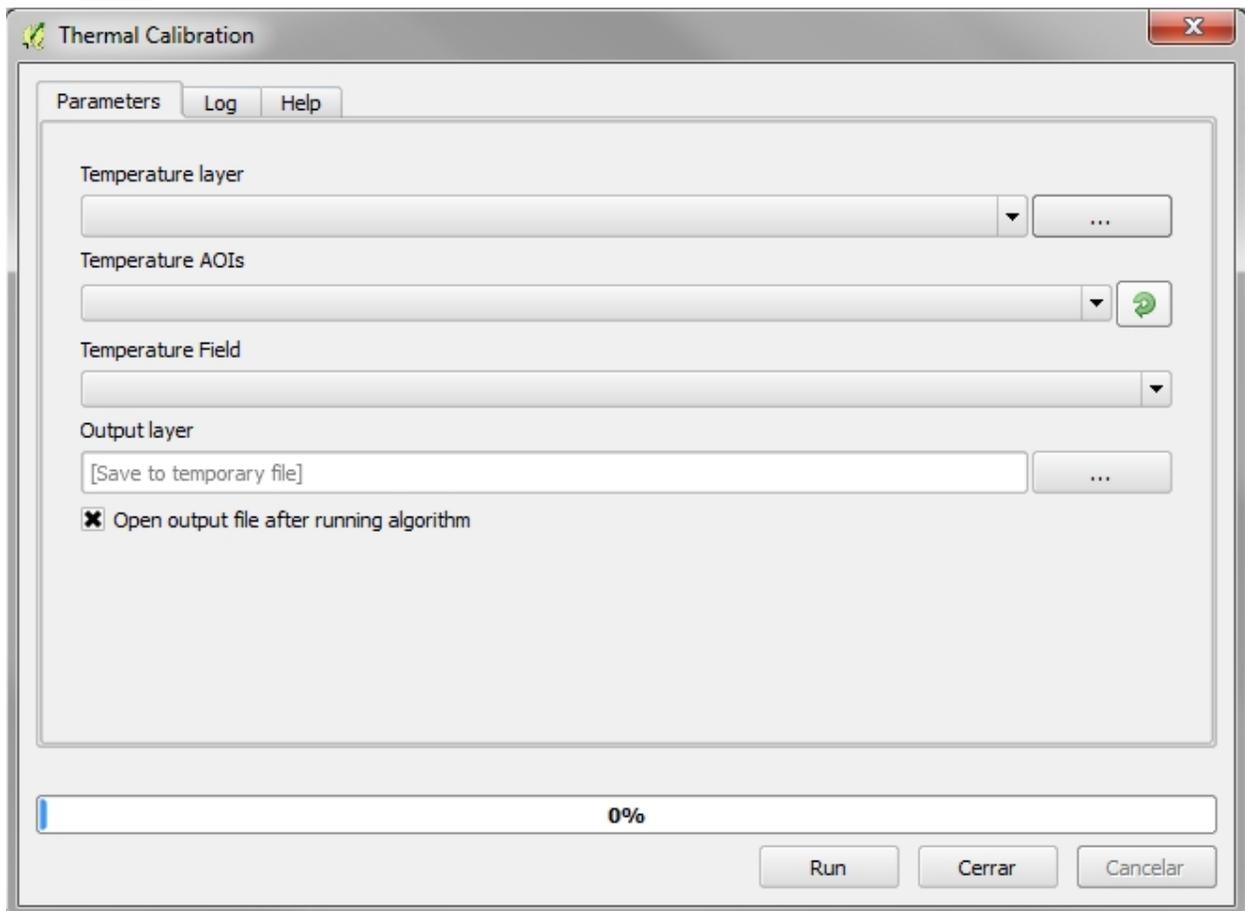
Use Empirical Line Compute Factors calibration to force spectral data to match selected field reflectance spectra. A linear regression is used for each band to equate DN and reflectance. This is equivalent to removing the solar irradiance and the atmospheric path radiance. The following equation shows how the empirical line gain and offset values are calculated.

This tool will also allow the user to calculate the difference between Crown Temperature minus Air temperature ($T_c - T_a$). This indicator has been widely demonstrated to be related with different physiological indicators such as stem water potential, stomatal conductance or sap flow rate. User need to select as input the thermal imaging and the air temperature collected in the same time of the airborne imaging acquisitions.



4.2.1 Thermal Calibration

This module allows to calibrate thermal images based on the calibration data obtained in the field. During the calibration process, spectral data is linearly correlated with selected field reflectance spectral.



Required Input Parameters

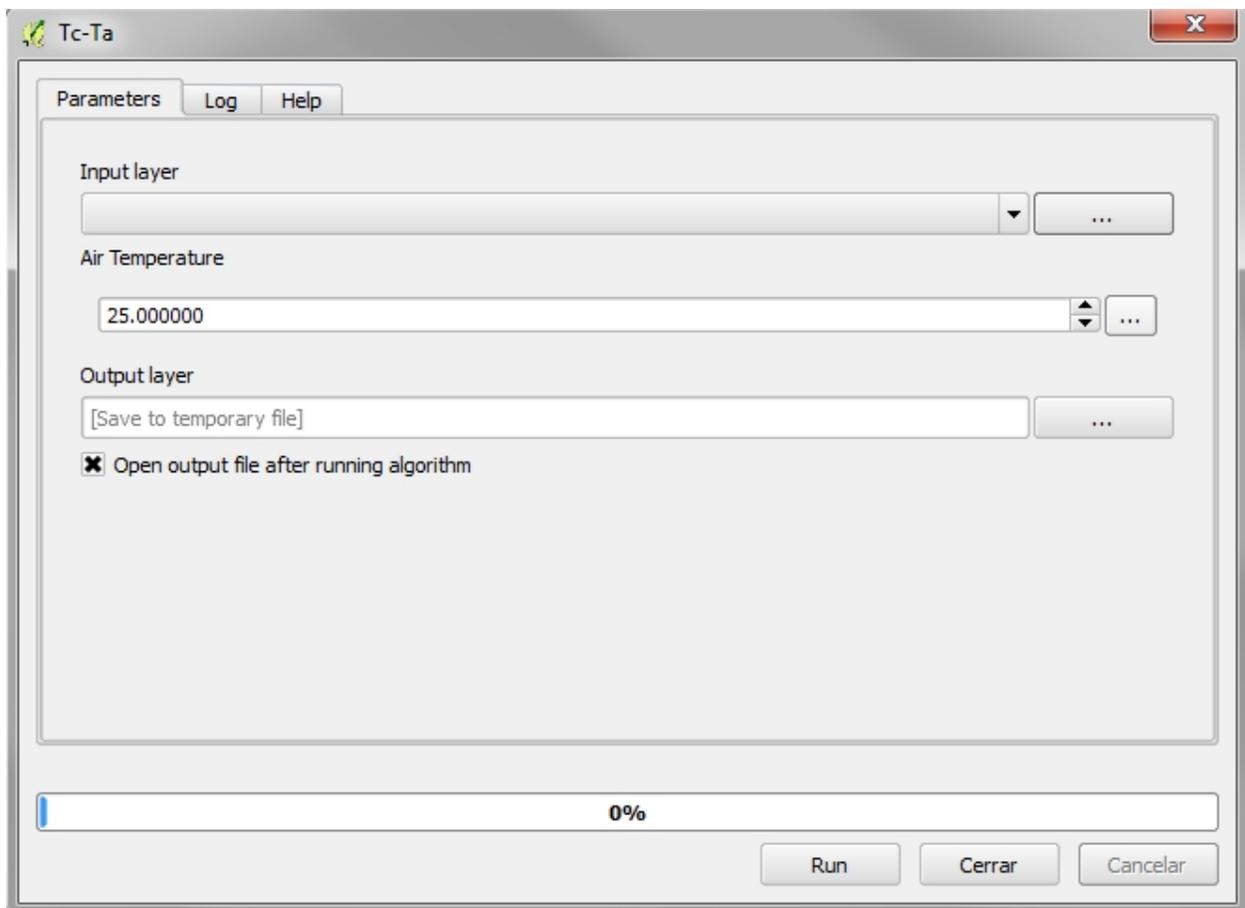
- **Temperature layer:** Thermal raster
- **Temperature AOIs:** Shapefile containing information on temperature field measurements
- **Temperature Field:** Vector's field containing temperature

Output Parameters

- **Output layer:** Calibrated thermal raster output.

4.2.2 Tc - Ta

This module normalizes the temperature raster image according to the air temperature during the image acquisition.



Required Input Parameters

- **Input layer:** Thermal raster
- **Air temperature:** Constant air temperature measure at flight time

Output Parameters

- **Output layer:** Thermal raster output including normalized temperature.

4.3 LiDAR Processing

LiDAR data is often provided as a number of tiles or flight lines. Depending on computers capacity, in some cases it might be convenient to merge flight files into a single file, or to divide data in overlapping tiles with an appropriate size.

Due to ThermoLiDAR software uses SPDLib suite for LiDAR data manipulation, data has to be converted into the native SPDLib file format. This means that LiDAR data must be converted into **Sorted Pulse Data format (SPD)** from **LAS** files, which is the file standard for the interchange of LASer data recommended by the American Society for Photogrammetry and Remote Sensing (**ASPRS**). Noisy data can be eventually removed before being processed. Once the SPD files have been obtained, ground points are classified and then point heights relative to the ground are inferred. Ground and no-ground points are interpolated to generate DTM, DSM and CHM. From height information a range of metrics mainly applied to forestry applications -but not only- can be derived. Here below this workflow is depicted.

4.3.1 Convert between formats

Since LiDAR processing modules make use of SPDLib tools, the first step is to convert the input dataset into SPD files. There are two types of SPD files, non-indexed and indexed. A format translation module has been included in QGIS to this purpose. The module allows the conversion between different formats and it is also used to re-project data.

Supported file formats

ThermoLiDAR plugin supports SPD/UPD, LAS and a wide range of ASCII formats but the current level of support is not intended to encompass all the available formats.

SPD/UPD

The *Sorted Pulse Data (SPD)* file format (Bunting, 2013) has been designed specifically for the storage of LiDAR waveform and discrete return data acquired by **TLS**, **ALS** and space borne systems, and includes support for multiple wavelengths within a single file. The SPD format also supports 2D spatial indexing of the pulses, and the *Unsorted Pulse Data (UPD)* files are SPD files without a spatial index.

LAS

The LASer (**LAS**) file format supported is aligned with the LAS v1.2 specifications. LAS 1.2 files are exported through the **LibLAS** library so as with the importer only discrete return data are supported.

ASCII

The ASCII format requires a `schema` written in XML to be supplied. The parser expects a single return per line and the resulting pulses will only contain a single return. The following are some examples of schemas for common ASCII formats.

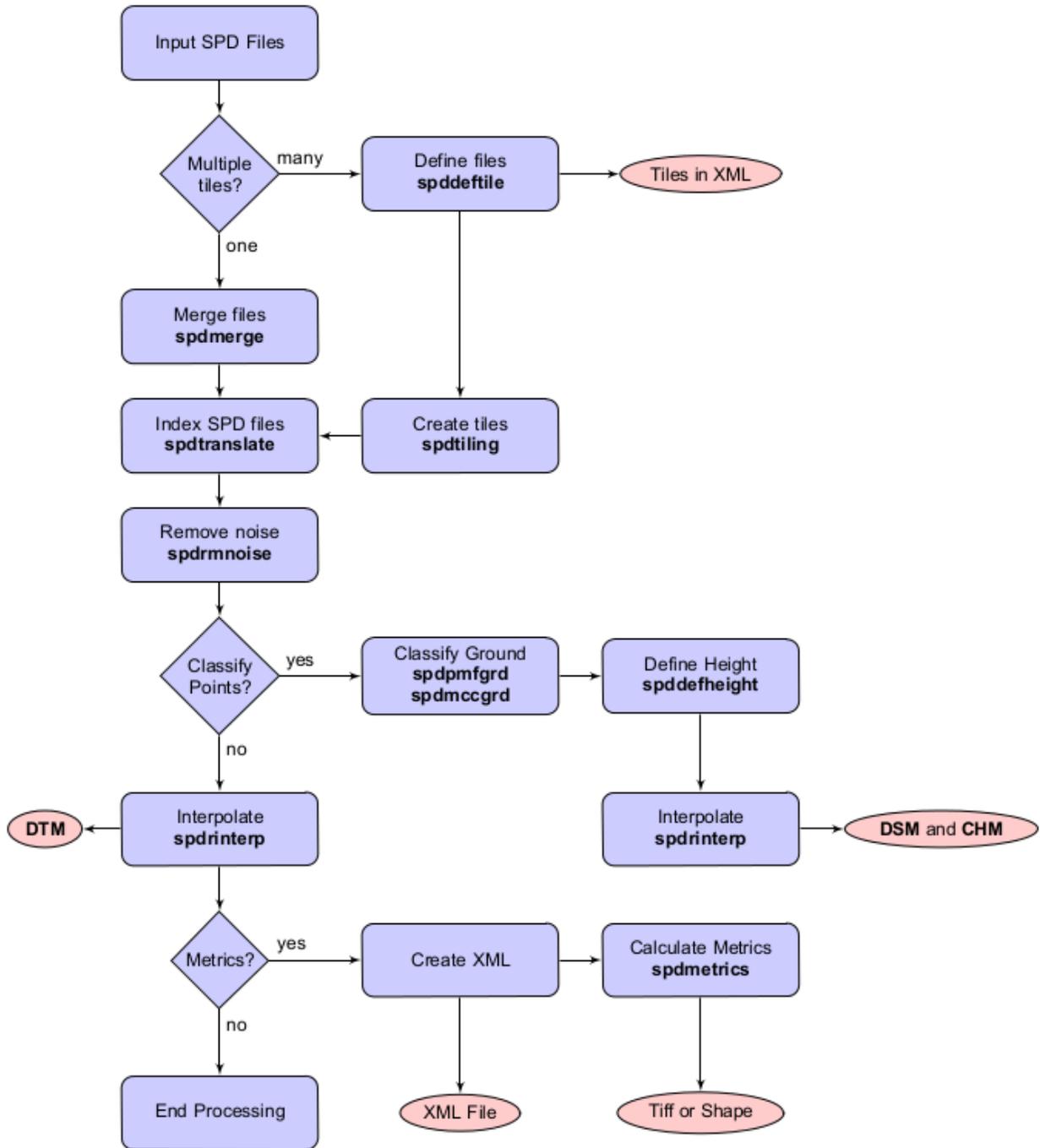


Figure 4.1: LiDAR workflow using the SPDLib toolset. In this case, SPD format files are supplied as input. Names of SPDLib commands are highlighted in bold font (i.e. **spdmerge**, **spdtranslate**, **spddefiles**, etc.). Pink boxes represent output products. [Bunting2013]

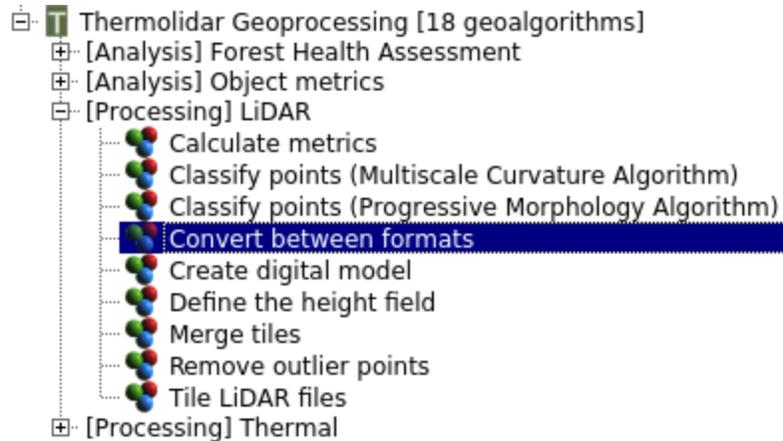


Figure 4.2: The module for format conversion is located in the *LiDAR* submenu of the ThermoLiDAR Toolbox

A schema for the PTS format:

```
<?xml version="1.0" encoding="UTF-8" ?>
<line delimiter=" " comment="#" ignorelines="0" >
  <field name="X" type="spd_double" index="0" />
  <field name="Y" type="spd_double" index="1" />
  <field name="Z" type="spd_float" index="2" />
  <field name="AMPLITUDE_RETURN" type="spd_uint" index="3" />
  <field name="RED" type="spd_uint" index="4" />
  <field name="GREEN" type="spd_uint" index="5" />
  <field name="BLUE" type="spd_uint" index="6" />
</line>
```

A schema for the XYZ format:

```
<?xml version="1.0" encoding="UTF-8" ?>
<line delimiter="," comment="#" ignorelines="0" >
  <field name="X" type="spd_double" index="0" />
  <field name="Y" type="spd_double" index="1" />
  <field name="Z" type="spd_float" index="2" />
  <field name="AMPLITUDE_RETURN" type="spd_uint" index="3" />
</line>
```

Reprojection

It is possible to define the projection of the SPD file explicitly using the *input* and *output projection*. Both options expect a text file containing the **WKT** (*Well Known Text*) string representing the projection information. In order to change projection, the input projection option is not required, but if it is known, then it should be advantageous to provide it.

Memory Requirements

When converting to an UPD very little memory is required as only a few pulses are held in memory at any one time, this is because no sorting of the pulses is required. On the other hand when generating an SPD file the data needs to be spatially sorted. Therefore, the whole file is read into memory and sorted into the spatial grid before being written output the file. This requires enough memory to store the whole dataset and index data structure in memory. If memory is not sufficient to complete this operation the file needs to be split into blocks to fit into memory.

The option to select splitting the file to disk while building the SPD file is *temporal path* which is the path and base file name while the tiles will be written. The *num rows* parameter specifies the number of rows of the final SPD file that will be written to each temporary tile. Note that the tile height in is *binsize x num rows* (in units the data is projected). The *num columns* option maybe set where datasets are very wide such that the tiles are not the full width of the output file. Whether this option is used, the final SPD file will result in a non-sequential rather than a sequential file. This means the data on disk is not order left-to-right top-to-bottom or top-left to bottom-right, which has some performance benefits. Obviously, allowing the SPD file to be built in stages is slower but once completed it is faster to make spatial queries within the file. Besides, other processing steps (i.e., classification and interpolations) can be applied to the whole file with only relatively small memory requirements.

Parameters

Required Input Parameters

- **Input:** SPD file that contains the LiDAR point clouds.
- **Index:** The location used to index the pulses and points (required):
 - **FIRST_RETURN**
 - **LAST_RETURN**
- **Input Format:** Format of the input file (Default SPD).
 - **SPD:** SPD input format with or without spatial index
 - **ASCII:** ASCII input format
 - **LAS/LAZ:** Both zipped or normal LAS input format
 - **LASNP:** LAS input without pulse information
- **Output Format:** Format of the output file (Default SPD).
 - **SPD:** SPD output format
 - **UPD:** SPD output format without spatial index
 - **ASCII:** ASCII output format
 - **LAS:** LAS output format
 - **LAZ:** Zipped LAS output format

Optional Input Parameters

- **Binsize:** (*float*) Bin size for SPD file index (Default 1)
- **Schema:** (*string*) schema for the format of the ASCII file being imported
- **Input Projection:** (*string*) WKT string representing the projection of the input file
- **Output Projection:** (*string*) WKT string representing the projection of the output file
- **Num. Columns:** (*integer*) Number of columns within a block (Default 0) - Note values greater than 1 result in a non-sequential SPD file.
- **Num. Rows:** (*integer*) Number of rows within a block (Default 25)
- **Temporal Path:** (*string*) Path where temporary files can be written to.

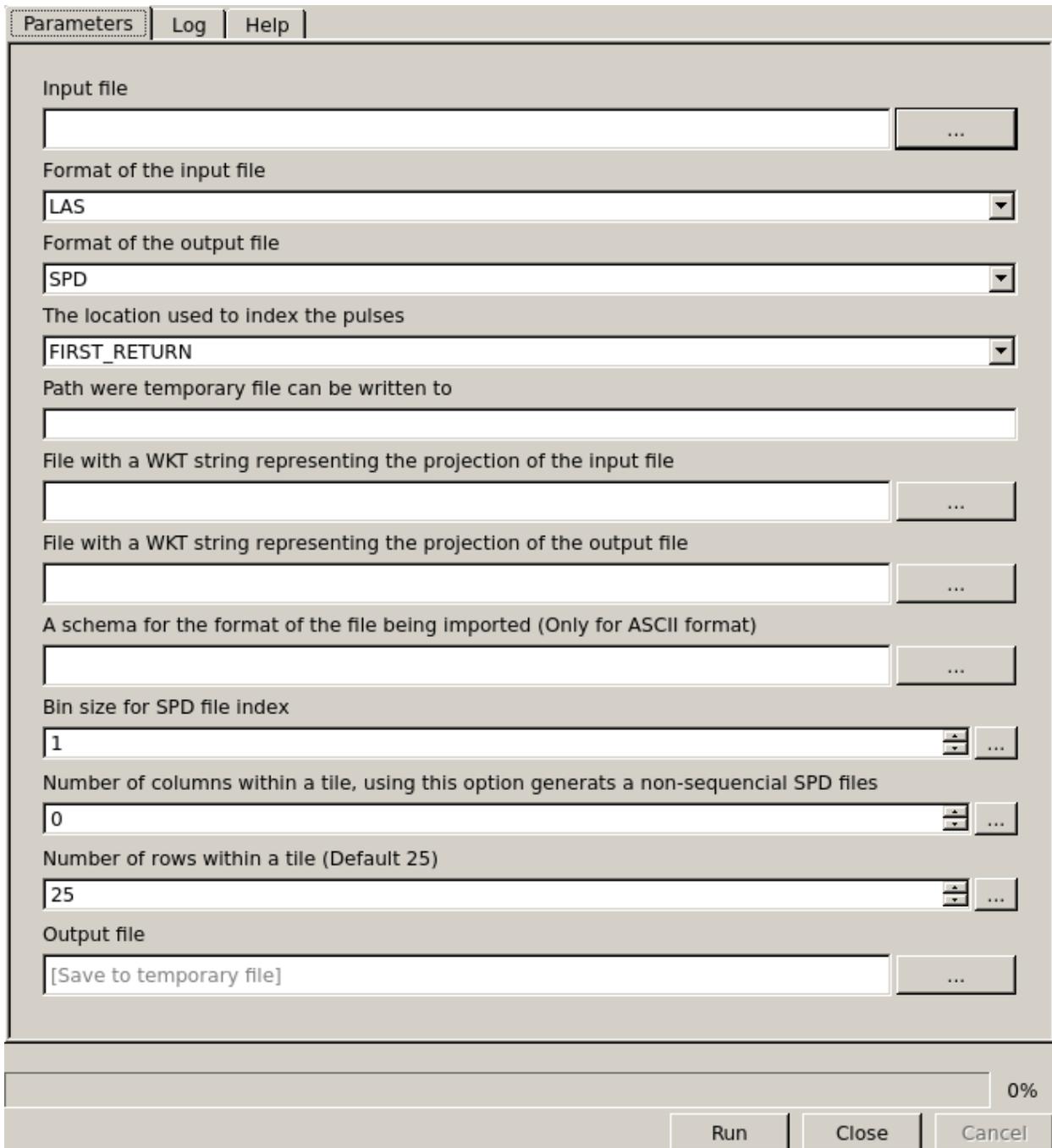


Figure 4.3: Interface to convert between different data formats

Output Parameters

- **Output:** The output SPD file

4.3.2 Merge files

In some situations it might be convenient to merge various files into a single SPD file. The merging module merges compatible files into a single non-indexed SPD file. It is possible to provide the projection information of the output file and input files if known.

This module allows displaying classes and returns IDs of the input files with list **returns IDs** and **list classes** options, respectively. The **ignore checks** option forces the input files to be merged in case files come from different sources or have different bin sizes.

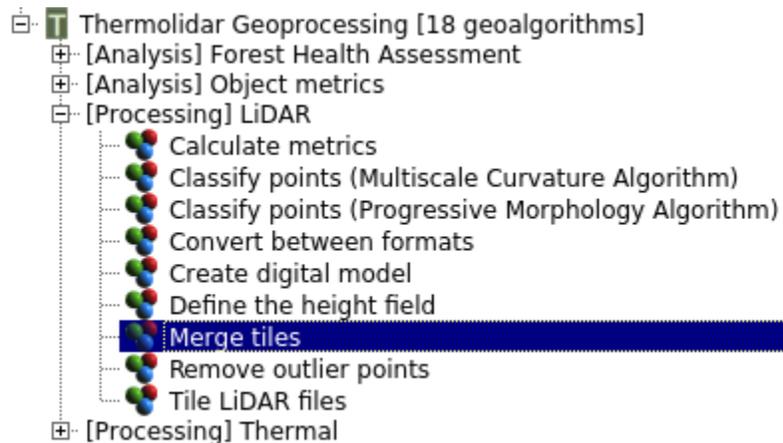


Figure 4.4: The module for merging files is located in the *LiDAR* submenu of the ThermoLiDAR toolbox

Parameters

Required Input Parameters

- **Input:** SPD file that contains the LiDAR point clouds (accept multiple files separated by comas).
- **Index:** The location used to index the pulses and points (required):
 - **FIRST_RETURN**
 - **LAST_RETURN**
- **Input Format:** Format of the input file (Default SPD).
 - **SPD:** SPD input format with or without spatial index
 - **ASCII:** ASCII input format
 - **LAS/LAZ:** Both zipped or normal LAS input format
 - **LASNP:** LAS input without pulse information

Parameters | Log | Help

Input file
 ...

Set source ID for each input file
 ▾

When indexing the file use the extent of the input file as the minimum extent of the output file
 ▾

Ignore checks between input files to ensure compatibility
 ▾

Format of the input file
 ▾

The location used to index the pulses
 ▾

A schema for the format of the file being imported (Only for ASCII format)
 ...

File with a WKT string representing the projection of the input file
 ...

File with a WKT string representing the projection of the output file
 ...

Lists the return IDs for the files listed (accepted multiple values separated by comas)

Lists the classes for the files listed (accepted multiple values separated by comas)

Output file
 ...

0%

Run Close Cancel

Figure 4.5: Interface to merge compatible files into a single non-indexed SPD file

Optional Input Parameters

- **List Returns IDs:** (*list of files*) Lists the return IDs for the files listed (accept multiple files separated by comas).
- **List Classes:** (*list of files*) Lists the classes for the files listed (accept multiple files separated by comas).
- **Keep Extent:** (*Yes/No*) Use the extent of the input files as the minimum extent of the output file when indexing the file.
- **Source ID:** (*Yes/No*) Set source ID for each input file
- **Ignore Checks:** (*Yes/No*) Ignore checks between input files to ensure compatibility
- **Schema:** (*string*) schema for the format of the ASCII file being imported
- **Input Projection:** (*string*) WKT string representing the projection of the input file
- **Output Projection:** (*string*) WKT string representing the projection of the output file

Output Parameters

- **Output:** The output SPD file

4.3.3 Split data into tiles

LiDAR data is supplied as flight lines or tiles with different shapes and sizes. It is always useful to divide laser data into equally sized square tiles, though. This helps to store, manage and access data easily. Single tiles should meet memory requirements in order to reduce computational times, which determines the maximum size of each file given an average point density. Overlapping zones between tiles help to prevent border errors and guarantee continuous raster models.

ThermoLiDAR has a built-in tool to create tiles given **tiles size** and the **overlap**. Output tiles are saved into the **output path** (this includes path and *prefix*) and are named as `_rowYYcolXX.spd`, being YY and XX the number of row and column of the corresponding tile. Tiles definition is stored in an output XML file (**output xml**) containing their column, row, extent and core extent (tile extent without overlap).

```
<tiles columns="23" overlap="50" rows="16" xmax="256500" xmin="239343.510" xtilesize="750" ymax="707246.440"
  <tile col="1" corexmax="240093.510" corexmin="239343.510" coreymax="695996.440" coreymin="695246.440"
  <tile col="2" corexmax="240843.510" corexmin="240093.510" coreymax="695996.440" coreymin="695246.440"
  ...
  <tile col="22" corexmax="255843.510" corexmin="255093.510" coreymax="707246.440" coreymin="706496.440"
  <tile col="23" corexmax="256593.510" corexmin="255843.510" coreymax="707246.440" coreymin="706496.440"
</tiles>
```

The module supports to create single tiles (**SINGLE** option) by supplying **row** and **column**, or to generate the complete set of tiles (**ALL** option).

The module also creates an auxiliary file listing the input LiDAR which can be eventually kept (**keep file list**) once the module has finished. It might happen that some tiles are empty; in that case those files can be removed enabling the **delete tiles** option.

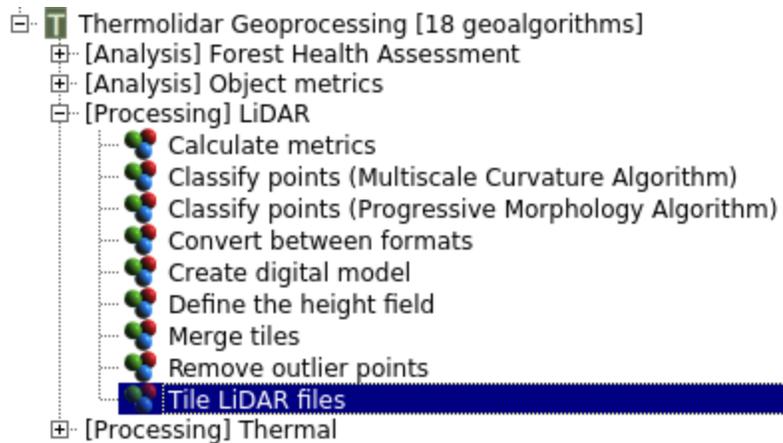


Figure 4.6: The module for tiling LiDAR data is within the *LiDAR* submenu of the ThermoLiDAR toolbox

Parameters

Required Input Parameters

- **Input:** SPD file that contains the LiDAR point clouds (accept multiple files separated by comas).

Optional Input Parameters

- **Extract Tiles: Where to extract**
 - **ALL:** Create all tiles
 - **SINGLE:** Extract an individual tile given its row and column
- **Delete Tiles:** (*Yes/No*) If shapefile exists delete it and then run
- **Keep File List:** (*Yes/No*) Keep auxiliary file containing a list of the input files to be tiled
- **Tile Size:** (*float*) Size (in units of the coordinate system) of the square tiles (Default: 1000)
- **Overlap Size:** (*float*) Size (in units of coordinate systems) of the overlap for tiles (Default 100)
- **Column:** (*integer*) The column of the tile to be extracted (only with single)
- **Row:** (*integer*) The row of the tile to be extracted (only with single)
- **Output Path:** (*string*) The output XML file that contains the tiles definition

Output Parameters

- **Output XML:** The output XML file that contains the tiles definition

4.3.4 Remove Noise

Many factors may introduce errors in LiDAR point clouds, including water vapour clouds, multipath, poor equipment calibration, or even a flock of birds. In order to avoid further errors and artefacts in final digital models and poor assess of height metrics, those points have to be removed.

Parameters | Log | Help

Input file
[Empty text box] [...]

Keep auxilliar spd file list
[No] [v]

Extract all tiles or single tiles
[ALL] [v]

Remove tiles which have no data
[Yes] [v]

Size (in units of coordinate systems) of the overlap for tiles
[100] [up/down] [...]

Size (in units of coordinate systems) of the tiles
[1000] [up/down] [...]

The column of the tile to be extracted (SINGLE tile)
[0] [up/down] [...]

The row of the tile to be extracted (SINGLE tile)
[0] [up/down] [...]

The output base path for the tiles
[Empty text box] [...]

Output XML file defining the tiles
[Save to temporary file] [...]

0%

Run Close Cancel

Figure 4.7: Interface for tiling a set of SPD files

This module removes vertical noise from LiDAR datasets by means of three different. Upper and lower **absolute thresholds** will clip the file to fit these values. **Relative threshold** will remove, for each bin within a SPD file, points outside the upper and lower values relative to the median height. Whilst **global threshold** will use the whole SPD file to calculate the median height and remove points relative to it.

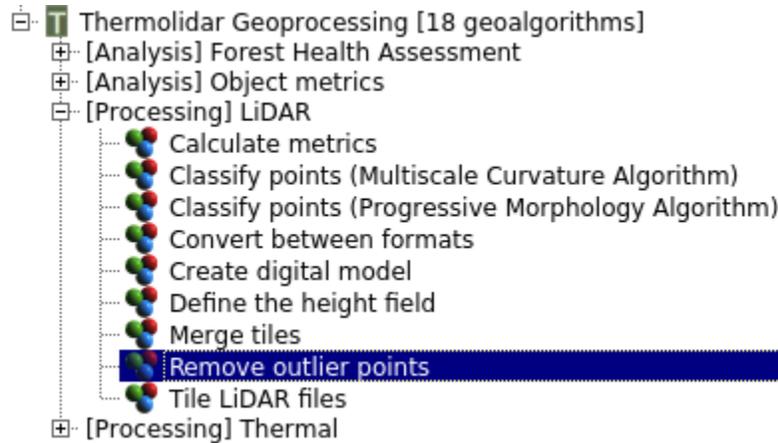


Figure 4.8: The module for removing noise from data is located in the *LiDAR* submenu of the ThermoLiDAR toolbox

Parameters

Required Input Parameters

- **Input:** SPD file that contains the LiDAR point clouds.

Optional Input Parameters

- **Global Rel. Upper Threshold:** (*float*) Global relative to median upper threshold for returns which are to be removed
- **Global Rel. Lower Threshold:** (*float*) Global relative to median lower threshold for returns which are to be removed
- **Relative Upper Threshold:** (*float*) Relative to median upper threshold for returns which are to be removed
- **Relative Lower Threshold:** (*float*) Relative to median lower threshold for returns which are to be removed
- **Absolute Upper Threshold:** (*float*) Absolute upper threshold for returns which are to be removed
- **Absolute Lower Threshold:** (*float*) Absolute lower threshold for returns which are to be removed
- **Column:** (*integer*) The column of the tile to be extracted (only with single)
- **Row:** (*integer*) The row of the tile to be extracted (only with single)

Output Parameters

- **Output:** The output SPD file without noise

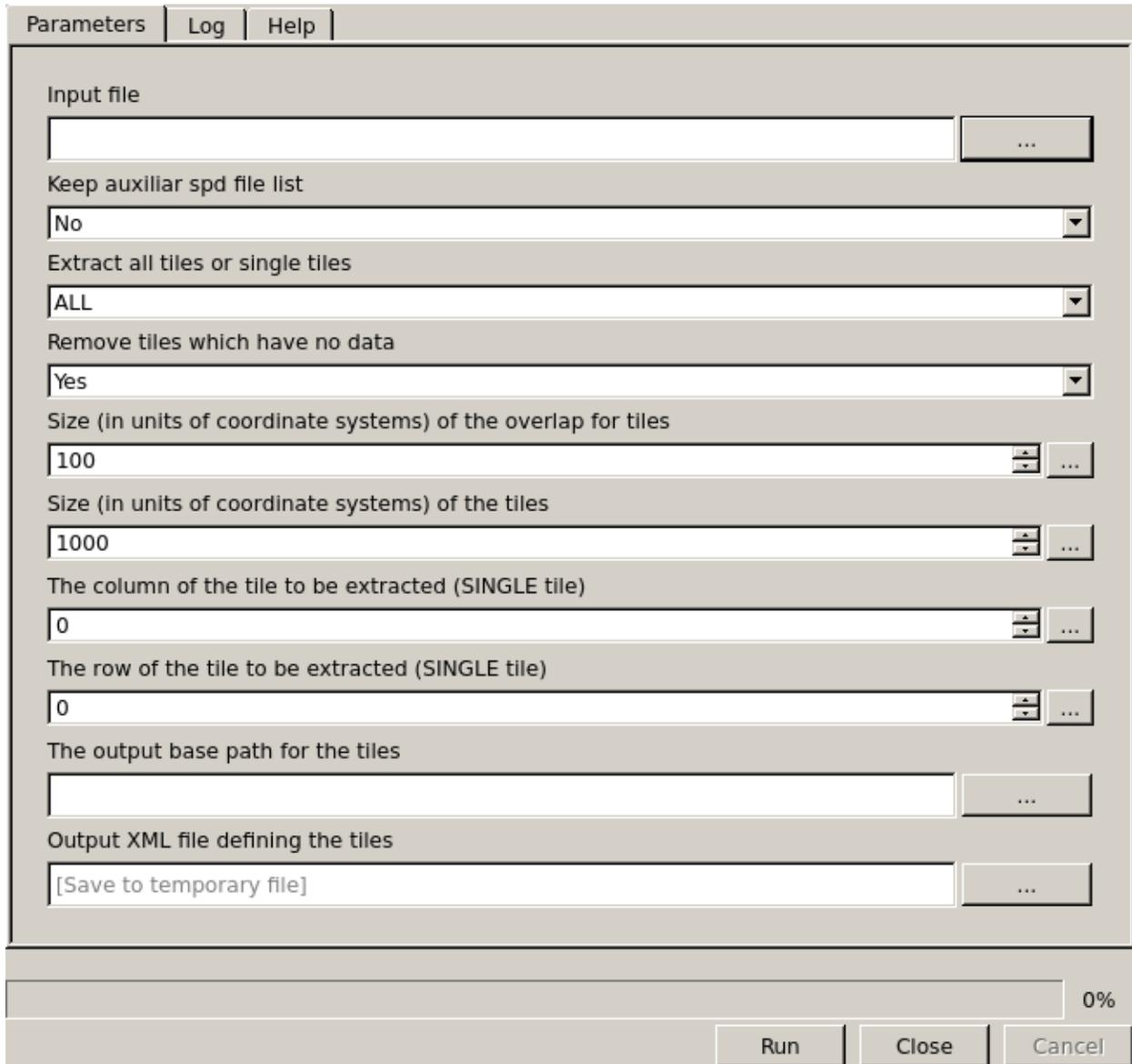


Figure 4.9: Interface for tiling a set of SPD files

4.3.5 Classify Ground Returns

Two different classification algorithms have been implemented into the plugin. These algorithms also called filters allow the classification of the LiDAR points, identifying which point belong to the ground. The filters implement the *Progressive Morphology* (Zhang et al., 2003; [Zhang2003]) and the *Multiscale Curvature* (Evans and Hudak, 2007; [EvansHudak2007]) methodologies.

Progressive Morphology filter

To classify ground returns an implementation of the *Progressive Morphology* algorithm (**PMF**) has been provided. The algorithm QGIS interface has only three options to be set. Under most circumstances the default parameters will be fit the purpose and it is recommended to use the simplest parameters configuration given by default.

The **class** option allows to apply the filter to particular classes (i.e., if ground returns have been already classified but they need tidying up).

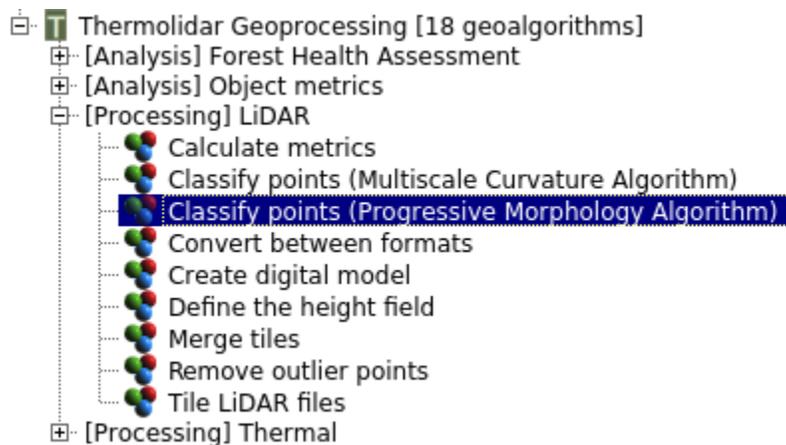


Figure 4.10: The module that implements the PMF algorithm to classify ground is within the *LiDAR* submenu of the ThermoLiDAR toolbox

Parameters

Required Input Parameters

- **Input:** SPD file that contains the LiDAR point clouds.

Optional Input Parameters

- **Binsize:** (*float*) Bin size for SPD file index (Default 1)
- **Class:** (*integer*) Only use points of particular class
- **Ground Threshold:** (*float*) Threshold for deviation from identified ground surface for classifying the ground returns (Default 0.3)
- **Median Filter:** (*integer*) Size of the median filter (half size i.e., 3x3 is 1) (Default 2)
- **No Median:** (Yes/No) Do not run a median filter on generated surface
- **Max. Elevation:** (*float*) Maximum elevation difference threshold (Default 5)
- **Initial Elevation:** (*float*) Initial elevation difference threshold (Default 0.3)

Parameters	Log	Help
Input file		
<input type="text"/>	<input type="button" value="..."/>	
Only use points of particular class		
<input type="text" value="0"/>	<input type="button" value="..."/>	
Generated an image rather than classifying points (useful for parameter selection)		
<input type="text" value="[Not selected]"/>	<input type="button" value="..."/>	
Bin size for SPD file index		
<input type="text" value="0"/>	<input type="button" value="..."/>	
Size (in bins) of the overlap between processing blocks		
<input type="text" value="10"/>	<input type="button" value="..."/>	
Number of columns within a tile (Generates a non-sequential SPD file)		
<input type="text" value="0"/>	<input type="button" value="..."/>	
Number of rows within a tile		
<input type="text" value="100"/>	<input type="button" value="..."/>	
Size of the median filter (half size i.e., 3x3 is; 5x5 is 2...)		
<input type="text" value="2"/>	<input type="button" value="..."/>	
Do not run a median filter on generated surface (before classifying ground point or export)		
<input type="text" value="No"/>	<input type="button" value="..."/>	
Threshold for deviation from identified ground surface for classifying ground returns		
<input type="text" value="0.300000"/>	<input type="button" value="..."/>	
Maximum elevation difference threshold		
<input type="text" value="5"/>	<input type="button" value="..."/>	
Initial elevation difference threshold		
<input type="text" value="0.300000"/>	<input type="button" value="..."/>	
Slope parameter related to terrain		
<input type="text" value="0.300000"/>	<input type="button" value="..."/>	
Maximum size of the filter		
<input type="text" value="7"/>	<input type="button" value="..."/>	
Initial size of the filter (half size i.e., 3x3 is; 5x5 is 2...)		
<input type="text" value="1"/>	<input type="button" value="..."/>	
Output file		
<input type="text" value="[Save to temporary file]"/>	<input type="button" value="..."/>	
0%		
<input type="button" value="Run"/>		
<input type="button" value="Close"/>		
<input type="button" value="Cancel"/>		

Figure 4.11: Interface of the *Progressive Morphology Algorithm* to classify ground points

- **Slope:** (*float*) Slope parameter related to terrain (Default 0.3)
- **Max. Filter:** (*float*) Maximum size of the filter (Default 7)
- **Initial Filter:** (*float*) Initial size of the filter (half size i.e., 3x3 is 1) (Default 1)
- **Overlap:** (*integer*) Size (in bins) of the overlap between processing blocks (Default 10)
- **Num. Columns:** (*integer*) Number of columns within a block (Default 0) - Note values greater than 1 result in a non-sequential SPD file.
- **Num. Rows:** (*integer*) Number of rows within a block (Default 25)

Output Parameters

- **Output:** The output SPD file containing classification

Multiscale Curvature filter

The plugin integrates an implementation of the *Multiscale Curvature* algorithm (MCC). As before, the QGIS interface has only three options to be set: input, output and class argument. Under most circumstances default parameters for the algorithm will be fit for purpose, but be careful that the bin size used within SPD is not too large as the processing will be at this resolution.

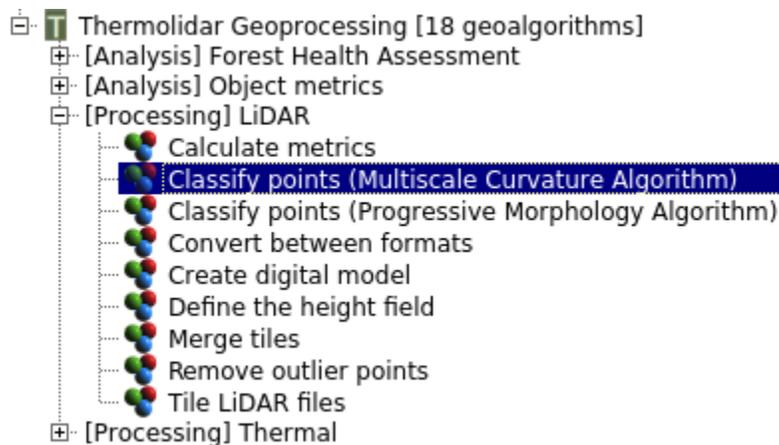


Figure 4.12: The module that implements the MCC algorithm to classify ground is within the *LiDAR* submenu of the ThermoLiDAR toolbox

Parameters

Required Input Parameters

- **Input:** SPD file that contains the LiDAR point clouds.

Optional Input Parameters

- **Binsize:** (*float*) Bin size for SPD file index (Default 1)
- **Class:** (*integer*) Only use points of particular class
- **Median:** (*Yes/No*) Use a median filter to smooth the generated raster instead of a (mean) averaging filter.

Parameters	Log	Help
Input file	<input type="text"/>	...
Only use points of particular class	<input type="text" value="0"/>	...
Bin size for SPD file index	<input type="text" value="0"/>	...
Size (in bins) of the overlap between processing blocks	<input type="text" value="10"/>	...
Number of columns within a tile (Generates a non-sequential SPD file)	<input type="text" value="0"/>	...
Number of rows within a tile	<input type="text" value="100"/>	...
Use only multiple return pulses to calculate the amount of change between iterations	<input type="text" value="No"/>	
Median filter to smooth the generated raster instead of an average filter	<input type="text" value="No"/>	
The threshold for the	<input type="text" value="0.100000"/>	...
The size of the smoothing filter (half size i.e., 3x3 is 1)	<input type="text" value="1"/>	...
The number of points used for the TPS interpolation	<input type="text" value="16"/>	...
Maximum search radius for the TPS interpolation	<input type="text" value="20"/>	...
Iteration step curvature tolerance parameter	<input type="text" value="0.500000"/>	...
Minimum curvature tolerance parameter	<input type="text" value="0.100000"/>	...
Initial curvature tolerance paramete	<input type="text" value="1"/>	...
Gap between increments in scale	<input type="text" value="0.500000"/>	...
The number of scales below the init scale to be used	<input type="text" value="1"/>	...
The number of scales above the init scale to be used	<input type="text" value="1"/>	...
Initial processing scale (Usually the native resolution of the data)	<input type="text" value="0"/>	...
Output file	<input type="text" value="[Save to temporary file]"/>	...

4.3. LiDAR Processing 55

0%

Run Close Cancel

- **Filter Size:** (*integer*) The size of the smoothing filter (half size i.e., 3x3 is 1; Default = 1)
- **Num. Points Tps:** (*integer*) The number of points used for the TPS interpolation (Default = 16)
- **Max. Radius Tps:** (*float*) Maximum search radius for the TPS interpolation (Default = 20)
- **Step Curve Tolerance:** (*float*) Iteration step curvature tolerance parameter (Default = 0.5)
- **Min. Curve Tolerance:** (*float*) Minimum curvature tolerance parameter (Default = 0.1)
- **Initial Curve Tolerance:** (*float*) Initial curvature tolerance parameter (Default = 1)
- **Scale Gaps:** (*float*) Gap between increments in scale (Default = 0.5)
- **Num. Scales Below:** (*integer*) The number of scales below the init scale to be used (Default = 1)
- **Num. Scales Above:** (*integer*) The number of scales above the init scale to be used (Default = 1)
- **Initial Scale:** (*float*) Initial processing scale, this is usually the native resolution of the data.
- **Max. Elevation Threshold:** (*float*) Maximum elevation difference threshold (Default 5)
- **Initial Elevation Threshold:** (*float*) Initial elevation difference threshold (Default 0.3)
- **Slope:** (*float*) Slope parameter related to terrain (Default 0.3)
- **Max. Filter:** (*float*) Maximum size of the filter (Default 7)
- **Initial Filter:** (*float*) Initial size of the filter (half size i.e., 3x3 is 1) (Default 1)
- **Overlap:** (*integer*) Size (in bins) of the overlap between processing blocks (Default 10)
- **Num. Columns:** (*integer*) Number of columns within a block (Default 0) - Note values greater than 1 result in a non-sequential SPD file.
- **Num. Rows:** (*integer*) Number of rows within a block (Default 25)

Output Parameters

- **Output:** The output SPD file containing classification

Filter points depending on class

The **class** option applies the filter to returns of a particular class (i.e., ground returns). This represents a way to improve the ground return classification is to combine more than one filtering algorithm to take advantage of their particular strengths and weaknesses. In fact, a particularly useful combination is to first run the PMF algorithm where a thick slice is taken (e.g., 1 or 2 metres above the raster surface) and then the MCC is applied to find the ground returns (setting the **class** option to 3).

It can be also useful for TLS as it can take a thick slice with MCC algorithm and then use the PMF algorithm to tidy that result up to get a good overall ground classification.

4.3.6 Define Height field

SPD files supports both elevation corresponding to a vertical datum and an above-ground height for each discrete return. Before data can be used for generating a Canopy Height Model (CHM) or any height related metric, height field has to be populated. This can be done in two ways. The simplest way is to use a DTM of the same resolution as the SPD file bin size. The disadvantage of using a DTM is that it is if the DTM is not accurate it can introduce some artefacts. Using this method the only parameters are the input files, both LiDAR file and the DTM, and an output file. The raster DTM needs to the same resolution as the SPD grid and it can be any raster format supported by the GDAL library.

The other option is to interpolate a value for each point generating a continuous surface and reducing any artefacts. The recommended approach for the interpolation is to use the Natural Neighbour method, as demonstrated by Bater and Coops (2009; [BaterCoops2009]).

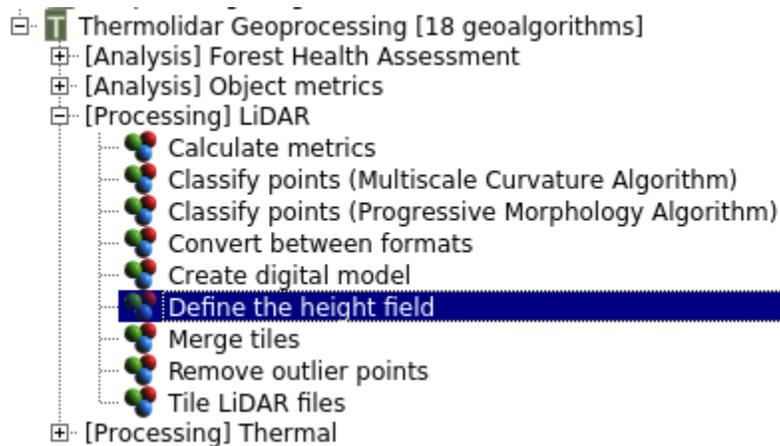


Figure 4.14: The module that defines the height above ground is located in the *LiDAR8 submenu of the ThermoLiDAR toolbox

Parameters

Required Input Parameters

- **Input:** SPD file that contains the LiDAR point clouds.

Optional Input Parameters

- **Elevation:** (*raster*) The input elevation image
- **Binsize:** (*float*) Bin size for SPD file index (Default 1)
- **Interpolator: Different interpolation methods to choose from**
 - **Natural Neighbor**
 - **Nearest Neighbor**
 - **TIN Plate**
- **Thin:** (*Yes/No*) Thin the point cloud when interpolating
- **Thin Resolution:** (*float*) Resolution of the grid used to thin the point cloud
- **Point per Bin:** (*integer*) The number of point allowed within a grid cell following thinning
- **Overlap:** (*integer*) Size (in bins) of the overlap between processing blocks (Default 10)
- **Num. Columns:** (*integer*) Number of columns within a block (Default 0) - Note values greater than 1 result in a non-sequential SPD file.
- **Num. Rows:** (*integer*) Number of rows within a block (Default 25)

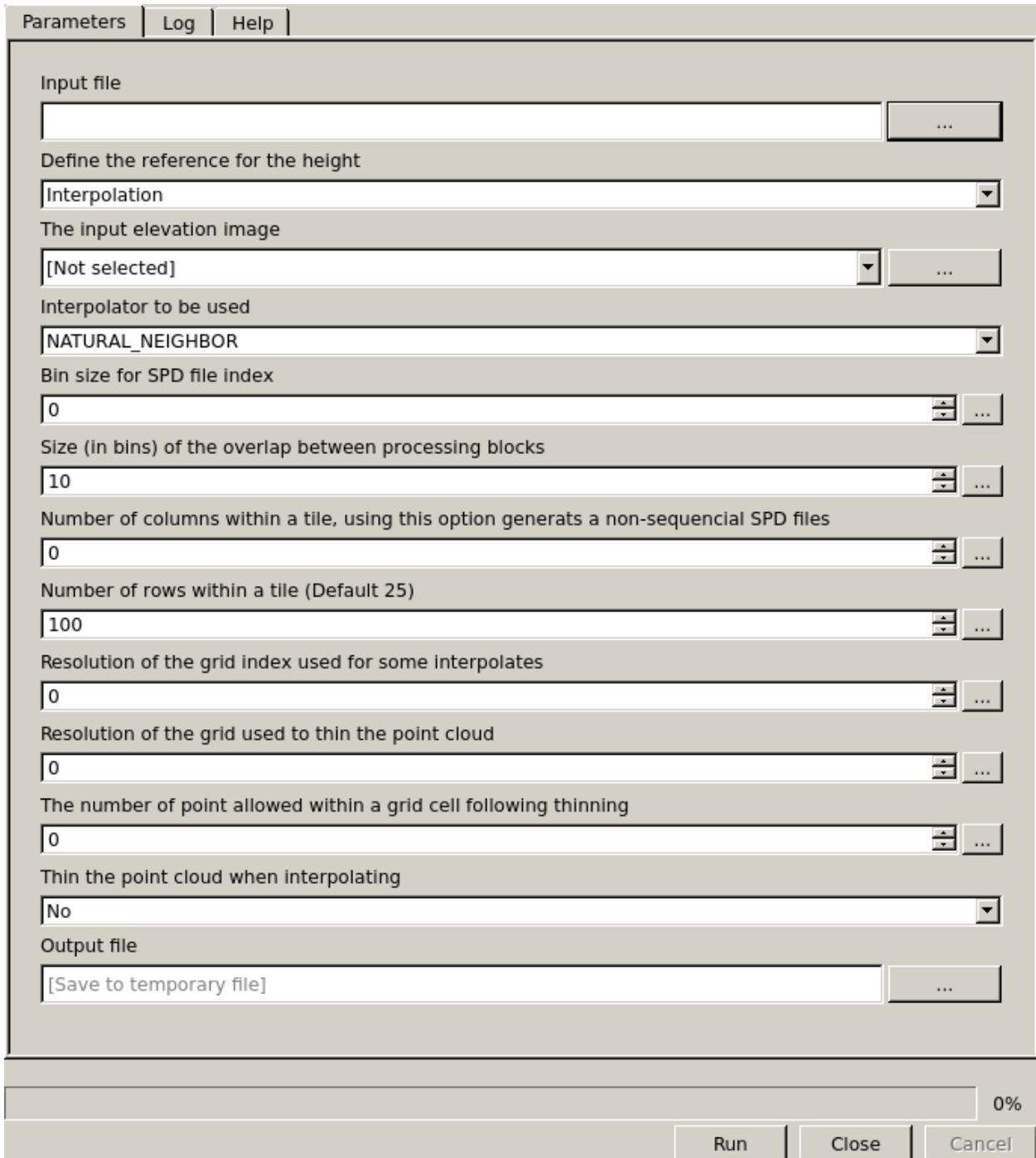


Figure 4.15: Interface of the module to define points heights from the ground

Output Parameters

- **Output:** The output SPD file

4.3.7 Interpolation Module

The most common products that can be created from a LiDAR dataset are Digital Terrain Models (DTMs), Digital Surface Models (DSMs) and Canopy Height Models (CHMs). To produce those products, it is necessary to interpolate a raster surface from the classified ground returns and top surface points. Create Digital Model within the ThermoLiDAR plugin permits to generate these products by choosing **model** option.

A key parameter is the output raster resolution or **binsize** which needs to be a multiple of the SPD input file spatial index. Different interpolators can be selected with the **interpolator** option. This module supports many raster formats by means of the GDAL library.

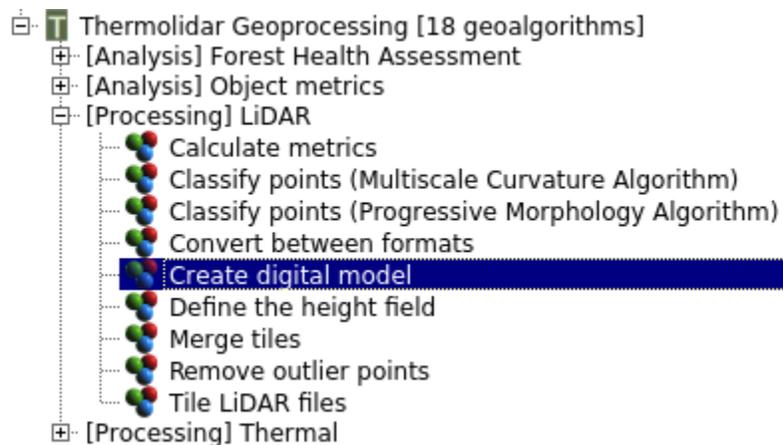


Figure 4.16: The interpolation module is in the *LiDAR* submenu of the ThermoLiDAR toolbox

Parameters

Required Input Parameters

- **Input:** SPD file that contains the LiDAR point clouds.
- **MODEL:**
 - **DTM:** Digital Terrain Model
 - **MDS:** Digital Surface Model
 - **CHM:** Canopy Height Model

Optional Input Parameters

- **Binsize:** (*float*) Bin size for SPD file index (Default 1)
- **Interpolator:** Different interpolation methods to choose from
 - **Natural Neighbor**

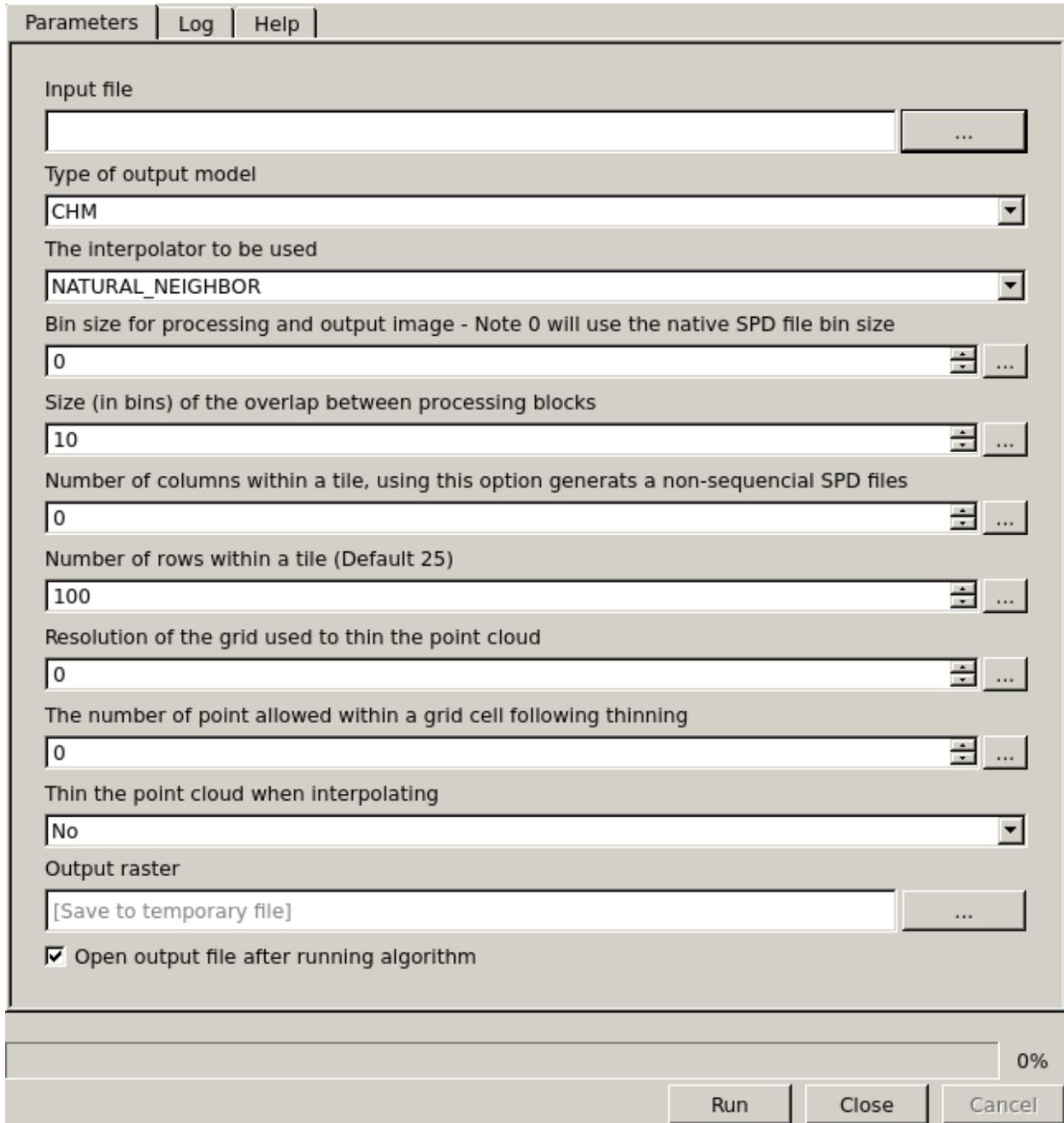


Figure 4.17: Interface to interpolate data

- **Nearest Neighbor**
- **TIN Plate**
- **Overlap:** (*integer*) Size (in bins) of the overlap between processing blocks (Default 10)
- **Num. Columns:** (*integer*) Number of columns within a block (Default 0) - Note values greater than 1 result in a non-sequential SPD file.
- **Num. Rows:** (*integer*) Number of rows within a block (Default 25)

Output Parameters

- **Output:** The raster file containing the interpolated model

Examples

Digital Surface Models (DSMs) are easily done by setting **model** to **DSM**. The module will perform an interpolation of the elevation information of all the points of the LiDAR file. The result is a surface representing the ground and all the objects attached to it as can be seen in figure below.

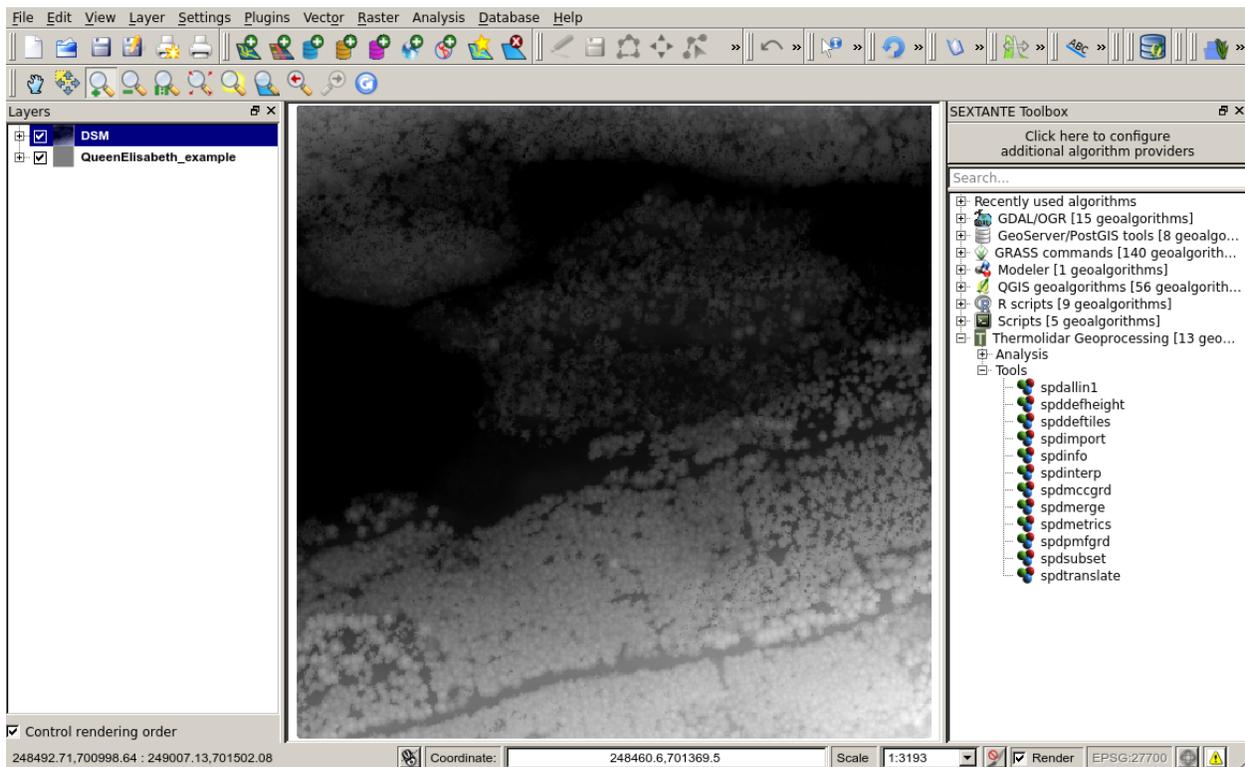


Figure 4.18: Example of visualization in grey scales of a DSM (1m resolution) generated with the ThermoLiDAR plugin

In case ground returns have been classified, then to interpolate elevation information of ground points will generate a Digital Terrain Model (DTM). For this purpose **model** has to be set to **DTM**. The output raster represents the bare ground surface (see next figure).

In a forest environment, those points not classified as ground are commonly classified as vegetation. To interpolate the height above ground information of vegetation points produces a Canopy Height Model (CHM). In this case, the

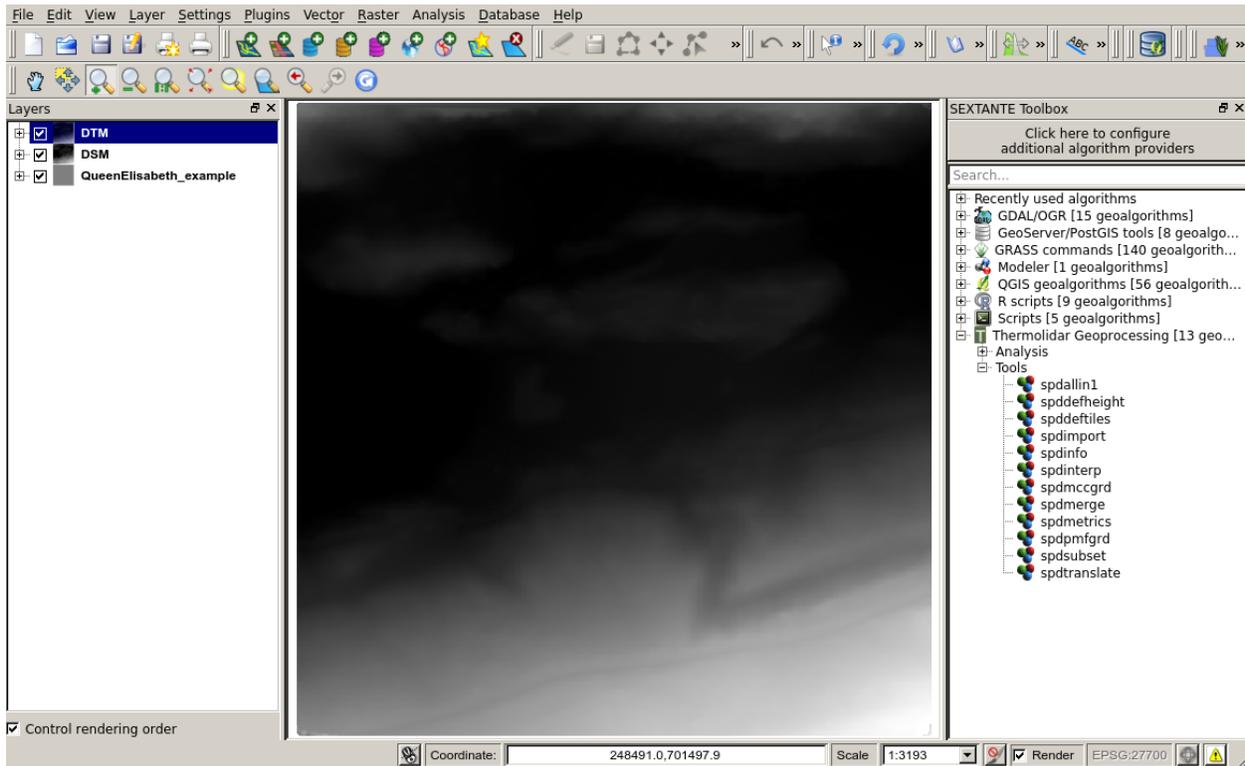


Figure 4.19: Example of visualization in grey scales of a DTM (1m resolution) generated with the ThermoLiDAR plugin

model option is set to **CHM**. The result is raster where canopies are perfectly depicted and ground (see figure below).

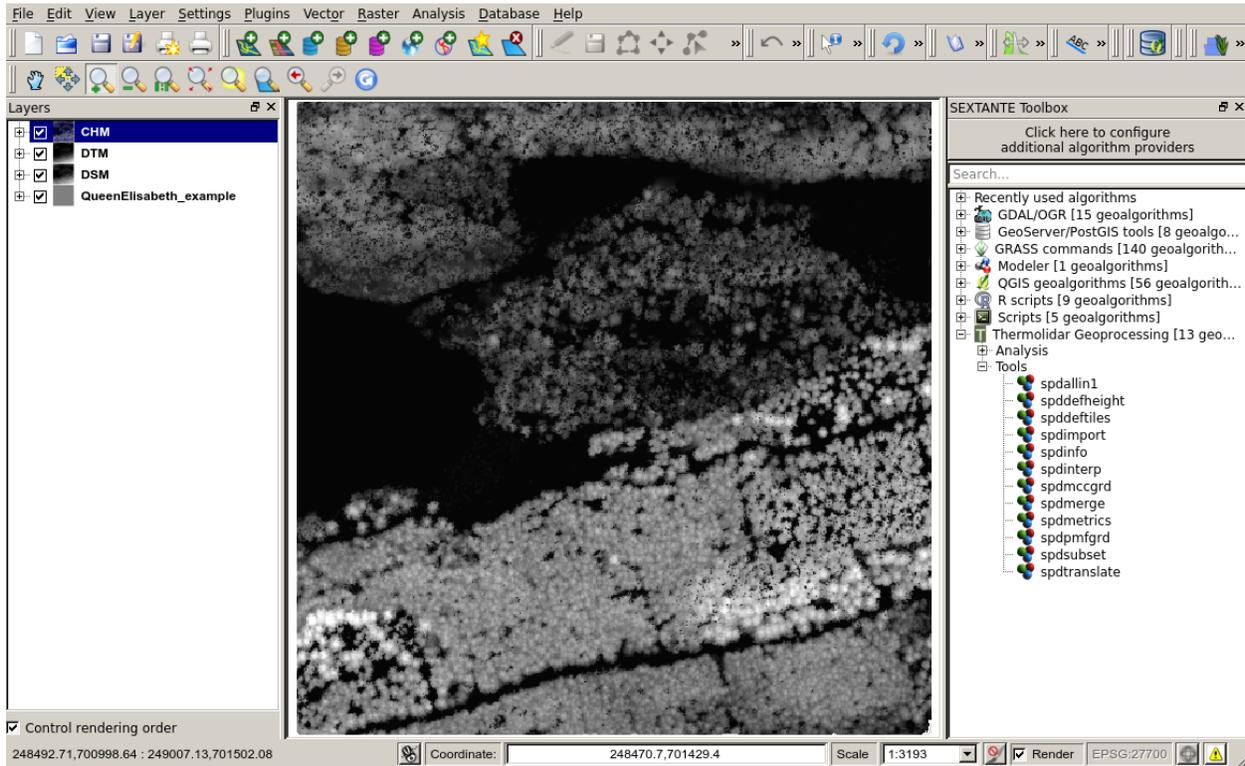


Figure 4.20: Example of visualization in grey scales of a CHM (1m resolution) generated with the ThermoLiDAR plugin

4.3.8 Generate metrics

ThermoLiDAR plugin is able to calculate different metrics at the same time if they are defined in an XML file. Metrics can be simple statistical moments, percentiles of point heights, or even count ratios. Mathematical operators can also be applied to either other metrics or operators, allowing a wider range of LiDAR metrics to be derived.

The XML file has to be defined a priori with a hierarchical list of metrics and operators. Within the **metrics** tags a list of metrics can be provided by the metric tag. Within each metric the **field** attribute is used to name the raster band or vector attribute.

The module supports different output data formats, that is, **raster** (all GDAL formats) and **vector**. **Raster** option extends the output to the entire input file, assessing the metrics for each pixel the final raster output and creating as many bands as metrics have been defined within the XML file. **Vector** option requires an input shapefile containing polygon entities. The output shapefile database will be populated with the metrics computed inside the polygons.

Parameters

Required Input Parameters

- **Input:** SPD file that contains the LiDAR point clouds.
- **Metrics:** (*file*) XML file containing the metrics template

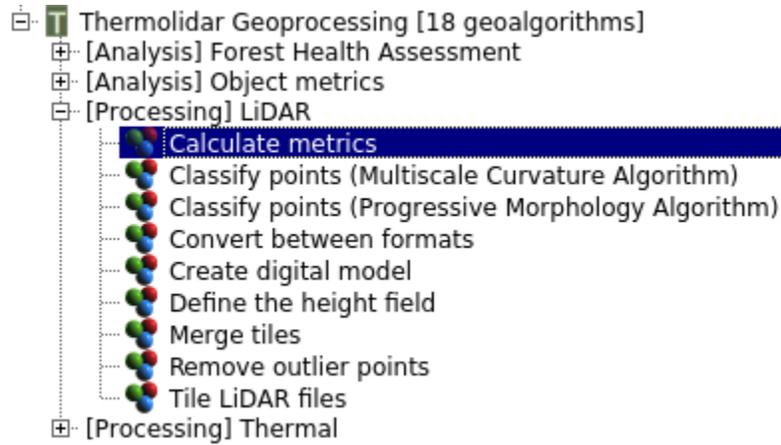


Figure 4.21: The module to generate metrics is located in the *LiDAR* submenu of the ThermoLiDAR toolbox

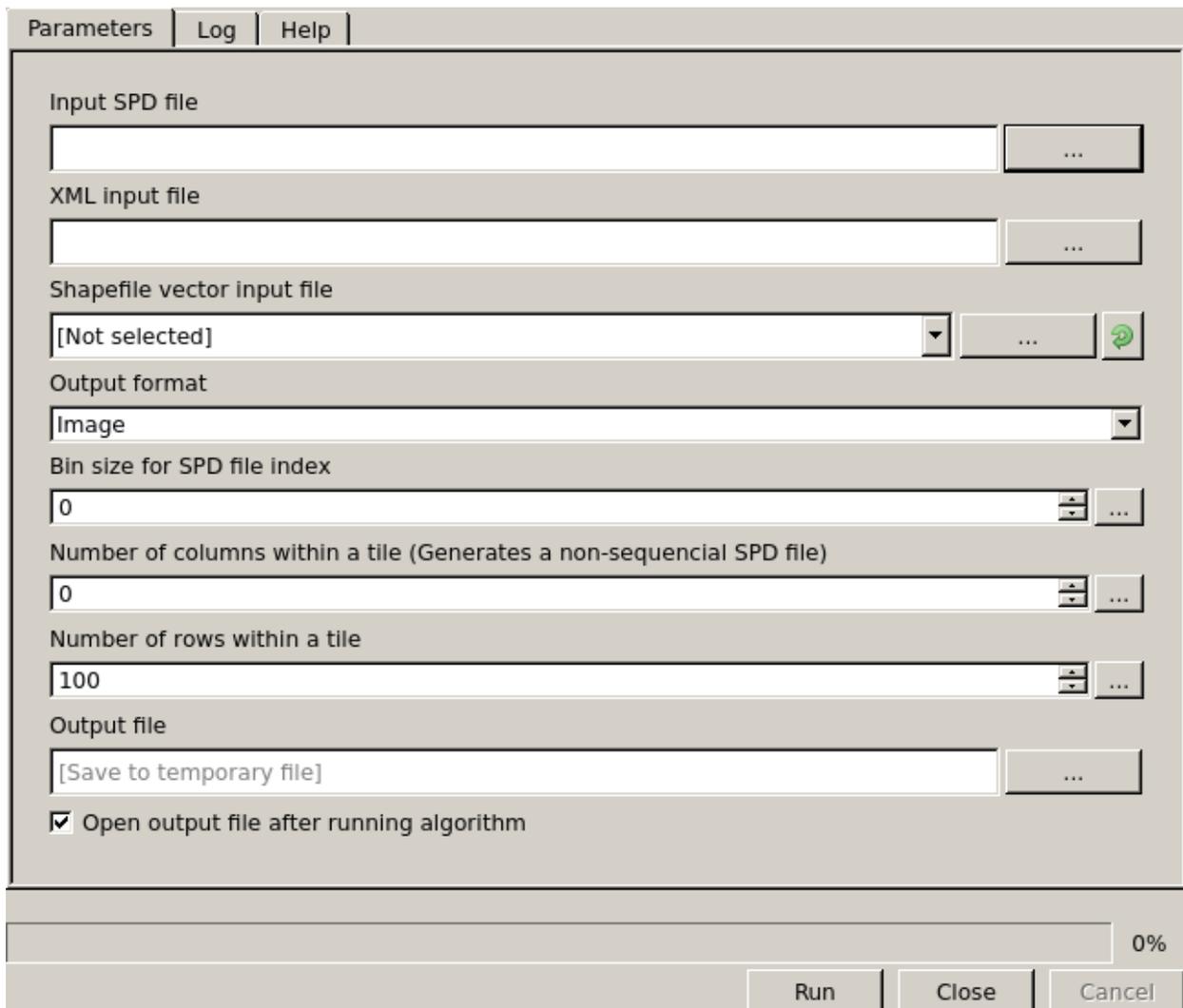


Figure 4.22: Interface to calculate metrics

- **Output Data Format:**
 - **Image:** Raster output
 - **Vector:** Vector output

Required Input Parameters

- **Binsize:** (*float*) Bin size for processing and the resolution of the output image. Note: 0 will use the native SPD file bin size
- **Num. Columns:** (*integer*) Number of columns within a block (Default 0) - Note values greater than 1 result in a non-sequential SPD file.
- **Num. Rows:** (*integer*) Number of rows within a block (Default 25)
- **Vector File:** (*shapefile*) Input shapefile (only with vector output).

Output Parameters

- **Output:** The raster file containing the interpolated model

How to define metrics?

Here below is an example of an SPD metrics XML file template containing the percentage of not-ground returns –as a mathematical operation of the number of not-ground returns and the total number of returns– maximum height, the average and median heights, the canopy cover and the height 95th percentile:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--
  Description:
    XML File for execution within SPDLib
    This file contains a template for the
    metrics XML interface.

  Created by Roberto Antolin on Thu Apr 24 16:33:36 2014.
-->

<spdlib:metrics xmlns:spdlib="http://www.spdlib.org/xml/" />
  <spdlib:metric metric="percentage" field="CoverRts" />
  <spdlib:metric metric="numreturnsheight" field="Out_Name" return="All" class="NotGrd" lowthreshold="0.2" />
  <spdlib:metric metric="numreturnsheight" field="Out_Name" return="All" class="All" />
</spdlib:metric>
<spdlib:metric metric="maxheight" field="MaxH" return="All" class="NotGrd" lowthreshold="0.2" />
<spdlib:metric metric="meanheight" field="MeanH" return="All" class="NotGrd" lowthreshold="0.2" />
<spdlib:metric metric="medianheight" field="MedianH" return="All" class="NotGrd" lowthreshold="0.2" />
<spdlib:metric metric="percentileheight" field="95thPerH" percentile="95" return="All" class="NotGrd" />
</spdlib:metrics>
```

The figure below shows the Height 95th percentile computed for the same dataset than the previous examples:

Metrics can be defined manually by typing each XML file. This allows the user to adequate the XML file to particular purposes by selecting both metrics and their options. However, the user can generate the XML file automatically by means of the module *Create metrics XML file*. The module offers the possibility to create the file with a single metric selected from a list. However, the list also accept the options **ALL**, **FOREST** and **PERCENTILES**:

- **ALL:** Selects all the metrics listed in the option list.

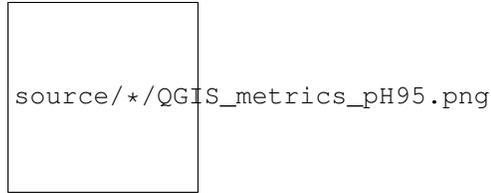


Figure 4.23: Height 95th percentile computed into a raster image (10m resolution)

- **FOREST:** Includes some metrics that are commonly used in forest applications. This metrics includes percentiles from 99th to 50th, *groundCover*, *canopyCover*, *maxHeight* and *meanHeight*.
- **PERCENTILES:** Includes all percentiles from 99th to 10th.

In case other metrics than those available in the list are needed, the user can specify them by writing their names separated by ; (semi-colon, and without spaces) in the field labelled with **string format**. This could be an example:

pH99;pH98;pH95;pH90;pH80;maxHeight;canopyCover;NumberReturnsNoGround

In this case, the module will omit any selection made in the first input parameter and it will create the XML file with the metrics supplied by the user. In case the string any metric is misspelled, the module will inform the user of the mistake and omit the metric. The file will be created with the metrics that are correctly written.

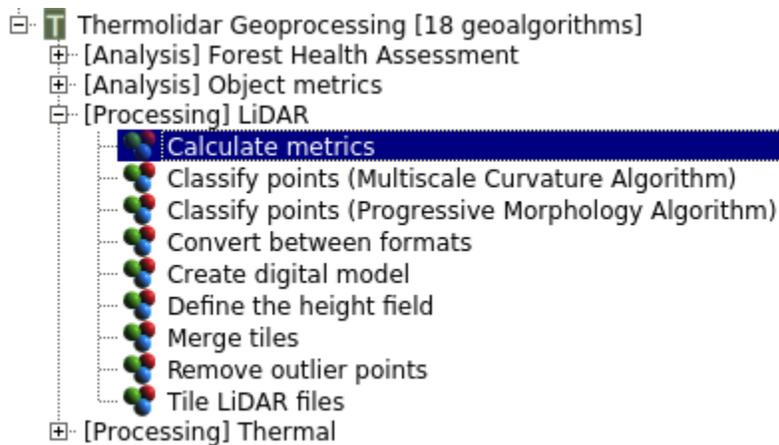


Figure 4.24: The module to generate metrics XML file is located in the *LiDAR* submenu of the ThermoLiDAR toolbox

Parameters

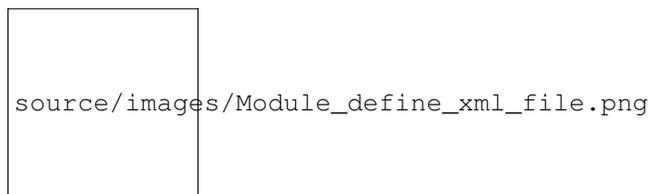


Figure 4.25: Interface of the module to generate XML file containing metrics

Optional Input Parameters

- **List of metrics:** Metrics to be used
- **String of metrics:** Metrics to be used

Output Parameters

- **Output:** The output XML file containing metric definitions

4.4 Statistics

To measure tree biological and physical properties (e.g. dominant height, mean diameter, stem number, basal area, timber volume, etc...) throughout an entire woodlands is impossible. For this reason, only a few sample plots are usually measured in field in order to related them to canopy height metrics derived from LiDAR data. These relationships are then used to estimate and extend those characteristics to the area covered by LiDAR data and create forest inventory cartography. Two different models have been implemented in the ThermoLiDAR plug-in.

4.4.1 Stepwise Multivariate Regression Model

The Stepwise Multivariate Regression model was firstly introduce by Naesset (1997a, 1997b) to estimates tree heights ([Naesset1997a]) and volumes ([Naesset1997b]). The methodology assumes that stands have been previously classified before sample plots are measured. Plots must have the same size and have to be regularly distributed throughout the study area. Height metrics derived from LiDAR have to be calculated within each single sample plot (see [Generate metrics](#)) excluding points lower than 2 meters, so that stones and shrubs are avoided. Metrics have to include (Naesset 2002; [Naesset2002]):

- Quantiles corresponding to the 0th, 5th, 10th, 15th, ..., 90th, 95th, 98th percentiles of the distribution.
- The maximum values
- The mean values
- The coefficients of variation
- Measures of canopy density

Naesset definition of canopy density is considered as the proportions of the first echo laser hits above 0th, 10th, ..., 90th percentiles of the first echo height distribution to total number of first echos.

For each sample plot a logarithmic regression equation is formulated:

$$Y = \beta_0 h_0^{\beta_1} h_{10}^{\beta_2} \dots h_{90}^{\beta_{11}} h_{max}^{\beta_{12}} h_{mean}^{\beta_{13}} d_{10}^{\beta_{14}} \dots d_{90}^{\beta_{23}} \dots$$

which, in linear form is expressed as:

$$\begin{aligned} \ln Y = & \ln \beta_0 + \beta_1 \ln h_0 + \beta_2 \ln h_{10} \dots + \beta_{11} \ln h_{90} + \beta_{12} + \\ & + \ln h_{max} + \beta_{13} \ln h_{mean} + \beta_{14} \ln d_{10} + \dots + \beta_{23} \ln d_{90}^{\beta_{23}} + \dots \end{aligned}$$

Where Y are the field values (**dependent variable**); h_i are height percentiles; h_{max} and h_{mean} are maximum and mean height, respectively; and d_j are Naesset canopy densities.

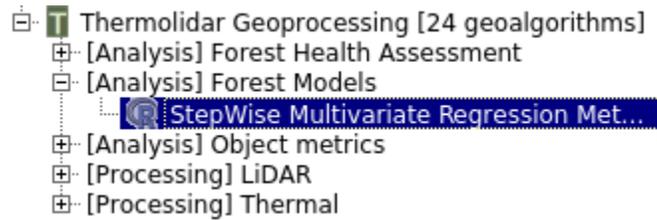


Figure 4.26: The module to calibrate the the forest model is located in *[Analysis] Forest Model* submenu of the ThermoLiDAR toolbox

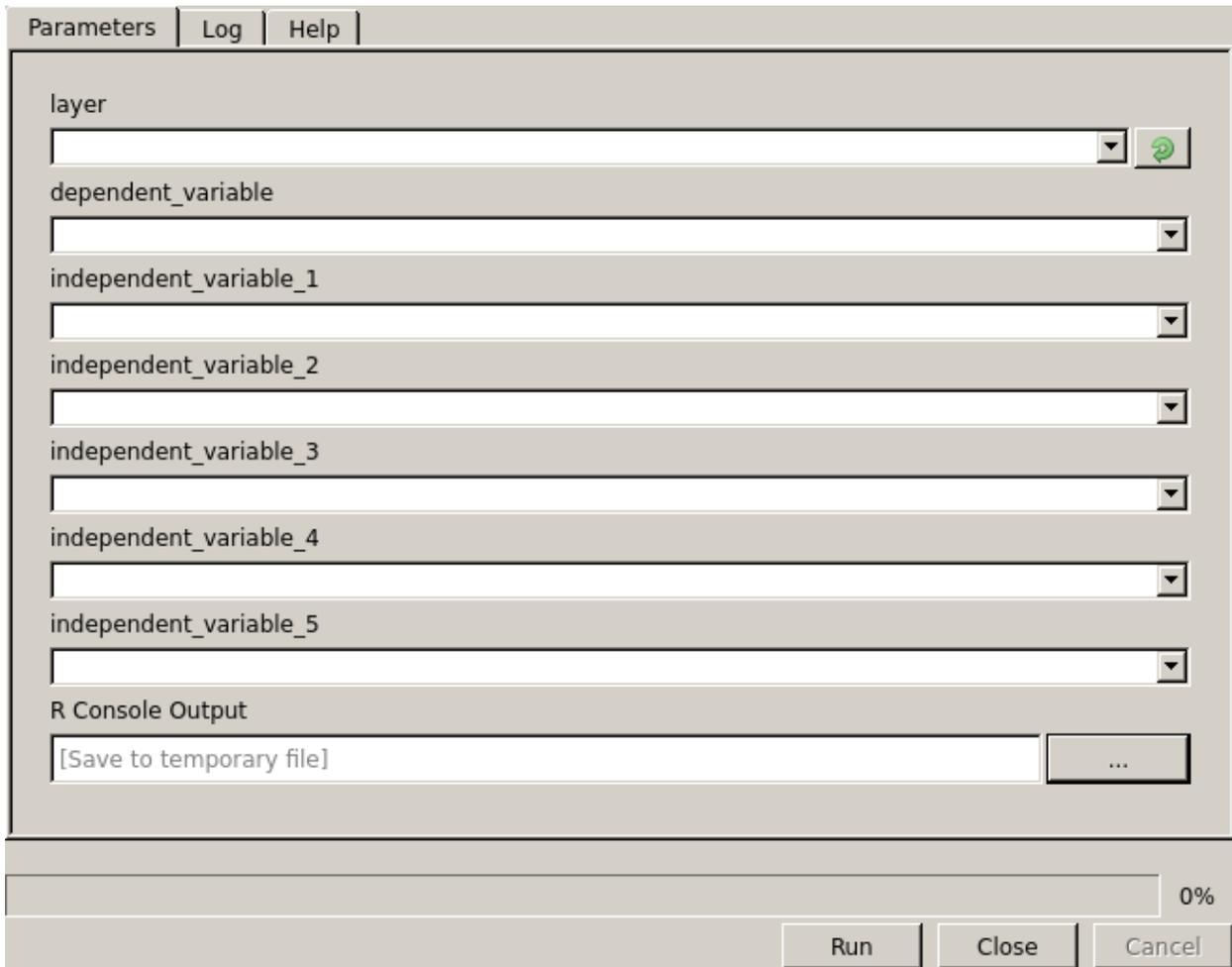


Figure 4.27: Interface to calculate calibrate the forest model

Parameters

Required Input Parameters

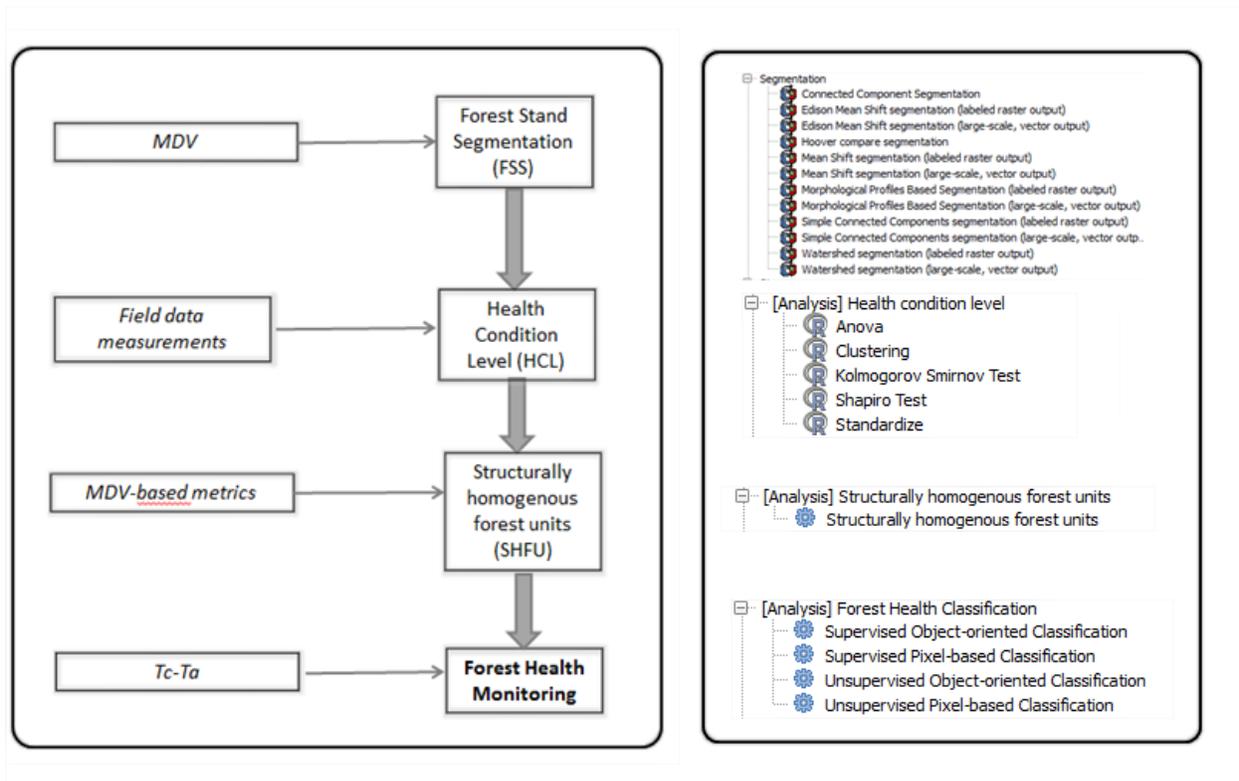
- **Layer:** Shapefile layer that contains the LiDAR metrics.
- **Dependent Variable:** Vector field containing the observations of the predictable variable
- **Independent Variable i:** Vector field containing the LiDAR metrics that will be use to characterized the model

Output Parameters

- **Output:** (html) File containing the final report of the multivariate regression

4.5 Forest Health Assessment

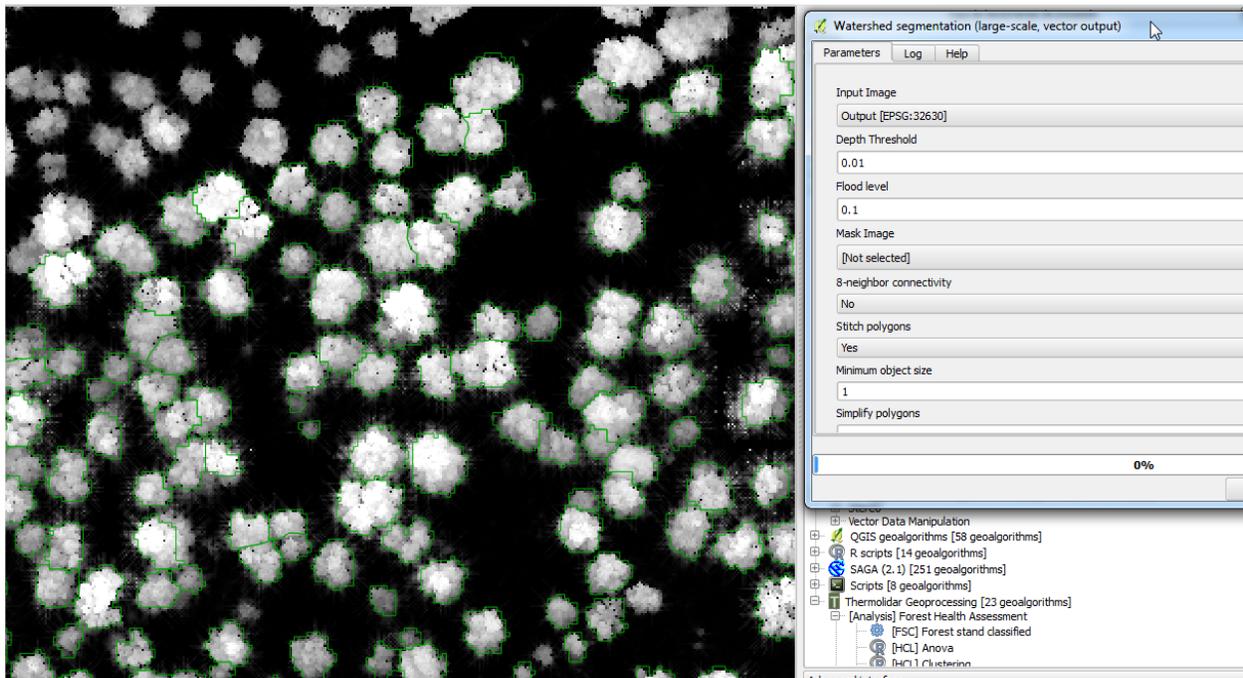
Forest Health assessment module consists of four different tools: Forest Stand Segmentation (FSS), Health Condition Level (HCL), Structurally Homogeneous Forest Units (SHFU) and Forest Health Monitoring (FHM). The following figure summarizes the main structure of this module and the input required throughout the process.



4.5.1 Forest stands segmentation

Within OD tools, users are willing to choose between developing a semi-automatic segmentation and using a pre-defined object feature. Segmentation tools are based on algorithms that segment an image into areas of connected

pixels based on the pixel DN value. ThermoLiDAR image segmentation tools will be based on region growing algorithms. The basic approach of a region growing algorithm is to start from a seed region (typically one or more pixels) that are considered to be inside the object to be segmented. The pixels neighbouring this region are evaluated to determine if they should also be considered part of the object. If so, they are added to the region and the process continues as long as new pixels are added to the region. Region growing algorithms vary depending on the criteria used to decide whether a pixel should be included in the region or not, the type connectivity used to determine neighbours, and the strategy used to visit neighbouring pixels. Image segmentation is a crucial step within the object-based remote sensing information retrieval process. As a step prior to classification the quality assessment of the segmentation result is of fundamental significance for the recognition process as well as for choosing the appropriate approach and parameters for a given segmentation task. Alternatively, user could be interested on using a pre-defined object feature. This object feature could be a segmentation shape file provided from other source or any other land cover mapping. Also, the user can use a pre-defined regular object, defining the size of the square to be used previously.



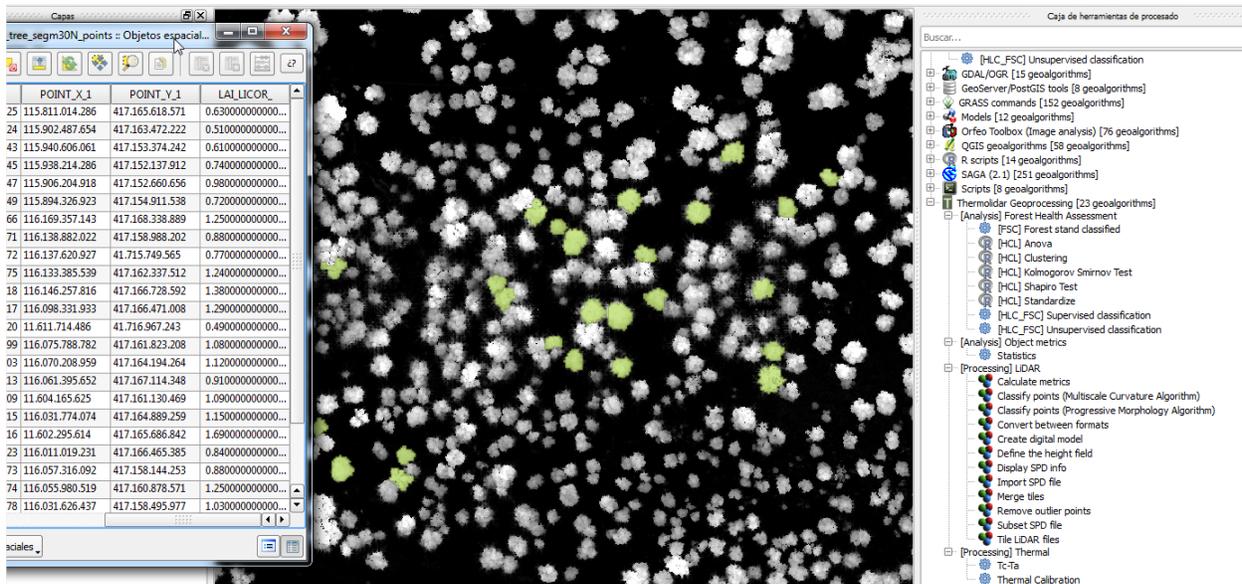
4.5.2 Forest Condition Levels

The most critical part in applying forest health condition indicators is the user's accuracy defining forest degradation levels. Besides user's training another critical factor is to select under analysis a robust physiological indicator and to carry out an accurate field measurements campaign.

Potential physiological indicators of forest decline such as pigment concentration, photosynthesis, respiration and transpiration rate holds great potential to shed light on the mechanisms and processes that occur as a result of drought stress. In the short-term, climate can change the physiological conditions of the forest resulting in acute damage, but chronic exposure usually results in cumulative effects on physiological process. These factors effects on the plants light reactions or enzymatic functions and increased respiration from reparative activities. Gradual decreases in photosynthesis, stomatal conductance, carbon fixation, water use efficiency, resistance to insect and cold resistance were found in most of trees which are very typical symptom of stress conditions

Long-term exposure of water stress to a combination of high light levels and high temperatures causes a depression of photosynthesis and photosystem II efficiency that is not easily reversed, even for water-stress-resistant forest species. The decrease in the photochemical efficiency of photosystem II (Φ PSII) is related to the conversion of violaxanthin to antheraxanthin and zeaxanthin produced by an increase in harmless non-radiative energy dissipation (qN) and

providing photo-protection from oxidative damage. One of the most widely physiological indicator applied in the analysis of long-term effect on forest health condition is de Leaf Area Index (LAI). The following is an example of the statistical analysis performs on LAI values measured from an Oak forest inventoried in the framework of THERMOLIDAR project.



4.5.3 Structurally Homogeneous Forest Units

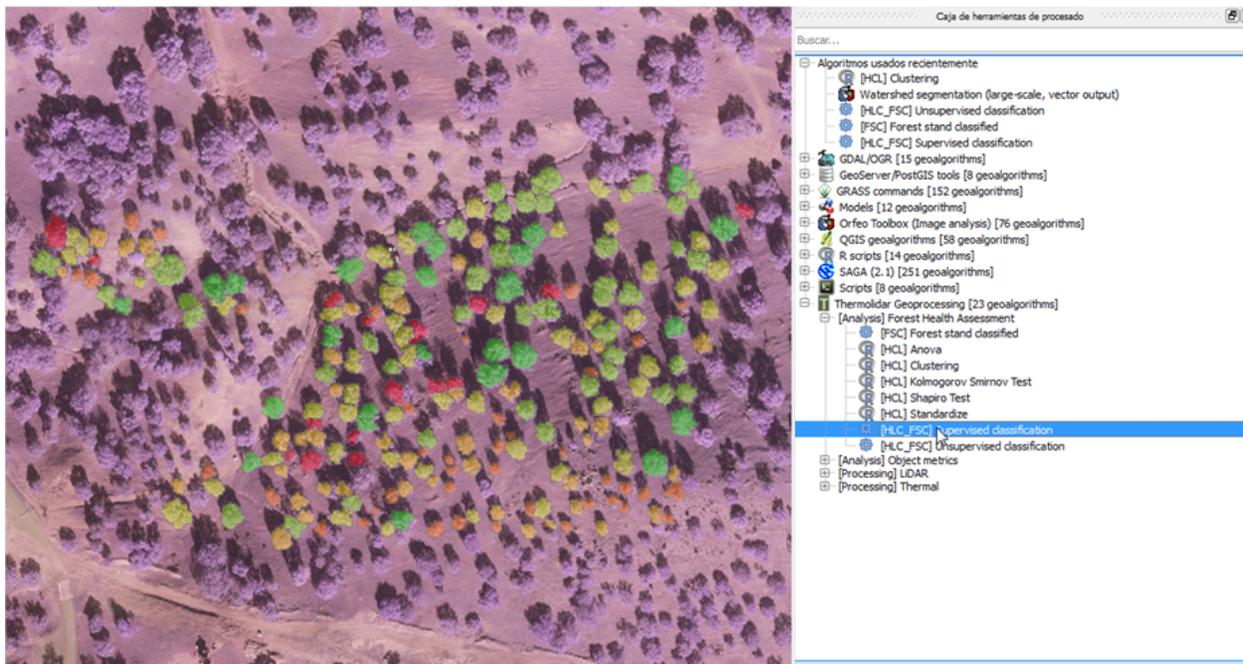
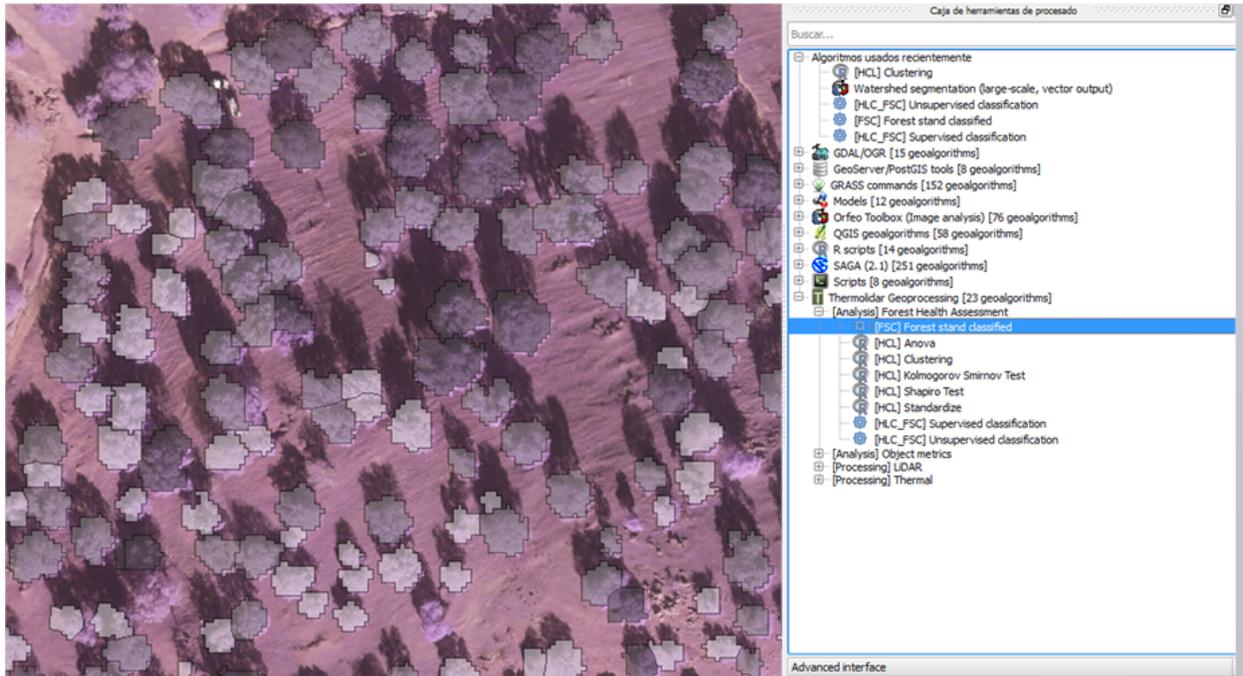
This tool provides the tools required for the classification of forest stands structurally different. The classification is based on two main structural parameters, average height of the trees and density. Input data needed to run this process is obtained from Lidar data. Alternatively, user can provide external forest maps in a .shp format type with an attribute of the number of class. The following figure shows an example of the units defined for the oak forest under analysis. Using a grey scale, trees were grouped in 3 classes with significant differences in terms of structural composition.

4.5.4 Forest Health Monitoring

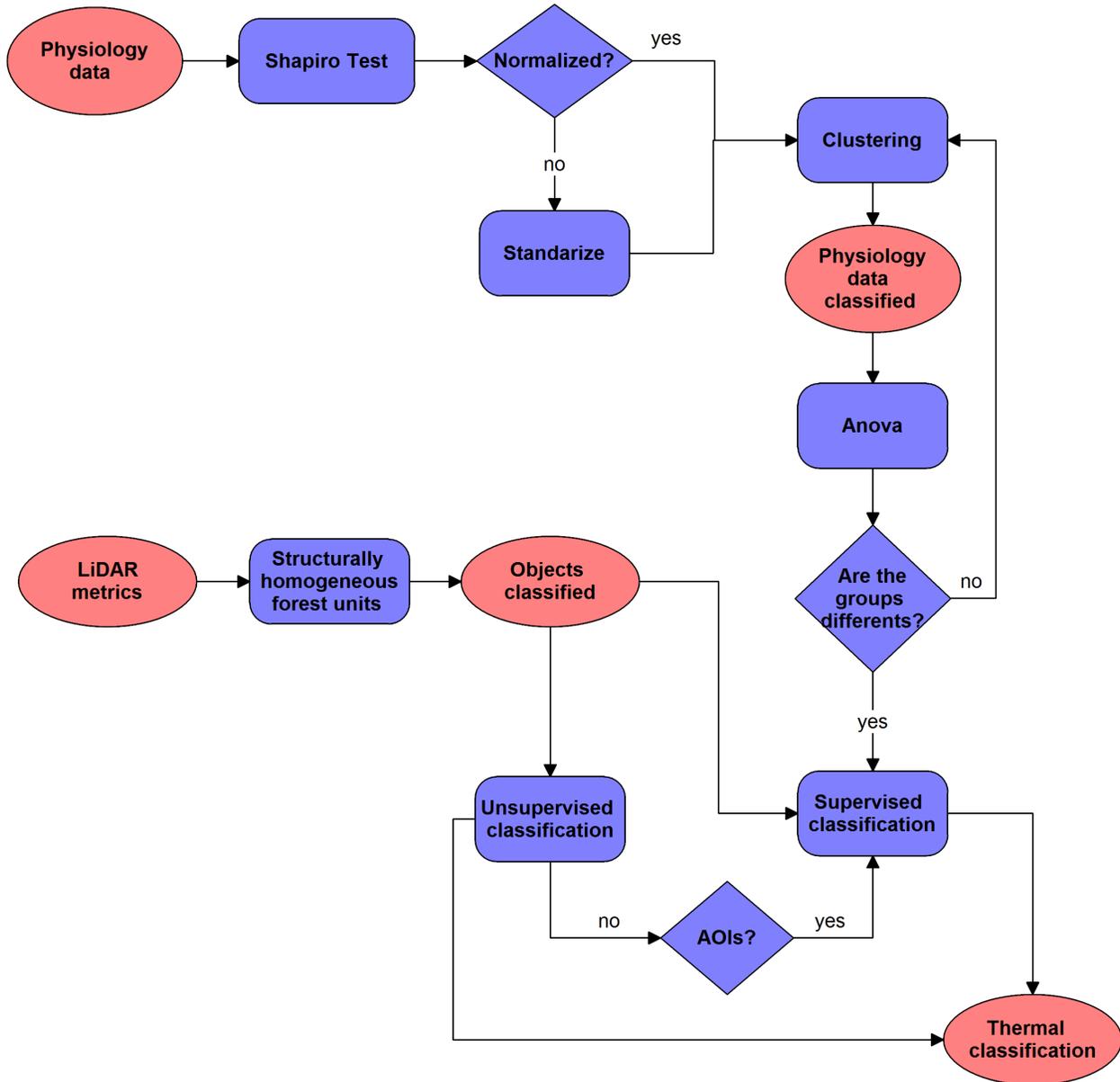
The main function of this tool is defining health condition differences in the vegetation at the stand level. Input parameters defined by users should mainly contain: thermal imaging data and the FSC Polygons (vector file with structurally homogeneous stands). Forest stands included in this analysis should be carried specifically based on one species. The user can perform a supervised or an unsupervised classification depending of the availability of field data measurements to define training areas. It should be highlight, that at this point of the analysis, users are willing to obtain an integrated mapping of forest health distribution levels based on thermal data but also standardized by forest stands units defined from lidar-based metrics. The following figure, shows an example of the units defined for assessment of the status of forest condition. Using a colour palette, trees were grouped in different classes with significant differences in terms of structural composition and physiological status. The colour palette ranges from red to green, where red colour is relate with trees with high level of damage and green colour represents trees with optimum health condition.

4.6 Data Analysis

This section provides the tools for analysis and interpretation of results. Once thermal and LiDAR data have been processed in the previous sections, the user can generate the mapping needed to interpret the physiological state of



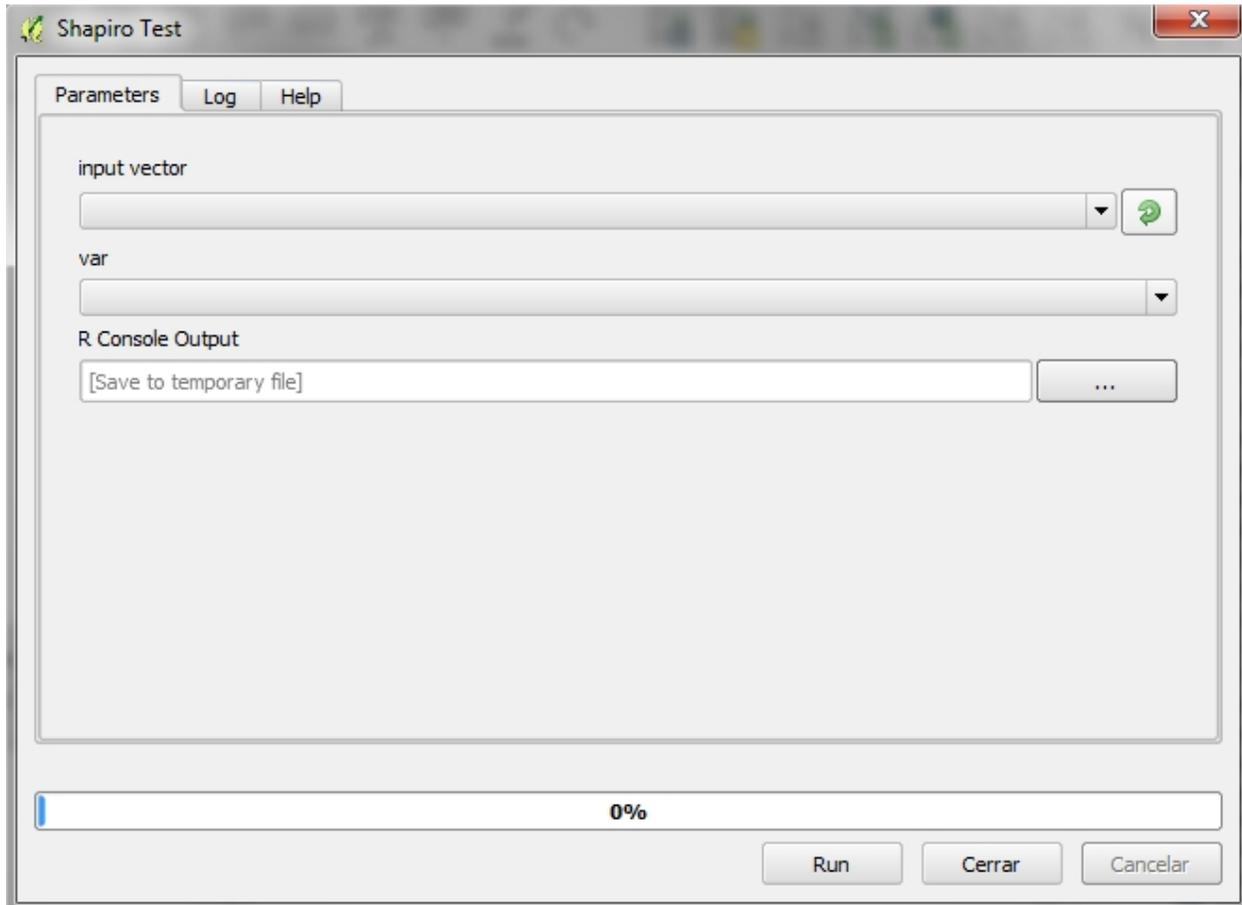
the forest mass analysed. First, the user has a set of data obtained in the field of physiology which are analysed and grouped, using the tools available at the module Health Condition Level. From the tools available in the structurally homogeneous forest units module, the user can perform a preliminary classification of the stands, based on structural homogeneity. This factor is important because thermal values behave differently according to the structure of objects. Finally, from the Forest Heath classification tools the user has the necessary tools to perform a classification based on the thermal values for the various homogeneous units. To improve the classification, the user can define training plots according to data collected in the field of physiology, visually are established different levels of affection.



4.6.1 Health Condition Levels

Shapiro Test

Before proceeding with the classification of items by level of damage according to several variables taken in the field, we verify that the set of physiological variables follow a normal distribution. For this we use the Shapiro test, located in the toolbox [Analysis] Health Condition Level > Shapiro Test.



Required Input Parameters

- **Input vector:** Vector file that contains information on physiological data.
- **Parameter:** Vector's field to analyze if it follows the normal distribution

Output Parameters

- **R Console Output: File with the output result. The output is composed of the following values**
 - Statistic - The value of the Shapiro-Wilk statistic.
 - p.value - An approximate p-value for the test. This is said in Royston (1995) to be adequate for p.value < 0.1.

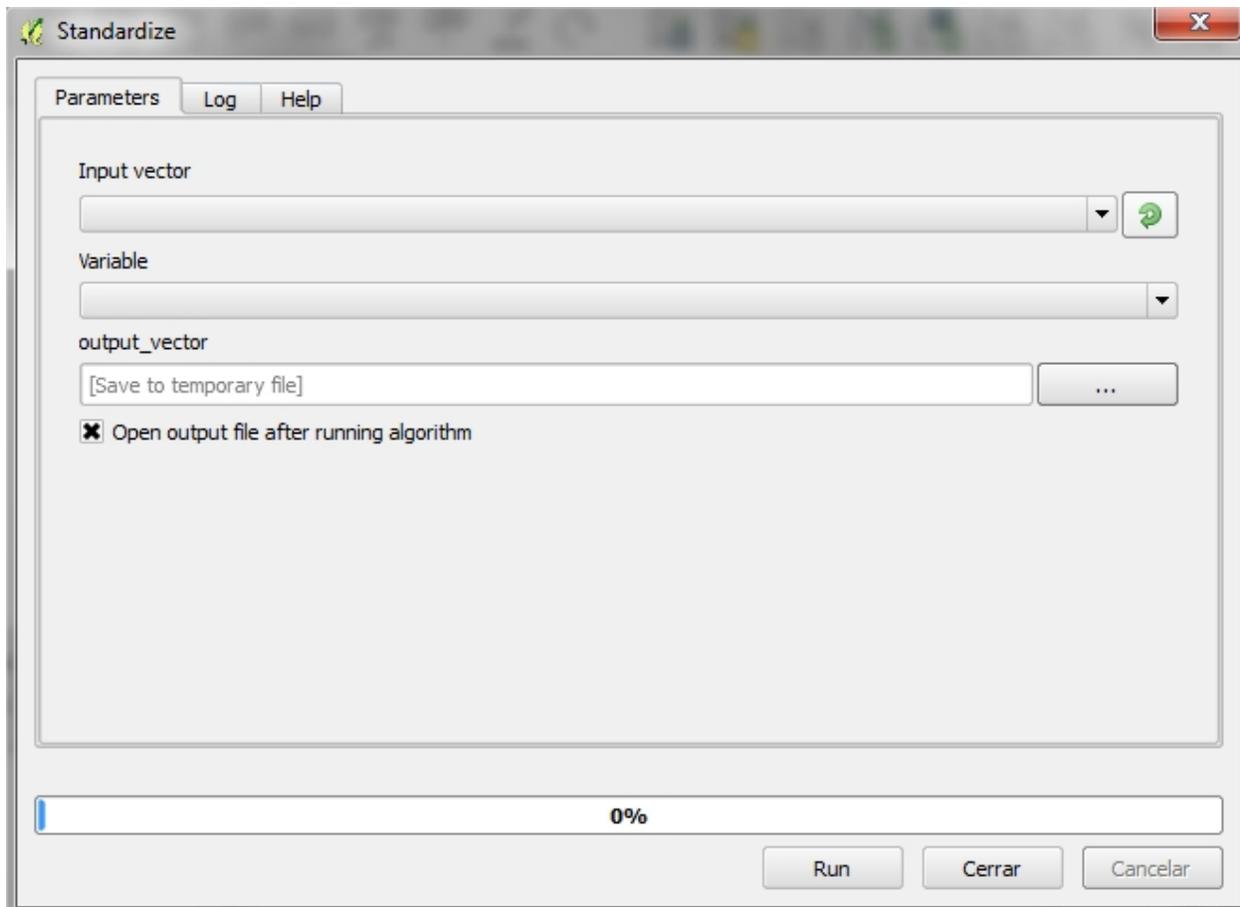
- Method - The character string “Shapiro-Wilk normality test”.
- data.name - A character string giving the name(s) of the data.

Interpretation

The null-hypothesis of this test is that the population is normally distributed. Thus if the p-value is less than the chosen alpha level, then the null hypothesis is rejected and there is evidence that the data tested are not from a normally distributed population. In other words, the data is not normal. On the contrary, if the p-value is greater than the chosen alpha level, then the null hypothesis that the data came from a normally distributed population cannot be rejected. E.g. for an alpha level of 0.05, a data set with a p-value of 0.02 rejects the null hypothesis that the data are from a normally distributed population. However, since the test is biased by sample size, the test may be statistically significant from a normal distribution in any large samples.

Standardize

To be able to use the variables in our analysis, they should follow a normal distribution. One way to force these follow a normal distribution is by definition. This characterization involves the conversion of the variable that follows a distribution $N(\mu, \sigma)$, a new variable with distribution $N(1,0)$. This tool is situated in [Analysis] Heath condition level > Standardize.



Required Input Parameters

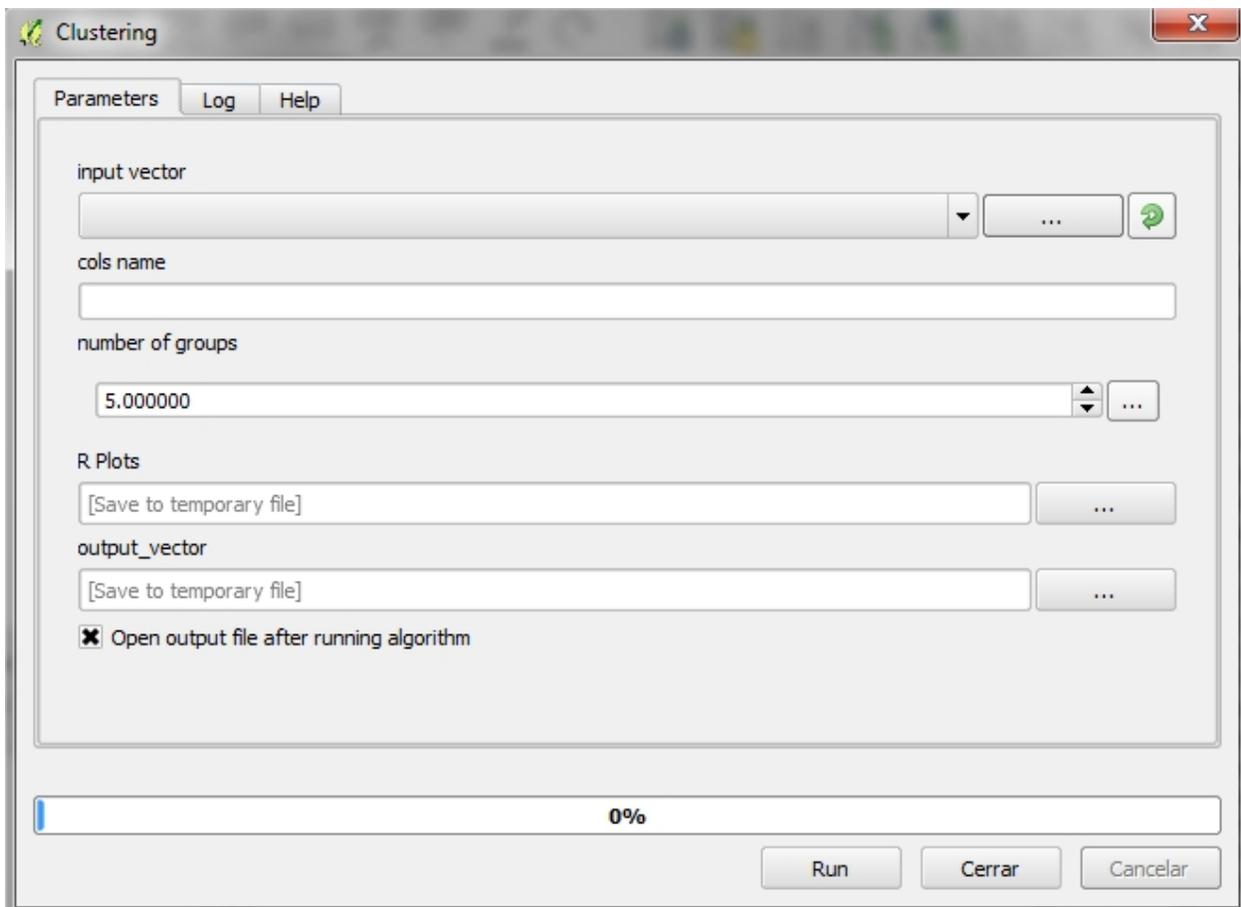
- **Input vector:** Shape that contains information on physiological data.
- **Variable:** Shapefile's field to standardize

Output Parameters

- **Output vector:** The user will output a new shapefile with the standardized variable.

Clustering

This tool allows us to group one or more physiological variables according to their degree of similarity between individuals in the sample. So, the goal of clustering is to determine the intrinsic grouping in a set of unlabelled data. But how to decide what constitutes a good clustering? It can be shown that there is no absolute *best* criterion which would be independent of the final aim of the clustering. Consequently, it is the user which must supply this criterion, in such a way that the result of the clustering will suit their needs. To make this tool has been chosen by a hierarchical approach. The user-supplied items are categorized into levels and sublevels within a class hierarchy, forming a hierarchical tree structure.



Required Input Parameters

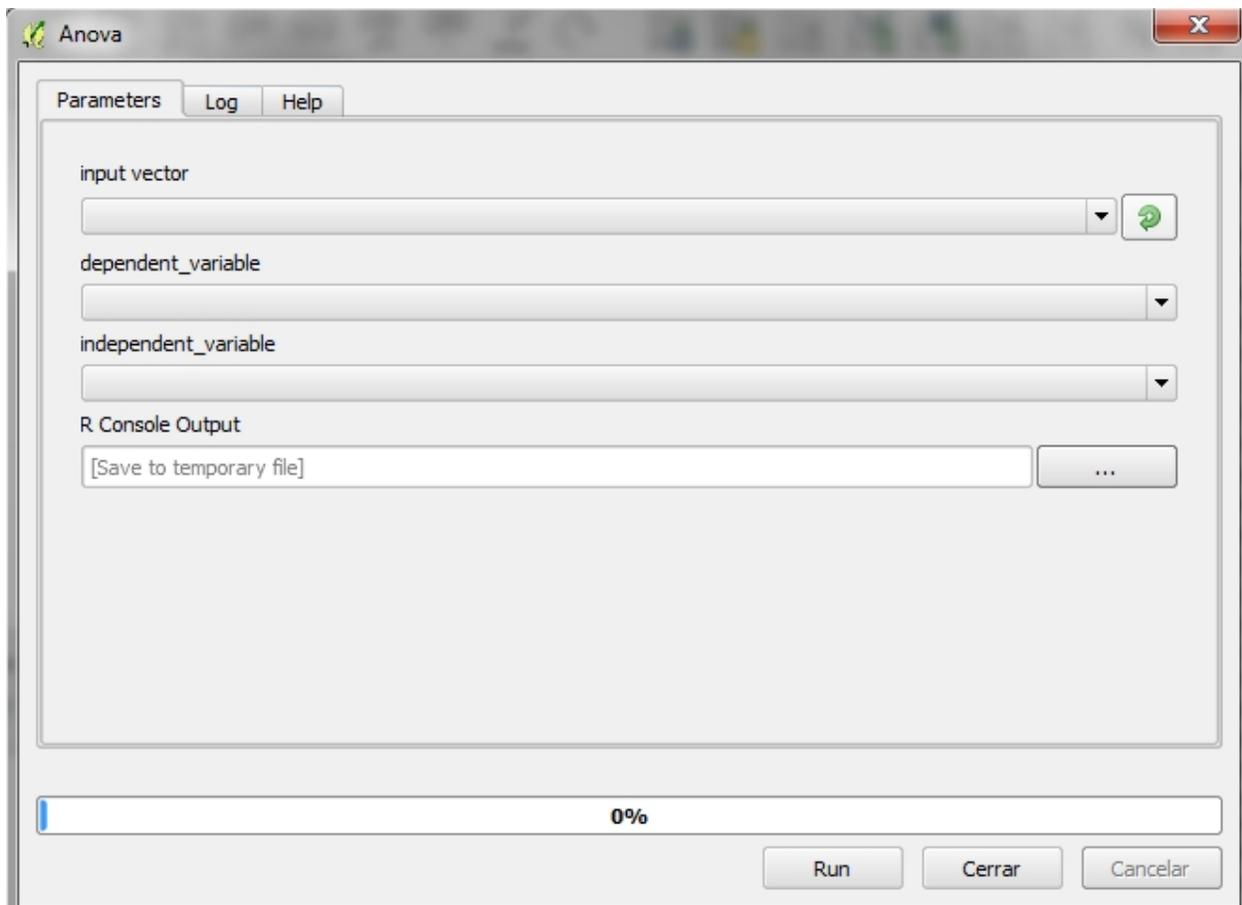
- **Input vector:** Shape containing information on physiological data.
- **Cols names:** Name the columns containing details physiology (separated by semicolons ‘;’)
- **Number of groups:** Number of groups

Output Parameters

- **R plots:** File with the R output result.
- **Output vector:** Output shapefile with a new variable (group) indicating that each group is member.

ANOVA

ANOVA test are conducted for each variable to indicate how well the variable discriminates between clusters. The hypothesis is tested in the ANOVA is that the population means (the average of the dependent variable at each level of the independent variable) are equal. If the population means are equal, it means that the groups did not differ in the dependent variable, and consequently, the independent variable is independent of the dependent variable.



Required Input Parameters

- **Input vector:** Shapefile that contains information about the cluster
- **Dependent variable:** Field shape that acts as a dependent variable
- **Independent variable:** Field shape that acts as an independent variable

Output Parameters

- **R Console Output:** File with the R output result. The result will be a list of ANOVA tables, one for each response (even if there is only one response). They have columns “Df”, “Sum Sq”, “Mean Sq”, as well as “F value” and “Pr(>F)” if there are non-zero residual degrees of freedom. There is a row for each term in the model, plus one for “Residuals” if there are any.

Interpretation

If the critical level associated with the F statistics (ie, the probability of obtaining values as obtained or older), is less than 0.05, we reject the hypothesis of equal means and conclude that not all the population means being compared are equal. Otherwise, we cannot reject the hypothesis of equality and we cannot claim that the groups being compared differ in their population averages.

4.6.2 Structurally Homogeneous Forest Units

Structurally Homogeneous forest units

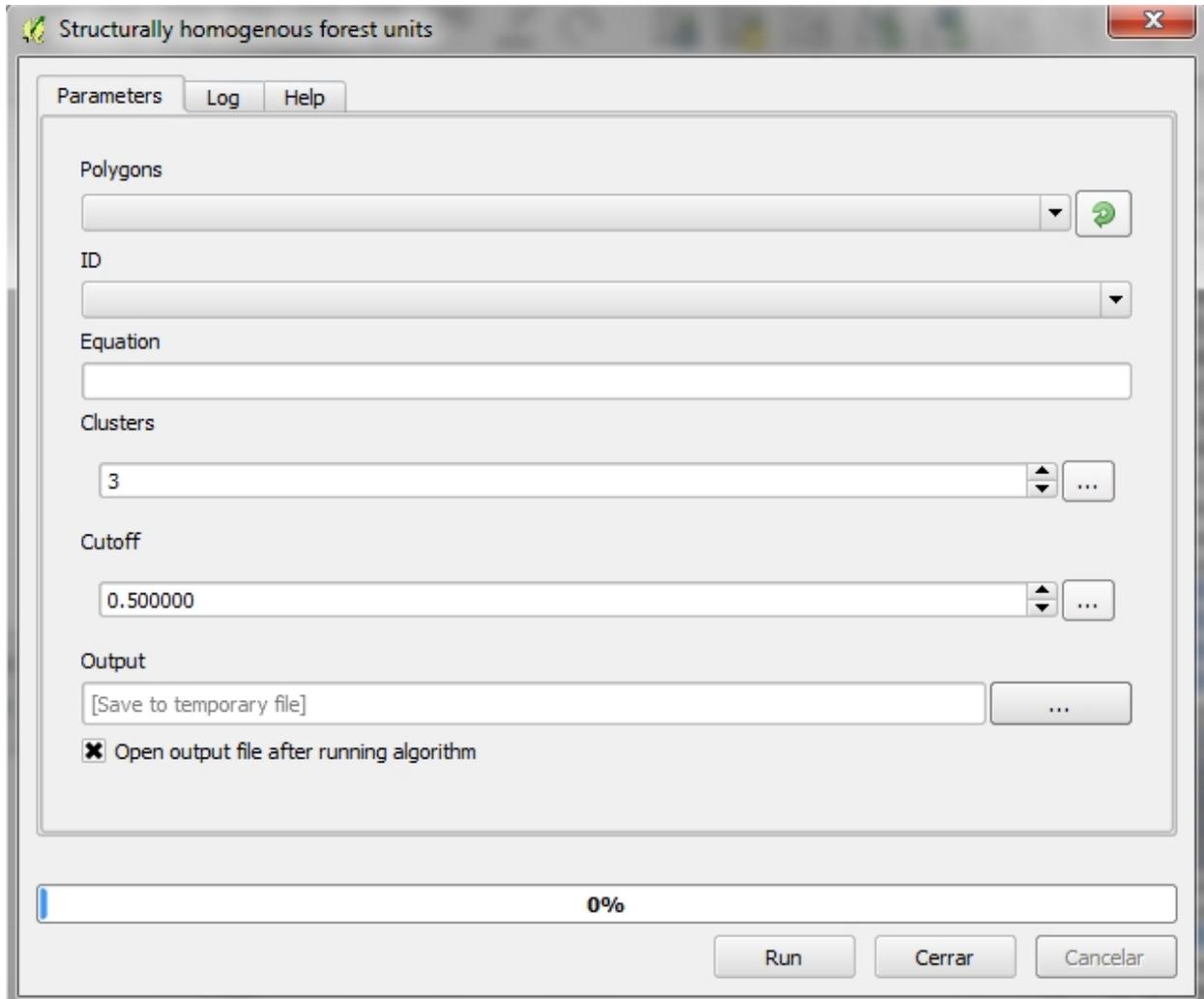
The analytical purpose of this tool is the definition of structurally homogeneous stands that allow us to minimize the effects of structure on the thermal information, and therefore allow us to obtain related health outcomes woodland. To do this, the software allows the user to define the structure of the stand from the height data. The calculation of uniformity is therefore a function of two variables directly derived from LiDAR data the 95th percentile obtained from MDV and the penetration rate, obtained from density points that penetrate the forest canopy.

Required Input Parameters

- **Polygons:** Vector file that contains polygons to classify.
- **ID:** Vector’s field that that indicates the id of each item
- **Equation:** Operator used to classify the feats.
- **Clusters:** Number of output groups. The value is 3 by default.
- **Cutoff:** Stop threshold algorithm. The value is 0.5 by default.

Output Parameters

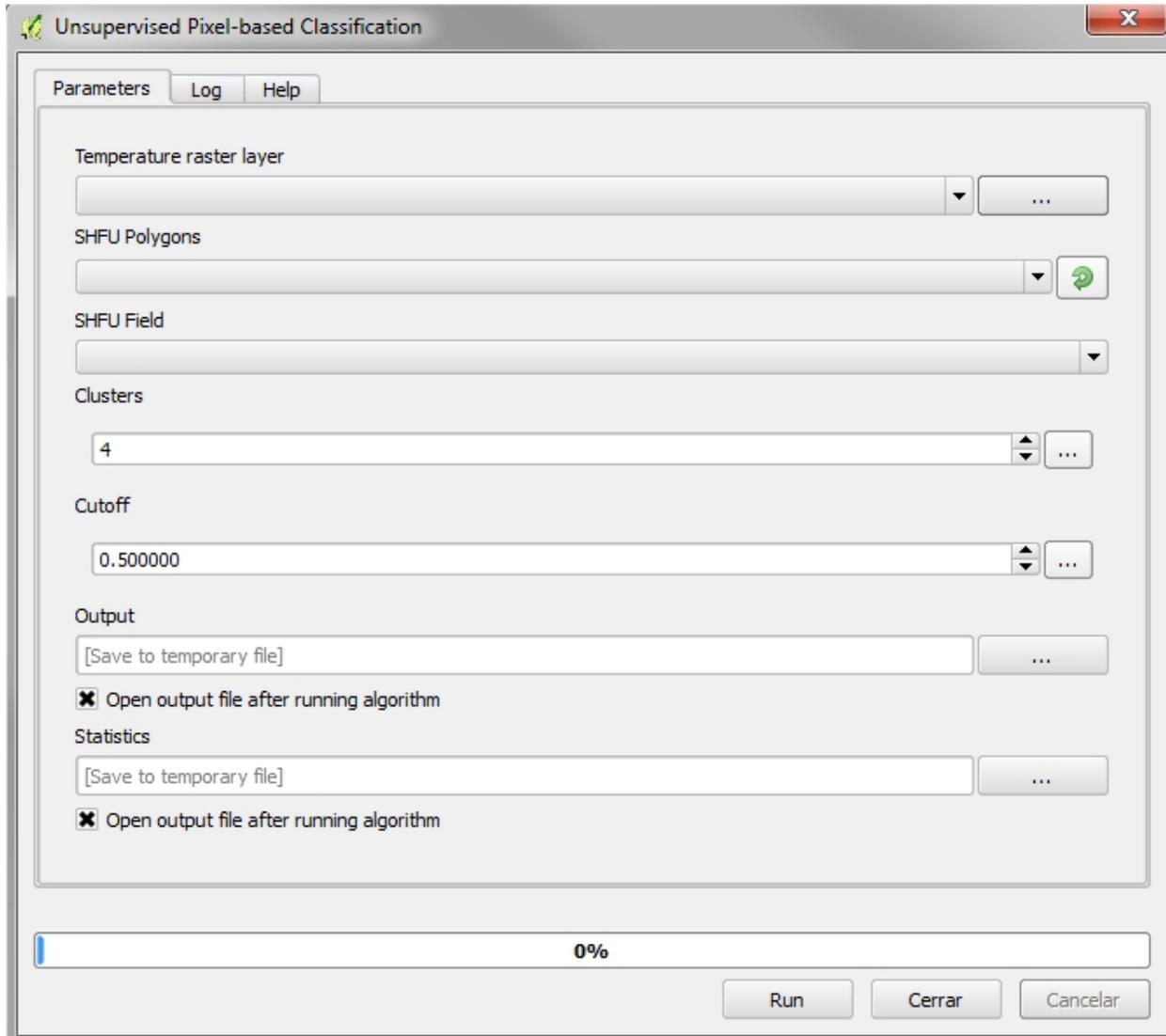
- **Output:** Vector file name containing the classification.



4.6.3 Forest Health Classification

Unsupervised Pixel-based Classification

The user has a stratification of the study area, and classified based on the structure. Through this tool, a classification of the pixels of temperatures will be performed, based on the classification of defined structure. Thus, the output will be a raster temperature for each of the groups of homogeneity.



Required Input Parameters

- **Temperature raster layer:** Input temperature raster
- **SHFU Polygons:** Vector layer that contains structurally homogenous stands.
- **SHFU Field:** SHFU layer field containing the group that belong each item.
- **Clusters:** Number of output classes.

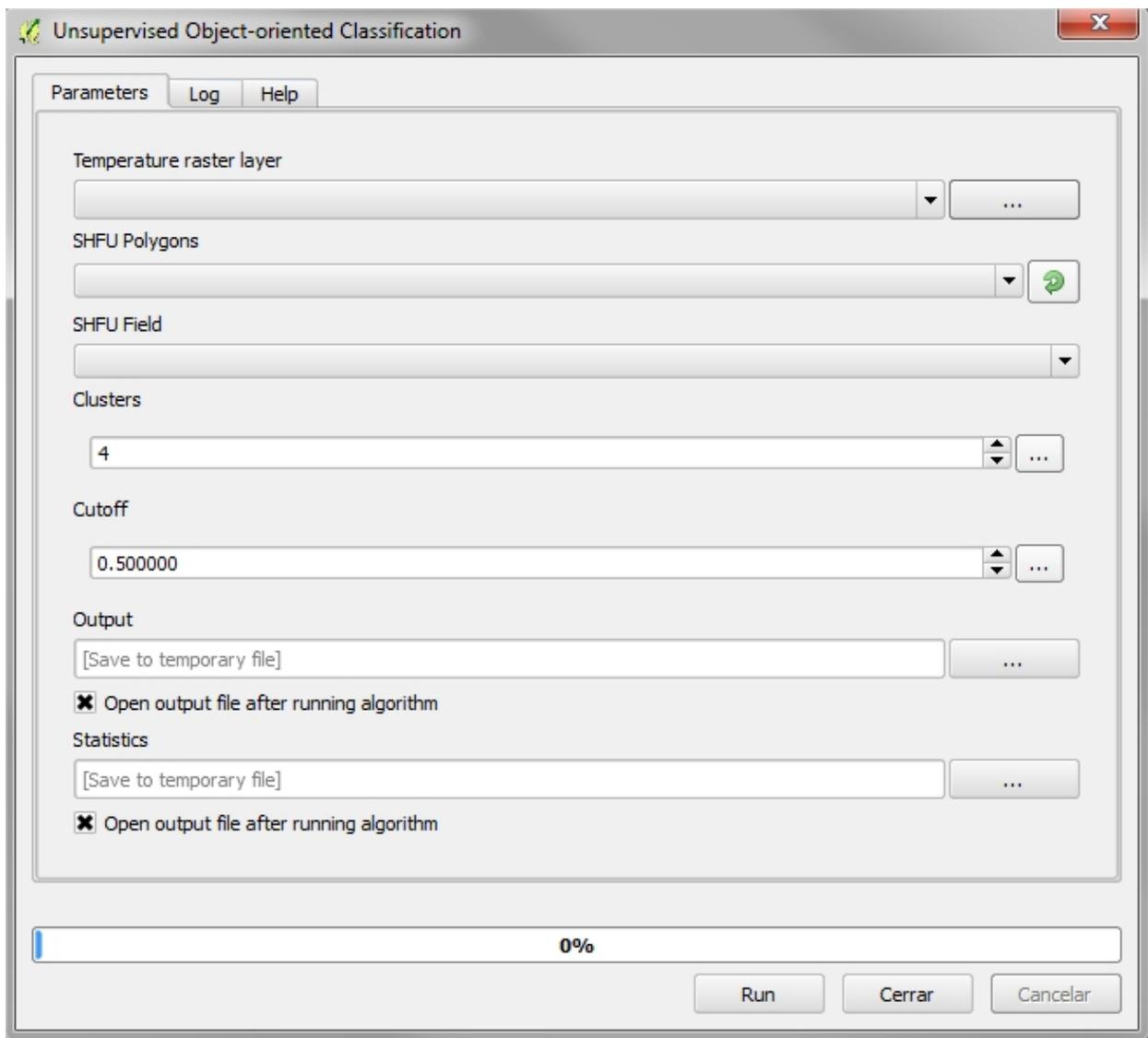
- **Cutoff:** Stop threshold. The value is 0.5 by default.

Output Parameters

- **Output:** Raster file name containing the classification of temperatures based on homogenous stand units.
- **Statistics:** CSV File containing statistics for groups.

Unsupervised Object-oriented Classification

The user has a stratification of the study area, and classified based on the structure. Through this tool, a classification of the objects of temperatures mean will be performed, based on the classification of defined structure. Thus, the output will be a raster temperature for each of the groups of homogeneity.



Required Input Parameters

- **Temperature raster layer:** Input temperature raster
- **SHFU Polygons:** Vector layer that contains structurally homogenous forest units.
- **SHFU Field:** SHFU layer field containing the group that belongs each item.
- **Clusters:** Number of output classes.
- **Cutoff:** Stop threshold. The value is 0.5 by default.

Output Parameters

- **Output:** Raster file name containing the classification of temperatures based on homogenous stand units.
- **Statistics:** CSV File containing statistics for groups.

Supervised Pixel-based Classification

Similarly as in the previous point, the user can perform a classification of the temperature response to the stand of homogeneity. Unlike supervised classification, the user has a number of AOIs that guide the classification process.

Required Input Parameters

- **Temperature raster layer:** Input temperature raster
- **SHFU Polygons:** Vector layer that contains structurally homogenous stands.
- **SHFU Field:** SHFU layer field containing the group that belong each item.
- **ROIs vector:** Vector file with training areas.
- **ROIs - SHFU Identifier:** Vector's field that contains the group's homogeneity that belongs each item of training areas.
- **ROIs - FSC Identifier:** Vector's field that contains the group "Forest health level" that belongs within their group of homogeneity.
- **k:** Number k nearest neighbors. The value is 3 by default.

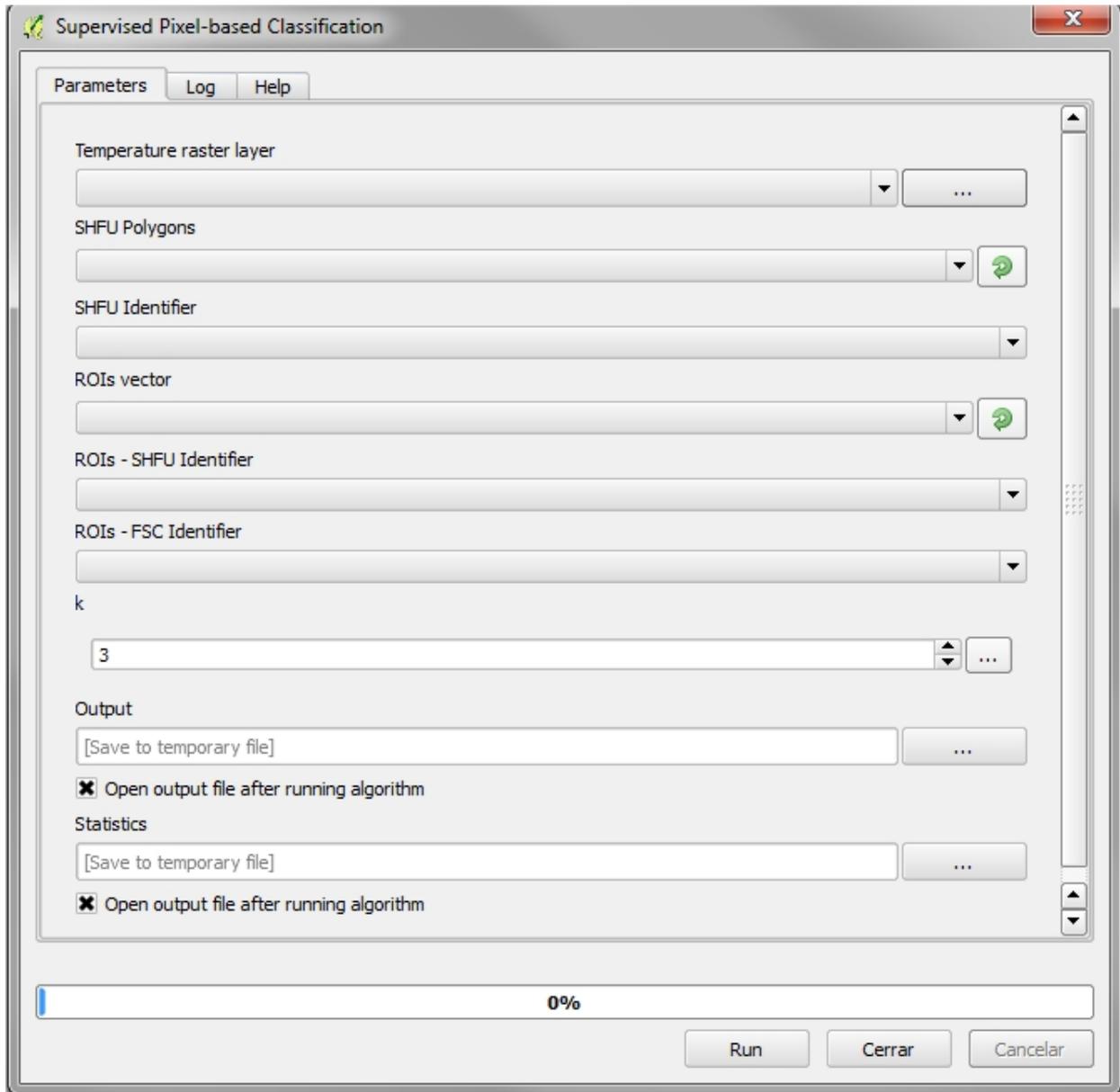
Output Parameters

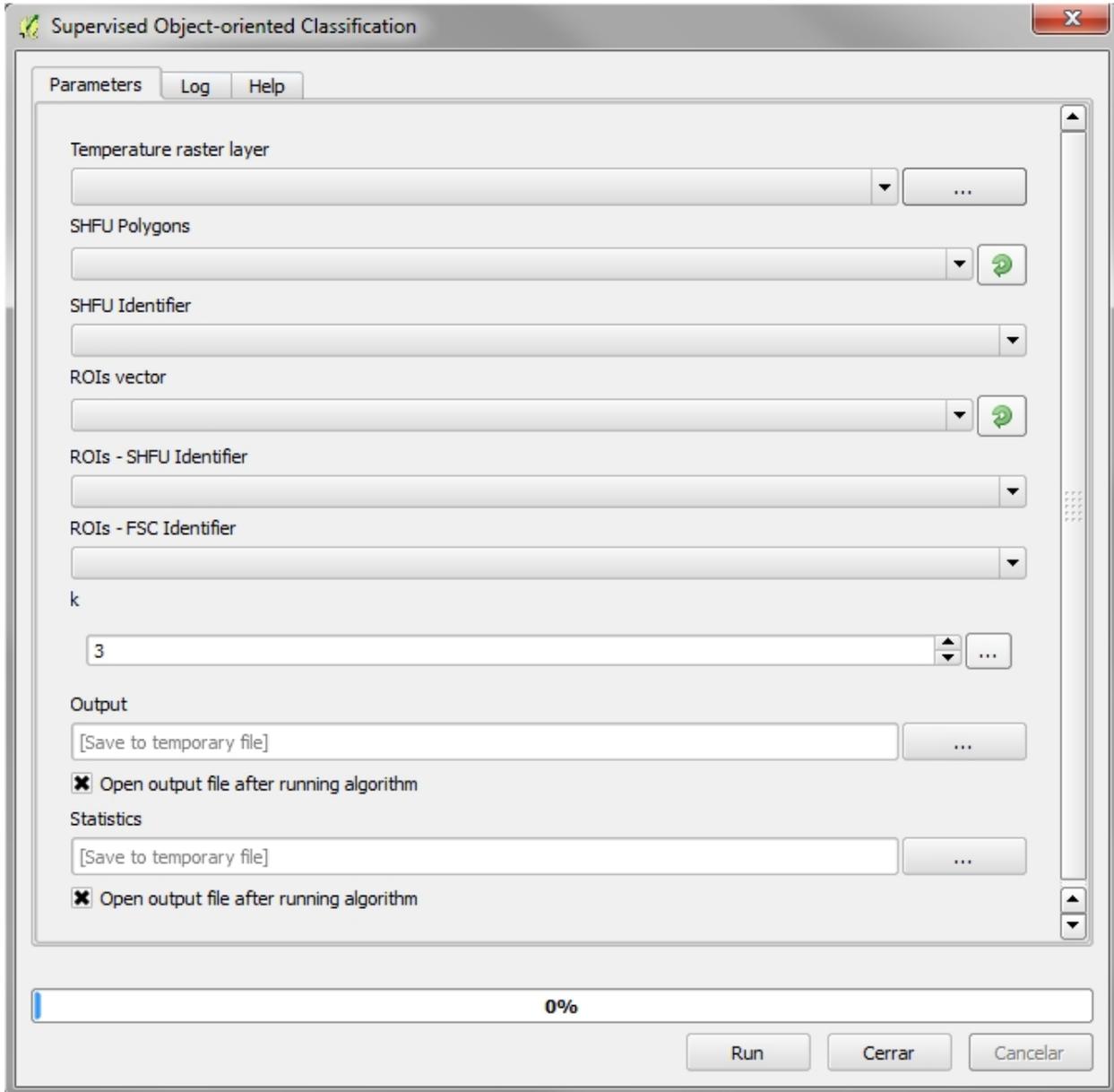
- **Output:** Raster file name containing the classification of temperatures based on homogenous stand units.
- **Statistics:** CSV File containing statistics for groups.

Supervised Object-oriented Classification

Required Input Parameters

- **Temperature raster layer:** Input temperature raster
- **SHFU Polygons:** Vector layer that contains structurally homogenous stands.
- **SHFU Field:** SHFU layer field containing the group that belong each item.





- **ROIs vector:** Vector file with training areas.
- **ROIs - SHFU Identifier:** Vector's field that contains the group's homogeneity that belongs each item of training areas.
- **ROIs - FSC Identifier:** Vector's field that contains the group "Forest health level" that belongs within their group of homogeneity.
- **k:** Number k nearest neighbors. The value is 3 by default.

Output Parameters

- **Output:** Raster file name containing the classification of temperatures based on homogenous stand units.
- **Statistics:** CSV File containing statistics for groups.

5.1 Case Study 1 - Huelva (Spain)

5.1.1 Thermal Processing

Thermal Calibration

For calibration of the thermal image we have the following data:

- Thermal image of the study area in Huelva, each pixel of the image represents the temperature value in degrees kelvin. **thld_training\huelva\Z1_holmOak\Temperature\t_huelvaZ1.tif**
- Set of calibration values stored in a vector file. Each item has the id and the value of collected temperature (in degrees Kelvin). **thld_training\huelva\Z1_holmOak\Temperature\t_aois\t_calibration.shp**

We proceed to load both the image and the shape to QGIS interface. To load the image in the main menu (Layer - Add Raster Layer ...) to load the shape of points (Layer - Add Vector Layer ...).

We proceed to perform the calibration of the thermal image through Thermolidar tool toolbox [**Processing**] **Thermal - Thermal Calibration**.

We introduce the values as shown in the interface:

We obtain as output calibration file of the thermal image, as shown in the following figure:

Tc - Ta

We proceed to subtract the air temperature to the calibrated raster temperatures. We have available the following information.

- Raster temperature calibrated in the previous section. **thld_training\huelva\Z1_holmOak\out\thermal_processing\t_calibration.tif**
- Air temperature acquired in flight time. In our case we will use the value of temperature **298.15 °K**

We proceed to perform the calibration of the thermal image through Thermolidar tool toolbox [**Processing**] **Thermal - Tc-Ta**.

We introduce the input parameters in the user interface:

Obtaining as output file the following result:

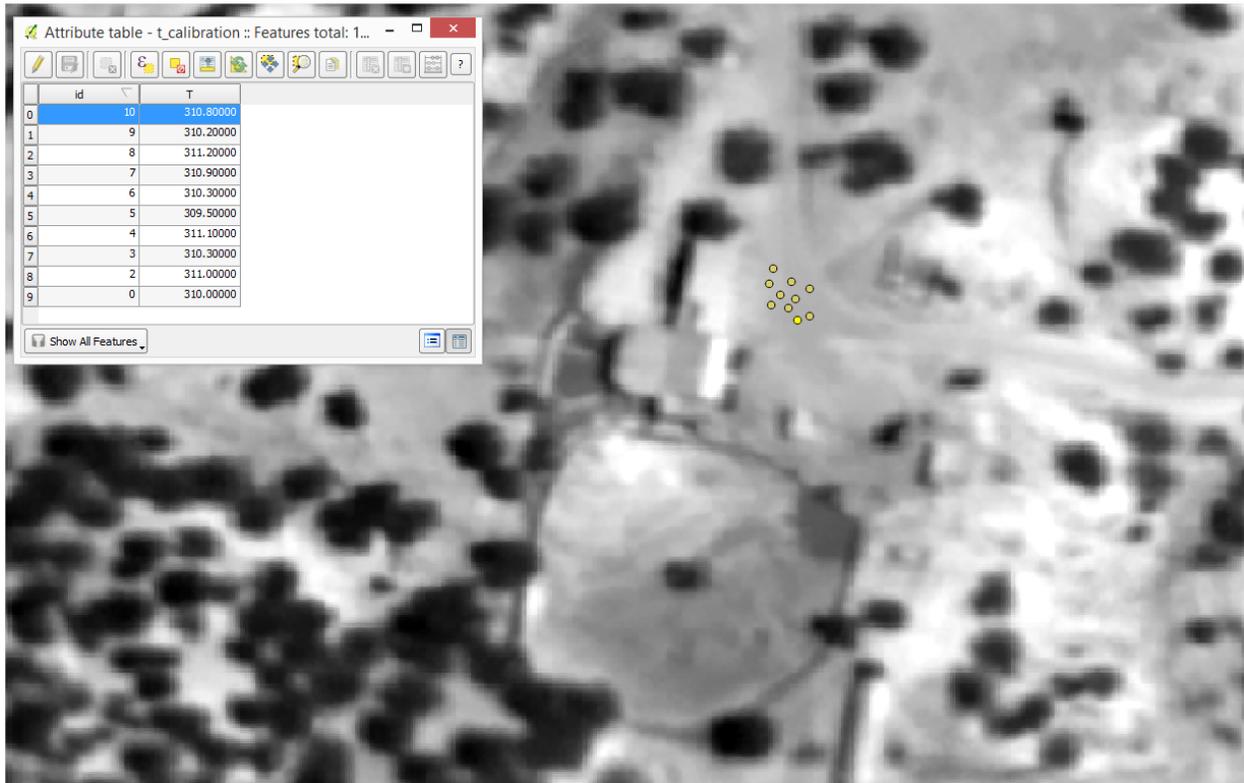


Figure 5.1: Thermal image of Huelva and attribute table containing the field thermal data

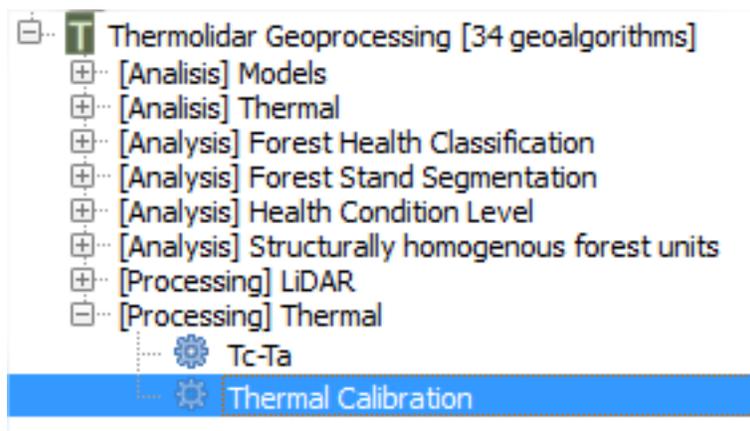


Figure 5.2: Thermal calibration tool

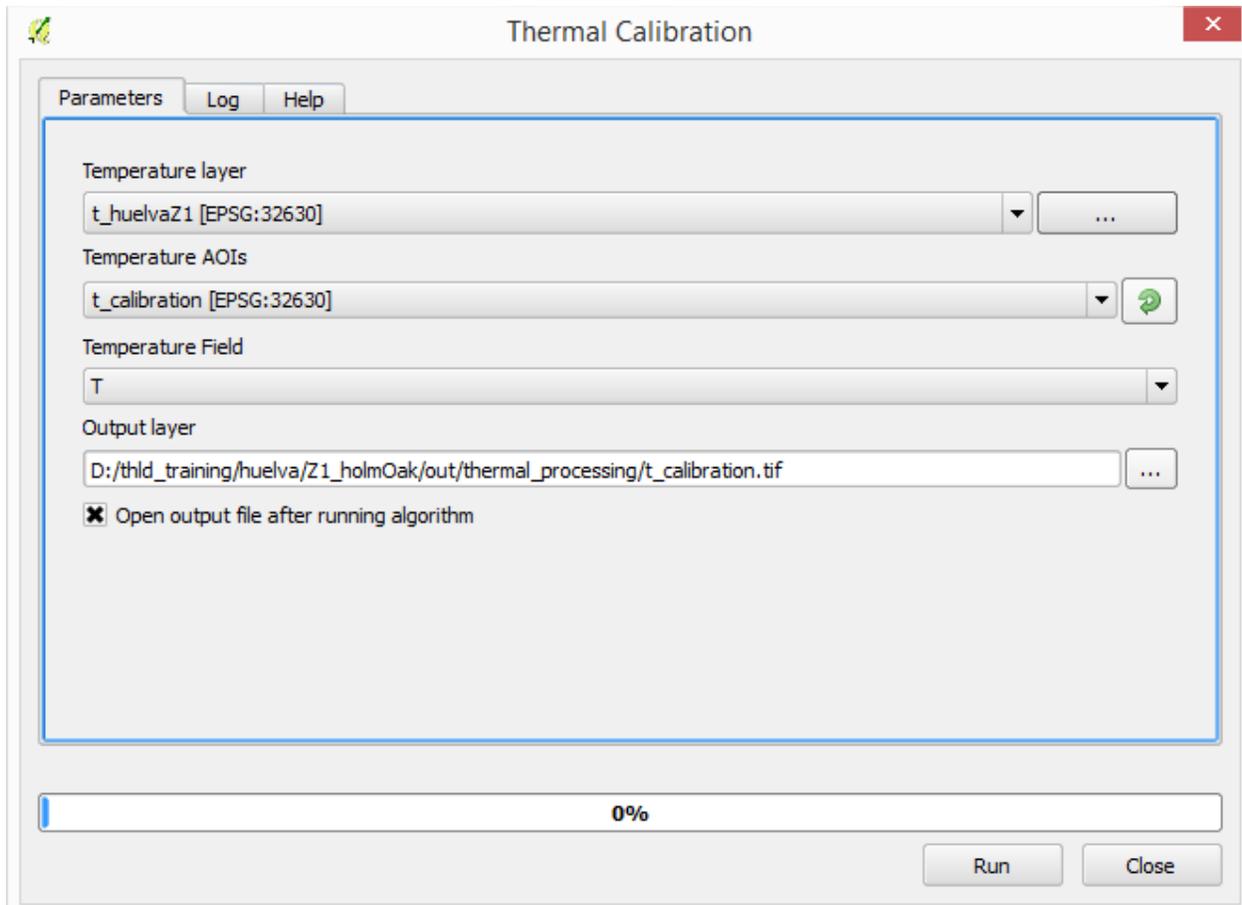


Figure 5.3: Interface of the Thermal Calibration module

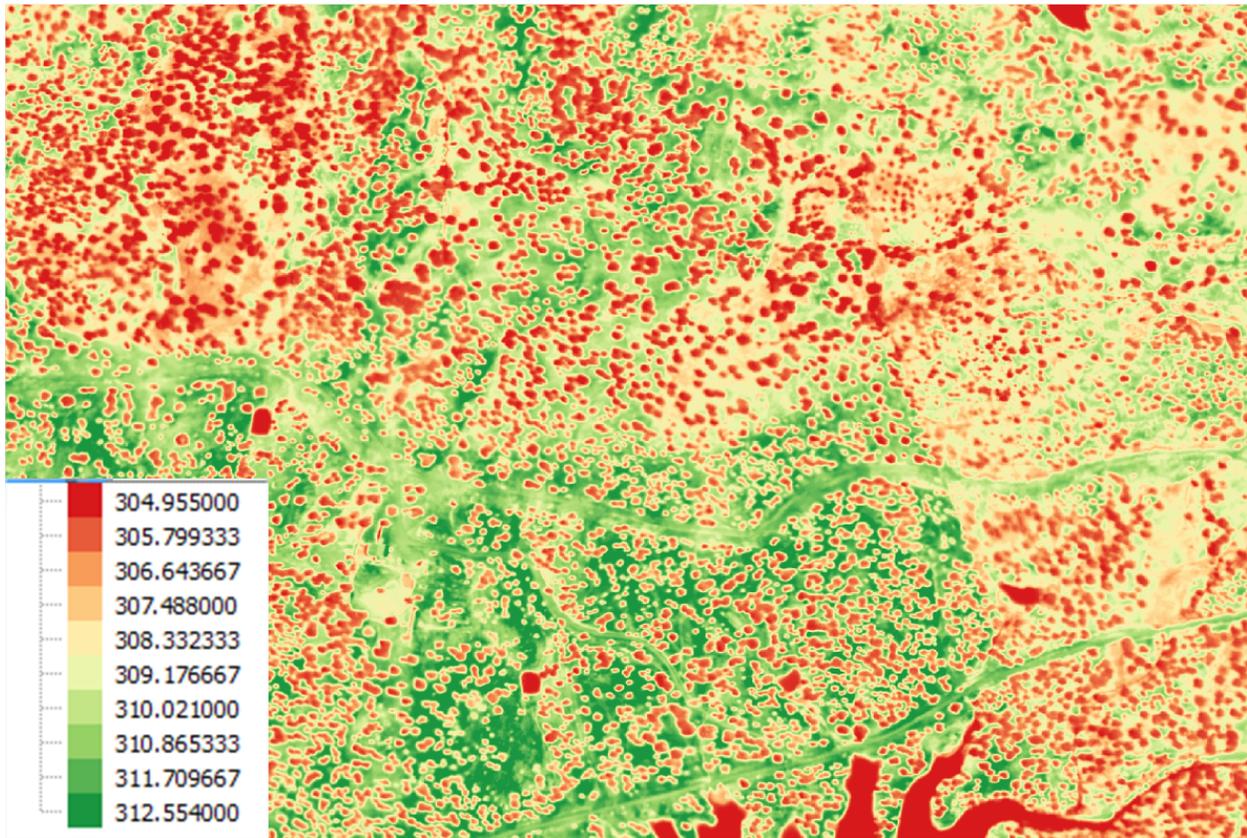


Figure 5.4: Thermal calibration output

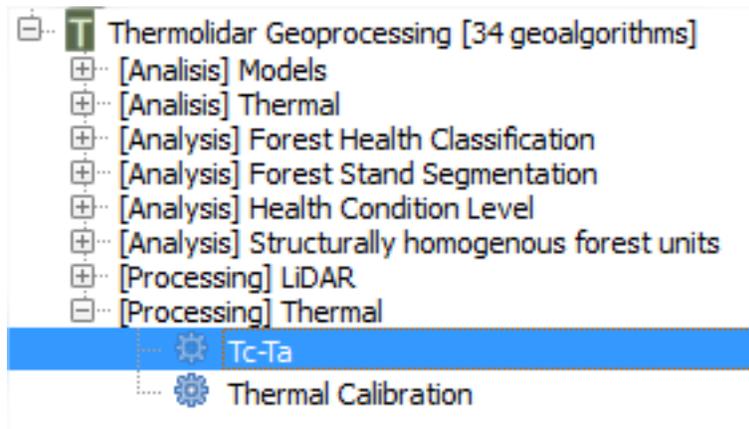


Figure 5.5: Tc Ta module is located in the “Thermal” submenu of the THERMOLIDAR plugin toolbox

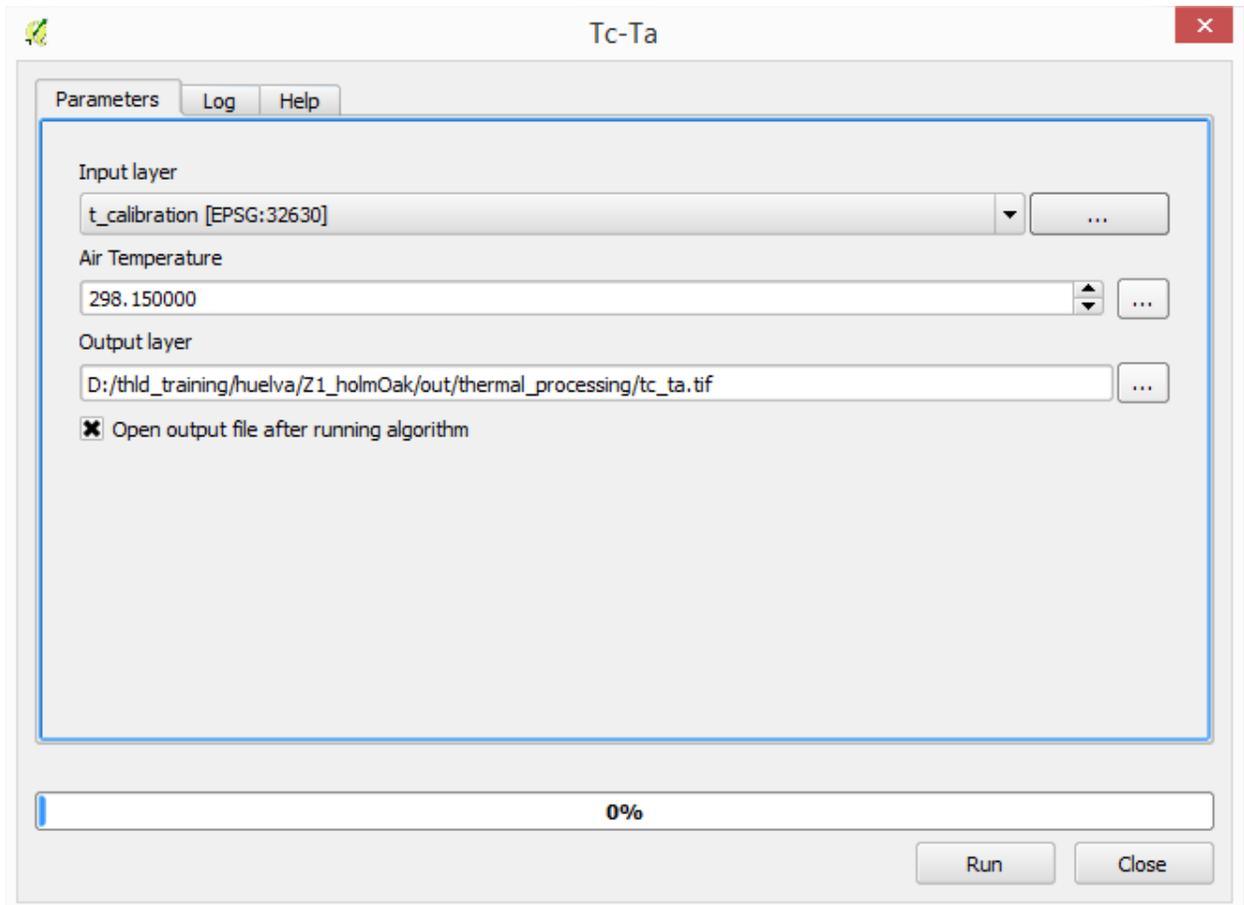


Figure 5.6: Interface of the Tc-Ta module

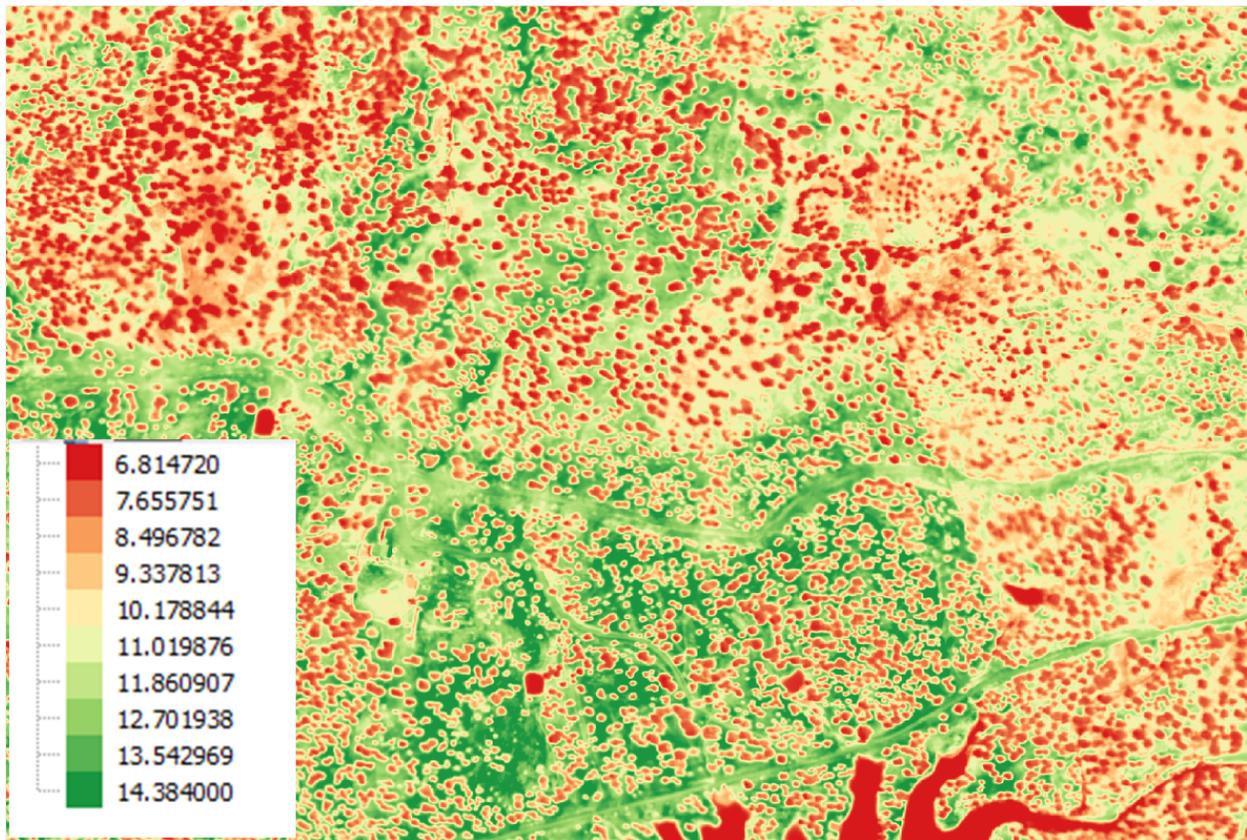


Figure 5.7: Output thermal image of Huelva

5.1.2 Data analysis

Health condition levels

We have the physiology data of Leaf Area Index (LAI) for 25 of the 216 trees inventoried in the area of Huelva, stored in the vector file: `thld_training\huelva\Z1_holmOak\FieldData\LAI\lai_huelva.shp`.

First, load the shape of polygons in QGIS (Layer - Add vector layer ...)

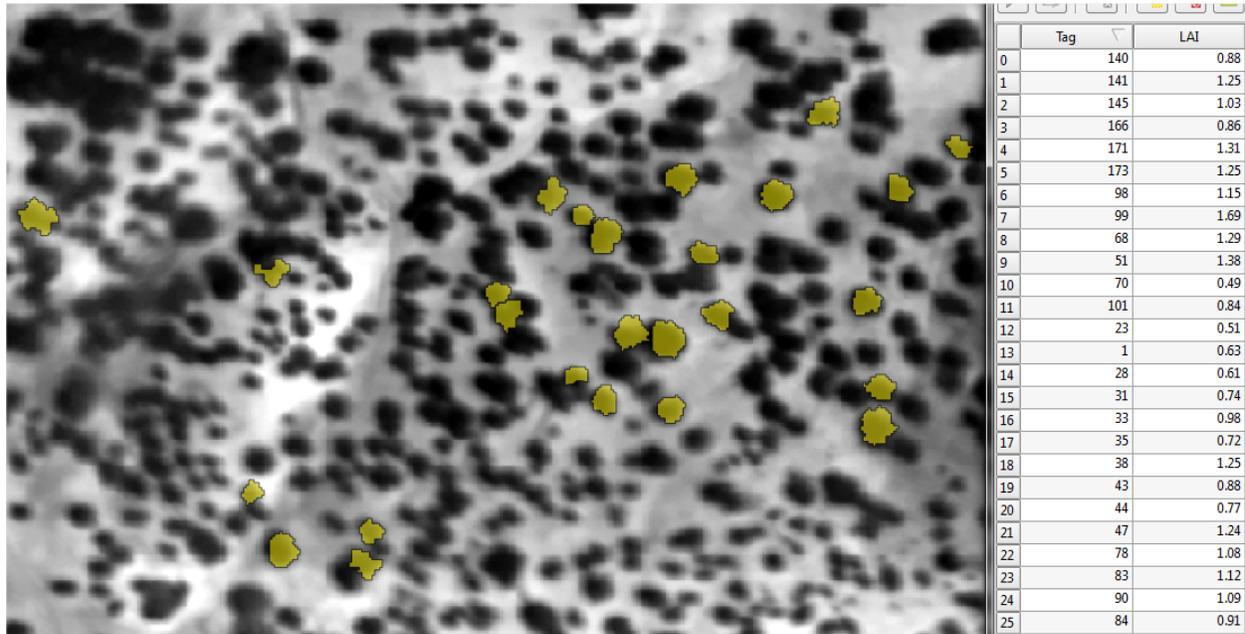


Figure 5.8: LAI measurements in Huelva

Before proceeding with the classification of items by level of damage according to several variables taken in the field, we verify that the set of physiological variables follow a normal distribution. For this we use the Shapiro test, located in the toolbox **[Analysis] Health Condition Level > Shapiro Test**.

Warning: The first time you use these tools, you will need to start QGIS in administrator mode (R install required dependencies)

Shapiro Test

- **Input vector:** Vector file that contains information on physiological data.
- **Var:** Vector's field to analyze if it follows the normal distribution. In this case the parameter LAI

We introduce the parameters as shown in the following figure:

Obtaining the following output:

In the example, the p-value is much higher than 0.05, so we conclude that LAI data follow a normal distribution. In the case where p-value is less than 0.05 the data would be discarded, or these should be normalized.

In the case that the variable does not follow a normal distribution, it is necessary to standardize using the **[Analysis] Health Condition Level > Standardize**

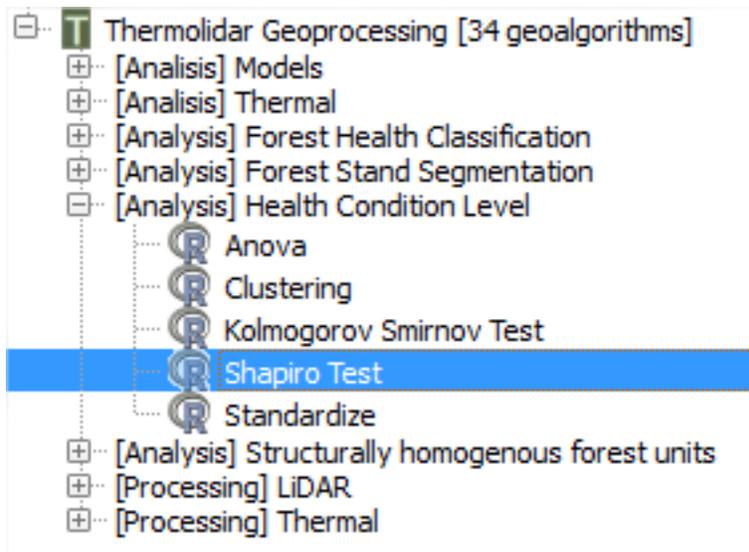


Figure 5.9: **Shapiro Test** is located in the “Health Condition Level” submenu of the THERMOLIDAR plugin toolbox

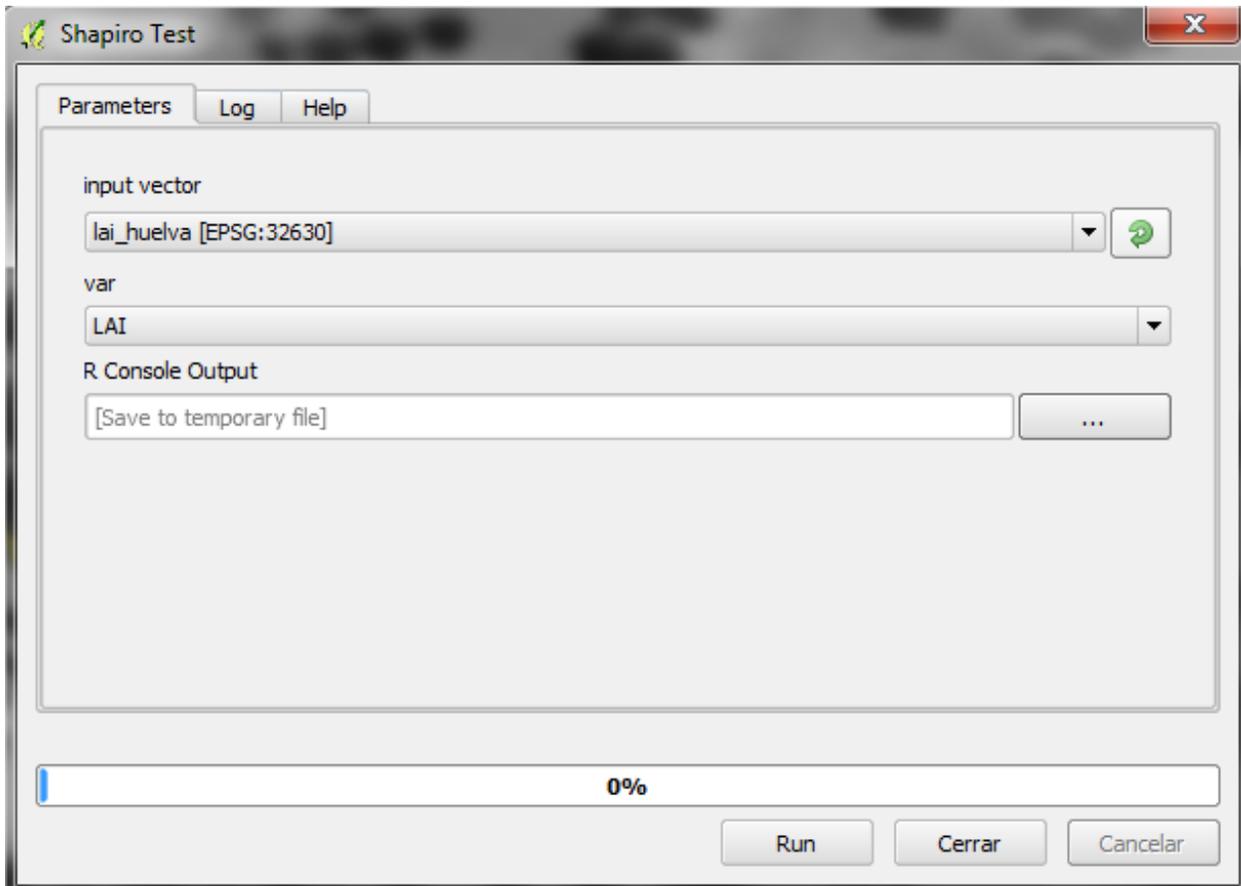


Figure 5.10: Interface of the “Shapiro Test” module

R Output

```
Shapiro-Wilk normality test

data: FAO[[var]]

W = 0.9748, p-value = 0.7491
```

Clustering

This tool allows us to group one or more physiological variables according to their degree of similarity between individuals in the sample. In this case we will group by the variable LAI, we have previously verified that follows a normal distribution. This tool creates many groups tool damage level depending on the specified physiological variable. In this case, we will select 3 levels as a function of LAI variable. We can find the tool in **[Analysis] Health Condition Level > Clustering**

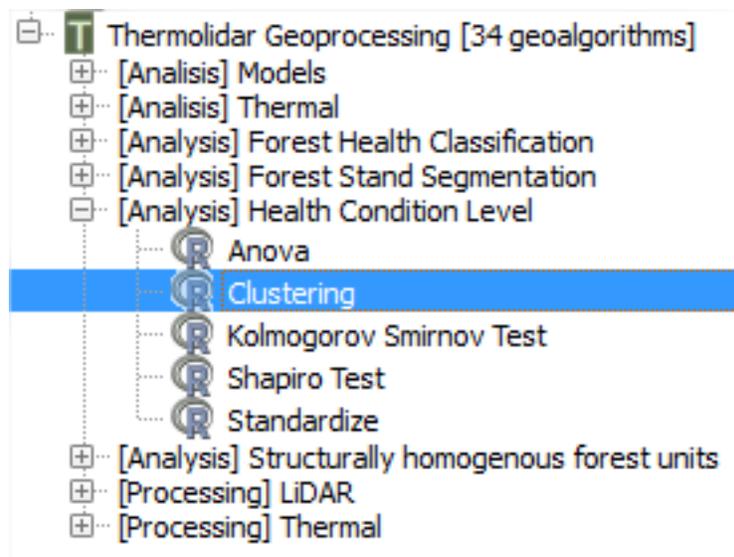


Figure 5.11: **Clustering** is located in the “Health Condition Level” submenu of the THERMOLIDAR plugin toolbox

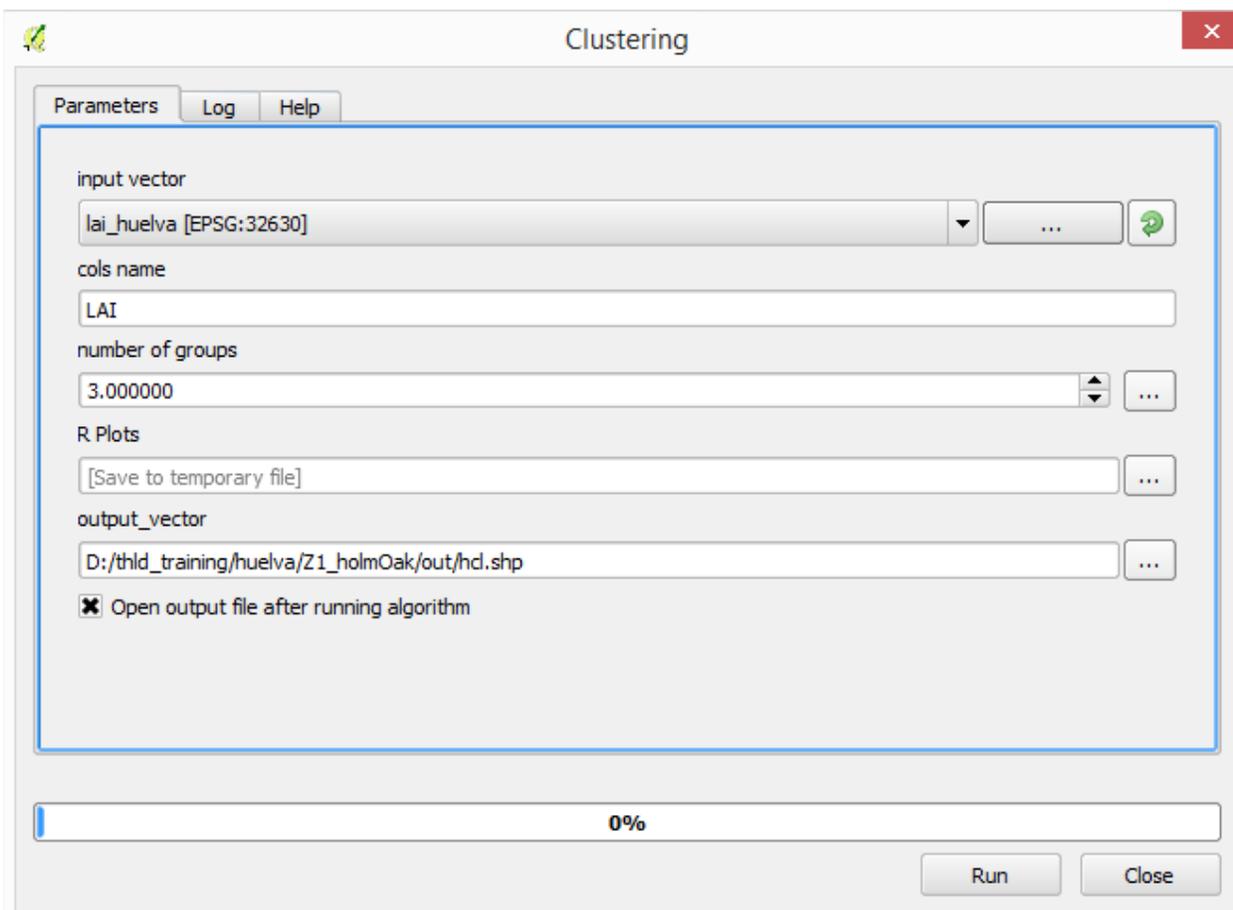
We introduce the values as shown in the following figure:

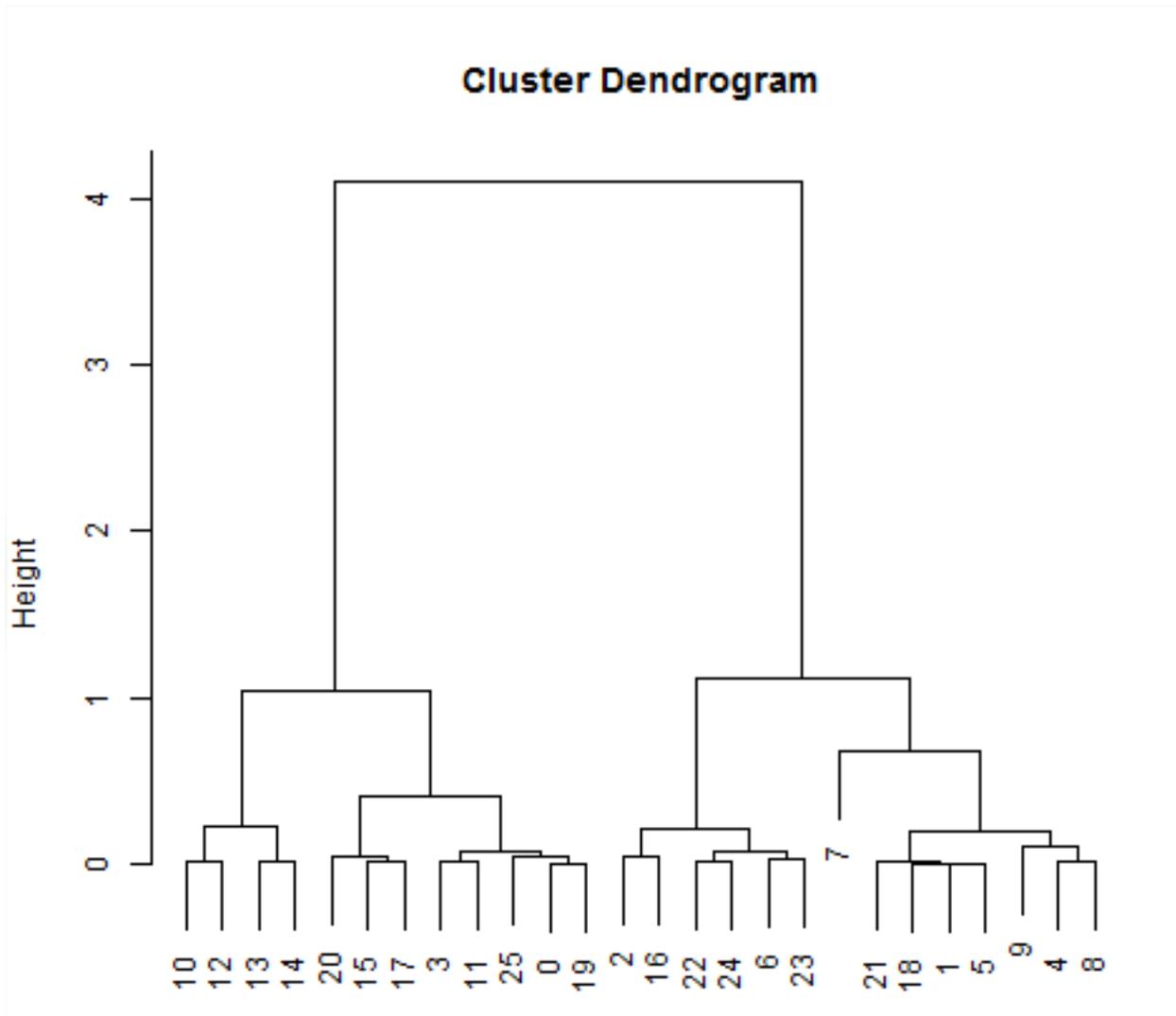
The vector output file will be stored in the folder **thld_training\huelva\VZ1_holmOak\out\hcl.shp** that will be used subsequently. Each of the trees will be classified as regions of interest to guide the supervised classification of the thermal image.

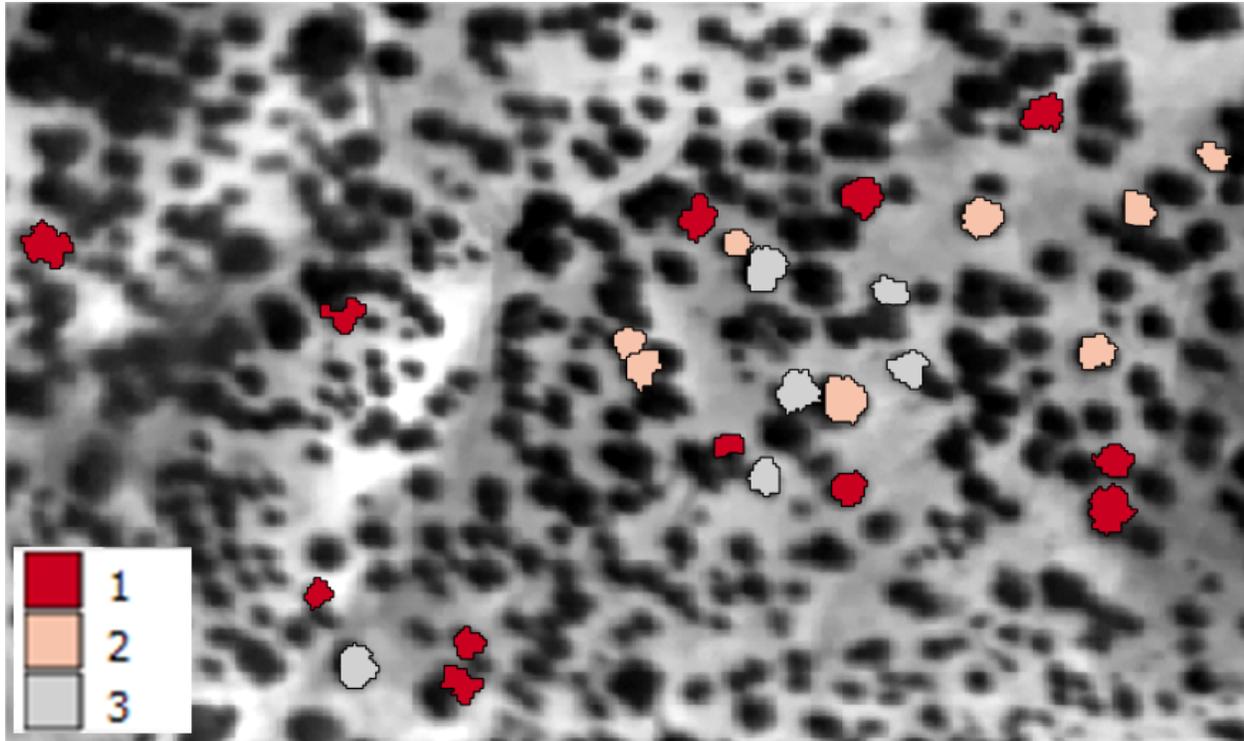
We obtain results in the dendrogram with the cluster of data:

LAI data grouped into 3 categories health condition, with the following results:

Finally, we must ensure that the groups are significantly different, according to the variables used. This is done through the tool **[Analysis] Heath condition level> ANOVA**.







ANOVA

Selected as the dependent variable the group that owns each parcel; and as the dependent variable that we want to check if it is significant in the group.

We get the following output:

If the critical level associated with the F statistics (ie, the probability of obtaining values as obtained or older), is less than 0.05, we reject the hypothesis of equal means and conclude that not all the population means being compared are equal. Otherwise, we cannot reject the hypothesis of equality and we cannot claim that the groups being compared differ in their population averages.

Structurally homogenous forest units

The analytical purpose of this tool is the definition of structurally homogeneous stands that allow us to minimize the effects of structure on the thermal information, and therefore allow us to obtain related health outcomes woodland.

We start from the metric of the objects to be classified. This vector file was generated at the point of obtaining metric lidar [**Processing**] **LiDAR > Calculate metrics**.

- `thld_training\huelva\Z1_holmOak\out\metrics.shp`

The user must enter the equation by which you want to group the stands, for example according to the equation of the dominant height. In our case, we use the equation obtained from the 95th percentile.

The tool used in this section is located in [**Analysis**] **Structurally homogeneous forest units**

In this example, the polygons will be classified into 3 different groups of homogeneity (Cluster parameter).

The Cutoff parameter determines the accuracy of the fit. A value of 0 is of higher precision and larger number of iterations will be needed to reach the final solution.

We get as output:

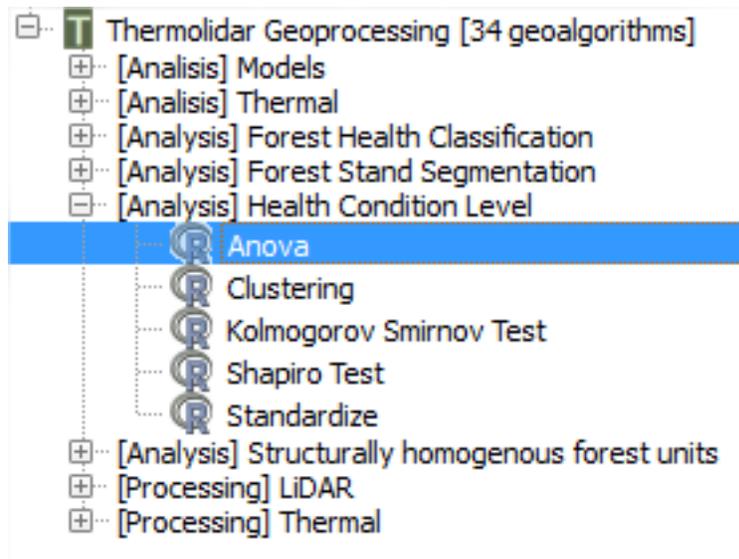
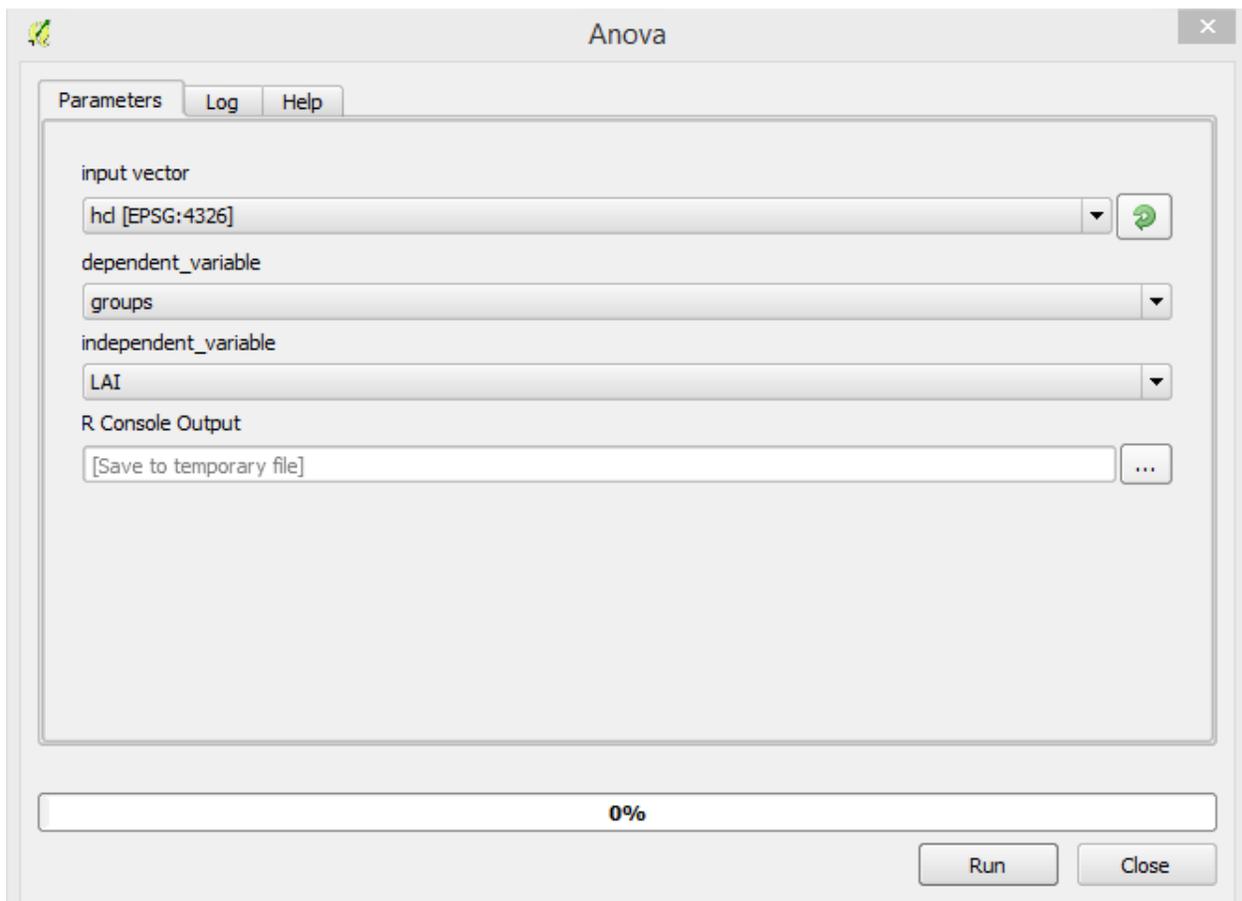


Figure 5.12: ANOVA is located in the “Health Condition Level” submenu of the THERMOLIDAR plugin toolbox



R Output

```
Df Sum Sq Mean Sq F value Pr(>F)
points[[independent_variable]] 1 5.971 5.971 13.46 0.00121 **
Residuals 24 10.644 0.444

---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

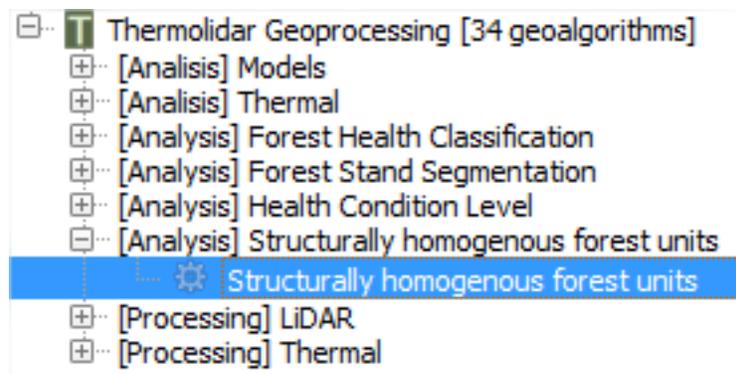


Figure 5.13: **SHFU** is located in the “Structurally Homogeneous Forest Units” submenu of the THERMOLIDAR plugin toolbox

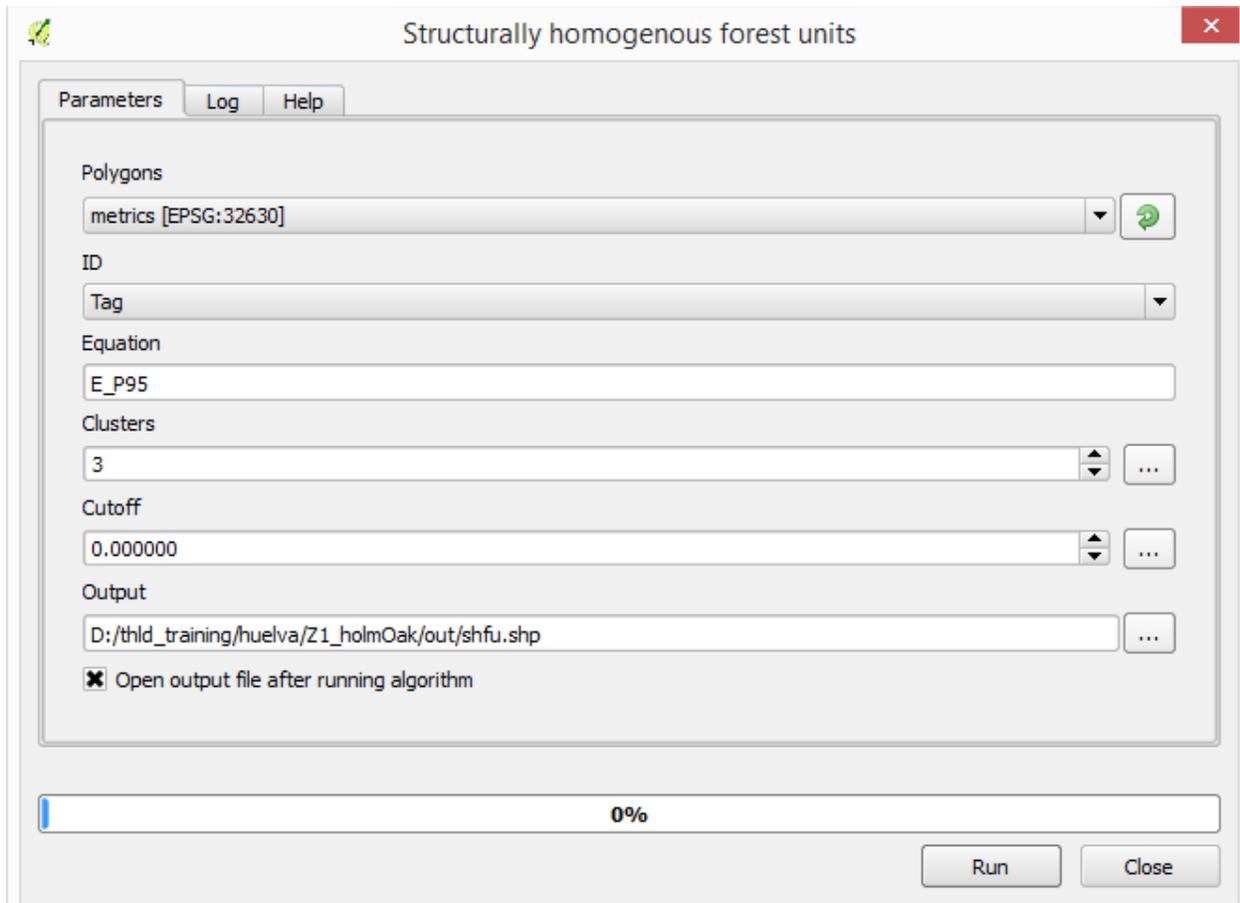
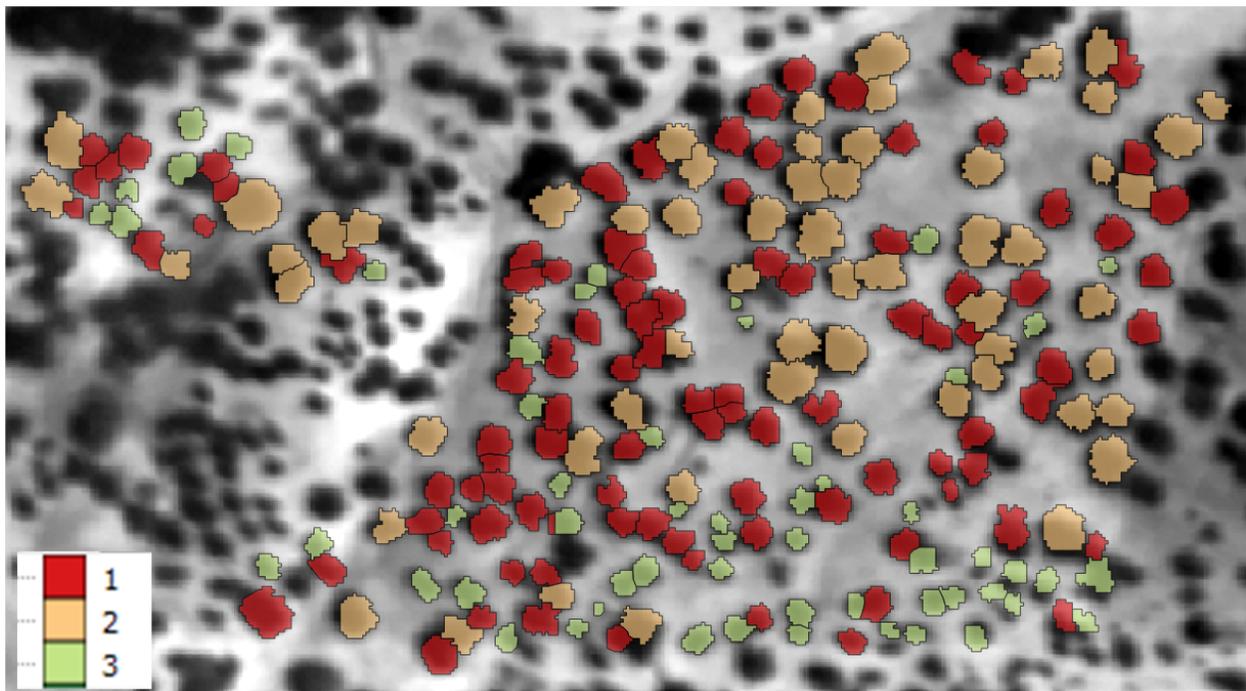


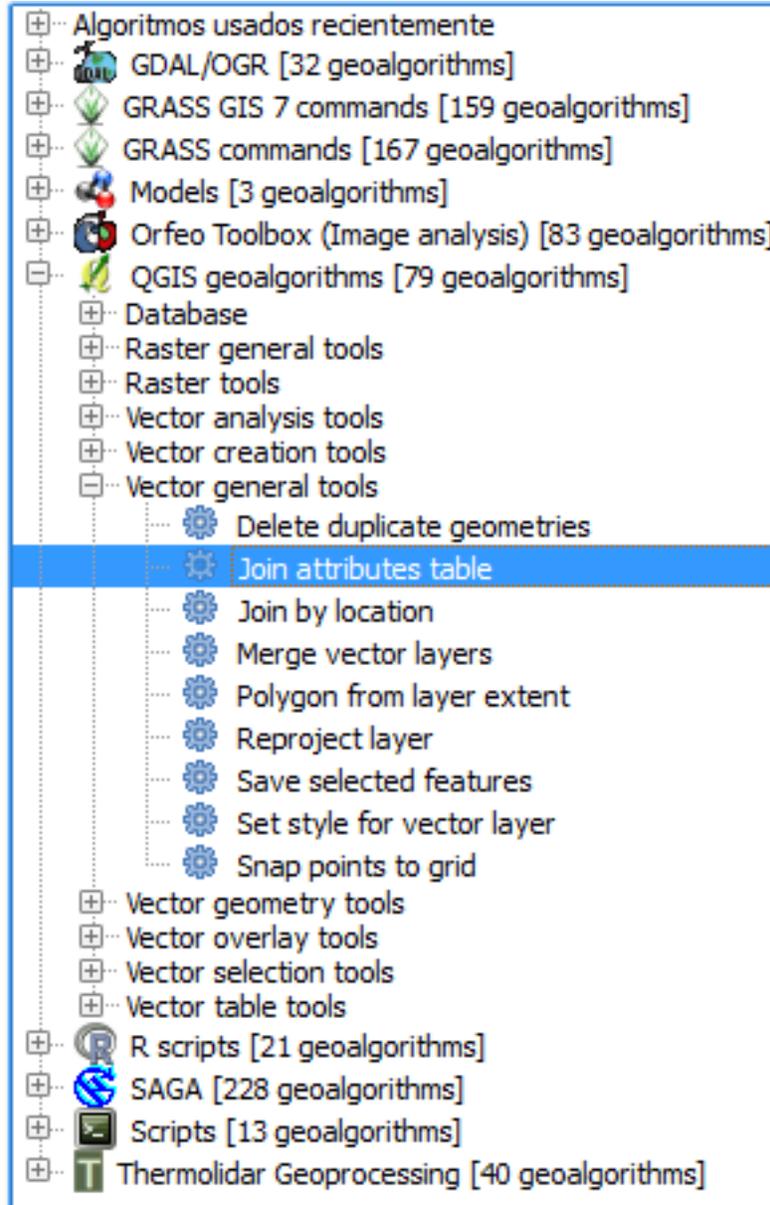
Figure 5.14: Interface of the “SHFU” module



Now, we must assign the SHFU group to the objects (regions of interest) described on the Forest Health Level section.

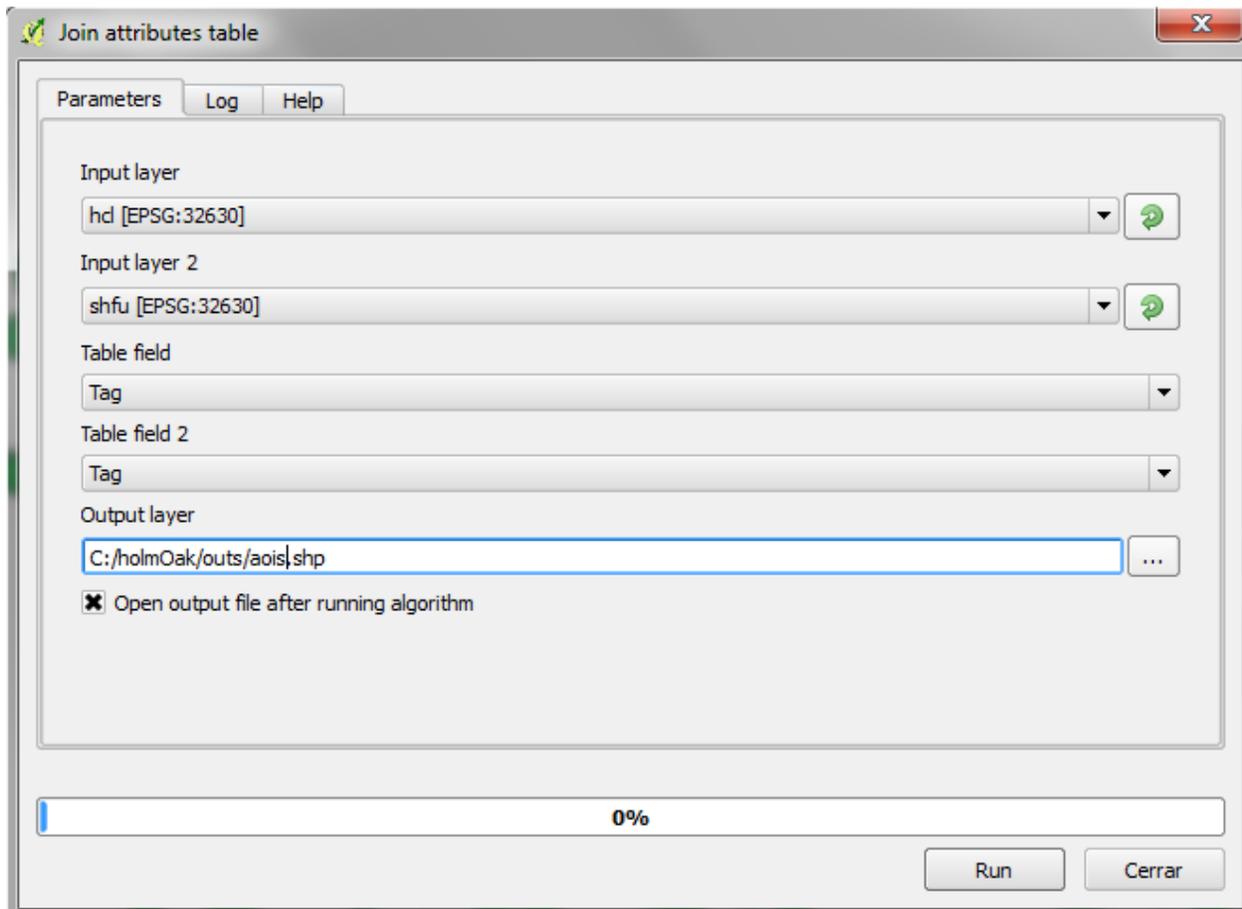
Thus we will use vector files: * **thld_training\huelva\V1_holmOak\out\hcl.shp** *
thld_training\huelva\V1_holmOak\out\shfu.shp

We will use the tool **QGIS Geographical > Vector general tools > Join attributes table**



We insert the parameters as shown in the following figure:

We will obtain a new output file (thld_training\huelva\V1_holmOak\out\aois.shp) will use to guide the classification of the thermal image, are of interest to the following fields: * **SHFU**. Homogeneous group that the object owns. * **HCL**. Health condition level groups.



Forest Health Classification

We proceed to perform the classification of the thermal image, based on the following information available:

- ****thld_training\huelva\Z1_holmOak\Temperature\t_huelvaZ1.tif**
- **thld_training\huelva\Z1_holmOak\out\shfu.shp**

For supervised classification we will use health condition levels obtained in previous sections.

- ****thld_training\huelva\Z1_holmOak\out\aois.shp**

Unsupervised pixel-based classification

Through this tool raster classify the temperature as many classes as you specify. Temperature classification is performed based on homogeneous units, so many output raster have been defined as SHFU be obtained.

In our case, we obtain three raster classified. Each raster has associated many categories defined temperature.

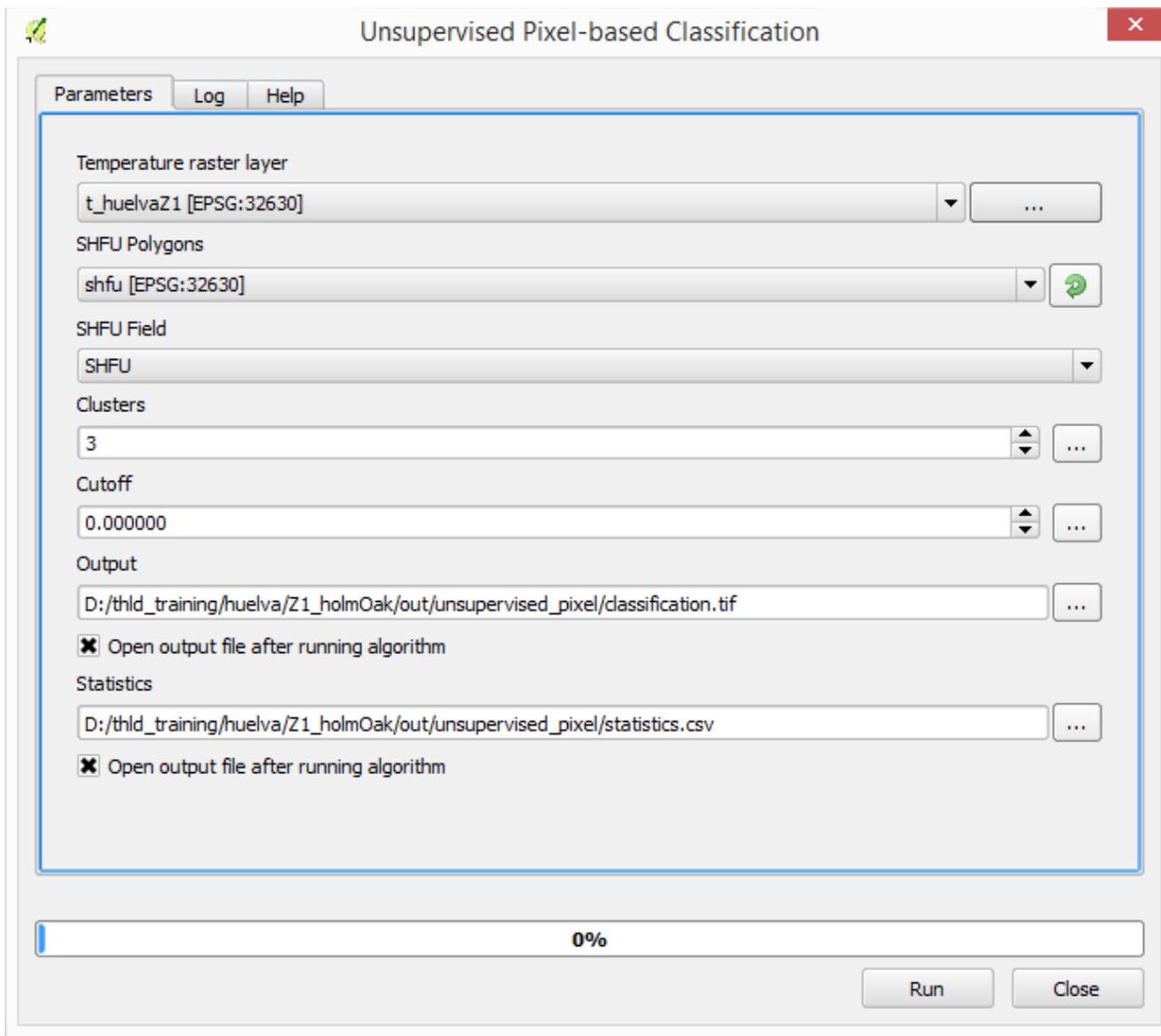
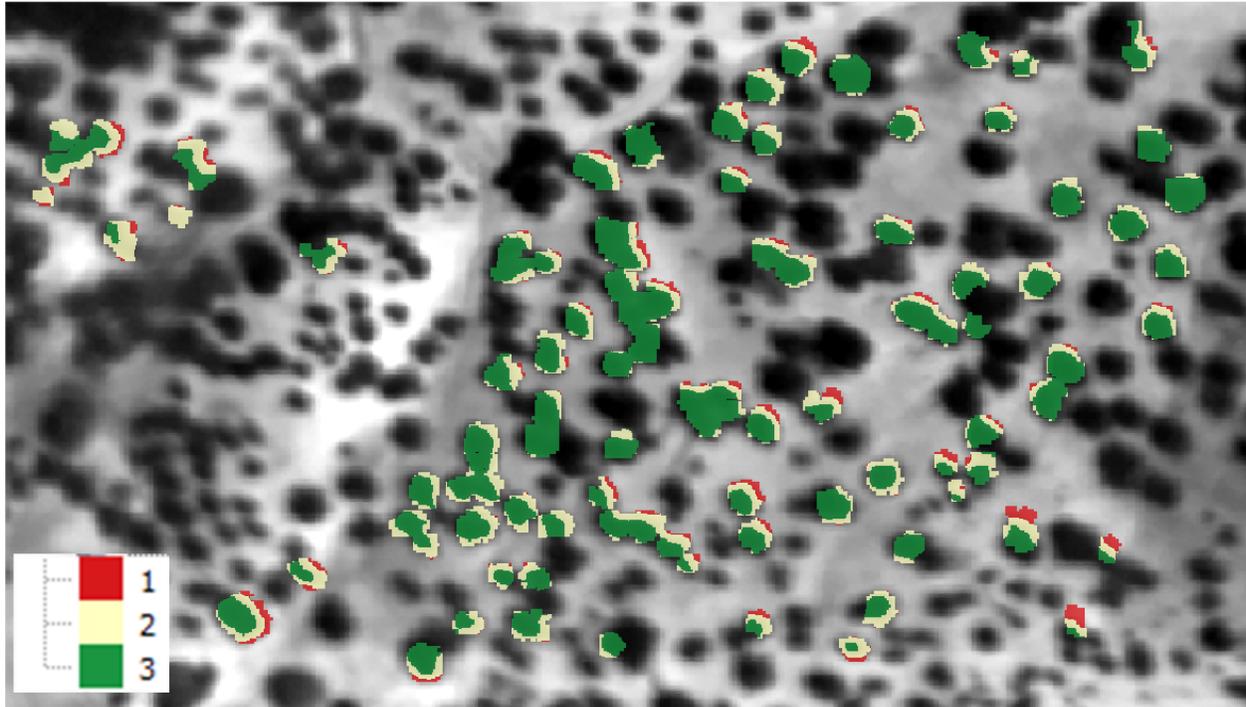


Figure 5.15: Interface of the “Unsupervised pixel-based classification” module

We get as output:



Unsupervised object-based classification

In the same way that the pixel-oriented, this tool classification raster classify the temperature as many classes as you specify. Temperature classification is performed based on homogeneous units, so many output raster have been defined as SHFU be obtained.

In our case, we obtain three raster classified. Each raster has associated many categories defined temperature.

The difference from the previous tool, is that instead of being classified pixels are classified objects. The classification is made based on the mean value of each of the objects.

We get as output:

Supervised pixel-based classification

Through this tool the temperature raster will be classified, based on the condition levels defined in the regions of interest (AOIs). * **thld_training\huelva\Z1_holmOak\out\aois.shp

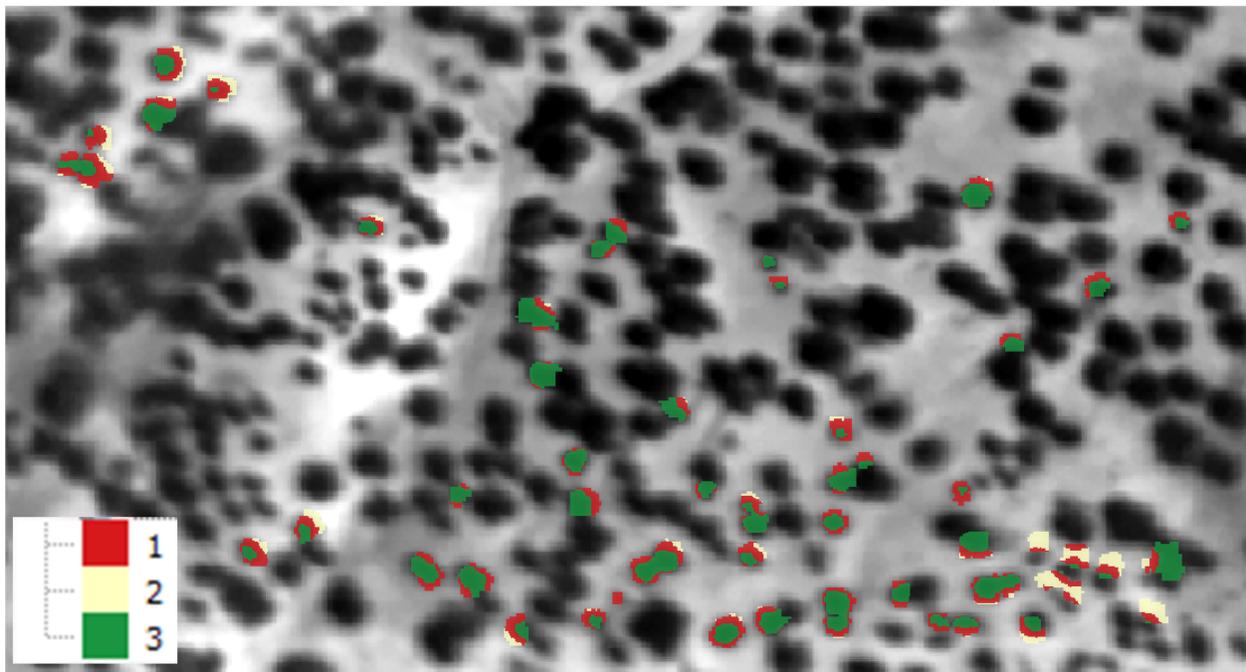
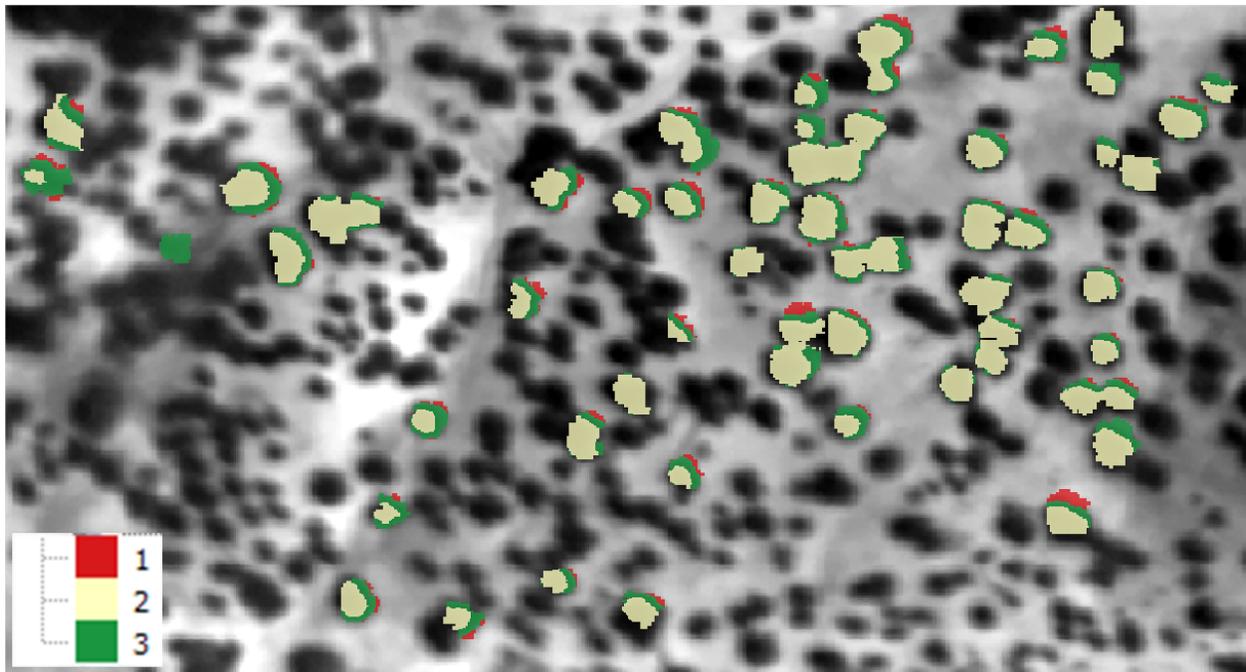
Temperature classification is performed based on homogeneous units, so many output raster have been defined as SHFU be obtained.

In our case, we obtain three raster classified. Each raster has associated many categories defined temperature.

We get as output:

Supervised object-based classification

Through this tool the temperature raster will be classified, based on the condition levels defined in the regions of interest (AOIs). * **thld_training\huelva\Z1_holmOak\out\aois.shp



	SHFU /	ID_CLUSTER	Num	Min	Max	Mean	Median	Var	Std	CV
0	1	1	511	30789	31460	31058.4324853	31044.0	29153.2537662	170.743239299	0.005497484117...
1	1	2	537	30252	30788	30520.0	30520.0	24075.5	155.162817711	0.005083971746...
2	1	3	535	29702	30251	29983.4747664	29984.0	24172.2723021	155.474346122	0.005185334499...
3	2	1	488	30740	31396	31011.6680328	30989.5	31766.6082573	178.231894613	0.0057472527574
4	2	2	539	29646	30199	29929.6085343	29930.0	24457.8408936	156.390028114	0.005225261397...
5	2	3	540	30200	30739	30469.5	30469.5	24345.0	156.028843487	0.005120820607...
6	3	1	519	30383	30903	30642.2408478	30642.0	22592.05578	150.306539379	0.004905207165...
7	3	2	477	30904	31660	31164.3165618	31144.0	29024.4563008	170.365654698	0.0054666899035
8	3	3	517	29843	30382	30123.5764023	30124.0	22532.3415351	150.107766405	0.004983065901...

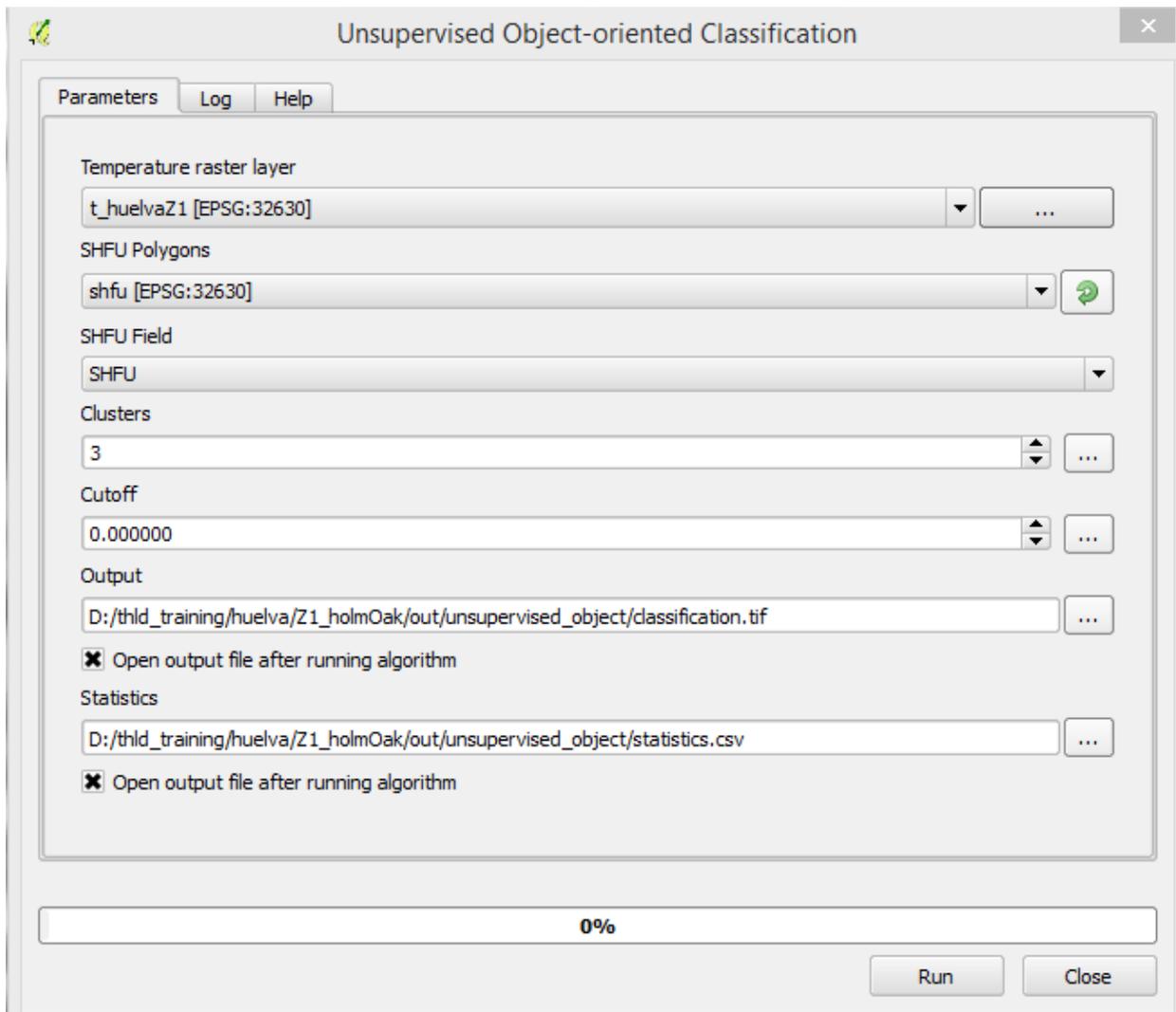
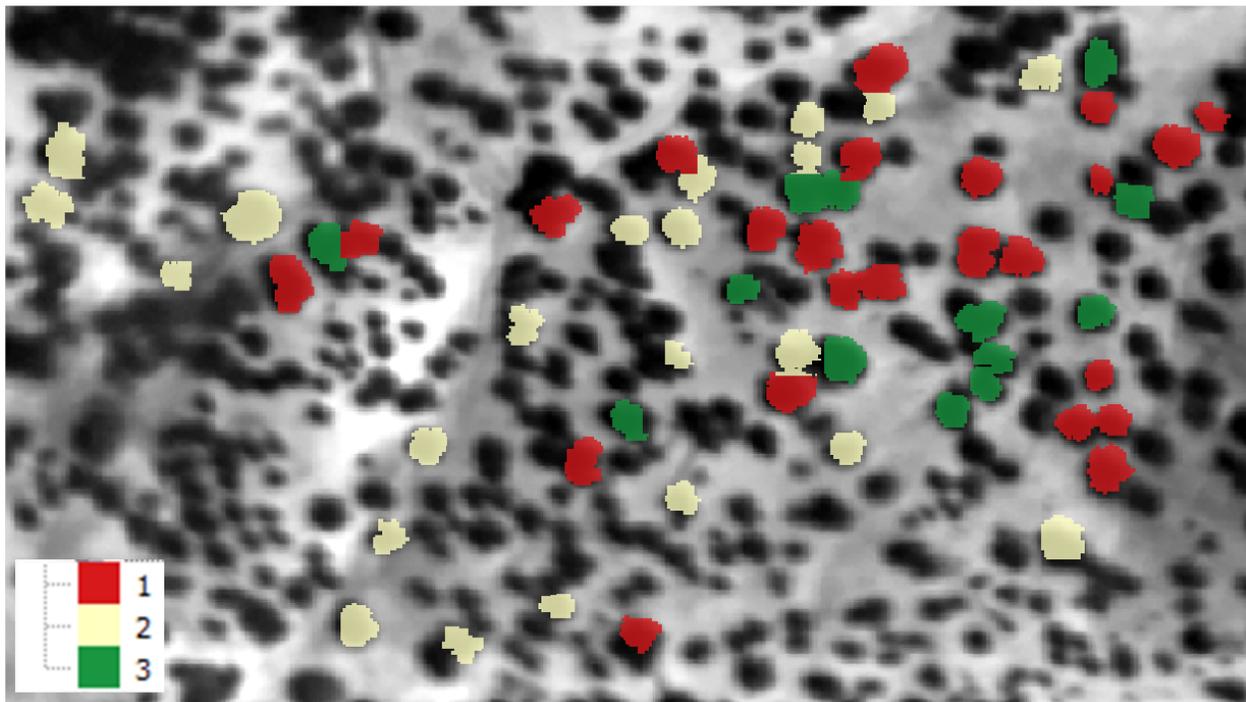
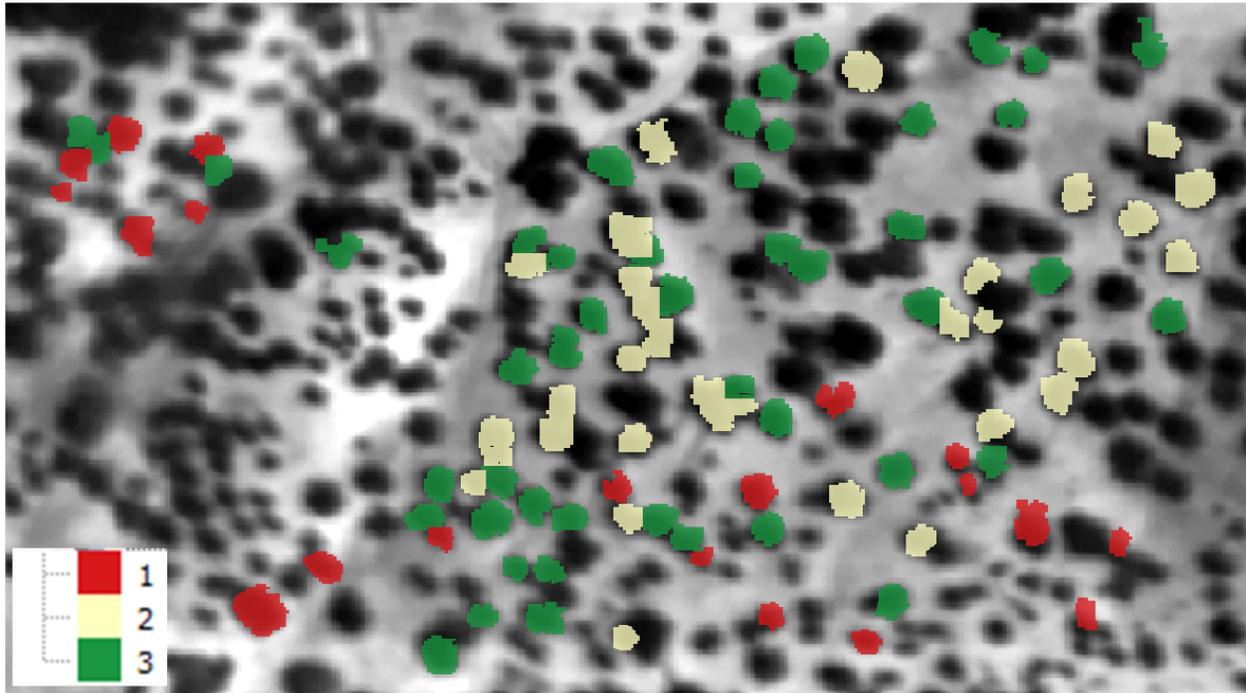
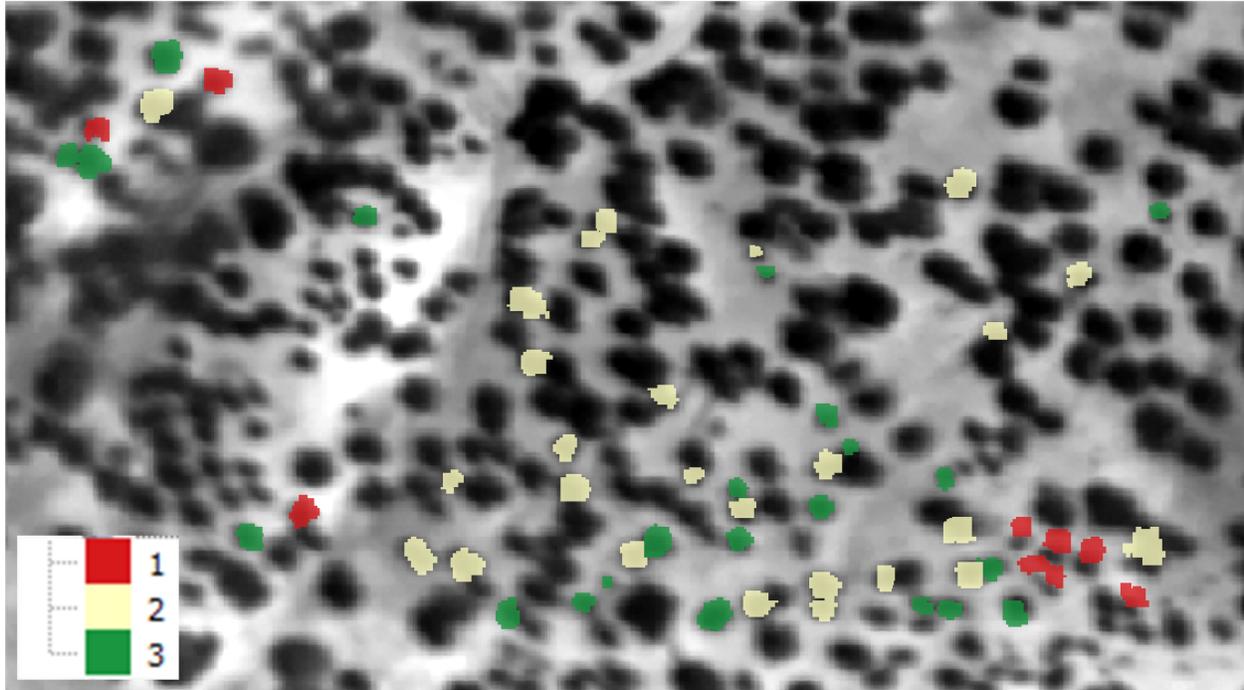


Figure 5.16: Interface of the “Unsupervised object-based classification” module





	SHFU	ID_CLUSTER	Num	Min	Max	Mean	Median	Var	Std	CV
0	1	1	20	30364.6256158	30823.1625767	30473.7348411	30452.7404905	10538.9176787	102.65923085	0.003368777453...
1	1	2	32	29908.1045045	30147.6330472	30058.2050893	30091.0334856	5629.3105279	75.0287313494	0.002496114825...
2	1	3	47	30159.4751204	30347.2754491	30245.5520914	30239.3269231	2468.16957067	49.6806760287	0.001642577919...
3	2	1	25	30033.7852217	30192.0571429	30111.9188791	30119.0623886	1886.19212471	43.4303134309	0.001442296441
4	2	2	22	30217.4928367	30447.9087912	30297.3804775	30286.8079691	3606.13764186	60.0511252339	0.001982056675...
5	2	3	13	29851.4924554	30016.1849427	29950.7063744	29950.7645688	2226.18254718	47.1824389703	0.001575336433...
6	3	1	9	30722.7982196	31100.4427481	30879.8669913	30845.9097222	19042.1251952	137.993207062	0.0044687111865
7	3	2	26	30076.3194888	30365.3936652	30272.1090968	30269.375351	3576.4235635	59.8032069667	0.001975521651...
8	3	3	22	30390.2904564	30656.5483871	30500.2360302	30493.5453438	7983.74763884	89.351819449	0.002929545179...

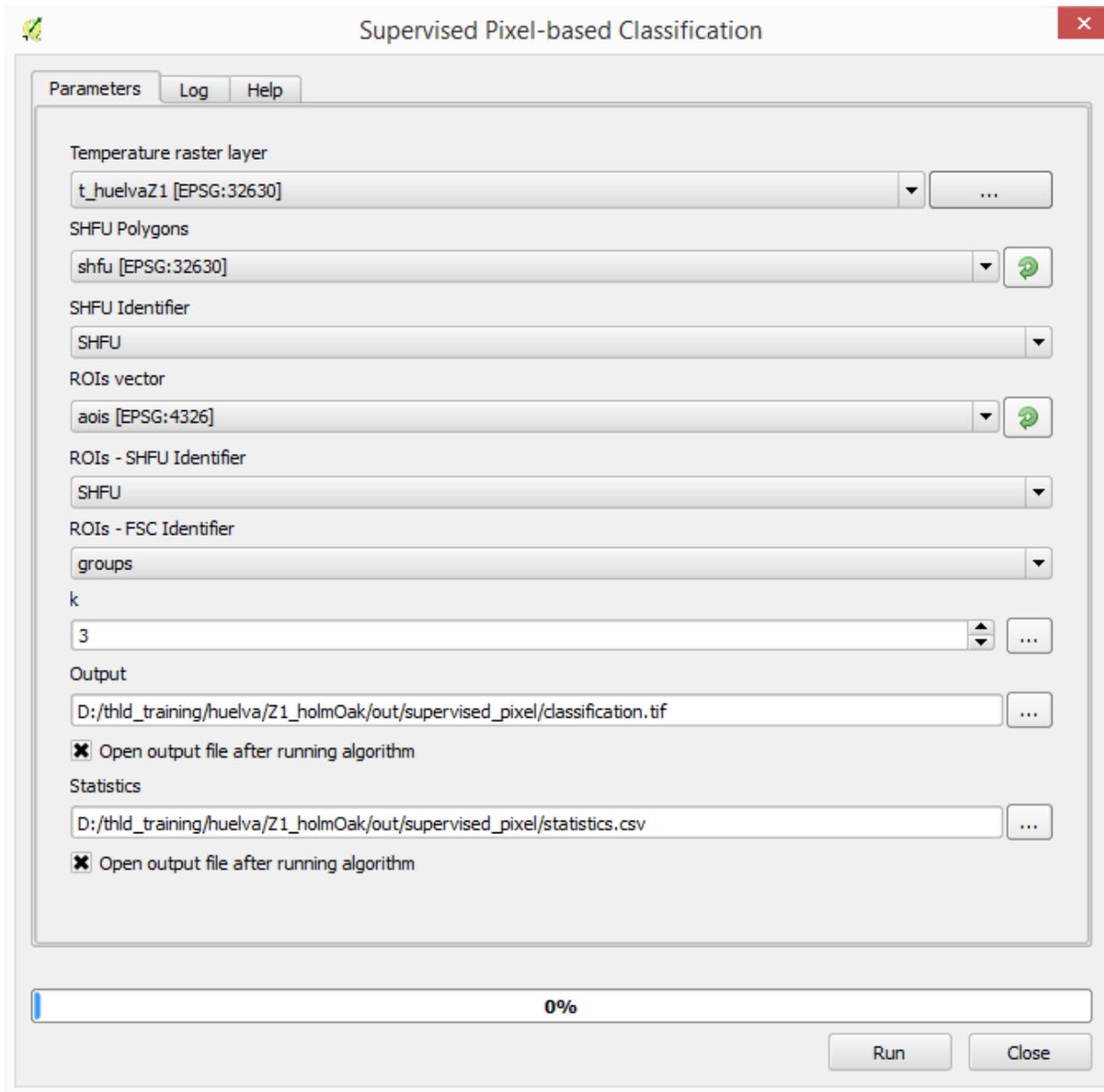
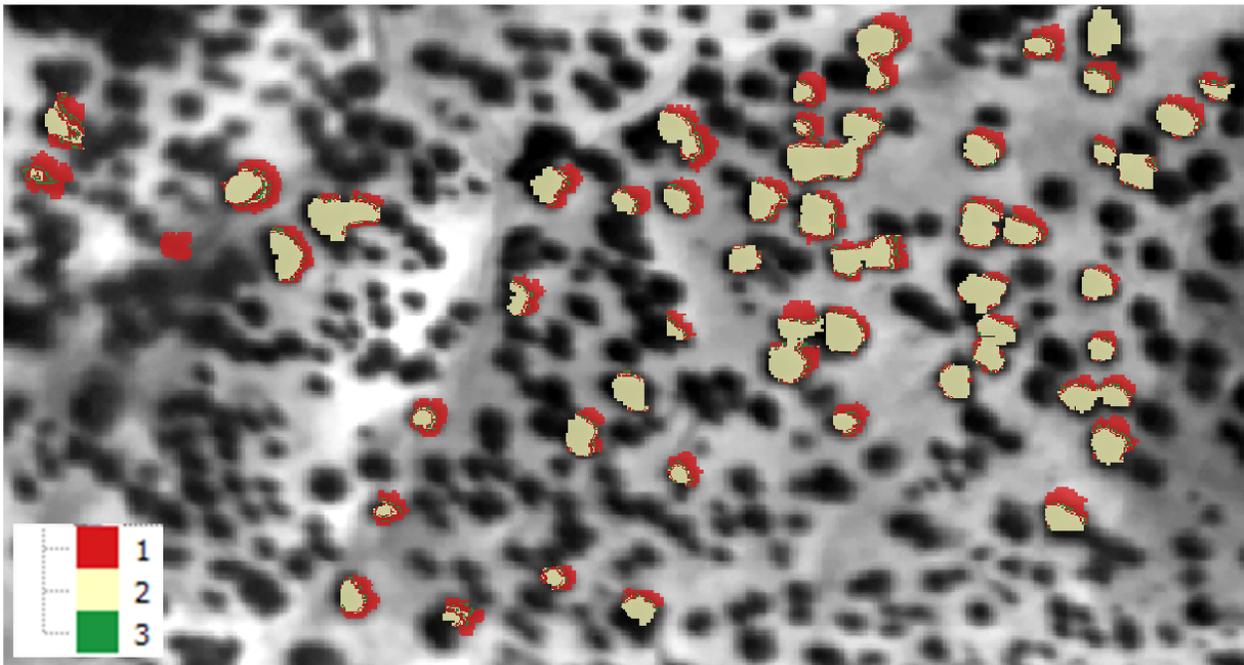
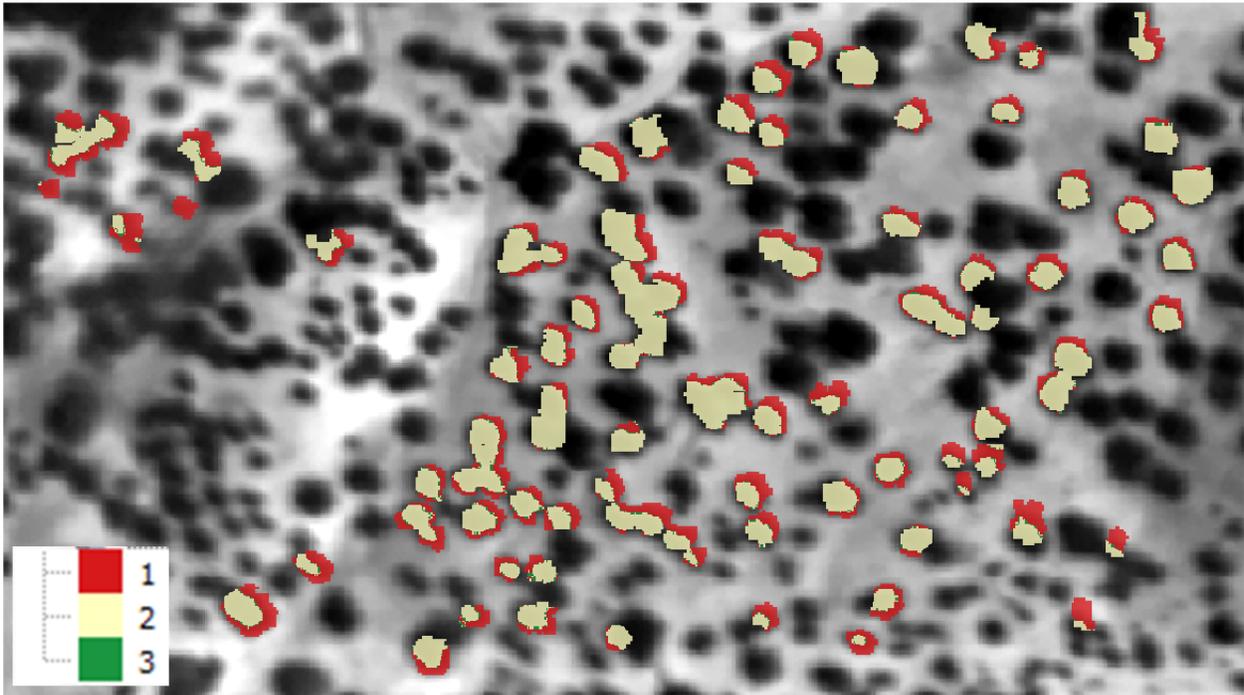
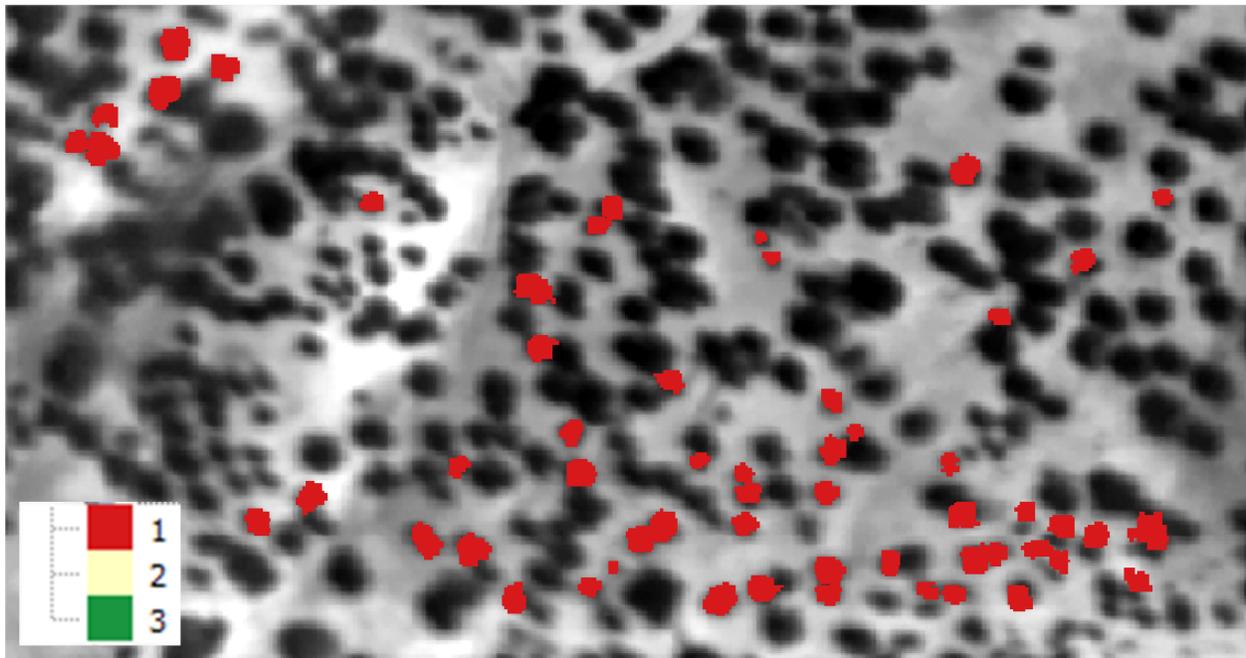


Figure 5.17: Interface of the “Supervised pixel-based classification” module





	SHFU	ID_CLUSTER	Num	Min	Max	Mean	Median	Var	Std	CV
0	1	1	17157	30250	31460	30558.6513376	30507.0	54318.6743006	233.063670057	0.007626765575...
1	1	3	1170	30229	30271	30241.4948718	30240.0	104.409297888	10.2180868017	0.000337882993...
2	1	2	28493	29702	30273	30010.2226512	30008.0	15438.585059	124.252102835	0.004140325924...
3	2	1	17235	30048	31396	30430.4779228	30392.0	70731.3594261	265.953679099	0.008739714169...
4	2	3	2213	30076	30264	30189.6525079	30227.0	4554.71599393	67.4886360355	0.002235488998...
5	2	2	22674	29646	30135	29914.9656435	29911.0	9385.75778398	96.8801206852	0.003238516862...
6	3	1	15254	29843	31660	30442.4579127	30382.0	112188.892053	334.946103206	0.0110025972333

Temperature classification is performed based on homogeneous units, so many output raster have been defined as SHFU be obtained.

In our case, we obtain three raster classified. Each raster has associated many categories defined temperature.

The difference from the previous tool, is that instead of being classified pixels are classified objects. The classification is made based on the mean value of each of the objects.

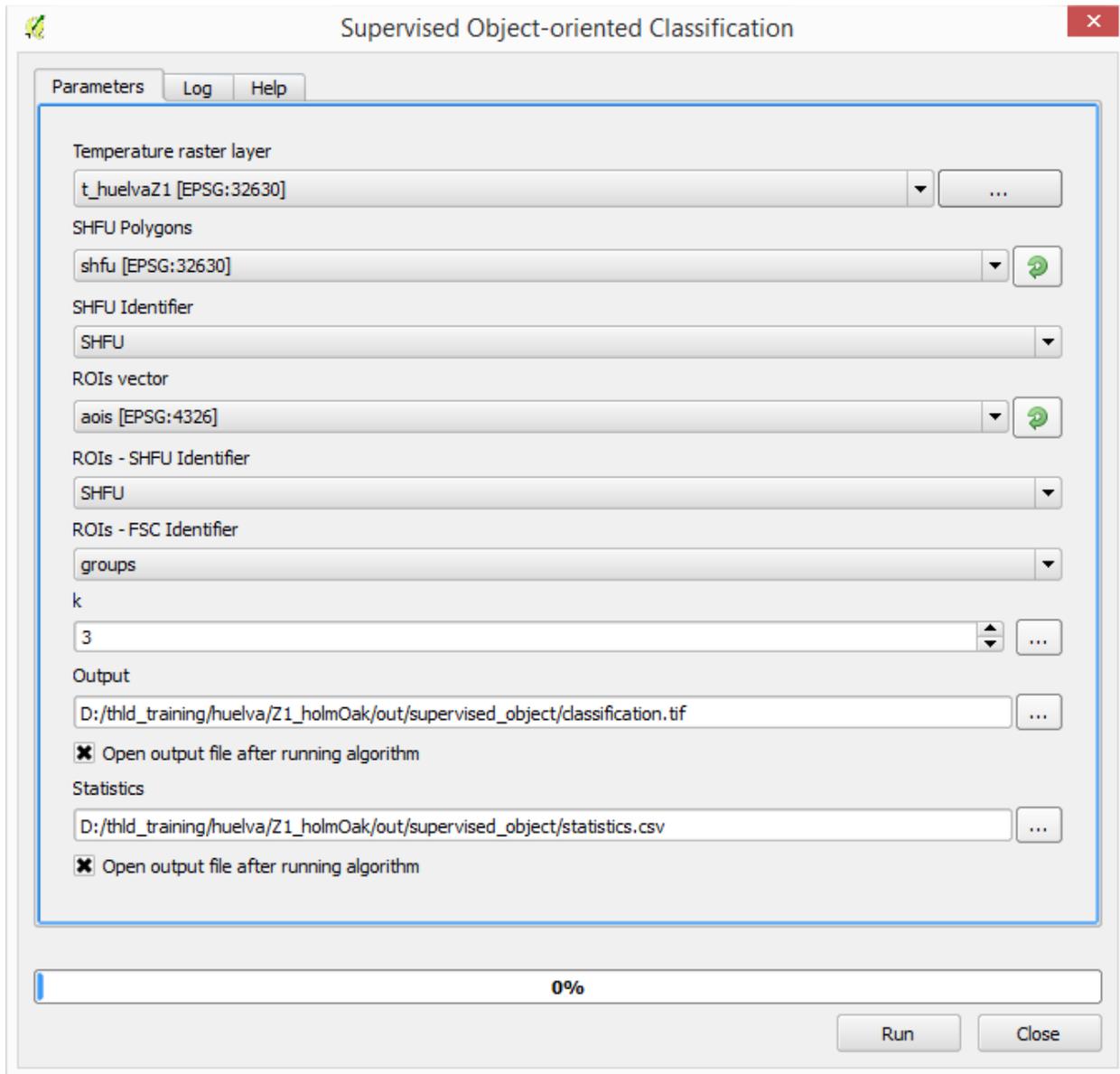
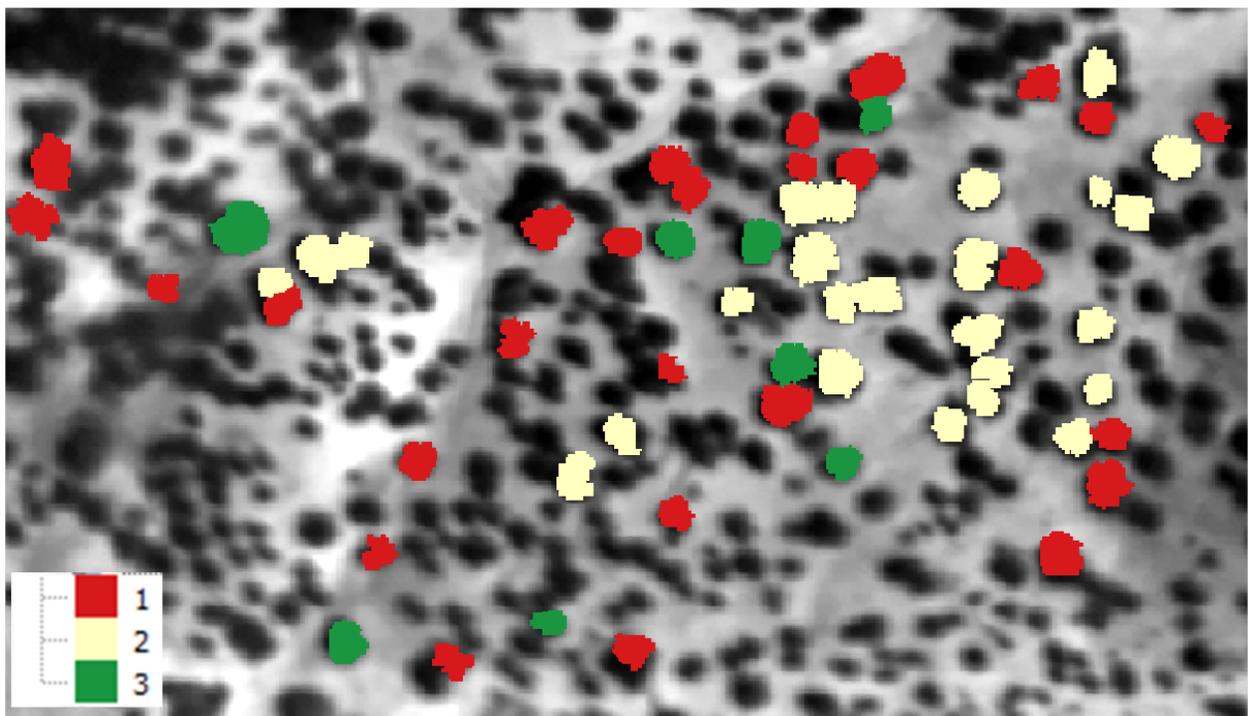
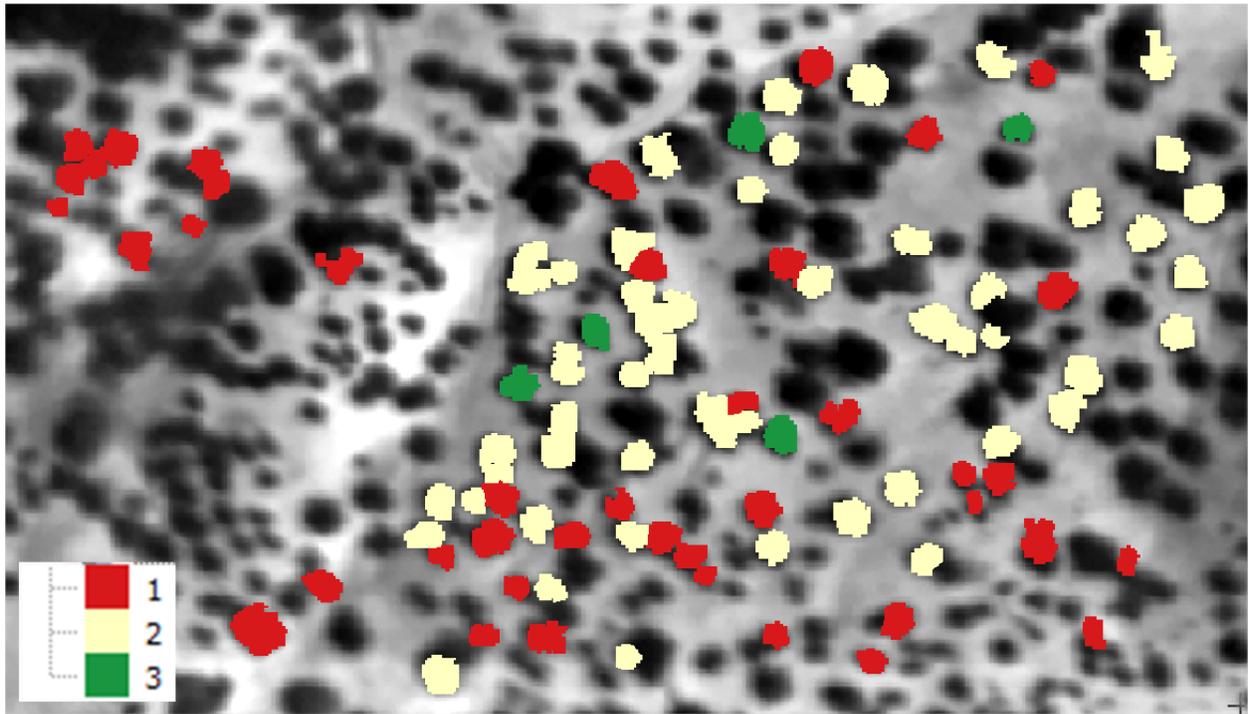
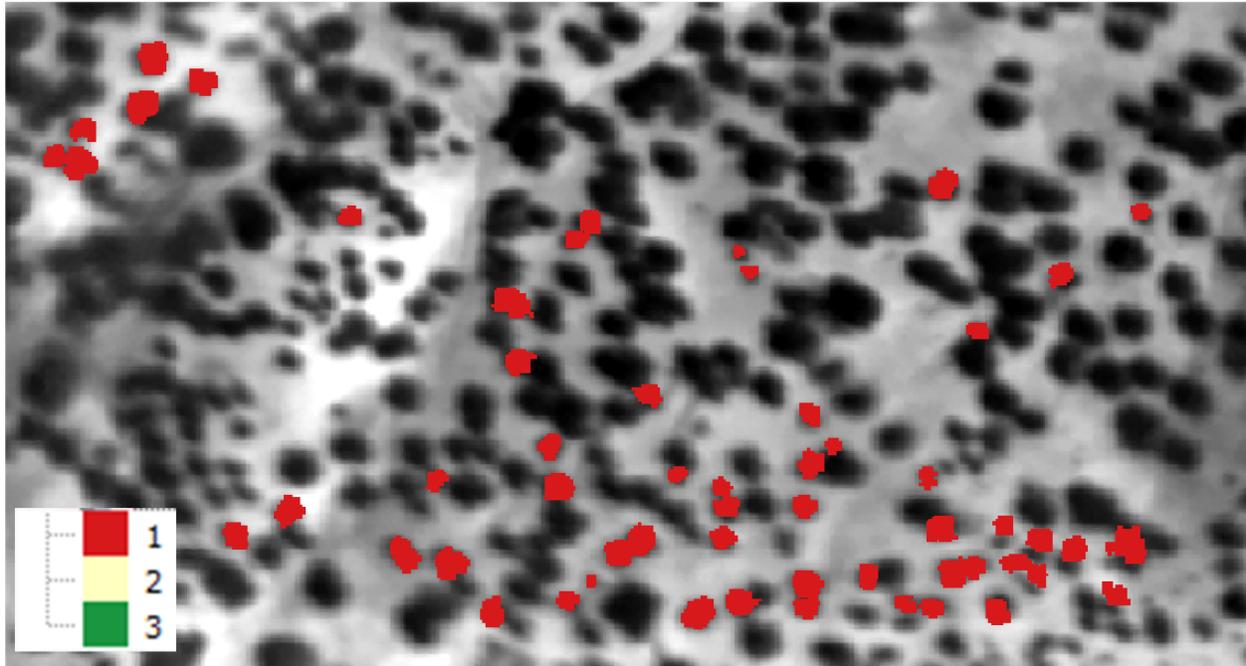


Figure 5.18: Interface of the “Supervised object-based classification” module

We get as output:





	SHFU	ID_CLUSTER	Num	Min	Max	Mean	Median	Var	Std	CV
0	1	1	42	30209.0518617	30823.1625767	30371.5873843	30338.2334476	15422.6215329	124.187847767	0.004088948206...
1	1	3	5	30203.3693548	30270.6541219	30233.7842693	30229.4658273	943.322575055	30.713556861	0.001015868757...
2	1	2	52	29908.1045045	30268.2144112	30117.3580094	30122.299718	9691.83589792	98.4471223445	0.003268783480...
3	2	1	27	30040.3016815	30447.9087912	30237.6018659	30269.1001855	11982.8881966	109.466379298	0.003620207044...
4	2	3	8	30071.1450472	30256.7344702	30223.0064683	30245.7502228	3879.90094025	62.2888508503	0.002060974672...
5	2	2	25	29851.4924554	30133.5325843	30020.008929	30016.1849427	6965.21560024	83.457867216	0.002780074696...
6	3	1	57	30076.3194888	31100.4427481	30456.1198615	30398.9675325	52654.2350153	229.465106313	0.007534285633...

5.2 Case Study 2 - Almería (Spain)

5.2.1 Thermal Processing

Thermal Calibration

First, we have a thermal image of the study area. Each digital image value represents the temperature in degrees kelvin.

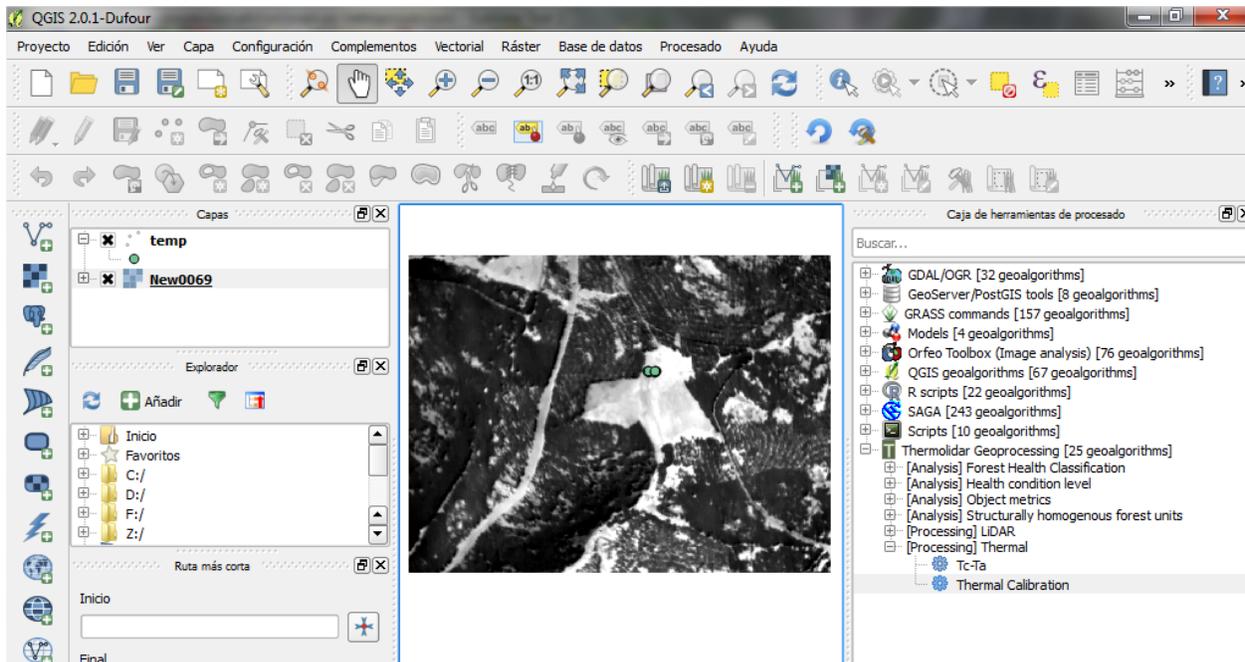


Figure 5.19: Thermal image of Sierra de los Filabres

Several values of temperature field of invariant surfaces (black and white cloth) have been collected, and GPS position of each sample.

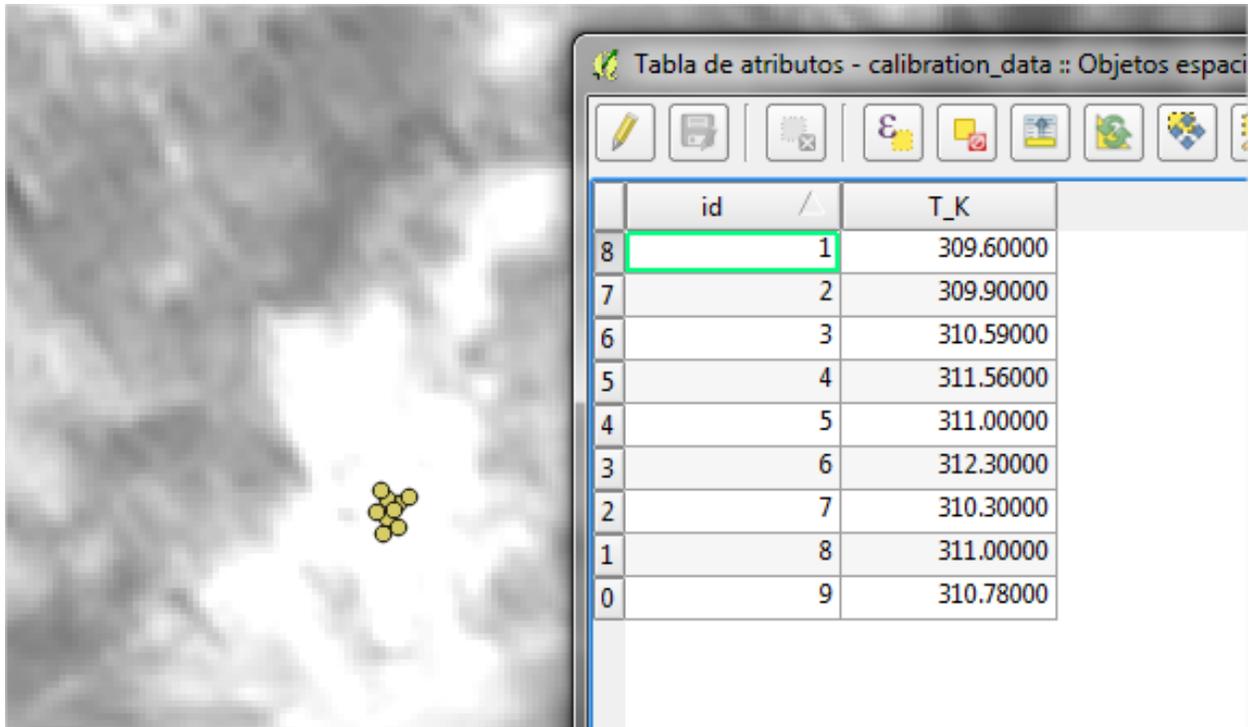
We introduce the input parameters in the user interface:

As a result the software generates a calibrated image of the study area.

Tc - Ta

We introduce the input parameters in the user interface:

- **Input layer:** Calibrated thermal raster (generated in the previous section)
- **Air temperature:** Constant air temperature measure at flight time. In this case, the air temperature is considered to 293.15 degrees kelvin



	id	T_K
8	1	309.60000
7	2	309.90000
6	3	310.59000
5	4	311.56000
4	5	311.00000
3	6	312.30000
2	7	310.30000
1	8	311.00000
0	9	310.78000

Figure 5.20: Attribute table of shapefile containing the field thermal data.

5.2.2 Data analysis

Health condition levels

In the following example we have acquired several variables physiology field (LAI), for a number of control plots in Filabres.

Before proceeding with the classification of items by level of damage according to several variables taken in the field, we verify that the set of physiological variables follow a normal distribution. For this we use the Shapiro test, located in the toolbox [Analysis] Health Condition Level > Shapiro Test.

Warning: The first time you use these tools, you will need to start QGIS in administrator mode (R install required dependencies)

Shapiro Test

- **Input vector:** Vector file that contains information on physiological data.
- **Var:** Vector's field to analyze if it follows the normal distribution. In this case the parameter lai_LICOR2

Obtaining the following output:

In the example, the p-value is much higher than 0.05, so we conclude that LAI data follow a normal distribution. In the case where p-value is less than 0.05 the data would be discarded, or these should be normalized.

In the case that the variable does not follow a normal distribution, it is necessary to standardize using the [Analysis] Health Condition Level> Standarize

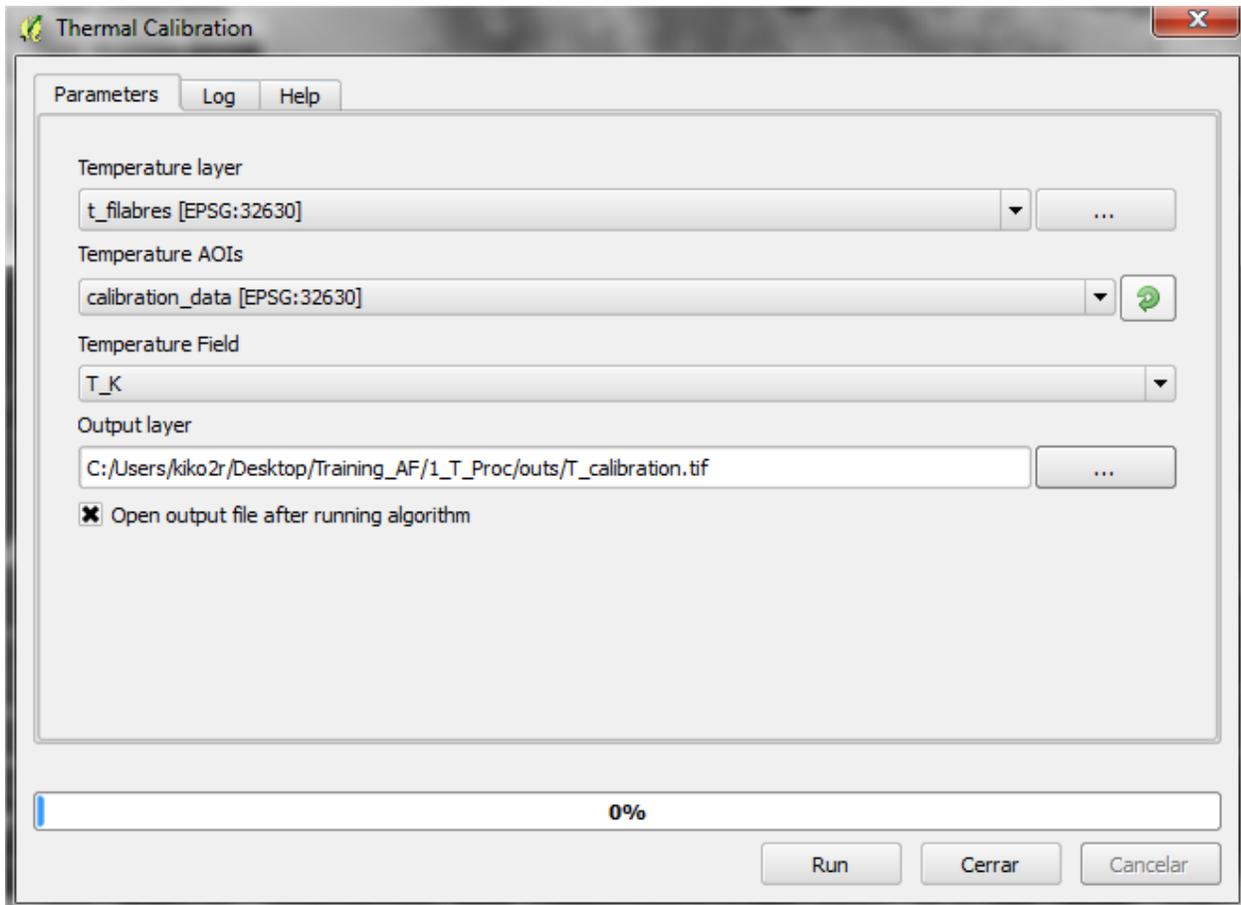


Figure 5.21: Interface of the Thermal Calibration module

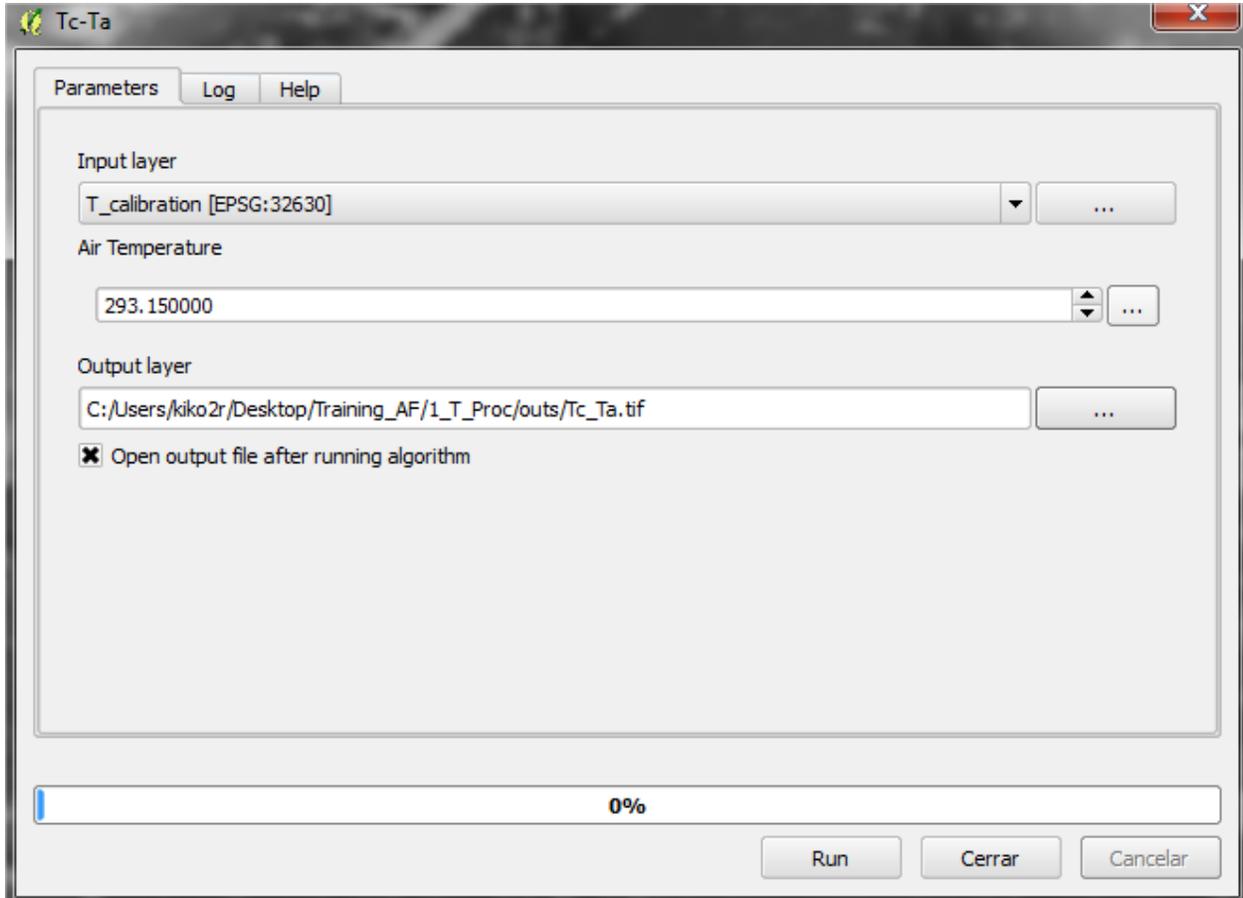


Figure 5.22: Interface of the Tc-Ta module

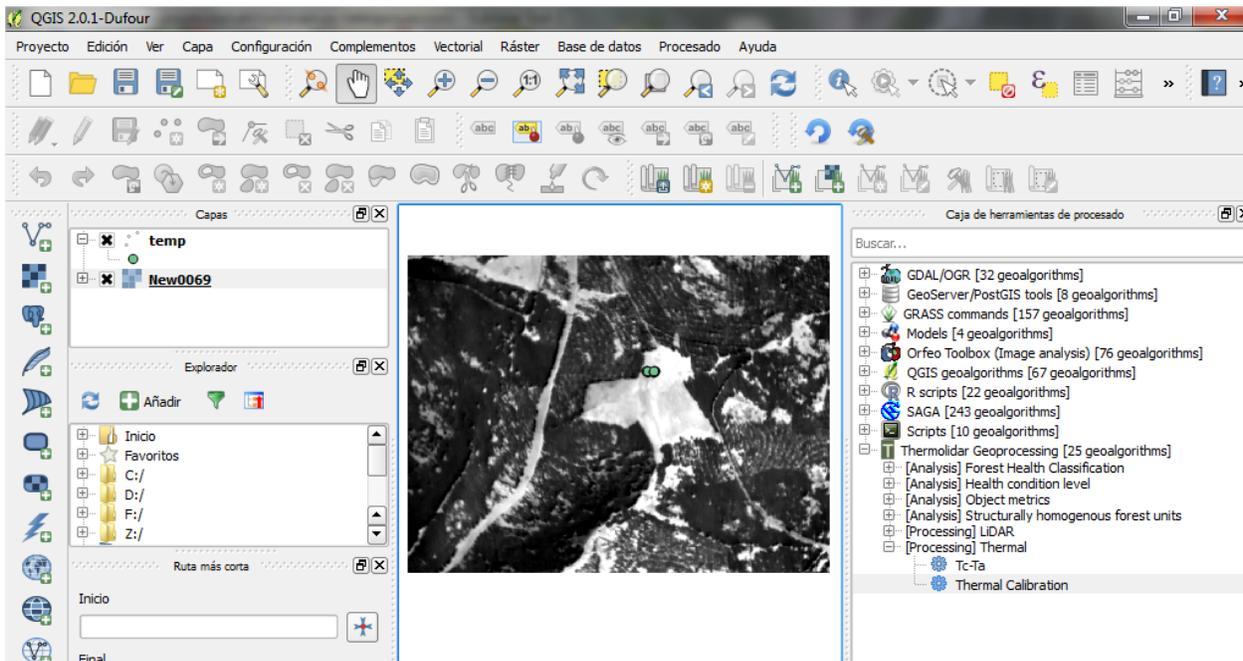


Figure 5.23: Output thermal image of Sierra de los Filabres

ID	lai_Locali	lai_Parcel	lai_5Ring	lai_Desv_t	lai_LICOR2	lai_Error_	lai_DIFN	
0	1	Filabres_junio_2...	Parcela_15_PN	1.402500000000...	0.089880874000...	2.110000000000...	0.060000000000...	0.310000000000...
1	2	Filabres_junio_2...	Parcela_15_PN_60	1.155000000000...	0.032950179000...	1.770000000000...	0.030000000000...	0.315000000000...
2	3	Filabres_junio_2...	Parcela_15_PN_30	1.170000000000...	0.070710678000...	1.840000000000...	0.020000000000...	0.316000000000...
3	6	Filabres_junio_2...	Parcela_25_PS	1.443750000000...	0.113884340000...	1.850000000000...	0.050000000000...	0.322000000000...
4	7	Filabres_junio_2...	Parcela_24_PS	1.531250000000...	0.057678295000...	1.710000000000...	0.060000000000...	0.328000000000...
5	21	Filabres_junio_2...	Parcela_1_PS_115	1.491250000000...	0.063569422000...	2.310000000000...	0.170000000000...	0.262000000000...
6	24	Filabres_junio_2...	Parcela_5_PS_115	1.341250000000...	0.081492769000...	2.250000000000...	0.040000000000...	0.257000000000...
7	28	Filabres_junio_2...	Parcela_10_PS	1.861250000000...	0.087085754000...	2.580000000000...	0.040000000000...	0.216000000000...
8	30	Filabres_junio_2...	Parcela_9_PS	1.537500000000...	0.088600226000...	2.290000000000...	0.040000000000...	0.284000000000...
9	31	Filabres_junio_2...	Parcela_11_PS	1.167500000000...	0.039910614000...	1.680000000000...	0.030000000000...	0.347000000000...
10	48	Filabres_junio_2...	Parcela_13_PN_...	1.363750000000...	0.106762420000...	1.940000000000...	0.050000000000...	0.294000000000...
11	60	Filabres_junio_2...	Parcela_21_PS	1.227500000000...	0.095580931000...	1.460000000000...	0.060000000000...	0.396000000000...
12	61	Filabres_junio_2...	Parcela_21_PS_30	0.871250000000...	0.280939114000...	1.470000000000...	0.100000000000...	0.313000000000...
13	62	Filabres_junio_2...	Parcela_21_PS_60	1.078750000000...	0.069987244000...	1.040000000000...	0.050000000000...	0.451000000000...

Figure 5.24: LAI measurements in Filabres

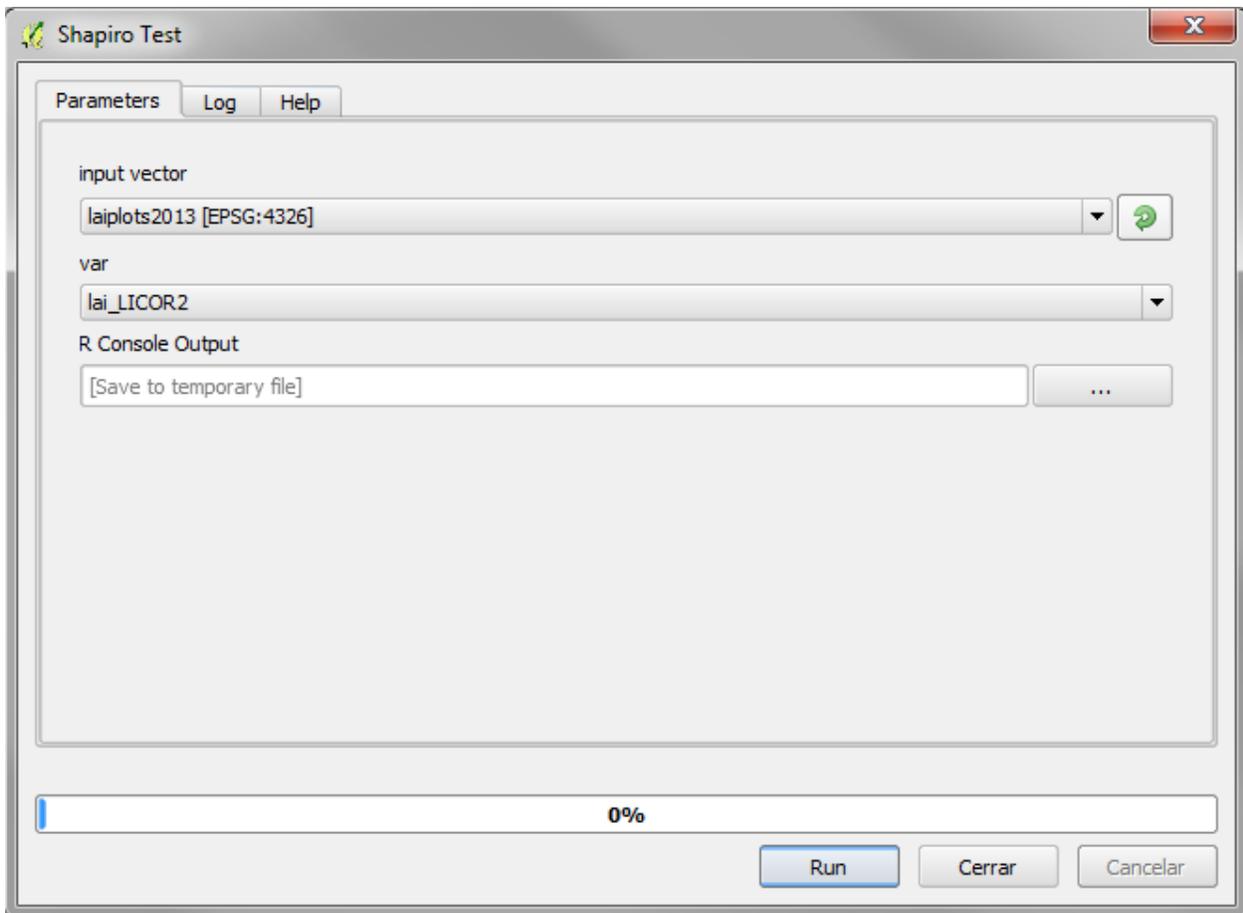


Figure 5.25: Interface of the “Shapiro Test” module

R Output

```
Shapiro-Wilk normality test  
  
data:  FAO[[var]]  
  
W = 0.9785, p-value = 0.9653
```

Standardize

We verified that the lai_LICOR2 variable follows a normal distribution.

- **Input vector:** Vector file that contains information on physiological data.
- **Var:** Shapefile's field to standardize. In this case the parameter lai_LICOR2

Clustering

This tool allows us to group one or more physiological variables according to their degree of similarity between individuals in the sample. In this case we will group by the variable lai_LICOR2, we have previously verified that follows a normal distribution. This tool creates many groups tool damage level depending on the specified physiological variable. In this case, we will select 3 levels as a function of LAI variable.

LAI data grouped into 3 categories health condition, with the following results:

Finally, we must ensure that the groups are significantly different, according to the variables used. This is done through the tool [Analysis] Heath condition level> ANOVA.

ANOVA

Selected as the dependent variable the group that owns each parcel; and as the dependent variable that we want to check if it is significant in the group.

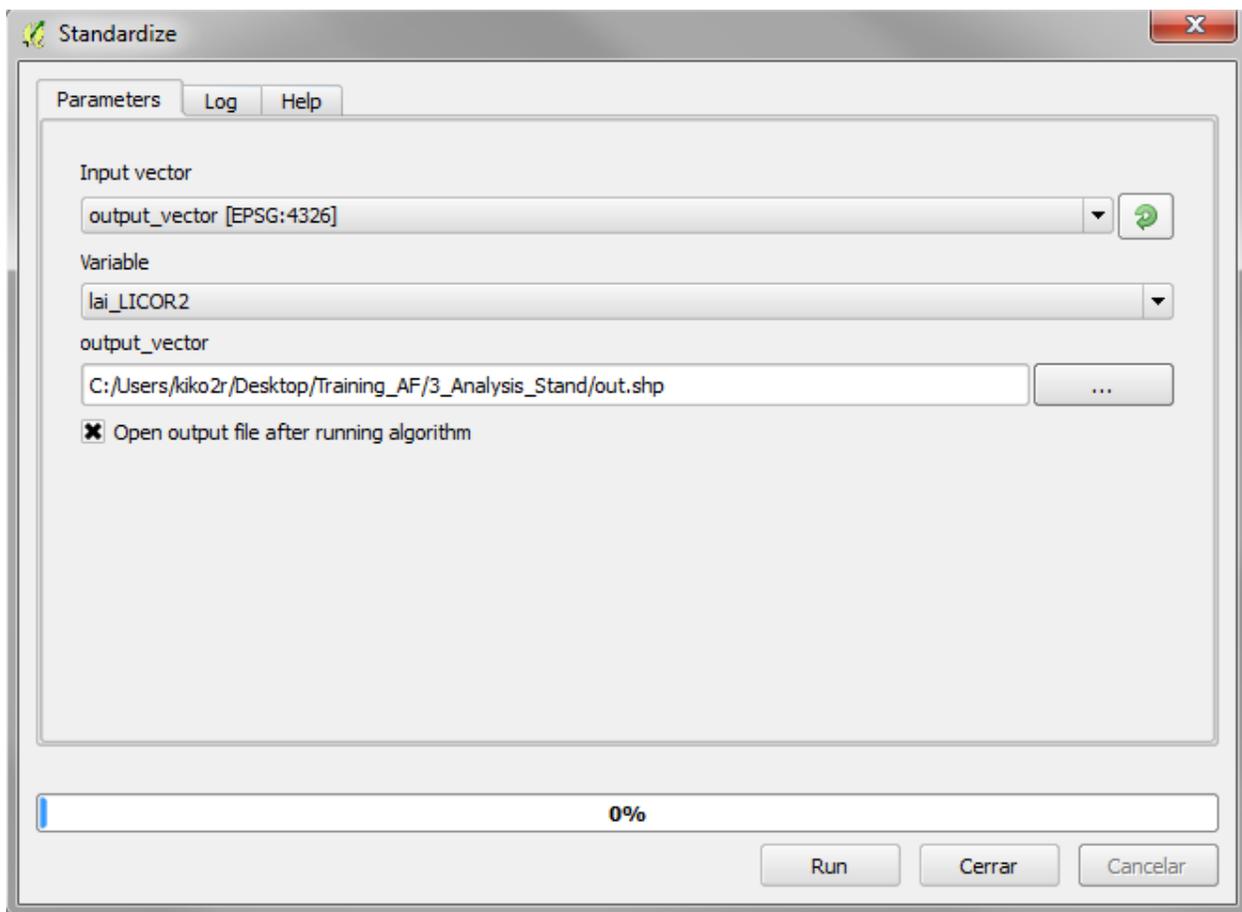
We get the following output:

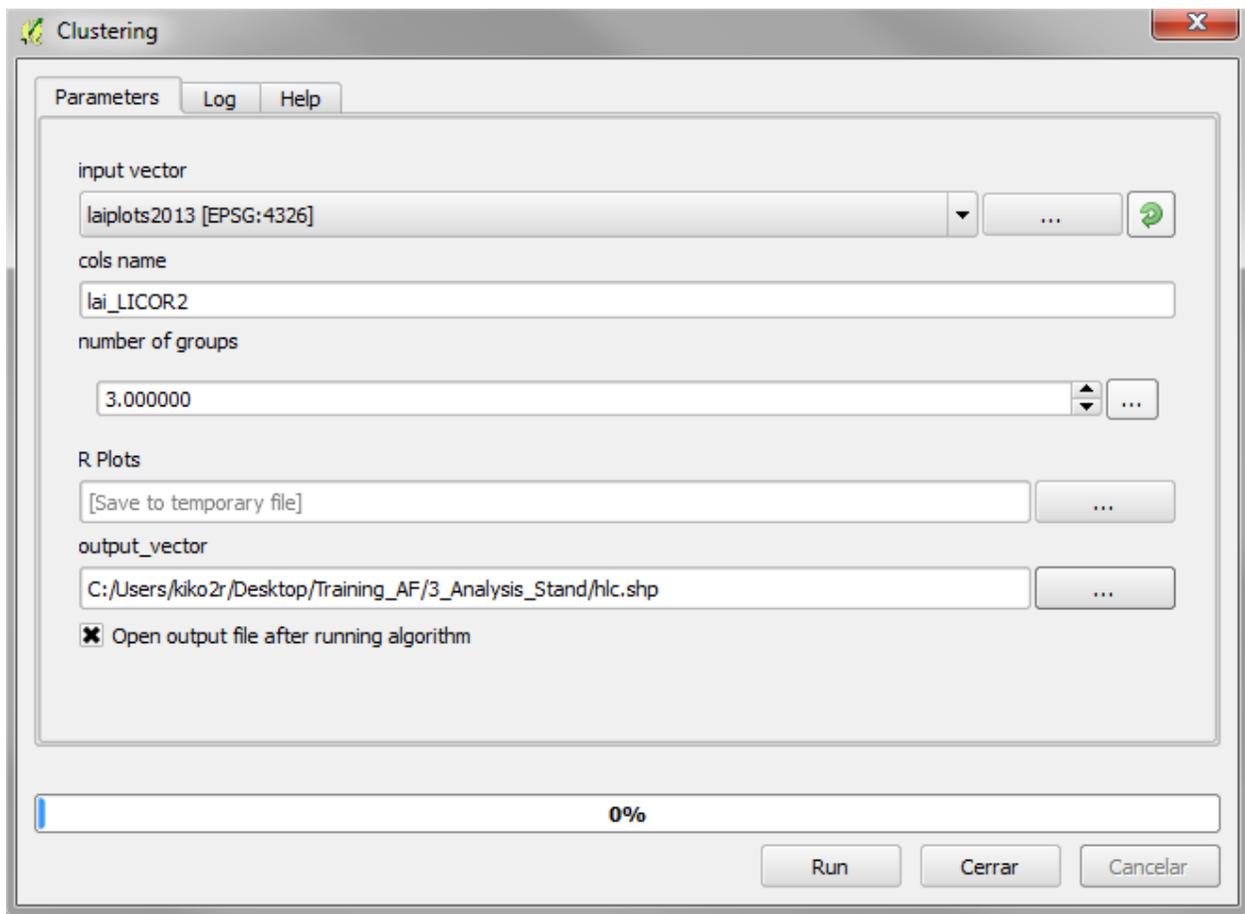
If the critical level associated with the F statistics (ie, the probability of obtaining values as obtained or older), is less than 0.05, we reject the hypothesis of equal means and conclude that not all the population means being compared are equal. Otherwise, we cannot reject the hypothesis of equality and we cannot claim that the groups being compared differ in their population averages.

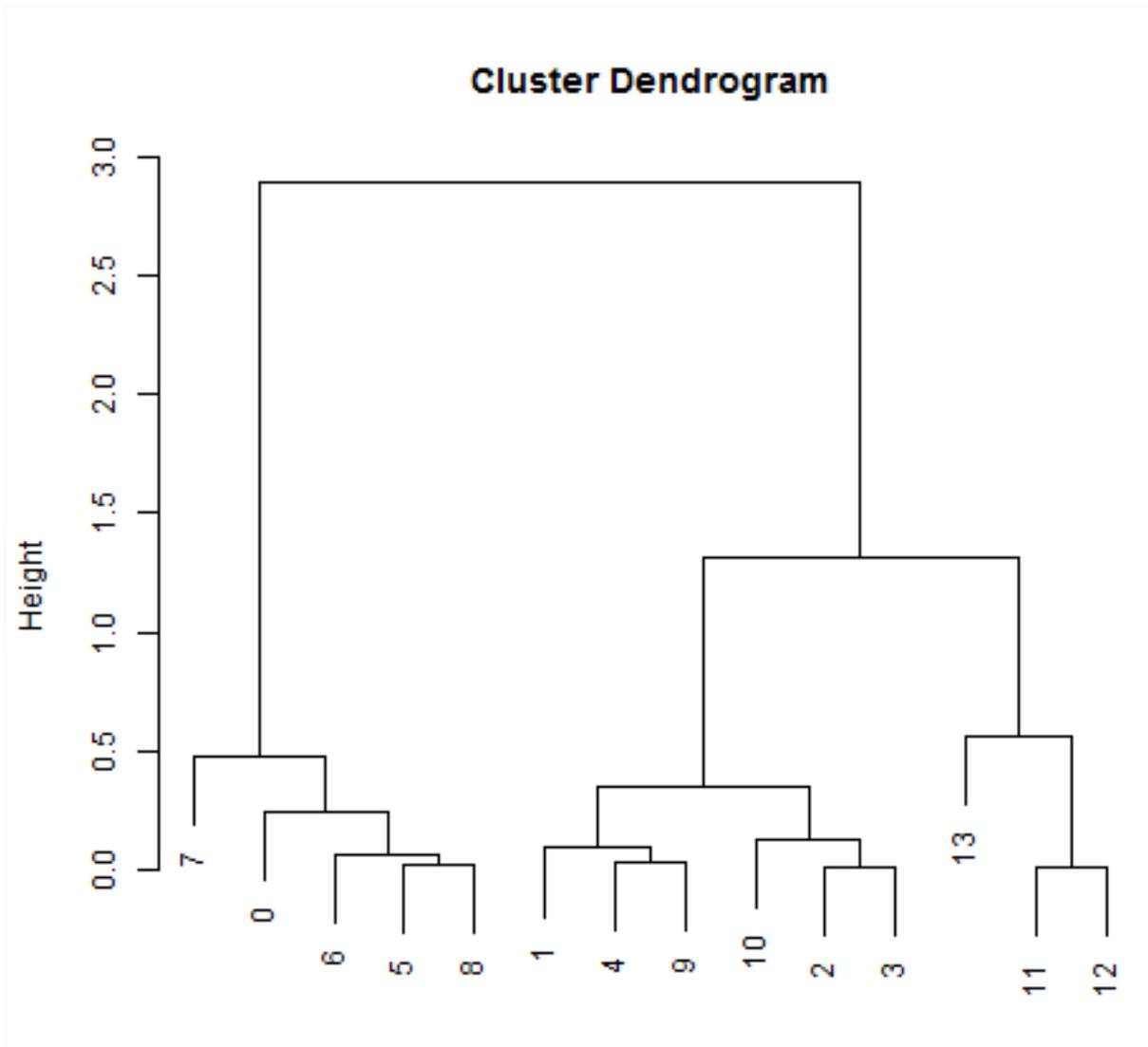
Structurally homogenous forest units

The analytical purpose of this tool is the definition of structurally homogeneous stands that allow us to minimize the effects of structure on the thermal information, and therefore allow us to obtain related health outcomes woodland.

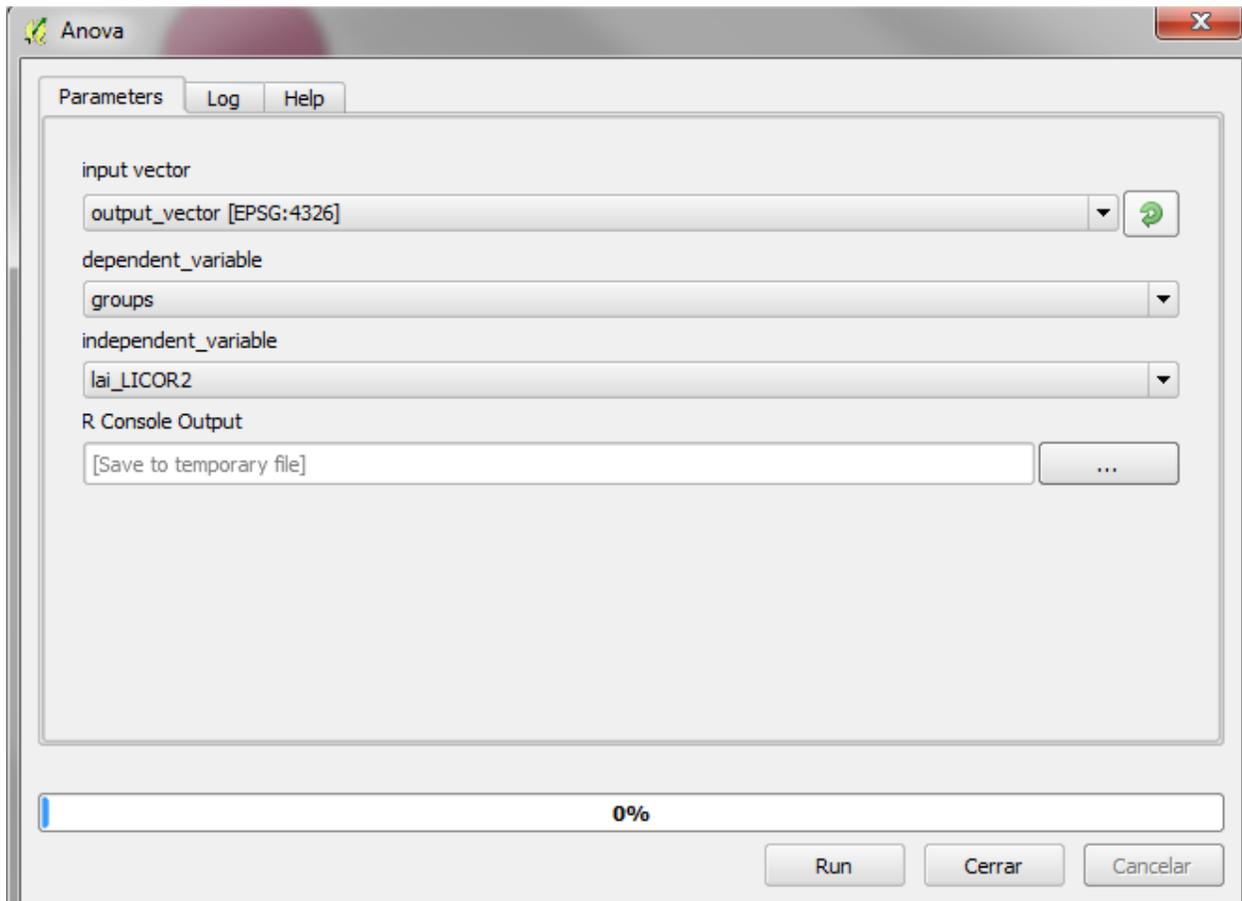
The user must enter the equation by which you want to group the stands, for example according to the equation of the dominant height. In our case, we use the equation obtained from the 95th percentile. In this example, the polygons will be classified into 3 different groups of homogeneity.







rN	groups
0000000...	1
00000000...	2
500000000...	2
2200000000...	2
32800000000...	2
.26200000000...	1
0.25700000000...	1
0.21600000000...	1
0.28400000000...	1
0.34700000000...	2
0.29400000000...	2
39600000000...	3
13000000000...	3
000000000...	3



R Output

```
Df Sum Sq Mean Sq F value Pr(>F)
points[[independent_variable]] 1 6.700 6.700 79.23 1.24e-06 ***
Residuals 12 1.015 0.085
---
Signif. codes: 0 '****' 0.001 '***' 0.01 '**' 0.05 '.' 0.1 ' ' 1
```

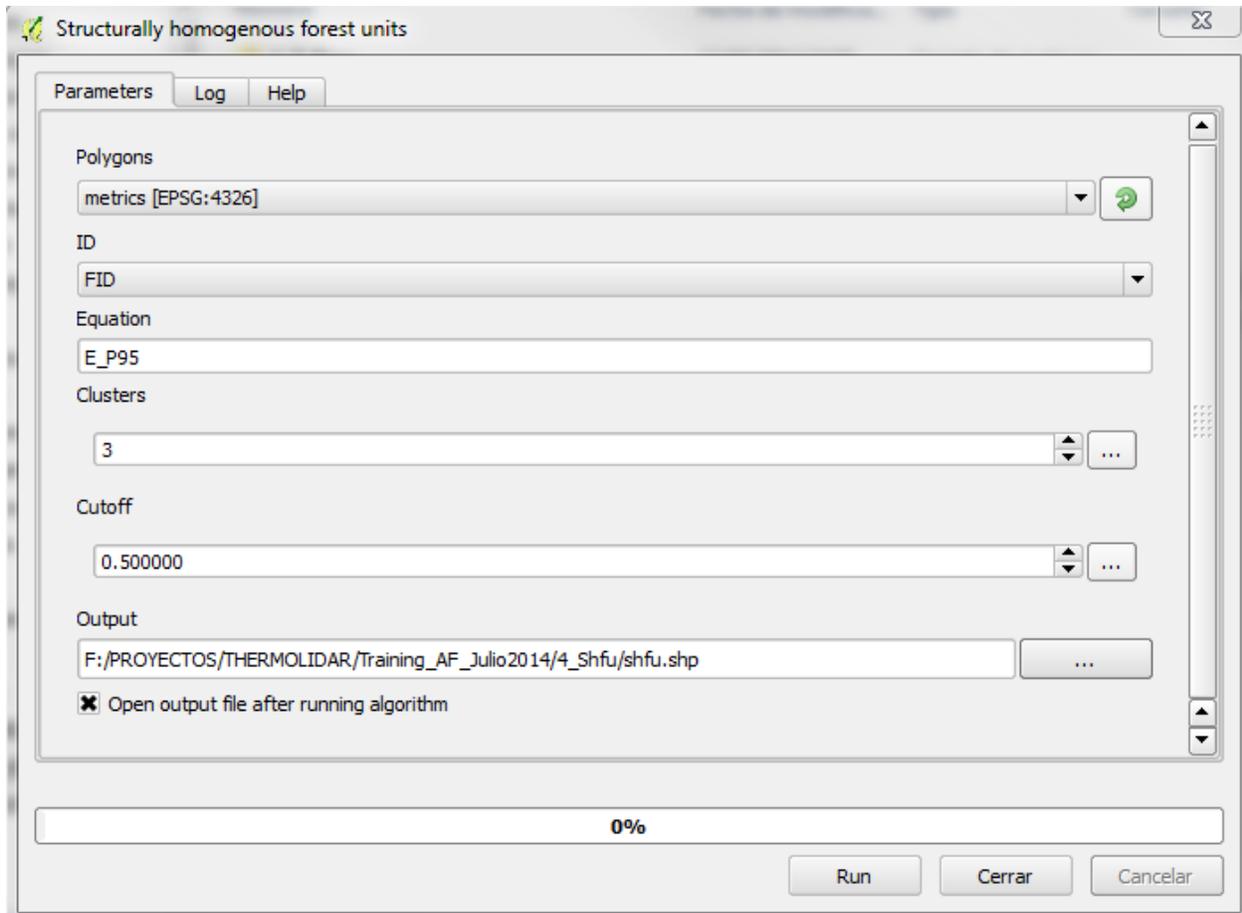
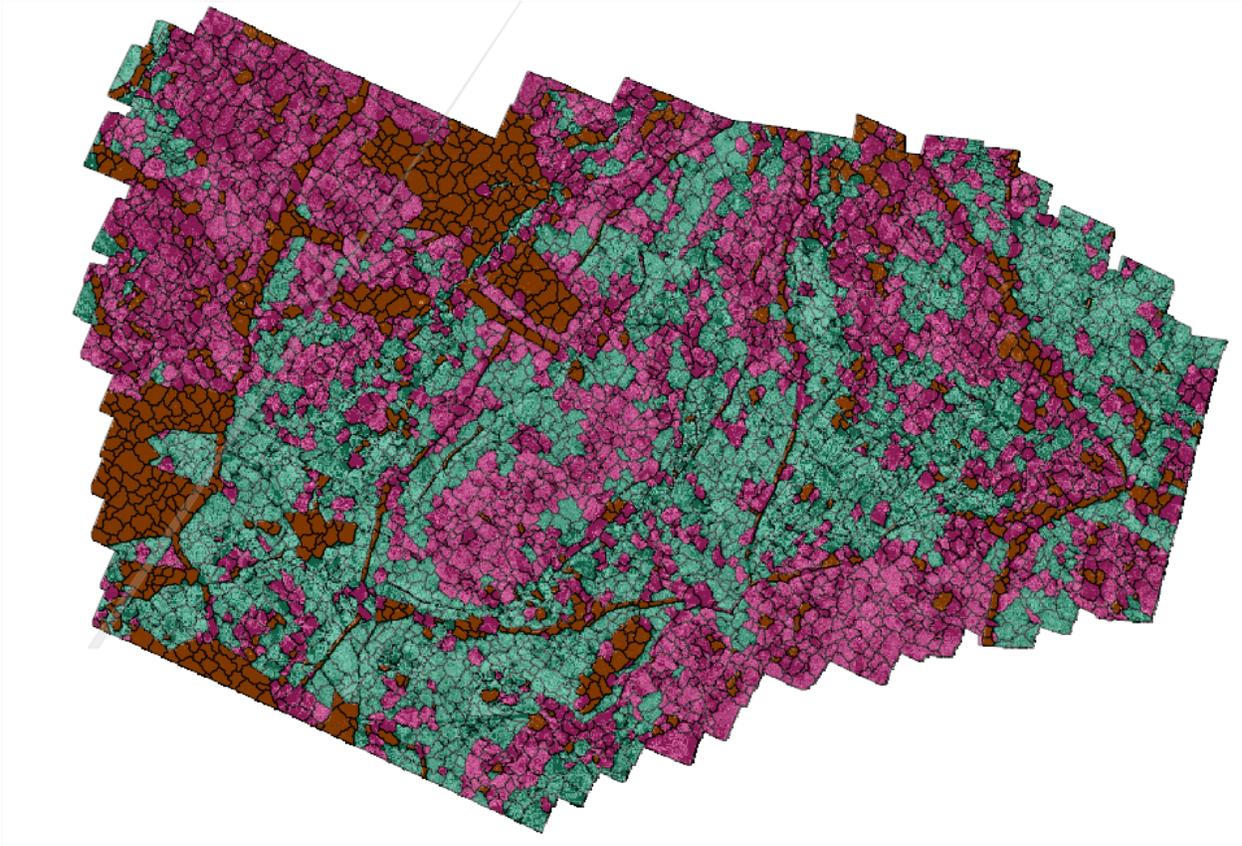


Figure 5.26: Interface of the “SHFU” module

We get as output:



Forest Health Classification

Unsupervised pixel-based classification

Through this tool raster classify the temperature as many classes as you specify. Temperature classification is performed based on homogeneous units, so many output raster have been defined as SHFU be obtained.

In our case, we obtain three raster classified. Each raster has associated many categories defined temperature.

We get as output:

Unsupervised object-based classification

In the same way that the pixel-oriented, this tool classification raster classify the temperature as many classes as you specify. Temperature classification is performed based on homogeneous units, so many output raster have been defined as SHFU be obtained.

In our case, we obtain three raster classified. Each raster has associated many categories defined temperature.

The difference from the previous tool, is that instead of being classified pixels are classified objects. The classification is made based on the mean value of each of the objects.

We get as output:

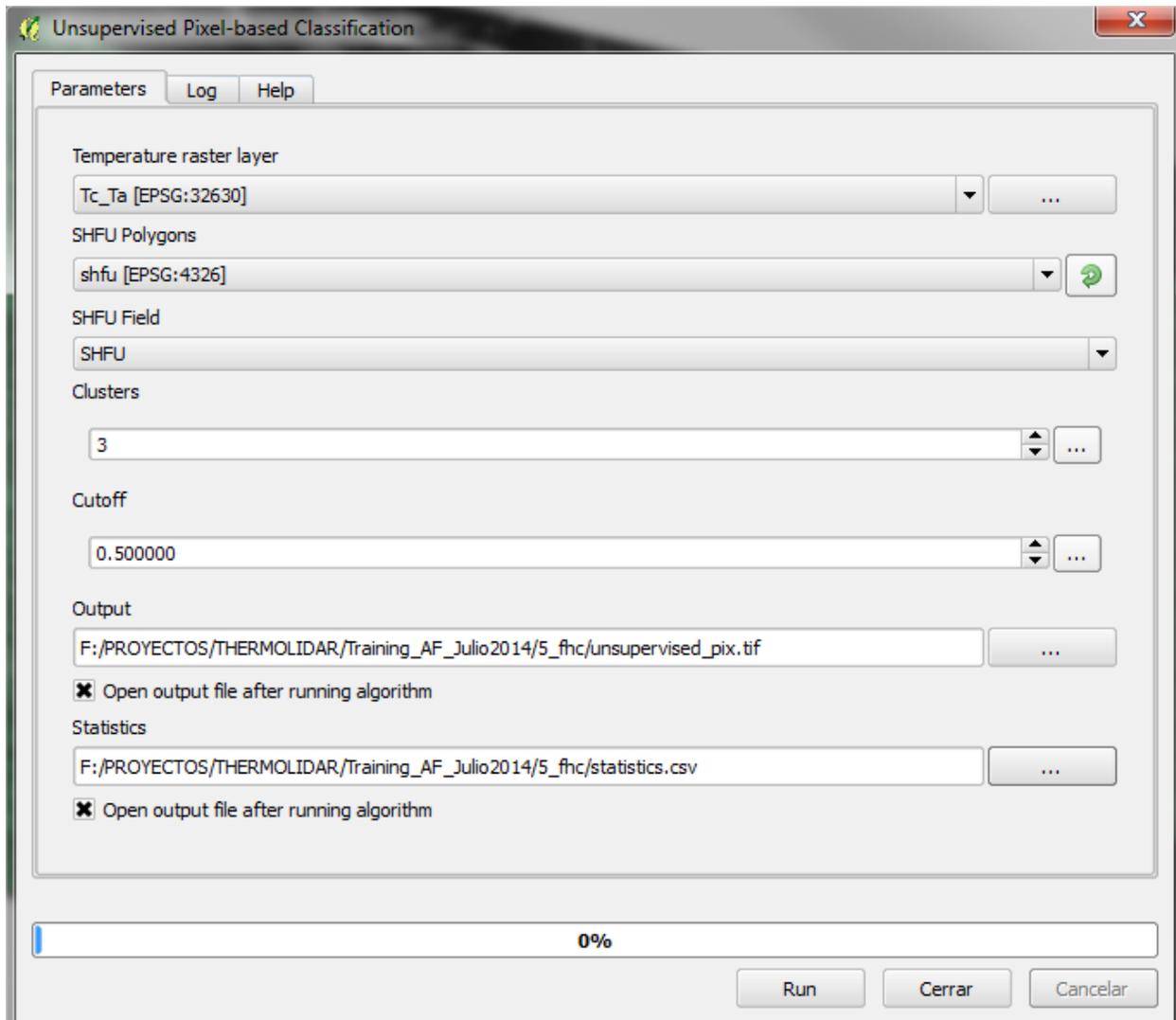
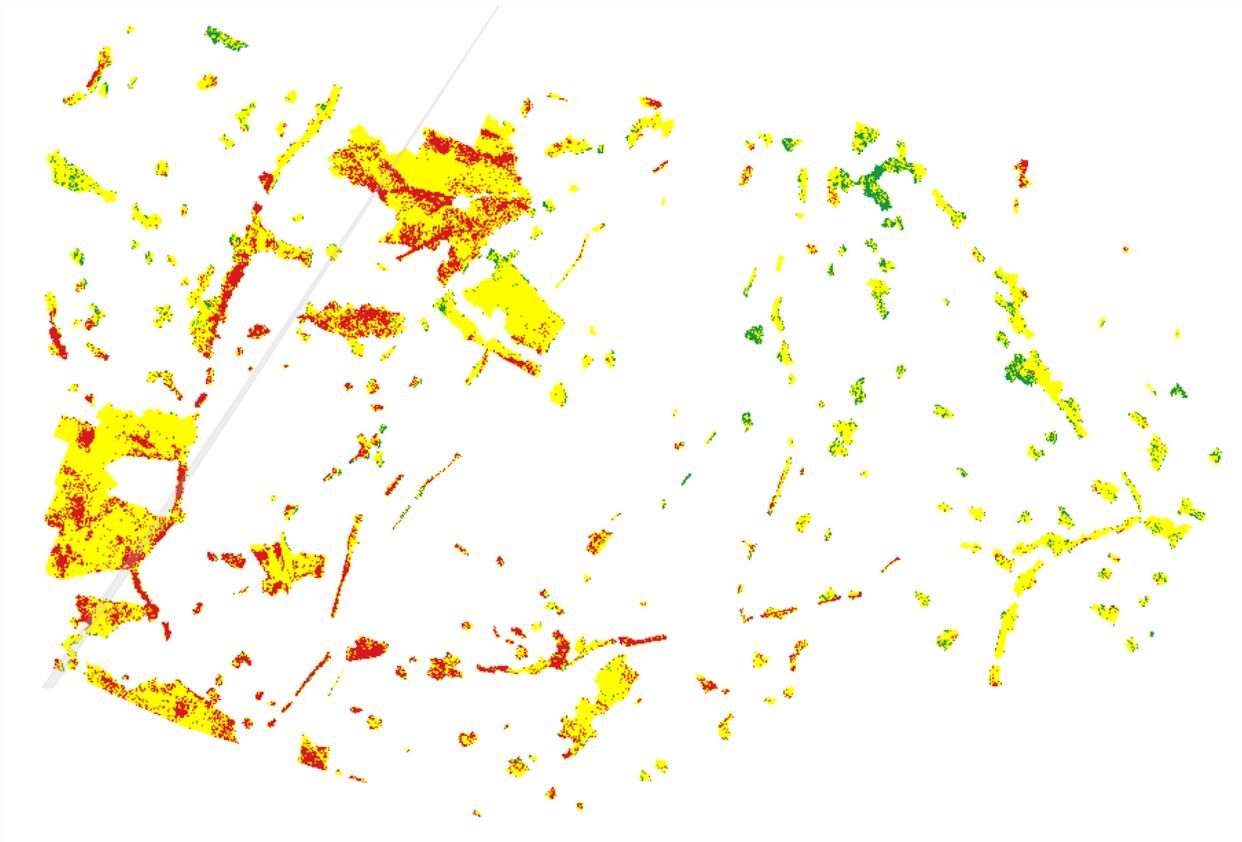
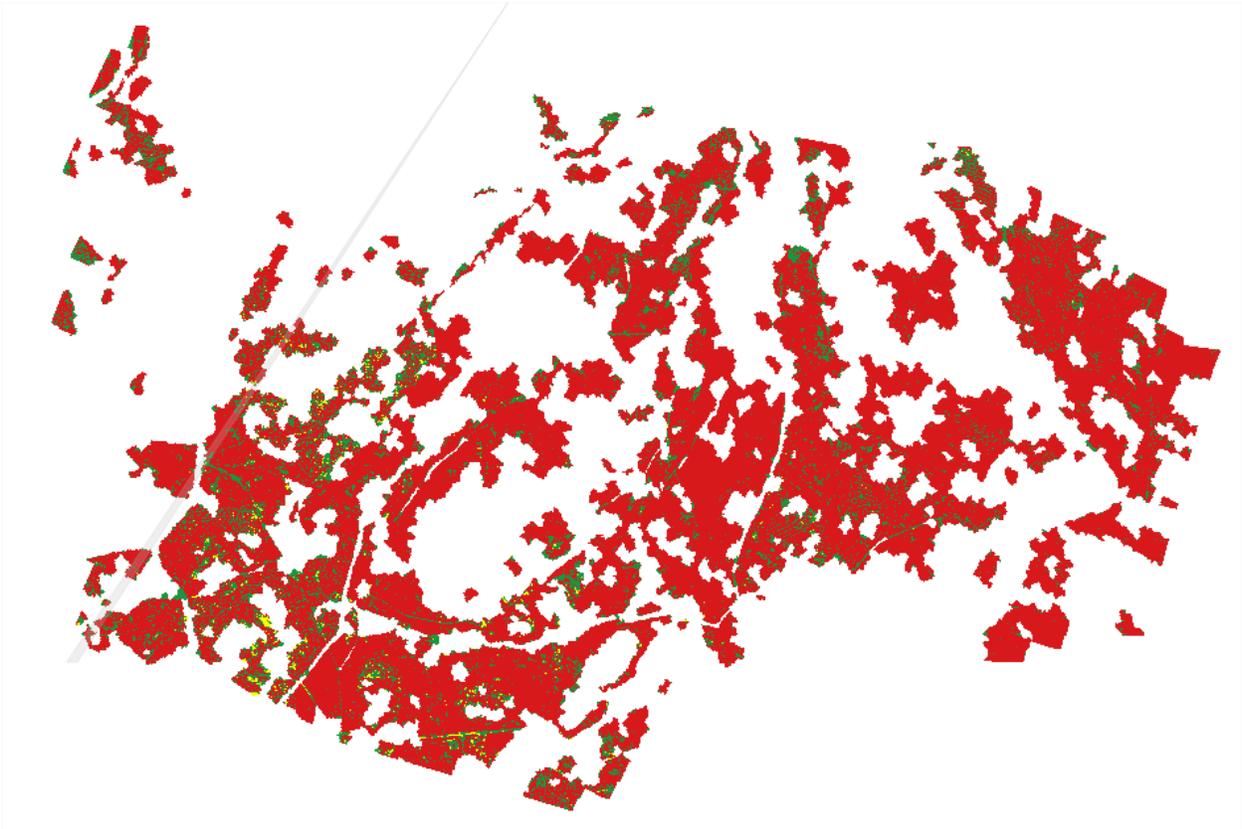


Figure 5.27: Interface of the “Unsupervised pixel-based classification” module





	SHFU /	ID_CLUSTER	Num	Min	Max	Mean	Median	Var	Std	CV
0	0	1	1237	-2.53174	8.33309	3.14388406679	3.14724111557	9.05556839621	3.00924714775	0.957174973318
1	0	2	912	17.4796	27.9976	21.3437612768	21.297662735	5.27431389535	2.29658744561	0.107599940602
2	0	3	1088	8.34847	17.4642	12.9062508415	12.9063205719	6.9535703344	2.63696233087	0.204316680596
3	1	1	1181	13.6037	24.5764	18.5671090991	18.5583438873	8.28136054245	2.87773531487	0.154991027387
4	1	2	1073	4.6036	13.596	9.10042710629	9.10363197327	6.76343576073	2.60066063929	0.28577347073
5	1	3	908	-18.0122	4.59588	-0.234968971739	0.785470604897	19.4255826338	4.40744627123	-18.7575671741
6	2	1	1209	-0.821112	9.54287	4.47570125341	4.47258901596	8.59250844904	2.9312980826	0.654936046137
7	2	2	911	18.4659	28.1287	22.3054373811	22.2878360748	5.03906646478	2.24478650762	0.100638533523
8	2	3	1062	9.55053	18.4582	14.0056184205	14.0082187653	6.625509423	2.57400649242	0.183783851247

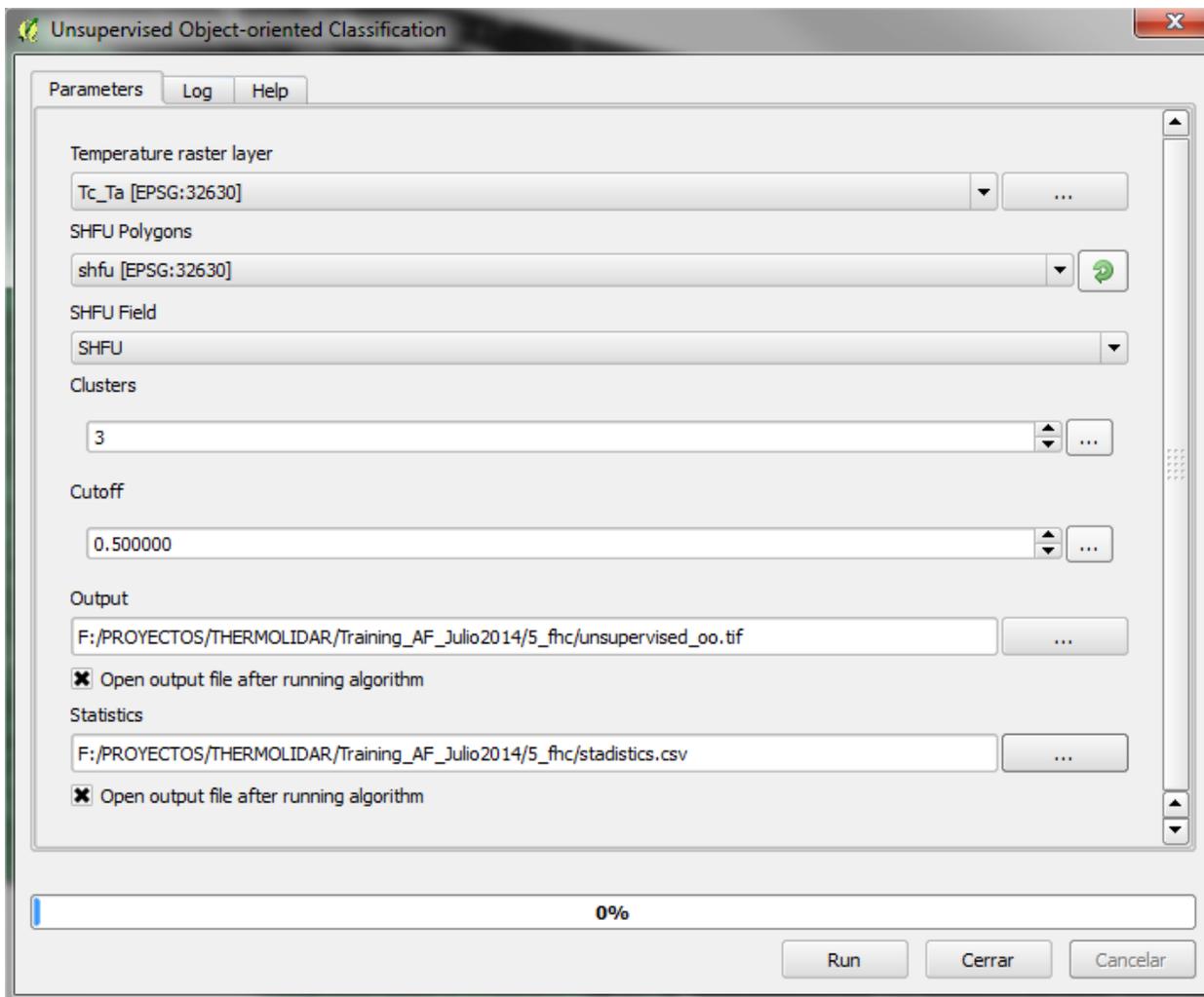
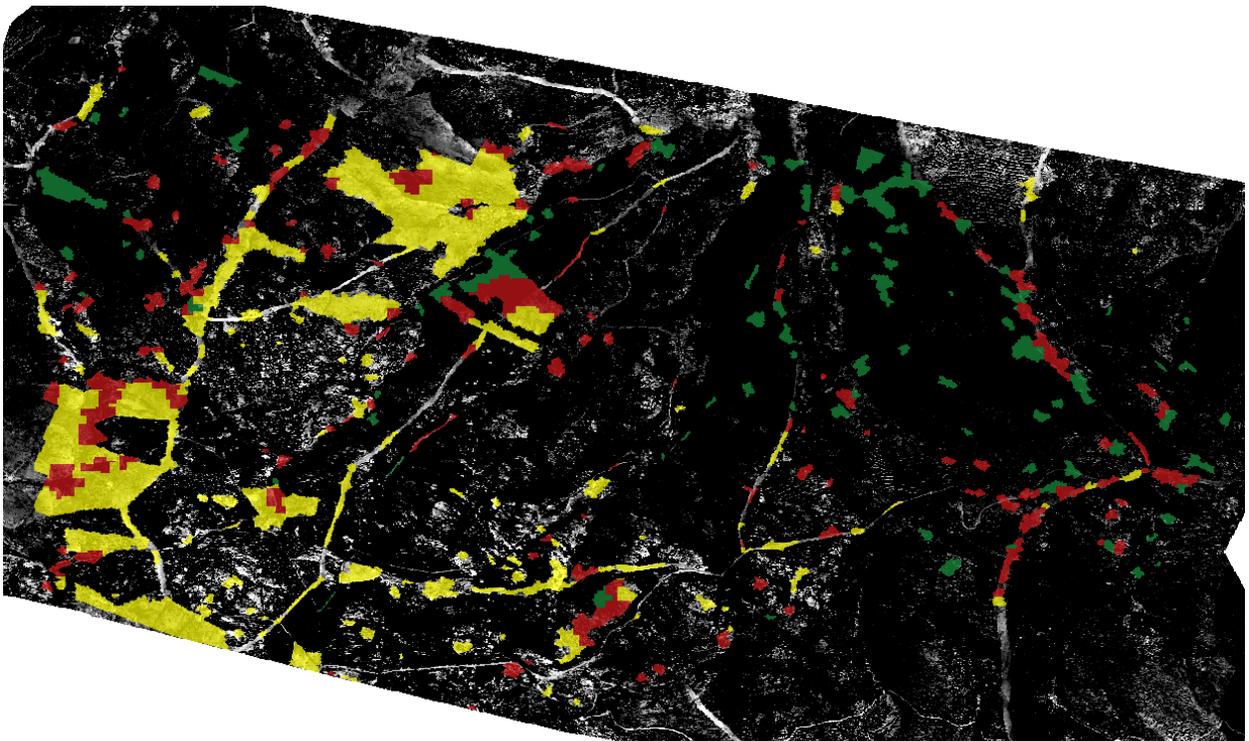
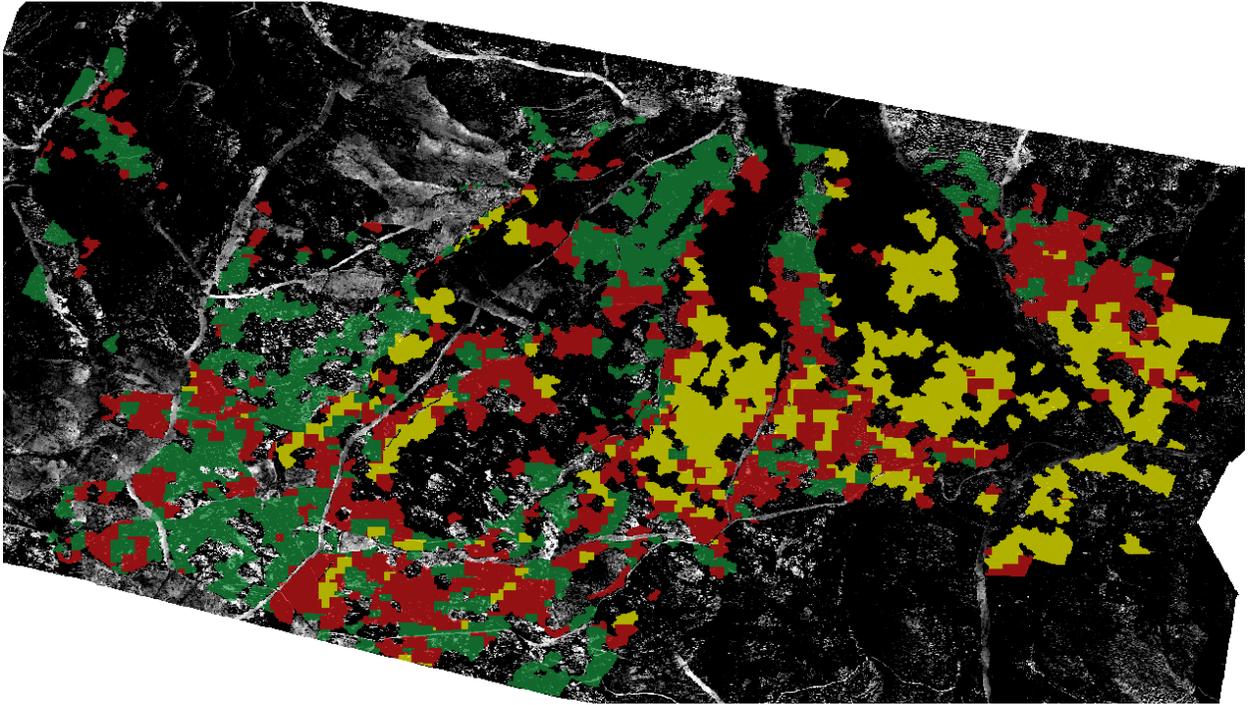
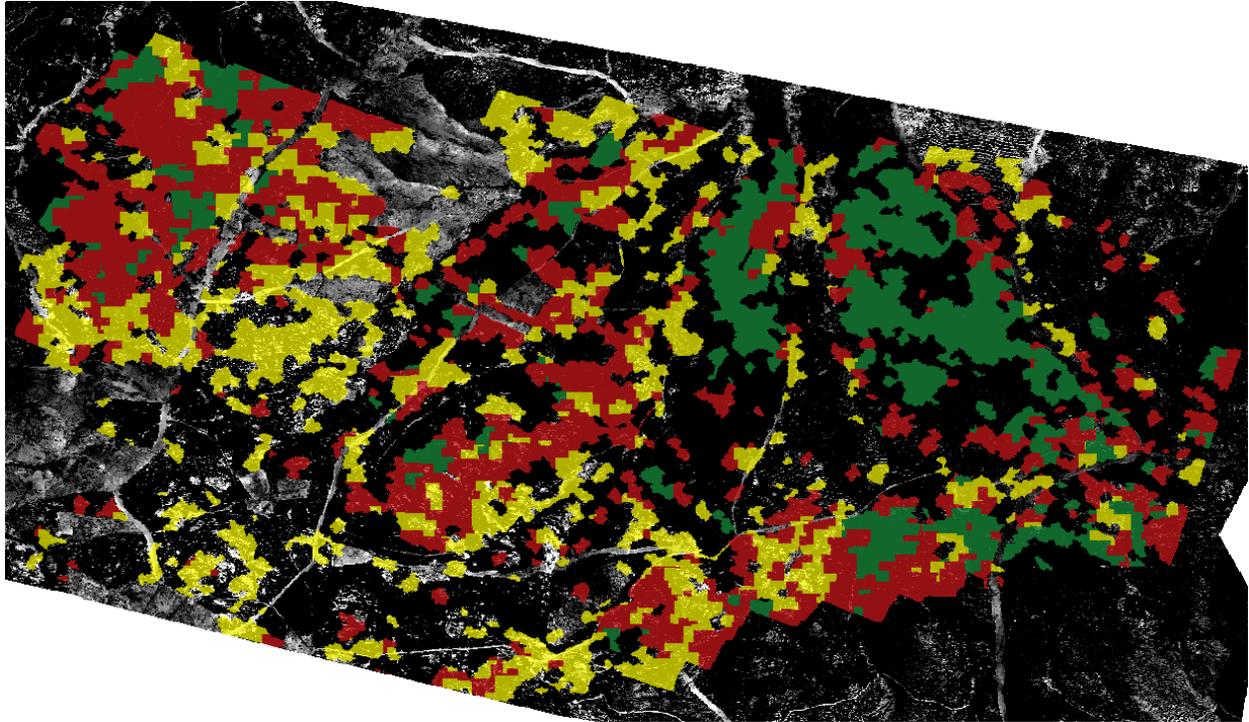


Figure 5.28: Interface of the “Unsupervised object-based classification” module





SHFU /	ID_CLUSTER	Num	Min	Max	Mean	Median	Var	Std	CV	
0	0	1	611	4.54645983191	6.0606050475	5.27386462475	5.26277883063	0.179042064385	0.423133624739	0.0802321740974
1	0	2	392	1.94225548488	4.53926634052	3.84627099352	3.944589015	0.291118702199	0.539554169847	0.140279811473
2	0	3	568	6.07050297596	13.3579859526	7.44076065583	7.1628590438	1.44392615223	1.20163478321	0.161493540619
3	1	1	209	8.16275034053	11.2300518456	9.9265033978	10.0288406819	0.777195234851	0.881586771028	0.0888114107957
4	1	2	377	11.2326354789	16.4421430672	12.8900781456	12.6693421276	1.29526258777	1.13809603627	0.0882924078046
5	1	3	135	3.11066703739	8.15440212906	6.36824409517	6.61714630026	1.798452415	1.3410639116	0.210586135135
6	2	1	793	4.92381839182	7.36172191965	6.09662741977	6.0719456266	0.480082905778	0.692880152536	0.113649745151
7	2	2	700	7.36347859398	15.2879598799	9.37704934715	8.84403647928	2.83809641714	1.68466507566	0.179658335292
8	2	3	403	2.10779622188	4.91848166033	4.10746052592	4.27234326208	0.39519403862	0.628644604383	0.153049457302

Indices and tables

- *genindex*
- *modindex*
- *search*

- [Bunting2013b] Bunting, P., Armston, J., Clewley, D., Lucas, R. M., 2013. Sorted pulse data (SPD) library. Part II: A processing framework for LiDAR data from pulsed laser systems in terrestrial environments. *Computers and Geosciences* 56, 207 – 215.
- [BaterCoops2009] Bater, C. W., Coops, N. C., 2009. Evaluating error associated with lidar-derived DEM interpolation. *Computers and Geosciences* 35 (2), pp. 289–300.
- [EvansHudak2007] Evans, J. S., Hudak, A. T., 2007. A multiscale curvature algorithm for classifying discrete return lidar in forested environments. *IEEE Transactions on Geoscience and Remote Sensing* 45 (4), pp. 1029 – 1038.
- [Zhang2003] Zhang, K., Chen, S., Whitman, D., Shyu, M., Yan, J., Zhang, C., 2003. A progressive morphological filter for removing nonground measurements from airborne LIDAR data. *IEEE Transactions on Geoscience and Remote Sensing* 41 (4), pp. 872 – 882.
- [Bunting2013] Bunting, P., Armston, J., Clewley, D., Lucas, R. M., 2013. Sorted pulse data (SPD) library. Part II: A processing framework for LiDAR data from pulsed laser systems in terrestrial environments. *Computers and Geosciences* 56, 207 – 215.
- [BaterCoops2009] Bater, C. W., Coops, N. C., 2009. Evaluating error associated with lidar-derived DEM interpolation. *Computers and Geosciences* 35 (2), pp. 289–300.
- [EvansHudak2007] Evans, J. S., Hudak, A. T., 2007. A multiscale curvature algorithm for classifying discrete return lidar in forested environments. *IEEE Transactions on Geoscience and Remote Sensing* 45 (4), pp. 1029 – 1038.
- [Naesset1997a] Naesset, E., 1997. Determination of mean tree height of forest stands using airborne laser scanner data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 52, pp. 49 – 56.
- [Naesset1997b] Naesset, E., 1997. Estimating timber volume of forest stands using airborne laser scanner data. *Remote Sensing of Environment*, 61, pp. 246 – 253.
- [Naesset2002] Naesset, E., 2002. Predicting forest stands characteristics with airborne scanning laser using a practical two-stage procedure and field data. *Remote Sensing of Environment*, 80, pp. 88 – 99.
- [Zhang2003] Zhang, K., Chen, S., Whitman, D., Shyu, M., Yan, J., Zhang, C., 2003. A progressive morphological filter for removing nonground measurements from airborne LIDAR data. *IEEE Transactions on Geoscience and Remote Sensing* 41 (4), pp. 872 – 882.