CLUZ for QGIS guide v2020.03.18

R.J. Smith, Durrell Institute of Conservation and Ecology, University of Kent

CLUZ (Conservation Land-Use Zoning software) is a QGIS 3 plugin that allows users to design protected area networks and other conservation landscapes and seascapes (Smith, 2019). It can be used for on-screen planning and also acts as a link for the Marxan conservation planning software. It was developed by Bob Smith, from the Durrell Institute of Conservation and Ecology (DICE), and funded by the UK Government's Darwin Initiative.

Most of the CLUZ functions work on any computer that can run QGIS 3. One known issue is running Marxan through CLUZ on Mac OS X computers or computers with complicated security settings. In such cases users should set up the Marxan files in CLUZ, run Marxan independently and then import the results into CLUZ.

CLUZ functions

CLUZ is a QGIS plugin that consists of the CLUZ menu, containing 17 functions, and 7 buttons.

CLUZ menu items

View and edit CLUZ setup file

The CLUZ setup file contains information on the location of the Marxan program, the folders that contain the Marxan data and where the Marxan output files will be stored. It also lists the location of the planning unit layer and the target table. All of the CLUZ functions use information from the CLUZ setup file, so the user has to load a suitable file or create a new one before carrying out any analysis. This function lets the user create or load an existing file and to edit its details.

Create initial CLUZ files

This module produces the three core files that CLUZ needs.

- It converts an existing polygon layer into a correctly formatted planning unit layer by adding fields called **Unit_ID**, **Area**, **Cost** and **Status**. Each polygon is given a unique identifier and its status is set as *Available*. The **Area** value of each polygon is calculated and there is also an option to use this number as the **Cost** value as well, if no other cost data exists.

- It produces a new CLUZ target table in the correct format with the required fields.

- It produces a Marxan abundance file (named puvspr2.dat) with the correct field headings and stores it in the specified input folder.

Convert polyline or polygon themes to Marxan abundance data

This function lets the user select polygon or line conservation feature distribution layers, calculates the amount of each feature in each planning unit and adds the data to the Marxan abundance file (the puvspr2.dat file) and CLUZ target file. The distribution layers must contain an ID field that gives a unique numeric identifier code for each conservation feature.

This function also lets the user divide each abundance value by a specified number to convert from one measurement system to another. For example, if a vegetation map uses a UTM

reference system that is measured in metres, the user can convert area values from m^2 to hectares by specifying that each abundance value should be divided by 10000.

Convert raster layer to Marxan abundance data

This function lets the user import raster data showing the non-overlapping distribution of a set of conservation features, e.g. a landcover or vegetation type layer. It does this by using the Zonal histogram function in QGIS, which calculates the amount of each feature in each planning unit, and then adding the resultant data to the Marxan abundance file (the puvspr2.dat file) and CLUZ target file. The raster file must be a single band integer layer, where the value of each pixel is the unique numeric identifier code for each conservation feature. CLUZ will not import data for pixels with a value of 0.

This function also lets the user divide each abundance value by a specified number to convert from one measurement system to another. For example, if a landcover map uses a UTM reference system that is measured in metres, the user can convert area values from m^2 to hectares by specifying that each abundance value should be divided by 10000.

Import fields from table into Marxan abundance file

This function lets the user add data from an existing comma delimited (csv) text file table to the Marxan abundance file (puvspr2.dat file) and CLUZ target file. The csv table must include a field that contains the planning unit ID values that match with those in the planning unit layer. All of the other fields must describe the abundance data, with each field listing the amount of a conservation feature found in each planning unit. The field names of these fields containing abundance data must include the unique numeric identifier of the conservation feature, e.g.

Unit_ID, veg1, veg2, spp4
1, 0.0, 0.4, 2.0
2, 0.3, 0.6, 1.0
3, 0.1, 1.5, 0.0

This function also allows you to divide each value by a specified number to convert abundance values from one measurement system to another (e.g. from m^2 to hectares).

Remove features from CLUZ target table and Marxan abundance file

This function lets the user delete records on unwanted conservation features from the CLUZ system by removing the relevant data from the target table and the Marxan abundance file.

Recalculate target table data

This function calculates the total amount of each conservation feature in the planning region and the amount that is conserved (ie in *Earmarked* or *Conserved* units) based on the Marxan abundance file (puvspr2.dat file) and planning unit layer attribute table. It should be used whenever changes have been made to the conservation portfolio without using CLUZ.

Troubleshoot all CLUZ files

This function inspects all of the CLUZ files to check they are in the correct format.

Display distributions of conservation features

This lets the user show the distribution of any of the conservation features listed in the target table based on the data stored in the Marxan abundance file (puvspr2.dat file).

Identify features in selected units

This provides information on the conservation features found within the planning units that have been pre-selected by the user (using QGIS's Select Features functions). CLUZ lists the amount of each conservation feature in the Available, Earmarked, Conserved and Excluded planning units, as well as listing the target for each feature and target shortfall.

Calculate richness scores

This calculates two types of richness metrics:

- Feature count – the number of conservation features in each planning unit.

- Restricted Range Score – the sum of the restricted range value for each conservation feature found in the planning unit. The restricted range value = area of feature in planning unit / total area of feature in planning region.

If the target table contains a field named Type then this module can be used to calculate values for a subset of conservation features, where each subset is defined by the number code in the Type field. For example, if the Type field codes vegetation types as 1 and species as 2, then this function can be used to count only the number of species found in each planning unit.

Calculate portfolio characteristics

This displays a number of characteristics about the portfolio of planning units (i.e. the Earmarked and Conserved planning units), as well as details on all the planning units in the planning region. There are four sub-functions and these are:

- Planning unit details. This gives the total area, total planning unit cost and total number of planning units for the Available, Earmarked, Conserved and Excluded planning units, as well as details for the Portfolio (the Earmarked and Conserved planning units) and the Region (all the planning units).

- Spatial details. This gives the number of patches, area of smallest patch, median area of patches, area of largest patch and the portfolio boundary length (as calculated from the bound.dat file).

- Selection frequency details. This shows the number of planning units falling within different categories of selection frequency values, as produced in Marxan. The user must specify the field in the planning unit layer that contains the selection frequency score and input the total number of runs used in the Marxan analysis that produced those scores. CLUZ then shows the number of planning units with different selection frequency values and ranges of values.

- Number of patches containing features. This shows the number of portfolio patches that each conservation feature is found within, where each patch is a cluster of Earmarked and/or Conserved planning units.

Create Marxan input files

This produces the additional files needed to run Marxan from the planning unit layer and target table. The spec.dat file gives details on each conservation feature, the pu.dat file gives details on each planning unit and the bound.dat file lists the length of edge shared by each planning unit.

The user can specify whether the bound.dat file should include data on the planning region boundary, i.e. edges that are not shared between planning units. Including these boundaries makes it less likely that Marxan will select planning units at the edge of the planning region.

Marxan uses these files every time it runs, so this function should be used whenever the data has been updated in CLUZ if Marxan is to reflect these changes.

Launch Marxan

This function opens the Launch Marxan dialogue window, which lets the user specify different parameters before running the program. Parameters are the number of iterations per run, the number of runs, the output file name, whether boundary cost values should be included and whether Marxan should save extra outputs. The Advanced options lets the user specify what proportion of planning units should be selected at the beginning of each run and the definition of whether a species target has been met. It also lets the user launch several parallel Marxan analyses to speed up how long it takes to complete the specified number of runs.

Load previous Marxan results

This lets the user add results from previous Marxan analyses to new fields in the planning unit layer table and display them in the View.

Calibrate Marxan parameters

This lets users undertake sensitivity analyses to measure how changing either the BLM (Boundary Length Modifier), Number of iterations, Number of runs or SPF (Species Penalty Factor) influences the Marxan results. The user has to specify the number of analyses, the minimum value of the parameter to be tested, the maximum value of the parameter to be tested and whether there should be exponential steps between the minimum and maximum steps. Selecting exponential steps means the calibration process tests a wide variety of values, e.g.

Parameter to Calibrate = Number of iterations, Number of analyses = 4; Min = 100000; Max = 100000000

☐ Use exponential steps	Values =	100000,	33400000,	66700000,	100000000
☒ Use exponential steps	Values =	100000,	1000000,	10000000,	100000000

The user also has to specify the Marxan output files base name, i.e. if the number of analyses is 4 and the base name is “blm_test” then the names of the Marxan output files will be blm_test1, blm_test2, blm_test3 and blm_test4.

Finally, the user has to specify the name of the output file, where MinPatch will save the following information from each of the Marxan analyses: Analysis (identifier number), Name (Marxan output files name), Iterations (number of iterations), Runs (number of runs), BLM (Boundary Length Modifier value used), Med Portfolio Cost (median cost of all the portfolios produced in the analysis = combined planning unit cost + boundary cost + penalty factor cost), Med Planning Unit cost (median portfolio planning unit cost), Med Boundary length (median portfolio boundary length), Med Feature Penalty cost (median portfolio feature penalty cost), Med MPM (median portfolio minimum proportion met), Med PU Count (median number of planning units in each portfolio).

Launch MinPatch

MinPatch is designed to update Marxan outputs to produce results where every patch of planning units is larger than a minimum size specified by the user. For more details see Appendix 1.

CLUZ buttons

Open Target Table

Clicking on this button will open the target table, allowing the user to view details of each conservation feature, their target and how much is currently conserved in the planning region. The values in the **PC_target** field are automatically updated each time the table is opened and are colour coded depending on whether the target has been met.

Display Marxan Abundance File Data

Clicking on this button lets the user select and display the amount of each of the specified conservation features in the Marxan abundance file (puvspr2.dat file).

Change Status of Earmarked Units to Available

Clicking on this button will change the status of all Earmarked planning units to Available and will update the target table to reflect these changes. This is useful when undertaking onscreen conservation planning, as it lets the user remove any changes and start afresh.

Open Marxan Target Results Table

Running Marxan through CLUZ produces the best layer, which shows the most efficient of the conservation portfolios identified by Marxan. Clicking on this button open a table that lists how many of the targets were met by this best portfolio. CLUZ will only display a target results table if Marxan has been run since opening QGIS and using CLUZ. NB Marxan sometimes fails to include the conservation feature names in its output files, and in this case it will be added by CLUZ.

Change Status of Best Units to Earmarked

Clicking on this button will give Earmarked status to any planning unit that was identified by Marxan as being part of the best portfolio. To do this, CLUZ uses information from the Best field, which is added to the planning unit layer table whenever Marxan is run.

Change Planning Unit Status

Clicking on this will open the **Change Status panel**, which lets the user change the status of selected planning unit to *Conserved*, *Earmarked*, *Available* or *Excluded*. *Conserved* units are those that are part of current or future protected areas. *Earmarked* units are those that have been selected by the user as belonging to a proposed conservation portfolio and *Available* units are those that have not been selected by the user to belong to the portfolio. *Excluded* units are those that will definitely not belong to any portfolio for ecological, economic or political reasons.

At its simplest level, conservation planning in CLUZ involves identifying and setting the *Conserved* and *Excluded* units at the beginning of the process and then deciding whether the remaining units should have *Earmarked* or *Available* status. The Change Status panel lets the user change the status value of selected planning units by choosing the relevant button . Turning off the **Allow changes in conserved and excluded status** check box means that only the status of *Earmarked* and *Available* units can be changed.

Turning on the check box lets the user change units to *Conserved* or *Excluded* status, and lets the status of *Conserved* or *Excluded* units be modified. This is generally only done at the beginning of the planning process.

Identify Features in Planning Unit

Choosing this tool and clicking on a planning unit will produce a window showing the conservation features that are found in the unit, together with the amount of each feature, its target and the percentage of that target that has been met.

CLUZ file format description and checklist

CLUZ uses three different files as the source of all the information it needs to display and analyse the conservation planning data. This section provides a checklist of the correct format for these files, which can be used when creating new CLUZ files or to identify possible problems with existing files.

The planning unit layer attribute table

The planning unit layer is the shapefile that shows the boundary of each of the planning units in the study region. The planning unit layer attribute table is the associated database that describes the features of each of these units. This table must include the following fields with the following format:

Field Name	Field Type	Field Width	Decimal places	Notes
Unit_ID	Integer	-	0	The id code of each planning unit must be a unique integer.
Area	Number	-	-	This field should contain data on the area of each planning unit.
Cost	Number	-	-	This field should contain data on the cost of including each planning unit in the conservation portfolio. You should assume that the cost value is the same as the area value if no other data are available.
Status	String	10	-	This field contains data on the conservation status of each planning unit, which must be listed as, ***Conserved***, ***Earmarked***, ***Available*** or ***Excluded***

Other things to note are:

1. Each row in the unit layer table must have a unique **Unit_ID** value. If you have a planning unit that consists of several polygons then use the **Dissolve** option in **GeoProcessing Tools** in QGIS to ensure that each planning unit is represented by only one row in the unit layer table.

2. The planning unit layer table can contain any other fields that you require.

The Marxan abundance file (puvspr2.dat file)

This is a comma delimited text file that is stored in the specified input folder. The first line of the file consists of the headings, which are **species**, **pu** and **amount**. The subsequent lines list the conservation feature's unique ID value, the planning unit unique ID value and the feature amount.

This text file must be ordered by the planning unit id, it cannot contain any other information and each value must be separated with a comma. The best way to avoid formatting errors is to import all of the data into the puvspr2.dat using CLUZ.

Other things to note are:

1. CLUZ also creates a file named sporder.dat, which is identical to the puvspr2.dat file but is ordered by the species id value, instead of the planning unit id value. This file is then used by Marxan to speed up the time taken to import the abundance data.

The CLUZ target table

The target table is a comma delimited text file (csv) that describes the information about each conservation feature. It lists the unique ID, name and target for each conservation feature and summarises the amount conserved and the total amount in the planning region. This table must include the following fields with the following format:

Field Name	Field Type	Notes
Id	Integer	The id code of each conservation feature must be a unique integer.
Name	String	This field should contain the name of the conservation feature. These names can be in two formats: (a) with spaces but no numbers or (b) with numbers but no spaces, ie “cloud forest” or “forest_type_2” is OK, “forest type 2” is not.
Target	Number	This field should contain data on the target for each conservation feature, which is the amount of each feature that should appear in the final conservation portfolio.
Ear+Cons	Number	This field lists the amount of each conservation feature that is found in ***Earmarked*** and ***Conserved*** units, based on the data in the unit layer and abundance tables. Update these conserved values using the **Recalculate target table data** module if the original data have been edited without using CLUZ.
Total	Number	This field lists the total amount of each conservation feature recorded in the abundance table. Update these total values using the **Recalculate target table data** module if the abundance data have been edited without using CLUZ.
PC_Target	Number	This shows the percentage of the target that has been met, based on the values in the **Target** and **Ear+Cons** fields. The values are recalculated each time the user opens the target table or uses the **Recalculate target table data** module. Values that are less than the target are shown in **red**, values that equal or are above the target are shown in **green**. If no target is set then the value is shown as **-1 in grey**.

Other things to note are:

1. You will need to edit the values in the CLUZ target table using Excel or a similar software package that can manipulate and save csv files.

2. The target table can contain any other fields that you require.

3. Name formatting is a common source of errors in Marxan, so CLUZ will edit names containing numbers whenever it produces the spec.dat file for Marxan. This has no impact when displaying the results in CLUZ.

References

Smith, RJ (2019). The CLUZ plugin for QGIS: designing conservation area systems and other ecological networks. Research Ideas and Outcomes 5, e33510. Download here.

Appendix 1: Running MinPatch through CLUZ

Introduction

One of the most efficient approaches for designing protected area (PA) networks is to use systematic conservation planning software. A number of software packages are available and they all include a spatial cost or constraint component in their algorithms, which lets the user influence the size of each PA identified. Many conservation planners want to set minimum PA size thresholds, as small PAs are less viable and more expensive to manage, but this can only be achieved with existing software packages by repeatedly reducing the fragmentation levels of the PA system until every PA meets the required threshold. Such an approach is inefficient because it increases the size of every PA, not just the smaller ones. This is why we developed MinPatch, which manipulates outputs from the Marxan conservation planning software so every PA meets the user-defined minimum size threshold.

Quick start guide

Follow this if you want to quickly see what MinPatch does. You can then read the rest of the guide (and ideally the references given at the end) to understand what MinPatch is actually doing.

Running MinPatch

1) Setup CLUZ and  Launch Marxan, remembering to turn on the **Produce extra Marxan outputs** option so it produces a text file describing the portfolio produced by each Marxan run.

2) Create the MinPatch details file following the instructions given in the sections below on **Specifying the main parameters** and **Creating the MinPatch details file**.

3)  Run MinPatch using the outputs you created in stage 1, the BLM value you set in stage 1 and the MinPatch details file you created in stage 2.

MinPatch outputs

MinPatch produces a number of output files, which are listed below. All of the names have the suffix "mp_" followed by the name of the Marxan run that produced the original files, i.e. files form the Marxan run "output1" have the prefix "mp_output1_".

a) *MinPatch portfolios*. MinPatch processes each Marxan file and produces a text file listing which planning units appear in the updated portfolios. The file lists each planning unit unique identifier and in the "solution" field gives a value of 1 if it is part of the portfolio and 0 if it is not.

b) *Patchstats file*. This csv file details for each portfolio the number of patches, portfolio area and median patch size before and after being updated by MinPatch (see below for more details).

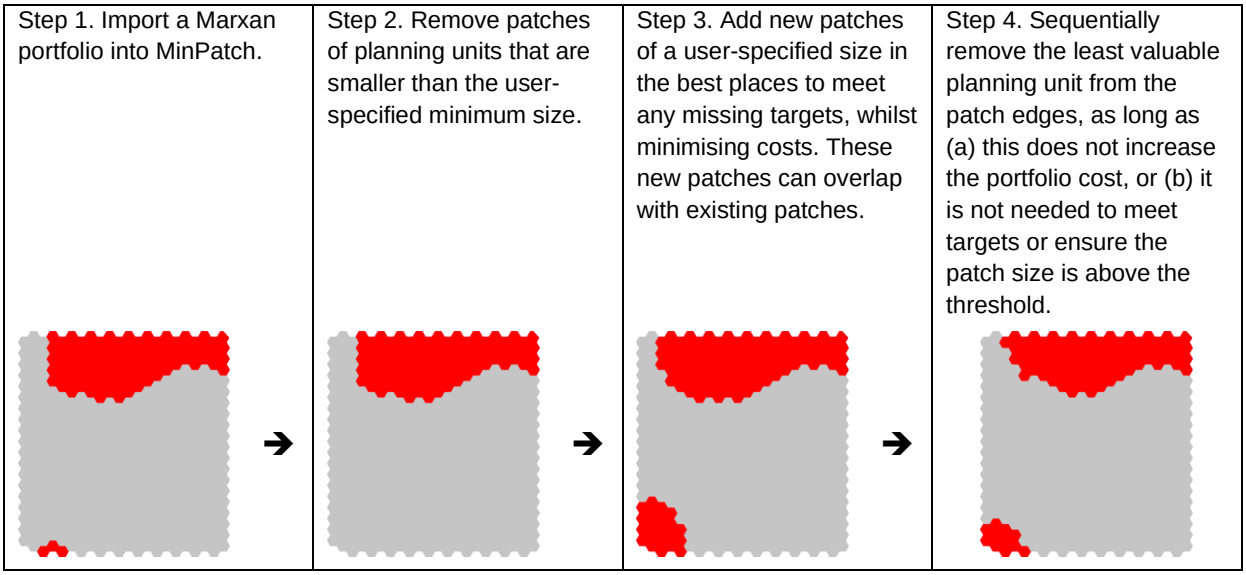
c) *Zone files*. If the user specifies that different minimum patch size parameters should be applied in different zones of the planning region then MinPatch produces additional files reporting on the portfolios' characteristics. The zonestats file reports the portfolio area and combined planning unit cost in each zone. The zonefeaturestats files reports the proportion of the target met by the planning units selected in each zone for each conservation feature.

d) *Best portfolio*. MinPatch identifies the best portfolio (i.e. the one with the lowest portfolio cost) and saves the text file with the suffix "_best" in the same format as the MinPatch portfolio files. CLUZ then imports and displays this best portfolio.

e) *Selection frequency output*. MinPatch produces an updated version of the Marxan selection frequency output by counting the number of times each planning unit appears in each MinPatch portfolio. It saves the results in a csv text file with the suffix "_summed", listing each planning unit and giving the selection frequency value in the "solution" field. CLUZ then imports and displays this selection frequency output.

How MinPatch works

MinPatch is a set of code written in the Python language and incorporated into the CLUZ plugin for QGIS. It manipulates the outputs from the Marxan conservation planning software. Marxan identifies near-optimal portfolios of planning units that meet conservation targets whilst minimising costs. Full details on the algorithms used in MinPatch are given in Smith et al (2010) but a simplified explanation of how MinPatch works is given below:



- The user can specify that different minimum patch size thresholds are applied in different zones. For example, marine conservation planners might want to specify larger PAs in offshore areas and terrestrial conservation planners might want to specify smaller PAs within 50 km of cities.

- MinPatch does not remove existing PAs from the portfolio, even if they do not meet the minimum size threshold.

- MinPatch manipulates every portfolio produced by a particular Marxan run and identifies the "best" portfolio. This best portfolio is the one with the lowest cost, measured using the same system that underpins Marxan (based on meeting targets, minimising the combined cost of the planning units and maintaining connectivity). The best MinPatch portfolio is not necessarily produced from the best Marxan portfolio.

- MinPatch also produces a selection frequency output, which counts the number of times each planning unit appears in the different MinPatch portfolios.

Specifying the main parameters

Minimum patch size

This is the key parameter used by MinPatch to ensure each patch of planning units meets the patch size threshold. This value is used in step 2 of MinPatch to identify and remove any patches of planning units that are too small (not including existing PAs). It is also used in step 4 as part of the simulated whittling process, so that MinPatch does not remove planning units that would make a patch smaller than the specified threshold.

New patch radius

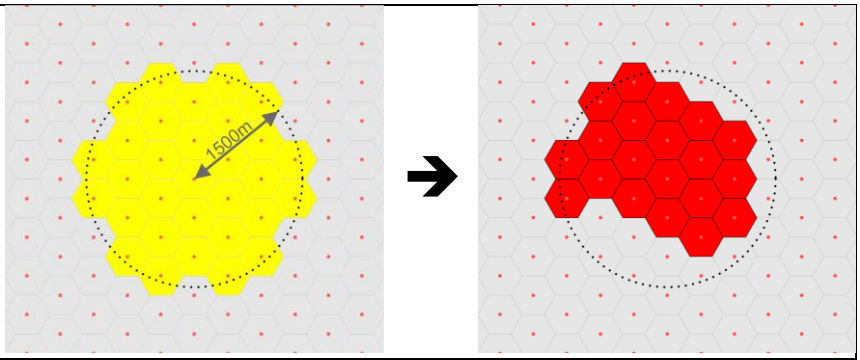
The new patch radius value is used by MinPatch to determine the size of the new patches that MinPatch adds in step 3. Initially, MinPatch identifies a new potential patch for each planning unit, with the planning unit as its centre and including the neighbouring planning units that fall within the raw patch radius. More specifically, a planning unit forms part of the new potential patch if its centroid is less than or equal to the specified radius distance from the centroid of the central planning unit. MinPatch stores this information in a file named patchPUID.dat in the input folder.

This means the patch radius value must be high enough that each new patch is larger than the minimum patch size value. It also means the user can influence patch compactness in the final MinPatch portfolios by manipulating the new patch radius values. This is because if the new patch radius value is low then the new patches are small, so there is less scope for the simulated whittling process to remove planning units and the final patches are likely to be relatively circular (Figure 1A). In contrast, if the new patch radius value is high then the new patches are larger and more planning units might be removed, so the patch shapes are likely to have more irregular shapes (Figure 1B).

Figure 1: Illustration of the results of using different new patch radius values on the shape of patches identified using MinPatch. The example is based on a set of hexagonal 25 ha planning units with a minimum patch size threshold of 500 ha and a new patch radius value of (A) 1500 m and (B) 2000 m.

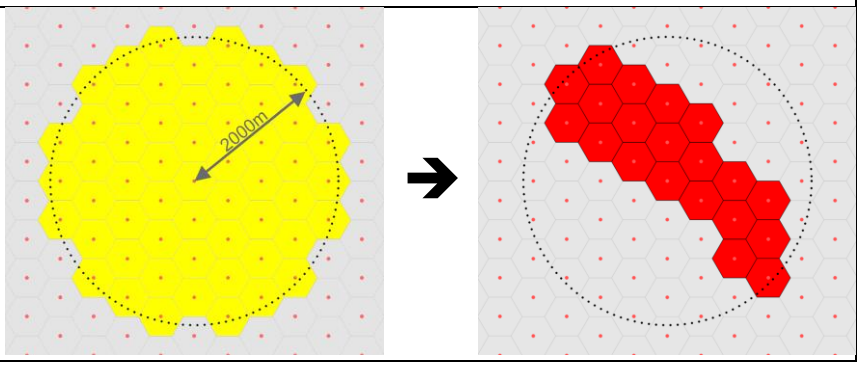
A) Setting the new patch radius of 1500m means that each new patch contains up to 31 planning units.

With a minimum patch size of 500 ha, simulated whittling can remove up to 11 planning units (assuming they are not needed to meet any targets) leaving a more compact final patch.



B) Setting the new patch radius of 2000m means that each new patch contains up to 55 planning units.

With a minimum patch size of 500 ha, simulated whittling can remove up to 35 planning units so the final patch can have a more irregular or elongated shape.



Creating the MinPatch details file

Most of the data used by MinPatch comes from the Marxan input and output files. The only file the user has to create is the MinPatch details file, which contains extra information about each planning unit listed in the Marxan pu.dat file. The MinPatch details file has to be in the same format as the Marxan input files, i.e. it is a comma delimited (csv) file with a .dat extension. For example, it could be named **minpatch.dat**. This file must contain the following five fields:

Field Name	Field Type	Notes
id	Integer	This should match the planning unit identifier values given in the pu.dat file and each value should be an integer.
area	Number	This field contains data on the area of each planning unit, as calculated and used in CLUZ. These values set the measurement units for the patch_area values, i.e. if the area values are measured in hectares then so should the patch_area values.
zone	Integer	This field specifies the zone unique identifier values. If only one zone identifier value is used then MinPatch will not undertake zonal analyses.
patch_area	Number	This field specifies the minimum area of any patch that the planning unit is included in. The minimum patch area is based on the same measurement units as the planning unit area values (e.g. m^2, hectares of km^2).
radius	Number	This field specifies the radius of any new patch that MinPatch adds that has the planning unit at its centre. The radius is based on the same measurement units as the coordinate system used in the shapefiles (e.g. m^2).

Example of MinPatch details file:

```
id,area,zone,patch_area,radius
1,7.04,1,500,1500
2,11.67,1,500,1500
3,11.53,2,750,2000
4,1.35,2,750,2000
...
```

MinPatch uses this information to create a file named **patchPUID.dat** that lists all the planning units found in the potential new patch centred around each planning unit. It is created by MinPatch the first time a particular analysis is run.

Other things to note are:

1) MinPatch uses the coordinates of the centroid of each planning unit that should be recorded in the Marxan pu.dat file in the fields named “xloc”, “yloc”. These fields are automatically added by creating the pu.dat file using CLUZ.

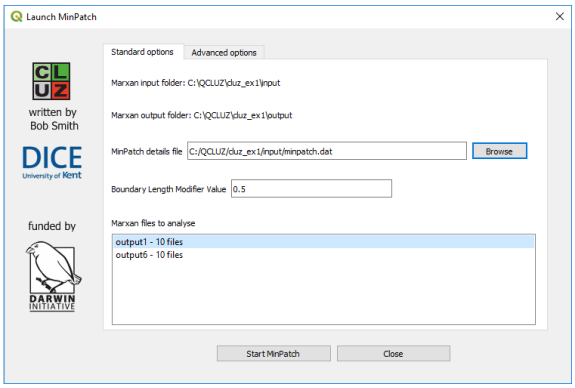
2) When setting the radius values you must ensure they create initial patches that are larger than the patch_area value. When doing so, remember that the measurement systems for the patch_area and radius can differ, e.g. you can store the area values in hectares and the radius values in metres. The measurement system for the patch_area values is taken from the one used for the area field in the MinPatch details file and the radius value is taken from the “xloc and “yloc” fields in the pu.dat file (which is taken from the coordinate system used in the planning units shapefile).

3) Make sure that all of your GIS data is projected, i.e. not measured in degrees. The area of a square degree changes with latitude and so you should use a system that measures distances using metres or something similar.

4) If a planning unit patch covers more than one zone then MinPatch will use the higher minimum patch area threshold when removing the under-sized patches and for simulated whittling.

MP Using MinPatch in CLUZ

Standard options

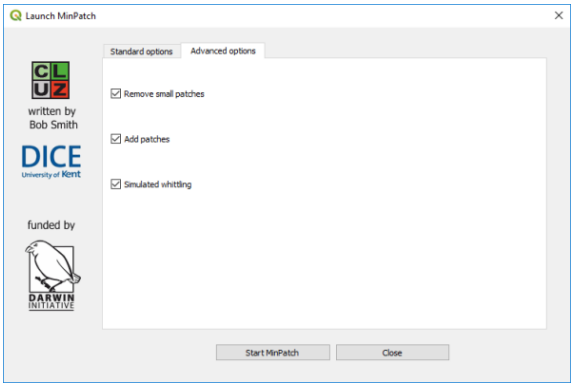


MinPatch uses the information from the CLUZ setup file to determine the **input** and **output** folders where the Marxan files are stored, so you will have needed to specify the correct folder paths in your CLUZ setup file.

You will then need to specify which **MinPatch details file** you wish to use. This should describe the details of the planning units listed in the Marxan pu.dat file.

You will also need to specify the **Boundary Length Modifier (BLM) value** to use in the analysis. In general, you should use the same BLM value as you used in the original Marxan analysis. But if you want to produce patches with more uneven edges then select a lower BLM value.

Advanced options



The advanced options let the user select whether MinPatch goes through the functions "Removes small patches", "Add patches" and "simulated whittling" (steps 2, 3 and 4 as described above).

The main reason for letting the user choose these options is so that they can sequentially add these functions to better understand the steps that MinPatch follows.

However, it can be useful to run MinPatch with all three functions turned off if you would like MinPatch to quickly produce the spatial statistics output describing the number of patches in each Marxan output. You can also use the simulated whittling option on its own to remove superfluous planning units from a portfolio, without removing patches that are below the minimum size threshold.

Details of outputs files

Fields in mp_[Marxan run name]_best.txt and mp_[Marxan run name]_r[run number].txt

planning_unit	Planning unit unique identifier.
solution	Coded 1 if planning unit is in the portfolio, 0 if it is not part of the portfolio.

Fields in mp_[Marxan run name]_summed.txt

planning_unit	Planning unit unique identifier code.
solution	Selection frequency score (number of times planning unit appeared in the portfolios produced in each run).

Fields in mp_[Marxan run name]_patchstats.csv

File_name	Name of the original Marxan file that lists the planning units found in the portfolio.
Bef_AllPatchCount	Number of planning unit patches in the original Marxan portfolio.
Bef_AllPatchArea	Combined area of the patches in the original Marxan portfolio.
Bef_medianAllPatch	Median patch size in the original Marxan portfolio.
Bef_ValidPatchCount	Number of planning unit patches in the original Marxan portfolio that meet the minimum patch size threshold.
Bef_ValidPatchArea	Combined area of the patches that meet the minimum patch size threshold in the original Marxan portfolio.
Bef_medianValidPatch	Median patch size of the patches that meet the minimum patch size threshold in the original Marxan portfolio.
Aft_AllPatchCount	Number of planning unit patches in the resultant MinPatch portfolio.
Aft_AllPatchArea	Combined area of the patches in the resultant MinPatch portfolio.
Aft_medianAllPatch	Median patch size in the resultant MinPatch portfolio.
Aft_ValidPatchCount	Number of planning unit patches in the resultant MinPatch that meet the minimum patch size threshold (this should be the same as the Aft_AllPatchCount value).
Aft_ValidPatchArea	Combined area of the patches that meet the minimum patch size threshold in the resultant MinPatch portfolio (this should be the same as the Aft_AllPatchArea value).
Aft_medianAllPatch	Median patch size of the patches that meet the minimum patch size threshold in the resultant MinPatch portfolio (this should be the same as the Aft_medianAllPatch value).
PortfolioPUCost	The sum of the planning unit cost values for each planning unit in the resultant MinPatch portfolio.
PortfolioBoundLength	The total length of the external boundary of the resultant MinPatch portfolio.
PortfolioBoundCost	The resultant MinPatch portfolio's boundary cost, calculated as the total external length multiplied by the Boundary Length Modifier.
PortfolioTotalCost	The total cost of the MinPatch portfolio, calculated by summing the planning unit cost and the boundary cost (each portfolio should meet all the targets, so there should be no associated conservation feature penalty costs).

Fields in mp_[Marxan run name]_zonefeaturestat[zone ID].csv

File_name	Name of the original Marxan file that lists the planning units found in the portfolio.
Feat_[feature ID]	Proportion of the target for the conservation feature found in that zone. Date on each feature found in the planning region should be represented in a different field.

Fields in mp_[Marxan run name]_ zonestats.csv

File_name	Name of the original Marxan file that lists the planning units found in the portfolio.
Zone[Zone ID]_Area	The total area of the portfolio found in the zone.
Zone[Zone ID]_Cost	The total planning unit cost of the portfolio found in the zone.

References

Metcalfe, K, Vaughan, G, Vaz, S and Smith, RJ (2015). Spatial, socio-economic, and ecological implications of incorporating minimum size constraints in marine protected area network design. *Conservation Biology*, **29**, 1615–1625. Download here.

Smith, RJ, Di Minin, E, Linke, S, Segan, D and Possingham, HP (2010). An approach for ensuring minimum protected area size in systematic conservation planning. *Biological Conservation*, **143**, 2525-2531. Download here.