MARXAN conservation solutions

The Nature Conservancy

User Manual

MARXAN with Zones

Version 2.0.1

Issued: November 2020

2020 update by: Norma Serra-Sogas, Alessia Kockel, Brooke Williams, Edward T. Game, Hugh P. Possingham & Jennifer McGowan

Originally written in 2008 by: Matthew Watts, Carissa Klein, Romola Stewart, Ian Ball & Hugh Possingham

PacMARA Pacific Marine Analysis & Research Association

The University of Queensland Australia

Suggested citation for this manual:

Serra N, Kockel A, Williams, B., Watts, M., Klein, C., Stewart, R., Ball, I., Game, E., Possingham, H., & McGowan J. (2020). Marxan with Zones User Manual. For Marxan with Zones version 1.0.1 and above. The Nature Conservancy (TNC), Arlington, Virginia, United States and Pacific Marine Analysis and Research Association (PacMARA), Victoria, British Columbia, Canada.

Citation for Marxan with Zones software:

Watts, M. E., Ball, I. R., Stewart, R. S., Klein, C. J., Wilson, K., Steinback, C., Lourival, R., Kircher, L., Possingham, H. P. (2009). Marxan with Zones: Software for optimal conservation based land- and sea-use zoning. *Environmental Modelling and Software, 24(12), 1513-1521.*

Preface

About this Manual

This manual is intended to equip readers with the basic knowledge required to use Marxan with Zones (Watts et al., 2009), an extension of the Marxan software (Ball et al., 2009). It covers all the relevant parameters and necessary data inputs, as well as key steps to successfully execute the program and interpret the outputs. It provides some guidance on the sorts of problems Marxan with Zones can solve but does not cover the full range of possible uses and applications. Some case studies and complementary literature are referenced in the manual to guide readers to additional sources of information and guidance.

Marxan with Zones has the capacity to address complex planning problems with diverse objectives. We strongly recommend familiarizing yourself with Marxan and its appropriate use before diving into your first Marxan with Zones exercise. It is important to understand how both Marxan and Marxan with Zones work to avoid their inappropriate use. Marxan with Zones can be a very powerful tool, but if misused, it can undermine a great deal of hard work in collecting and collating good data, not to mention providing misleading advice and undermining the credibility of the decision support tool for conservation planning and land- or ocean-use zoning.

Complementary Literature

This manual should be used in tandem with the Marxan User Manual and the Marxan Good Practices Handbook (MGPH), which are available for download from the Marxan website (https://marxansolutions.org/). Though these manuals focus on standard Marxan, much of the content and guidance provided are also applicable to Marxan with Zones. Together, these three manuals should provide the resources needed to undertake skilled and defensible analysis using Marxan with Zones. In addition, we strongly suggest reading some of the peer reviewed articles that use Marxan with Zones in various zoning applications – some of which are cited in this manual. These articles demonstrate what types of questions the software can answer, how problem definitions are set up, the kinds of data that can be used, and how different objectives and constraints influence the resulting outputs.

Acknowledgements

The ongoing development of Marxan software would not be possible without the financial support of generous donor organisations from around the world. Early supporters were Ecotrust, who provided substantial funding for the development of the software to support the planning of the California Marine Life Protection Act. In addition, the Australian National Heritage Trust contributed substantial funds to the development of the software and the underpinning research program. We also acknowledge initial funding from the Australian Applied Environment Decision Analysis centre. This updated manual was generously supported by The Nature Conservancy.

Disclaimer

The planning scenarios presented in this manual are all hypothetical. They have been developed exclusively for demonstration purposes.

Table of Contents

List of Figures

List of Tables

List of Information Boxes

List of Acronyms

BLM	Boundary Length Modifier
FPF	Feature Penalty Factor
GIS	Geographic Information System
MGPH	Marxan Good Practices Handbook
MPH	Minimum Proportion Met
MSP	Marine Spatial Planning
SCP	Systematic Conservation Planning

1 Introduction

1.1 What is Marxan with Zones?

Marxan with Zones (Watts et al., 2009) is a free and open source software designed to support spatial planning for multi-use zones. Marxan with Zones allows for multiple zones, zoning contributions, costs, and the spatial relationships between zones to all be considered in spatial optimization. This tool can be applied in various contexts to help identify configurations of sites (or 'planning units') that contribute to a wide range of ecological, social, cultural, and economic objectives. This makes Marxan with Zones well placed to support local and regional planning across marine, freshwater and terrestrial areas, where the goal is often to achieve a balance across a range of competing objectives.

Marxan with Zones is an extension of the Marxan software (Ball et al., 2009), the most widely used decision support tool for conservation planning. Marxan was designed to address a particular type of conservation problem, known as the 'minimum set problem', where the goal is to achieve the minimum target amount of conservation features for the smallest possible cost (see Box 1 for key terms and definitions). To solve this problem, Marxan uses a simulated annealing algorithm. The algorithm works by minimizing one cost function in the pursuit of meeting targets for features. With each run, it generates a solution that assigns each planning unit in the study region as either selected (e.g., protected area) or not selected (e.g., unprotected area).

Marxan with Zones follows the same 'minimum set problem' framework, while:

1. assigning planning units to the most appropriate zone given the problem set-up (note: zone assignment may change across runs); and
2. allowing the user to incorporate multiple costs in the analysis according to the zone and the activities that are allowed or not allowed in the zone.

These advanced functions of Marxan with Zones are discussed in Sections 1.1.1 and 3.2.

Box 1. Key Terms and Definitions

- Planning region (*also known as planning area, study area or extent*): The spatial domain over which the planning process occurs. This area is divided into smaller planning units.
- Planning units: Spatial units within the entire planning region, which can be defined using grids, hexagons, or irregular shapes. Each planning unit contains information on the distribution of features and associated costs.

- Zoning framework: The set of rules by which activities are permitted, regulated, or prohibited for a specific purpose/objective across a planning region.
- Zone: The discrete spatial representation of the zoning framework.
- Feature: The feature for which a target is set for inclusion in one or more zones. Features can be ecological (e.g. habitats, species, migration routes), socioeconomic (e.g., farming, fishing, sustainable forestry), and cultural (e.g., heritage sites, dreaming sites, or traditional use areas).
- Target: The 'overall target' is the minimum quantity or proportion of the feature to be included across all zones.
- Zone target: The minimum quantity or proportion of the feature in the planning region to be included in a specific zone.
- Zone contribution: A multiplier used to assign differential contributions to meeting the overall targets across different zones.
- Costs: Values that are minimized in the pursuit of achieving targets. Each planning unit can be assigned one or more cost. Costs may vary (e.g., land acquisition cost, opportunity cost, management cost) but should reflect various consequences associated with allocating a planning unit to a specific zone under the zoning framework.
- Objective function: The mathematical expression of the Marxan optimization problem which seeks to meet targets while minimizing costs and keeping solutions compact.
- Feature penalty factor: A multiplier for the penalty applied to the objective function when a feature target is not met in the current reserve scenario.
- Solution: The binary output of a Marxan with Zones run, where each planning unit is assigned to a specific zone (this allocation may change across runs).
- Score: The value assigned to each run based on the performance of the solution against the sum of the planning unit costs, boundary costs, and penalties. The Score is useful for comparisons.
- Best solution: The solution with the lowest objective function score.
- Selection frequency (*also known as 'summed solution'*): The number of times a given planning unit is assigned to a particular zone across a series of solutions (or 'runs').
- Run: The number of repeated times the algorithm is applied to solve the problem and produce a solution. This is defined by the user (e.g., if Marxan with Zones is set to run 100 times, it will generate 100 solutions, which is considered the best practice number of runs).

1.1.1 The Conceptual Case for Marxan with Zones

Standard Marxan is designed to solve a two-zone problem, such as identifying areas for protection. This approach inherently assumes the areas that are not selected - the “unprotected” zone - contribute nothing to the objectives of the zoning plan. This is sometimes referred to as the “Scorched Earth” assumption – meaning we get to keep the biodiversity in the protected zone while none of the biodiversity found in the unprotected zone counts towards our objectives. In reality, this “unprotected” zone is made up of many different management areas which regulate activities across the land and sea. For example, a marine conservation zone is often accompanied by regulated fisheries management zones and/or community or traditional fishing areas. Biodiversity is generally managed in these zones to varying degrees and therefore contribute to our objectives in different ways.

Similarly, different zones will impact different industries in different ways and Marxan with Zones can accommodate those differences in its consideration of costs. For example, the cost of allocating a planning unit to a marine reserve zone that prohibits fishing, could be a loss of fishing grounds resulting in less overall catch (i.e., a fishing opportunity cost). Yet this cost would not apply if the same planning unit is allocated to a multiple use zone where fishing is permitted.

Marxan with Zones accommodates these considerations and allows users to develop zoning plans that can account for different conservation, social, cultural and economic objectives simultaneously. It also helps users explore trade-offs between conflicting objectives (e.g., conservation versus socioeconomic objectives), and reduce conflict between different activities (e.g., mining, forestry, and conservation).

Box 2. Protected Areas Designed via Marxan versus Marxan with Zones

A study in California by Klein et al. (2010) compared marine protected area (MPA) networks designed via Marxan and Marxan with Zones in terms of the fishing value lost to each fishery industry. The potential MPA network produced by Marxan had 10-30% proportion of fishing value lost. However, when Marxan with Zones was used it only led to 2-10% proportion of fishing value lost. The reduction in the impact on fisheries is due to the inclusion of targets for each fishery and the definition of zones where those targets can be met within Marxan with Zones.

1.2 Marxan with Zones in Practice

Marxan with Zones can be applied to address a broad range of spatial planning challenges. Like Marxan, it is being employed worldwide as a decision support tool for systematic conservation planning (Box 3), with a growing number of case studies in the scientific and grey literature show casing its versatility at local, national, and even transnational scales across terrestrial, freshwater, and marine systems.

Some applications of Marxan with Zones are listed below and you can review additional case studies at https://marxansolutions.org:

- Designing protected area networks to achieve conservation objectives and resolve user conflict (e.g., Grantham et al., 2013; Jumin et al., 2018; Parker et al., 2015; Reyers et al., 2012)
- Land-use planning in multifunctional landscapes (e.g., Law et al., 2017)
- Marine zoning within a broader marine spatial planning process (e.g., Agostini et al., 2010)
- Zoning for freshwater conservation (e.g., Domisch et al., 2019; Hermoso et al., 2015)
- Exploring trade-offs between conflicting objectives in land-use planning (e.g., Adams et al., 2016; Fastré et al., 2020), marine conservation planning (e.g., Klein et al., 2010; Mazor et al., 2014; Ruiz-Frau et al., 2015), and marine spatial planning (e.g., Yates et al., 2015)
- Addressing considerations of social equity and fairness in conservation planning (e.g., Domisch et al., 2019; Gurney et al., 2015; Kockel et al., 2019, 2020; Weeks et al., 2010)

Box 3. Systematic Conservation Planning

Systematic conservation planning (SCP) is a framework for planning and developing conservation areas to achieve set objectives (Margules & Pressey, 2000; Pressey & Bottrill, 2009). It is a major departure from conventional conservation planning approaches, which have often been applied to select conservation areas based on urgency, scenery, and ease of designation (Kukkala & Moilanen, 2013).

SCP is widely considered 'best practice' in conservation because it facilitates a transparent, inclusive and defensible decision-making process. The importance of these principles for conservation is discussed in the Marxan Good Practice Handbook (MGPH), available at www.marxansolutions.org.

The SCP framework consists of 11 stages (Pressey & Bottrill, 2009):

1. Scoping and costing the planning process
2. Identifying and involving stakeholders
3. Describing the context for conservation areas
4. Identifying conservation goals
5. Collecting data on socio-economic variables and threats
6. Collecting data on biodiversity and other natural features
7. Setting conservation objectives/targets
8. Reviewing current achievement of objectives
9. Selecting additional conservation areas
10. Applying conservation actions to selected areas
11. Maintaining and monitoring conservation areas

The ninth stage, commonly referred to as 'spatial conservation prioritization', involves the selection of new sites to achieve conservation objectives and is where Marxan software plays a critical role.

1.3 Mathematical Formulation of Marxan with Zones

The aim of the Marxan with Zones software is to minimize the total cost of the solution set subject to meeting the representation targets. This is the Marxan with Zones minimum set problem, which can mathematically be defined as:

$$\text{minimize} \quad \sum_{i=1}^{m}\sum_{k=1}^{p} c_{ik}\, x_{ik} + b \sum_{i1=1}^{m}\sum_{i2=1}^{m}\sum_{k1=1}^{p}\sum_{k2=1}^{p} cv_{i1,i2,k1,k2}\, x_{i1,k1} x_{i2,k2} \qquad (1)$$

$$\text{subject to} \quad \sum_{i=1}^{m}\sum_{k=1}^{p} a_{ij}\, a_{ik}\, x_{ij} \geq t1_j \;\forall \qquad (2)$$

$$\text{and subject to} \quad \textstyle\sum_{i=1}^{m} a_{ij} x_{ik} \;\geq\; t2_{jk} \;\forall\; j \textit{ and } \forall\; j \qquad (3)$$

where m is the number of planning units and p is the number of zones (Watts et al., 2009).

The first term of Equation 1 represents the sum of the costs for a configuration of planning units where each planning unit is allocated to a particular zone and is composed of a control variable and cost matrix. The control variable $x_{ik} \in \{0,1\}$ records which of the k zones each planning unit i is allocated to. A value of 1 indicates that the planning unit i is allocated to zone k, while a zero value means the planning unit i is not allocated to zone k. Each planning unit can only be allocated to one zone, hence $\sum_{k=1}^{p} x_{ik} = 1 \;\forall\; i$. The cost matrix c_{ik} is the cost of placing each planning unit i in zone k.

The second term of Equation 1 represents the connectivity cost of a configuration of planning units assigned to particular zones, and is composed of a connectivity matrix recording the cost of the connections between planning units $i1$ and $i2$ if and only if $i1$ is in zone $k1$ and $i2$ is in zone $k2$.

In Equation 2, a_{ij} is a feature matrix that records the amount of each feature j in each planning unit i; the parameter $t1_j$ is a representation target for each feature, j, that records the amount of each feature required to be included in the zone configuration. The term ca_{jk} is a contribution matrix that records the level of contribution (e.g. level of protection if planning for a multi-zone protected area) offered to each feature j by each zone k. Typically, this contribution will be 1 for zones in which the feature achieves full representation, 0 for zones not suitable for the protection of the feature and an intermediate value for a zone that offers partial inclusion for a feature. For example, when planning for a multi-zone protected area network, a conservation feature might enjoy full representation (100% contribution) in a conservation zone, no representation in zones where natural resources (e.g. timber, fish, etc.) are extracted (0% contribution) and partial protection where ecologically sensitive natural resource extraction is allowed (0.5% contribution).

In Equation 3, $t2_{jk}$ is a zone target matrix that records the amount of each feature j required to be captured in a particular zone k. For example, the user may specify that a particular species have at least half of its feature target conserved in full no-take reserves.

If both targets $t1$ and $t2$ are used, the software attempts to satisfy their requirements simultaneously. Users should take care to enter targets that can be simultaneously achievable to avoid situations where the algorithm cannot find an answer.

A feature penalty equation is used to implement the two target constraints in the Marxan with Zones objective function (5):

$$\sum_{j=1}^{n} FPF_j FR_j \left(H(s1)\left(\frac{s1}{t1_j}\right) + \sum_{k=1}^{p} H(s2)\left(\frac{s2}{t2_{jk}}\right) \right) \tag{4}$$

where there are *n* features under consideration.

The penalty is zero if every feature *j* has met its targets $t1$ and $t2$ in the selected system, and it is greater than zero if both or one of the targets are not met.

The terms FPF_j and FR_j are the feature penalty factor and the feature representation respectively, which are scaling factors used when a feature has not met its targets. FPF_i is a scaling factor which determines the importance of meeting the targets for feature *j*. FR_j is computed as the representation cost of meeting the targets of feature *j.* This representation cost is given in terms of the configuration cost plus the connectivity cost (Eq.1) and it is computed for each feature j.

The shortfalls $s1$ and $s2$ are the amount of the two different representation targets not met and are given by $s1 = t1_j - \sum_{i=1}^{m} \sum_{k=1}^{p} a_{ij} ca_{jk} x_{ik}$ and $s2 = t2_{jk} - \sum_{i=1}^{m} a_{ij} x_{ik}$. They are reported as a proportion and they equal to 1 when feature j has not met its target and approached to zero when the representation levels approach the target amounts. The Heaviside function H ensures that each equation becomes zero when the representation targets are greater than the target amount.

Combining Equations 1 and 4 gives the objective function for Marxan with Zones:

$$\sum_{i=1}^{m}\sum_{k=1}^{p} c_{ik}x_{ik} + b\sum_{i1=1}^{m}\sum_{i2=1}^{m}\sum_{k1=1}^{p}\sum_{k2=1}^{p} cv_{i1,i2,k1,k2}\, x_{i1,k1}x_{i2,k2} + FPF_jFR_j\left(H(s1)\left(\frac{s1}{t1_j}\right) + \sum_{k=1}^{p} H(s2)\left(\frac{s2}{t2_{jk}}\right)\right) \tag{5}$$

The objective function gives a value to a configuration of planning units that belong to a defined number of zones. As in Marxan, Marxan with Zones uses simulated annealing to minimize the objective function score by varying the control variable x_{ik}, which tells us which planning units i is allocated to which zone k. The lower the score, the better we are meeting set representation targets at the lowest cost.

1.4 How Does Marxan with Zones Finds Good Solutions?

The number of possible solutions to even a simple zoning problem is vast. Computer algorithms have been developed to help find good solutions to this problem. An algorithm is a mathematical process or set of rules used for problem solving. Within Marxan with Zones, there are two algorithms that can be used to formulate a solution: simulated annealing and iterative improvement. Each of these algorithms can be used alone or in combination with each other. The user can define which algorithm, or combination of algorithms, Marxan with Zones must use to formulate a solution (See Section 5.3.1.10). More information about these algorithms is found in Appendix B-2.

Box 4. Marxan with Zones Score Calculation

The objective function for Marxan with Zones is:

$$\sum_{i=1}^{m}\sum_{k=1}^{p} c_{ik}x_{ik} + b\sum_{i1=1}^{m}\sum_{i2=1}^{m}\sum_{k1=1}^{p}\sum_{k2=1}^{p} cv_{i1,i2,k1,k2}\, x_{i1,k1}x_{i2,k2}$$
$$+ FPF_j FR_j\left(H(s1)\left(\frac{s1}{t1_j}\right) + \sum_{k=1}^{p} H(s2)\left(\frac{s2}{t2_{jk}}\right)\right)$$

which can be simplified to:

$$\overbrace{\sum_{PUs}\text{Cost}}^{1} + \overbrace{BLM\sum_{PUs} CVxBoundary}^{2} + \overbrace{\sum_{\substack{Con\\Value}} FPF \times Penalty}^{3}$$
$$= \boldsymbol{Marxan\ with\ Zones\ Score}$$

1. The first term of the simplified equation is the cost of the zone configuration; the linear combination of costs of all the planning units within the zone configuration, by zone.
2. The second term is the boundary length and fragmentation, which can be used to apply a control on the level of fragmentation in the zone configuration. In Marxan with Zones, the multiplicative factor BLM (or boundary length modifier) needs to be set to 1 to allow the boundary length to be added to the cost. Also, there are multiplicative factors within the connectivity matrix *CV* which allow the user to explore different spatial configurations by applying weights between planning units allocated to different zones. The units of the weights within the connectivity matrix have to be set in a way that allow the boundary length to be comparable to the cost (or costs) measure of the planning units. For example, if the most costly planning unit is 100 and the longest boundary length is 1000, then the weights within the connectivity matrix should start at 0.1.
3. The third term is the total penalty for not achieving feature targets, either the contribution targets or the zone targets.

More information about the objective function can be found in Appendix B-1.

2 Download, Software Requirements & Supporting Software

2.1 Software Download

Marxan with Zones can be downloaded for free from https://marxansolutions.org/software. The download package includes the latest version of the software, Marxan with Zones version 2.0.1, along with some supporting materials.

The following files are included in the download package:

- MarZone_x64 (the Marxan with Zones program executable for Windows);
- MarxanZone_v201_Linux64 (the Marxan with Zones program executable for Linux);
- MarZone_v201_Mac64 (the Marxan with Zones program executable for Mac);
- input.dat (an example Input Parameter File);
- a folder labelled ‘Tutorial_Marxan_with_zones”, containing different examples of required and optional input files for running Marxan with Zones;

These files can be saved anywhere on your computer.

2.2 System Requirements

The Marxan with Zones software is available for different operating systems, including Windows, Mac, and Linux. The system requirements for running Marxan with Zones are somewhat modest. As a rule of thumb, if a computer is powerful enough to run GIS software, then it will be adequate for running Marxan with Zones. The number of zones, planning units, features, and costs that can be incorporated into Marxan with Zones is limited by the applications memory address space, which is currently 512 GB with a 64 bit computer operating system and 64 bit C compiler.

The more planning units, features, and optional files you have, the slower Marxan with Zones will run. Of course, the more powerful your computer (MHz and RAM), the faster Marxan with Zones will run. Depending on these factors, the time required for Marxan with Zones to provide good solutions to your problem can range from minutes to days.

2.3 Supporting Software

In this manual, we describe how to run Marxan with Zones as a stand-alone program. However, there are several user interfaces that can assist in running Marxan with Zones. These include Zonae Cogito, CLUZ (Conservation Land-Use Zoning), the ArcMarxan plugin for ArcGIS, and the QMarxan plugin for QGIS. Many users have found these interfaces particularly helpful for generating certain input files and displaying key outputs. Guidance on using these programs can be obtained from corresponding websites or user manuals. Table 1 provides a quick comparison view of the different supporting software/plugins for Marxan with Zones. A brief description of each tool is provided in the following subsections.

Table 1. *Comparison of Marxan with Zones supporting software*

		Functionalities			
Software	GIS software required	Create input files	Runs Marxan with Zones	Parameter calibration	Direct visualisation of results
Zonae Cogito	N/A	No[1]	Yes	Yes	Yes
CLUZ v.3	QGIS	Yes[2]	Yes	Yes	No
ArcMarxan Toolbox v.2.0.1	ArcGIS	Yes[2]	No	No	No
QMarxan Plugin v.2.0.1	QGIS	Yes[2]	No	Yes	No

[1] Users can modify some parameters of Marxan input files.
[2] Only Marxan input files, which are also required to run Marxan with Zones.

2.3.1 Zonae Cogito

Zonae Cogito comes from the Latin 'Zonae', meaning zones, and 'Cogito', meaning to think or reflect on. In other words, the title means to think about zones. The purpose of the software is to act as a decision support and database management system for the family of Marxan software. It incorporates open source GIS software components, and is a freely available software package at https://marxansolutions.org/software. Zonae Cogito was written by Matthew Watts and Romola Stewart from The Ecology Centre, University of Queensland. The current version of Zonae Cogito is compatible with 32 bit and x64 Windows operating systems exclusively.

2.3.2 CLUZ (Conservation Land-Use Zoning)

CLUZ v.3 (Conservation Land-Use Zoning software) is a QGIS plug-in designed to support users in the design of protected area networks and other conservation landscapes and seascapes. It can be used for on-screen planning and acts as a link for Marxan and Marxan with Zones. CLUZ was developed by Bob Smith and funded by the UK Government's Darwin Initiative. The plug-in can be downloaded and installed directly though QGIS and the manual and tutorials can be accessed at https://anotherbobsmith.wordpress.com/software/cluz/

2.3.3 ArcMarxan and QMarxan plugins for ArcGIS and QGIS

The ArcMarxan v.2.0.1 and QMarxan Toolboxes v.2.0.1 are a free and open source python plugin for ArcMap 10.2 (and above) and QGIS 3.0 (and above) respectively. These plugins were created by Apropos Information Systems (https://aproposinfosystems.com/). These toolboxes are only equipped to support the development of the standard set of input files for Marxan (i.e., input.dat, feat.dat, pu.dat, puvfeat.dat and bound.dat). Additional files for running Marxan with Zones (including optional files) cannot be developed using these tools (see Section 5.1 for alterative methods to generate new files for Marxan with Zones).

Download ArcMarxan at:

https://aproposinfosystems.com/en/solutions/arcgis-plugins/arcmarxan-toolbox/

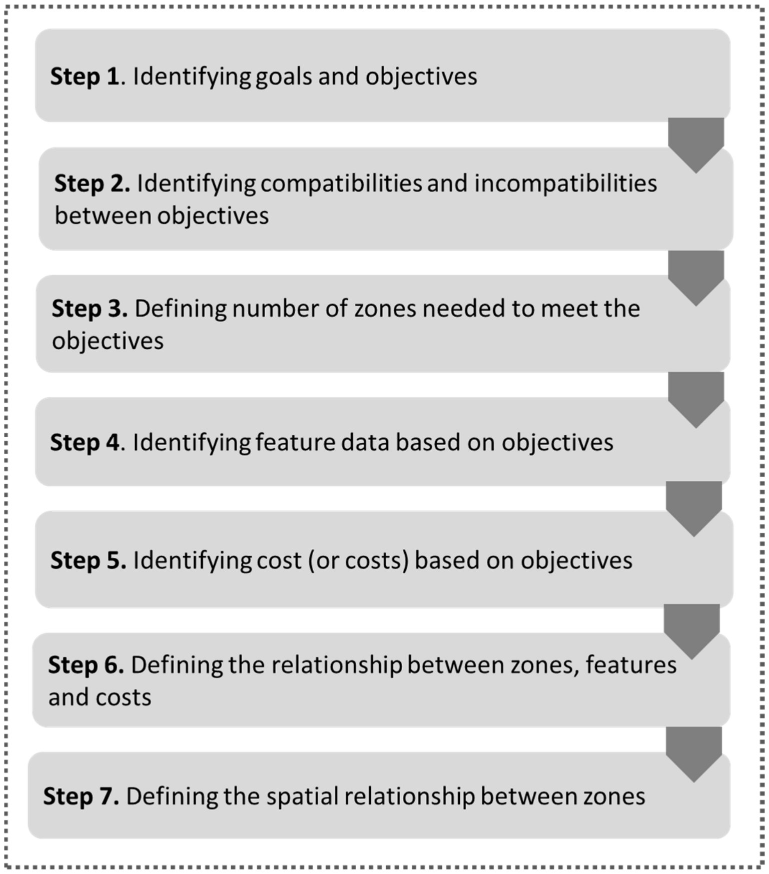
Download QMarxan at:

https://aproposinfosystems.com/en/solutions/qgis-plugins/qmarxan-toolbox/

3 Getting Started

3.1 Defining a Zoning Problem

Below we describe one approach to framing a zoning planning process. The first step is to define clear goals and objectives for the zoning framework (Figure 1). The next step is to identify compatibilities and incompatibilities between the different objectives. This information will in turn inform decisions on how to set up the problem with respect to how many zones are needed, what feature data should be collected or compiled, what constraints and costs should be considered in each zone, and which feature targets and zone contributions should be set for the selected zones. Following these decisions, the user can prescribe preferred spatial relationships between zones, such as the degree of spatial compactness and buffering of zones. Once all steps have been completed, it is common practice and generally recommended to analyse additional scenarios to examine trade-offs between objectives.



Step 1. Identifying goals and objectives
Step 2. Identifying compatibilities and incompatibilities between objectives
Step 3. Defining number of zones needed to meet the objectives
Step 4. Identifying feature data based on objectives
Step 5. Identifying cost (or costs) based on objectives
Step 6. Defining the relationship between zones, features and costs
Step 7. Defining the spatial relationship between zones

Figure 1. *Conceptualization of a Marxan with Zones problem*

3.2 An Example Case Study for Defining a Marxan with Zones Problem

This section illustrates a hypothetical case study following the steps outlined in the Section 3.1. The case study focuses on designing a multi-use marine park for Rottnest Island, Australia. The planning scenario presented here is based on actual biodiversity and human use data compiled by the Department of Environment and Conservation for the Rottnest Island Authority, as part of the development of the Rottnest Island Marine Management Strategy. Yet the scenario is completely hypothetical. It has been developed exclusively for demonstration purposes and should not be considered as a reflection of the real world.

Step 1: Identifying goals and objectives

The goal for Rottnest Island is to identify a zoning configuration for a multi-use marine park to conserve marine biodiversity, while maintaining ongoing recreational and fishing activities. To address this goal, three quantitative objectives are identified for the marine park (Table 2).

Table 2. *Objectives for the Rottnest Island multi-use marine park*

Type of Objective	Objective
Conservation	Conserve at least 40% of each biodiversity feature
Recreation	Maintain at least 80% of current areas for non-extractive recreational and tourism activities
Fisheries	Maintain at least 80% of current fishing areas

Step 2: Identifying compatibilities and incompatibilities between objective

The next step is to identify compatibilities and incompatibilities between the objectives (

).

Table 3. *Compatibility matrix for objectives (✓ = somewhat compatible, ✕ = not compatible)*

Activity Objective	Fisheries	Recreation	Conservation
Conservation	✕	✓	N/A
Recreation	✓	N/A	
Fisheries	N/A		

In this example, conservation and fisheries objectives are identified as incompatible because fishing activities may cause significant negative impacts to natural resources (e.g., habitat damage, stock depletion). Recreation activities, such as scuba diving tourism, tend to have less of an environmental impact, but may still result in environmental damage (e.g., trampling, anchor damage, fuel/oil spillage) that can hinder conservation outcomes. Yet recreation and tourism activities often depend on healthy environments, and when managed well, these activities can contribute to the conservation of natural environments. Conservation and recreation objectives are therefore identified as 'somewhat compatible', although it is recognized that there should be some areas where no human activities are allowed to ensure exclusive biodiversity conservation goals are achieved. Likewise, fisheries and recreation objectives are identified as 'somewhat compatible' but given that these activities may sometimes conflict with one another, separate zoning allocations for these activities is desirable.

Step 3: Defining number of zones

Based on the information outlined in

, a zoning framework with three management zones is identified for Rottnest Island (Table 4). The zones differ based on their restrictions on human activities.

Table 4. *Proposed zone categories for Rottnest Island*

Zone Name	Zone definition
Zone 1: Multiple Use	Multiple use zone permits recreational and fisheries activities
Zone 2: Partial Protection	Partially protected zone only permits recreational activities; fishing is not allowed
Zone 3: High Protection	Fully protected no-take zone for biodiversity conservation; no human activities allowed

Step 4: Identifying feature data based on objectives

In this case study, ecological data represent twenty-eight biodiversity features, including benthic habitats, coastal landforms and marine species. In addition, human use data was collected for activities that take place in the waters around Rottnest Island. These activities are divided into recreational activities (e.g., surfing, diving, recreational boat mooring and shipwrecks) and fishing activities (e.g., trolling, shore-based fishing, etc.).

Step 5: Identifying cost (or costs) based on objectives

This case study includes different costs in each zone based on the activities allowed in the zones. The following costs are considered in this scenario: planning unit area, distance to shore and total fisheries catch.

Step 6: Identifying relationships between zones, features and costs

As outlined in Table 2, broad quantitative targets at the planning region level have already been specified for each objective. It is also possible to use zone specific targets (see Section 5.4.5 for an example), but for simplicity, these are not applied in this example. Instead, specific zone contributions are defined according to the permitted activities and their influence of different objectives (Table 5). In this case, some feature targets could be achieved across a combination of zones, where some zones contribute more than others at meeting specific targets. For example, areas designated as partial protection zones will contribute 100% to meeting targets for recreational features, and 20% to meeting conservation feature targets. If important habitats are placed in Zone 1, we assume they may be severely impacted by human activities and so we do not consider habitats or species in that zone towards achieving our conservation targets. See Box 5 for more examples related to zone contributions.

Table 5. *Level of contribution towards meeting objectives for each zone*

Activity Objective	Zone 1: Multiple Use	Zone 2: Partial Protection	Zone 3: High Protection
Conservation	0%	20%	100%
Recreation	20%	100%	0%
Fisheries	100%	0%	0%

Table 6 contains information about how costs are applied in each zone. We use a binary classification to organise the information where a 1 means the cost should be considered in that zone, while a 0 means it is not relevant in zone. In this scenario, planning unit area cost is only applied in Zone 3 to minimize the cost associated with monitoring and managing large amounts of conservation areas. Fisheries catch cost is applied in Zone 2 and Zone 3 because no fishing activities are allowed in either of each zone. Finally, the inverse distance to shore cost is applied in Zone 1 and 2 to encourage planning units found near shore to be assigned to either of the two zones, given that nearshore areas are generally easier to access for fishing and recreation.

Table 6. *Information about which costs will be applied in each zone-*

Cost	Zone 1: Multiple Use	Zone 2: Partial Protection	Zone 3: High Protection
Planning Unit Area	0	0	1
Fisheries catch	0	1	1
Inverse distance to shore	1	1	0

<u>Step 7: Defining spatial relationship between zones</u>

To minimize conflict between conservation and fisheries, the spatial configuration between zones is set up to encourage spatial separation between the high protection zone (Zone 3) and the multi-use zone (Zone 1). This is achieved by calibrating the zone boundary cost (see Section 5.4.2 for more information) to encourage the high protection zone (Zone 3) to be buffered by the partial protection zone (Zone 2). An example of a resulting solution (i.e., a potential zoning system) for Rottnest Island is shown in Figure 2.

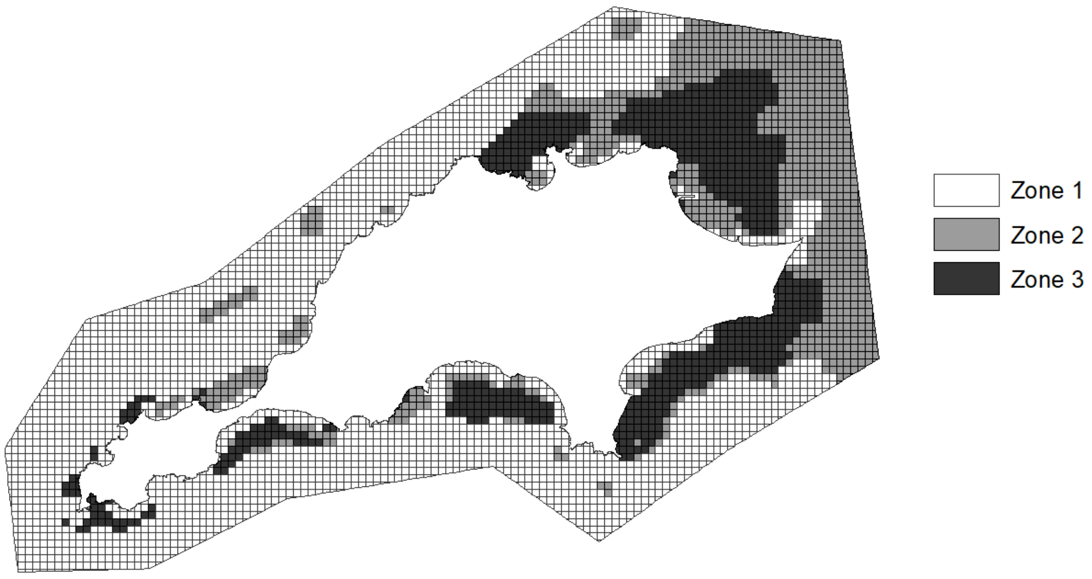


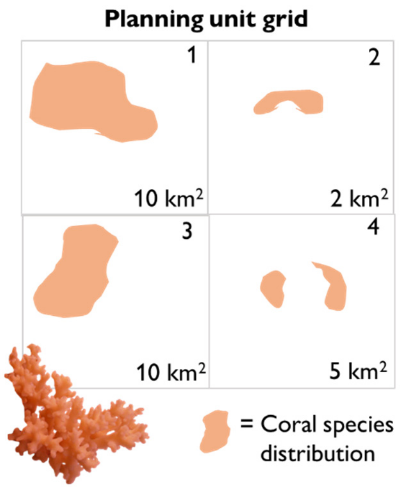
Figure 2. *Example solution for Rottnest Island case study*

Box 5. Toy Problems for Zone Contribution

The following toy problems convey how the zone contribution influences Marxan with Zones solutions.

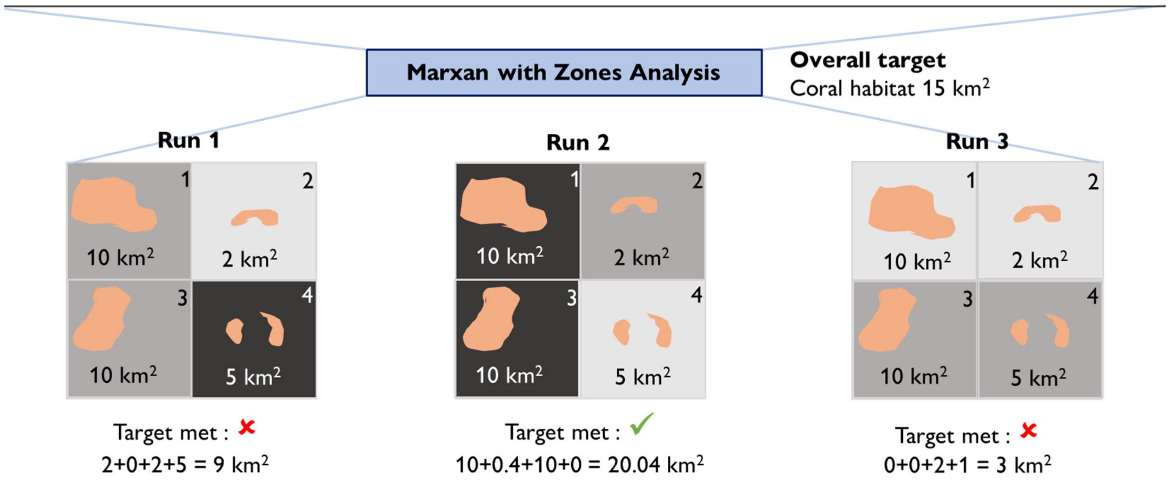
Example 1 - Coral

In this example, we have single coral feature with a distribution across 4 planning units (PU) and the amount of coral found in each PU is specified. Based on the zone contributions assigned in the table below (same as Table 5), we can examine how this contribution works in Marxan's accounting of the overall target achievement, which we have set at 15 km^2 coral habitat.



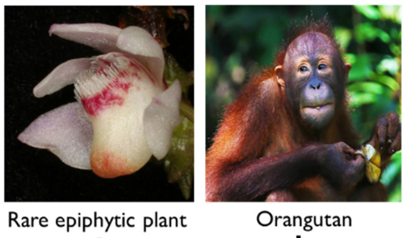
Zones, relative feature contributions (based on the zone they are located in), and contribution (km^2)

Zone description		Zone 1 – Multiple use	Zone 2 – Partial Protection	Zone 3 – High Protection
Relative zone contribution		***0%***	***20%***	***100%***
Planning unit ID	1	0	2	10
	2	0	0.4	2
	3	0	2	10
	4	0	1	5

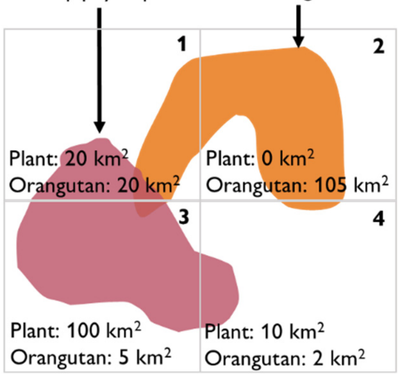


Placing a planning unit with coral habitat into Zone 1 or Zone 2 means the contribution of that planning unit towards the overall target for the coral feature is multiplied by the value specified in table above. In solution 1, the coral target is not met because the partial protection zones only give 20% of the coral found in PU1 and PU3. In solution 2, PU1 and PU3 are allocated to the High Protection Zone, so we count the full amount of coral reef and meet our target.

Example 2 - Orangutans & Epiphytes



One of the strengths of Marxan with Zones is that you can assign unique zone contributions to every feature and zone in your planning problem. Let's look at an example with more than one feature. Here we have two species found in Borneo: the orangutan (*Pongo pygmaeus*) and a rare epiphytic plant (*Thrixspermum erythrolomum).*

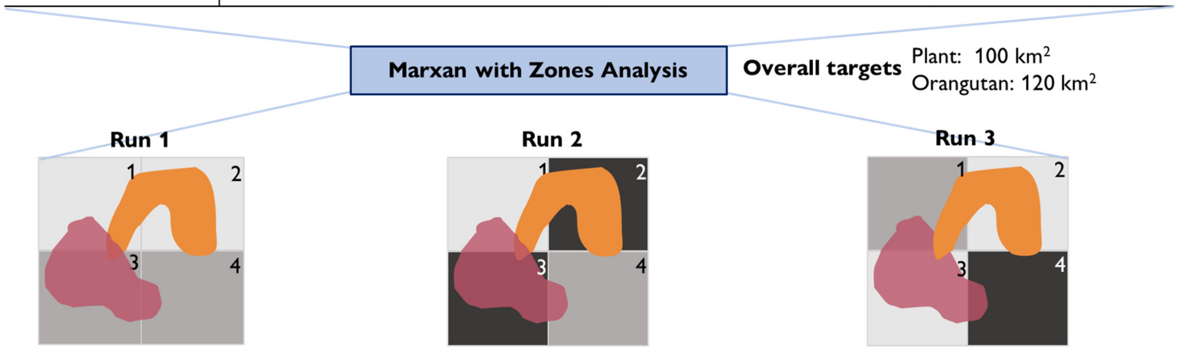


Planning unit grid

These species share the landscapes with many human activities and our zoning framework allows for three zones: 1) Conservation, 2) Sustainable Logging and 3) Agriculture. Each zone contributes differently to the species targets. For example, zones that allow for sustainable logging contribute less for meeting the plant's targets (30%) than for orangutans (60%) - as orangutans can still manage in modified landscapes with less structural complexity while the epiphytic plant s will have a more difficult time. Both will thrive in a fully protected area, so the contribution of the Conservation Zone is 100% for both.

Zones, features, relative feature contributions (based on the zone they are located in) & contribution (km²)

Zone description		Zone 1 – Full conservation		Zone 2 – Sustainable logging		Zone 3 – Agriculture	
Feature		Plant	Orangutan	Plant	Orangutan	Plant	Orangutan
Relative zone contribution		*100%*	*100%*	*30%*	*60%*	*10%*	*0%*
Planning unit ID	1	20	20	6	12	0.2	0
	2	0	105	0	63	0	0
	3	100	5	30	3	10	0
	4	10	2	3	1.2	0.1	0



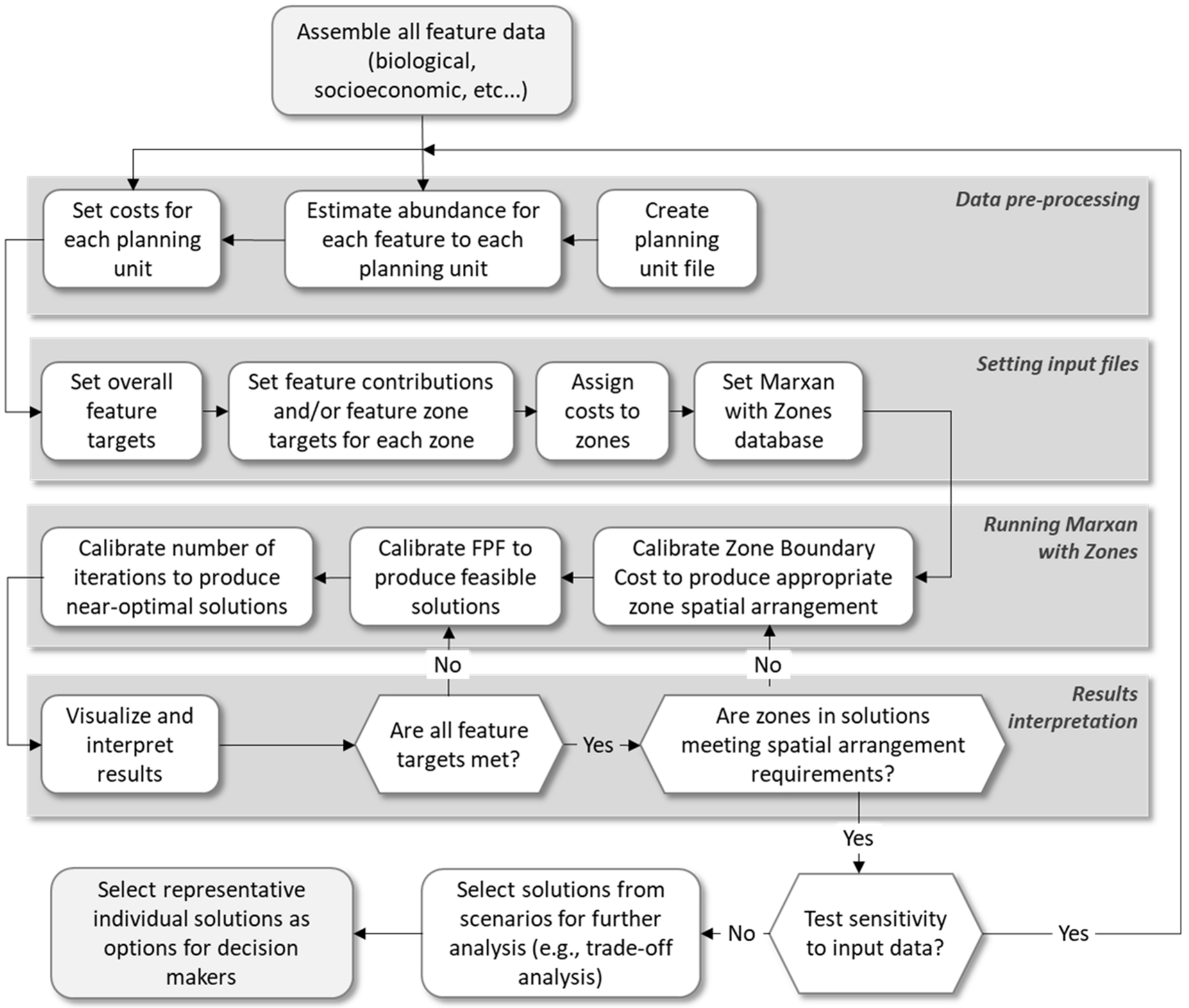
Run 1	Run 2	Run 3
Plant target met: ✗	Plant target met: ✗	Plant target met: ✓
Orangutan target met: ✓	Orangutan target met: ✗	Orangutan target met: ✓
Plant = 20 + 0 + 30 + 3 = 53	Plant = 20 + 0 + 10 + 3 = 33	Plant = 6 + 0 + 100 + 0 = 106
Orangutan =20 + 105 + 3 + 1.2 = 129.2	Orangutan = 20 + 0 + 0 + 1.2 = 21.2	Orangutan = 12 + 105 + 5 + 0 = 122

3.3 Key Steps of a Marxan with Zones Analysis

Sections 3.1 and 3.2 provided guidance for defining a zoning problem. This section presents four main steps or components of a Marxan with Zones analysis:

1. Pre-processing of data (see Section 4)
2. Setting up the input files and the scenario parameters (see Section 5)
3. Running Marxan with Zones (see Section 6)
4. Viewing and interpreting the outputs (see Section 7 and 8)

Each step of a Marxan with Zones analysis involves a series of tasks (Figure 3). Tasks related to running the software will generally take the shortest time. Tasks associated with pre-processing data, developing input files, and interpreting results will often be much more time-consuming.



Assemble all feature data (biological, socioeconomic, etc...)
Data pre-processing
Set costs for each planning unit
Estimate abundance for each feature to each planning unit
Create planning unit file
Setting input files
Set overall feature targets
Set feature contributions and/or feature zone targets for each zone
Assign costs to zones
Set Marxan with Zones database
Running Marxan with Zones
Calibrate number of iterations to produce near-optimal solutions
Calibrate FPF to produce feasible solutions
Calibrate Zone Boundary Cost to produce appropriate zone spatial arrangement
No
No
Results interpretation
Visualize and interpret results
Are all feature targets met?
Yes
Are zones in solutions meeting spatial arrangement requirements?
Yes
Select representative individual solutions as options for decision makers
Select solutions from scenarios for further analysis (e.g., trade-off analysis)
No
Test sensitivity to input data?
Yes

Figure 3. *Example of a Marxan with Zones analysis framework (adapted from MGPH)*

It is important to note that the successful use of Marxan with Zones will never involve a once-off, sequential application of these four steps. Instead, in any given project, these steps should be repeated numerous times, where the results of each run are used to refine the details of following runs. Notably, the calibration of key parameters (zone boundary cost, feature penalty factor or FPF, number of iterations, number of repetitions) will involve several Marxan with Zones runs to determine appropriate parameter values. For each run, the information in output tables need to be examined to determine whether targets have been met, and if alterations to targets or other parameters need to be made. Visual inspections of results will also identify if solutions are meeting desired spatial arrangements and whether the zone boundary cost still requires calibration.

Given the need for multiple scenarios, it is important to be well organised and to have an efficient file management protocol (see suggestions on file management in Section 5.2). Once the input files have been set up, it should be quite easy to modify the scenario, re-run Marxan with Zones and investigate the results.

A more detailed treatment of each of the main steps for running a Marxan with Zones analysis follows in subsequent sections.

4 Pre-Processing of Data

4.1 Overview of Data Preparation

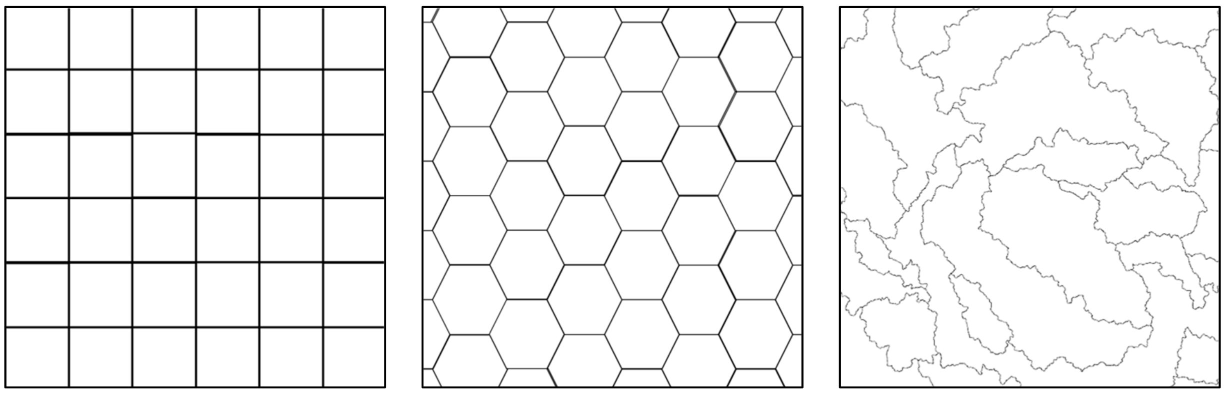
Data compilation, management and preparation of features and cost data are typically the most time-consuming aspects of any Marxan with Zones analysis. This section focuses on data preparation. It summarizes two essential steps for preparing spatial data for incorporation into Marxan with Zones:

1. dividing the planning region into planning units, and
2. determining the distribution of features and costs across planning units.

These steps are used to convert spatial information into a format that can subsequently be used to generate the Marxan with Zones input files. Both steps require some knowledge of a GIS software, such as ArcGIS or QGIS. We encourage users to acquire the necessary GIS skills through external resources or trainings, as this manual includes minimal instructions for using GIS software. Users are also encouraged to consult the Marxan Good Practice Handbook (MGPH) for guidance on assessing, managing, and preparing datasets, as this guidance is also valid for Marxan with Zones.

4.2 Dividing the Planning Region into Planning Units

An essential pre-processing step for any Marxan with Zones exercise is to divide the planning region into a set of planning units. In their simplest form, planning units may be defined by overlaying the planning region with a grid of squares (Figure 4a), a lattice of hexagons (Figure 4b), or irregularly shaped planning units (Figure 4c).



(a) Square PUs (b) Hexagonal PUs (c) Irregular PUs

Figure 4. *Three types of planning units that could be used in Marxan with Zones*

A great deal of care should be taken when deciding upon the size and shape of planning units, as it will influence the results of your Marxan with Zones analyses. The MGPH provides guidance for determining the appropriate planning unit size and shape. In general, planning unit size and shape is informed by the scale of the planning region (i.e., global, regional, national, or local), the resolution of datasets being used, the objectives of the planning exercise, and the intended use of the outputs (e.g., general area prioritisation or specific plans for implementation).

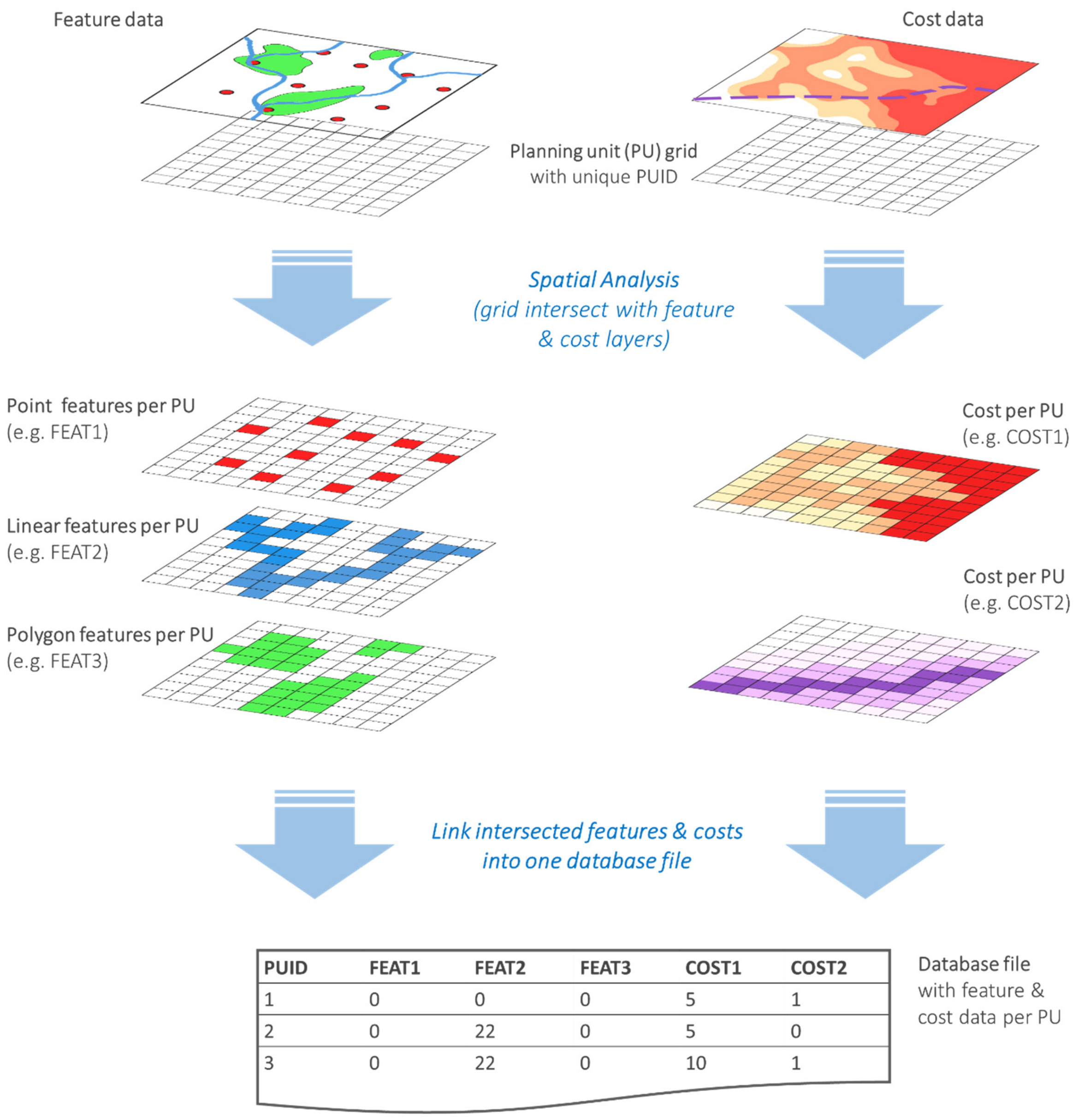
In addition to planning unit size and shape, the number of planning units may also need to be considered in certain cases. There is a limit on the number of planning units that Marxan with Zones can handle. This is not, however, a fixed number as it depends on the number of features you wish to plan for and to some extent on the power of your computer. The MGPH discusses this topic in more detail.

4.3 Determining the Distribution of Features & Costs

A second essential step of data processing is to determine the distribution of the planning features and costs across the planning units. This requires completing the following tasks in a GIS software:

1. Data layers representing planning features must each be converted into a single unified dataset associated with the planning unit data layer. This means assembling all the requisite feature data, and then calculating how much of each feature is located within each planning unit.
2. Data layers representing different costs will also need to be assembled and associated with each planning unit. Contrary to Marxan, Marxan with Zones accepts more than one cost value per planning unit. There should be a non-zero cost in every planning unit to avoid selection errors.
3. Individual layers containing tallied information on the amount of each feature and cost in each planning unit can now be linked into a single database file. This database file can subsequently be used in a Marxan supporting software (see Section 2.3) to generate the standard set of input files required to run Marxan with Zones.

Figure 5 illustrates each task using a few data layers on features and costs. It also provides an example of a subset of a resulting database file. Note that the first column of the database table lists the planning unit identifier, while the following columns each correspond to a feature or cost. In sum, the database contains information on the amount of each feature and cost in each planning unit.


Feature data
Cost data
Planning unit (PU) grid with unique PUID
Spatial Analysis (grid intersect with feature & cost layers)
Point features per PU (e.g. FEAT1)
Linear features per PU (e.g. FEAT2)
Polygon features per PU (e.g. FEAT3)
Cost per PU (e.g. COST1)
Cost per PU (e.g. COST2)
Link intersected features & costs into one database file
Database file with feature & cost data per PU

PUID	FEAT1	FEAT2	FEAT3	COST1	COST2
1	0	0	0	5	1
2	0	22	0	5	0
3	0	22	0	10	1

Figure 5. *GIS workflow for determining the distribution of features and costs*

5 Input Files, Parameters & Variables

5.1 Overview of Input Files

The section summarizes the input files required to run Marxan with Zones, as well as the optional files that can be used to facilitate additional functionality (Table 7). Marxan with Zones builds on the standard set of input files for Marxan: input.dat, feat.dat, pu.dat, puvfeat.dat, and bound.dat files. All the standard Marxan files are required to run Marxan with Zones, apart from the bound.dat file which is optional. Marxan with Zones also needs three new input files to function: zone.dat, cost.dat, and zonecost.dat files. A few optional files (e.g., zoneboundcost.dat, zonetarget.dat, and zonecontrib.dat files) can also be included in Marxan with Zones for specific functions.

An example of each input file is contained in the download package for Marxan with Zones (sample input files are in a folder called 'Sample_Input_Files'; automatically downloaded when you download Marxan with Zones). Note that these input files are for demonstration purposes only.

Table 7. *Input files for Marxan with Zones*

File	Default Name	Standard Marxan File or New File	Required in Marxan w/ Zones
Input Parameter File	input.dat	Standard Marxan	Yes
Feature File	feat.dat	Standard Marxan	Yes
Planning Unit File	pu.dat	Standard Marxan	Yes
Planning Unit versus Feature File	puvfeat.dat	Standard Marxan	Yes
Zones File	zones.dat	New	Yes
Costs File	costs.dat	New	Yes
Zone Cost File	zonecost.dat	New	Yes
Boundary Length File	bound.dat	Standard Marxan	No
Zone Boundary Cost	zoneboundcost.dat	New	No
Planning Unit Zone	puzone.dat	New	No
Planning Unit Lock	pulock.dat	New	No
Zone Target	zonetarget.dat	New	No
Zone Target 2	zonetarget2.dat	New	No
Zone Contribution	zonecontrib.dat	New	No
Zone Contribution 2	zonecontrib.dat	New	No

The required and optional input files are summarized below.

- The **Input Parameter File** is used to set values for all the main parameters that control the way in which Marxan with Zones works. It also tells the software where to find the input files and where to place the output files.
- The **Feature File** (also referred to as 'Species Feature File' or spec.dat) contains information about each of the features being considered, including their name, targets, and the penalty that should be applied if targets are not met.
- The **Planning Unit File** lists all the planning units and assigns one or more costs to each planning unit.
- The **Planning Unit versus Feature File** (also known as 'Planning Unit versus Species Feature File', puvsp.dat, or puvspr.dat) contains information on the distribution of features in each of the planning units.
- The **Boundary Length File** contains information on the boundary length or 'boundary cost' between pairs of planning units.
- The **Zones File** lists the names and numeric identifiers of all possible zones.
- The **Costs File** is used to assign each cost name in the Planning Unit File with a numeric identifier.
- The **Zone Cost File** is used to assign a weighting factor for each cost in each zone.
- The **Zone Boundary Cost File** can be used if you want boundaries between different zones to have different costs. This is useful when seeking a particular spatial configuration between zones.
- The **Planning Unit Zone** is used to restrict certain planning units to 2 or more zones.
- The **Planning Unit Lock** is used to restrict certain planning units to a single zone.
- The **Zone Target** or **Zone Target 2** is used to allow zone-based targets to be set in Marxan with Zones.
- The **Zone Contribution** or **Zone Contribution 2** work in tandem with overall targets in the Feature File to specify differential contribution rates to overall target.

These input files are described in greater detail in Sections 5.3 and 5.4.

5.2 Input Files Management & Format

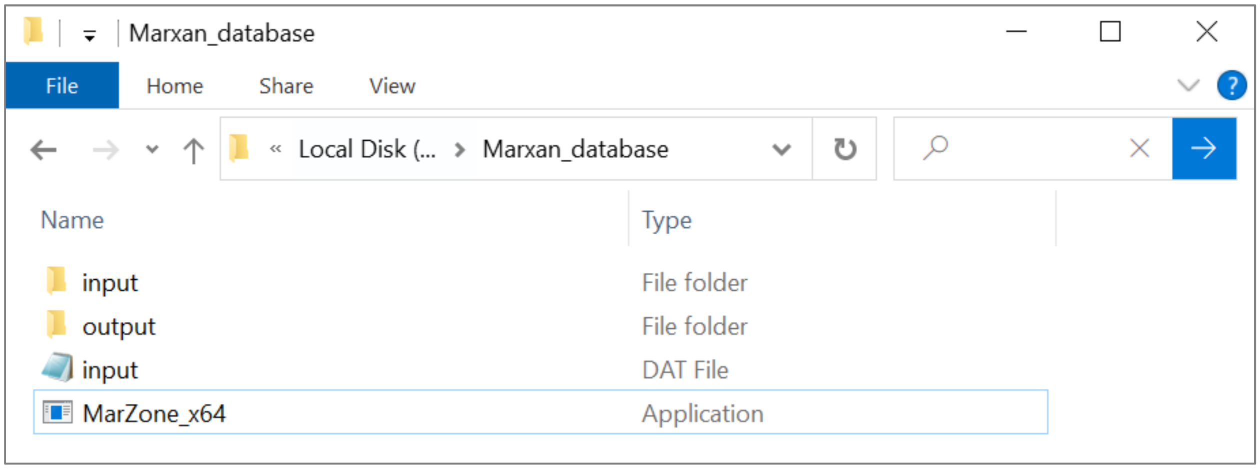
This section provides guidance for creating and managing input files, as well as for setting up a Marxan with Zones database folder. There are different options for developing, editing, and viewing the input files for Marxan with Zones. The choice of method will depend on your skills, available software and personal preference. Box 6 provides some recommendations for developing of input files.

Box 6. The Development of Input Files

We recommend using one of the GIS software and associated Marxan plugins (see Section 2.3) to develop the standard set of Marxan input files (i.e., input.dat, feat.dat, pu.dat, puvfeat.dat, and bound.dat). These files can then be read and modified in a spreadsheet or text editor program, such as Windows Notepad, Microsoft Excel, and Open Office. These spreadsheet or text editor programs can also be used to develop, edit, and view additional input files associated with Marxan with Zones.

Like standard Marxan, all input files in Marxan with Zones use the .dat file extension (i.e., input files must be saved as a .dat file format). To generate a .dat file, simply add the suffix '.dat' after the file name when saving the file. Files can be comma, space, or tab delineated.

An example of a Marxan with Zones database folder is shown in Figure 6. All input files must be held in a folder called 'input' located within the Marxan with Zones database. The only exception is the input.dat file, which must be held in the same location as the MarZone_x64.exe (i.e., the Marxan with Zones application).



Name	Type
input	File folder
output	File folder
input	DAT File
MarZone_x64	Application

Figure 6. *Recommended set up for a Marxan with Zones database folder*

A Marxan with Zones database must contain:

- an 'input' folder, which contains all the input files for running Marxan with Zones (apart from the input.dat file);
- an 'output' folder, which will contain the generated output files after Marxan with Zones has completed its run;
- the 'MarZone_x64.exe' or the Marxan with Zones application;
- the 'input.dat' file, which defines the main parameters for running the software.

A Marxan with Zones database can also contain a 'pulayer' folder with the planning unit shapefile (i.e., the spatial layer file containing planning units). This folder is not required to run Marxan with Zones, but it is useful to visualize outputs externally in GIS software.

The next sections describe the structure of the required input files (Section 5.3) and optional input files (Section 5.4) for Marxan with Zones. All input files must be in the .dat file format. Except for the Input Parameter File, all files consist of a header line and a main body. The header line is a list of specific names describing what is contained in each column of the main body. Each file contains a set of required and optional header names for Marxan with Zones to run. The headers in each file must be written in all lower-case letters with no punctuation, spaces, or numerals (unless otherwise noted). Variables in a single line can be separated by a variety of characters, including spaces, tabs, or commas.

5.3 Required Input Files

This section describes the function, format, and associated variables of the input files required to run Marxan with Zones:

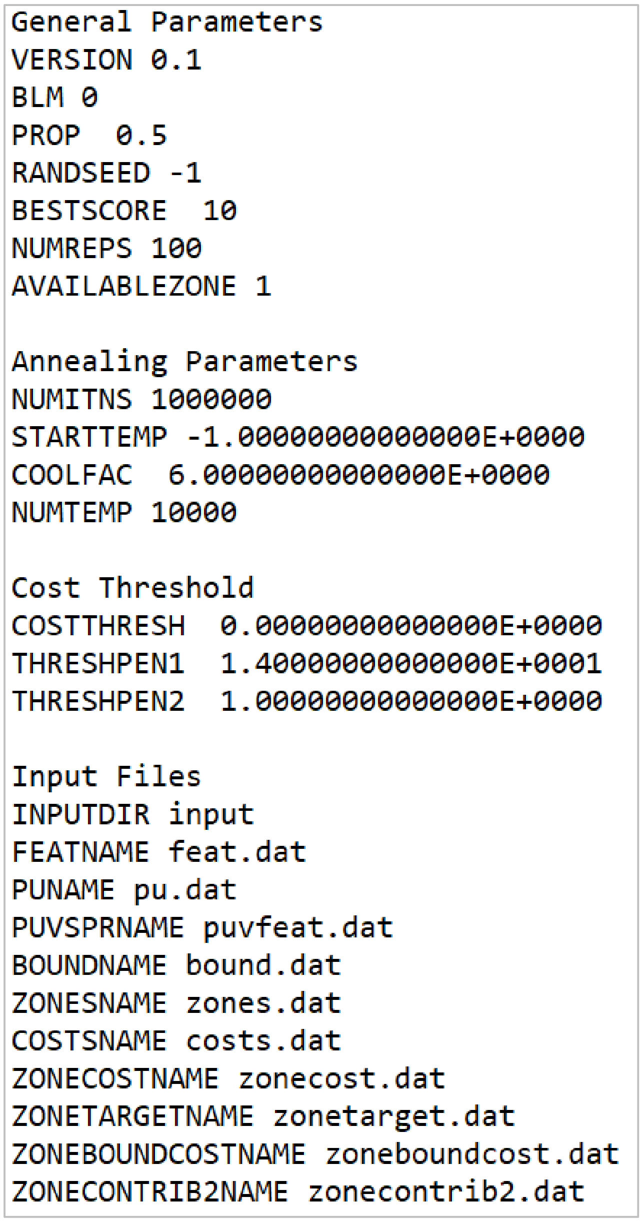
- Input Parameter File (input.dat)
- Feature File (feat.dat)
- Planning Unit File (pu.dat)
- Planning Unit versus Feature File (puvfeat.dat)
- Zones File (zones.dat)
- Costs File (costs.dat)
- Zone Cost File (zonecost.dat)

⚠ All the required files must be included to run Marxan with Zones properly. Missing any of the required files will cause Marxan with Zones to halt and generate an error message.

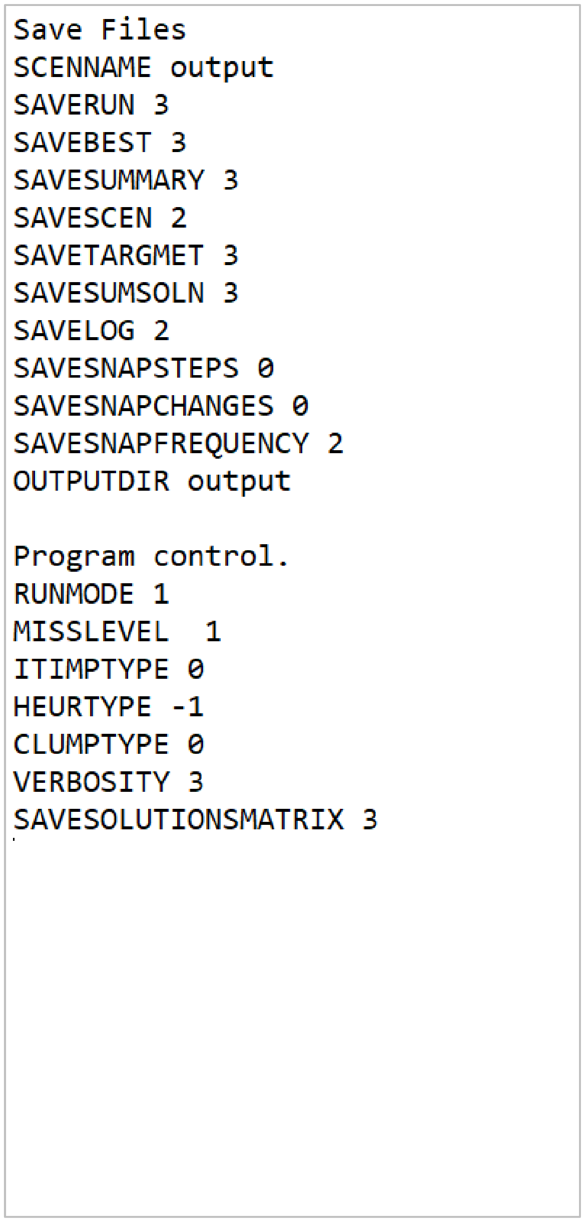
5.3.1 Input Parameter File (input.dat)

The Input Parameter File (input.dat) contains the main parameters that control how Marxan with Zones finds solutions (i.e. which algorithm(s) are used and what parameters contribute to the objective function). In addition, it tells Marxan with Zones where to find the required input files and whether it needs to consider any optional input files. It also indicates which output files it needs to generate and where it should save them.

An example of the input.dat file is provided in Figure 7. This example does not include all possible variables that may be held in the input.dat file.



```
General Parameters
VERSION 0.1
BLM 0
PROP  0.5
RANDSEED -1
BESTSCORE  10
NUMREPS 100
AVAILABLEZONE 1

Annealing Parameters
NUMITNS 1000000
STARTTEMP -1.00000000000000E+0000
COOLFAC  6.00000000000000E+0000
NUMTEMP 10000

Cost Threshold
COSTTHRESH  0.00000000000000E+0000
THRESHPEN1  1.40000000000000E+0001
THRESHPEN2  1.00000000000000E+0000

Input Files
INPUTDIR input
FEATNAME feat.dat
PUNAME pu.dat
PUVSPRNAME puvfeat.dat
BOUNDNAME bound.dat
ZONESNAME zones.dat
COSTSNAME costs.dat
ZONECOSTNAME zonecost.dat
ZONETARGETNAME zonetarget.dat
ZONEBOUNDCOSTNAME zoneboundcost.dat
ZONECONTRIB2NAME zonecontrib2.dat
```



```
Save Files
SCENNAME output
SAVERUN 3
SAVEBEST 3
SAVESUMMARY 3
SAVESCEN 2
SAVETARGMET 3
SAVESUMSOLN 3
SAVELOG 2
SAVESNAPSTEPS 0
SAVESNAPCHANGES 0
SAVESNAPFREQUENCY 2
OUTPUTDIR output

Program control.
RUNMODE 1
MISSLEVEL  1
ITIMPTYPE 0
HEURTYPE -1
CLUMPTYPE 0
VERBOSITY 3
SAVESOLUTIONSMATRIX 3
```

Figure 7. *An example of the Input Parameter File (input.dat). Left image shows the first half of the file, and right image the second half of the file*

A complete list of variables and default values for the input.dat file is provided in Table 8. As depicted in the table (also in Figure 7), the variables are organized into six groups: General Parameters, Annealing Parameters, Cost Threshold, Input Files, Save Files, and Program Control. Each variable or group of variables are described in greater detail in the following subsections.

Table 8. *Variable names and default values for the Input Parameter File (input.dat)*

Category	Variable Name	Default Value	Description
General Parameter	BLM	0	Boundary Length Modifier
	PROP	0.5	Proportion of planning units to start in run
	RANDSEED	-1	Random seed number
	NUMREPS	100	Number of separate runs with same starting condition
	AVAILABLEZONE	1	'Available' zone
Annealing Parameters	NUMITNS	0	Number of iterations for annealing for each run
	START TEMP	1	Starting temperature for annealing
	COOLFAC	0	Cooling factor for annealing
	NUMTEMP	1	Number of temperature decreases for annealing
Cost Threshold	COSTTHRESH	0	Cost threshold
	THRESHPEN1	0	Size of cost threshold penalty
	THRESHPEN2	0	Shape of cost threshold penalty
Input Files	INPUTDIR	input	Input directory where data files are stored
	FEATNAME	feat.dat	Name of Feature File
	PUNAME	pu.dat	Name of Planning Unit File
	PUVSPRNAME	puvfeat.dat	Name of Planning Unit versus Feature File
	ZONESNAME	zones.dat	Name of Zones File
	COSTNAME	costs.dat	Name of Cost File
	BOUNDNAME	bound.dat	Name of Boundary Length File
	ZONEBOUNDCOSTNAME	zoneboundcost .dat	Name of Zone Boundary Cost File
	PUZONENAME	puzone.dat	Name of Planning Unit Zone File
	PULOCKNAME	pulock.dat	Name of Planning Unit Lock File
	ZONETARGETNAME	zonetarget.dat	Name of Zone Target File
	ZONETARGET2NAME	zonetarget2.dat	Name of Zone Target 2 File
	ZONECONTRIBNAME	zonecontrib.dat	Name of Zone Contribution File
	ZONECONTRIB2NAME	zonecontrib2.dat	Name of Zone Contribution 2 File
Save Files	SCENNAME	output	Name of folder where output files will be saved
	SAVERUN	3	Save each run
	SAVEBEST	3	Save the best run

Save Files (cont.)	SAVESUMMARY	3	Save summary information
	SAVESCEN	2	Save scenario information
	SAVETARGMET	3	Save targets met information
	SAVESUMSOLN	3	Save summed solution information
	SAVESOLUTIONSMATRIX	3	Save all runs in a single matrix
	SOLUTIONSMATRIXHEADERS	1	Include header rows in solutions matrix
	SAVEPENALTY	3	Save computed feature penalties
	SAVELOG	2	Save log files
	SAVEANNEALINGTRACE	3	Report detail for simulated annealing
	ANNEALINGTRACEROWS	1000	Number of iterations to report detail for
	SAVEITIMPTRACE	3	Report detail for iterative improvement
	ITIMPTRACEROWS	1000	Number of iterations to report detail for
	SAVEZONECONNECTIVITYSUM	3	Save zone connectivity sum for runs
	OUTPUTDIR	output	Output directory to store output files
Program Control	RUNMODE	1	Run option
	MISSLEVEL	1	Species missing proportion
	ITIMPTYPE	0	Iterative improvement
	VERBOSITY	2	Screen option

There are two ways to develop the input.dat file for a given project:

- create a new file using one of the supporting software for Marxan and add the additional fields needed in Marxan with Zones; or
- copy and modify the input.dat file that comes bundled with the Marxan with Zones download package.

In either case, a text editor program (e.g., Windows Notepad or equivalent) can be used to make changes to the input.dat file.

In the input.dat file, each variable must be entered on a separate line. To set the value of a variable, start a line with the variable name and follow it with the value for that variable. The variable name and value must be separated by a single space. The variable names must be in upper case characters with no spaces. The variables can occur in any order in the file, but an error will result if any variable is defined twice. Any line which does not start with one of the valid variable names will be ignored, so it is possible to include comments or notes between variables in this file.

Most of the variables have default values which will be used if they are not defined. The exceptions to this are the variables that tell Marxan where to find the necessary input files, where to save the output files, and the variable 'RUNMODE'. There are no universally best values for the parameters contained within the input.dat file. Although similar values may work well for different applications, you will need to determine the most appropriate parameter values for each project. This is best done in an iterative fashion in which the results are investigated, the parameters changed, the program run again, and the new results compared with the old ones.

The following subsections give a description of each variable in the input.dat file, along with guidance for setting appropriate parameter values. Guidance for calibrating the number of repeat runs (NUMREPS) and the number of iterations (NUMITNS) is also provided.

5.3.1.1 Boundary Length Modifier

Variable Name:	BLM
Required:	No
Description:	In standard Marxan, the Boundary Length Modifier (BLM) is a parameter used to control the spatial compactness of solutions (compact versus fragmented solutions). In Marxan with Zones, however, the spatial relationships between zones are controlled by the Zone Boundary Cost file (zoneboundcost.dat).
Getting Started:	The BLM in Marxan with Zones should be set to '0' if no considerations about the spatial arrangement between zones need to be made. Otherwise, the BLM should be set to '1' to "activate" the Zone Boundary Cost file. Unlike Marxan, no calibration of the BLM is needed in Marxan with Zones. Rather, the Zone Boundary Cost file should be calibrated if it is used (see Section 5.4.2).

5.3.1.2 Starting Prop

Variable Name:	PROP
Required:	No
Description:	When Marxan with Zones starts a run, it must generate an initial solution. The 'PROP' variable defines the proportion of planning units to start in the initial solution at the start of each run.
Getting Started:	This variable must be a number between '0' and '1'. If '0' is chosen, then no planning units will be included in the initial reserve. A value of '1'

	means all planning units will be included. A value of ‘0.5’ means 50% of planning units will be randomly included. In practice, the setting has no effect on the operation of simulated annealing, provided a sufficient number of iterations is used.

5.3.1.3 Random Seed

Variable Name:	RANDSEED
Required:	No
Description:	This variable controls whether the same ‘random’ selection of planning units is included in the initial solution at the start of each run. Using a constant positive integer for this variable will make Marxan with Zones use the same random seed each time it is run.
Getting Started:	The random seed number must be an integer. If the value is negative, the program will randomly select a seed. A value of ‘-1’ is recommended, except for debugging purposes. Along with debugging purposes, a positive value could be used if you want more than one application of the program to be identical.

5.3.1.4 Repeat Runs

Variable Name:	NUMREPS
Required:	Yes
Description:	This variable indicates the number of separate runs you want Marxan with Zones to perform; that is, it defines the number of solutions you want the software to generate. Each new run is independent of the previous one, but they will all use the same parameter and variable values.
Getting Started:	When running a new scenario for the first time, it is always advisable to begin with a very small number of runs (e.g., 10). This will allow you to check that the program is performing as desired (i.e. the solutions are meeting the required targets) without having to wait a long time. In

	order to get an idea of selection frequency (the frequency with which planning units are assigned to each zone across multiple runs), you will generally need to perform many runs. One hundred runs is probably a minimal value to start with and is an intuitive value from which to calculate selection frequency. Adding more runs will sample more of the solution space but will of course increase the processing time. The final number you decide on must be a balance between the time taken and the information gained. Hence, the ideal number of repeat runs will vary between projects.

5.3.1.5 Simulated Annealing Parameters

Variables:	NUMITNS STARTTEMP COOLFAC NUMTEMP
Required:	Yes (when using Simulated Annealing)
Description:	These four variables control the way the Simulated Annealing algorithm proceeds. They will come into play when Simulated Annealing is chosen in the 'RUNMODE' (see Section 5.3.1.10). The NUMITNS sets the number of iterations for annealing for each run (i.e., the number of times Marxan with Zones tries to generate a solution for each run). The STARTTEMP sets the starting temperature for annealing. The COOLFAC determined how quickly the system cools. The NUMTEMP defines the number of temperature decreases for annealing.
Getting Stared:	In practice, you will rarely need to adjust these variables, apart from the NUMITNS. For the STARTTEMP, a value of '1' indicates that adaptive annealing will be used, and the program will automatically select a starting temperature. This is recommended for use in Marxan with Zones. If the STARTTEMP is set to -1, then the COOLFAC is not necessary to run Marxan with Zones. This is recommended for use in Marxan with Zones.

	The NUMTEMP must be less than or equal to the number of iterations. A value of '10,000' is ideal and is recommended in Marxan with Zones. Setting it lower can make the regime too coarse; setting it too high will lead to round-off error problems with temperature. The NUMITNS sets the number of times Marxan with Zones tried to generate a solution for each run. This number of iterations has a substantial bearing on how long each run takes. In general, the number of iterations determines how close Marxan gets to the optimal solution (or at least a very good solution). The more iterations set, the longer the program will run, and the more likely Marxan with Zones will generate a better solution (i.e. lower objective function value). ⚠ The number of iterations needs to be scaled according to the number of zones used to achieve an efficient operation of Marxan with Zones. For example, if for a standard Marxan analysis with two zones you needed 1,000,000 iterations to achieve efficient solutions, for a Marxan with Zones analysis with six zones you will need at least 3,000,000 iterations as a starting point. To calibrate this parameter, increase the number of iterations until there is no substantial improvement in score or cost, then choose an acceptable trade-off between solution efficiency (average score of solutions) and execution time (number of iteration or run time).

5.3.1.6 Available Zone

Variable Name:	AVAILABLEZONE
Required:	Yes
Description:	This variable indicates which zone number from the Zones Input File (zones.dat) is your available zone. In most cases, the available zone is treated as a zone without specific objectives or management actions.

5.3.1.7 Cost Threshold

Variable Name:	COSTTHRESH THRESHPEN1

	THRESHPEN2
Required:	No
Description:	In standard Marxan, the cost threshold (COSTTHRESH) can be included to tell Marxan to find a solution below a total cost. In Marxan with Zones, it can be included to cap the zoning configuration to a set cost. As discussed in Section 1.1, Marxan with Zones is designed to solve a 'minimum set' problem, where the goal is to meet feature targets for the least cost. Another class of problem is known as the 'maximum coverage' problem, where the goal is to maximize outcomes for a set cost (e.g., achieve the best conservation outcomes for a given fixed budget). Although including a cost threshold does not make Marxan with Zones solve the strict 'maximum coverage' problem, it is comparable and can be used in cases where you have feature targets you hope to meet and cannot exceed a predetermined budget. For more information see Appendix B, section 1.6.
Getting Started:	These variables could be used if you want Marxan with Zones to cap the zoning configuration to a set cost. If the cost threshold variables are not used, this section does not need to be included in the input.dat file for Marxan with Zones. Alternatively, the cost threshold variable can be set to '0' in the input.dat file to disable it.

5.3.1.8 Input Files

Variable Name:	INPUTDIR FEATNAME PUNAME PUVSPRNAME ZONESNAME COSTSNAME ZONECOSTNAME BOUNDNAME ZONEBOUNDCOSTNAME PUZONENAME PULOCKNAME

	ZONETARGETNAME ZONETARGET2NAME ZONECONTRIBNAME ZONECONTRIB2NAME
Required:	Yes
Description:	The 'INPUTDIR' variable is used to tell Marxan with Zones the name of the folder containing the input files (this folder is commonly named 'input'). The remaining variables each correspond to a particular input file. They are used to tell Marxan with Zones the name of the input files contained in the 'input' folder. While all possible input files are listed in this section, the user should only list the input files they want Marxan with Zones to consider. Marxan with Zones will only consider input files with file names listed in the input.dat file.
Description and Getting Started:	The input section only needs to include the required Marxan with Zones files plus the optional files that you choose to utilize. The protocols for file naming and storage have previously been discussed (see introduction of Section 5.3.1). ⚠ Before running Marxan with Zones, it is important to verify that the files are in the correct folder with file names that match those in the input.dat file. The software will not run properly if it cannot find the files due to a naming or storage error. Likewise, the INPUTDIR must be done correctly or Marxan with Zones will not run. To avoid this error, we recommend following the file management protocol outlined in Section 5.2 and using the default folder name ('input') for the folder containing the input files.

5.3.1.9 Save Files

Variable Name:	SCENNAME SAVERUN SAVEBEST SAVESUMMARY SAVESCEN

	SAVETARGETMET SAVESUMSOLN SAVESOLUTIONSMATRIX SOLUTIONSMATRIXHEADERS SAVEPENALTY SAVELOG SAVEANNEALINGTRACE ANNEALINGTRACEROWS SAVEITIMPTRACE ITIMPTRACEROWS SAVEZONECONNECTIVITYSUM OUTPUTDIR
Required:	Yes
Description:	The SCENNAME is the name you wish Marxan with Zones to append to all output files it saves (e.g., setting the SCENNAME to 'output' would save the summed solution output with the name 'output_ssoln.dat'). The 'OUTPUTDIR' is used to tell Marxan with Zones the name of the folder where it should save the output files. The naming and location protocols for this folder are discussed in Section 5.2. The remaining variables are used to tell Marxan with Zones which output files to generate and how outputs should be saved (e.g., .dat, .txt, or .csv format). A description of each output file is provided in the output file section of this manual (Section 7.4). The following codes are used to select the desired format of output files: • Use code '1' to save a file as .dat • Use code '2' to save a file as .txt • Use code '3' to save a file as .csv • Use code '0' if you do <u>not</u> want a file to be generated
Getting Started:	While the default name for the SCENNAME is 'output', it is useful to select a name that will help you identify the scenario that generated the outputs. For instance, you could set the SCENNAME to 'scenario1' for your first scenario, so that all the output files can

	easily be linked to this scenario (e.g., ‘scenario1_ssoln.dat would be the name given to the summed solution output file). Below are recommended format codes for output files (also listed in Table 8): • SAVERUN 3 • SAVEBEST 3 • SAVESUMMARY 3 • SAVESCEN 2 • SAVETARGETMET 3 • SAVESUMSOLN 3 • SAVELOG 2 • SAVESNAPFREQUENCY 2 ⚠ Like the input folder, it is critical to ensure that the output folder in OUTPUTDIR is set up correctly or Marxan with Zones may not run.

5.3.1.10 Run Options

Variable Name:	RUNMODE
Required:	Yes
Description:	There are two basic algorithms that can be used to formulate a solution in Marxan with Zones: simulated annealing and iterative improvement. These algorithms can be used alone or in combination with each other. The run mode selected in the input.dat file determines which algorithm, or combination of algorithms, is used to formulate a solution in Marxan with Zones. There are four different run options, which can be set using the following codes: -1 = Use no methods 1 = Apply annealing followed by the iterative improvement algorithm 4 = Use only the iterative improvement 6 = Use only annealing

Getting Started:	For Marxan with Zones, the most useful mode option is simulated annealing followed only by iterative improvement (mode 1). This is because Simulated Annealing searches the solution space effectively, and the Iterative Improvement then ensures that the solution represents the best option in the immediate area of the decision space (known as a 'local minimum'). For most applications of Marxan with Zones, this will be the best run option and will rarely need to be changed. For more information see Appendix B, Section B-2.

5.3.1.11 *Missing Proportion*

Variable Name:	MISSLEVEL
Required:	No
Description:	This is the proportion of the target, which a feature must reach in order for it not to be counted as missing. A value of '1' means that 100% of the target for a feature must be included in the solution, or it will be considered an unmet target. There are situations where Marxan can get extremely close to the target (e.g. 99% of the desired level) without actually meeting the target. You can specify a level for which you are pragmatically satisfied that the amount of representation is close enough to the target to report it as met.
Getting Started:	This value should always be high, i.e. greater than or equal to '0.95'. If you are setting it lower than '0.95', you should probably think about changing your targets. As a guide, it is often useful to run Marxan with Zones with the 'MISSLEVEL' set at '1' and then re-run with it set at a slightly lower value and see if there is much of a difference in system cost. Setting this variable does not change the way the Marxan with Zones algorithm works, it merely changes the way target achievement is reported in screen and file output.

5.3.1.12 *Iterative Improvement*

Variable Name:	ITIMPTYPE
Required:	No
Description:	Iterative improvement will only be used when a run mode using the iterative improvement algorithm is selected. If Iterative Improvement is being used to help find solutions, this variable defines what type of Iterative Improvement will be applied. There are five possible iterative improvement types. The type is set using the following codes: -1 = Do not use iterative improvement 0 = Normal Iterative Improvement 1 = Two Step Iterative Improvement 2 = Swap Iterative Improvement 3 = Normal Iterative Improvement followed by Two Step Iterative Improvement
Getting Started	Using the default value of '0' is recommended in Marxan with Zones.

5.3.1.13 *Screen Output*

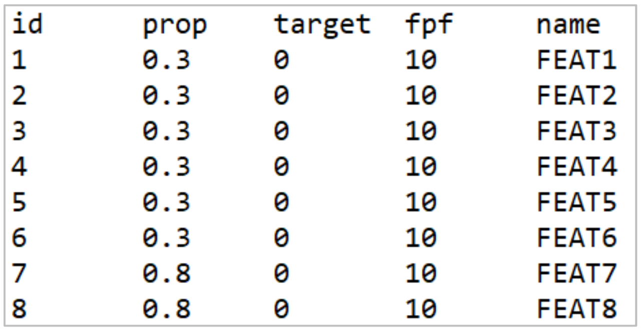
Variable Name:	VERBOSITY
Required:	Yes
Description:	This value indicates how much information Marxan with Zones prints to the screen (the verbosity) while it is running. There are four levels of information that can be selected using the following numbered codes: 0 = Silent Running 1 = Results Only 2 = General Progress 3 = Detailed Progress A value of '0' will display the lowest level of information and a level of '3' will show the highest amount of information. A more detailed

	description of each value is provided in the Screen Output section (Section 7.3).
Getting Started	The default for this variable is 'General Progress' and in most cases this will be the best choice. Printing results to the screen does not increase Marxan with Zones' run time substantially unless 'Detailed Progress' is used. It is generally worthwhile to use at least 'Results Only' so that you have some idea of how many runs have been completed. 'Detailed Progress' is useful for seeing how the process of annealing works and can also help identify problems Marxan with Zones runs (e.g. if the numbers do not change and it is "stalled"). For this reason, some users always use this setting, to visually check that the program appears to be running well. Only apply 'Silent Running' if you are confident in Marxan with Zones execution and you are saving all necessary outputs.

5.3.2 Feature File (feat.dat)

The Feature File (feat.dat) is the same as in standard Marxan. It contains information about each of the features being considered, including the feature unique identifier, the feature name, and the feature target. The features may represent any biological, social, cultural, or economic feature of interest in the planning region with a spatial component. An example of the feat.dat file is shown in

Figure 8.



```
id      prop    target  fpf     name
1       0.3     0       10      FEAT1
2       0.3     0       10      FEAT2
3       0.3     0       10      FEAT3
4       0.3     0       10      FEAT4
5       0.3     0       10      FEAT5
6       0.3     0       10      FEAT6
7       0.8     0       10      FEAT7
8       0.8     0       10      FEAT8
```

Figure 8. *An example of the Feature File (feat.dat)*

This file is sometimes referred to as the Species Feature File with the file name 'spec.dat'. This alternative name reflects Marxan's historical roots in the natural sciences, where early applications of the decision support tool focused almost exclusively on the conservation of biological features, such as species and habitats. However, features in Marxan and Marxan with Zones may represent social and economic features, as well as conservation features. As such, this manual refers to this file by its more appropriate name, the Feature File. Marxan with Zones will work with either file name, so long as the file name matches the one entered in the input.dat file.

Table 9 contains a complete list of all possible variables for the feat.dat file. The file must contain an 'id' field, as well as a 'target' or 'prop' field.

Table 9. *Variable of the Feature File (feat.dat)*

Variable Name	Required	Description
id	Yes	The numeric identifier for this feature.
target	No, if prop is used	The target amount (in unit of puvfeat.dat file) of the feature to include across all zones.
prop	No, if target is used	An alternative to target; the proportion of the total amount of the feature which must be included in the zones.
targetocc	No	The number of occurrences of the feature required.
propocc	No	The percentage of occurrences of the feature required.
fpf	No	The feature penalty factor for that feature.
name	No	Indicates the name of that feature.

Feature targets can either be set as an amount in the 'target' field or as a proportion of the total amount in the 'prop' field. The 'targetocc' and 'propocc' fields can also be used for setting targets. The selection of appropriate feature targets will depend on the goals and objectives of the zoning project. Targets do not need to have uniform values for all features. Whatever the chosen targets, it is important that they are well justified, as they will have an enormous bearing on the character of potential zoning systems. The MGPH provides guidance for setting targets in standard Marxan, most of which is equally applicable to Marxan with Zones.

5.3.2.1 Feature ID

Variable Name:	id
Required:	Yes
Description:	The numeric identifier for each feature. The id must be a positive integer.

Getting Started:	Be careful not to duplicate id numbers as Marxan ones will ignore all but the last one.

5.3.2.2 Target (amount)

Variable Name:	target
Required:	Only if 'prop' is not used.
Description:	The variable 'target' can be used to set a total amount of each feature to be included across all zones (i.e., the overall target amount). For example, if the target is to capture at least 10 occurrences of Feature A, then a value of '10' should be set in the 'target' field for that feature. The set values represent constraints on potential solutions to solving the zoning problem. That is, for a solution to be feasible, it must include at least this amount of each feature across all zones.
Getting Started:	The 'target' field is used to set targeted amounts. Targets set in the variable can take any value from '0' to the total sum of that feature found in all planning units. ⚠ You must be careful not to set a higher target than can possibly be achieved given the occurrence of a feature in the planning units, as these targets will not be achievable. The target value must be expressed in the same units as the puvfeat.dat file. However, units from different features can vary (e.g., hectares of coverage for FEAT1, number of occurrences for FEAT2, and length for FEAT 3). ⚠ If an 'overall' target is specified through the 'target' field, then a Zone Target File (zonetarget.dat or zonetarget2.dat) or Zone Contribution File (zonecontrib.dat or zonecontrib2.dat) must be used. See Section 5.4.5 or Section 5.4.6 for more information.

5.3.2.3 Target (proportion)

Variable Name:	prop

Required:	Only if 'target' is not used.
Description:	The variable 'prop' is an alternative to 'target' field described above. It is used to target a set proportion of the total amount of a feature for inclusion in the zones. For instance, a value of '0.3' would indicate that 30% of that feature should be capture in zone(s). With this target, Marxan with Zones will work to find solutions that capture 30% of the total abundance of that feature based on data in the Planning Unit versus Feature File (puvfeat.dat). Total abundance is the sum of the amount found in all planning units, including those that may be restricted to one or more zone(s). Hence, the proportion is based on the total amount defined in the puvfeat.dat file.
Getting Started:	Feature targets set in the 'prop' field must be a value between '0' and '1', where '0' represents 0% of the total abundance and '1' represents 100% of the total abundance. ⚠ If an 'overall' target is specified through the 'prop' field, then a Zone Target File (zonetarget.dat or zonetarget2.dat) or Zone Contribution File (zonecontrib.dat or zonecontrib2.dat) must be used. See Section 5.4.5 or Section 5.4.6 for more information.

5.3.2.4 Number of Occurrences

Variable Name:	targetocc
Required:	No
Description:	The number of occurrences of the feature required. If the feature occurs in a planning unit, regardless of its amount, that is considered one occurrence. This can be used in conjunction with or instead of 'target'.
Getting Started	⚠ If an 'overall' target is specified through the 'targetocc' field, then a Zone Target File (zonetarget.dat or zonetarget2.dat) or Zone Contribution File (zonecontrib.dat or zonecontrib2.dat) must be used. See Section 5.4.5 or Section 5.4.6 for more information.

5.3.2.5 *Percentage of Occurrences*

Variable Name:	propocc
Required:	No
Description:	The percentage of occurrences of the feature required. This can be used in conjunction with or instead of the 'prop' field.
Getting Started:	⚠ If an 'overall' target is specified through the 'propocc' field, then a Zone Target File (zonetarget.dat or zonetarget2.dat) or Zone Contribution File (zonecontrib.dat or zonecontrib2.dat) must be used. See Section 5.4.5 or Section 5.4.6 for more information.

5.3.2.6 *Feature Penalty Factor*

Variable Name:	fpf
Required:	No
Description:	The 'fpf' stands for the Feature Penalty Factor (sometimes referred to as the 'spf' or Species Penalty Factor). The fpf is a multiplier that determines the size of the penalty that will be added to the objective function if the target for a feature is not met in a solution. The higher the value, the greater the relative penalty, and the more emphasis Marxan with Zones will place on ensuring that the feature's target is met. The fpf can be the same for all features or unique to each feature.
Getting Started:	Choosing a suitable value for this variable is essential to achieving good solutions in Marxan and Marxan with Zones alike. If it is too low, resulting solutions may fall short of the feature targets. If it is too high, it may impair the software's ability to find efficient solutions. It will often require some experimentation to determine appropriate FPFs. This should be done in an iterative fashion (see Box 7).

Box 7. Calibration of Feature Penalty Factor (FPF)

The Feature Penalty Factor (FPF) needs to be calibrated to find a high enough FPF value that allows you to meet all feature targets.

There are 2 main methods to calibrate the FPF:

1. Run a series of scenarios that explore a range of FPF values (e.g., 1, 10, 50, 100, etc.). Next, plot the FPF values versus the number of missing values. Examine the plot and identify the lowest FPF value able to meet all targets. This is (or is close to) an appropriate FPF value. This method is suitable when there aren't many competing objectives, or when the targets are relatively low across all features. Zonae Cogito (see Section 2.3.1) includes a tool called "Parameter Calibration Tool", which can be used to calibrate the FPF using this method (more information in Watts et al, 2011).
2. Zonae Cogito also includes a second tool which is a variation of the first method and called "Adaptive Calibration Tool". This tool allows the user to specify a minimum proportion of targets that need to be achieved. The tool works by increasing and decreasing the FPF for all features iteratively until it converges on an efficient FPF that meets the desired minimum target level. For more information about the steps the algorithm follows see Watts et al, 2011.
3. When a large number of target objectives cannot be achieved simultaneously, or when a trade-off between competitive objectives is being explored, then a different approach must be used to calibrate FPF values. First, calibrate the FPF for one group of features (e.g., conservation features), while holding the FPF for another group of "opposing" features (e.g. fisheries features) constant. The same technique is then applied for the group of "opposing" features while holding the FPF for the initial group of features constant. This allows you to identify the FPF values required under neutral priority. Based on this 'neutral' value for different groups of features, you can then vary the FPF value for conservation features if meeting conservation targets is a priority over meeting other targets, and vice-versa. For more information about how this technique is applied, see Watts, Steinback and Klein (2008) and Gurney et al (2015).

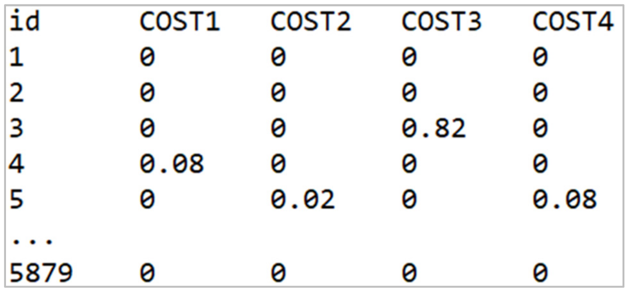
The calibration of FPF should be performed before the calibration of the Zone Boundary Cost (see Section 5.4.2). There might be a need to adjust FPF values after the Zone Boundary Cost calibration. These calibration steps are needed because the parameters interact with each other. Hence, changing one will affect the behavior of the other.

5.3.2.7 Name of feature

Variable Name:	Name
Required:	No
Description and Getting Started:	Indicates the name of the feature. Do not include any spaces or non-alphanumeric characters in the name.

5.3.3 Planning Unit File (pu.dat)

The Planning Unit File (pu.dat) defines the unique identifier of each planning unit in the study region and contains information about costs. This file is the same as in standard Marxan. However, while Marxan is restricted to a single cost field, Marxan with Zones can include multiple cost fields with different names. As such, each planning unit in the pu.dat file may contain one or more costs. An example of the Planning Unit File is shown in Figure 9. In this example, there are 5879 planning units and 4 types of costs.



```
id      COST1   COST2   COST3   COST4
1       0       0       0       0
2       0       0       0       0
3       0       0       0.82    0
4       0.08    0       0       0
5       0       0.02    0       0.08
...
5879    0       0       0       0
```

Figure 9. *An example of a portion of the Planning Unit File (pu.dat)*

The file consists of at least two fields: 'id' field and one or more 'costname' fields (Table 10). The number of 'costname' fields is determined by the number of different costs, where each field reflects one cost variable. To keep track of different costs, the header 'costname' can be replaced with the actual name of the cost (e.g., 'LANDCOST', 'FISHING', 'AREA').

Table 10. *Variable names and requirements for Planning Unit Fie (pu.dat)*

Variable Name	Required	Description
id	Yes	The numeric identifier for this planning unit.
costname	No	The individual cost of each planning unit.

5.3.3.1 Planning Unit ID

Variable Name:	Id
Required:	Yes
Description:	The numeric identifier for each planning unit.
Getting Started:	Values for the variable 'id' can be any number (i.e. there is no requirement to start at number 1), but they must not contain spaces, letters or punctuation. ⚠ This number should not be confused with the variable, 'id', in the Feature File (feat.dat in Section 5.3.2).

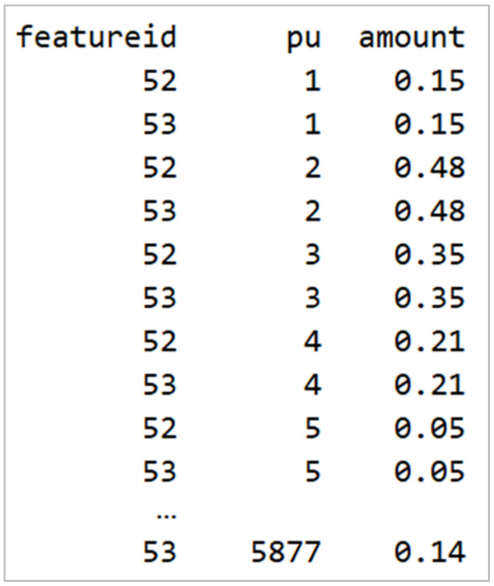
5.3.3.2 Planning Unit Cost

Variable Name:	costname
Required:	No
Description:	The individual cost of including each planning unit in the planning system. Each planning unit can have more than one cost value, which are added into the pu.dat file with different names. The header 'costname' can be replaced with the actual name of the cost but must not include delimiters (spaces, tabs, etc.). Marxan with Zones will use a default value of '1' if this variable is not specified.
Getting Started:	Costs are flexible and can pertain to a range biological, social, cultural, or economic implications of assigning a planning unit to a given zone. Information regarding the type of cost measures that can be considered consult the MGPH. In Marxan with Zones, there are two additional and new files that relate to costs: the Costs File (costs.dat) and the Zone Cost File (zonecostfile.dat). The costs.dat file is simple; it is used to assign a unique identifier and a name to each cost (see Section 5.3.6 for more information). The zonecost.dat file is used to determine how each cost

	will be applied in each zone. This file can be used to assign multiple costs to a zone and to apply different costs to different zones (see Section 5.3.7 for more information).

5.3.4 Planning Unit versus Feature File (puvfeat.dat)

The Planning Unit versus Feature File (puvfeat.dat) contains information on the distribution of features across the planning units, like in standard Marxan[1]. Features can represent a range of ecological, economic and social information. An example of the puvfeat.dat file is shown in Figure 10.



```
featureid      pu  amount
       52       1    0.15
       53       1    0.15
       52       2    0.48
       53       2    0.48
       52       3    0.35
       53       3    0.35
       52       4    0.21
       53       4    0.21
       52       5    0.05
       53       5    0.05
       …
       53    5877    0.14
```

Figure 10. *An example of a portion of the puvfeat.dat file*

The file must have three columns in the following order - featureid, puid, amount - sorted by the planning unit id for Marxan with Zones to execute properly (Table 11).

Table 11. *Variable names and requirements for Planning Unit versus Feature File (puvfeat.dat)*

Variable Name	Required	Description
featureid	Yes	Feature identifier.
puid	Yes	Planning unit identifier.
amount	Yes	Amount of feature in the planning unit.

[1] This file is sometimes referred to as the Planning Unit versus Species Feature File with the file name 'puvsp.dat' or 'puvspr.dat'. Marxan with Zones will run with any of the versions of the file name (i.e., puvfeat.dat, puvsp.dat, or puvspr.dat), so long as the file name in the input folder matches the file name with the .dat extension in the input.dat file. The variable name in the input.dat file must always be entered as PUVSPRNAME. Marxan will halt and generate an error if you use the file name PUVFEATNAME instead.

5.3.4.1 Feature ID

Variable Name:	featureid
Required:	Yes
Description:	This is the unique identifier for each feature. This must correspond to the id numbers used in the Feature File (feat.dat).

5.3.4.2 Planning Unit ID

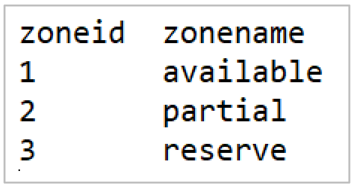
Variable Name:	puid
Required:	Yes
Description:	This is the planning unit identifier. The planning unit id numbers must correspond to the numbers used in the Planning Unit File (pu.dat).

5.3.4.3 Feature Amount

Variable Name:	amount
Required:	Yes
Description:	This identifies the amount of feature in the planning unit.
Getting Stated:	The measurement unit between features can be different. However, the units within a feature must be consistent. The amount for a given feature must be in the same units used to set the target amount for that feature (in the feat.dat file). You should not list cases where a feature does not occur in a planning unit (i.e., you should not have a row with an amount of '0'). Rather the row should be omitted altogether from the file. Marxan with Zones will assume that features only occur in planning units where an amount has been entered. The default amount for a planning unit/feature pair that is omitted from the file is zero.

5.3.5 Zones File (zones.dat)

The Zones File (zones.dat) defines a unique identifier and a zone name for each zone. An example of this file with three types of management zones (an available zone, a partially protected zone, and a reserve zone) is shown in Figure 11.



```
zoneid  zonename
1       available
2       partial
3       reserve
```

Figure 11. *An example of the zones.dat file*

The file must have two fields in the following order: zoneid, zonename, sorted by lowest to highest zoneid and in a consecutive order (Table 12).

Table 12. *Variable names and requirements for the Zones File (zones.dat)*

Variable Name	Required	Description
zoneid	Yes	The numeric identifier number for the zone.
zonename	Yes	The name of the zone.

5.3.5.1 Zone ID

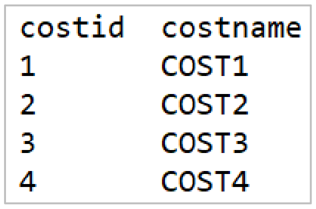
Variable Name:	zoneid
Required:	Yes
Description:	The numeric identifier for the zone. The zoneid must be a positive integer and the file must be sorted by lowest to highest zoneid.
Getting Started:	⚠ Zone id '1' must be specified as the 'available' zone, which is essentially the zone for 'everything else'. This zone could have designated contribution targets and/or zone targets as well. Marxan with Zones may halt and generate errors if this zone is not included in this file.

5.3.5.2 Zone Name

Variable Name:	zonename
Required:	Yes
Description:	This variable indicates the name of the zone.
Getting Started:	It is useful to assign a zone name that reflects an objective, activity or characteristic associated with the zone (e.g., 'conservation' for a zone that aims to protect biodiversity). Do not include any spaces or non-alphanumeric characters in the name.

5.3.6 Costs File (costs.dat)

The Costs File (costs.dat) defines a unique identifier and a name for each cost. An example of the costs.dat file is shown in Figure 12 In this example, there are four costs.



```
costid  costname
1       COST1
2       COST2
3       COST3
4       COST4
```

Figure 12. *An example of the costs.dat file*

This file must include two fields in the following order: costid, costname, and sorted by lowest to highest costid (Table 13). This file does not contain any data for each cost; cost values are held in the pu.dat file, like in standard Marxan.

Table 13. *Variable names and requirements for the Costs File (costs.dat)*

Variable Name	Required	Description
costid	Yes	The numeric identifier for the cost.
costname	Yes	The name of the cost.

If the costs.dat file is not included in Marxan with Zones, the cost values indicated in the pu.dat file will be invalid and a default cost of '1' for all planning units will be used.

5.3.6.1 Cost ID

Variable Name:	costid
Required:	Yes
Description:	The numeric identifier for the cost. The costid must be a positive integer and the file must be sorted by lowest to highest costid.

5.3.6.2 Cost Name

Variable Name:	costname
Required:	Yes
Description:	This variable indicates the name of the cost.
Getting Stated:	You can choose any name for a given cost, so long as it does not include any spaces or non-alphanumeric characters.

5.3.7 Zone Cost File (zonecost.dat)

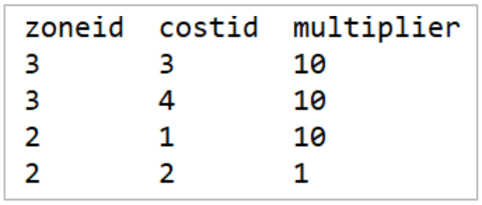
In Marxan with Zones, it is possible to have multiple costs applied to a zone, and it is possible to apply different costs to different zones. This is done through the Zone Cost File (zonecost.dat), a new and required file for Marxan with Zones.

The Zone Cost File contains information on which costs to apply to each zone, as well as a multiplier for the cost. The file must have three fields in the following order: zoneid, costid, multiplier; sorted lowest to highest by zoneid, then by costid (Table 14). The zone-specific multiplier is used to weight and sum costs per zone. It allows the user to assign different weightings to different costs. If some combinations of zone and cost are not present, then that cost will be given a weighting of '0' and will not be considered to formulate a solution.

Table 14. *Variable names and requirements for the Zone Cost File (zonecost.dat)*

Variable Name	Required	Description
zoneid	Yes	The zone identifier.
costid	Yes	The cost identifier.
multiplier	Yes	The number (fraction or integer) that will be multiplied by the specified cost in a given zone.

An example of the zonecost.dat file is shown in Figure 13. In this example, two costs (costid 3 and 4) are applied to Zone 3 (zoneid 3) with a multiplier set to '10'. Zone 2 (zoneid 2) also has two costs applied (costid 1 and 2), but costid 1 has a higher multiplier than costid 2. The high multiplier is used to discourage planning units with high cost values from being included in this zone (except where feature targets cannot be met elsewhere). There are no costs associated with Zone 1, so entries with zoneid 1 are omitted from the file.



```
zoneid  costid  multiplier
3       3       10
3       4       10
2       1       10
2       2       1
```

Figure 13. *An example of the zonecost.dat file*

5.3.7.1 Zone ID

Variable Name:	zoneid
Required:	Yes
Description:	This is the zone identifier. It must be compatible with the Zones File (zones.dat).

5.3.7.2 Cost ID

Variable Name:	costid
Required:	Yes
Description:	This is the cost identifier. It must be compatible with the Costs File (costs.dat).

5.3.7.3 Multiplier

Variable Name:	multiplier
Required:	Yes

Description:	This number can be a fraction or an integer. In a given zone, it will be multiplied by the specified cost. All costs in a given zone will be multiplied by the specified multiplier and then added to give a total cost for each planning unit. The use of weights allows the user to assign different weightings to different costs. For example, if there are 3 costs in one zone, the total cost for that zone would be calculated using the following equation: ***Total C = (C1 * M1) + (C2 * M2) + (C3*M3)*** where *C* = cost and *M* = Multiplier

5.4 Optional Input Files

This section describes the function, format, and associated variables of the optional input files for Marxan with Zones:

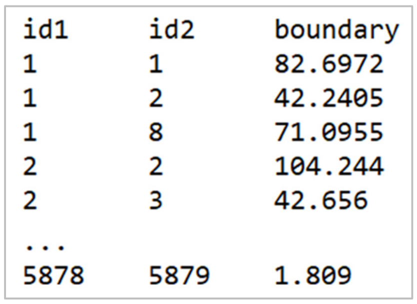
- Boundary Length File (bound.dat)
- Zone Boundary Cost File (zoneboundcost.dat)
- Planning Unit Zone (puzone.dat)
- Planning Unit Lock File (pulock.dat)
- Zone Target (zonetarget.dat) and Zone Target 2 (zonetarget2.dat)
- Zone Contribution (zonecontrib.dat) and Zone Contribution 2 (zonecontrib2.dat)

These are all new files and unique to Marxan with Zones. The only exception is the bound.dat file, which is one of the standard input files in Marxan.

If you wish to use any of these optional files, they must be included in the input folder of your Marxan with Zones database and the associated file name must be entered in the input.dat file (Marxan with Zones will only consider files included in the input.dat file).

5.4.1 Boundary Length File (bound.dat)

The Boundary Length File (bound.dat) contains information on the boundary 'costs' of two planning units (i.e., the cost associated with the boundary between each planning unit). Like standard Marxan, it is an optional file in Marxan with Zones. The file is only required when the boundary length modifier (BLM) value in the input.dat file is set to '1'. An example of a portion of a bound.dat file is shown in Figure 14.



```
id1     id2     boundary
1       1       82.6972
1       2       42.2405
1       8       71.0955
2       2       104.244
2       3       42.656
...
5878    5879    1.809
```

Figure 14. *An example of a portion of the Boundary Length File (bound.dat)*

The bound.dat file must have three fields in the following order – id1, id2, boundary (Table 15). The values in the 'boundary' field are commonly the length of the boundary between a pair of planning units (as identified in the 'id1' and 'id2' fields). Alternatively, it can be modified to reflect the 'cost' of separating two planning units. The boundary values can be used to reflect the inverse distance and direction between two planning units. For an example, see Hermoso et al. (2015).

Table 15. *Variable names and requirements for Boundary Length File (bound.dat)*

Variable Name	Required	Description
id1	Yes	Planning unit identifier.
id2	Yes	Planning unit identifier of the neighbouring planning unit to id1 or the same as id1 for an irremovable boundary.
Boundary	Yes	The actual length of the boundary or the 'cost' of separating two planning units.

5.4.1.1 Planning Unit IDs

Variable Name:	id1 id2
Required:	Yes
Description:	The 'id1' is the planning unit identifier. The 'id2'is the planning unit identifier of the neighbouring planning unit to 'id1'. These do not have to be adjacent planning units, though they usually are. See Section 5.4.1.2 and the MGPH for more details. ⚠ It is important not to duplicate boundaries as Marxan with Zones will sum duplicate boundaries together when calculating the boundary cost.

5.4.1.2 Boundary

Variable Name:	Boundary
Required:	Yes
Description:	The value for the variable 'boundary' can be derived in a variety of ways but it is essentially a relative measure of how important it is to include one planning unit in the zoning system, given the inclusion of the other. For instance, if the planning unit in the column, 'id1', has been added to the system, how important is it that the planning unit in the column,

	'id2', is also included, and vice versa. Because this is analogous to a 'cost' that must be paid if both planning units are not included, this variable is generally referred to as 'boundary cost'.
Getting Started	In its most typical application, boundary cost reflects the actual geographical length of the boundary between two adjacent planning units, and this is a good place to begin. This cost can, however, be easily adjusted to reflect some other association between planning units, for instance, based on distance and direction. For an example see Hermoso, et al., 2015. It is very important that if some relative measure, other than actual boundary length is used, the chosen metric must be well justified. Transparency and defensibility are two of the core strengths of systematic conservation planning (also important in other spatial planning applications). Two planning units that are not adjacent to each other can still incur a 'boundary cost' if there is an important relationship between them. For instance, if a species requires habitat found in disparate planning units, either at different times of the year or during different life stages, then it makes little sense to protect one and not the other. This method could also be used to identify paths of connectivity between planning units, for instance, larval transport in marine systems, or hydrological flows in aquatic systems. In some cases, there will be no possibility of removing a boundary by including a neighbouring planning unit in the zoning system. This may happen, for instance, at the edge of a territorial jurisdiction or mandated planning region. Because planning units at the edge of a region will generally have shorter 'shared' boundaries, selection may be biased towards these planning units. This may be undesirable. To avoid biasing the selection of these planning units, they should feature in the Boundary Length File as 'irremovable boundaries'. This can be accomplished by specifying the length of a planning unit's boundary with itself, i.e. by repeating the same planning unit id in both the 'id1' and 'id2' columns. The value entered in 'boundary' should always be '0' or greater. Although it is not generally necessary to specify cases where there is no cost between planning units, a zero cost boundary can be useful if you want to identify two planning units as neighbours but there is no actual

	boundary cost. This may be necessary if you have set minimum clump sizes for some conservation features (see Section 5.3.2.5). Consult the MGPH for more guidance regarding the bound.dat file.

5.4.2 Zone Boundary Cost File (zoneboundcost.dat)

The Zone Boundary Cost File (zoneboundcost.dat) is a new and optional file for Marxan with Zones. It can be used to prescribe the spatial relationship between zones. This is useful to encourage further separation of conflicting uses (e.g., separate a forestry zone from a protected area zone) or to cluster zones which share compatible management objectives (e.g., place a buffer zone that is partially protected adjacent to a conservation zone that is fully protected).

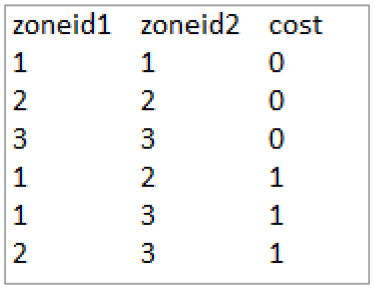
The file must have three fields in the following order – zoneid1, zoneid2, cost - sorted lowest to highest by zoneid1, then by zoneid2 (Table 16). This file is similar to the bound.dat file but relates the boundary 'cost' of zones rather than planning units.

Table 16. *Variable names and requirements for Zone Boundary Cost (zoneboundcost.dat)*

Variable Name	Required	Description
zoneid1	Yes	Zone identifier.
zoneid2	Yes	Zone identifier, different from zoneid1.
cost	Yes	The cost between zoneid1 and zoneid2.

If the zoneboundcost.dat file is not present, all boundaries between zones will be assigned a cost of '0'. Likewise, if a zone-zone boundary cost is not indicated in this file, it will be given a cost of '0'.

An example of the zoneboundcost.dat file is shown in Figure 15. In this example, we want all three zones be spatially compact. To do so, we are going to set a cost of 1 between each zone combination. This is going to cause an aggregation of planning units that belong to the same zone, because it is less costly than a spread of planning units that belong to different zones.



zoneid1	zoneid2	cost
1	1	0
2	2	0
3	3	0
1	2	1
1	3	1
2	3	1

Figure 15. *An example of the Zone Boundary Cost File (zoneboundcost.dat)*

In practice, the zoneboundcost.dat file will need to be calibrated to determine appropriate cost values for a given project. Guidance for calibrating the zone boundary cost is provided in Box 8.

5.4.2.1 Zone ID1

Variable Name:	zoneid1
Required:	Yes
Description:	Zone identifier - must match the zone id indicated in the zones.dat file for that zone.

5.4.2.2 Zone ID2

Variable Name:	zoneid2
Required:	Yes
Description:	Zone identifier - must match the zone id indicated in the zones.dat file for that zone.

5.4.2.3 Cost

Variable Name:	Cost
Required:	Yes
Description:	The cost between zoneid1 and zoneid2. If cost is not indicated, a default cost of '1' will be assigned. This field accepts decimal values (e.g., 0.1) and negative values (-1).
Getting Started:	The zoneboundcost.dat file must be calibrated to determine the appropriate cost values. See Box 8 for guidance on calibrating this parameter in Marxan with Zones.

Box 8. Calibrating the Zone Boundary Cost File

The aim for calibrating the zone boundary cost is to determine the multiplication factor (or cost) needed to achieve different spatial arrangements of zones. Using the Rottnest Island case study, we demonstrate a method to calibrate the zone boundary relationship for spatially clumped zones. Recall that the zone boundary relationships are set in the Zone Boundary Cost file (zoneboundcost.dat). For the Rottnest Island case study, the default zoneboundcost.dat file is:

```
zoneid1  zoneid2  cost
1        1        0
2        2        0
3        3        0
1        2        0
1        3        0
2        3        0
```

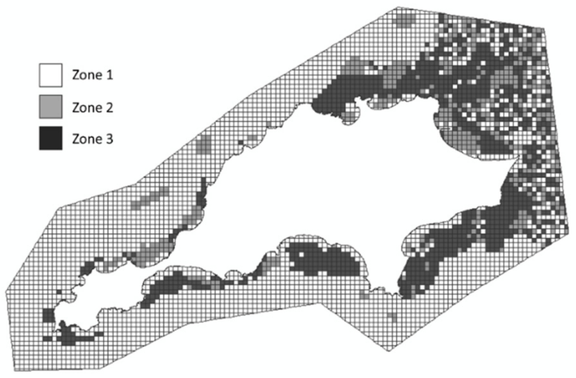
This can also be presented in matrix form as:

	Zone 1	Zone 2	Zone 3
Zone 1 (Multi Use)	-	0	0
Zone 2 (Partial Protection)	0	-	0
Zone 3 (High Protection)	0	0	-

Each element in the matrix represents the boundary relationship between the pair of zones referencing that element. When calibrating the zone boundary matrix, we always set the relationship between any zone and itself to zero.

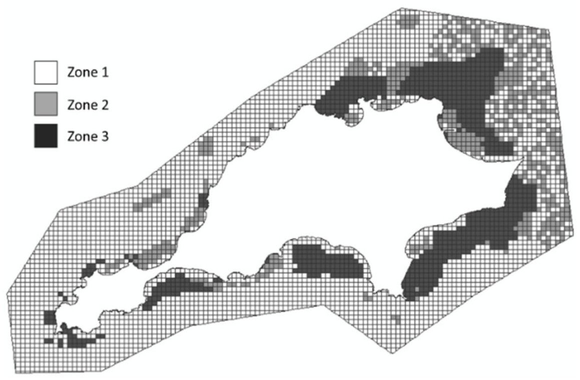
The following steps can be used to calibrate the Zone Boundary Cost to achieve spatially clumped zones:

1. Run the system using the default matrix and view spatial output of solutions.
2. If you are not satisfied by the level of spatial clumping (see map below), increase the “cost” between one zone and all the other zones by a small number. A good starting point is to use values that allow the largest boundary between planning units to become a similar order of magnitude to the most expensive planning unit. In the Rottnest Island case study, the highest planning unit cost is 17 and the longest boundary is 320, so you may want to start with a value no larger than 0.01.



zoneid1	zoneid2	cost
1	1	0
2	2	0
3	3	0
1	2	0
1	3	0
2	3	0

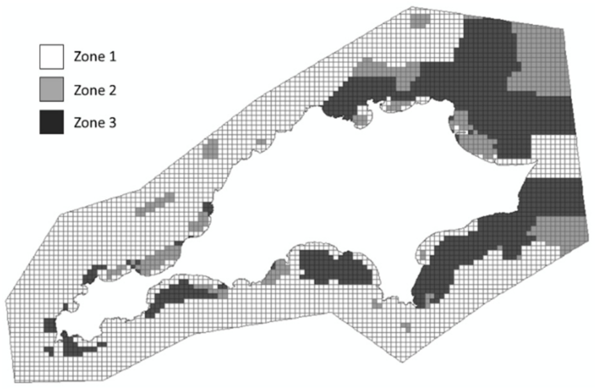
3. Run the system and view the spatial outputs of solutions. In this example, increasing the cost between Zone 1 and Zone 3, along with Zone 2 and Zone 3, has forced Zone 3 to clump because it is more efficient (or least costly in terms of the boundary cost).



zoneid1	zoneid2	cost
1	1	0
2	2	0
3	3	0
1	2	0
1	3	0.001
2	3	0.001

4. Steps 2 and 3 can be repeated until a desired level of clumping has been achieved for a zone.

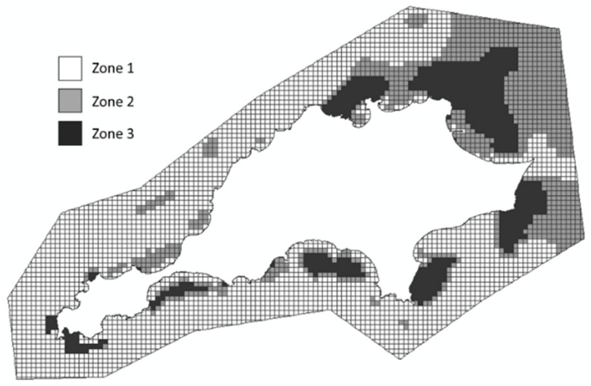
5. Increase the "cost" for the remaining zone combinations that have not achieved the desired level of clumping. Run the system and view outputs. In the example below, the cost between Zone 1 and Zone 2 has been increased to achieve clumping for Zone 2.



zoneid1	zoneid2	cost
1	1	0
2	2	0
3	3	0
1	2	0.001
1	3	0.001
2	3	0.001

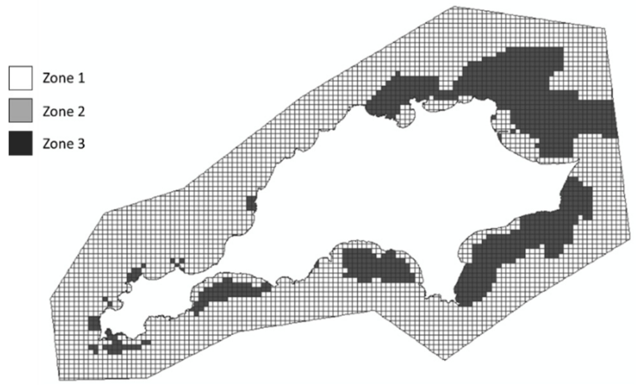
6. Repeat steps 5 and 6 until all zones have achieved some level of clumping.

If the goal is to achieve a zone-nested configuration, that is, one zone to buffer around another zone, then we need to increase the weights between the zones that we want to keep apart. In example below, by applying a larger weight between Zone 1 and Zone 3, we are forcing Zone 2 to become a buffer between the two zones.



zoneid1	zoneid2	cost
1	1	0
2	2	0
3	3	0
1	2	0.0001
1	3	0.05
2	3	0.0001

⚠ If the "cost" between pairs of zones is too high, then this may cause one zone to be excluded from the zone configuration. In the example below, because the weights between Zone 2 and any other zone are higher than any other combination of zones, is forcing Zone 2 to be squeezed out of the zone configuration.



zoneid1	zoneid2	cost
1	1	0
2	2	0
3	3	0
1	2	1
1	3	0.001
2	3	1

5.4.3 Planning Unit Zone File (puzone.dat)

The Planning Unit Zone File (puzone.dat) is an optional file for Marxan with Zones that can be used to restrict certain planning units to two or more zones. For example, you may want to restrict planning units with existing protected areas to zones that provide partial or full protection of species and habitats.

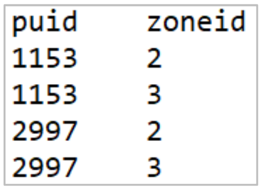
Do not use this file to lock planning units to a single zone; use the Planning Unit Lock File (pulock.dat) for this specific purpose.

The puzone.dat file must have two fields in the following order – puid, zoneid - sorted lowest to highest by puid, then by zoneid (Table 17).

Table 17. *Variable names and requirements for Planning Unit Zone (puzone.dat)*

Variable Name	Required	Description
puid	Yes	Planning unit identifier that is restricted to a specific zone.
zoneid	Yes	Zone identifier that the planning unit in 'puid' is restricted to.

An example of the puzone.dat file is shown in Figure 16. In this example, the puzone.dat file is being used to restrict two planning units (puid 1153 and puid 2997) to Zone 2 or 3.



```
puid    zoneid
1153    2
1153    3
2997    2
2997    3
```

Figure 16. *An example of the Planning Unit Zone (puzone.dat)*

5.4.3.1 Planning Unit ID

Variable Name:	puid
Required:	Yes
Description:	Unique identifier for the planning unit that is restricted to a specific zone. The same planning unit can be listed more than once to indicate restriction to more than one zone.

5.4.3.2 Zone ID

Variable Name:	zoneid
Required:	Yes
Description:	Zone identifier that the planning unit in puid is restricted to.

5.4.4 Planning Unit Lock File (pulock.dat)

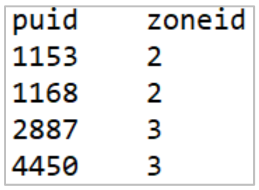
The Planning Unit Lock File (pulock.dat) is an optional file in Marxan with Zones that can be used to restrict certain planning units to a single zone. For example, planning units corresponding to land parcels with forestry permits could be locked into a forestry zone.

The pulock.dat file must have two fields in the following order – puid, zoneid - sorted lowest to highest by puid (Table 18).

Table 18. *Variable names and requirements for Planning Unit Lock (pulock.dat)*

Variable Name	Required	Description
Puid	Yes	Planning unit identifier that is restricted to a specific zone.
Zoneid	Yes	Zone identifier that the planning unit in 'puid' is restricted to.

An example of a pulock.dat file is shown in Figure 17. In this example, the first two planning units listed in the file are locked into Zone 2, whereas the last two planning units are locked into Zone 3.



```
puid    zoneid
1153    2
1168    2
2887    3
4450    3
```

Figure 17. *An example of the Planning Unit Lock File (pulock.dat)*

5.4.4.1 Planning Unit ID

Variable Name:	puid
Required:	Yes
Description:	Unique identifier of the planning unit that is restricted to a specific zone.

5.4.4.2 Zone ID

Variable Name:	Zoneid
Required:	Yes
Description:	Zone identifier that the planning unit in 'puid' is restricted to.

5.4.5 Zone Target (zonetarget.dat) & Zone Target 2 (zonetarget2.dat) Files

The Zone Target Files (zonetarget.dat and zontarget2.dat) are optional files in Marxan with Zones. These files allow users to set targets for features in each zone (i.e., zone-specific targets). This is useful when users want to separate conflicting features (e.g., conservation features and fishing features) into appropriate zones. For example, targets for fishing features can be set for a multi-use zone to ensure a certain proportion of fishing sites is maintained for fishing activities.

However, zone targets are related to the overall targets in the Feature File (feat.dat) in a sense, because the total zone targets for a given feature cannot be larger than the overall target for that feature. For example, if the overall target for a conservation feature is 30%, you can set 20% to be met in a high conservation zone, and 10% in a partial protection zone. Unlike the feat.dat file, the zone targets are not related to the Zone Contribution file (zonecontrib.dat or zonecontrib2.dat).

Zone targets can be used instead of, or at the same time as overall targets.

There are two types of Zone Target Files: zonetarget.dat and zontarget2.dat. Only one of the files should be used. If both are used, the zonetarget2.dat file will override the zonetarget.dat file. The function, structure, and difference between these files is described below.

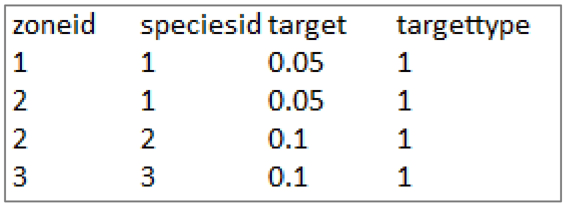
Zone Target (zonetarget.dat): The zonetarget.dat file allows the user to set a target for each feature in each zone (i.e. zone-specific target). It specifies which zone each target is to be met in. The file must be sorted from lowest to highest value, first by the 'zoneid' field, and then by the 'featureid' and 'target' fields (Table 19). The 'target type' field indicates whether the target values are absolute amounts (0), proportions (1), occurrences (2), or occurrence proportions (3).

Table 19. *Variable names and requirements for Zone Target (zoneboundcost.dat)*

Variable Name	Required	Description
zoneid	Yes	The zone identifier.

featureid	Yes	The feature identifier.
target	Yes	The target amount (in unit of puvfeat.dat file), percentage, occurrence or occurrence percentage of the feature to include in the specified zone (i.e. zone-specific target).
targettype	No	Target type indicates the type of target specified in the target column (i.e. amount, percentage, occurrence, occurrence percentage).

An example of the zontarget.dat file is provided in Figure 18. In this example, we have three features and three zones. Based on the Zone Target File (zonetarget.dat), we want to achieve 5% of Feature ID1 in Zone 1 and 5% in Zone 2, while for Feature ID2 and ID3 we want to achieve their overall targets exclusively in Zone 2 and Zone 3 respectively.



zoneid	speciesid	target	targettype
1	1	0.05	1
2	1	0.05	1
2	2	0.1	1
3	3	0.1	1

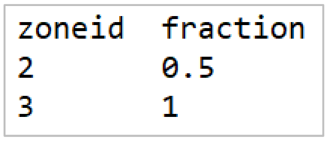
Figure 18. *An example of the Zone Target File (zonetarget.dat)*

<u>Zone Target 2 (zonetarget2.dat):</u> This file is very similar to the zonetarget.dat file. Like the zonetarget.dat, the zontarget2.dat file must include the 'zoneid' and 'featureid' fields (Table 20). Unlike the zonetarget.dat, the zonetarget2.dat file does not include the 'featureid' field. As such, targets specified in the zonetarget2.dat apply to all features within the specified zone.

Table 20. *Variable names and requirements for Zone Target 2 (zoneboundcost2.dat)*

Variable Name	Required	Description
zoneid	Yes	The zone identifier.
target	Yes	The target amount (in unit of puvfeat.dat file), percentage, occurrence or occurrence percentage of the feature to include in the specified zone (i.e. zone-specific target).
targettype	No	Target type indicates the type of target specified in the target column (i.e. amount, percentage, occurrence, occurrence percentage).

An example of a zonetarget2.dat file is provided in Figure 19.



```
zoneid  fraction
2       0.5
3       1
```

Figure 19. *An example of zonetarget2.dat*

5.4.5.1 *Zone ID*

Variable Name:	zoneid
Required:	Yes
Description:	The zone identifier.

5.4.5.2 *Feature ID* (only applicable to the zonetarget.dat file)

Variable Name:	featureid
Required:	Yes
Description:	The feature identifier. This variable is only required in the zonetarget.dat file. It is not present in the zonetarget2.dat file.

5.4.5.3 *Target*

Variable Name:	target
Required:	Yes
Description:	The target amount (in unit of puvfeat.dat file), percentage, occurrence or occurrence percentage of the feature to include in the specified zone (i.e. zone-specific target).
Getting started	Zone-specific targets are typically used when there is a need to separate competing objectives in different zones, for example conservation objectives and resource extraction (see Mazor et al., 2014). They have also been used to explore trade-offs between achieving conservation targets and socio-economic interests (see Klein et al., 2010).

5.4.5.4 *Target Type*

Variable Name:	featureid
Required:	Yes
Description:	Target type indicates the type of target specified in the target column (i.e. amount, percentage, occurrence, occurrence percentage). If this column is not included, Marxan with Zones will use a default of '0'. Target type values: 0 = Amount (in unit of puvfeat.dat file) target of feature. Similar to 'target' in the feat.dat file. 1 = Percentage target as proportion of total amount of feature. Similar to 'prop' in the feat.dat file. 2 = Occurrence target. If the feature occurs in a planning unit, regardless of its amount, it is considered one occurrence. Similar to 'targetocc' in the feat.dat file. 3 = Percentage target as proportion of total occurrences of feature. Similar to 'propocc' in feat.dat.

Box 9. Zones target setting in Rottnest Island case study

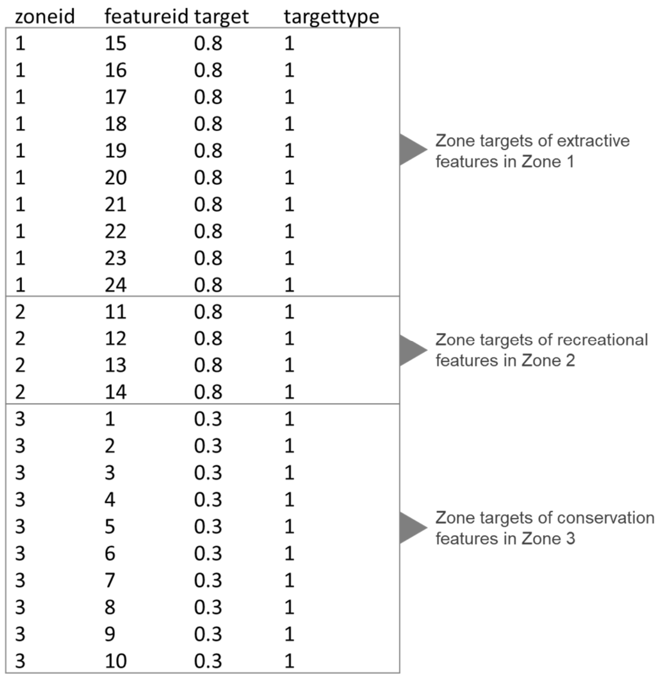
The goal here is to create a multiple use park system made up of three management zones where different activities are allowed to take place in each zone:

- Zone 1 or Resource Extraction Zone, where only commercial fishing activities are allowed
- Zone 2 or Recreational Activities Zone, where only recreational activities such as diving and snorkeling are allowed.
- Zone 3 or Conservation Zone, where no human activities are allowed and where conservation objectives can be met.

Using the Zone Target file in Marxan with Zones we can set zone-specific targets and set restrictions for each of the zones. And example of how these zone targets can be specified below:

	Zone 1	Zone 2	Zone 3
Conservation Features	0	0	0.3
Recreational Features	0	0.8	0
Extractive Features	0.8	0	0

The corresponding zonetarget.dat file that would be used in this scenario is shown below:



zoneid	featureid	target	targettype
1	15	0.8	1
1	16	0.8	1
1	17	0.8	1
1	18	0.8	1
1	19	0.8	1
1	20	0.8	1
1	21	0.8	1
1	22	0.8	1
1	23	0.8	1
1	24	0.8	1
2	11	0.8	1
2	12	0.8	1
2	13	0.8	1
2	14	0.8	1
3	1	0.3	1
3	2	0.3	1
3	3	0.3	1
3	4	0.3	1
3	5	0.3	1
3	6	0.3	1
3	7	0.3	1
3	8	0.3	1
3	9	0.3	1
3	10	0.3	1

Zone targets of extractive features in Zone 1
Zone targets of recreational features in Zone 2
Zone targets of conservation features in Zone 3

5.4.6 Zone Contribution (zonecontrib.dat) & Zone Contribution 2 (zonecontrib2.dat) Files

The Zone Contribution (zonecontrib.dat) and Zone Contribution 2 (zonecontrib2.dat) files are not required to run Marxan with Zones, although one of these files must be used if an overall target is specified in the Feature File (feat.dat). The feat.dat and zonecontrib.dat (or zonecontrib2.dat) files work in tandem with each other. The feat.dat specifies the overall target of features, while the zoncontrib.dat (or zoncontrib2.dat) specifies differential contribution rates to the overall target. If Marxan with Zones is run without specifying the zone contribution, the target for each feature in the feat.dat file will be met across all of the zones, and each zone will contribute at 100% to meeting the target.

⚠ The Zone Contribution Files are not related to the Zone Target Files.

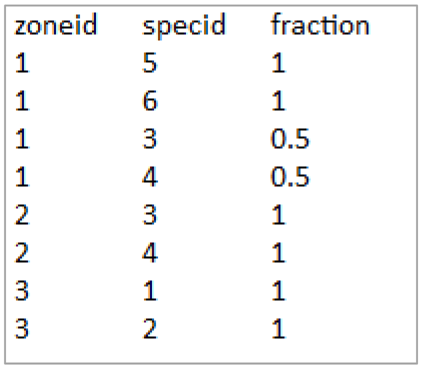
The structure and function of the Zone Contribution (zonecontrib.dat) and Zone Contribution 2 (zonecontrib2.dat) files are described below, along with the different between the two versions.

Zone Contribution (zonecontrib.dat): This file allows you to set (by percentage) the level of contribution of each feature in each zone. This is useful if you have several different types of features and if you want the overall target to be met across multiple zones. This file must contain three fields: zoneid, featureid and fraction (Table 21).

Table 21. *Variable names and requirements for Zone Contribution (zonecontrib.dat)*

Variable Name	Required	Description
zoneid	Yes	The zone identifier.
featureid	Yes	The feature identifier.
fraction	Yes	The contribution fraction for this feature in this zone as applied to the overall target specified in the feature file.

An example of the Zone Contribution File is shown in Figure 20. In this example, the contents of a planning unit in Zone 1 can contribute 50% towards meeting recreation targets (e.g., surfing and diving) and 100% towards meeting fishing targets (e.g., trolling). The contents of a planning unit allocated in Zone 2 can only contribute at meeting recreational targets, while planning units allocated to Zone 3 can only contribute at meeting conservation feature targets (e.g., coral and fish species).



zoneid	specid	fraction
1	5	1
1	6	1
1	3	0.5
1	4	0.5
2	3	1
2	4	1
3	1	1
3	2	1

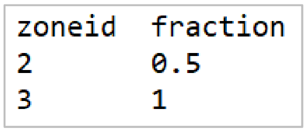
Figure 20. *An example of the Zone Contribution File (zonecontrib.dat)*

Zone Contribution 2: The "zonecontrib2.dat" file allows users to specify a contribution fraction for each zone. It contains the same 'zoneid' and 'fraction' fields as the zonecontrib.dat file, but it does not include the 'featureid' field (Table 22).

Table 22. *Variable names and requirements for Zone Contribution 2 (zonecontrib2.dat)*

Variable Name	Required	Description
zoneid	Yes	The zone identifier.
fraction	Yes	The contribution fraction for all features in this zone as applied to the overall target specified in the feature file.

This file should be used if all features in a zone have the same contribution fraction. An example of the zonecontrib2.dat file is shown in Figure 21.



```
zoneid  fraction
2       0.5
3       1
```

Figure 21. *An example of the Zone Contribution 2 File (zonecontrib2.dat)*

5.4.6.1 *Zone ID*

Variable Name:	Zoneid
Required:	Yes
Description:	The zone identifier.

5.4.6.2 *Feature ID* (only applicable to the zonecontrib.dat file)

Variable Name:	featureid
Required:	Yes
Description:	The feature identifier. This variable is only required in the zonecontrib.dat. It is not present in the zonecontrib2.dat.

5.4.6.3 *Fraction*

Variable Name:	fraction
Required:	Yes
Description:	This is the contribution fraction. Negative contributions are not allowed. Zone contribution: The contribution fraction for *this* feature in this zone as applied to the overall target specified in the feature file. Zone contribution 2: The contribution fraction for *all* features in this zone as applied to the overall target specified in the feature file.
Getting started	Zone contributions are used to incorporate how effective (either ecologically or based on management actions) different zones are at meeting set objectives, be either conservation objectives or other socio-economic objectives. Incorporating zone effectiveness requires information on how effective each zone is at helping achieve the targets specified. However, this information is often not available because it is costly and time consuming to acquire (Makino et al., 2013). Often, scenarios are used to explore a range of different contribution levels to analyze this affect the location and cost of spatial priorities for different zones. Using expert knowledge to inform zone effectiveness can be very valuable in absence of empirical data (Mills et al., 2011).

Box 10. Zones contributions setting in Rottnest Island case study

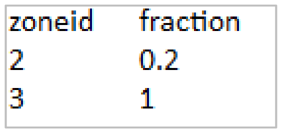
In this scenario, we continue with the multi-use park system and the three different management zones with different levels of protection:

- Zone 1 or Multiuse Zone
- Zone 2 or Partial Protection Zone
- Zone 3 or Full Protection Zone

Because each zone has different levels of protection each will contribute differently at meeting overall conservation targets (e.g., 30% of each conservation feature). For example, if a conservation feature is in Zone 3 (full protection) the amount of that feature will contribute fully (that is, 100%) to meeting the overall target for that feature (e.g., if 100 m^2 of seagrass in present in Zone 3, then the total contribution is also 100 m^2 when meeting 30% of the overall seagrass conservation target). However, if a conservation feature is found in Zone 2, then the amount of that feature will contribute only 20% to meeting the overall target for that feature (e.g., if 100 m^2 of seagrass is present in Zone 2, only 20 m^2 will count towards meeting 30% of overall seagrass conservation target). Because Zone 1 does not guarantee any level of protection, no amount of conservation features found in this zone will contribute at meeting overall conservation targets. Zone contributions set for conservation features are summarized in the table below:

	Zone 1	Zone 2	Zone 3
Conservation Features	0%	20%	100%

In this scenario, the zone contributions file (zonecontrib2.dat) would be set as outlined below:



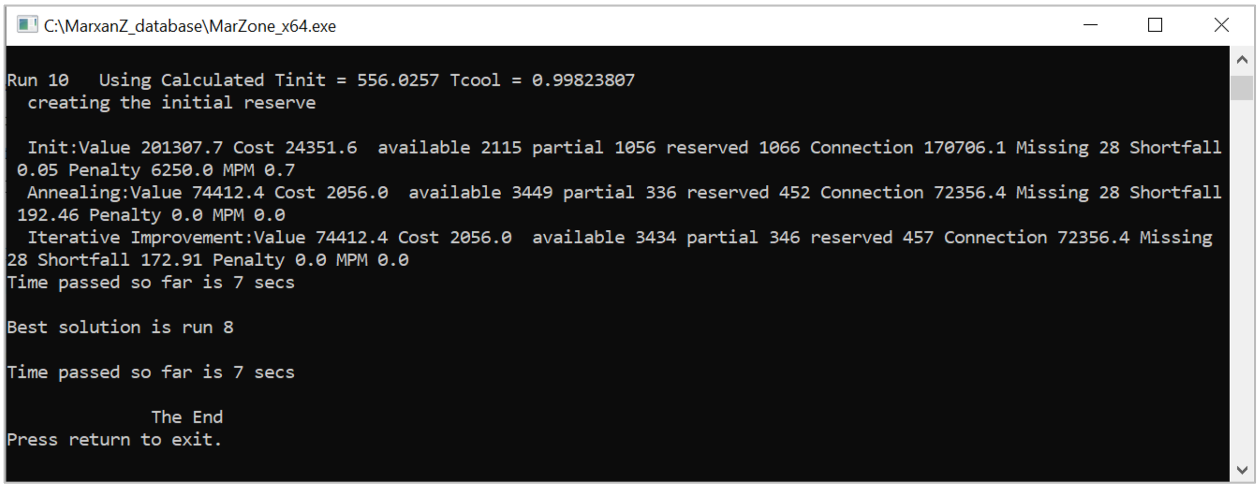
```
zoneid   fraction
2        0.2
3        1
```

6 Running the Software

Running the Marxan with Zones program is extremely simple. Once all the input files are ready, you simply need to double click on the 'MarxanZone.exe' file and the program will start automatically. To run successfully, however, the folder containing the program must be set up so that Marxan with Zones can find the required files and save the necessary outputs.

If Marxan with Zones executes successfully (Figure 22 displays a screen output of a successful run), a program screen showing information on the details and progress of each run will be displayed (unless Silent Running has been selected; see Section 5.3.1.13 and Section 7.3). Do not worry if this proceeds too quickly for you to read. All the necessary details will be saved in the output file folder in the Screen Log File, called 'output_log.dat' (see Section 7.4.10). When Marxan with Zones completes the pre-set number of runs, it will stop, and the program screen will remain visible. Pressing 'Enter' will exit the program and close the screen.

⚠ If Marxan with Zones closes prematurely or halts with an error message, it is likely to mean there is a problem with the format of one or more input files (see Appendix A for more information).


C:\MarxanZ_database\MarZone_x64.exe

```
Run 10   Using Calculated Tinit = 556.0257 Tcool = 0.99823807
  creating the initial reserve

  Init:Value 201307.7 Cost 24351.6  available 2115 partial 1056 reserved 1066 Connection 170706.1 Missing 28 Shortfall
 0.05 Penalty 6250.0 MPM 0.7
  Annealing:Value 74412.4 Cost 2056.0  available 3449 partial 336 reserved 452 Connection 72356.4 Missing 28 Shortfall
 192.46 Penalty 0.0 MPM 0.0
  Iterative Improvement:Value 74412.4 Cost 2056.0  available 3434 partial 346 reserved 457 Connection 72356.4 Missing
28 Shortfall 172.91 Penalty 0.0 MPM 0.0
Time passed so far is 7 secs

Best solution is run 8

Time passed so far is 7 secs

              The End
Press return to exit.
```

Figure 22. *A successful run in MarxanZone.exe*

7 Outputs

7.1 Overview of Output Files

Marxan with Zones can generate a variety of outputs, including multiple solutions for zoning configurations. The Input Parameter File (input.dat) is used to specify which outputs Marxan with Zones should save, along with the format for saving them (.csv, .dat, or .txt). The different types of outputs are described in Section 7.4.

When interpreting outputs, it is important to remember that Marxan with Zones provides decision support and is not the decision maker. The software does not provide the ultimate solution, and many of the input parameters require experimentation. Each output solution should be subject to visual inspection, and local knowledge should be incorporated to update and improve future runs. Sensitivity analysis of key parameters can further improve the robustness of results. See Section 8 for more guidance for interpreting results.

7.2 Output File Management & Format

In the Input Parameter File (input.dat), the 'OUTPUTDIR' is used to tell Marxan with Zones the name of the folder where it should save the output files. We suggest using the default folder name 'output' and then changing the file name once the run is complete.

Output files can be saved with a .dat, .txt., or .csv extension. The desired file format of each output file is specified by the user in the input.dat file. We recommend using the extension .csv for most output files. Exceptions include the Screen Log File and the Scenario Details File, which should be saved as a .txt or .dat format.

Marxan with Zones does not generate any mapped results, although solutions and other outputs (e.g., summed solution) can be visualised within select supporting software, such as Zonae Cogito, or with the use of GIS software like ArcGIS (www.esri.com) and QGIS (www.qgis.org). Basic instructions for visualizing key outputs in ArcGIS or QGIS are provided in information boxes in Section 7.4.

7.3 Screen Output

As Marxan with Zones runs, some outputs can be displayed on the screen. For the screen output, users can choose between four levels of information: 0 (Silent Running), 1 (Results Only), 2 (General Progress), and 3 (Detailed Progress). The level of information is termed the 'verbosity level' and is set using the VERBOSITY variable in the Input Parameter File (input.dat), as previously described in Section 5.3.1.13. The default setting is level 2 and is recommended for most cases.

If the verbosity level is set to '0', then no information about the scenario will be displayed on the screen. This option should only be used if you are confident in Marxan with Zones execution and if you are saving all necessary outputs manually.

Level 1 provides a bit of information, displaying the run numbers and the time it took to generate all runs. Each run number will appear on the screen as Marxan with Zones generates a final solution for that run. While no information about each run is provided, the run numbers lets you know how many runs have been completed and can give you an idea of how long it may take to complete all the runs (e.g., if it takes one minute to complete a single run, then it will take 100 minutes to complete 100 runs).

Levels 2 and 3 will display the run numbers and more time measures, along with details on the data being entered, information on the initial solution, and results of the solution for each run (see Table 23 for a description of summary information for each run). If adaptive annealing is being used, the annealing parameters calculated during pre-processing (initial temperature and decrement) will also be displayed on the screen. If simulated annealing is being used and the verbosity is set to '3', the screen will include information about the current configuration score each time the temperature is decremented.

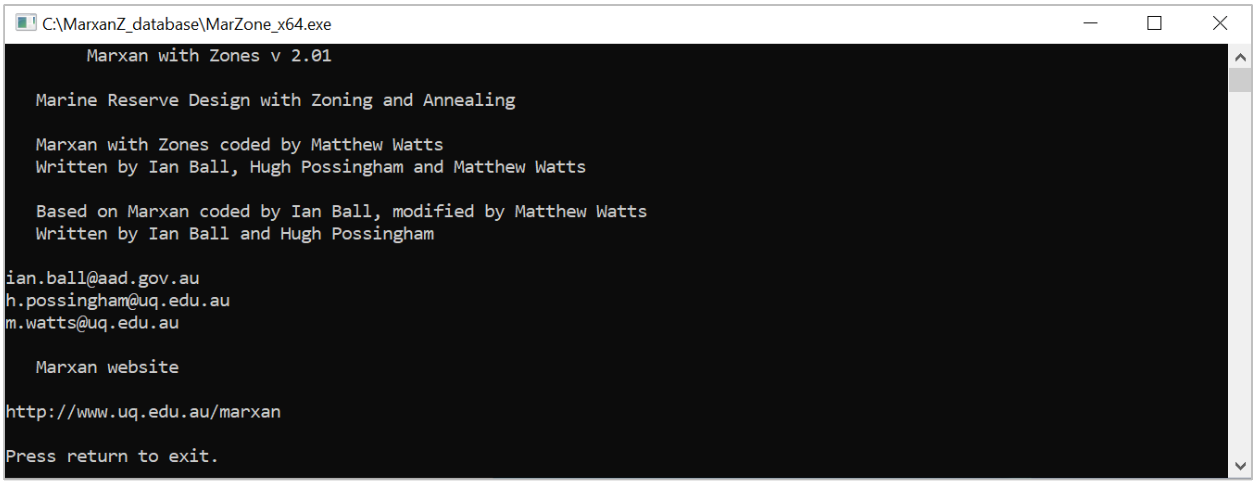
Table 23. *Basic summary information of each run*

Information	Description
Run	The number of the repeat run (e.g., Run 1, Run 2, Run 3, etc.)
Value	The overall objective function value for the solution
Cost	The cost of the solution; i.e., the sum of the costs of all planning units in the solution in the units of the cost field in the pu.dat file
Zonename	The number of planning units in that zone
Connection	The boundary 'cost' (or connection strength) of the solution
Missing	The number of features that did not achieve their targets in the solution
Shortfall	The amount by which the targets for features have not been met
Penalty	The penalty added to the objective function for not meeting all feature targets
MPM	The minimum proportion met (MPM) for the worst performing feature

We recommend using at least level 2 for the following reasons. First, if you a running more than one optimisation procedure (i.e., simulated annealing followed by iterative improvement), the higher verbosity levels allow you to get a feel for how much work each of the different procedures is doing. For instance, if most of the gains in value and target achievement are being made in the iterative improvement phase, you know that either the annealing parameters or the penalties need alteration. Second, if you want to begin using a fixed annealing schedule, this output can give you an idea of the sort of values Marxan with Zones is calculating using its adaptive annealing module. Finally, if you have set some constraints on the initial solution, this output will quickly confirm that this information is being included.

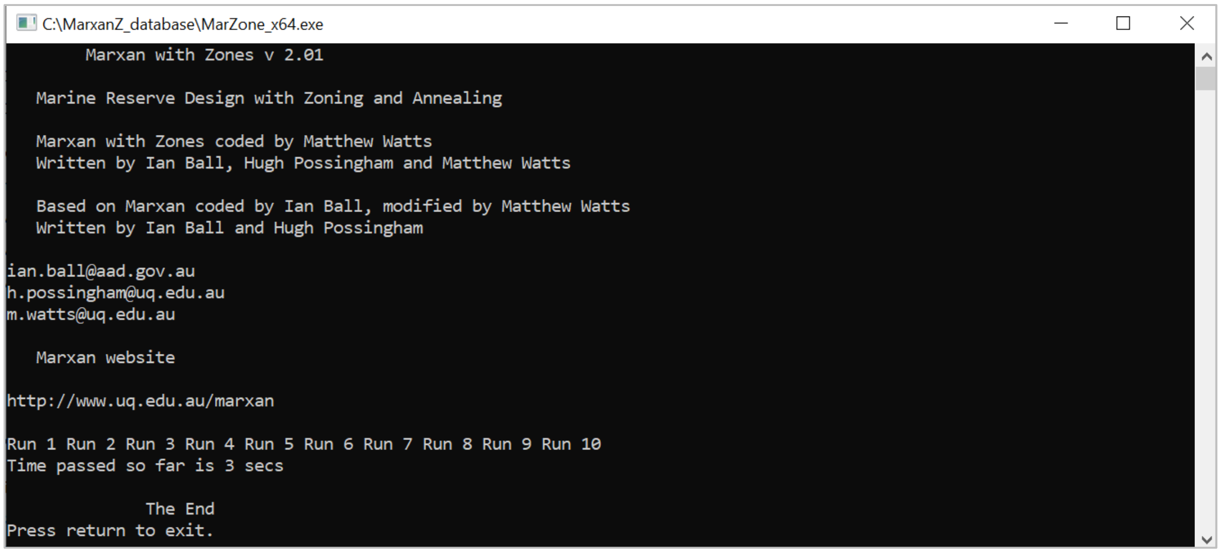
We recommend keeping the verbosity to its default setting (i.e., Level 2), as level 3 typically provides more detail than is necessary and can increase the processing time. However, level 3 can be set to help identify certain problems (e.g. if the numbers do not change and it is "stalled") as it allows users to observe the annealing progress. For this reason, some users prefer to use this setting to visually check that the program appears to be running well.

Note that you can save the screen output as an output file, called Screen Log File (see Section 7.4.10 for more information), to look at it after Marxan with Zones has finished running.



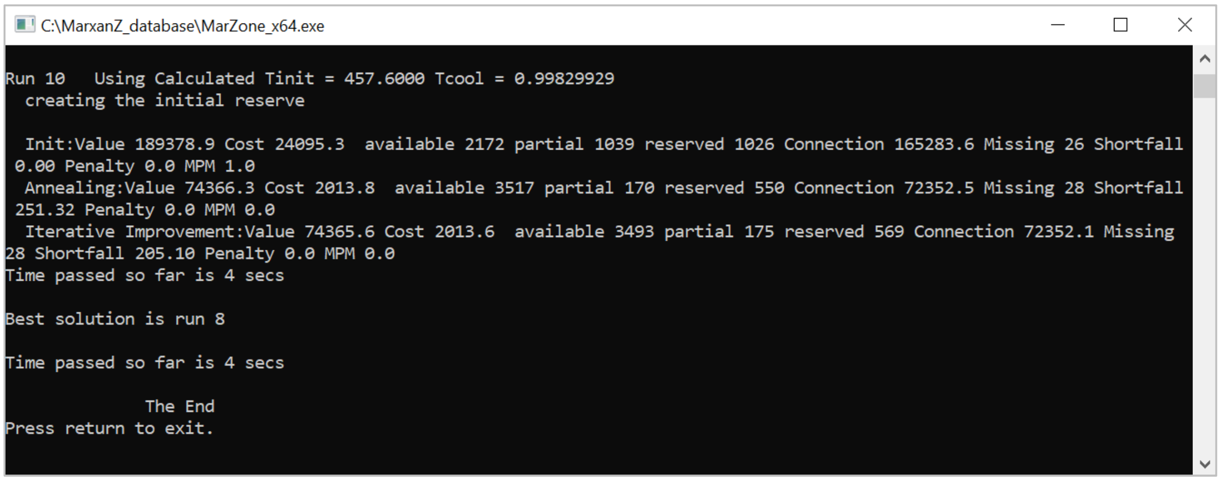
```
        Marxan with Zones v 2.01

   Marine Reserve Design with Zoning and Annealing

   Marxan with Zones coded by Matthew Watts
   Written by Ian Ball, Hugh Possingham and Matthew Watts

   Based on Marxan coded by Ian Ball, modified by Matthew Watts
   Written by Ian Ball and Hugh Possingham

ian.ball@aad.gov.au
h.possingham@uq.edu.au
m.watts@uq.edu.au

   Marxan website

http://www.uq.edu.au/marxan

Press return to exit.
```

Figure 23. *Example of the screen output with verbosity level 0 (Silent Running)*



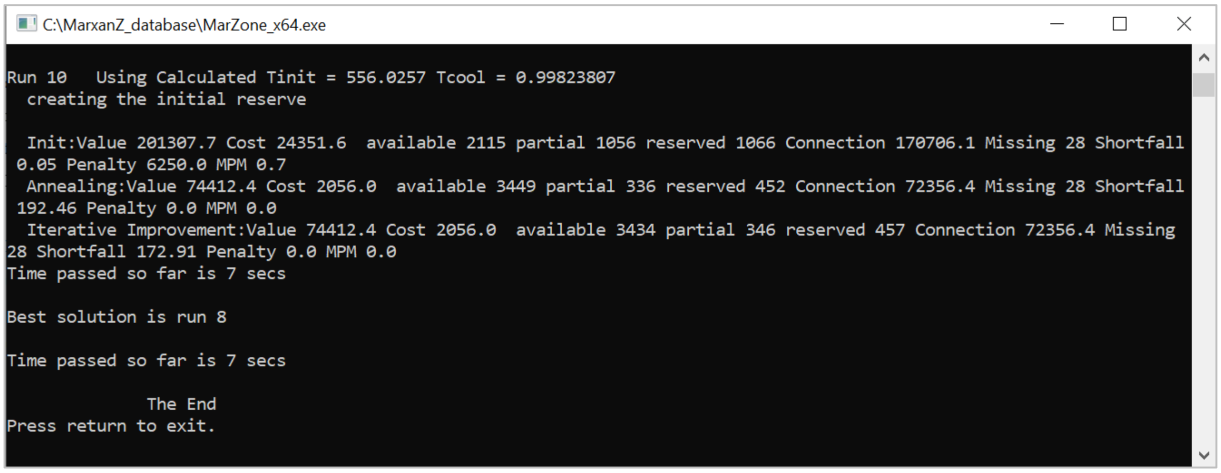
```
C:\MarxanZ_database\MarZone_x64.exe
        Marxan with Zones v 2.01

   Marine Reserve Design with Zoning and Annealing

   Marxan with Zones coded by Matthew Watts
   Written by Ian Ball, Hugh Possingham and Matthew Watts

   Based on Marxan coded by Ian Ball, modified by Matthew Watts
   Written by Ian Ball and Hugh Possingham

ian.ball@aad.gov.au
h.possingham@uq.edu.au
m.watts@uq.edu.au

   Marxan website

http://www.uq.edu.au/marxan

Run 1 Run 2 Run 3 Run 4 Run 5 Run 6 Run 7 Run 8 Run 9 Run 10
Time passed so far is 3 secs

              The End
Press return to exit.
```

Figure 24. *Example of the screen output with verbosity level 1 (Results Only)*



```
C:\MarxanZ_database\MarZone_x64.exe

Run 10   Using Calculated Tinit = 457.6000 Tcool = 0.99829929
  creating the initial reserve

  Init:Value 189378.9 Cost 24095.3  available 2172 partial 1039 reserved 1026 Connection 165283.6 Missing 26 Shortfall
 0.00 Penalty 0.0 MPM 1.0
  Annealing:Value 74366.3 Cost 2013.8  available 3517 partial 170 reserved 550 Connection 72352.5 Missing 28 Shortfall
 251.32 Penalty 0.0 MPM 0.0
  Iterative Improvement:Value 74365.6 Cost 2013.6  available 3493 partial 175 reserved 569 Connection 72352.1 Missing
28 Shortfall 205.10 Penalty 0.0 MPM 0.0
Time passed so far is 4 secs

Best solution is run 8

Time passed so far is 4 secs

              The End
Press return to exit.
```

Figure 25. *Example of the screen output with verbosity level 2 (General Progress)*

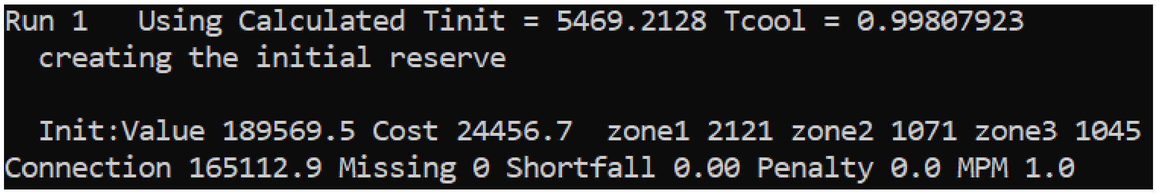


```
C:\MarxanZ_database\MarZone_x64.exe

Run 10   Using Calculated Tinit = 556.0257 Tcool = 0.99823807
  creating the initial reserve

  Init:Value 201307.7 Cost 24351.6  available 2115 partial 1056 reserved 1066 Connection 170706.1 Missing 28 Shortfall
 0.05 Penalty 6250.0 MPM 0.7
  Annealing:Value 74412.4 Cost 2056.0  available 3449 partial 336 reserved 452 Connection 72356.4 Missing 28 Shortfall
 192.46 Penalty 0.0 MPM 0.0
  Iterative Improvement:Value 74412.4 Cost 2056.0  available 3434 partial 346 reserved 457 Connection 72356.4 Missing
28 Shortfall 172.91 Penalty 0.0 MPM 0.0
Time passed so far is 7 secs

Best solution is run 8

Time passed so far is 7 secs

              The End
Press return to exit.
```

Figure 26. *Example of the screen output with verbosity level 3 (Detailed Progress)*

Box 11. Example of Summary Information on the Screen Output

The following is an example of a basic summary information line displayed on the screen (included in verbosity level 2 and 3):



```
Run 1   Using Calculated Tinit = 5469.2128 Tcool = 0.99807923
  creating the initial reserve

  Init:Value 189569.5 Cost 24456.7  zone1 2121 zone2 1071 zone3 1045
Connection 165112.9 Missing 0 Shortfall 0.00 Penalty 0.0 MPM 1.0
```

Run 1 indicates that this is the first run of the algorithm. As illustrated here, the run number may not appear on the same line as the valuation of the solution.

Value 189569.5 is the cost plus the boundary value plus the penalty for missing features. Because the penalty is included in this value, the meaning of the value requires interpretation.

Cost 24456.7 is the cost for the solution. It is the sum of the costs of all planning units in the solution in the units of the cost field in the pu.dat file.

Zone1, Zone2, and Zone3 corresponds to the number of planning units in the associated zone. All zones will be listed in the summary line as the zone name indicated in the zones.dat file. The planning units in the zones will sum to equal the total number of planning units in the solution (i.e., 2121 + 1071 + 1045 = 4,237 planning units).

Connection 165112.9 is the boundary cost (or connection strength) of the solution.

Missing 0 is the number of objectives which are under-represented in the solution. These objectives are the targets defined in the feat.dat and zonetarget.dat files. They are screened according to the 'MISSLEVEL' parameter in the input.dat file.

Shortfall 0 is the total target amount (based on zone targets and/or zone contributions) not achieved for all feature targets.

Penalty 0 is the penalty for not meeting representation targets (zone targets and/or zone contributions) for all the features. If features have met their targets, as is the case here, then it will be 0.0.

MPM 1 is the minimum proportion met (MPM) for the worst performing feature, which in this case is '1' based on the 'MISSLEVEL' parameter set in the input.dat file.

7.4 Output Files

Marxan with Zones can generate several different output files. The files generated will depend on what is specified in the 'Save Files' section of the Input Parameter File (input.dat). Table 24 list all the possible output files and identifies the corresponding variable name in the input.dat file. It also includes examples of file names. Where a number is included in the file name (e.g. output_r001.csv), this is the run number (or solution number) that generated that particular output.

Table 24. *Output file types and names*

Example File Name	Corresponding Variable Name in *input.dat*	Description
output_r001.csv	SAVERUN	Solutions for each run
output_best.csv	SAVEBEST	Best solution for all runs
output_mv001.csv	SAVETARGETMET	Missing value information for each run
output_mvbest.csv	SAVETARGETMET	Missing value information for the best run
output_sum.csv	SAVESUM	Summary information
output_sen.txt	SAVESCEN	Scenario details
output_ssoln.csv	SAVESUMSOLN	Summed solution
output_solutionsmatrix.cvs	SAVESOLUTIONSMATRIX	Planning units selected in for each run
output_penalty.csv	SAVEPENALTY	Penalty values for each feature for all runs?
output_log.txt	SAVELOG	Screen log file
output_ zoneconnectivitysum00001.csv output_ zoneconnectivitysumbest.csv	SAVEZONECONNECTIVITYSUM	Zone connectivity sum

The examples of file names in Table 24 all begin with the prefix 'output', which is the default value for the SCENNAME in the input.dat file. However, the file prefix can take on whatever name is specified by the user for variable SCENNAME. For example, if 'scenario1' is set for the SCENNAME variable, then all output files will be saved with the file name prefix 'scenario1' (e.g., scenario1_r001.csv).

The following subsections describe each of the output files in detail.

7.4.1 Solution for Each Run

Name in input.dat: SAVERUN

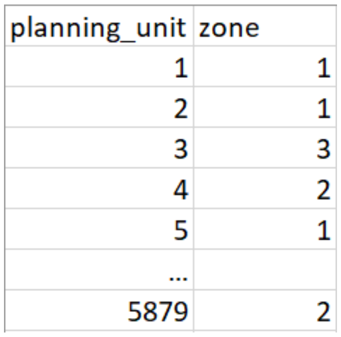
Example of output file name: output_**r001**.csv, where 'output' is the name supplied by the user in the SCENNAME parameter of the input.dat file.

We recommend saving this file as .csv (use code 3 in the input.dat file; see Section 5.3.1.9), a format that can be imported into a GIS environment and then joined to the planning unit shapefile for visualization of results.

Description: A Solution File is produced for each run of the algorithm, so each file corresponds to a unique run. For a given solution, the file identifies which planning units were selected in each zone. The run number associated with the file is indicated by the number included in the file name (e.g., output_r005 corresponds to run 5).

The file has two fields, 'planning_unit' and 'zone'. Each row has a planning unit identifier (in the 'planning_unit' field), followed by the zone identifier (in the 'zone' field) that identifies the zone that has been assigned to that planning unit.

An example of a portion of the Solution File is provided in Figure 27. In this example, there are 5879 planning units, all of which have either been assigned to Zone 1, 2, or 3 for Run 1.



planning_unit	zone
1	1
2	1
3	3
4	2
5	1
...	
5879	2

Figure 27. *An example of a subset of a solution file (e.g., output_r001.csv)*

7.4.2 Best Solution

Name in input.dat: SAVEBEST

Example of output file name: output_**best**.csv, where 'output' is the name supplied by the user in the input.dat file.

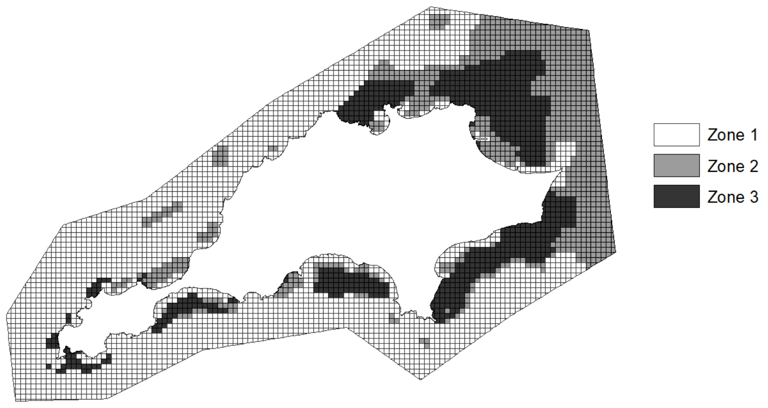
⚠ We recommend saving this file as .csv (use code 3 in the input.dat file; see Section 5.3.1.9), a format that can be imported into a GIS environment and then joined to the planning unit shapefile for visualization of results.

Description: The Best Solution File is the same as the Solution File described above (Section 7.4.1), but it corresponds to the run that produced the solution with the lowest score. It is important to remember that the best run is only superior regarding the objective function score, which may only be marginally better than the other solutions. It may not represent the best zoning system for implementation. Thus the 'best' has a very narrow meaning here and should not be communicated to stakeholders or decision-makers as the ideal solution. Rather, it should be viewed and presented as a very good solution, within a continuum of zoning options. More discussion on this topic can be found in the MGPH.

Box 12. How to Visualize the Solution of a Given Run in QGIS or ArcMap

Marxan with Zones does not generate mapped outputs. However, the solutions generated by the software (i.e., the solution for each run or the 'best' solution) can be visualized using a GIS software, such as QGIS or ArcGIS. The following instructions outline basic steps to visualize a solution in QGIS or ArcGIS.

- Step 1: Open ArcMap or QGIS and add the spatial file that contains your planning unit layer (e.g., a grid polygon shapefile).
- Step 2: Add the solution file (e.g., output_r001.csv) to your project and append it to your planning unit layer using the planning unit id field available in both files.
- Step 3: Export the planning unit layer with the new appended information to make the linkage permanent, then add the new layer to your project.
- Step 4: Visualize the solution by setting different colors for unique values in the 'zone' field, where each value represents a different zone.



7.4.3 Missing Values for Each Run

Name in input.dat: SAVETARGMET

Example of output file name: output_**mv001**.csv, where 'output' is the name supplied by the user in the input.dat file.

⚠ We recommend saving this file as .csv (use code 3 in the input.dat file, see Section 5.3.1.9), so it can be easily opened in Excel or a similar program.

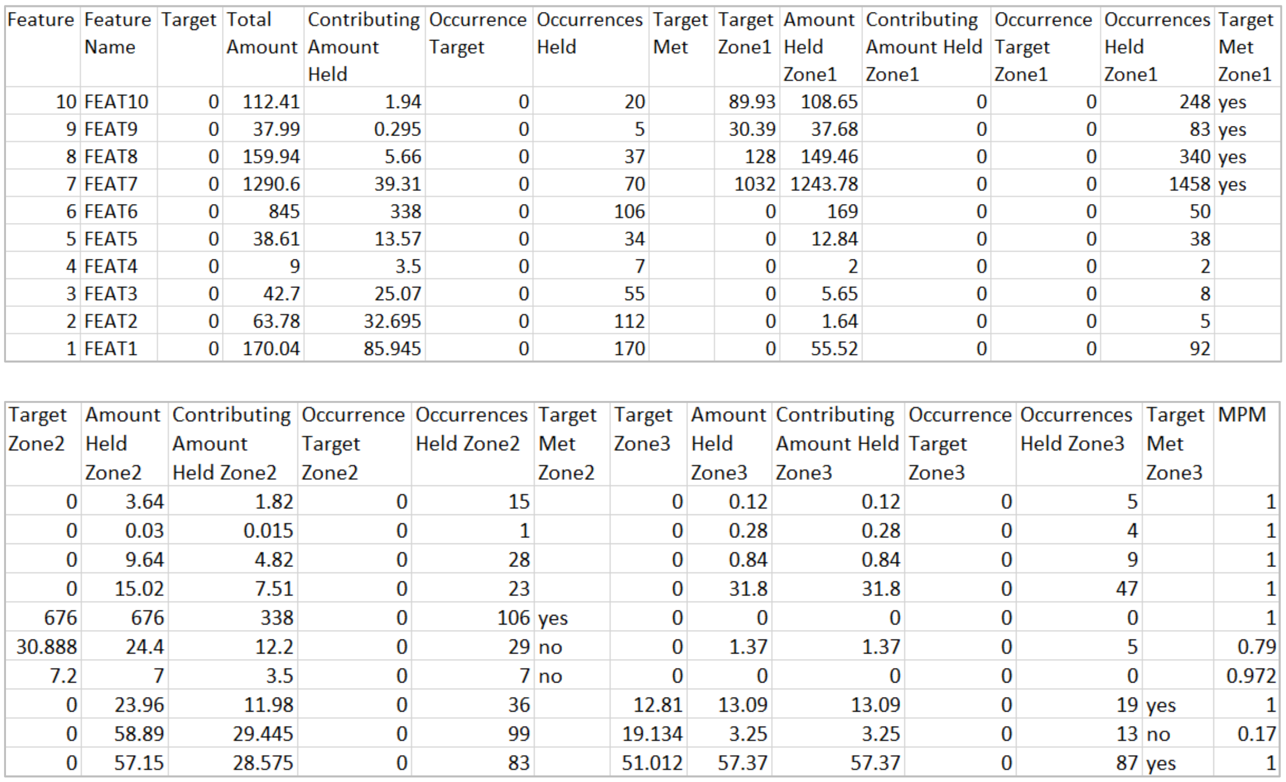
Description: This file contains information on how well the final solution from each run did in terms of meeting targets. A Missing Values File is produced for each run. The run number associated with the file is indicated by the number in the file name (e.g., output_mv005 corresponds to run 5). The Missing Values File includes the following fields: feature, feature name, target, total amount, contributing amount held, occurrence target, occurrences held, and target met (Table 25).

Table 25. *Description of Missing Value File Headers*

Header	Description
Feature	The unique identifier of the feature, indicated in feat.dat
Feature Name	The optional name for the feature, indicated in feat.dat
Target	The target amount for the feature, indicated in feat.dat
Total Amount	Total amount of the feature in the study region
Contributing Amount Held	Amount of the feature captured multiplied by the feature's contribution fraction for the zone it is captured in
Occurrence Target	The targeted number of occurrences for the feature
Occurrences Held	Number of occurrences of the feature captured
Target Met	An alphabetic variable that returns 'yes' if all the targets set for that feature are met, otherwise it returns 'no'
*Target Zone #**	The target amount for the feature in zone 1, indicated in zonetarget.dat
*Amount Held Zone #**	Amount of the feature captured in zone 1
*Contributing Amount Held Zone #**	Amount of the feature captured in zone 1 multiplied by the features contribution fraction in zone 1
*Occurrence Target Zone #**	The target number of occurrences for the feature in zone 1, indicated in zonetarget.dat
*Occurrences Held Zone #**	The number of occurrences of the feature captured in zone 1
*Target Met Zone #**	'Yes' if both targets above are met, otherwise 'no'
MPM	Minimum Proportion Met

* These field will be replicated for each zone. For Zone 1, the headers are Target Zone 1, Amount Help Zone 1, Contributing Amount Held Zone 1, Occurrence Target Zone 1, Occurrence Held Zone 1, and Target Met Zone 1.

An example of the missing values file is shown in Figure 28. In this example, overall targets were not set in the feat.dat file. Zone targets were specified for different features and zones in the zonetarget.dat file. For example, features 7 to 10 (i.e., FEAT7, FEAT8, FEAT9, FEAT10) had zone targets set for Zone 1, features 4 to 6 had zone targets for Zone 2, and features 1 to 3 has zone targets for Zone 3. As shown in the 'Target Met' field for Zone 1, all of the zone targets were met in this zone. The 'Target Met' field for the remaining zones show that targets were met for 2 of 3 features for Zone 3 and only 1 of 3 features for Zone 2. In the example, the Occurrence Target fields are all '0' values because occurrence targets were set in the feat.dat file. Zone contributions were set for Zone 2 and Zone 3, but not for Zone 1.



Feature	Feature Name	Target	Total Amount	Contributing Amount Held	Occurrence Target	Occurrences Held	Target Met	Target Zone1	Amount Held Zone1	Contributing Amount Held Zone1	Occurrence Target Zone1	Occurrences Held Zone1	Target Met Zone1
10	FEAT10	0	112.41	1.94	0	20		89.93	108.65	0	0	248	yes
9	FEAT9	0	37.99	0.295	0	5		30.39	37.68	0	0	83	yes
8	FEAT8	0	159.94	5.66	0	37		128	149.46	0	0	340	yes
7	FEAT7	0	1290.6	39.31	0	70		1032	1243.78	0	0	1458	yes
6	FEAT6	0	845	338	0	106		0	169	0	0	50	
5	FEAT5	0	38.61	13.57	0	34		0	12.84	0	0	38	
4	FEAT4	0	9	3.5	0	7		0	2	0	0	2	
3	FEAT3	0	42.7	25.07	0	55		0	5.65	0	0	8	
2	FEAT2	0	63.78	32.695	0	112		0	1.64	0	0	5	
1	FEAT1	0	170.04	85.945	0	170		0	55.52	0	0	92	

Target Zone2	Amount Held Zone2	Contributing Amount Held Zone2	Occurrence Target Zone2	Occurrences Held Zone2	Target Met Zone2	Target Zone3	Amount Held Zone3	Contributing Amount Held Zone3	Occurrence Target Zone3	Occurrences Held Zone3	Target Met Zone3	MPM
0	3.64	1.82	0	15		0	0.12	0.12	0	5		1
0	0.03	0.015	0	1		0	0.28	0.28	0	4		1
0	9.64	4.82	0	28		0	0.84	0.84	0	9		1
0	15.02	7.51	0	23		0	31.8	31.8	0	47		1
676	676	338	0	106	yes	0	0	0	0	0		1
30.888	24.4	12.2	0	29	no	0	1.37	1.37	0	5		0.79
7.2	7	3.5	0	7	no	0	0	0	0	0		0.972
0	23.96	11.98	0	36		12.81	13.09	13.09	0	19	yes	1
0	58.89	29.445	0	99		19.134	3.25	3.25	0	13	no	0.17
0	57.15	28.575	0	83		51.012	57.37	57.37	0	87	yes	1

Figure 28. *An example of a missing values file (e.g., output_mv001.csv), shown in two parts for demonstration purposes*

7.4.4 Missing Value Information for the Best Run

<u>Name in input.dat:</u> SAVETARGMET

<u>Example of output file name:</u> output_**mvbest**.csv, where 'output' is the name supplied by the user in the input.dat file.

⚠ We recommend saving this file as .csv (use code 3 in the input.dat file, see Section 5.3.1.9), so it can be easily opened in Excel or a similar program.

Description: This file is the same as the Missing Values File described above (Section 7.4.3) except it is for the 'best' solution (i.e., the solution with the lowest objective function score).

7.4.5 Summary Information

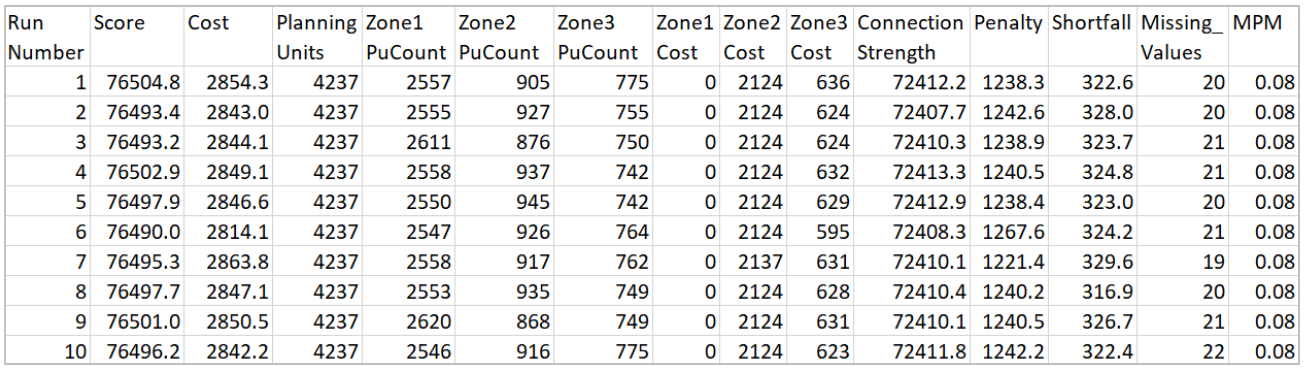
Name in input.dat: SAVESUM

Example of output file name: output_**sum**.csv, where 'output' is the name supplied by the user in the input.dat file.

⚠ We recommend saving this file as .csv (use code 3 in the input.dat file, see Section 5.3.1.9), so it can be easily opened in Excel or a similar program.

Description: This file contains summary information on each repeat run. It basically provides a report on how the solutions performed relative to the targets. The file contains multiple fields (Table 26). The last three fields (penalty, shortfall, and missing values) are particularly useful for examining solutions where some features have not met their target. Note that it is possible to have a high penalty and still be very close to the targets, particularly if the feature penalty factor for targets have been set very high. The shortfall is a good indication of whether the features are very close or very far from their targets. The number of missing values gives further information along this vein. If there are five features which each have missed their targets, but the combined shortfall is very small, then they could all be only narrowly missing their targets (e.g., ≥99%) and the user might not be particularly concerned.

An example of the Summary File is shown in Figure 29. In this example, there are three zones and ten repeat runs. Note that the PuCount and Cost fields are repeated for each zone.



Run Number	Score	Cost	Planning Units	Zone1 PuCount	Zone2 PuCount	Zone3 PuCount	Zone1 Cost	Zone2 Cost	Zone3 Cost	Connection Strength	Penalty	Shortfall	Missing_ Values	MPM
1	76504.8	2854.3	4237	2557	905	775	0	2124	636	72412.2	1238.3	322.6	20	0.08
2	76493.4	2843.0	4237	2555	927	755	0	2124	624	72407.7	1242.6	328.0	20	0.08
3	76493.2	2844.1	4237	2611	876	750	0	2124	624	72410.3	1238.9	323.7	21	0.08
4	76502.9	2849.1	4237	2558	937	742	0	2124	632	72413.3	1240.5	324.8	21	0.08
5	76497.9	2846.6	4237	2550	945	742	0	2124	629	72412.9	1238.4	323.0	20	0.08
6	76490.0	2814.1	4237	2547	926	764	0	2124	595	72408.3	1267.6	324.2	21	0.08
7	76495.3	2863.8	4237	2558	917	762	0	2137	631	72410.1	1221.4	329.6	19	0.08
8	76497.7	2847.1	4237	2553	935	749	0	2124	628	72410.4	1240.2	316.9	20	0.08
9	76501.0	2850.5	4237	2620	868	749	0	2124	631	72410.1	1240.5	326.7	21	0.08
10	76496.2	2842.2	4237	2546	916	775	0	2124	623	72411.8	1242.2	322.4	22	0.08

Figure 29. *Example of the summary output file (output_sum.dat)*

Table 26. Description of Summary File Headers

Header	Description
Run Number	The run number for that line.
Score	The objective function score for the solution from that run. This value is the sum of the Cost, the Connectivity Strength, and the Penalty. This field is useful to identify the run with the lowest score or the 'best solution'.
Cost	Total cost value of the run given by the cost or costs values associated with each planning unit.
Planning Units	Number of planning units in the planning area.
PuCount*	Number of planning units in the solution for each zone.
Cost*	The total cost of planning units for each zone.
Connectivity Strength	Boundary cost of the solution. This is the penalty applied when two planning units belonging to the same zone are not adjacent to each other. If the boundary length is not included or the BLM is set to '0', then this value would read '0'. Section 1.4 explains how this value is calculated.
Penalty	The penalty score for missing feature targets for the solution. If all features are adequately represented (that is, have met zone targets and/or zone contributions), then the penalty value will be '0'. This value gives an indication of the cost required to meet remaining targets. This is something that is not captured simply by looking at the shortfall. It is also used to rank the success of runs, looking only at those solutions that have a low penalty. Section 1.4 explains how this value is calculated.
Shortfall	The amount by which the targets (zone targets and zone contributions) for features have not met in the solution for that run. The shortfall reported here is the total shortfall summed across all conservation features. The shortfall is a good indication of whether missing features are close or far from their targets. If there are a number of features which have missed their target but the combined shortfall is very small, then the planner might not be too concerned.
Missing Values	The number of features that have not met their target in the final solution for that run and across all zones. This is screened according to the MISSLEVEL variable set in the Input Parameter file (see Section 5.3.1.11). If the missing level is set to '1', then every feature which falls below its target level is counted as missing. If the missing level is lower than '1' (e.g., 0.98), Marxan with Zones may not report a feature as missing even if the solution contains slightly less than the target amount.
MPM	The Minimum Proportion Met (MPM) for the worst performing feature. This value corresponds to the lowest MPM value in the missing value file for that run number.

*Field repeated for each zone.

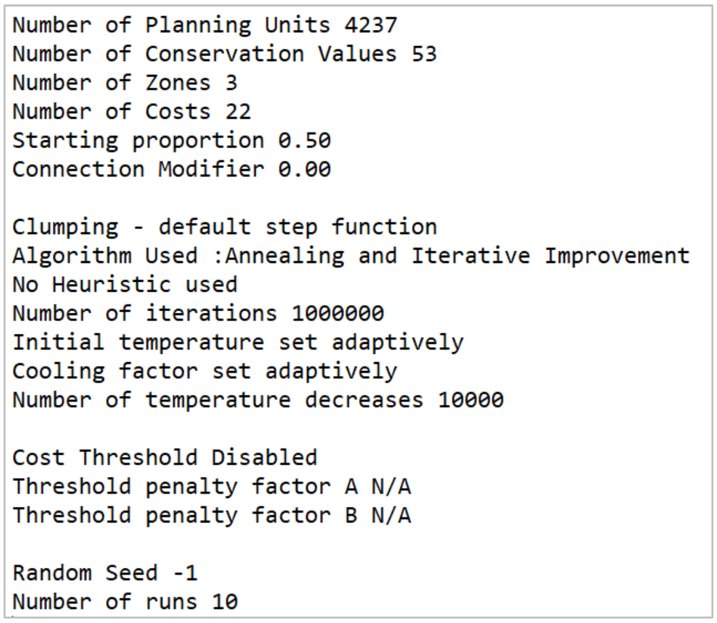
7.4.6 Scenario Details

Name in input.dat: SAVESCEN

Name of output file: output**_sen.dat**, where 'output' is the name supplied by the user in the input parameter file.

⚠ This file can be saved as a .dat file (use code 1 in the input.dat file, see Section 5.3.1.9). Files with a .dat extension can be opened with Notepad or similar programs.

Description: This file documents the major parameter values for that scenario. This file is useful to keep track of the parameters that produced certain results, especially when multiple scenarios are run. The information can help determine appropriate values for commonly modified parameters. An example of the file is shown in Figure 30.



```
Number of Planning Units 4237
Number of Conservation Values 53
Number of Zones 3
Number of Costs 22
Starting proportion 0.50
Connection Modifier 0.00

Clumping - default step function
Algorithm Used :Annealing and Iterative Improvement
No Heuristic used
Number of iterations 1000000
Initial temperature set adaptively
Cooling factor set adaptively
Number of temperature decreases 10000

Cost Threshold Disabled
Threshold penalty factor A N/A
Threshold penalty factor B N/A

Random Seed -1
Number of runs 10
```

Figure 30. *Example of the scenario details output file (output_sen.dat)*

7.4.7 Summed Solution

Name in input.dat: SAVESUMSOLN

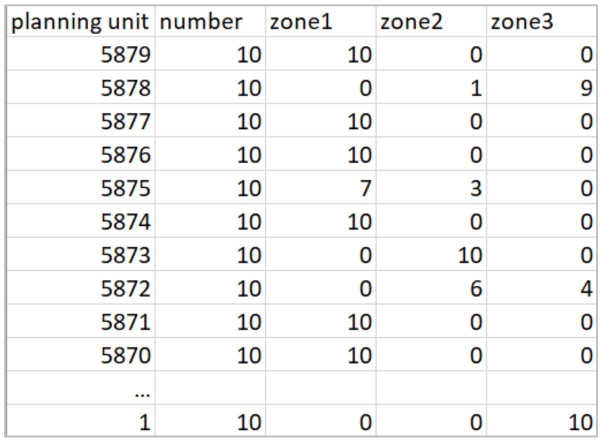
Example of output file name: output**_ssoln**.csv, where 'output' is the name supplied by the user in the input.dat file.

⚠ We recommend to save this file as .csv (use code 3 in the input.dat file, see Section 5.3.1.9) to it can be easily imported to a GIS environment for its visualization.

Description: This file is the summed solution of all individual runs in a scenario. It indicates how often a planning unit was included in an individual zone. It is a useful way to explore the irreplaceability of planning units in a zone.

The file has a 'planning unit' and a 'number' field, as well as an additional field for each zone. Each line has a planning unit identifier (in the planning unit field), followed by the number of runs (in the 'number' field), and the number of times that planning unit was assigned to an individual zone (e.g., Zone 1).

An example of a portion of this file is shown in Figure 31. In this example, the planning unit identifier 5875 was assigned to Zone 1 in 7 out of 10 runs, in Zone 2 in 3 out of 10 runs, and never selected as part of Zone 3.



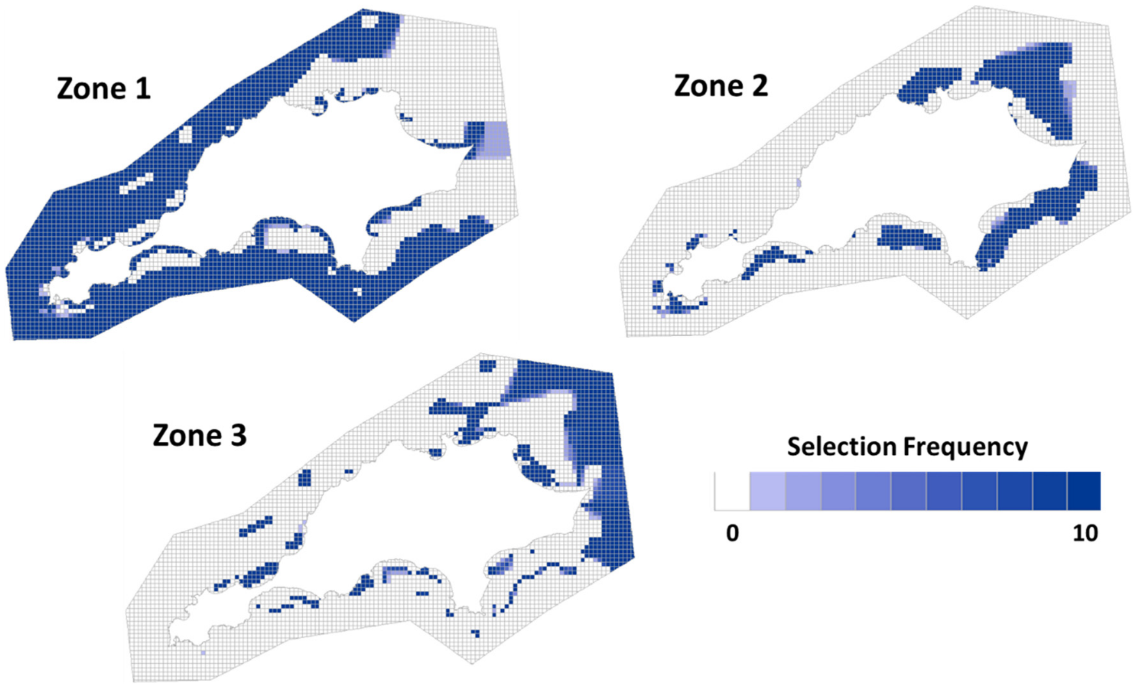
planning unit	number	zone1	zone2	zone3
5879	10	10	0	0
5878	10	0	1	9
5877	10	10	0	0
5876	10	10	0	0
5875	10	7	3	0
5874	10	10	0	0
5873	10	0	10	0
5872	10	0	6	4
5871	10	10	0	0
5870	10	10	0	0
...				
1	10	0	0	10

Figure 31. *Example of a subset of the summed solution output file (output_ssoln.csv)*

Box 13. How to visualize the summed solution in QGIS or ArcMap

Marxan with Zones does not provide a map version of the Summed Solution File (i.e., the selection frequency), although the file can be mapped and visualized externally using a GIS software, like QGIS or ArcGIS. The following instructions outline the basic steps to visualize the selection frequency in QGIS or ArcGIS.

- Step 1: Open ArcMap or QGIS and add the spatial file that contains your planning unit layer (e.g., a grid polygon shapefile).
- Step 2: Add the Summed Solution File (e.g., output_ssoln.csv) to your project and append it to your planning unit layer using the planning unit identifier field available in both files.
- Step 3: Export the planning unit layer with the new appended information to make the linkage permanent, then add the new layer to your project.
- Step 4: The selection frequency for each zone should be mapped and visualized separately. Visualize the selection frequency for a given zone by using graduate colors and displaying the values in the 'zone#' field. It is recommended to visualize '0' values in their own class.



7.4.8 Solution Matrix

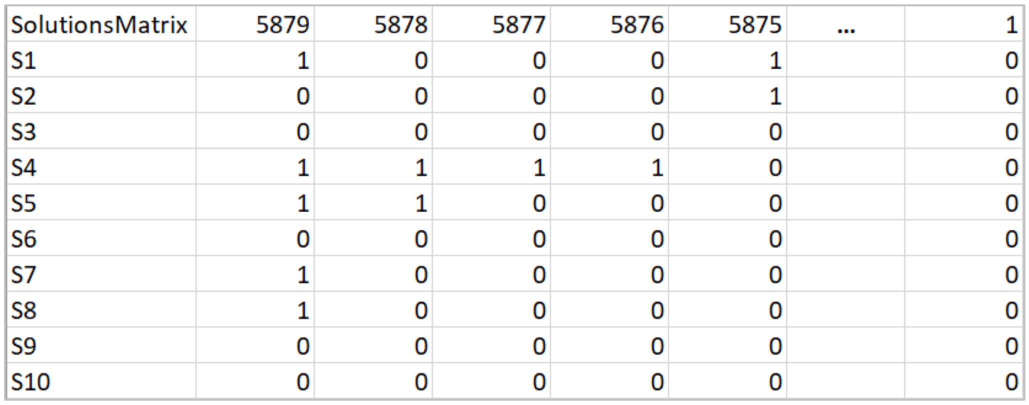
Name in input.dat: SAVESOLUTIONSMATRIX

Example of output file name: output_**solutionsmatrix_zone1**.csv, where 'output' is the name supplied by the user in the input.dat file.

We recommend saving this file as .csv (use code 3 in the input.dat file, see Section 5.3.1.9) so it can easily be opened in Excel or a similar program.

Description: This file will be generated for each zone (e.g., if there are three zones, three solution matrix files will appear in the output folder, one for each zone). For a given zone, the file records which planning units were selected in that zone for all runs. This file is a useful way to export information on planning unit selection for cluster analysis in R or other statistical software packages.

If the SOLUTIONSMATRIXHEADERS variable in the input.dat file has a value of '1', a header row is included in the file. If you choose to include the header row, a list of the planning unit identifiers will be added in the first row. Subsequent rows will include information for each solution produced in the analysis. Each matrix cell corresponds to a planning unit and solution. Matrix cells with a value of '1' indicate that the planning unit in that solution was assigned to the zone for the file, and vice-versa for '0' values. An example of this file is presented in Figure 32.



SolutionsMatrix	5879	5878	5877	5876	5875	...	1
S1	1	0	0	0	1		0
S2	0	0	0	0	1		0
S3	0	0	0	0	0		0
S4	1	1	1	1	0		0
S5	1	1	0	0	0		0
S6	0	0	0	0	0		0
S7	1	0	0	0	0		0
S8	1	0	0	0	0		0
S9	0	0	0	0	0		0
S10	0	0	0	0	0		0

Figure 32. *Example of a subset of the solution matrix output file (output_solutionmatrix_zone1.csv)*

7.4.9 Penalty File

Name in input.dat: SAVEPENALTY

Name of output file: output_**penalty**.dat, where 'output' is the name supplied by the user in the input.dat file.

Description: This file contains the penalty computed by Marxan with Zones for all features in an individual run in a scenario. It indicates how expensive it was to satisfy the targets for a feature. It is useful for understanding the relative difficulty Marxan with Zones has in meeting the targets for features.

7.4.10 Screen Log File

Name in input.dat: SAVELOG

Example of output file name: output_**log**.dat, where 'output' is the name supplied by the user in the input.dat file.

This file can be saved as .dat file (use code 1 in the input.dat file, see Section 5.3.1.9). Files with .dat extension can be opened with Notepad or similar software.

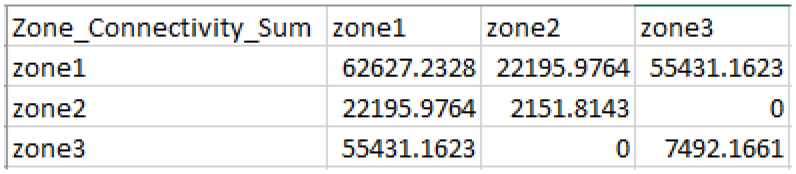
Description: This text file contains exactly what was displayed as the screen output while running Marxan with Zones (see Section 7.3). This can be useful in de-bugging, or if for instance, you want to go back through the runs to investigate how much work is being done during the simulated annealing phase relative to iterative improvement.

7.4.11 Zone Connectivity Sum

Name in input.dat: SAVEZONECONNECTIVITYSUM

Example of output file name: output_**zoneconnectivitysum00001**.csv and output_**zoneconnectivitysumbest**.csv, where 'output' is the name supplied by the user in the input parameter file.

Description: Connectivity value between zones for each solution and for the best solution. An example of this file is presented in Figure 33.



Zone_Connectivity_Sum	zone1	zone2	zone3
zone1	62627.2328	22195.9764	55431.1623
zone2	22195.9764	2151.8143	0
zone3	55431.1623	0	7492.1661

Figure 33. Example of a subset of the solution matrix output file (output_solutionmatrix_zone1.csv)

8 Good Practices

8.1 Overview

This manual should provide you with all the information you need to complete a Marxan with Zones analysis. Ensuring that your analyses are robust and defensible is beyond the scope of this document. In this section, we mention some of the things you must be prepared to undertake in order to ensure that Marxan with Zones delivers quality, defendable outputs. We also provide advice for communicating Marxan with Zones results to stakeholders and decision makers. As previously stated, we strongly encourage you to read the MGPH and relevant literature before undertaking Marxan with Zones analyses.

8.2 Experimentation

The most important thing to remember is that Marxan with Zones does not provide 'one-stop' zoning solutions. As outlined in this manual (see Section 3.3), many of the parameters will require experimentation before you can expect Marxan with Zones to deliver reasonable solutions (e.g., what FPF value, if any, can ensure that all feature targets are meet). Each parameter should ideally be set in a stepwise and systematic manner. However, this can be challenging as parameters are not independent of each other. Parameters can often interact in unexpected ways. Similarly, changes to your input data can mean that the parameter values should be re-calibrated.

Various methods have been suggested to help select appropriate Marxan with Zones parameter values. However, there is no substitute for simply exploring as many scenarios as your project time and budget permit and ensuring that there is adequate time allocated for these experimentations. Understanding your data and problem construction well will help you gain expertise and familiarity with how Marxan software works.

8.3 Visual Inspection

Although visual display is perhaps the most basic and often subjective of post-processing procedures, the power of the human eye to see visual trends should not be underrated and can detect issues that can be missed with sophisticated spatial statistics. Available data are not always ideal and there are many subtleties of spatial planning that cannot be incorporated into Marxan with Zones.

Knowledge of your planning region will help avoid obvious errors in zoning systems. The knowledge of any problems should be used to update future scenarios being run in Marxan with Zones. Simply modifying a solution at the end of an analysis is likely to lead to both inadequacies and inefficiencies in the solution.

To help the visual inspection process, it can be useful to visually compare the solutions with your data layers. For instance, you may notice that solutions are primarily driven by the distribution of costs rather than features, or that the distribution of only a few features largely explains the shape of the solutions. While these are not necessarily problems, they are very good to know, and can influence the next iteration of selecting parameter values.

8.4 Sensitivity Analyses

Even if you are happy with the quality of the zoning system solutions derived via Marxan with Zones, it is important to consider how the solutions would change if some of the scenario details changed. If for example, small changes in your cost data lead to large changes in the optimal zoning system, then you would want to ensure that your use of a particular cost structure is well justified.

The robustness of your solutions to small changes in the scenario details should ideally be explored through formal sensitivity analysis where the results of modifying parameters, constraints and data are compared both qualitatively and quantitatively. That said, due to the large number of variables and features in any given analysis, most sensitivity analyses cannot look at everything, and therefore only what are considered key attributes are examined.

Sensitivity analyses should include scenario details not commonly subject to experimentation, such as the size and shape of planning units, the targets, extent of the planning region, and different types of feature and cost data. Determining if the output from different scenarios is similar will help assess the sensitivity of your solutions. Reporting the sensitivity of solutions against different factors can be very useful to highlight the impact of social or political constraints on solutions, or to help direct investment in the collection of data. See the MGPH for more information on sensitivity analyses.

8.5 Communicating Results

Communicating Marxan with Zones results to a wide range of audiences is often a challenging and laborious process, but it is essential to move from plans on paper to on-the-ground actions. The following recommendations for presenting and communicating Marxan with Zones results can facilitate this process.

Ensure that the audience has a basic understanding of Marxan with Zones prior to presenting results: The aim of introducing and explaining Marxan with Zones to stakeholders is not to oversell or undersell Marxan, but to acknowledge its advantages and disadvantages, along with its role as a decision support tool. When stakeholders are indirectly involved with Marxan with Zones, it may not be necessary to mention it or to describe specific aspects of the tool. In other cases, particularly when stakeholders are more directly involved in the analysis or in evaluating outputs, they must clearly understand to some extent how Marxan with Zones works.

Tailor presentation and communication strategies to the audience: Stakeholders will be particularly attuned to the Marxan with Zones results in relation to their own interests. For instance, scientific experts may want to know details on data inputs, assumptions, and targets relating to their area of expertise. Stakeholders from different sectors may be less interested in these details. Rather, they will want to know how their specific interests and concerns were accounted for in planning and how the proposed zoning plans could impact their interests. Thus, communication should be audience driven. Outputs should be presented and explained in language that is meaningful to the audience. It is equally important to ask for feedback that may identify additional data sources or considerations to include in another iteration or revision.

Present multiple solutions: Practitioners should consider presenting more than one spatial output or solution to a zoning problem. This will allow the audience to compare several zoning options to address inherent concerns while meeting objectives.

Present maps of individual solutions together with selection frequency maps: Maps showing the selection frequency (summed solutions) and individual solutions should be presented side by side. This can help clarify differences between the two types of outputs and can help avoid misinterpretations.

Present appropriate context: When presenting results, make sure to explain the meaning behind the maps. Stakeholders will want to know how the results reflect their own interests and compare this to how other stakeholders or other regions are affected. When communicating with maps to a broader audience, be aware that you can lose control of the communication process. In other words, maps can speak for themselves and are powerful communication tools in and of themselves.

Prepare maps with embedded information, disclaimers, and appropriate symbology: When mapping Marxan with Zones outputs, include information on the analysis settings, such as the number of features, feature targets, zone targets. Maps dissociated from sufficient information relevant to the generation of the outputs displayed can be misleading. As with any map making, basic cartographic rules should be followed. Colour gradients can be used to display outputs, but it is important to be mindful of colour schemes for audience members who may be colour blind or who only have access to grey-scale printers. Remember, that much can be interpreted and misinterpreted from the colours, symbols and other cartographic characteristics of maps. For example, solid lines on a map can invoke "lines drawn in the sand", whereas dotted lines or faded boundaries can relay areas that are open for discussion. Finally, "work in progress" signs or "draft" watermarks embedded in maps are very important in order to avoid outputs being misinterpreted as a 'final result' that might cause stakeholders to react negatively if it looks like their areas of interest will be heavily impacted.

<u>Choose the right person to communicate results</u>: In many cases, scientists or government staff have been responsible for communicating with stakeholder groups. However, they may not necessarily be the most appropriate for communicating Marxan with Zones results. The right person to communicate results is someone who has a good relationship (i.e., trust) with stakeholders; a fair understanding of stakeholder values, interests, and needs; and a proven ability to communicate information in non-technical terminology. Hence, the right person may vary based on the stakeholder groups present.

9 Courses & Training

In addition to this manual, there are courses and trainings available on using Marxan, Marxan with Zones, and supporting software (e.g., ArcGIS, QGIS). Online tutorials on using Marxan and certain supporting software are available at https://marxansolutions.org.

Courses and trainings are offered though different institutions and organizations, such as the Pacific Marine Analysis and Research Association (PACMARA).

PacMARA is a charitable organization of science and planning professionals dedicated to building and increasing capacity in marine and coastal planning. It offers technical and non-technical courses on Marxan, Marxan with Zones, and other Marxan extensions (e.g., Marxan Connect), along with courses on related concepts and topics (e.g., marine spatial planning). PacMARA also provides facilitation, analysis, and support for managers, policymakers, academics, and other members of the conservation community. Visit https://pacmara.org to enquire about support service and upcoming courses/trainings.

References

Marxan with Zones Software

Watts, M. E., Ball, I. R., Stewart, R. S., Klein, C. J., Wilson, K., Steinback, C., Lourival, R., Kircher, L., Possingham, H. P. (2009). Marxan with Zones: Software for optimal conservation based land- and sea-use zoning. Environmental Modelling and Software, 24(12), 1513-1521.

Marxan with Zones Manual

Serra N, Kockel A, Williams, B., Watts, M., Klein, C., Stewart, R., Ball, I., Game, E., Possingham, H., & McGowan J. (2021). Marxan with Zones User Manual. For Marxan with Zones version 1.0.1 and above. The Nature Conservancy (TNC), Arlington, Virginia, United States and Pacific Marine Analysis and Research Association (PacMARA), Victoria, British Columbia, Canada.

Marxan Good Practice Handbook (MGPH)

Ardron, J.A., Possingham, H.P., and Klein, C.J. (eds). 2010. Marxan Good Practices Handbook, Version 2. Pacific Marine Analysis and Research Association, Victoria, BC, Canada. 165 pages. www.pacmara.org.

Channel Island Example

Airame, S. 2005. Channel Islands National Marine Sanctuary: Advancing the Science and Policy of Marine Protected Areas. Pages 91-124 in A. Scholz and D. Wright, editors. Place Matters: Geospatial Tools for Marine Science, Conservation, and Management in the Pacific Northwest. Oregon State University Press, Corvallis.

References Cited in Manual

Peer reviewed literature and reports using Marxan are compiled at The University of Queensland's Ecology Centre. An updated list can be obtained from the Marxan website: http://www.uq.edu.au/marxan/index.html?page=80365&p=1.1.6.3

Adams, V. M., Pressey, R. L., & Álvarez-Romero, J. G. (2016). Using optimal land-use scenarios to assess trade-offs between conservation, development, and social values. *PLoS ONE*, *11*(6), 1–20. https://doi.org/10.1371/journal.pone.0158350

Agostini, V. N., Margles, S. W., Schill, S. R., Knowles, J. E., & Blyther, R. J. (2010). Marine Zoning in Saint Kitts and Nevis A Path Towards Sustainable Management of Marine Resources. In *The Nature Conservancy*. https://doi.org/10.1016/j.ocecoaman.2014.11.003

Ball, I. R., Possingham, H. P., & Watts, M. E. (2009). Marxan and relatives: Software for spatial conservation prioritization. In A. Moilanen, K. A. Wilson, & H. P. Possingham (Eds.), *Spatial conservation prioritisation: Quantitative methods and computational tools* (pp. 185–210). Oxford University Press.

Connolly, D. J. (1990) "an improved annealing scheme for QAP", European Journal of Operations Research, 46, 93-100.

Domisch, S., Kakouei, K., Martínez-López, J., Bagstad, K. J., Magrach, A., Balbi, S., Villa, F., Funk, A., Hein, T., Borgwardt, F., Hermoso, V., Jähnig, S. C., & Langhans, S. D. (2019). Social equity shapes zone-selection: Balancing aquatic biodiversity conservation and ecosystem services delivery in the transboundaryDanube River Basin. *Science of the Total Environment*, *656*, 797–807. https://doi.org/10.1016/j.scitotenv.2018.11.348

Ehler, C., & Fanny, D. (2009). *Marine spatial planning: a step-by-step approach towards ecosystem-based management*.

Fastré, C., Possingham, H. P., Strubbe, D., & Matthysen, E. (2020). Identifying trade-offs between biodiversity conservation and ecosystem services delivery for land-use decisions. *Scientific Reports*, *10*(1), 1–12. https://doi.org/10.1038/s41598-020-64668-z

Grantham, H. S., Agostini, V. N., Wilson, J., Mangubhai, S., Hidayat, N., Muljadi, A., Muhajir, Rotinsulu, C., Mongdong, M., Beck, M. W., & Possingham, H. P. (2013). A comparison of zoning analyses to inform the planning of a marine protected area network in Raja Ampat, Indonesia. *Marine Policy*, *38*, 184–194. https://doi.org/10.1016/j.marpol.2012.05.035

Gurney, G. G., Pressey, R. L., Ban, N. C., Álvarez-Romero, J. G., Jupiter, S., & Adams, V. M. (2015). Efficient and equitable design of marine protected areas in Fiji through inclusion of stakeholder-specific objectives in conservation planning. *Conservation Biology*, *29*(5), 1378–1389. https://doi.org/10.1111/cobi.12514

Hermoso, V., Cattarino, L., Kennard, M. J., Watts, M., & Linke, S. (2015). Catchment zoning for freshwater conservation: Refining plans to enhance action on the ground. *Journal of Applied Ecology*, *52*(4), 940–949. https://doi.org/10.1111/1365-2664.12454

JANIS (1997) Nationally agreed criteria for the establishment of a comprehensive, adequate and representative reserve system for forests in Australia. A report by the Joint ANZECC/MCFFA National Forest Policy Statement Implementation Subcommittee. National forest conservation reserves: Commonwealth proposed criteria. Commonwealth of Australia, Canberra.

Jumin, R., Binson, A., McGowan, J., Magupin, S., Beger, M., Brown, C. J., Possingham, H. P., & Klein, C. (2018). From Marxan to management: Ocean zoning with stakeholders for Tun Mustapha Park in Sabah, Malaysia. *Oryx*, *52*(4), 775–786. https://doi.org/10.1017/S0030605316001514

Klein, C. J., Steinback, C., Watts, M., Scholz, A. J., & Possingham, H. P. (2010). Spatial marine zoning for fisheries and conservation. *Frontiers in Ecology and the Environment*, *8*(7), 349–353. https://doi.org/10.1890/090047

Kockel, A., Ban, N. C., Costa, M., & Dearden, P. (2019). Evaluating approaches for scaling-up community-based marine-protected areas into socially equitable and ecologically representative networks. *Conservation Biology*, *00*(0), 1–11. https://doi.org/10.1111/cobi.13368

Kockel, A., Ban, N. C., Costa, M., & Dearden, P. (2020). Addressing distribution equity in spatial conservation prioritization for small-scale fisheries. *PloS One*, *15*(5), e0233339. https://doi.org/10.1371/journal.pone.0233339

Law, E. A., Bryan, B. A., Meijaard, E., Mallawaarachchi, T., Struebig, M. J., Watts, M. E., & Wilson, K. A. (2017). Mixed policies give more options in multifunctional tropical forest landscapes. *Journal of Applied Ecology*, *54*(1), 51–60. https://doi.org/10.1111/1365-2664.12666

Makino A, Klein CJ, Beger M, Jupiter SD, Possingham HP (2013) Incorporating Conservation Zone Effectiveness for Protecting Biodiversity in Marine Planning. PLoS ONE 8(11): e78986. doi:10.1371/journal.pone.0078986

Mazor, T., Possingham, H. P., Edelist, D., Brokovich, E., & Kark, S. (2014). The crowded sea: Incorporating multiple marine activities in conservation plans can significantly alter spatial priorities. *PLoS ONE*, *9*(8). https://doi.org/10.1371/journal.pone.0104489

Mills, M., S. D. Jupiter, R. L. Pressey, N. C. Ban, and J. Comley. 2011. Incorporating Effectiveness of Community-Based Management in a National Gap Analysis for Fiji. Conservation Biology, 26,1155-1164.

Parker, S. R., Truscott, J., Harpur, C., & Murphy, S. D. (2015). Exploring a Resilience-Based Approach to Spatial Planning in Fathom Five National Marine Park, Lake Huron, Canada, Using Marxan with Zones. *Natural Areas Journal*, *35*(3), 452–464. https://doi.org/10.3375/043.035.0308

Pressey, R. L., & Bottrill, M. C. (2009). Approaches to landscape- and seascape-scale conservation planning: convergence, contrasts and challenges. *Oryx*, *43*(4), 464. https://doi.org/10.1017/S0030605309990500

Reyers, B., O'Farrell, P. J., Nel, J. L., & Wilson, K. (2012). Expanding the conservation toolbox: Conservation planning of multifunctional landscapes. *Landscape Ecology*, *27*(8), 1121–1134. https://doi.org/10.1007/s10980-012-9761-0

Ruiz-Frau, A., Kaiser, M. J., Edwards-Jones, G., Klein, C. J., Segan, D., & Possingham, H. P. (2015). Balancing extractive and non-extractive uses in marine conservation plans. *Marine Policy*, *52*, 11–18. https://doi.org/10.1016/j.marpol.2014.10.017

Watts, M.E., Steinback, C., Klein, C. (2008). User Guide: Applying Marxan with Zones. North Central Coast of California Marine Study. University of Queensland and Ecotrust.

Watts, M. E., Ball, I. R., Stewart, R. S., Klein, C. J., Wilson, K., Steinback, C., Lourival, R., Kircher, L., & Possingham, H. P. (2009). Marxan with Zones: Software for optimal conservation based land- and sea-use zoning. *Environmental Modelling and Software*, *24*(12), 1513–1521. https://doi.org/10.1016/j.envsoft.2009.06.005

Watts, W., Steward, R., Segan, D., Kircher, L. 2011. Using the Zonae Cogito Decision Support System. Applied Environmental Decision Analysis Centre. The Ecology Centre. University of Queensland

Weeks, R., Russ, G. R., Bucol, A. A., & Alcala, A. C. (2010). Incorporating local tenure in the systematic design of marine protected area networks. *Conservation Letters*, *3*(6), 445–453. https://doi.org/10.1111/j.1755-263X.2010.00131.x

Yates, K. L., Schoeman, D. S., & Klein, C. J. (2015). Ocean zoning for conservation, fisheries and marine renewable energy: Assessing trade-offs and co-location opportunities. *Journal of Environmental Management*, *152*, 201–209. https://doi.org/10.1016/j.jenvman.2015.01.045

Appendix A- Troubleshooting

This appendix includes examples of common error messages encountered while running Marxan with Zones, and a description of the possible causes and solutions.

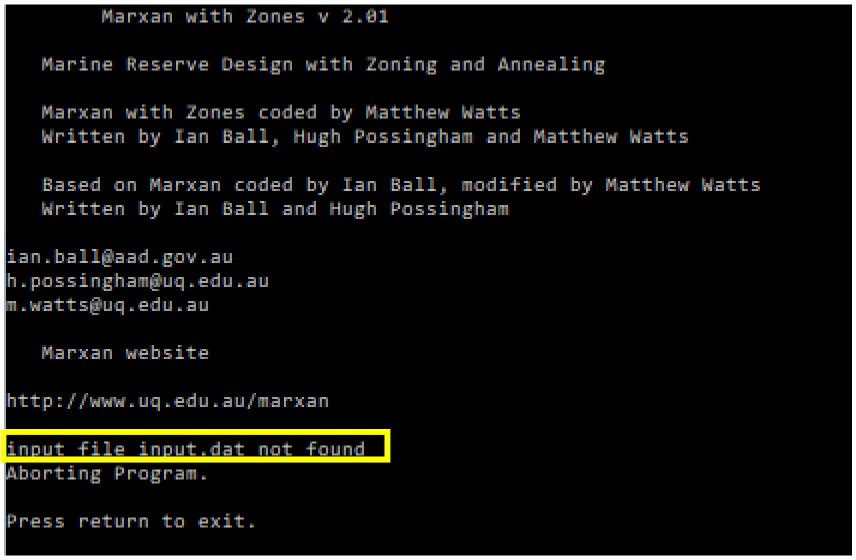
Please note that this list is not exhaustive. If you encounter issues or error messages not listed here, please contact us by sending an e-mail to marxancloud@gmail.com

A-1 Invalid input file (input.dat)

A-1.1 "Input file input.dat not found"

Error Message: The last thing Marxan with Zones shows is *"input file input.dat not found*".

Cause of Error Message: This occurs when you don't have an input.dat file containing the Marxan settings in the same folder as your Marxan with Zones executable file (e.g. MarxanZone_x64.exe).



```
        Marxan with Zones v 2.01

   Marine Reserve Design with Zoning and Annealing

   Marxan with Zones coded by Matthew Watts
   Written by Ian Ball, Hugh Possingham and Matthew Watts

   Based on Marxan coded by Ian Ball, modified by Matthew Watts
   Written by Ian Ball and Hugh Possingham

ian.ball@aad.gov.au
h.possingham@uq.edu.au
m.watts@uq.edu.au

   Marxan website

http://www.uq.edu.au/marxan

input file input.dat not found
Aborting Program.

Press return to exit.
```

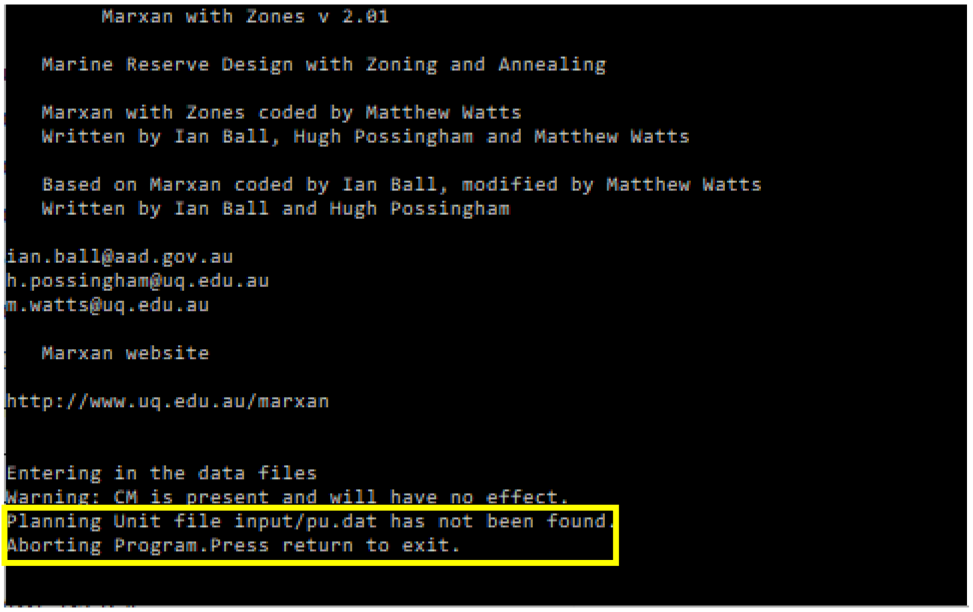
A-1.2 "Entering in the data files; Planning Unit files input/pu.dat has not been found"

Error Message: The last thing Marxan with Zones shows is "*Entering in the data files; Planning Unit files input/pu.dat has not been found; Aborting Program. Press return to exit*".

Cause of Error Message: This error message can be due to different causes.

- This occurs when the folder containing the input files (e.g. 'Input' folder) and specified in input.dat via INPUTDIR is not included or does not match the actual folder name (or directory) on your computer (e.g. the input.dat file contains the text "input" and your computer has a folder named "input_scenario").

1. This occurs when your planning unit file name pu.dat specified in input.dat via PUNAME is not included or does not match the actual planning unit file name on your computer (e.g. the input.dat file contains the text "pu.dat" and your computer has a file named "planningunit.dat").



```
        Marxan with Zones v 2.01

   Marine Reserve Design with Zoning and Annealing

   Marxan with Zones coded by Matthew Watts
   Written by Ian Ball, Hugh Possingham and Matthew Watts

   Based on Marxan coded by Ian Ball, modified by Matthew Watts
   Written by Ian Ball and Hugh Possingham

ian.ball@aad.gov.au
h.possingham@uq.edu.au
m.watts@uq.edu.au

   Marxan website

http://www.uq.edu.au/marxan


Entering in the data files
Warning: CM is present and will have no effect.
Planning Unit file input/pu.dat has not been found.
Aborting Program.Press return to exit.
```

A-1.3 "Missing planning unit id for line 1"

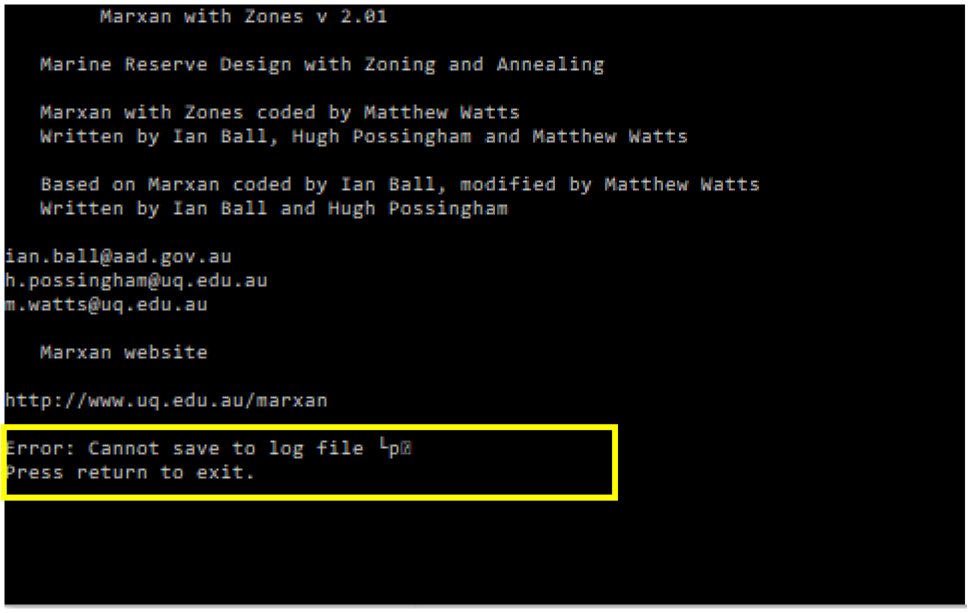
Error Message: The last thing Marxan shows is "*Error: Missing planning unit id for line 1"*

Cause of Error Message: This occurs when the heading containing the planning unit unique identifier is different than "id".

A-1.4 "Error: Cannot save to log file"

Error Message: The last thing Marxan with Zones shows is "*Error: Cannot save to log file…*".

Cause of Error Message: This occurs when the folder for the output files (e.g. 'Output' folder) and specified in input.dat via OUTPUTDIR is not included or does not match the actual folder name (or directory) on your computer (e.g. the input.dat file contains the text "input" and your computer has a folder named "scenario_outputs").

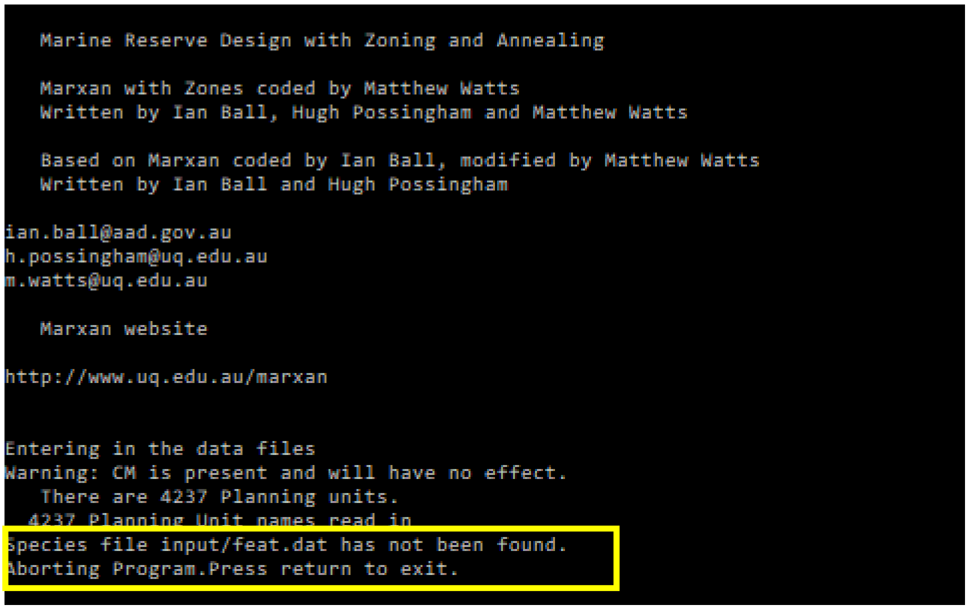


```
       Marxan with Zones v 2.01

   Marine Reserve Design with Zoning and Annealing

   Marxan with Zones coded by Matthew Watts
   Written by Ian Ball, Hugh Possingham and Matthew Watts

   Based on Marxan coded by Ian Ball, modified by Matthew Watts
   Written by Ian Ball and Hugh Possingham

ian.ball@aad.gov.au
h.possingham@uq.edu.au
m.watts@uq.edu.au

   Marxan website

http://www.uq.edu.au/marxan

Error: Cannot save to log file └p
Press return to exit.
```

A-1.4 "Species file input/feat.dat has not been found"

Error Message: The last thing Marxan with Zones shows is "*Species file input/feat.dat has not been found. Aborting Program. Press return to exit*".

Cause of Error Message: This occurs when your species file name spec.dat specified in input.dat via SPECNAME is not included or does not match the actual species file name on your computer (e.g. the input.dat file contains the text "spec.dat" and your computer has a file named "spc.dat").

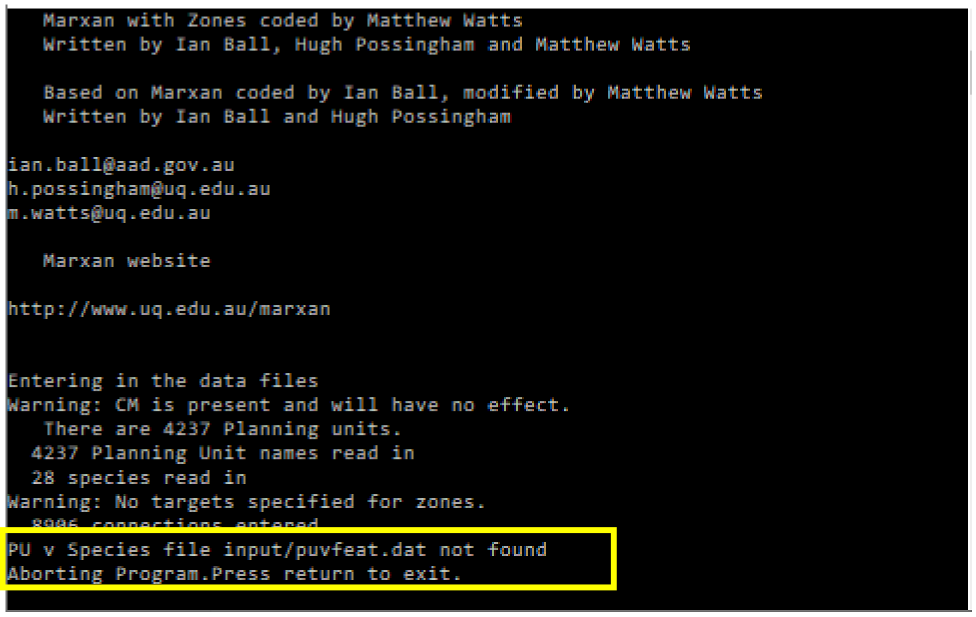


```
   Marine Reserve Design with Zoning and Annealing

   Marxan with Zones coded by Matthew Watts
   Written by Ian Ball, Hugh Possingham and Matthew Watts

   Based on Marxan coded by Ian Ball, modified by Matthew Watts
   Written by Ian Ball and Hugh Possingham

ian.ball@aad.gov.au
h.possingham@uq.edu.au
m.watts@uq.edu.au

   Marxan website

http://www.uq.edu.au/marxan


Entering in the data files
Warning: CM is present and will have no effect.
   There are 4237 Planning units.
  4237 Planning Unit names read in
Species file input/feat.dat has not been found.
Aborting Program.Press return to exit.
```

A-1.5 “PU v Species file input/ not found”

Error Message: The last thing Marxan with Zones shows is " *PU v Species file input/ not found. Aborting Program. Press return to exit*”.

Cause of Error Message: This occurs when your planning unit versus species file name ‘puvspr.dat’ specified in input.dat via PUVSPRANME is not included or does not match the actual species file name on your computer (e.g. the input.dat file contains the text "puvspr.dat" and your computer has a file named "puspc.dat").



```
   Marxan with Zones coded by Matthew Watts
   Written by Ian Ball, Hugh Possingham and Matthew Watts

   Based on Marxan coded by Ian Ball, modified by Matthew Watts
   Written by Ian Ball and Hugh Possingham

ian.ball@aad.gov.au
h.possingham@uq.edu.au
m.watts@uq.edu.au

   Marxan website

http://www.uq.edu.au/marxan


Entering in the data files
Warning: CM is present and will have no effect.
   There are 4237 Planning units.
  4237 Planning Unit names read in
  28 species read in
Warning: No targets specified for zones.
  8906 connections entered
PU v Species file input/puvfeat.dat not found
Aborting Program.Press return to exit.
```

A-2 Invalid planning unit file (pu.dat)

A-2.1 Marxan with Zones crashes

No Error Messages given

Cause: The ‘put.dat’ file contains extra newline characters (e.g. blank spaces or extra lines) at the end of the file and these need to be deleted. Or the pu.dat file has planning units ids that are not present in any other file.

Appendix B- Marxan with Zones Objective Function and Algorithms used

This appendix contains technical details about the way Marxan with Zones runs. It provides supplementary information on the objective function and the algorithm underpinning Marxan with Zones. While this information is not necessary to conduct basic runs, knowing how the program runs will assist in understanding how the changes you make to different parameters affect the results.

B-1 The Objective Function

The objective function gives a value for each zoning configuration of planning units. This means that a single planning unit or no planning units at all can be given an objective function value. A configuration containing zero planning units would be cheap to implement but would probably not meet the planning objectives and so the objective function value should be very poor.

If we have an objective function which gives any possible zoning configuration a value, or score, then we can compare any two zone configurations and say which one is better than the other (according to the objective function). Because the objective function value can be evaluated by a computer, the door is open to using a wide range of methods to automatically create zone configurations that have good objective function values.

In Marxan with Zones, this objective function is used by the iterative improvement algorithm and by simulated annealing. The objective function was designed with the aim to integrate it with a simulated annealing optimiser but the two are distinct entities. Simulated annealing is a general purpose optimiser and the objective function defines the constraints and objectives for a zone configuration without explicitly defining how an optimal zonation will be found.

The objective function consists of two main sections; the first is a measure of the ‘cost’ of the zone configuration and the second a penalty for breaching various criteria. In its simplest form, it is a combination of the economic cost of the zone configuration and a penalty for not meeting all of the objectives, if any are unmet. These criteria can include a cap on the ‘cost’ of the zone configuration and always includes the target representation level for each feature. As well as this is the optional measure of the fragmentation of the zone configuration and an optional cost threshold penalty. In this objective function, the lower the value the better the zone configuration:

$$\sum_{PUs} Cost \ + BLM \sum_{PUs} Boundary \ + \sum_{\substack{Con\\Value}} FPF \ \times Penalty \ + CostThresholdPenalty(t)$$

Cost is the sum of each cost measure multiplied by the value indicated in zonecost.dat file of each of the PU within the zone configuration.

'Boundary' is the cost of the boundary surrounding each zone configuration. The constant BLM is the boundary length multiplier, which will determine if the boundary cost will be accounted for in the objective function. For example, if a value of 0 is given to BLM then the boundary length is not included in the objective function.

The next term is a penalty given for not adequately representing a feature, summed over all features for each zone. FPF stands for 'feature penalty factor' and is a weighting factor for the feature which determines the relative importance for satisfying that particular feature's target. The penalty term is a penalty associated with each underrepresented feature. It is expressed in terms of cost and boundary length and is roughly the cost and additional modified boundary needed to adequately capture a feature which is not adequately represented in the current zone configuration.

The cost threshold penalty is a penalty applied to the objective function if the target cost is exceeded. It is a function of the cost and possibly the boundary of the system and in some algorithms will change as the algorithm proceeds (which is the t in the above formula). This penalty is also optional and can be excluded from the objective function.

B-1.1 Cost of the Zone Configuration

The objective function that Marxan with Zones uses has the constraint that the cost of a zone configuration is the linear combination of costs of all the planning units within the zone configuration, broken down by zone.

B-1.2 Boundary Length and Fragmentation

By including a boundary length term in the objective function we can apply a control on the level of fragmentation in the zone configuration. In order to allow the boundary length to be added to the cost measure, a multiplicative factor is used. This is because the boundary length is most probably going to be in units which are different from the cost measure. Not only are the units incompatible without a boundary length modifier, but the importance of compactness over cost is not immediately obvious. Changing the weights within the zone boundary cost allow planners to explore this issue.

B-1.3 Representation Requirements for Features

In the objective function, if the feature does not meet one or more of its requirements then it will attract a penalty depending upon how far below representation it is and the relative value of the feature to the other features.

B-1.4 Feature Penalty

The feature penalty is the penalty given to a zone configuration for not achieving feature targets. It is based on a principle that if a feature is below its target representation level, then the penalty should be close to the cost for raising that feature up to its target representation level. For example: if the requirement was to represent each feature by at least one instance then the penalty for not having a given feature would be the cost of the least expensive planning unit which holds an instance of that feature. If you were missing a number of features, then you could produce a zone configuration that was fully representative by adding the least expensive planning units containing each of the missing features. This would not increase the objective function value for the zone configuration, in fact, if any of the additional planning units had more than one of the missing features, then the objective function value would decrease. It would appear to be ideal to recalculate the penalties after each change had been made to the zone configuration. However, this would be time consuming and it turns out to be more efficient to work with penalties which change only in the simplest manner from one point in the algorithm to the next. A greedy algorithm is used to calculate the cheapest way in which each feature could be represented on its own and this forms the base penalty for that feature. Marxan with Zones adds together the cheapest planning units which would achieve the representation target. This approach is described in the following pseudo-code:

1. For each planning unit calculate a ‘cost per area’ value.
    A. Determine how much of the target for the given feature is contributed by this planning unit.
    B. Determine the economic cost of the planning unit
    C. Determine the boundary length of the planning unit
    D. The overall cost is economic cost + boundary length x BLM x zone boundary cost
    E. Cost-per-hectare is then the value for feature divided by the overall cost.

2. Select the planning unit with the lowest cost-per-hectare. Add its cost to the running cost total and the level of representation for the feature to the representation level total.
   A. If the level of representation is close to the target then it might be cheaper to pick a ‘cheap’ planning unit has the required amount of the feature regardless of its cost per area.
3. Continue adding up these totals until you have found a collection of planning units which adequately represent the given feature.
4. The penalty for the feature is the total cost (including boundary length x BLM x zoneboundarycost) of these planning units.

Thus, if one feature were completely unrepresented then the penalty would be the same as the cost of adding the simple set of planning units, assuming that they are isolated from each other for boundary length purposes. This value is quick to calculate but will tend to be higher than optimum. Sometimes, there are more efficient ways of representing a feature than that determined by a greedy algorithm, consider the following example.

Example 1: Feature A appears on a number of planning units, the best ones are:

Planning Unit	Cost	Amount of A represented
1	2	3
2	4	5
3	5	5

The target for A is 10 units. If we use the greedy algorithm we would represent this with planning units 1, 2, and 3 (selected in that order) for a total cost of 11.0 units. If we chose only planning units 2 and 3, we would still adequately represent A, but our cost would be 9 units.

This example shows a simple case where the greedy algorithm does not produce the best results. The greedy algorithm is rapid and produces reasonable results. The program will tend to overestimate and never underestimate the penalties when using a greedy algorithm. It is undesirable to have a penalty value which is too low because then the objective function might not improve by fully representing all features. If there are some features which need not be fully represented, this should be handled entirely by use of the feature penalty factor, which is described below. It is not problematic to have penalties which are higher then they absolutely need to be, sometimes it is desirable. The boundary cost for a planning unit in the above pseudo-code is the sum of all of its boundaries. This assumes that the planning unit has no common boundaries with the rest of the zone configuration and hence will again tend to overestimate the cost of the planning unit and the penalty.

The penalty is calculated and fixed in the initialization stage of the algorithm. It is applied in a straight forward manner - if a feature has reached half of its target then it scores half of its penalty. The problem with this is that you might find yourself in a situation where you only need a small amount to meet a feature's target, but that there is no way of doing this which would decrease the objective value. If we take Example 1 once again, then the penalty for feature A is 11 units. If you already have planning units 1 and 4 in the zone configuration, then you have 9 units of the feature and the penalty is 11.0 x (10-9)/10 = 1.1 units. So the feature attracts a penalty of 1.1 units and needs only 1 more unit of abundance to meet its target. There is no planning unit with a cost that low - the addition of any of the remaining planning units would cost the zone configuration much more than the gain in penalty reduction.

This problem can be fixed by setting a higher FPF (Feature Penalty Factor) for all features. The FPF is a multiplicative factor for each feature, described below.

It is possible that the target for a feature is set higher than can possibly be met. In Australia where the JANIS (1997) requirements state that 15% of the pre-European area of each forest ecosystem type should be protected, we can easily have targets which are larger than the current area of a given forest ecosystem. Currently, when this is the case, the algorithm will scale up the penalty so that if, for example, it costs 100 units to protect all of a given ecosystem but that represents only half of the target, then the initial penalty will be 200 units. This means that if you get half-way to your target then the penalty for that feature will be half the maximum penalty, no matter how high the target or whether it is a feasible target.

B-1.5 Feature Penalty Factor

The feature penalty factor (FPF) is a multiplicative factor which can be unique to each feature. It is primarily based on the relative worth of that feature but it includes a measure of how important it is to get it fully represented. The actual effect that it will have varies between the methods which use the objective function. If it is below 1 then the algorithm might refuse to add a planning unit to protect that feature if there are no other features on the planning unit. An algorithm might fall slightly short in the representation of features, getting close to, but not at or above, the target value. To ensure that each feature meets the target it can sometimes be desirable to set the FPF at greater value than 1.

B-1.6 Cost Threshold Penalty

The cost threshold is an option which allows the user to cap the zone configuration to a set cost + modified boundary length. This has been included to make it possible to look at a reverse version of the problem. The reversal of the problem would be to find the zone configuration which has the best representation for all features constrained by a maximum cost. It works by applying a penalty if the given threshold value is exceeded.

The penalty is the amount by which the threshold is exceeded multiplied by the cost threshold penalty. The penalty depends upon the stage of the annealing algorithm (how far into the annealing process the system is given as a proportion). The multiplier is:

$$Cost\ Threshold\ Penalty = Amount\ over\ Threshold\ \times (Ae^{-bt} - A)$$

Here t is the time during the run and varies between 0 and 1. A and B are control parameters. B controls how steep the curve is (A high B will have the multiplier varying little until late in the run). A controls the final value. A high A will penalize any excess of the threshold greatly, a lower A might allow the threshold to be slightly exceeded. The multiplier starts at 0 when t is zero. Both A and B require some experimentation to set. The cost threshold penalty is built into the objective function and hence will apply to the annealing module along with the iterative improvement module.

B-2 Algorithms Method

There are two basic algorithms that can be used to formulate a solution in Marxan with Zones: simulated annealing and iterative improvement. Each of these algorithms can be used alone or in combination with each other. The run mode selected in the input parameter file determines which algorithm, or combination of algorithms, is used to formulate a solution in Marxan with Zones.

B-2.1 Simulated Annealing

Simulated annealing is based on iterative improvement with stochastic acceptance of bad moves to help avoid getting stuck prematurely in a local minimum. The implementation used in Marxan with Zones will run for a set number of iterations. For each iteration, a planning unit is chosen at random and may or may not be already in the zone configuration. The change to the value of the zone configuration - which would occur if this planning unit were added or removed from the system - is evaluated just as it was with iterative improvement. This change is combined with a parameter called the temperature and then compared to a uniform random number. The planning unit might then be added or removed from the system depending on this comparison.

The temperature starts at a high value and decreases during the algorithm. When the temperature is high, at the start of the procedure, then both good and bad changes are accepted. As the temperature decreases the chance of accepting a bad change decreases until, finally, only good changes are accepted. For simplicity, the algorithm should terminate before it can only accept good changes and the iterative improvement should follow it, because at this point the simulated annealing algorithm behaves like an inefficient iterative improvement algorithm.

There are two types of simulated annealing in Marxan with Zones. One is fixed schedule annealing in which the annealing schedule (including the initial temperature and rate of temperature decrease) is fixed before the algorithm commences. The other is adaptive schedule annealing in which the algorithm samples the problem and sets the initial temperature and rate of temperature decrease based upon its sampling.

B-2.2 Adaptive Annealing Schedule

The adaptive annealing schedule commences by sampling the system a number of times (number of iterations/100). It then sets the target final temperature as the minimum positive (ie least bad) change which occurred during the sampling period. The maximum is set according to the formula:

$$Tinit = Min\ Change + 0.1\ \times (Max\ change - Min\ change)$$

This is based upon the adaptive schedule in (Conolly, 1990). Here, Tinit is the initial temperature, the changes (min and max) are the minimum and maximum bad changes which occurred. In our case a bad change is one which increases the value of the objective function (ie a positive value).

B-2.3 Fixed Annealing Schedule

With fixed schedule annealing the parameters which control the annealing schedule are fixed for each implementation of the algorithm. This is done typically by some trials of the algorithm with different parameters for a number of iterations which is shorter by an order of magnitude to the number to be used in the final run. The parameters will often benefit by being changed for longer runs but still based on the trials. The trials include looking at final results and also tracking the progress of individual runs. The annealing schedules which arise from the fixed schedule process are generally superior to the adaptive annealing schedule. Adaptive annealing is advantageous as it does not require a skilled user to use the algorithm and because it is quicker. It is faster in terms of the processing time required as there is much less in the way of initial runs, it is considerably faster in terms of the time which the user must apply in running the algorithm. For this reason it is taken as a major algorithm to be examined here. Land-use designers and managers will tend to use the standard options and automatic methods to a large extent so that the ability of adaptive annealing to design zone configuration is very useful, although obviously not definitively important. Adaptive annealing is also important for broad investigations, tests and trials on the system which would precede the more careful and detailed use of a fixed schedule annealing algorithm.

B-2.4 Setting a Fixed Annealing Schedule

When setting a fixed schedule the two parameters to change are the initial and final temperature. The final temperature is set by choosing an appropriate value for the cooling function. If the final temperature is too low then the algorithm will spend a lot of time at a local minimum unable to improve the system and continuing to try. If the final temperature is too high then much of the important annealing work will not be completed and the zone configuration will largely be delivered by the iterative improvement schedule which follows simulated annealing and finds a nearby local minimum 'at random'. If the initial temperature is too high then the system will spend too much time at high temperatures around the high temperature equilibrium and less time where most of the annealing work is to be done.

Thus the best way to get the general feel for what the parameters should be is to run the algorithm and sample the value of the current system regularly to see when the equilibrium at various temperatures seems to be achieved, what they are and when the system no longer changes or improves. This makes it easy to set a provisional final temperature and also gives estimates at what reasonable initial temperatures might be. From here tests can be run looking at the final output from multiple runs and different parameters at a much shorter number of iterations.

Once good values have been found they need to be scaled up for the longer number of iterations. This is because the length of time spent at lower or critical temperatures is important and will drive the search for good parameters. Extending the length of the algorithm will increase the time spent at these temperatures longer than is necessary. Thus the method used is to keep the final temperature the same and increase the level of the initial temperature so that it will spend a similar length of time at lower levels but allow it to search the global space to a greater extent. For a short run it is often best to have the system running at some critical temperature for as long as possible. For a longer run it is advantageous to increase the range of temperatures used.

B-2.5 Iterative Improvement

Iterative improvement is a simple optimization method. It has largely been supplanted by simulated annealing but can still be profitably used to aid the other algorithms. There are three basic types of iterative improvement which can be used in Marxan Z. They differ in the set of possible changes which are considered at each step. Each of them starts with a ‘seed’ solution. This can be any kind of zone configuration with some, all, or no planning units allocated to zones. It is useful to use the final result from another algorithm such as simulated annealing as the starting solution for iterative improvement.

In this case the iterative improvement algorithm is used solely to ensure that no simple improvements are possible.

At each iteration, the algorithm will consider a random change to see if it will improve the value of the objective function if that change were made. If the change does improve the system then it is made, otherwise another, as yet untested, change is tested at random.

This continues until every possible change has been considered and none will improve the system. The resulting zone configuration is a local minimum (or local optimum).

The three basic types of iterative improvement differ in the types of change that they will consider. The simplest type is called ‘normal iterative improvement’ and the only changes that are considered are adding or removing each planning unit from the zone configuration. This is the same ‘move set’ as is considered by the greedy algorithm and by simulated annealing.

The second type of iterative improvement is called ‘swap’ and it will randomly select planning units, if the selected planning unit can improve the system by being added or removed from it then this is done otherwise an exchange is considered. If the chosen planning unit is already in a zone configuration then the changes considered are removing that planning unit but adding another one somewhere else. If the chosen planning unit is not in the zone configuration then the changes considered are adding this to the system but removing one that is already in the system. Every possible ‘swap’ is considered in random order, stopping when one is found which will improve the system. Because the number of possible exchanges can be very large, this is a much slower option.

The third type is called ‘two step’, in this method as well as testing each planning unit (in random order) to see if adding or removing it would improve the system, each possible combination of two changes is considered. These changes include, adding or removing the chosen planning unit in conjunction with adding or removing every other planning unit. The number of such moves is even greater than in the ‘swap’ method, so that this method should be used with care.

There is a fourth option which is to run the normal method first, to get a good local optimum and then run the ‘two step’ method afterward. Because the number of improvements that the ‘two step’ finds should be much smaller after a normal iterative improvement algorithm has passed over the ‘seed’ solution this should be much faster than running the ‘two step’ method on its own.

The main strength of iterative improvement is that the random element allows it to produce multiple solutions. On average the solutions might be poor, but if it can produce solutions quickly enough, then it may produce some very good ones over a great many runs. It is theoretically possible to reach the global minimum by running iterative improvement starting from either an empty zone configuration or a situation where every planning unit starts in a randomly selected zone.

The main use of iterative improvement will still be to follow a different algorithm for some fine-scale polishing up. This is particularly true for the ‘swap’ and ‘two step’ methods. Even following another algorithm these might take long to run.

MARXAN

conservation solutions