An introduction to Systematic conservation planning with prioritzr

Martin Jung & Louise O'Connor

2024-06-06

Table of contents

Preface

Welcome to the training course in systematic conservation planning with the prioritzr. This training course was originally held at the 2024 European Congress of Conservation biology in Bologna, although the materials found here will be preserved even after the conference and be openly available to everyone.

What you will learn

- The basic concepts of Systematic conservation planning (SCP) and Integer Linear Programming (ILP) in particular
- How to prepare your input data for a Conservation planning project
- How to setup and run your first prioritization
- How outputs can be analysed and interpreted.
- How to adding complexity factors and changing your conservation planning outcomes
- Advanced topics such as accounting for connectivity and management zones

Completing all course materials will take you on average 120 minutes, although people who have been exposed to similar methods or introduction before might take less. training materials before might less amount of time.

In this training course a number of different terms will be used. Whenever there are uncertainties with regards to definitions, see the Glossary.

If you have already heard before about the basic concepts of SCP and ILP (For example from the lecture then feel to jump to section 2 and data preparation Chapter 2.

Before you start...

In order to run the materials on this course website, some preparatory steps need to be taken. Please see the installation instructions in Appendix A if you have never used **prioritizr** before!

Part I

Introduction to SCP

1 Introduction

Welcome to this short introduction to systematic conservation planning with prioritizr! On this page you will learn about the basic concepts of systematic conservation planning (SCP) and more specifically algorithmic solutions identifying planning outcomes.

Course info

If you have taken part in person to the introduction on the day, you might want to skip this section and directly start with handling and preparing data at Chapter 2 .

1.1 Systematic conservation planning

The classical definition of Systematic conservation planning (SCP) is that of a structured, scientific approach to identifying and prioritizing areas for conservation (REFs). Its goal is to ensure that biodiversity is maintained and ecosystems are protected in a way that maximizes ecological, economic, and social benefits. Although SCP has been conceived specifically for creating and expanding reserve networks (usually protected areas), it can be used for much more including for example the identification of restoration, land-use planning or monitoring options.

It is also a common misconception that a project implementing SCP is only about prioritization (the algorithm part). Rather, it describes a whole framework typically ranging from

1. Defining Conservation goals and objectives
2. Eliciting pathways to impact and theory of change with stakeholders
3. Compiling and preparing data
4. Identifying targets, constraints and costs
5. Formulating a planning problem and identifying priorities for it
6. Evaluating said priorities through robust performance metrics
7. Implementing the priorities in exchange with stakeholders

8. Monitoring the performance and adapting plans where necessary.

1.1.1 Key concepts

1.2 Exact algorithms and integer programming

See Hanson *et al.* (2019) for additional discussion of optimality in linear programming.

1.3 Tools and software

We stress that prioritizr is not the only software available

Part II

Problem creation

2 Preparing input data

2.1 Planning units

Say something about what a PU file is, load and plot from the example data

2.2 Features

Explain what features are and plot

2.3 Costs

Explain what costs are and how they are used

2.4 Other constraints (Protected areas)

Showcase a protected area file and exclusion areas as example

2.5 Other data for the prioritization

Mention weights, targets, etc and give example for each

2.6 Preparing data in different formats

Showcase different dataformats as input alternative

```r
library(tidyverse)
trees |>                                                        ①
  mutate(
  volume_girth = volume * girth                                 ②
  )
```

① Take dataset and mutate
② Update with interaction term

3

Part III

Solving a problem

4 Solving a conservation planning problem

4.1 Ensure that a solver is available

4.2 Create a solution

4.3 Plot the solution

5 Interpret and analyse outputs

5.1 Plot the solution

5.2 Calculate performance evaluation metrics

Area statistics, Mean representation, Target shortfall

5.3 Irreplaceability

Calculater ferrier irreplacibility and RWR

Part IV

Adding complexity

6 Objective functions

6.1 The need for targets

6.2 Minimum set

6.3 Maximum coverage

6.4 Creating ranked priority maps

7 Adding complexity to conservation planning

7.1 Decision variables

7.2 Adding (socio-economic) costs

7.3 Feature weights

7.4 Linear penalities

Part V

Advanced topics

8 Connectivity

Conservation planning can be used to obtain area-based solutions to identify options for (improved) conservation of species. In reality however many seemingly 'optimal' solutions in terms of complementarity (e.g. covering the best areas for conserving selected features) might not work for species that persist only in isolated populations, which are thus more prone to extinction. Here a strategy is not to identify (and conserve) a single site, but manage a network of sites that are ideally as much as possible connected.

What this imply for area-based conservation planning? It means ideally sites are selected in a way that not only maximizes complementarity but also results in compact and/or structurally and functionally connected areas.

The aim of this section is to describe different way of 'directly' considering connectivity in area-based conservation planning with *prioritizr*. For a comprehensive overview on the general principles of considering connectivity in area-based planning we recommend several recent reviews and perspectives Hanson *et al.* (2022)

! Note

Much of the code examples in this section might take quite a bit of time to run and requires knowledge of how to set up a problem formulation. We suggest to try these options only as you are familiar with modifying problem formulations and altering outputs.
For demonstration purposes we focus on the Alpine region for these examples. You can obtain a shapefile of their outline here.

Although by no means comprehensive, we broadly consider three commonly applied but different ways of considering connectivity in prioritzr.

1. Boundary penalties that prefer larger compared to smaller sites (Ball *et al.* 2009).
2. TODO: Connectivity constraints. Synchronous and asynchronous
3. TODO: Connectivity features

8.1 Boundary penalties

The inclusion of boundary penalties is one of the oldest and most widely applied ways of forcing a prioritization output (Ball *et al.* 2009). By setting a boundary length modifier (BLM) or penalty constant, we effectively penalize solutions that result in overly fragmented patches. Since it is a penality it does not fully prevent them however.

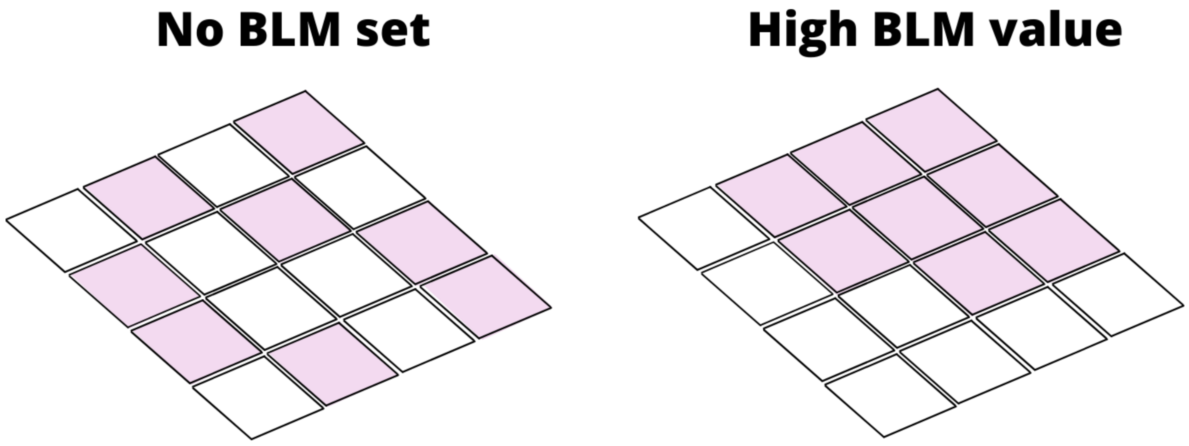


Figure 8.1: Boundary length modifier (BLM), which is effectively a penalty (Source: Marxan solutions)

Unfortunately, and similar to other penalty values, there are no specific guidelines of what might work or not, so often it might be worth exploring a few options.

As in previous tutorials we first load our data. However as noted above, we focus on the Alpine region only to make this interpretable. To do so we first crop and mask our PU and feature data to the alps.

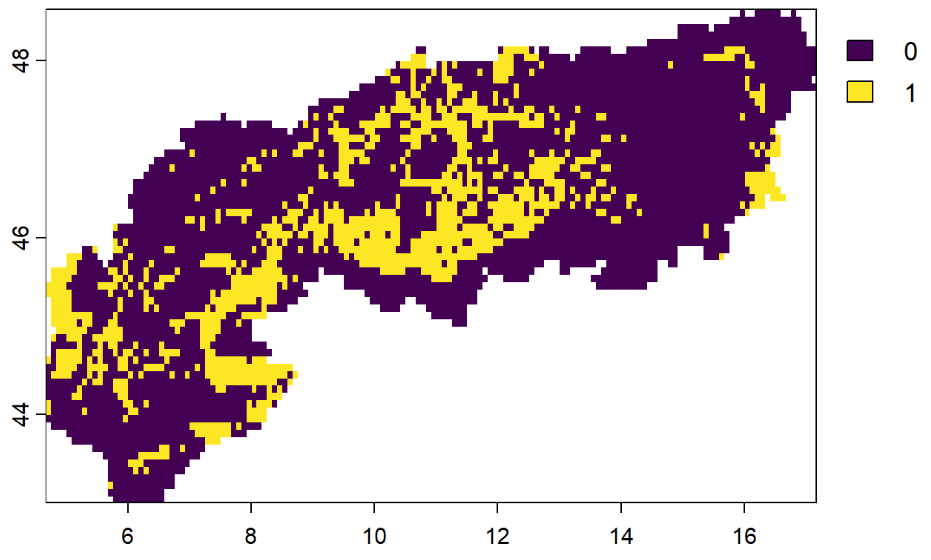
```r
# Crop. Focus on the alps here
alps <- sf::st_read('extdata/boundary_alps/AlpineConvention.shp') |>
  sf::st_transform(crs = sf::st_crs(4326))

PU <- PU |> terra::crop(alps) |> terra::mask(alps)
spp <- spp |> terra::crop(alps) |> terra::mask(alps)
```

Now we can create a conservation planning problem for this region.

```r
p <- problem(PU, spp) |>                               ①
  add_min_set_objective() |>                           ②
  add_relative_targets(targets = 0.3) |>               ③
  add_binary_decisions() |>                            ④
  add_default_solver()                                 ⑤
```

① A problem with the cropped data (Planning units and features)
② Using a minimum set operation here.
③ Arbitrary targets of 30% of the feature distribution
④ Binary decisions
⑤ Use the fastest solver installed/available (usually Gurobi or cbc)

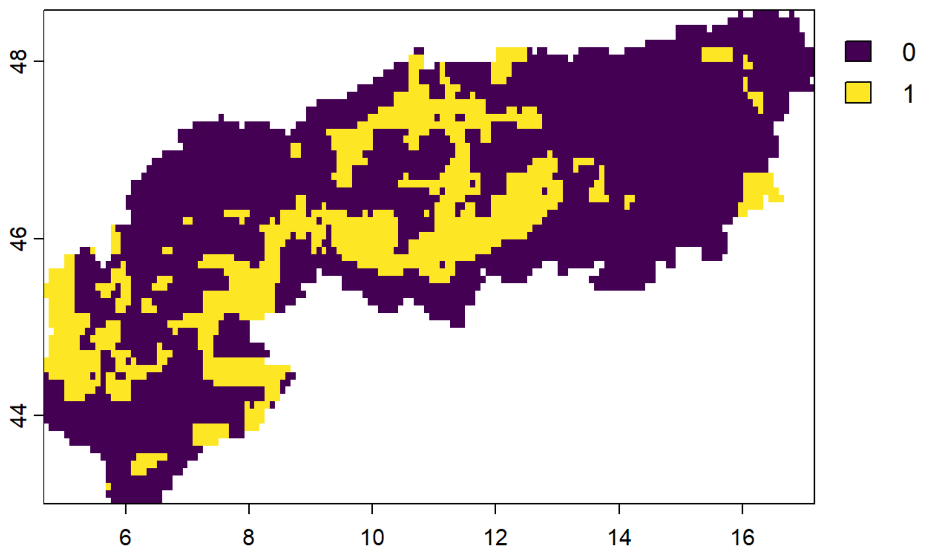


Now lets add some boundary constraints to the same problem.,

```r
# First we precompute the boundary matrix (large matrix of neighbourhoods)
bm <- boundary_matrix(PU)
# Then we rescale it for better performance
bm <- rescale_matrix(bm)

# Now create a new problem using the settings from above, but with a boundary penality
s_blm <- p |>
  add_boundary_penalties(penalty = 1e-4, data = bm) |>                    (1)
  solve()
```

① Specify a boundary penalty. Usually this requires some trial-and-error.



As you can see the solution is effectively more '*clumped*'. But what about the area selected? Do we need more area to get the best complementary solution here?

```
# calculate costs (sum of area)
dplyr::bind_rows(
  eval_cost_summary(p, s),
  eval_cost_summary(p, s_blm)
)

# Answer is...?
```

Performance

Boundary length penalties generally solve faster with simpler objective functions, such as a minimum set objective function.

8.2 Connectivity constraints

Glossary

Table 8.1: A glossary of key terms used in this Training course

Term	Abbreviation if any	Definition
Boundary Length Modifier	BLM	A penalty constant added to a conservation problem that penalizes selecting isolated patches. Results in overall more compact solutions.
CARE	CARE	A often used abbreviation that stands for ***C****onnectivity*, ***A****dequacy*, ***R****epresentation*, and ***E****ffectiveness* which key principles that should be considered when designing a conservation network. See the Marxan website for more information.
Conservation Prioritization		The computational process of identifying (spatial) priorities for a given conservation objective (such as for identifying protected areas). Usually comes in in the form of a map.
Integer		In programmatic terms a full number (e.g. -1, 1, 2, 3, …)
Integer Linear Programming	ILP	Mathematical problem formulation using Linear Programming (ILP) where the variables are integer values and the objective function and equations are linear.
Planning unit	PU	The fundamental unit at which decisions in SCP are realized. Can be of multiple formats such as grid cells or farms
Systematic Conservation Planning	SCP	A framework and step-wise approach towards mapping conservation areas. Usually involves multiple steps such as the identification of a problem and the theory of change, data collection and preparation, conservation prioritization, evaluation and finally implementation. See Margules & Pressey (2000)

Term	Abbreviation if any	Definition

References

Ball, I.R., Possingham, H.P. & Watts, M. (2009). Marxan and relatives: Software for spatial conservation prioritisation. *Spatial conservation prioritisation: Quantitative methods and computational tools*, *14*, 185–196.

Beger, M., Metaxas, A., Balbar, A.C., McGowan, J.A., Daigle, R., Kuempel, C.D., Treml, E.A. & Possingham, H.P. (2022). Demystifying ecological connectivity for actionable spatial conservation planning. *Trends in Ecology & Evolution*, S0169534722002221.

Daigle, R.M., Metaxas, A., Balbar, A.C., McGowan, J., Treml, E.A., Kuempel, C.D., Possingham, H.P. & Beger, M. (2020). Operationalizing ecological connectivity in spatial conservation planning with marxan connect (N. Golding, Ed.). *Methods in Ecology and Evolution*, *11*, 570–579.

Hanson, J.O., Schuster, R., Strimas-Mackey, M. & Bennett, J.R. (2019). Optimality in prioritizing conservation projects. *Methods in Ecology and Evolution*, *10*, 1655–1663.

Hanson, J.O., Vincent, J., Schuster, R., Fahrig, L., Brennan, A., Martin, A.E., Hughes, J.S., Pither, R. & Bennett, J.R. (2022). A comparison of approaches for including connectivity in systematic conservation planning. *Journal of Applied Ecology*, *59*, 2507–2519.

Margules, C.R. & Pressey, R.L. (2000). Systematic conservation planning. *Nature*, *405*, 243–253.

A Installation of all required software

Opposed to other conservation planning software (e.g. Zonation 5) using prioritizr requires prior knowledge on how to use **R**.

A.1 Install R

R is a programming language and environment specifically designed for statistical computing and graphics. It is widely used among statisticians and data analysts for its extensive capabilities in data manipulation, statistical modelling, and graphical representation.

To install R, please go to the following website, then:

1. Click on the link at the top for your respective operating system
2. Recommended is the **base** version of R particular for new users. Select the latest version 4.4, download and execute.
3. Follow the instructions in the installation popup.

Although older R-versions can work as well (e.g. R 4.3), we recommend the latest version with which the training materials have been tested.

In addition, we also recommend the installation of **RTools** on the same website (here for example for Windows). RTools contains a range of code compilation software, such as a C++ compiler. These software are often necessary to install additional R-packages, particular when they are not available in binary format.

To download RTools, click the "Rtools44 installer" link, download and execute and follow the instructions.

A.2 Install a IDE such as Rstudio

By default R is terminal based, meaning inputs are parsed as entered. To create reproducible scripts we recommend the use of an integrated development environment (IDE) and here in particular Rstudio. Of course other alternative IDEs can also be used such as for example Visual Code. It is free to use in its basic version and available for most operating systems, including Windows 10/11, Linux and MacOS distributions.

To download and install Rstudio follow the instructions on this website.

A.3 Install a solver in R

To use prioritizr and solve a conservation problem, we require a solver. Solvers are specialized algorithms or software designed to find the best solution (or an optimal solution) to a mathematical problem that involves maximizing or minimizing a particular function subject to certain constraints. For different mathematical problems, for example linear or mixed programming, different solvers are often necessary or perform better.

Many state-of-the-art solvers are proprietary and often used by large companies to solve problems related to supply chain or financial risk managements. Although freely available and open-source solver slowly catch up, they usually cannot compete with proprietary such as Gurobi or CPLEX. For a comprehensive overview of different available and supported solvers a detailed vignette can be found on the prioritizr website.

For new users we recommend the use of the *HiGHS* solver, which is free to use and can be installed across a range of operating systems. To enable it run the following code and make sure it runs through without issues.

```r
install.packages("highs")
```

If for some reason the installation of he package fails, another option could be the *cbc* solver, which can currently only be installed directly from the developers Github repository. For this to work you likely need to have RTools installed (see A.1 above).

```r
if (!require(remotes)) install.packages("remotes")
remotes::install_github("dirkschumacher/rcbc")
```

Gurobi

The Gurobi solver is among the fastest supported ones for prioritizr. Unfortunately it is not openly available and purchasing it can be quite costly. However for academic users (those with an academic email) and researchers it is possible to obtain a time-limited (usually 12 months) license for research projects. This License can also be renewed. For further information see the installation vignette on the prioritizr homepage!

A.4 Install required R packages

In addition to the R and the solver packages above, we need to install several packages related to (spatial) data handling. These include for example *dplyr*, *terra* and sf, but also *ggplot2* for plotting.

To install please run the following code in your R terminal:

```r
install.packages("dplyr")
install.packages("terra")
install.packages("sf")
install.packages("ggplot2")
install.packages("tidyterra")
```

Make sure that every line executes without an error. If you see an error, check first online for potential solutions (google) and afterwards get in touch with the course organizers.