# An introduction to Systematic conservation planning with prioritzr

Martin Jung & Louise O'Connor

2024-06-16

# 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 3.

> **i** 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 3 .

## 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 (Margules & Pressey (2000)). 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.

## 1.2 Exact algorithms and integer programming

Exact algorithms in spatial planning are computational methods designed to find optimal solutions to spatial planning problems, where spatial planning involves the organization, management, and allocation of land and resources within a given area. These algorithms guarantee to find the best possible solution based on the defined criteria, constraints, and objectives of the problem.

Exact algorithm enable the solving of SCP problems as a mathematical model, such as a mixed (MILP) or integer linear programming (ILP) typically. Linear in this context refers to this common formulation of a planning problem, although non-linear problem formulations (e.g. quadratic or even more complex functions) are also possible. All LP problems have in common a specific objective function such as the maximum coverage or minimum set problem. See Hanson *et al.* (2019) for additional discussion of optimality in linear programming.

## 1.3 Tools and software

There are a range of tools and software for creating prioritizations in a SCP framework. Typical other well-known complementarity-based spatial conservation prioritization software are for example Zonation and Marxan, both of which use heuristic approaches for identifying priorities.

For ILP problems the prioritizr R-package is the easiest and most comprehensive package currently available, although other options exist as well. It should be stressed that in principle any mathematical or programming language can be used to solve ILP problems. The *prioritizr* package simply provides a convenience wrapper.

# Part II

# Problem creation

# 2 Obtaining data for the course

To get started, please download the data from github from **here**. Unzip the data into a folder called data to follow along with the tutorial. Alternatively you can also fork or clone the entire tutorial from github if you feel comfortable with git (A version control system).

## 2.1 Description of testing data

Specifically this folder contains:

1. Species Distributions: species distributions modeled under current climatic conditions and future climate scenario RCP 8.5 for 67 tree species. Note: all spatial raster data is at ~10x10 km resolution (WGS84) and harmonized spatially.
2. other data, including:
    1. planning units raster
    2. NDVI: proxy for dense and healthy vegetation cover.
    3. ghm: global human modification index. Species distribution models, NDVI and GHM spatial data were originally extracted and prepared by Thiago Cavalcante for this course and the Zonation 5 software. For further details on the species distributions, NDVI and GHM layers, see https://github.com/zonationteam/Zonation5-training/tree/main/2024%20Europe/Exercises/data
    4. urban_prct and HI_forest_prct.tif : percentage coverage per 10x10 grid cell of urban/periurban, and high-intensity forests, respectively. Aggregated to 10x10k from European land systems dataset by Dou *et al.*, 2021, originally at 1km resolution.
    5. protectedareas.tif : percentage of protected area coverage per grid cell, considering all protected areas for the study area, extracted from WDPA
    6. protectedareas_I_II.tif : percentage of strictly protected area coverage per grid cell, with strictly protected areas defined as IUCN categories I and II and extracted from WDPA

7. species_red_list.csv : global and European assessment of the red list status for each of these 67 species

**Steps for data preparation**

For the use in prioritizr features, and all other spatial data, need to be perfectly harmonized with the planning units data and (same extent, resolution, number of grid cells). This step must be done prior to the prioritisation, as part of the data preparation.
For this training workshop, the data is already prepared, but bear in mind that data preparation is an essential step in the conservation planning process. Note that in practice for any SCP project the data preparation might take considerable time for extraction and harmonization.

# 3 Loading prepared input data

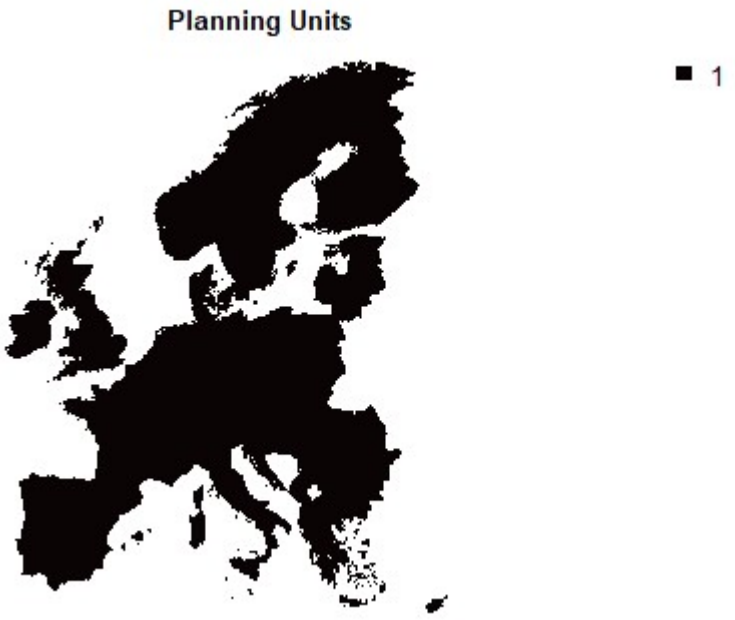
In the following sections we will load and explore the various data sources used for the planning in this course.

## 3.1 Planning units

Planning units (PU) contains the spatial data of the study area. Although a range of different data formats are theoretical possible in prioritizr. PU are generally defined in SCP as the spatial units at which decisions are realised. For this tutorial we primarily rely on a raster format, specifically 10x10 km grid cells in Europe.

Let's read and plot the planning units raster:

```
# Required packages
library(terra)
library(viridisLite)

# Load the Planning unit
PU <- rast("data/PlanningUnits.tif")
plot(PU, col = viridisLite::mako(n = 1))
```




The value of the planning units can determine the cost of each planning unit in the prioritisation. In our case, we often want to reach 30% area coverage (out of total area) for example. The cost for achieving this is the amonut of land value in the Planning Units raster, here specified as equal value of 1 (so that the budget will be expressed in number of grid cells in prioritizr).

> **!** Important
>
> Note especially when planning over larger extents the amount of area within a PU might differ depending on the geographic projection used. For this tutorial and simplicity, we rely on a longitude-latitude projection, which does not reflect area accurately (It is not an equal-area projection). In other words: PU in the north of Europe might contain less area than PU in the south of Europe despite having the same cost.
>
> When planning your own SCP project use a geographic projection appropriate for your case study!

## 3.2 Features

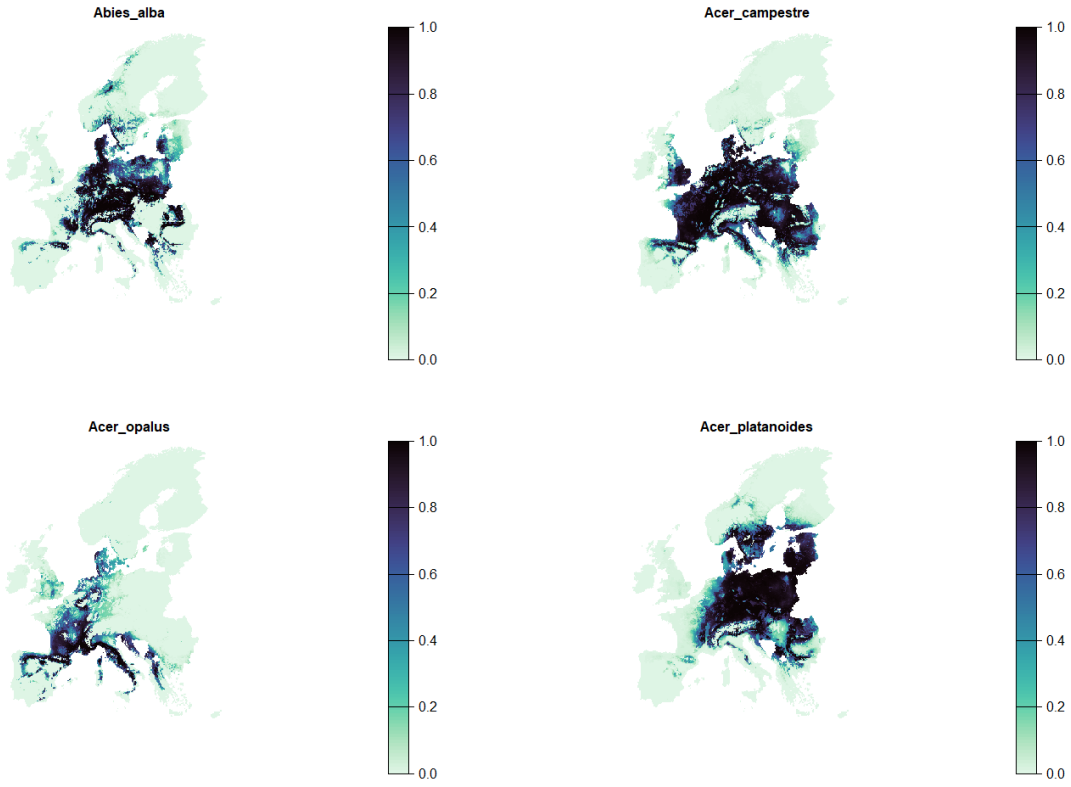
A feature is spatial data on the distribution of a biodiversity entity, typically a species, habitat, ecosystem service or similar.

Here, we consider the SDM of 67 tree species in Europe as features. We will focus on current distributions but we also provide projected distributions under a future climate scenarip (RCP 8.5) as part of this workshop.

Let's read the current SDM as a raster stack and plot one species as an example:

```
# Get the file names of the testing data
spp.list <- list.files(path = "data/SpeciesDistributions/", full.names = T, recursive = T, p

# Load all files and rename them
spp <- rast(spp.list[grep("current", spp.list)])
names(spp) <- gsub("_ens-sdms_cur2005_prob_pot", "" ,names(spp))           ①

# Plot first four species distributions
spp |> subset(1:4) |>
  plot(axes = F,col = viridisLite::mako(n = 100, direction = -1), main = c(names(spp)[1:4]))

# also load the SDM projected under climate scenario rcp 8.5
# read sdm under climate scenario rcp 8.5
spp.rcp85 <- rast(spp.list[grep("rcp85", spp.list)])

# Similarly rename feature layers by species names
names(spp.rcp85) <- gsub("_ens-sdms_rcp85_fut2065_prob_pot.tif", "" ,names(spp.rcp85))
```

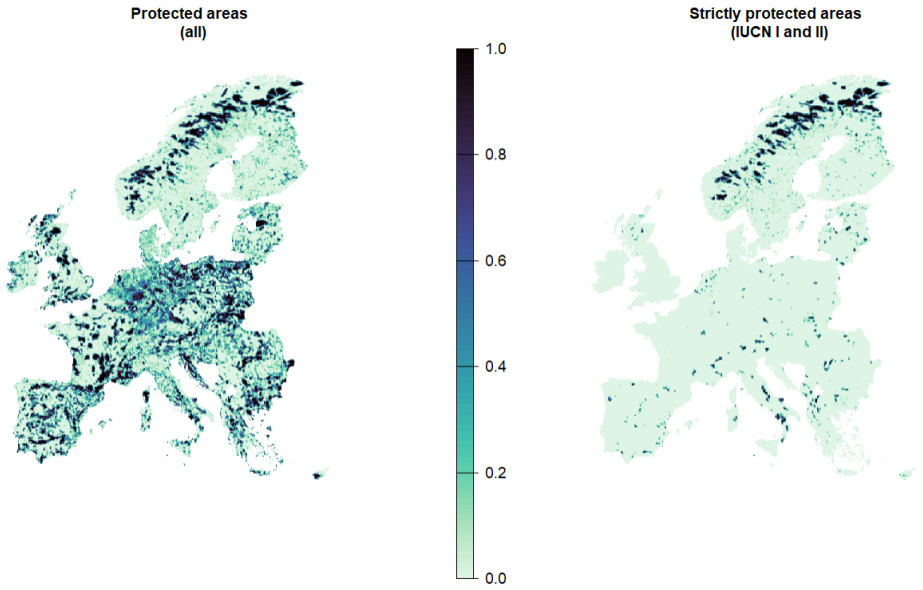
① We rename feature layers by species names. This will enable to link the features rasters to a table of feature characteristics, weights, targets, taxonomy.




## 3.3 Existing protected areas

Often, we do not start from scratch: we often want to identify top priorities that **complement and expand on existing** protected areas. See the Chapter 2 section for more information on what is contained in those two protected area layers.

```
# load protected areas data
PA <- rast("data/protectedareas.tif")

# load strict protected areas
stPA <- rast("data/protectedareas_I_II.tif")


plot(c(PA, stPA), axes = F, col = viridisLite::mako(n = 100, direction = -1), main = c("Prot
```



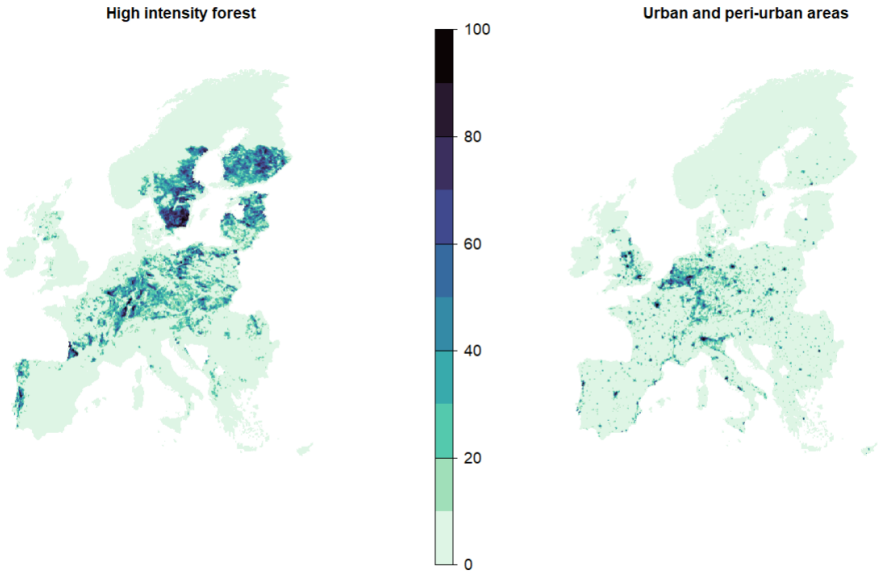

## 3.4 Areas under constrained use (locked-out or no-go areas)

Some areas are usually unavailable for SCP. Here we use layers of high-intensity forests and urban areas as a proxy, derived from 1km² European land systems data from Dou *et al.* (2021) . We lock out the planning units that have over 50% of urban and peri-urban, or over 50% of high intensity forest. In doing so, we assume that, in these high-intensity areas, conservation would likely conflict with economic interests.

```
## create locked out constraints to define areas that should be left out of the solution.
## from Dou et al., 2021
## aggregated at 10x10 k and aligned with the planning units raster
HI.forest <- rast("data/HI_forest_prct.tif")

urban <- rast("data/urban_prct.tif")

plot(c(HI.forest, urban), axes = F, col = viridisLite::mako(n = 10, direction = -1), main =

# --- #
# For further use we make a mask to lock out these areas
locked.out <- sum(HI.forest, urban)                                    ①
rclmat <- matrix(ncol = 3, nrow = 2, byrow = T,                        ②
                 c(0,50, 0,
```

```
                    50, 101, 1))

locked.out.bin <- terra::classify(locked.out, rclmat) ## convert to binary : 1 = pu that hav
```

① Sum up the area shares of both classes as a proxy.
② Define a matrix for reclassification as highlighted above.



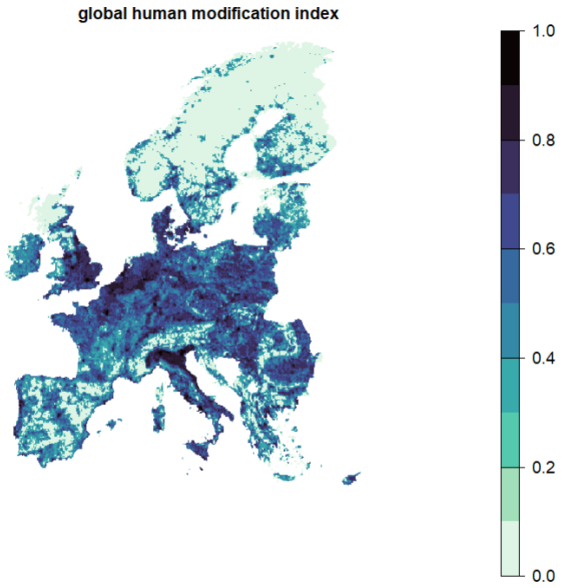

## 3.5 Costs

In the context of SCP Costs are typically spatially-explicit socio-economic data that can be factored into a prioritization to account for the feasibility of implementing conservation in a planning unit. In the planning they are typically used a constrain to penalize or limit the allocation of PU to a solution. There are different types of costs that commonly used:

- Acquisition cost = price of land/water area
- Opportunity cost = lost revenue to other land use types
- Transaction cost = e.g. cost of negotiating protection
- Management cost = maintenance and management of the PA

In reality, we rarely have this information and need to use proxies. Here, we use global human modification (GHM) as a proxy for socio-economic costs. Including the GHM as a cost layer would assume that highly human-dominated landscapes would be more costly to protect, than others.

```
gHM <- rast("data/gHM.tif")

# For simplicity we here use a threshold so that sites that have GHM index lower than specif
gHM[gHM<0.3] <- 0

plot(gHM, axes = F, col = viridisLite::mako(n = 10, direction = -1), main = "global human mo
```



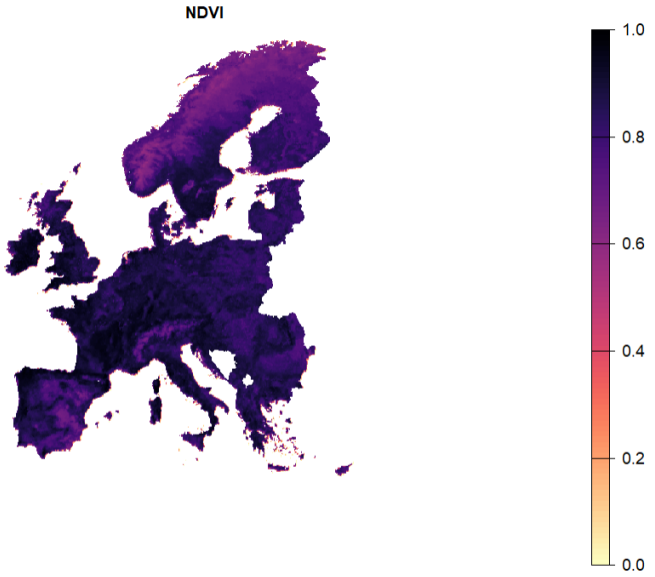

**About costs**

Generally be mindful about the use of costs in SCP as chosing any specific costing estimate can be quite impactful in driving final solutions.
For some further background reading we recommend McCreless *et al.* (2013), Kujala *et al.* (2018) and Armsworth (2014)

## 3.6 Vegetation quality

NDVI is often interpreted as dense and healthy vegetation, and one may be interested in selecting sites with a higher NDVI, for example when attempting to identify Green Infrastructure sites.

```
ndvi <- rast("data/ndvi.tif")
```




## 3.7 Other data for the prioritization

Spatial prioritisations can also be shaped by the importance of certain feature relative to others (e.g. threat status). This can be addressed by the use of feature-specific weights.

For setting weights, we will use the red list dataset and assign higher weight to more vulnerable species, following Jung *et al.* (2021).

Specifically we apply weights in this example as

- default weight of 1 for Least Concern species.
- 2 for near-threatened and data-deficient species
- 4 for vulnerable species

- 6 for endangered species
- 8 for critically endangered species.

```
## read red list information
redlist.trees <- read.csv('data/species_red_list.csv')

## assign weight based on red list status
redlist.trees$weight <- ifelse(redlist.trees$Global == "Vulnerable" | redlist.trees$Europe =
                                    ifelse(redlist.trees$Global == "Near Threatened" | red
                                           ifelse(redlist.trees$Global == "Data Deficient"

## must be in the same order as the features (spp) rasterstack
rownames(redlist.trees) <- redlist.trees$spp_name
redlist.trees <- redlist.trees[names(spp),]
```

Different weights are of course possible, for example by relying on expert or stakeholder feedback, evolutionary distinctiveness or cost benefits. Ultimately this is up to the spatial planner.

## 3.8 Targets

Another important aspect of planning are area-based targets, which define the amount of the distribution of each feature that is deemed sufficient to protect. Although one could set flat targets if there is a valid reasoning (e.g. 10% of all features), the most typical approach for targets is to use log-linear targets Rodrigues *et al.* (2004). Another is to use the IUCN criteria to set targets based on the minimizing extinction risk Jung *et al.* (2021).

Note that targets, similar as costs (see Section 3.5) can substantially drive the solution. Thus care should be taken how such targets are defined and used in SCP.

> **!** Necessity of targets
>
> Not every objective function in *prioritizr* requires targets. However the specification of targets is usually recommended as it forces the planner to think about the critical question of "How much do we want and need to conserve or manage". If such decisions are not taken by the analyst, it is usually taken by the algorithmic approach.

# 4 Create and understand planning problems

In the previous section (Chapter 3) we loaded a range of already prepared datasets including PU and features. Now we are ready to create our first conservation planning problem.

Prioritizr makes use of a 'tidyverse' informed and human-readable syntax where a problem is defined by adding data, features and constraints sequentially to an object. This is thus quite similar as the use of 'dplyr' in R.

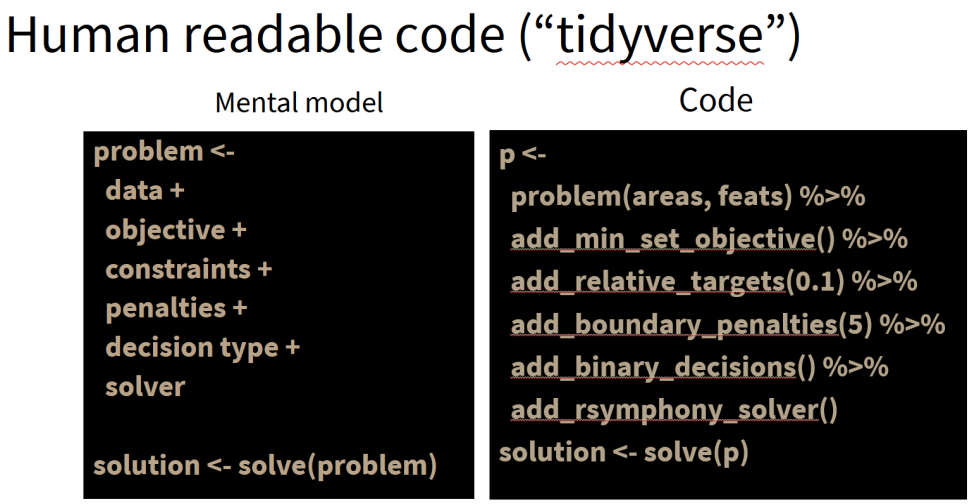



Figure 4.1: Tidyverse inspired flow

**Pipe**

We will use in this workshop the pipe symbol to chain different R functions (such as those from prioritizr) together. Useable are both the classical pipe from the magrittr package ( %>% ) and the pipe used by default since R version 4.0 ( |> ). We use both pipes often interchangeably in this workshop.

## 4.1 Our first planning problem

For our first problem we will create a problem that finds the best areas for 30% protected area coverage on European land. Go to the next section ( Chapter 5) to learn how to solve and interpret the outputs from a prioritizr problem.

**Tip:** Hover over the numbers on the right to learn more about each function. Note that the order

```
# Load the prioritizr package
library(prioritizr)

# define area budget (unit: grid cells)
budget.area <- round(0.3 * length(cells(PU)))                    ①

p <- problem(PU, spp) %>%                                        ②
  add_min_shortfall_objective(budget = budget.area) %>%          ③
  add_relative_targets(targets = 1) %>%                          ④
  add_cbc_solver() %>%                                           ⑤
  add_proportion_decisions()                                     ⑥
```

① This effectively defines the total budget as 30% of the length of all PU. This works since the length is identical to the sum (cost =1).
② Here we define a problem using the planning unit layer and the different species layer. Internally this will create an intersection of both. Other possible inputs to this function could be zones.
③ An objective function is added here. In this case we use the minimum shortfall objective.
④ For simplicity we define target as 100% for all species distributions
⑤ Here we add a solver. We rely here on CBC which in tests hasthe best performance among open-source solvers.
⑥ Here we add proportional decisions means that proportions of planning units can be selected in the solution. This typically solve faster than binary decisions.

## 4.2 Understanding the problem object

Now that we have created a problem, let's have a look at the object.

```
# Simply run
p
```

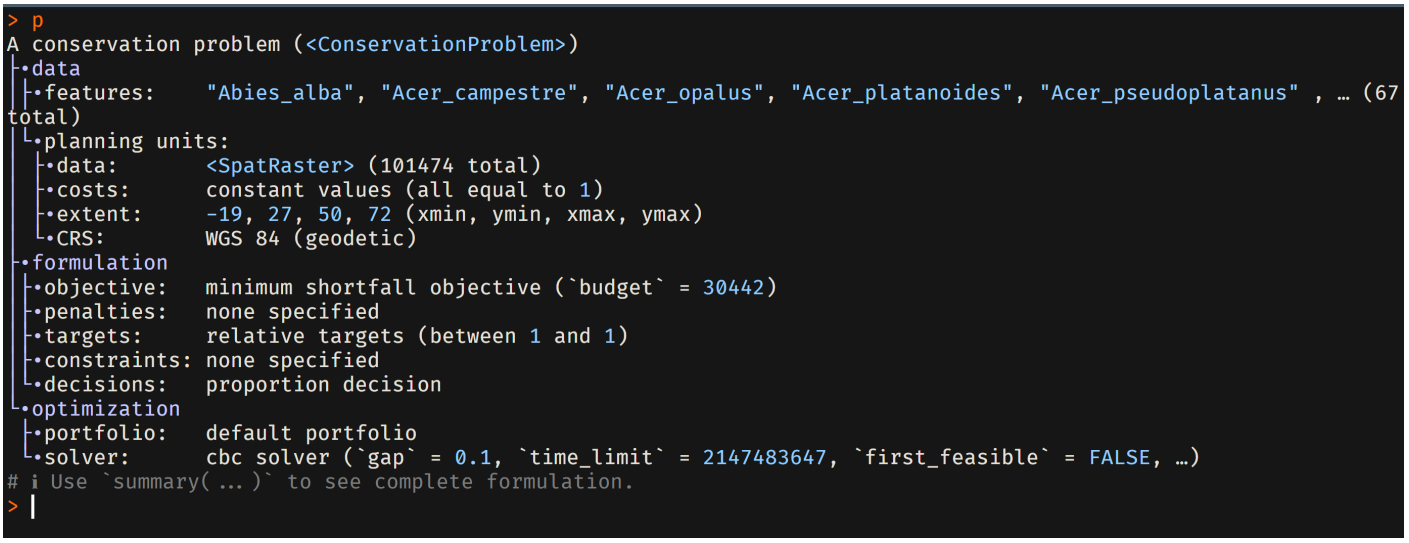



Figure 4.2: The output of a prioritizr object.

As visible the object contains information about the Planning units, including the spatial extent and geographic projection, as well as any features and complexity factors related to the formulation of the problem.

Running **p$summary()** will provide a summary with more detail.

> Object
>
> The prioritizr planning objects contain a range of different functions that can be queried and executed, for example to obtain summaries or specific datasets and parameters contained within. For example object$data will return: (a) features, (b) planning units, (c) an intersection call rij_matrix and more information.

## 4.3 Different datasets for planning

In this tutorial we use throughout gridded datasets. However it should be noted that - internally - *prioritizr* does not operate on spatial files but on tabular data. This is also true when not gridded but vector data are provided.

Why is this relevant to know? When creating a problem with spatial data (gridded or vector files), at the time of problem creation these data are internally converted into large tabular

data. When planning over many species or planning units it can be computational efficient to not have prioritizr, but to do this conversion directly and then supply tabular data.

Typical steps involved here are:

1. Converting gridded planning unit data to a table in long form (row) containing both the cell id, the planning unit id and the cost
2. Convert the features into a long table containing the planning unit id, the feature id (and name) and the amount stored.
3. Intersecting the tables created in step 1) and 2)
4. Preparing any other tables for weights or targets, aligned with the planning unit id.

> **Note**
>
> Another context might be when data has already been formatted for use in another software such as Marxan. Prioritizr is able to directly use the formatted tables prepared for a typical Marxan application, thus making it easy to switch between software.

# Part III

# Solving a problem

# 5 Solving and interpreting solutions

In this section we will create a solution to the previously set up planning problem (object **p**, see Chapter 4). Specifically we will solve the problem, analyse its outputs and calculate a range of metrics and indicators describing it.

## 5.1 Find a solution for a conservation problem

In the previous section we defined a conservation problem based on a planning unit file, features, the specification of an objective function and decision variable. We also added a solver which we can now use to find a solution to the problem specified.

```
library(rcbc) # Load the library for the solver just to be sure!
# solve and create the solution
s <- solve(p)
```

Running this code will create a lot of output in the terminal and different solvers make different outputs and take different length of times to create solutions (they are effectively external software of varying sophistication). You can pipe in a different solver (such as **add_highs_solver()**) to test this out. See also Hanson *et al.* (2019)

## 5.2 Plot the solution

The output of the solved problem from above is essentially a spatial raster that can be plotted.

```
# plot the solution map
plot(s,  col = viridisLite::mako(n = 10, direction = -1), axes = F)
```

With the specified objective function, can you summarize the amount of area contained in the solution? How much would you expect?

## 5.3 Calculate performance evaluation metrics

Now that we have created a solution and visualized it, the next obvious question would be: How good is it? What are we conserving and what maybe not? These are critical questions for any SCP application and different problem formulations will achieve different levels of representation. Performance metrics are usually used to answer such questions in any SCP workflow, and they can assess a solution based on its spatial distribution and/or the features conserved within.

During the problem setup we defined a set of targets for each feature, so naturally a question could be for how many species we reach the target and also how far are we off (see also Jantke *et al.* (2019)). We used a minimum shortfall objective, thus our objective is to minimize the shortfall (e.g. distance) between the amount covered by the feature as constrained by the

budget. Thus we can most feasibly assess the performance of this solution for the species by assessing their representation and their target shortfall.

```
# In Prioritizr there are convenience functions that can summarize the
# coverage of species in terms of amount held
rpz_target_spp <- eval_target_coverage_summary(p, s)                    ①

# mean representation across all species
mean(rpz_target_spp$relative_held)                                      ②

## mean target shortfall across all species
mean(rpz_target_spp$relative_shortfall)                                 ③
```

① This calculates the coverage of the features (taken from the problem) over the solution, also providing the initial amount.

② Here we calculate mean representation, e.g. how much habitat is held by the solution across all features.

③ This calculate the average shortfall, so the difference between held amount and target across features.

# 6 Irreplaceability

The solution quantified above is effectively asked what share (proportion) of land is needed to minimize the most targets for all included features in a complementary way. However we are only looking at the full solution, while in reality some PU might be more or less important in achieving the best or optimal outcome.

One simple way is of course to calculate to step-wise increase the budget available for the planning and then iteratively rank the set of PU in terms of when they enter a solution (see here for more, see Section 9.3 for more). This is usually the most straight-forward approach for objective functions that support the specification of a budget.

Another is to calculate metrics that relate the amount available with the amount in the solution across features, which are often called irreplaceability metrics (See also Kukkala & Moilanen (2013)). For this situation prioritizr supports 3 different ways of quantifying irreplaceability, each with their own caveats.

1. The replacement cost scores are the most precise as they actually relate to whole problem formulation, thus make use of the specified targets and constraints to quantify how "replaceable" a given PU in a solution is. The downside is really the computational effort and this really only recommended for small and moderate sized problems, and may not be feasible for large problems (e.g., more than 100,000 planning units or features).
2. Irreplaceability scores can also be calculated using the method set by Ferrier et al, which can be relatively quicker. Note that this function only works for problems that use targets and a single zone. It will throw an error for problems that do not meet these criteria.
3. Lastly there is the method by Albuquerque & Beier (2015) for calculating rarity weighted richness estimates. Those tend to compare reasonably well to standard prioritizations (e.g. maximum coverage) that scale with differences in range size, yet it does not make use of set targets or any other complexity factors in the problem formulation.

For simplicity and also to not wait an unreasonable amount of time, we here use a smaller geographic subset to illustrate the concept. Specifically we download an outline of the alpine region only and rerun our prioritization just for this region. You can obtain a shapefile of their outline here.

```r
# Load the alps and clip all PU and features to this geographic extent
alps <- sf::st_read('data/AlpineConvention.shp') |>
  sf::st_transform(crs = sf::st_crs(4326))

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

# Recreate and solve the problem
budget.area <- round(0.3 * terra::global(PU_alps,"sum",na.rm=T)[,1] )

p_alps <- problem(PU_alps, spp_alps) %>%
  add_min_shortfall_objective(budget = budget.area) %>%
  add_relative_targets(targets = .3) %>%
  add_cbc_solver() %>%
  add_proportion_decisions()

s_alps <- solve(p_alps)

# to calculate importance scores using replacement cost:
ir1 <- eval_replacement_importance(p_alps, s_alps)

# calculate importance scores using Ferrier et al 2000 method,
# and extract the total importance scores
ir2 <- eval_ferrier_importance(p_alps, s_alps)[["total"]]

# calculate importance scores using rarity weighted richness scores
ir3 <- eval_rare_richness_importance(p_alps, s_alps)

# So we can see that actually only very few PU are highly irreplaceable
# (regardless of method) for this smaller geographic subset
# For ir1 (replacement cost) we actually find that no PU is fully irreplaceable
plot(c(ir1, ir2, ir3), axes = F, col = viridisLite::magma(n = 100, direction =-1))
```

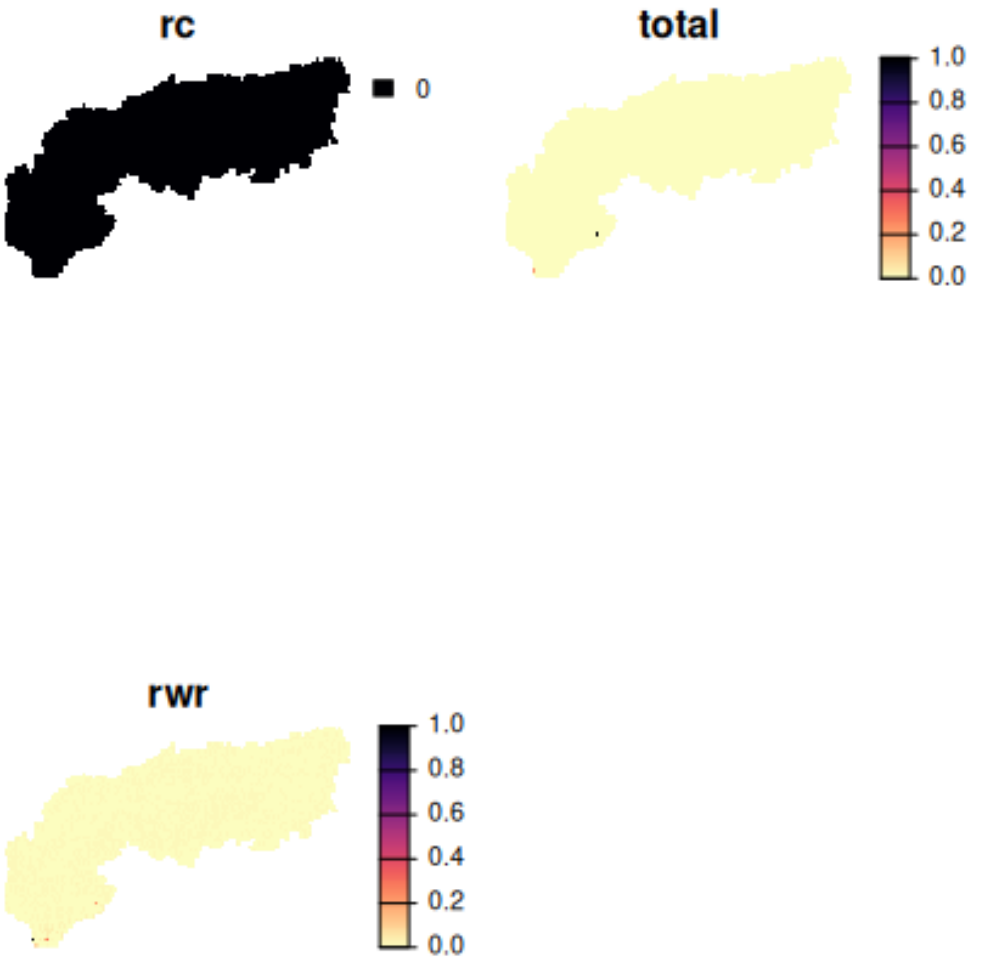



Figure 6.1: Comparison of some irreplaceability metrics

## 6.1 Portfolios

One of the key principles of exact algorithms is to identify solutions that are close as possible to the optimum (e.g. the best possible) as well as being able to quantify the gap between the found solution and an optimum. This differentiates them for example from other prioritization software where often 'selection frequencies' returned (Marxan).

However, in practice it can also be useful to obtain not only the best or optimal solution, but also the next best or a small portfolio of options (like the top 10).

> **Portfolio methods**
>
> There are different portfolio methods (random, shuffle, cut, gap, etc…) available that provide different outputs, note that many portfolio methods are only available for Gurobi. See the help file for alternatives when in doubt.

```
# For this portfolio example we again add create a solution,
# but specificy a cut portfolio with the top 10 solutions.
sp <- solve(p_alps |>  add_cuts_portfolio(number_solutions = 10) )

# Reduce them into one
sp <- Reduce(c,sp)
names(sp) <- paste0("top",1:10)

# These portfolios could again be used to investigate a selection frequency
# in these near-optimal solutions

sp |> sum() |>
  plot(main = "Portfolio selection frequency")
```

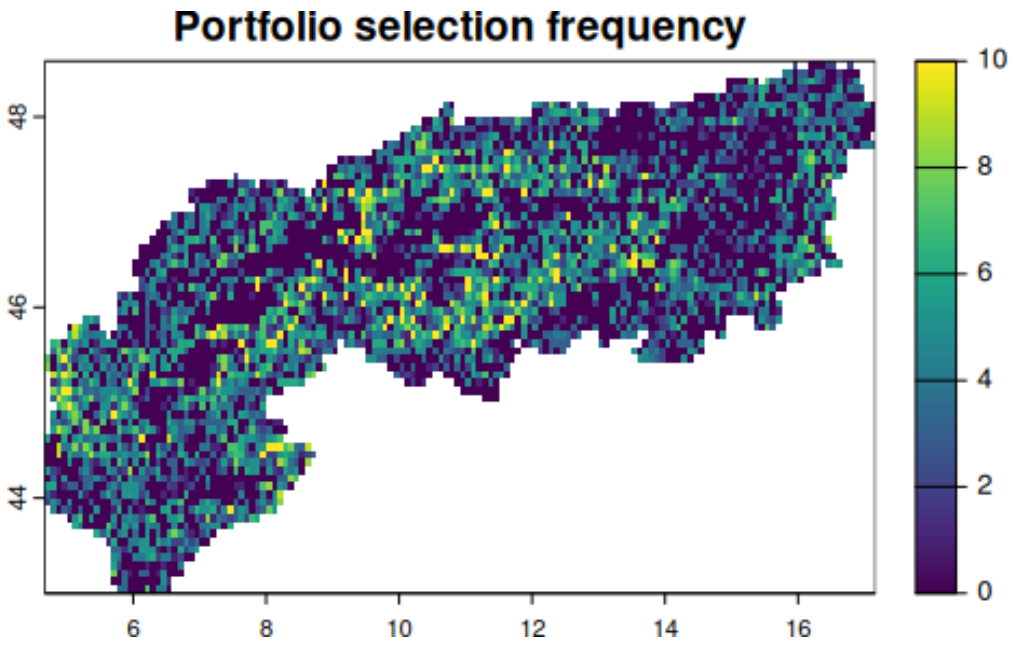



Figure 6.2: Selection frequency of near-optimal solutions

# Part IV

# Adding complexity

# 7 Objective functions

One of the most powerful abilities of mixed or integer linear programming (MILP) compared to other prioritization software approaches is the possibility of readily exchanging objective functions dependending on the question to be answered or the planning objective.In theory many different objective functions could be used in SCP if they can be mathematically formulated (down to complex non-linear optimizations that maximize both area and population abundance of species).

A common distinction among objective functions is whether they make use of a "budget" (e.g. how much total area at most) and/or "targets" (how much of a feature). There are even some objective functions that require neither. Furthermore objective functions differ in their ability of how benefits are accumalted, such as for instance that every little improvement towards a target counts (linear) or whether the whole target needs to be achieved (approaching a step function).

The most classical objective functions are the minimum set and the maximum coverage function (Cabeza & Moilanen (2001)), both of which we introduce below. However other objective functions are possible as well and the *prioritizr* package focusses primarily on a range of common objective functions used in the context of area-based conservation planning. In the examples below we focus on implementation and less on a detailed mathematical description of the objective functions. Here please have a look at the prioritizr help files.

For a good and more recent literature overview and comparison of different objective functions, we recommend the following reading materials (Cabeza & Moilanen 2001; Arponen *et al.* 2005; Beyer *et al.* 2016; Alagador & Cerdeira 2020)

> **!** About the use of targets
>
> Most - but not all - objective functions require the use of targets in some way. Although for example the maximum utility objective function works without targets, it is usually not recommend owing to larger assumptions on how benefits for features accumulate (see also Section 3.8).

## 7.1 Target-based objective functions

### 7.1.1 Minimum set

The minimum set objective function is the most commonly applied on and also the one exclusively supported by the popular Marxan software (see Ball *et al.* (2009), and https://marxansolutions.org/). It does not use budgets, but rather tries to identify the minimum amount of area that would satisfy all targets.

> **Note**
>
> If the feature targets are too ambitious it might not be feasible to find a solution for a problem specified with this objective function. Similarly if all targets are set to 100% it naturally will require all PU where the features are present.

```
# Define a minimum set problem and solve
s1 <- problem(PU, spp) |>
  add_min_set_objective() |>                                    (1)
  add_relative_targets(targets = 0.3) |>
  add_binary_decisions() |>
  add_default_solver() |>
  solve()
```

(1) Defined here. If no targets are specified, an error will be raised.

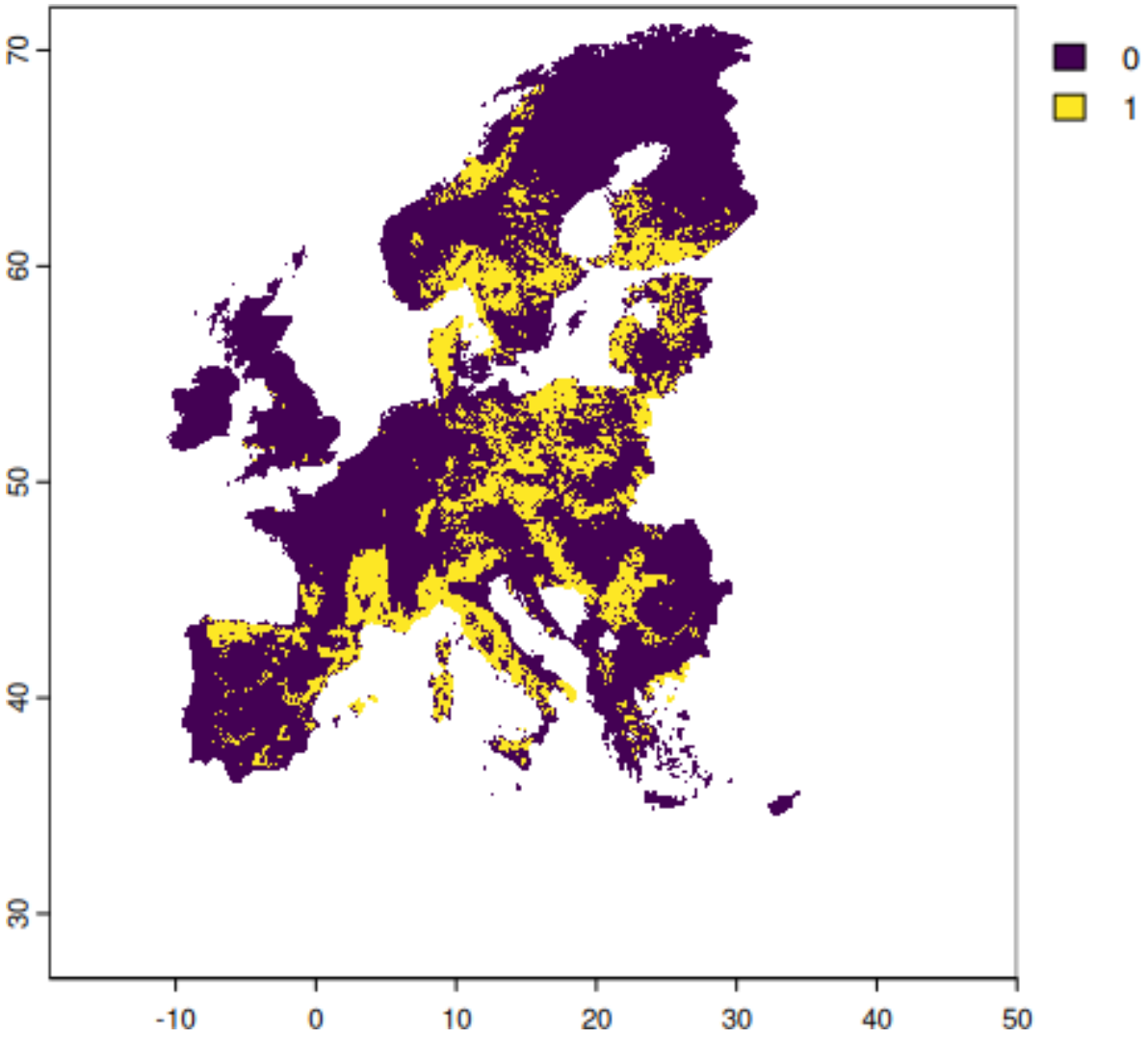



Figure 7.1: A minimum set objective function

### 7.1.2 Maximum coverage objective

The next very commonly used objective function is the maximum coverage objective. This objective seeks to reach as many targets as possible without exceeding a set budget and meeting any cost. Here it differs from the minimum set problem as it is also constrained by a budget and not only by the targets ( Cabeza & Moilanen (2001) ).

Maximum coverage solutions often benefit from additional constraints or penalties that help

to prevent the occurrence of many small fragmented patches.

> **Note**
>
> In prioritizr this objective function is (confusingly) called "add_max_features_objective()"

```
# A dummy 30% of PU area budget
budget.area <- round(0.3 * terra::global(PU,"sum",na.rm=T)[,1])

s2 <- problem(PU, spp) |>
  add_max_features_objective(budget = budget.area)  |>                    (1)
  add_relative_targets(targets = 0.3)  |>                                 (2)
  add_binary_decisions() |>
  add_default_solver() |>
  solve()
```

(1) Sets the objective using the budget specified above.
(2) Dummy targets to secure at least 30% of each feature distribution while staying within the budget. Given the large area budget, this should be easy to solve.

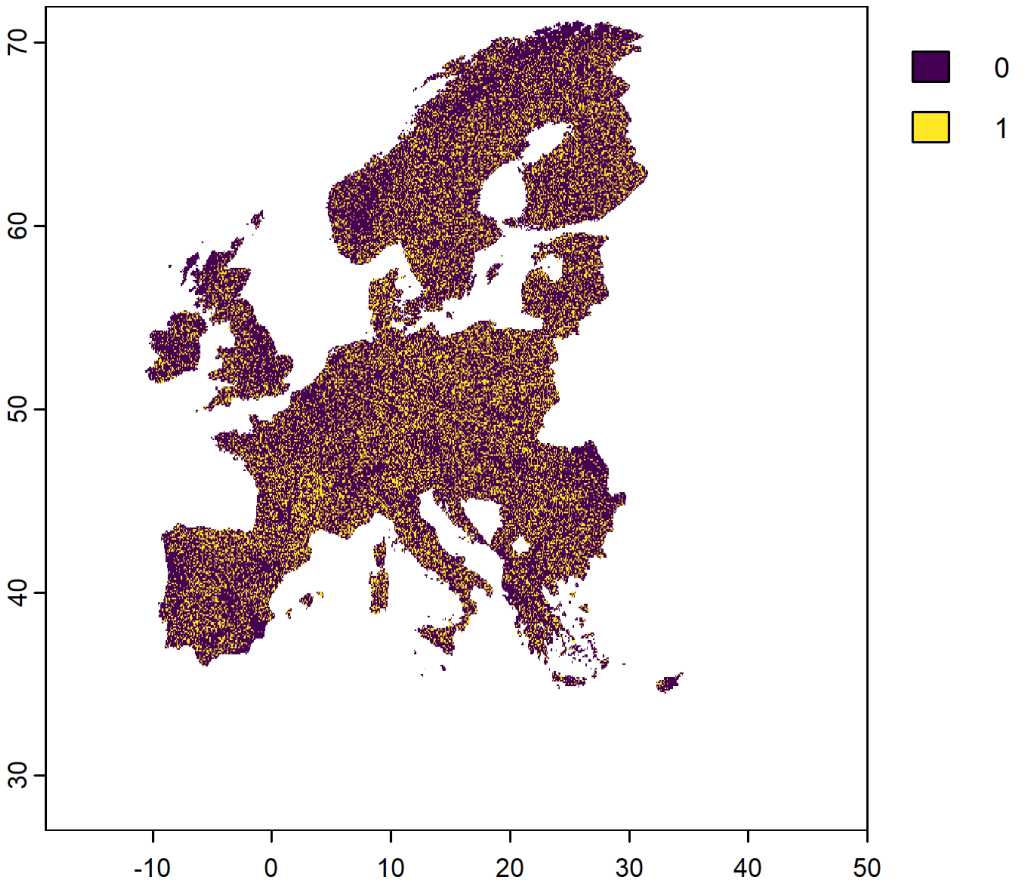



Figure 7.2: Maximum coverage objective function

### 7.1.3 Minimum shortfall

Closely related to the maximum coverage objective function is the (new'ish) minimum shortfall objective. This objective function tries to, instead of maximizing the coverage of features targets, minimizes the difference between the target and amount in the solution ( Arponen *et al.* (2005)). Because of the way it is formulated, it is particular useful for conservation problems that have proportional allocation and intend to have benefits increase linearly (Jung *et al.* (2021)).

In the example below we try to emulate that by trying to secure not only the distribution of tree species but also a reasonable amount of greenness (NDVI). For the latter we set the target to 100%, thus aiming to secure as much as we can together with the other targets.

Sensu Jung *et al.* (2021) it can also be beneficial here to specify weights particular for features that are numerically underrepresented (if the goal is equivalent representation in the solution).

```
# A dummy 30% of PU area budget
budget.area <- round(0.3 * terra::global(PU,"sum",na.rm=T)[,1])

# Define targets
tr <- matrix(nrow = terra::nlyr(spp)+1)                                    ①
tr[,1] <- .3
tr[nrow(tr),] <- 1                                                         ②

s3 <- problem(PU, c(spp, ndvi)) |>                                         ③
  add_min_shortfall_objective(budget = budget.area)  |>                    ④
  add_relative_targets(targets = tr)  |>                                   ⑤
  add_binary_decisions() |>
  add_default_solver() |>
  solve()
```

① We set the manual relative targets here equal to the number of features in the problem. Note the +1 to account for the added NDVI.
② Target for the last feature (NDVI) set to 100%, thus will never be reached.
③ Adding both species and NDVI layer here as a proxy of vegetation greenness
④ The minimum shortfall objective with the 30% budget.
⑤ Adding the manually defined targets by feature from above here.

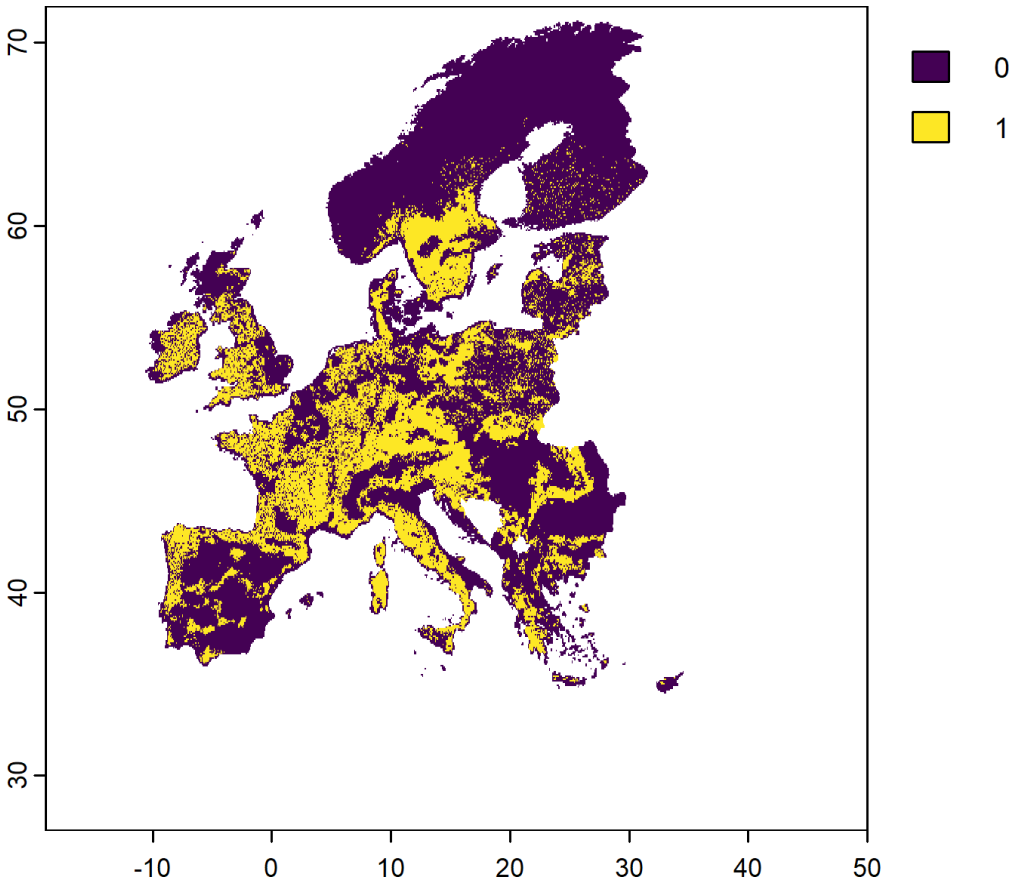



Figure 7.3: A minimum shortfall solution with vegetation amount

### 7.1.4 Minimum shortfall (largest)

This objective function is very similar to the one above, with the notable difference being mathematically that it minimizes the largest target shortfall, instead of the total (weighted sum) of all target shortfalls.

```
# Budgets and targets as for minimum shortfall objective!

s4 <- problem(PU, c(spp, ndvi)) |>
  add_min_largest_shortfall_objective(budget = budget.area)  |>
  add_relative_targets(targets = tr)  |>
  add_binary_decisions() |>
```

```
  add_default_solver() |>
  solve()
```

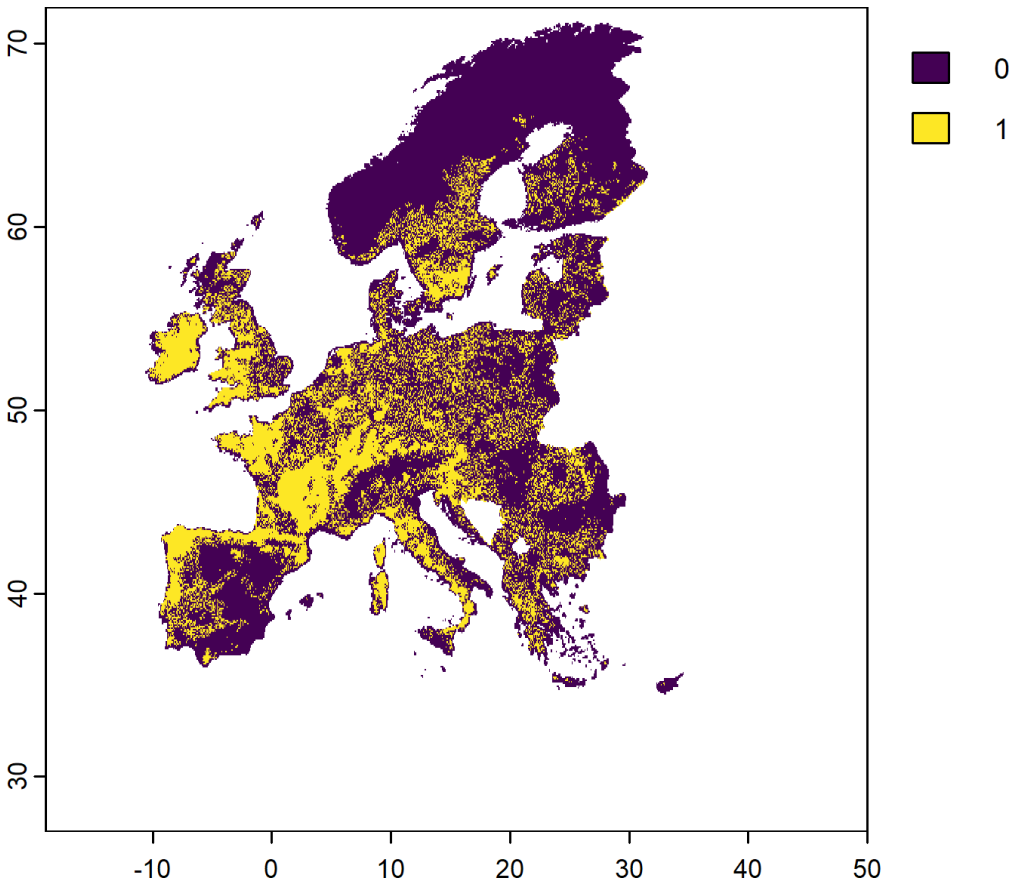



Figure 7.4: Minimizing the largest shortfall instead of the sum across features

### 7.1.5 Other objective functions (phylogenetic)

There are two more objective functions supported by the package that are specifically customized towards phylogenetic data and inter-species relationships. Since those are rather specific, we do not cover them specifically in this tutorial. You can read more about them here and here if of interest.

## 7.2 Non-target based objective functions

### 7.2.1 Maximum utility

The maximum utility objective function is one of the objective functions that does not require any targets. It essentially maximizes utility (or feature abundance) within a given budget across features.

> **Warning**
>
> Because of the way it is set up mathematically it can be biased towards areas where particular common species occur as it does maximize across all features equally. The use of weights, costs or penalities is thus highly recommended.

```
# A dummy 30% of PU area budget
budget.area <- round(0.3 * terra::global(PU,"sum",na.rm=T)[,1])

s5 <- problem(PU, spp) |>
  add_max_utility_objective(budget = budget.area)  |>                    (1)
  add_binary_decisions() |>
  add_default_solver() |>
  solve()
```

(1) Note the difference in name (utility vs coverage). The maximum utility objective does not require targets, only a specified budget.

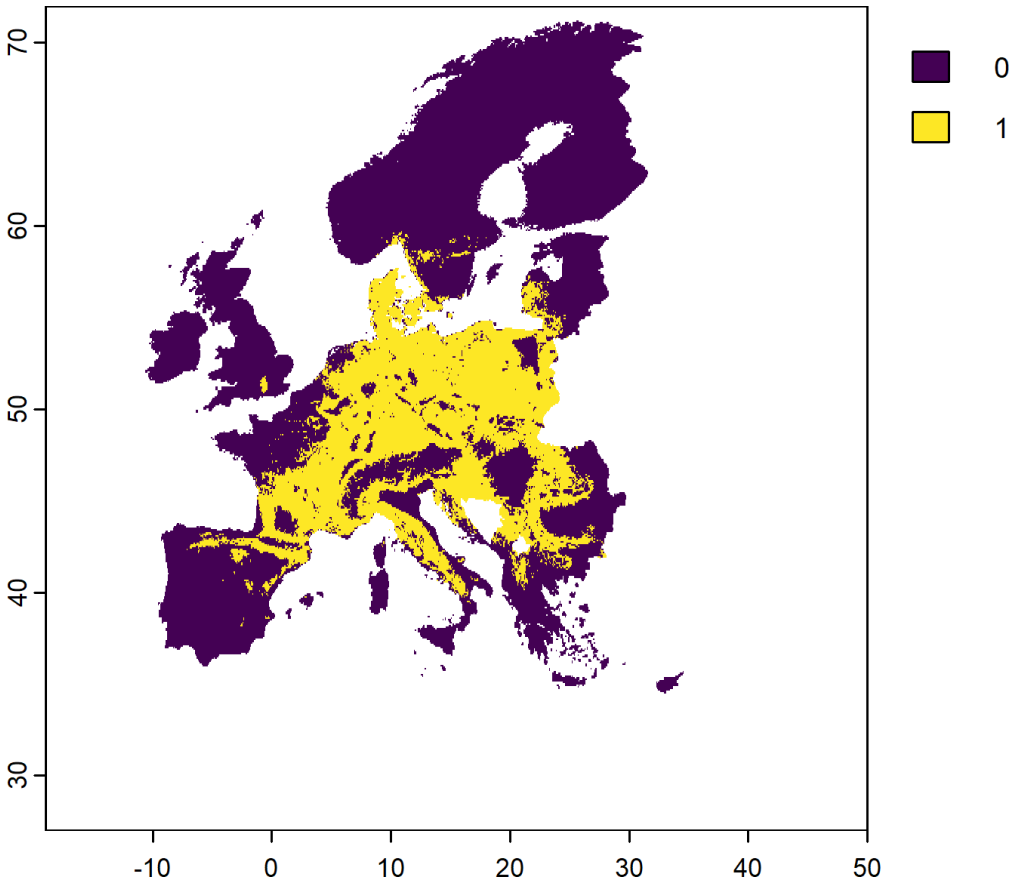



Figure 7.5: Maximum utility objective function solution

### 7.2.2 Maximum cover

Another objective function without any targets is the maximum coverage objective function. This searches for solutions that represent at least one instance of as many features as possible within a given budget.

Since it does not aim to secure as much as possible, only at least a single PU containing the features, this objective function is usually used for SCP problems where features are highly compartmentalized and a large number of categorical and/or continuous layers is used.

> **Note**
>
> Not to be confused with the "add_max_features_objective()" objective function!

```
# We subset the data to the alps here for demonstration purposes
alps <- sf::st_read('data/AlpineConvention.shp') |>
  sf::st_transform(crs = sf::st_crs(4326))

PU_alps <- PU |> terra::crop(alps) |> terra::mask(alps)
PA_alps <- PA |> terra::crop(alps) |> terra::mask(alps)
stPA_alps <- stPA |> terra::crop(alps) |> terra::mask(alps)
spp_alps <- spp |> terra::crop(alps) |> terra::mask(alps)
spp.rcp85_alps <- spp.rcp85 |> terra::crop(alps) |> terra::mask(alps)

# We will modify some features for the use of this objective function.
# Specifically we create reclassified versions of features and protected areas
spp1 <- c(PA_alps * spp_alps); names(spp1) <- paste0("currentpa_",names(spp)) (1)
spp2 <- c(stPA_alps * spp_alps); names(spp2) <- paste0("currentstpa_",names(spp)) (2)
spp3 <- c(PA_alps * spp.rcp85_alps); names(spp3) <- paste0("futurepa_",names(spp.rcp85))
                                                                                (3)
spp4 <- c(stPA_alps * spp.rcp85_alps); names(spp4) <- paste0("futurestpa_",names(spp.rcp85))
                                                                                (4)
# Combine all
spp_pa <- c(spp1,spp2,spp3,spp4)

# A dummy 30% of PU area budget
budget.area <- round(0.3 * terra::global(PU_alps,"sum",na.rm=T)[,1])

s6 <- problem(PU_alps, spp_pa) |>                                               (5)
  add_max_cover_objective(budget = budget.area)  |>                             (6)
  add_binary_decisions() |>
  add_default_solver() |>
  solve()
```

(1) Get the share of the current range per species covered by protected areas
(2) Get the share of the current range per species covered by strictly protected areas
(3) Get the share of the future range per species covered by protected areas
(4) Get the share of the future range per species covered by strictly protected areas
(5) Define the problem with the features created above (number= 268)
(6) Maximum coverage objectives requires only a budget.

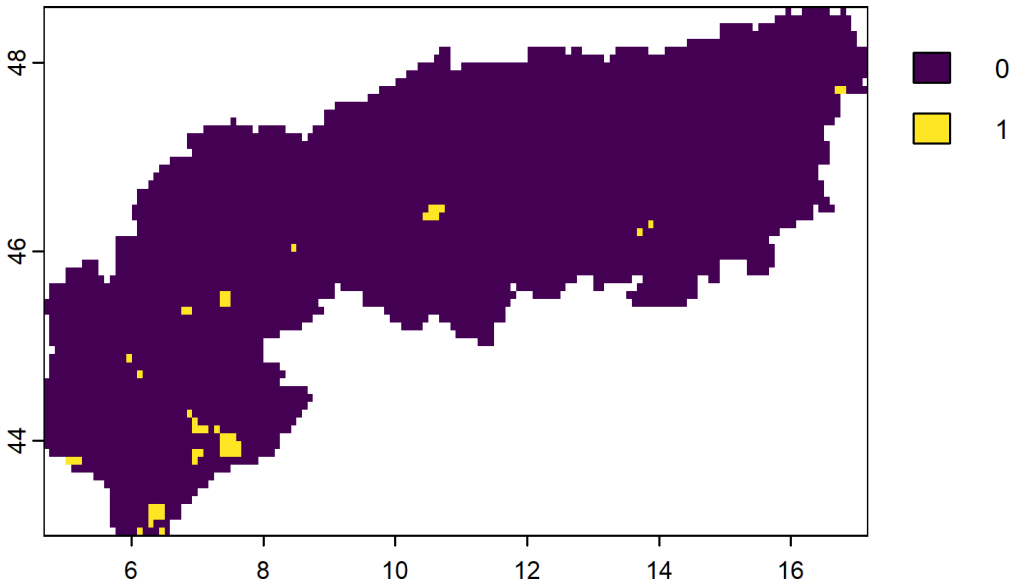



Figure 7.6: Maximum coverage objective function

The PU selected in the solution above ensure that each of the 268 features considered (current and future protected species) are covered at least once somewhere in the study region.

# 8 Adding complexity to conservation planning

In the sections below we will explore some ways of how complexity aspects can be added to a planning problem. With complexity in this context is meant the addition of any planning aspects that go beyond 'default' inputs. For example adding lock-in or lock-out constraints, altering decision variables or feature weights.

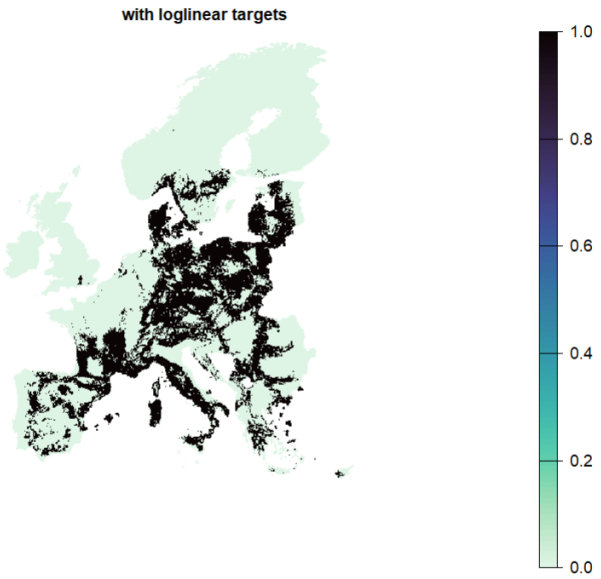
## 8.1 Modify targets

In the previous section we often made use of objective functions that require targets. Targets can be specified in multiple way and added to a conservation problem as manual, relative or absolute targets (the functions here written in the same way).

Another very common ways of specifying targets is the use of a log-linear function as first defined by Rodrigues *et al.* (2004) . Here, instead of assigning equal targets to all features, let's use log-linear targets, so that: features that have a smaller range size (e.g. 10 grid cells) get a target of 100% (their target is their entire range size); and features that are widespread (e.g. with a range size of at least 10,000 grid cells) have a target of 50% of their range size.

```
# Define a problem
p1 <- problem(PU, spp)%>%
  add_min_shortfall_objective(budget = budget.area)%>%
  add_loglinear_targets(10, 1, 10^4, 0.5) %>%                    ①
  add_cbc_solver()%>%
  add_proportion_decisions()

s1 <- solve(p1)

# plot map
plot(s1)
```

① Loglinear targets require a lower and upper target and amount to be specified. They can also handle capped values (such as habitats not larger than XX km2.




> **Tip**
>
> Targets can also be informed by both the range size and the minimum amount necessary to prevent species extinction broadly following IUCN Redlist criteria (see Jung *et al.* (2021) ).

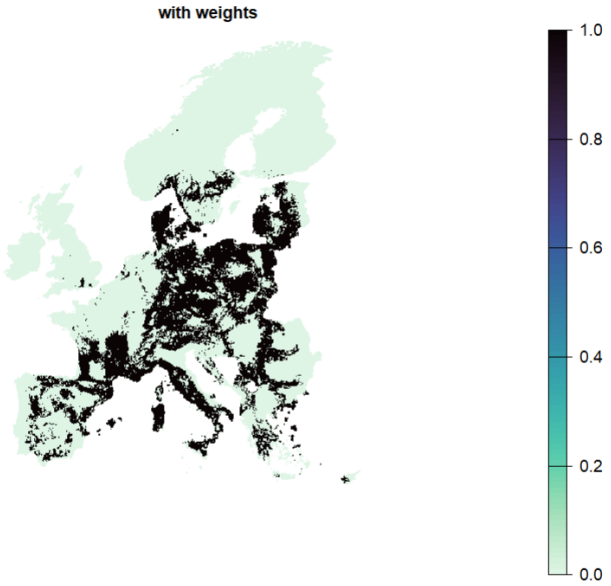
## 8.2 Add feature specific weights

In most cases features in a solution are not equally important. For example there might be genuine national interests in conserving local populations or species might be otherwise more threatened, which would not really well be covered by targets specified on range size alone. In this situations it might be worth changing the weights for these species.

Another common reason for weight is when there is feature imbalance, e.g. some types of features (species) are more common that others (Nature contributions to people). See Jung *et al.* (2021) for more information for this case.

```
# Here we specfiy weights to the tree species as a multiplicator based on redlist criteria.
p2 <- p1 %>%
  add_feature_weights(redlist.trees$weight)
```

```
s2 <- solve(p2)
plot(s2)
```




Notice how in this case the solution does not change drastically, since only a few species are listed as vulnerable. But try and altering a few weights manually (for example multiplying them by 100) just as test to see how solutions change.

## 8.3 Plan for future distributions under climate change

Species distribution model (SDMs) are common inputs to SCP exercises. Just as common is also to use SDMs to make future projections of a given species under a climate scenario. More and more approaches are being developed to make use of such future projections in SCP exercises and in particular with linear programming as well (see for example Alagador & Cerdeira (2017)).

The simplest approach of making use of future projections is to incorporate them as features:

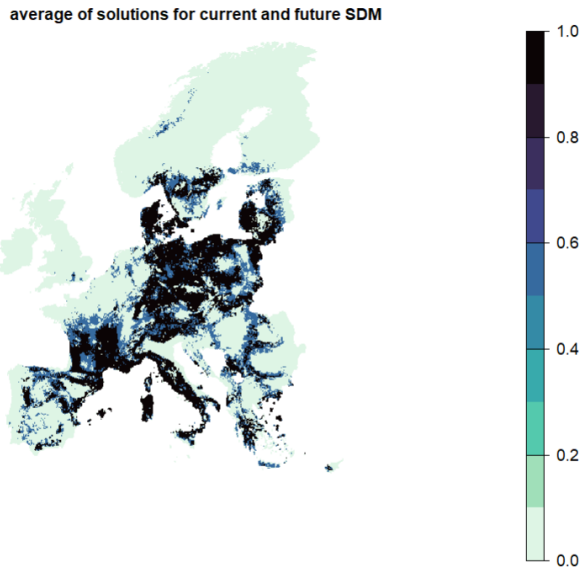
```
# create problem with future distributions as features:
p2_bis <- problem(PU, spp.rcp85) %>%                                    (1)
  add_min_shortfall_objective(budget = budget.area)%>%
  add_loglinear_targets(10, 1, 10^4, 0.5) %>%
  add_feature_weights(redlist.trees$weight) %>%
```

```
  add_cbc_solver()%>%
  add_proportion_decisions()                                                    ②

s2_bis <- solve(p2_bis)
```

① Notice that we use a different feature set here
② We make use proportional decisions here.

Which sites emerge as top priorities for these species, in both current and future climate conditions?

```
mean_s_climate <- mean(s2, s2_bis)

plot(mean_s_climate, col = viridisLite::mako(n = 1, direction = -1), axes = F)
```




You could also try to prioritize for both current and future (and their intersection at once). What do you expect will happen in this case? See also Kujala *et al.* (2013) who developed this conceptual approach.

## 8.4 Adding protected areas

Expanding a reserve network of protected areas is a common research question for SCP studies. Protected areas an be incorporated in various ways in the problem formulation and he we are showing some of them:

### 8.4.1 Locked in and bounded constraints

The simplest way is to 'lock' the area covered by Protected areas (PAs) into the solution. This means that all solutions will have - at a minimum - this amount of area present.

```
p3 <- p2 %>%
  add_locked_in_constraints(PA)

s3 <- solve (p3)
```

In this case the budget cannot be met, because protected areas (small or large) are present in more than 30% of all planning units. The locked in constraints functionality locks in cells that have non zero and non NA values. This functionality is not suitable for European PA at 10x10k resolution: we would need, for example, to change the PA layer to a binary layer with a threshold.

Let's try again with the manual bounded constraints functionality to incorporate the proportion of the planning unit that is currently protected.

```
# create manual bounded constraints dataframe with protected area coverage per planning unit
pa_constraints <- data.frame(pu = cells(PA),                                    ①
                             lower = unname(PA[!is.na(PA)]),                    ②
                             upper = 1)                                         ③

p3 <- p2 %>%
  add_manual_bounded_constraints(pa_constraints)                                ④


s3 <- solve(p3)
```

① grid cell ID
② lower bound that needs to be included in the solution = proportion of grid cell already protected
③ upper bound set to 1 everywhere, so that the whole planning unit can be selected
④ locks in proportional PA coverage per planning unit

### 8.4.2 Towards climate resilient priorities

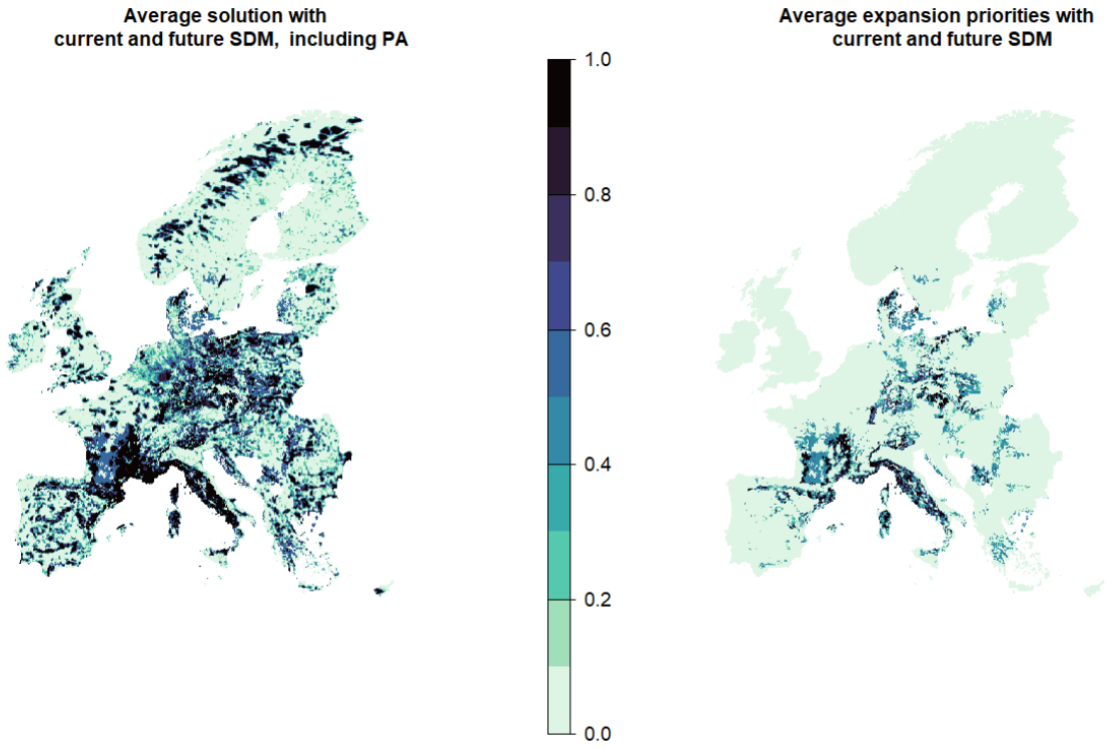
Now, let's find top priorities for the expansion of existing protected areas but that uses projections of species distributions under a future climate scenario (RCP8.5).

```
## create problem with future distributions as features:
p3_bis <- problem(PU, spp.rcp85)%>%
  add_min_shortfall_objective(budget = budget.area)%>%
  add_loglinear_targets(10, 1, 10^4, 0.5) %>%
  add_feature_weights(redlist.trees$weight) %>%
  add_manual_bounded_constraints(pa_constraints)%>%                    ①
  add_cbc_solver()%>%
  add_proportion_decisions()                                           ②

s3_bis <- solve(p3_bis)
```

① to lock in proportional PA coverage per planning units
② Despite specifying this often entire grid cells (planning units) will be selected in the solution rather than a proportion

What areas emerge as climatically resilient protected area expansion priorities for these 67 species? Average across the two solutions that expand on protected areas with current and future distributions:

```
mean_s_climate_PA <- mean(s3, s3_bis)

expansion_climate_PA <- mean_s_climate_PA - PA

plot(c(mean_s_climate_PA, expansion_climate), axes = F, col = viridisLite::mako(n = 10, dire
```




## 8.5 Add locked-out constraints

Locked-out constraints are the exact opposite of Section 8.4.1 explained above. Rather than starting from a given 'baseline' we want to explicitly exclude specific areas from any solution. This functionality can be particular helpful when there is prior knowledge about areas unlikely to be relevant for the planning problem (e.g. densely-population urban areas for protected area planning).

```
p4 <- p3 %>%
  add_locked_out_constraints(locked.out.bin)                    ①

s4 <- solve(p4)
```

① This MUST come AFTER the manual bounded constraints (if using), otherwise locked out constraints are ignored. Note that locked out constraints can sometimes also conflict with the manual bounded constraints, in other words locked in PA might become locked out…

## 8.6 Adding (socio-economic) costs

In prioritizr, the actual "cost" is tied to the value of the planning units, which then determines the budget. The normal or most common way of including costs in the prioritization is thus through the planning unit object. For example, if the aim is not meet a certain amount of area target but rather to select areas until a 'budget' has been reached, then the ideal way is usually to modify the PU file directly (not shown in this tutorial, but can be easily modified).

If however both are relevant questions to be asked, so when we need to express the budget in terms of number of grid cells and not overall socio-economic cost of the solution, we need to include any socio-economic constraints as (linear) penalties.

Linear penalties can be used to avoid the selection of sites with a high value, for example, socio-economic costs if available. Here, we use the human modification index as proxy for the cost of selecting a given planning unit.

```
p5 <- p4 %>%
  add_linear_penalties(penalty = 1, data = gHM)                    ①

s5 <- solve (p5)
```

① Note that when penalty score is set too high, this sometimes prevents the budget area from being met.

Similar as weights, there is no definitive way of what can good penality constants here. In the best case those are informed by ecological or other reasoning and subject to a sensitivity test (try altering the values through a sequence).

> **Note**
>
> Note: if one wanted to express the entire budget of the problem in monetary terms, the costs would need to be included in the planning units data.

## 8.7 Linear penalties with negative penalty score

Linear penalties can also be used with a negative penalty score, to nudge the selection of sites with a high value. For example, one may use linear penalties with a negative penalty score to incorporate pre-defined ecological corridors; known climate refugia; intactness; etc.

Here, we use NDVI as an example, which can be interpreted as a proxy for vegetation health.

```
p6 <- p5 %>%
  add_linear_penalties(penalty = -1, data = ndvi)                    ①

s6 <- solve(p6)
```

① Negative penalty score can be used if we want to nudge selection of sites with high value in the spatial data layer.

> **!** Important
>
> Sometimes adding constraints and penalties will tend to drive the solution much more strongly than the biodiversity features themselves. To limit the influence of the penalty data layer, you can consider decreasing the penalty value.

## 8.8 Decision variables

So far, we solved problems as proportional decisions. Proportional decisions means that proportions of planning units (rather than the whole one) can be selected in the solution. This typically solves much faster and better than binary decisions. Furthermore as you might have observed, it can be quite rare that actual proportions of PU (rather than the whole one) are selected.

Let's try solving the problem with a binary decision instead (i.e. a planning unit gets selected, or not).

```
PA_large <- PA
PA_large[PA_large<0.5] <- 0                                          ①

pa_constraints_bin <- data.frame(pu = cells(PA_large), #  cell ID
                                 lower = unname(PA_large[!is.na(PA_large)]),
                                 upper = 1)

## create problem with binary decision:
p7 <- problem(PU, spp)%>%                                            ②
  add_min_shortfall_objective(budget = budget.area)%>%
  add_loglinear_targets(10, 1, 10^4, 0.5) %>%
  add_feature_weights(redlist.trees$weight) %>%
  add_manual_bounded_constraints(pa_constraints_bin)%>%
  add_locked_out_constraints(locked.out.bin) %>%
```

```
  add_linear_penalties(1, data = gHM) %>%
  add_linear_penalties(-1, data = ndvi) %>%
  add_cbc_solver()%>%
  add_binary_decisions()                                                    ③

s7 <- solve(p7)
```

① need to use different constraints for Protected areas since the 30% budget cannot be met with binary decision + manually bounded constraints
② rewrite problem since we cannot overwrite the previously defined decision variable.
③ Planning units (grid cells) will be either selected, or not selected, in the solution rather than a proportion.

> **Tip**
>
> It is also possible to constrain the proportion of a planning unit directly through a cap throughout. For example when it is seen as too insensible to select PU at all. See the semi-continious decision type for this case

## 8.9 Modify the budget

In previous examples we have often specified a budget. This maximum budget (relevant for some objective functions) can of course also be changed. For example, we might be interested in finding top priorities for 10% strict protection. To do that, we need to change the budget, and the protected area layer, to find priorities that complement and expand on strictly protected areas only (IUCN i and II).

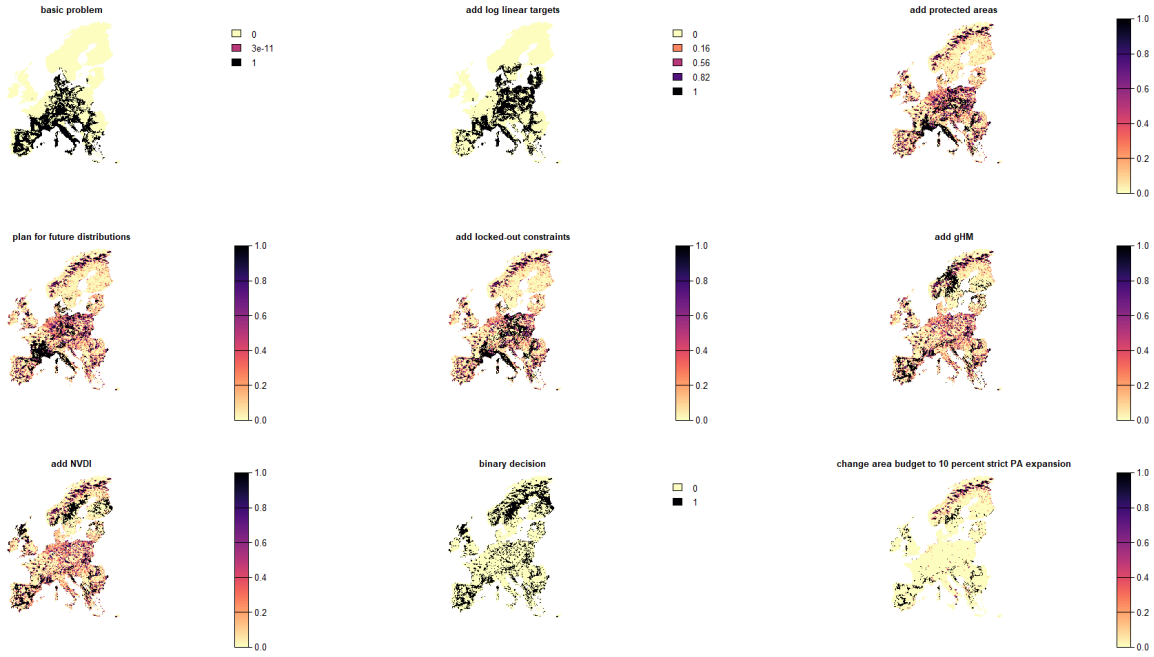
```
## modify the budget: e.g. 10% top priorities that expand on strict protected areas
budget.area <- round(0.1 * length(cells(PU)))


stpa_constraints <- data.frame(pu = cells(stPA),
                               lower = unname(stPA[!is.na(stPA)]),
                               upper = 1)

## create new problem for expansion of strict protected areas: new budget, new manual bounde
p8 <- problem(PU, spp)%>%
  add_min_shortfall_objective(budget = budget.area)%>%
```

```
  add_loglinear_targets(10, 1, 10^4, 0.5) %>%
  add_feature_weights(redlist.trees$weight) %>%
  add_manual_bounded_constraints(stpa_constraints)%>%
  add_locked_out_constraints(locked.out.bin) %>%
  add_linear_penalties(1, data = gHM) %>%
  add_linear_penalties(-1, data = ndvi) %>%
  add_cbc_solver()%>%
  add_proportion_decisions()

s8 <- solve(p8)
```

# 9 Compare and analyse different solutions

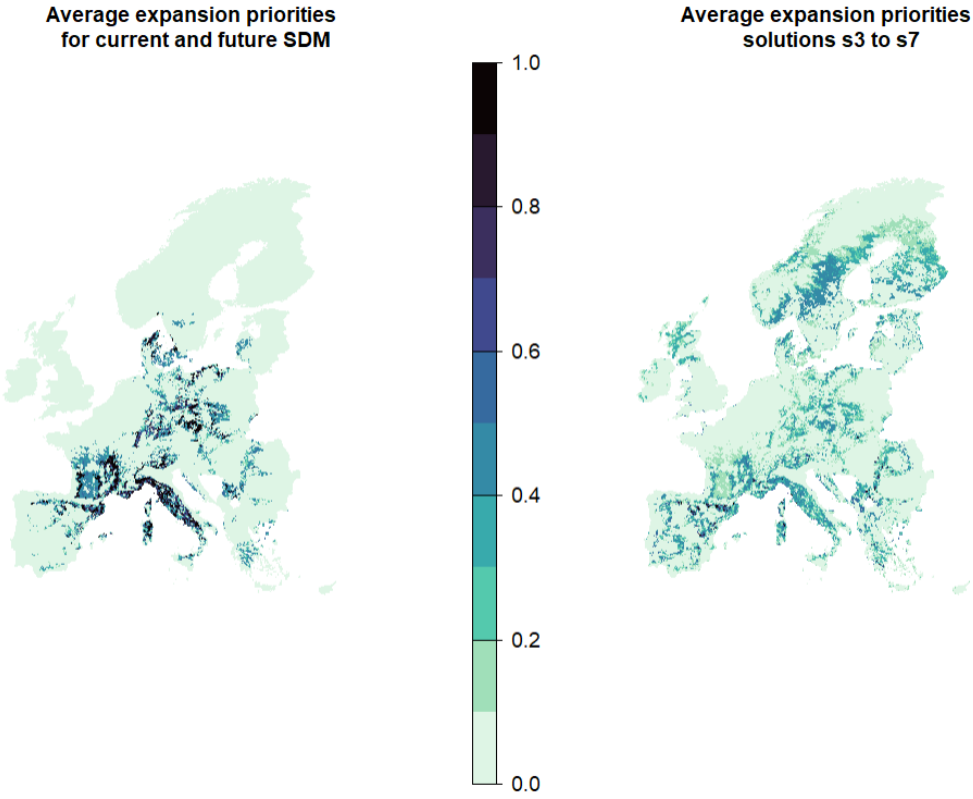
## 9.1 Compare spatial outputs

```
# plot all solutions to compare them
plot(c(s, s1, s3, s3_bis, s4, s5, s6, s7, s8),
     main = c("basic problem", "add log linear targets",
              "add protected areas", "plan for future distributions" , "add locked-out const
              "binary decision", "change area budget to 10 percent strict PA expansion"),
     col = viridisLite::magma(n = 100, direction = -1),
     axes = FALSE)
```



```
##### map safe bets for expansion priorities across all variations of the problems expanding
mean_s <- mean(s3, s3_bis, s4, s5, s6, s7)

exp <- mean_s - PA

exp[exp<0]<-0                                                                    ①

## Compare this map with the one obtained without considering additions in solutions s4-s7 :

plot(c(expansion_climate, exp),  col = viridisLite::mako(n = 10, direction = -1), main = c("
```

① set negative values to zero (these correspond to planning units that were locked out by urban/forestry layer, but that are currently protected)



The influence of input data and methodological choices

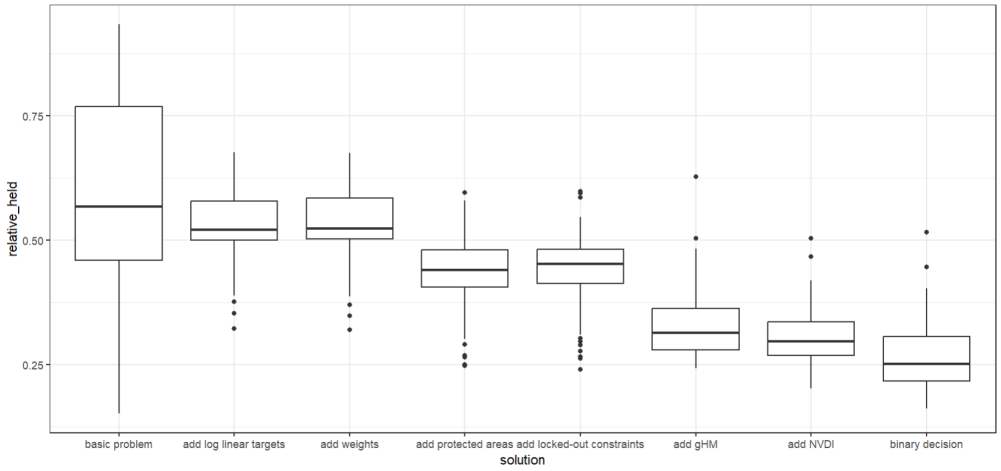
In addition, the more features are included, the more the solution converges, hence it is always better to include as many high-quality features data as possible, to increase comprehensiveness of the planning and obtain an ecologically robust solution.
Remember that the solutions are highly dependent on methodological choices, and specifically on the input data (features, costs), constraints, and the objective function used, as well as the software.
For a review on the influence of different types of data and methodological choices in conservation prioritisation, see Kujala *et al.* (2018)

## 9.2 Compare performance of solutions

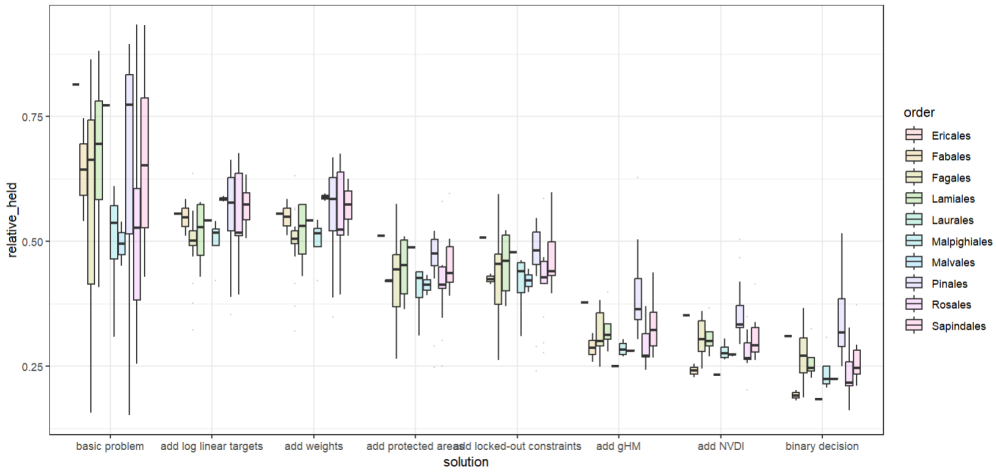
We can compare the performance of solutions for the species by assessing their representation in each solution as well as the target shortfall.

```
library(ggplot2)

# First we create rasterstack of solutions for which you want to compare performance
# here, we compare the solutions that optimize for current distributions within 30% budget a

solutions <- c(s, s1, s2, s3, s4, s5, s6, s7)
names(solutions) <- c("basic problem", "add log linear targets", "add weights",
                      "add protected areas",  "add locked-out constraints", "add gHM", "add
                      "binary decision" )

## analyse representation gains in the different solutions with a given budget
## for individual species
scenarios_performance_species <- data.frame(solution = character(),
                              feature = character(),
                              class = character(),
                              order = character(),
                              relative_held = numeric(),  ## representation: percentage of d
                              relative_shortfall = numeric()) ## shortfall to target: how fa

## loop across solutions to extract representations for species and target shortfall
for (i in 1:nlyr(solutions)){
  cat(paste0(i, " \n"))  # keep track
  rpz.s_i <- eval_target_coverage_summary(p1, solutions[[i]]) ## for each species. Note that
```

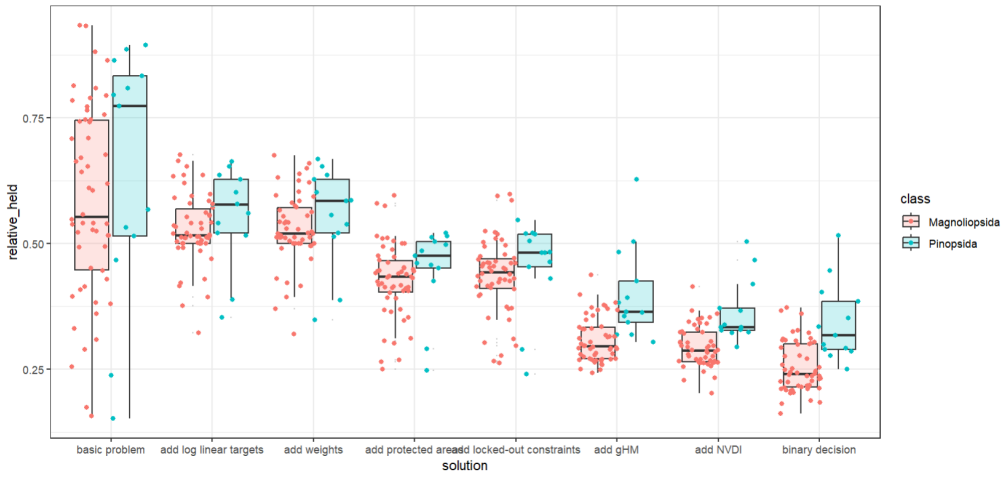
```
  rpz.s_i$order <- redlist.trees$order[match(rpz.s_i$feature, redlist.trees$spp_name)]
  rpz.s_i$class <- redlist.trees$class[match(rpz.s_i$feature, redlist.trees$spp_name)]
  rpz.s_i$solution <- names(solutions)[i]
  rpz_i <- as.data.frame(rpz.s_i)
  scenarios_performance_species <- rbind(scenarios_performance_species,
                           rpz_i[, c("solution", "feature",  "class","order", "relative_held
                           )
}


scenarios_performance_species$solution <- factor(scenarios_performance_species$solution, lev

## compare performance of different solutions in terms of representation
ggplot(scenarios_performance_species, aes(x = solution, y = relative_held)) +
  geom_boxplot()+
  theme_bw()
```



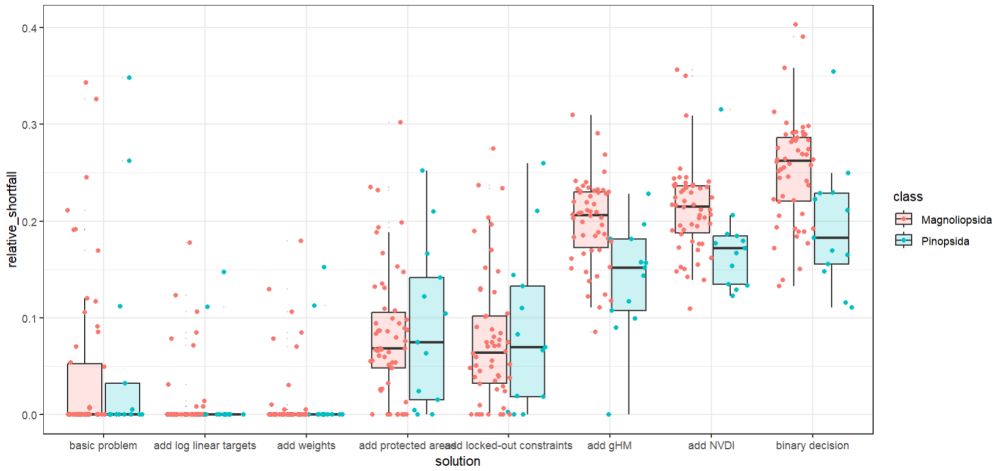

```
# subdivide per groups of species to be more ecologically informative
ggplot(scenarios_performance_species, aes(x = solution, y = relative_held)) +
  geom_boxplot(aes(fill = order), alpha = 0.2, outlier.size = 0)+
  theme_bw()
```




```
## group by Class instead of family
## add jitter points to see individual species representations
ggplot(scenarios_performance_species, aes(x = solution, y = relative_held)) +
  geom_boxplot(aes(fill = class), alpha = 0.2, outlier.size = 0)+
  geom_point(aes(x = solution, y = relative_held, colour = class), position = position_jitte
  theme_bw()
```




```
## compare performance of different solutions in terms of target shortfall
ggplot(scenarios_performance_species, aes(x = solution, y = relative_shortfall)) +
  geom_boxplot(aes(fill = class), alpha = 0.2, outlier.size = 0)+
  geom_point(aes(x = solution, y = relative_shortfall, colour = class), position = position_
  theme_bw()
```




Tip

What do these two performance metrics tell us?

## 9.3 Create a spatial ranking of conservation importance

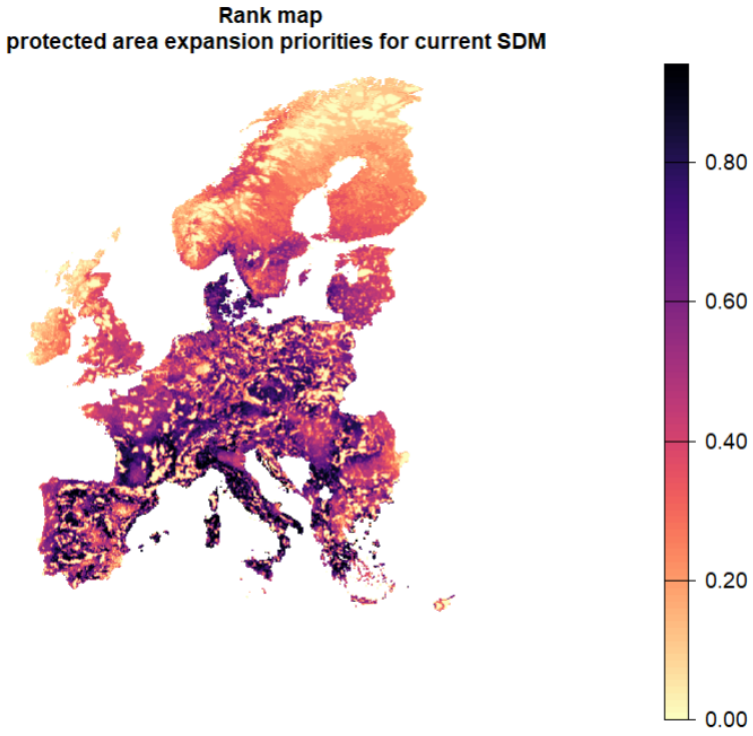
Sometimes one may be interested in the relative ranking in the conservation value of planning units without a fixed budget. but we can make one by solving iteratively while gradually increasing the area in the solution (i.e. the budget). The average of all solutions can give a ranking of the grid cells in the study area in terms of conservation importance.

Let's produce a ranking map with increasing the budget. We will build on solution #3 that expands on protected areas for current distributions, but does not include other constraints. We will start with the existing protected area and incrementally add budget until the whole study area is reached. Then, we can average across all solutions to obtain the ranking.

```
## initialise a raster stack with existing PA to store solutions as budget area increases.
incremental.solutions <- PA

protected.land <-  round(sum(PA[PA>0]))
total.land <- sum(PU[PU>0])

steps <- c(seq(from =protected.land, to = total.land, by = 5000 )[-1], total.land-1)

## skip the first as this is the initial PA layer + add the total land amount
```

```r
## the argument "by" can be decreased for finer ranking.

## Note: this will take a while (1-2 minutes per run)
for (budget.area in steps){
  p_i <- problem(PU, spp)%>%
    add_min_shortfall_objective(budget = budget.area)%>%
    add_relative_targets(1) %>%                                          ①
    add_feature_weights(redlist.trees$weight) %>%
    add_manual_bounded_constraints(pa_constraints)%>%
    add_cbc_solver()%>%
    add_proportion_decisions()

  s_i <- solve(p_i)
  incremental.solutions <- c(incremental.solutions, s_i)
}


ranking.expansion.priorities <- mean(incremental.solutions) - PA

plot(ranking.expansion.priorities, col = viridisLite::magma(n = 100, direction =-1), axes =
```

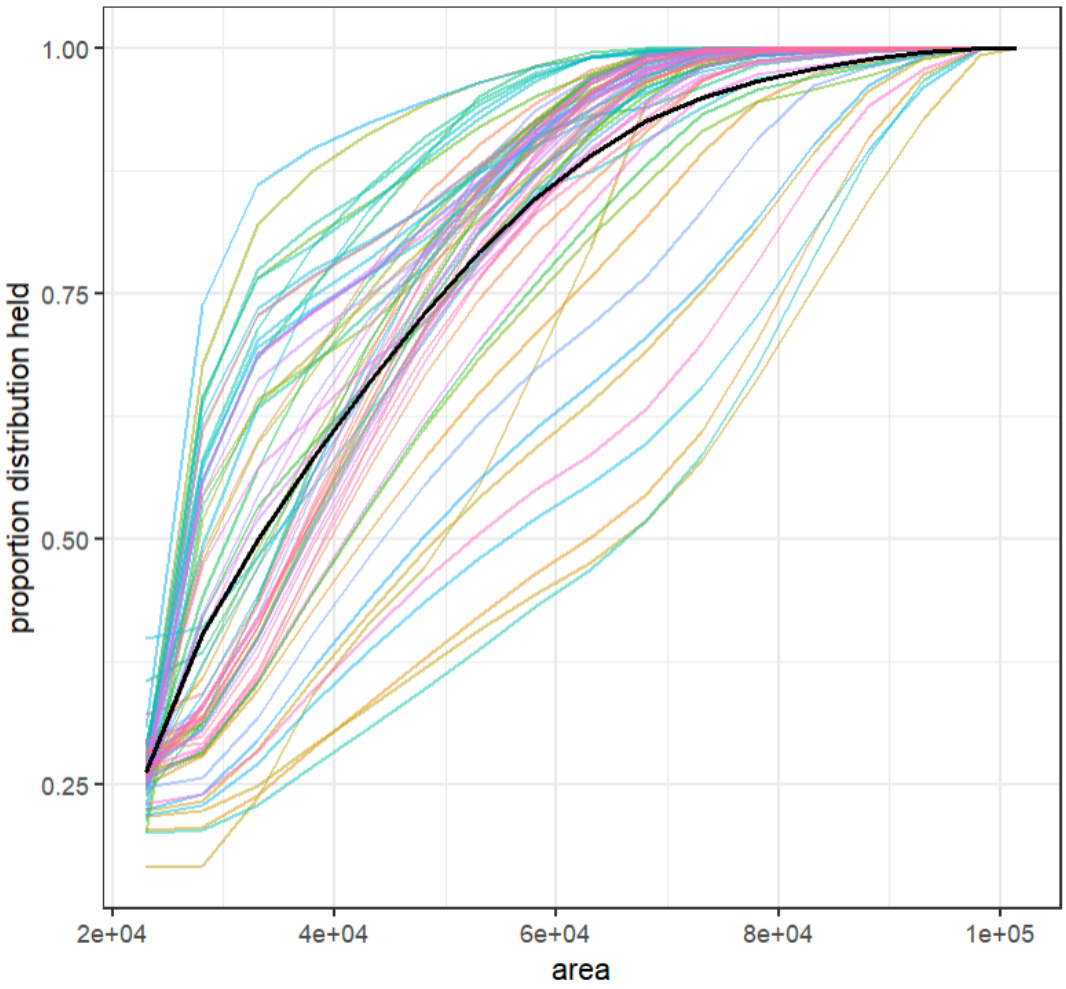
① Here we use relative targets of 1 that are equal for all species, such that each species should be fully represented across its entire distribution. This is because the solution only contains the area that is necessary to meet the targets. If all targets are met within an amount of area that is smaller than the budget specified, the budget is ignored.




The question is now, how does feature representation increase with added area? Let's find out by plotting a performance curve, i.e. representation gains with increasing area, starting from the current representation within protected areas up to the total study area.

```
#### create representation curves

curves <- as.data.frame(matrix(ncol = 3, nrow = 0))

colnames(curves) <- c("area", "species", "relative_held")



incremental.solutions <- c(incremental.solutions, PU) ## add PU with all grid cells value =

steps <- c(seq(from =protected.land, to = total.land, by = 5000 ), total.land) ## add curren

for (n in 1:nlyr(incremental.solutions)){
  rpz_n <- eval_feature_representation_summary(p1, incremental.solutions[[n]])
  df_n <- data.frame(area = steps[n],
```

```
                    species = rpz_n$feature,
                    relative_held = rpz_n$relative_held)
  curves <- rbind(curves, df_n)
  }



## now plot the curves for each species + the mean
ggplot(data = curves, aes(x = area, y = relative_held, colour = species))+
  geom_line(alpha = 0.5) + ## one line per species
  stat_summary(fun.y=mean, colour="black", lwd = 0.9, geom="line") + ## plot mean on top in
  scale_colour_discrete(guide = "none")+ ## hide the legend
  ylab("proportion distribution held") +
  theme_bw()
```




Here, we only ranked the area that is currently non protected. It would be possible to create a rankmap and performance curves considering the whole landscape, including currently protected areas. What would you expect the performance curves to look like in this case?

**Rank maps and performance curves**

A rank map and associated performance curves are key outputs that Zonation provides automatically with each prioritisation. prioritizr does not automatically make these outputs as prioritizr most often enables to solve an optimisation problem under a fixed budget. But we can indirectly create these two useful outputs with prioritizr by iterating over increasing budget area.

# Part V

# Advanced topics

# 10 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 (Daigle *et al.* 2020) (Beger *et al.* 2022) (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 four 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. Connectivity penalties that penalize (unconnected) solutions (Alagador *et al.* 2012).
3. Connectivity constraints to (hard) constrain solutions to certain criteria such as proximity (Hanson *et al.* 2022).

4. Connectivity features such present/future layers or connectivity layers (Kujala *et al.* 2013).

## 10.1 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.

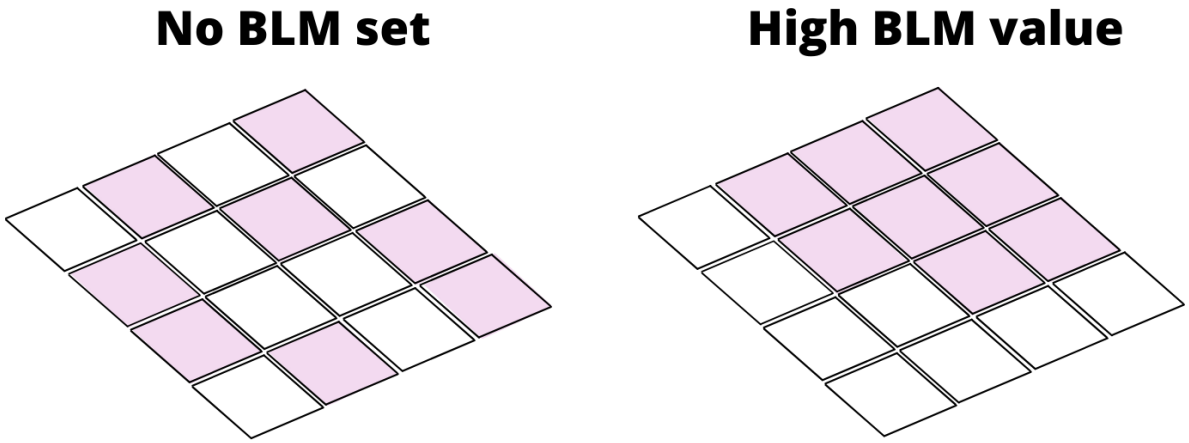



Figure 10.1: Boundary length modifier (BLM), which is effectively a penalty (Source: Marxan solutions)

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.

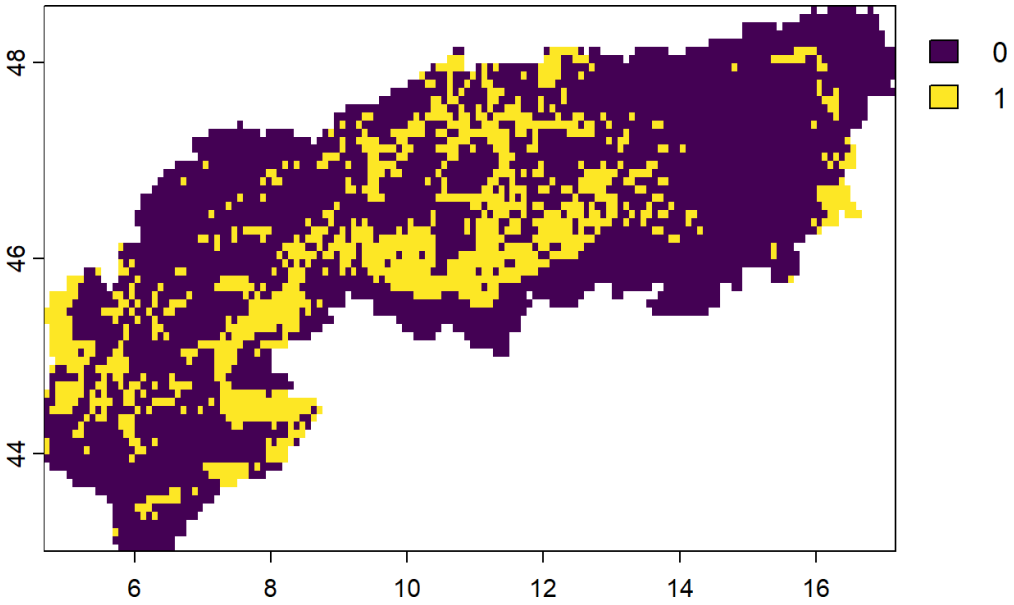
As in previous tutorials we first load our data. However as noted above, we focus on the Alpine region only to make this interpretable. To do so we first crop and mask our PU and feature data to the alps.

```
# 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.

```
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)

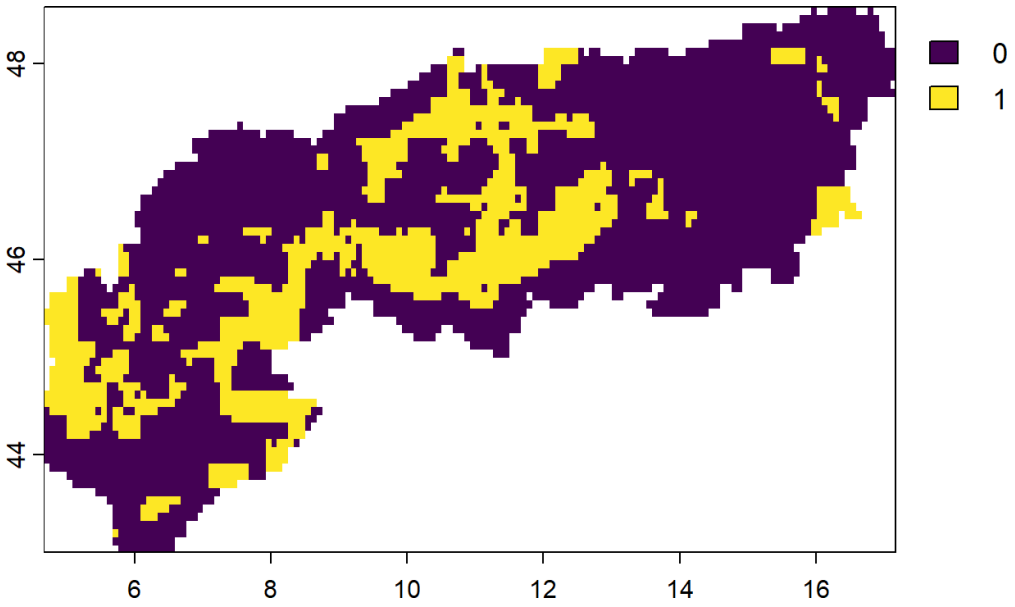



Now lets add some boundary constraints to the same problem.,

```
# 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.

## 10.2 Connectivity penalties

Another more direct way to ingest some connectivity into a problem formulation is to use a certain auxillary layer, for example green infrastructure, (inverse) costs of transversal or connectivity estimates run through software like Circuitscape, as linear penalty. When including connectivity estimates as penalties in conservation planning we usually distinguish between symmetric and asymmetric penalties.

### 10.2.1 Symmetric connectivity penalties

Symmetric connectivity penalties describe information that is non-directional, in other words the same penalties apply when for example a species moves from west to east or from east to west across the study region (see also (Alagador *et al.* 2012)).

In the following example we again define a minimum set problem as before. We then load a pressure layer (the Human modification index) under the assumption that higher human modification values reduce the (structural) connectivity value of a landscape. Again we require a penalty term and it is advised to carefully calibrate this constant in practice.

```
# Define a minimum set problem
p <- problem(PU, spp) |>
  add_min_set_objective() |>
  add_relative_targets(targets = 0.3) |>
  add_binary_decisions() |>
  add_default_solver()


# Load the Human Modification index and clip to the alps
HM <- rast("extdata/gHM.tif") |> terra::crop(alps) |> terra::mask(alps)

# Now prepare the connectivity matrix and rescale
bm <- connectivity_matrix(PU,HM)                                    ①
# rescale matrix
bm <- rescale_matrix(bm)                                            ②

# Update the problem formulation and solve with a small penalty.
s_con1 <- p |>
  add_connectivity_penalties(penalty = 1e-4, data = bm) |>          ③
  solve()
```

```
plot(s_con1)
# It also possible to evaluate the connectivity values via
eval_connectivity_summary(p,s_con1, data = bm)
```

① This command calculates a cross-product between the Planning unit and a pressure layer
② Rescaling is usually necessary to achieve better convergence
③ The Penalty constant chosen reflects the magnitude of influence dedicated to this layer.

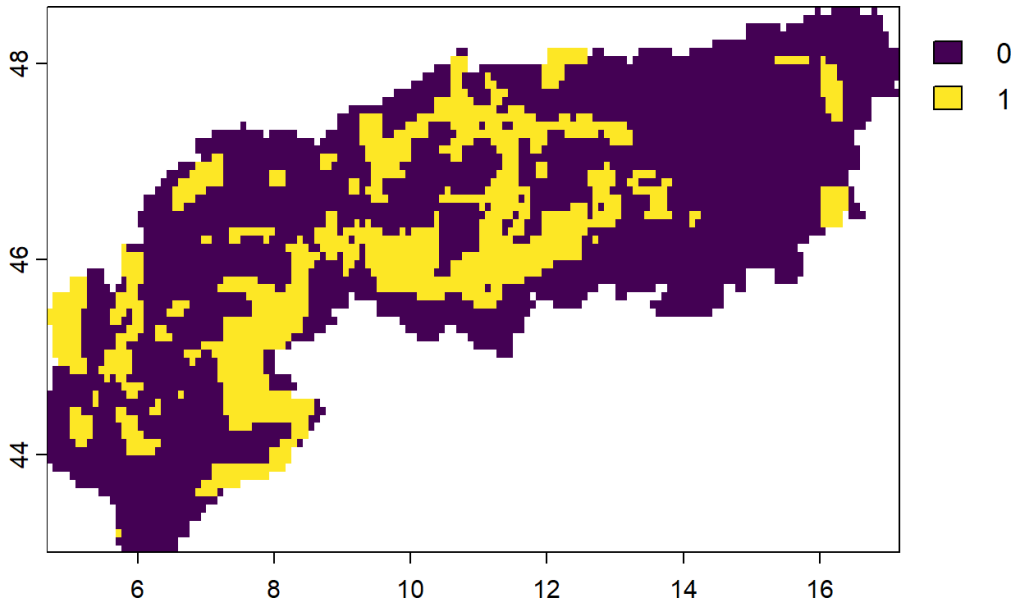



Figure 10.2: Prioritization with symmetric connectivity penalties

**Influence of penalty values**

Try changing the penalty parameter. How do the results change? If you encounter unusual results (all values identical) the reason is often an inappropriate penalty.
In real world example it usually recommended to calibrate such quite parameters so as to ensure realistic outcomes. See this vignette for more information on how to do so.

Another alternative approach could be to not use a separate layer, but constrain the area-based prioritization by some prior knowledge about minimum or maximum distance constraints. For example, one can envisage a case where we know that most species are unlikely to disperse further than 10 km from any selected patch. In this case it can be beneficial to avoid prioritizing such areas for conservation to avoid further fragmentation and possibly extinction of local populations.

Let's try it out (Note: this can take quite a bit longer to solve):

```
# Here we precompute a proximity matrix with maximum distance of about ~10km (WGS84 projecti
cm <- proximity_matrix(PU, distance = 0.1)                                          ①
# rescale boundary data
cm <- rescale_matrix(cm)

# Do one with boundary constraints
s_con2 <- p |>
  add_connectivity_penalties(penalty = 1e-4, data = cm) |>
  add_cbc_solver(time_limit = 240,first_feasible = TRUE) |>
  solve()

plot(s_con2)
```

① Note the different command compared to before. This calculates proximity constraints.

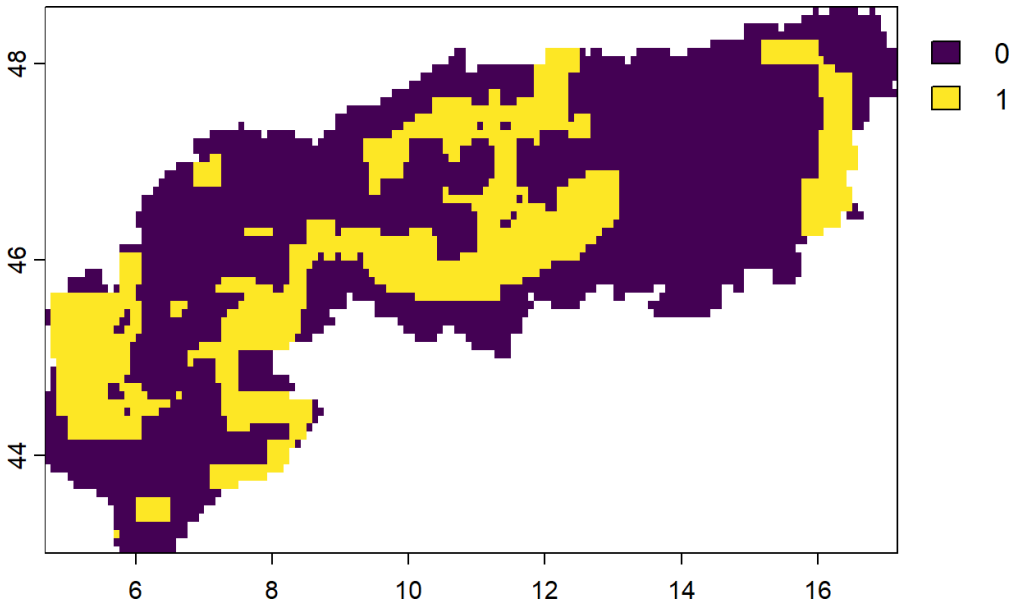



Figure 10.3: Proximity penalties(10km)

> **Tip**
>
> There is also a matrix function called 'adjacency_matrix()'. Can you imagine what this one does?

### 10.2.2 Asymmetric connectivity penalties

Opposed to symmetric connectivity penalties (Section 10.2.1), asymmetric penalties have some kind of directionality. For example in situations where species can only move down PU such as rivers blocked by a dam, or for planning problems with migration corridors (south to north) ((Beger *et al.* 2010)). Adding this penalty to a problem penalizes solutions that have low directional connectivity among PU.

```
# Make a directional dummy layer based on the cell numbers
dummy <- PU
dummy[!is.na(PU)] <- terra::cells(dummy)

# Now prepare the connectivity matrix and rescale
cm <- connectivity_matrix(PU, dummy)                                    ①
# rescale matrix
cm <- rescale_matrix(cm, max = 1)                                       ②

# We only use the diagonal for this simple example, thus going north to south
cm <- Matrix::triu(cm)                                                  ③

# Update the problem formulation and solve with a penalty.
s_asc <- p |>
  add_asym_connectivity_penalties(penalty = 1, data = cm) |>            ④
  solve()

plot(s_asc)
```

① We again create a connectivity matrix using the dummy cell numbers
② Rescale and make sure values are from 0 to 1 for better convergence.
③ We take only the diagonal for simplicity. This effectively removes one geographical dimension (top to bottom).
④ Solve the solution. Note the higher penalty for this dummy example.

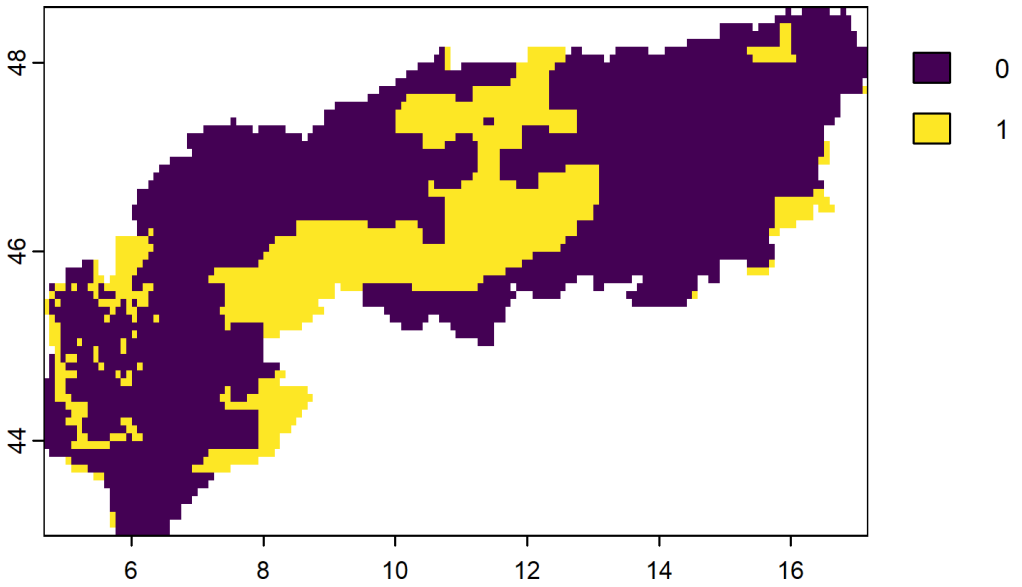



Figure 10.4: Asymmetric connectivity penalty from North to South

## 10.3 Connectivity constraints

So far we have made use of penalties to *nudge* solutions into to being more connected or less fragmented. Penalties however can not guarantee *per se* that a solution satisfies the desired criteria for example having only a few rather than many continuous patches. Constraints force a solution to, regardless of the optimality gap used to generate a prioritization, always exhibit the intended characteristics (or being infeasible).

### 10.3.1 Neighbour constraints

This simply constraint specifies that each selected PU has to have at least X neighbours in the solution.

```
# Define a problem
p <- problem(PU, spp) |>
  add_min_set_objective() |>
  add_relative_targets(targets = 0.3) |>
  add_binary_decisions() |>
  add_default_solver()
```

```
# Obtain only solutions with PU that have at least 2 neighbouring PU
s <-
  p %>%
  add_neighbor_constraints(k = 2) %>%                                    ①
  solve()

plot(s)
```

① Try changing the k parameter to 3 or 4. What happens?

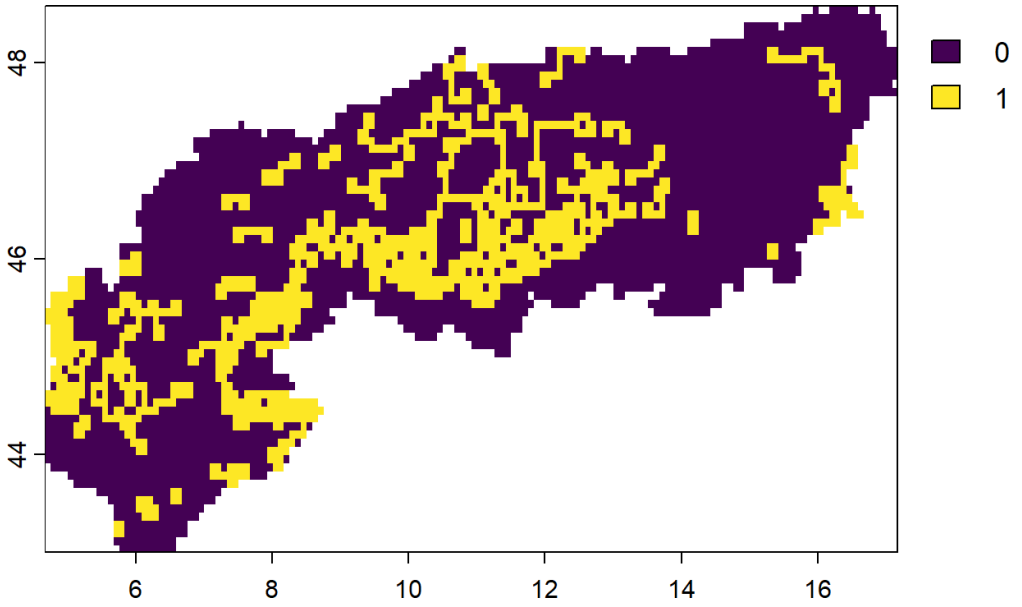



Figure 10.5: A solution with a hard connectivity constraint of having at least 2 neighbouring PUs

### 10.3.2 Contiguity constraints

On the extreme end of the SLOSS (Single large vs several small) debate are single continuous reserves. Such planning solutions can be beneficial for example when the aim is to adequately conserve the most area under large budget constraints. For such cases prioritizr supports so called contiguity constraints, which form a single large reserve instead of multiple.

Contiguity constraints are very time-consuming to solve and an installation of a commercial solver (like Gurobi) is highly advised.

```
# create problem with added contiguity constraints and solve it
s2 <-
  p |>
  add_contiguity_constraints() |>
  add_relative_targets(targets = 0.1) |>
  add_gurobi_solver(time_limit = 2400, first_feasible = TRUE) |>
  solve()
```

### 10.3.3 Linear constraints

Linear constraints are not directly linked to connectivity, but can in theory used for this purpose (and more). Linear constraints simply specify that the solution has to satisfy a criteria, such as for example having at least XX% of area or covering at least YY% of 'connectivity' features. They are thus quite similar to including connectivity as a feature (Daigle *et al.* 2020) (see also below for connectivity features), but are implemented directly as constraints.

For example, in this problem formulation we constrain the solutions to only those that also contain a certain (admittedly) arbitrary amount of 'greenness' (quantified by the NDVI).

```
# Load and clip the ndvi layer
ndvi <- rast("extdata/ndvi.tif") |> crop(alps) |> mask(alps)

# The threshold for linear constraints. We want at least this much!
threshold <- global(ndvi, "sum", na.rm = TRUE)[[1]] * 0.3

# Update the solution.
s3 <-
  p |>
  add_linear_constraints(
    data = ndvi, threshold = threshold, sense = ">="                    ①
  ) |>
  solve()

plot(s3)
```

① We specify a greater/equal sense here. Different directions such $<$ or $<=$ are also possible.

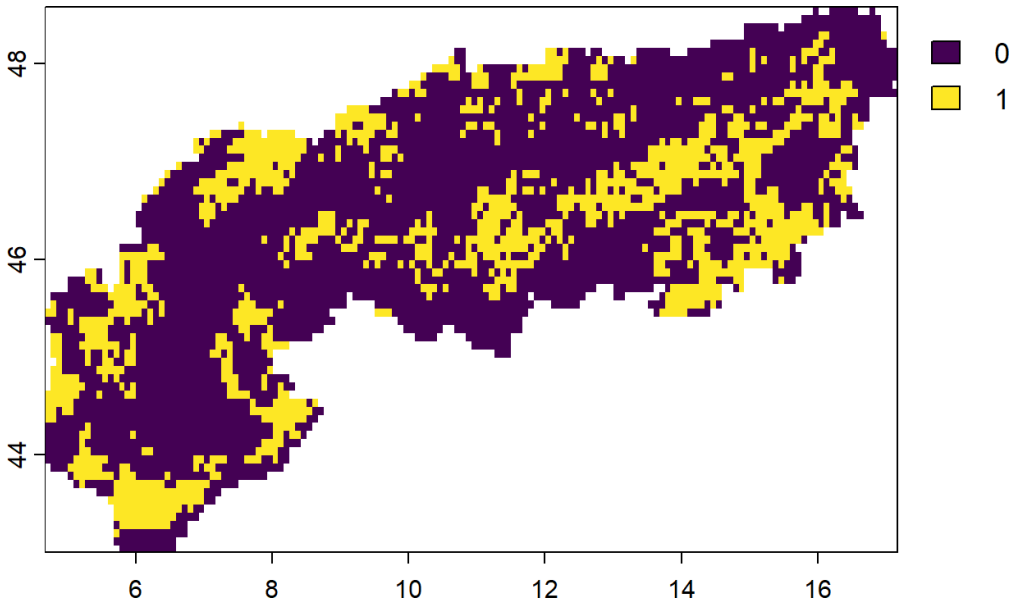



Figure 10.6: A prioritization including certain minimal amounts of greenness as constraint

Can you think of a reason why it might be beneficial to modify the input layers beforehand? Consider that it can incurs costs (in terms of area) to select PU as part of the solution.

> **Tip**
>
> Linear constrains are extremely flexible and can be used to constrain priorities into many directions. For example, with them it is easily feasible to obtain a solution that satisfies at least 10% of total area over the studyregion, while maximizing target achievement.

## 10.4 Connectivity features

Another, relatively straight forward way, to 'account' for connectivity is to directly add features representing connectivity *per se* and ensure that solutions conserve not only the areas a species occurs in but also the area it transverse through. For example (Kujala *et al.* 2013) considered both current and future projected distributions of species (constrained by dispersal distance) to identify potential stepping stones or refugia in response to climate change. For a comprehensive overview see also the recent work on climate-smart metrics for conservation planning (Buenafe *et al.* 2023).

As an example here we aim to identify the top 'priorities' that account for present as well as future distributions of species in a simplified manner. This approach can certainly be improved further, for example by considering dispersal constraints or weights of present against future distributions (discounting), but illustrates the concept.

```
budget.area <- round(0.3 * length(cells(PU))) ##  30 percent

# Identify the solution for a maximum coverage problem and contemporary only
s0 <- problem(PU, spp) |>
  add_max_features_objective(budget = budget.area) |>
  add_relative_targets(targets = 0.3) |>
  add_binary_decisions() |>
  add_default_solver() |>
  solve()

# Now add the future distributions of the species as well as their
spp.list <- list.files(path = "extdata/SpeciesDistributions/", full.names = T, recursive = T
sppf <- rast(spp.list[grep("rcp85", spp.list)])
# Crop and mask
sppf <- sppf |> terra::crop(alps) |> terra::mask(alps)

# Add to stack
s1 <- problem(PU, c(spp, sppf)) |>
  add_max_features_objective(budget = budget.area) |>
  add_relative_targets(targets = 0.3) |>                                    (1)
  add_binary_decisions() |>
  add_default_solver() |>
  solve()

# Overlay and compare
comb <- s0+s1 |> as.factor()                                                (2)
levels(comb) <- c("no priority", "current/future only", "current and future")
plot(comb, legend = "bottom")
```

(1) Note that we specify identical targets for present/future. Ideally targets are specified by feature rather than flat as done here.
(2) Since the decision variable is binary, we can simply sum the result.

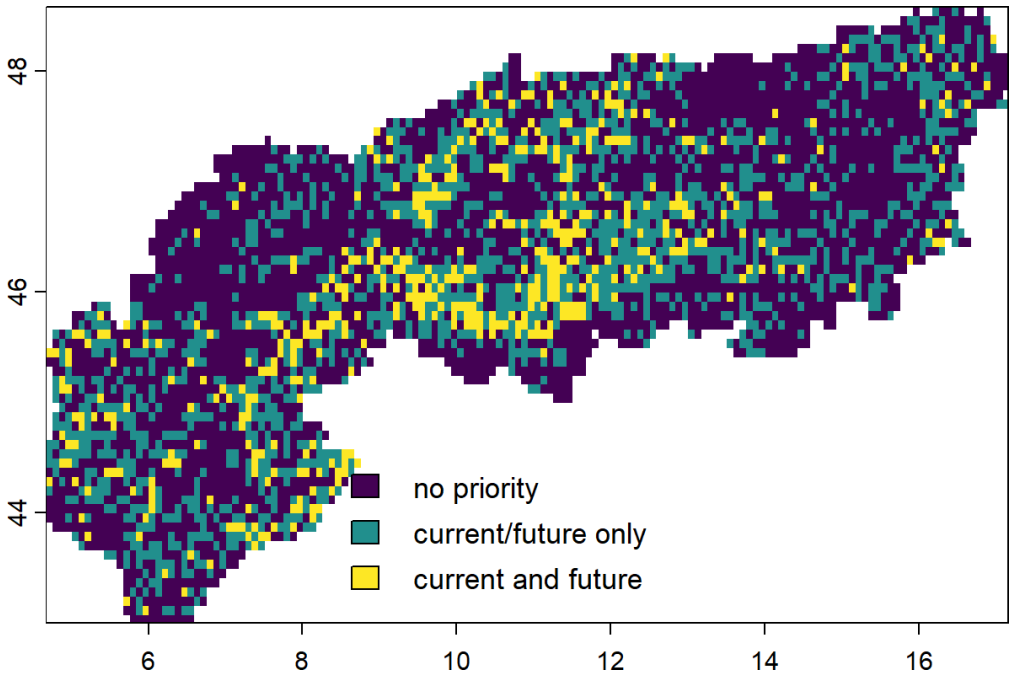



Figure 10.7: Vertical connectivity with future projections

# 11 Adding zones

The technical prioritization in SCP with prioritizr is primarily about allocating area for a given objective to a range of planning units (PUs). Yet in many (if not most) situations there is a need to not only allocate land to a single but multiple outcomes. For example, when we aim to prioritize areas across realms (land and sea) both of which are affected by different costs, features and penalties. Similarly, land could be directly prioritized to certain land system classes (forests, croplands, wetlands) instead of all land, thus increasing interpretability as well as control over the outputs. The concept of having different allocations in the same problem formulation is commonly known as 'zoning' and has been popularized by Marxan (Watts *et al.* 2009).

The prioritizr website contains an excellent tutorial about how different (management) zones can be added to a planning problem, thus we will only cover the essentials here using the testing data that comes with the training course.

For demonstration purposes we focus on the Alpine region for these examples. You can obtain a shapefile of their outline here.

We consider a situation in which we have limited resources (financially or logistically) and would like to identify different priorities for areas with low or with high human modification. To do so we effectively separate our study region into low and high modified management zone.

Targets can be specified per zone individually, but in the solution each PU needs to be allocated to one of the zones or not be selected at all.

```
# Prepare the various layers we use here
alps <- sf::st_read('extdata/boundary_alps/AlpineConvention.shp') |>
  sf::st_transform(crs = sf::st_crs(4326))

hmi <- rast("extdata/gHM.tif") |> terra::crop(alps) |> terra::mask(alps)
ndvi <- rast("extdata/ndvi.tif") |> terra::crop(alps) |> terra::mask(alps)
pa <- rast("extdata/protectedareas.tif") |> terra::crop(alps) |> terra::mask(alps)
PU <- PU |> terra::crop(alps) |> terra::mask(alps)
spp <- spp |> terra::crop(alps) |> terra::mask(alps)
```

```
# Budget total of 30% totally
barea <- terra::global(PU,"sum",na.rm=TRUE)[,1]*0.3

# Respecify targets equal to the number of features
tr <- matrix(nrow = terra::nlyr(spp),ncol = 2)
tr[,1] <- 0.3 # Low use zone target
tr[,2] <- 0.1 # High used zone target

tr[c(10,20,30,40,50),2] <- 0                                              (1)
tr[c(1,2,3,4,5),1] <- 1                                                   (2)

# create problem
p <- problem(c(PU,PU),                                                    (3)
             zones(                                                       (4)
               "low_hmi" = spp,
               "high_hmi" = spp*hmi)                                      (5)
             ) |>
  add_max_features_objective(budget = barea) |>
  add_relative_targets(targets = tr) |>
  add_binary_decisions() |>
  add_default_solver()

s0 <- solve(p)
s0p <- category_layer(s0) |> as.factor()                                  (6)
levels(s0p) <- c("not selected", "low_hmi", "high_hmi")
```

(1) Some features in the highly used zone might also not receive any benefit at all
(2) While for others in the low-used zone we aim to conserve as much as possible (target= 100%)
(3) The same PU layer is used. This could also be separated by zones with different costs.
(4) Here we specify the feature (amount) contributing to each zone.
(5) Note that for highly modified zone we reduce the amount of suitable habitat by the amount of modified land.
(6) To display the multi-zone layer as a single categorical raster.

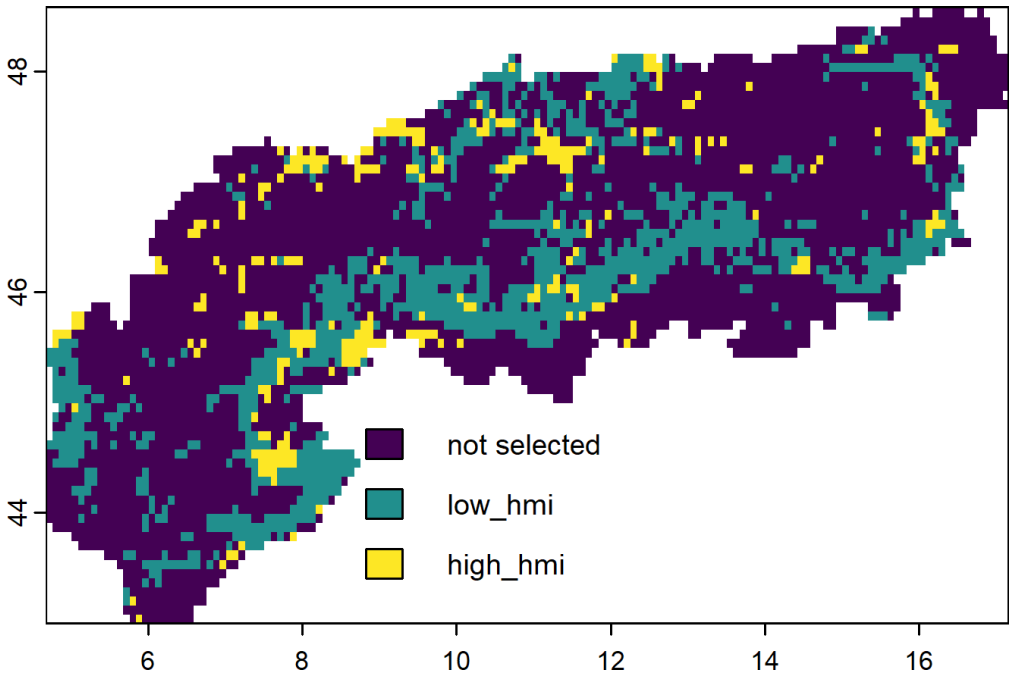



Figure 11.1: Selected features per zone

What do we achieve with each zone? Here we can calculate the representation by zone in terms of the absolute held amount (related also to the total or zone amount of area).

```
reps <- eval_feature_representation_summary(p, s0)

# Apparently some species benefit more than others from co-benefits (distribution covered be
ggplot(reps,
       aes(x = feature, y= absolute_held, fill= summary)) +
  geom_bar(stat = "identity") +
  facet_wrap(~summary)
```

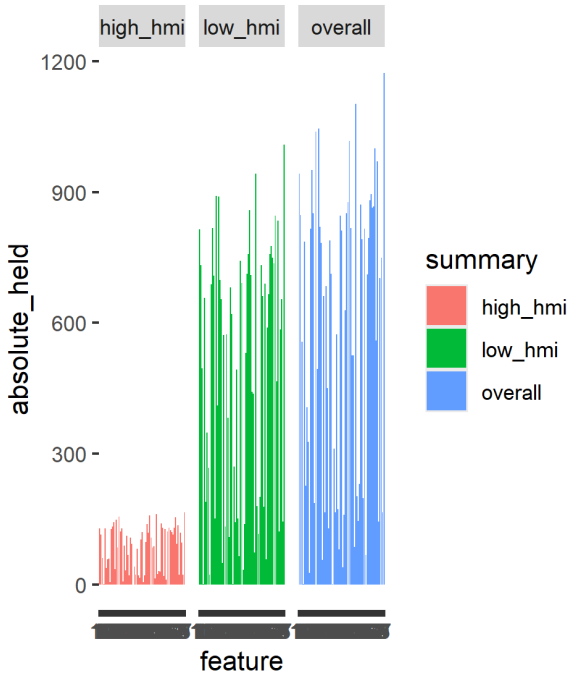



Figure 11.2: Relative amount held overall and per zone

---

## 11.1 Zoning for PA expansion and green infrastructure.

Finally, Let's think of a another example where the aim is to expand the current protected area network, while conserving as much biodiversity and green infrastructure as possible. We again define 2 management zones, one for current protected areas and expansions thereof and one for the remaining land (green infrastructure).

```
# Make a manual bounded constraint data.frame to account for
# fractional shares of current protected areas
mcon <- data.frame(pu = c( terra::cells(PU), terra::cells(PU) ),
                   zone = c(
                     rep("protected_area", length(terra::cells(PU))),
                     rep("gi", length(terra::cells(PU)))
                   ),
                   lower = 0, upper = 1
```

```
)
# Respecify the lower and upper amount of area
mcon$lower[mcon$zone=="protected_area"] <- terra::values(pa, dataframe=T) |> tidyr::drop_na(

# Budget total of 30% totally for the PA zone, 100% for the rest
barea <- c(
  terra::global(PU,"sum",na.rm=TRUE)[,1] * .3,
  terra::global(hmi,"sum",na.rm=T)[,1]
)

# Respecify targets
tr <- matrix(nrow = terra::nlyr(spp)+1, ncol = 2)                    ①
tr[,1] <- 0.3 # Protected area zone flat target
tr[,2] <- 1   # Green infrastructure zone, everything goes
tr[nrow(tr),1] <- 0                                                  ②

# create problem
p <- problem(c(PU,PU),
             zones(
               "protected_area" = c(spp,ndvi),                       ③
               "gi" = c(spp*hmi,ndvi)
             )
             ) |>
  add_min_shortfall_objective(budget = barea) |>
  add_manual_bounded_constraints(data = mcon) |>
  add_relative_targets(targets = tr) |>
  add_proportion_decisions() |>
  add_default_solver()

# Solve
s <- solve(p)

plot(s)
```

① Note the addition plus one here for the greenness layer.
② This specifies a target of 0 for NDVI and the protected area zone, thus no benefits can be gained here
③ Note the addition of NDVI to the features. Also a simple discounting of modified land for the species features

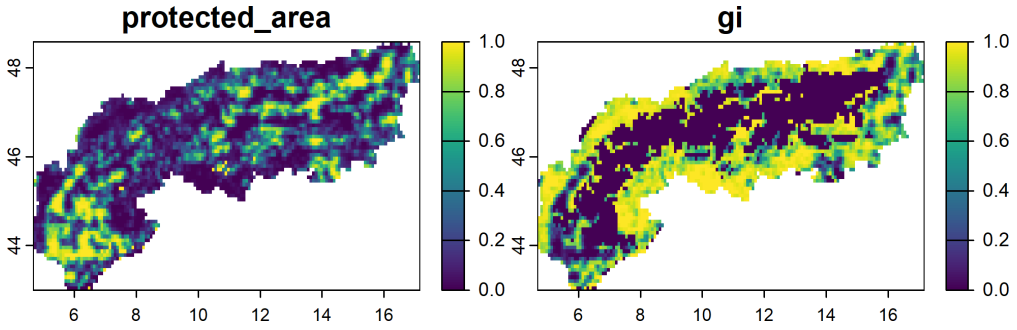



Figure 11.3: Expanding protected areas to 30% and the remainder to Green infrastructure

This solutions expands from the currently protected land in the alps (29%) to (30%). Obviously not much but this also demonstrates that often the level of policy ambition - when focussing on area alone - can be relatively modest. Although in practice even small expansions can be quite challenging in implementation.

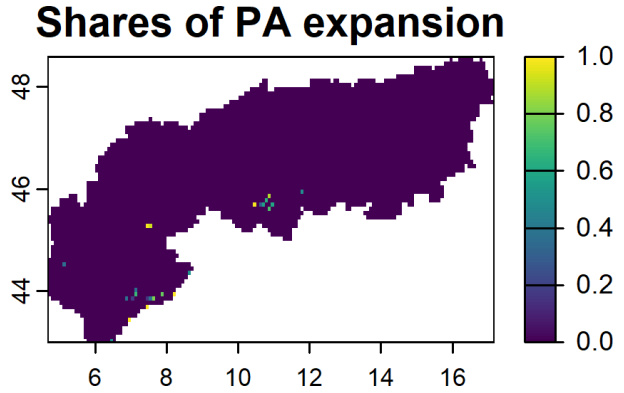



Figure 11.4: Subset of shares that increase from current protected areas

Note that a very similar and more elegant way can be to use linear constraints ( add_linear_constraints() ) applied per zone to limit the allocation of area per zone.

# Glossary

Table 11.1: A glossary of key terms used in this Training course

| Term | Abbreviation if any | Definition |
|---|---|---|
| Allocation | | Synonym use for process taken by the optimization and the decision variable per zone within a PU. Example: Binary decision is to identify expansion or not. The solver allocates PU to a solution based on complementarity. |
| 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. |
| Constraint | | A (often linear) constant or parameter that limits the selection of certain PU as part of the solution. |
| Cost | c | A single or multiple constant typically used in SCP to penalize solutions and any allocation of land to PU. |
| Exact algorithm | | A method to solve mathematical optimization problems using a solver. |
| Feature | | Spatial data representing the distribution of an individual biodiversity unit, such as a species, habitat, ecosystem service, etc. |
| 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. |

| Term | Abbreviation if any | Definition |
|---|---|---|
| Penalty | p | In the context of SCP commonly referring to a constant parameter used to penalize solutions. For example a costing or connectivity matrix. |
| Planning unit | PU | The fundamental unit at which decisions in SCP are realized. Can be of multiple formats such as grid cells or farms |
| Solver | | An algorithm to identify 'solutions' to a mathematical problem. Often available as open- or closed-source software. |
| 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) |
| Zonation | | A SCP software for creating conservation priority rankings and priority maps and tradeoffs. Uses a meta-heuristic approach and benefit functions for ranking. The latest version is Zonation 5. |
| Zone | Z | Zones are spatial or thematic management units over which decisions are made in a SCP problem. Examples: Core zone, Buffer zone and sustainable management zone in a national park. |

# References


Alagador, D. & Cerdeira, J.O. (2017). Meeting species persistence targets under climate change: A spatially explicit conservation planning model. *Diversity and Distributions*, *23*, 703–713.

Alagador, D. & Cerdeira, J.O. (2020). Revisiting the minimum set cover, the maximal coverage problems and a maximum benefit area selection problem to make climate-change-concerned conservation plans effective. *Methods in Ecology and Evolution*, *11*, 1325–1337.

Alagador, D., Trivino, M., Cerdeira, J.O., Bras, R., Cabeza, M. & Araujo, M.B. (2012). Linking like with like: Optimising connectivity between environmentally-similar habitats. *Landscape Ecology*, *27*, 291–301.

Albuquerque, F. & Beier, P. (2015). Rarity-weighted richness: A simple and reliable alternative to integer programming and heuristic algorithms for minimum set and maximum coverage problems in conservation planning. *PloS one*, *10*, e0119905.

Armsworth, P.R. (2014). Inclusion of costs in conservation planning depends on limited datasets and hopeful assumptions. *Annals of the New York Academy of Sciences*, *1322*, 61–76.

Arponen, A., Heikkinen, R.K., Thomas, C.D. & Moilanen, A. (2005). The value of biodiversity in reserve selection: Representation, species weighting, and benefit functions. *Conservation Biology*, *19*, 2009–2014.

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., Linke, S., Watts, M., Game, E., Treml, E., Ball, I. & Possingham, H.P. (2010). Incorporating asymmetric connectivity into spatial decision making for conservation. *Conservation Letters*, *3*, 359–368.

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.

Beyer, H.L., Dujardin, Y., Watts, M.E. & Possingham, H.P. (2016). Solving conservation planning problems with integer linear programming. *Ecological Modelling*, *328*, 14–22.

Buenafe, K.C.V., Dunn, D.C., Everett, J.D., Brito-Morales, I., Schoeman, D.S., Hanson, J.O., Dabalà, A., Neubert, S., Cannicci, S., Kaschner, K. & others. (2023). A metric-based framework for climate-smart conservation planning. *Ecological Applications*, *33*, e2852.

Cabeza, M. & Moilanen, A. (2001). Design of reserve networks and the persistence of biodiversity. *Trends in ecology & evolution*, *16*, 242–248.

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.

Dou, Y., Cosentino, F., Malek, Z., Maiorano, L., Thuiller, W. & Verburg, P.H. (2021). A new european land systems representation accounting for landscape characteristics. *Landscape Ecology*, *36*, 2215–2234.

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.

Jantke, K., Kuempel, C.D., McGowan, J., Chauvenet, A.L. & Possingham, H.P. (2019). Metrics for evaluating representation target achievement in protected area networks. *Diversity and Distributions*, *25*, 170–175.

Jung, M., Arnell, A., De Lamo, X., García-Rangel, S., Lewis, M., Mark, J., Merow, C., Miles, L., Ondo, I., Pironon, S. & others. (2021). Areas of global importance for conserving terrestrial biodiversity, carbon and water. *Nature Ecology & Evolution*, *5*, 1499–1509.

Kujala, H., Lahoz-Monfort, J.J., Elith, J. & Moilanen, A. (2018). Not all data are equal: Influence of data type and amount in spatial conservation prioritisation. *Methods in Ecology and Evolution*, *9*, 2249–2261.

Kujala, H., Moilanen, A., Araujo, M.B. & Cabeza, M. (2013). Conservation planning with uncertain climate change projections. *PloS one*, *8*, e53315.

Kukkala, A.S. & Moilanen, A. (2013). Core concepts of spatial prioritisation in systematic conservation planning. *Biological Reviews*, *88*, 443–464.

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

McCreless, E., Visconti, P., Carwardine, J., Wilcox, C. & Smith, R.J. (2013). Cheap and nasty? The potential perils of using management costs to identify global conservation priorities. *PLoS One*, *8*, e80893.

Rodrigues, A.S., Akcakaya, H.R., Andelman, S.J., Bakarr, M.I., Boitani, L., Brooks, T.M., Chanson, J.S., Fishpool, L.D., Da Fonseca, G.A., Gaston, K.J. & others. (2004). Global gap analysis: Priority regions for expanding the global protected-area network. *BioScience*, *54*, 1092–1100.

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 & Software*, *24*, 1513–1521.

# 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.

> ℹ 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.

```
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).

```
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:

```
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.

# B Frequently asked questions (FAQ)

On this page we list some answers to possible issues or problems encountered when running. See the sub headers for more information.

> **Note**
>
> This page will be updated during the day in case new issues are discovered. In case any issue can not be answered by the information on this site, please get in touch with the course organizers (Martin or Louise).

## B.1 I don't understand the outputs

If you can not interpret the outputs based on the course materials and instructions, please see the help pages of the function (enter ??command in the R console or F1 on your keyboard).

The Prioritizr homepage can also be a quite valuable resource for looking up parameters and instructions. If nothing else, get in touch with the coordinators!

## B.2 I can't install any software

To install R, RStudio and often also R-packages on any Computer (Windows/Linux/MacOS) usually requires adminstrator (or *sudo*) rights.

If you are not able at all to install any or all of the software listed in the installation instructions (@sec-installation), please **get in touch with the course organizers** and we will try our best to find a way forward!

## B.3 My Computer is freezing

Solving particular large conservation planning problems can take quite some computational resources. This becomes especially an issue with larger conservation problems, for example when planning over larger area or more highly resolved planning units (*i.e.* spatial scale).

By solving your planning problem the entire dataset can be bigger than you might anticipate (Number of features times number of planning units times number of constraints) and needs to be processed as a whole. Because of this the amount of memory available on your operating system is usually the limitation. For example, in a global prioritization effort done with ~10km planning units ((Jung *et al.* 2021)), at least 140GB of RAM (Computer memory) was needed to solve the conservation problems.

If - during the solving - your computer suddenly starts to freeze, then you likely don't have enough computational resources to solve the problem formulation. In this case I would recommend to subset the features and PU to a smaller extent, for example using the outline of the Alps from here.

Then subset as follows:

```
alps <- sf::st_read("layer")
layer |> terra::crop(alps) |> terra::mask(alps)
```

## B.4 Solving the problem takes too long

Other than using a faster solver or simplifying the problem (see also suggestion above), there are few options available directly with the solver:

- (Parameter gap in the solver) Increase the gap (Default is 0.1) to a larger estimate. This can result in suboptimal but still feasible solutions which are usually very close.
- (Parameter time_limit in the solver) Increasing this number caps the computation time. Units are in seconds.
- (Parameter first_feasible in the solver) Setting this to TRUE makes the solver return the first feasible solution, which might not be optimal one, but is usually quite close.