# An introduction to Systematic conservation planning with prioritzr

Martin Jung & Louise O'Connor

2024-06-13

# Table of contents

# Preface

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

## What you will learn

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

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

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

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

> **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 2 .

## 1.1 Systematic conservation planning

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

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

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

8. Monitoring the performance and adapting plans where necessary.

### 1.1.1 Key concepts

## 1.2 Exact algorithms and integer programming

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

## 1.3 Tools and software

We stress that prioritizr is not the only software available. For example, Zonation and Marxan are other well-known complementarity-based spatial conservation prioritization software.

# Part II
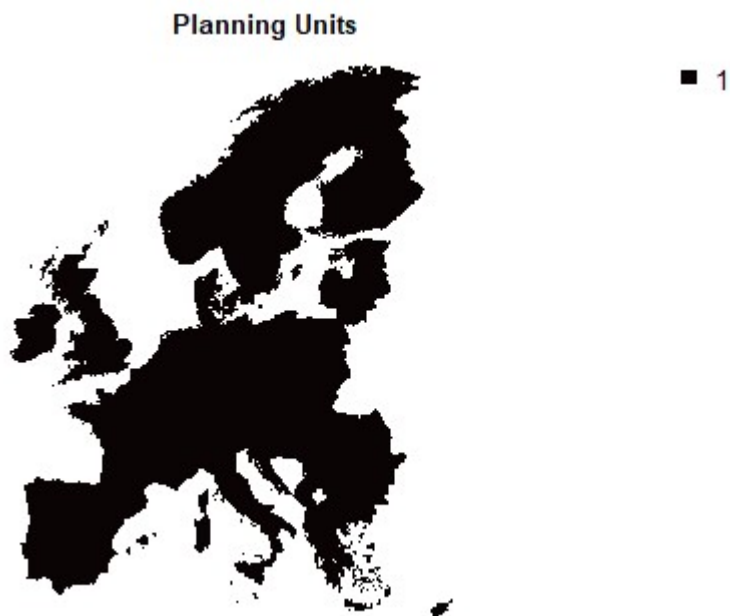
# Problem creation

# 2 Preparing input data

## 2.1 Planning units

Planning units (PU) contains the spatial data of the study area. Here, we use raster format, so the planning units are 10x10 km grid cells in Europe.

Let's read and plot the planning units raster.

```r
library(terra)
library(viridisLite)

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

The value of the planning units determines the cost of each planning unit in the prioritisation. In our case, we want to reach 30% area coverage. This means that in the Planning Units raster, each grid cell should have an equal value of 1 (so that the budget will be expressed in number of grid cells in prioritizr).

## 2.2 Features

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

Here, we consider the SDM of 67 tree species in Europe as features. We will focus on current distributions but we also have projected distributions under future climate (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:

```
spp.list <- list.files(path = "data/SpeciesDistributions/", full.names = T, recursive = T, p

spp <- rast(spp.list[grep("current", spp.list)])


names(spp) <- gsub("_ens-sdms_cur2005_prob_pot", "" ,names(spp))                    ①

## plot first four species distributions
plot(c(spp[[1]],spp[[2]], spp[[3]], spp[[4]]), axes = F,col = viridisLite::mako(n = 100, dir



## 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)])

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

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

11
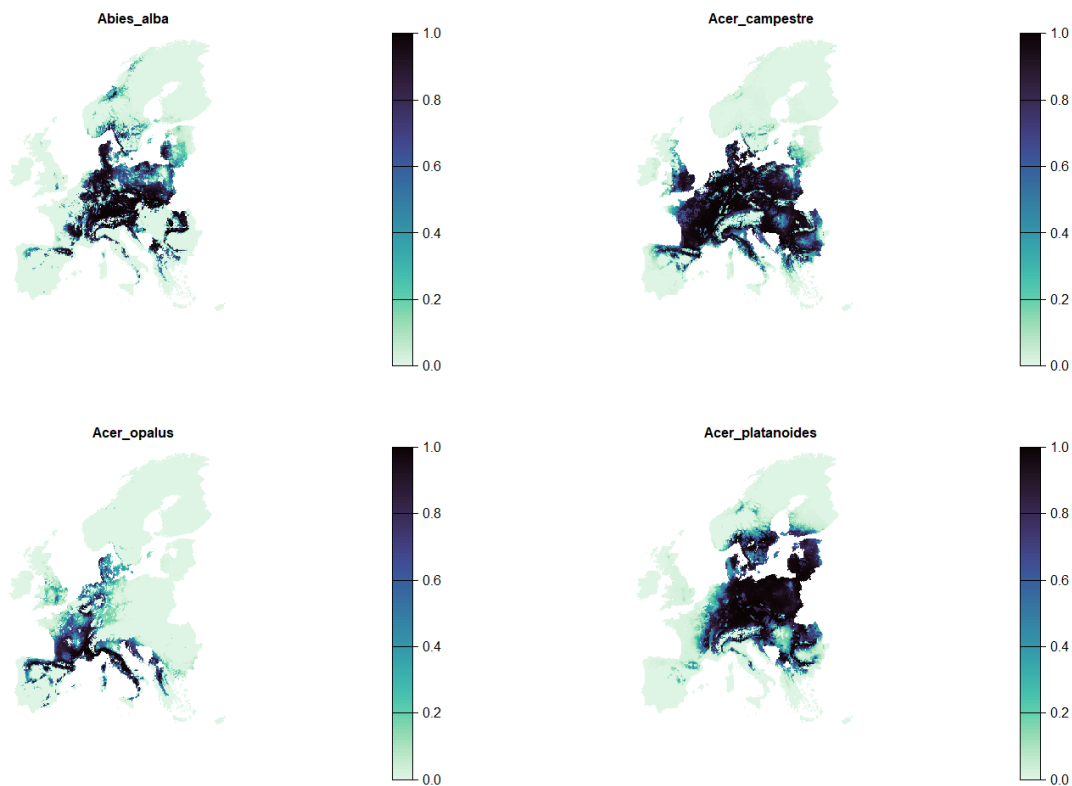
Note: features, and all other spatial data, need to be perfectly harmonized with the planning units data and (same extent, resolution, number of grid cells). This step must be done prior to the prioritisation, as part of the data preparation. For this training workshop, the data is already prepared, but bear in mind that data preparation is an essential step in the conservation planning process, which often takes time.

## 2.3 Protected areas

Often, we do not start from scratch: we often want to identify top priorities that **complement and expand on existing** protected areas.

```
# 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
```
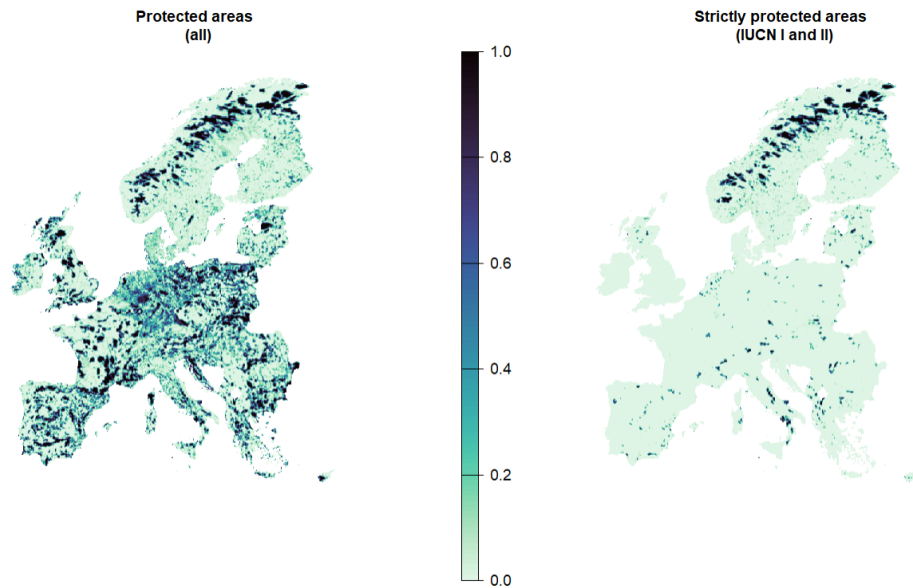


## 2.4 Locked-out or no-go areas

Here we use layers of high-intensity forests and urban areas as a proxy, using 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 =
```

13

```
locked.out <- sum(HI.forest, urban)
rclmat <- matrix(ncol = 3, nrow = 2, byrow = T,
                 c(0,50, 0,
                   50, 101, 1))

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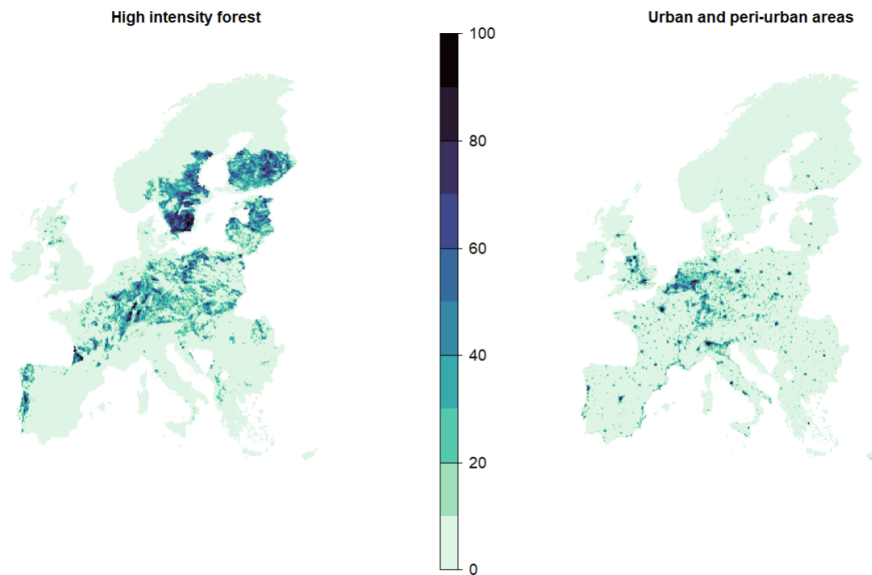


## 2.5  Costs

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. There are different types of costs:

- 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")

gHM[gHM<0.3] <- 0 ## set threshold so that sites that have GHM index lower than specified th

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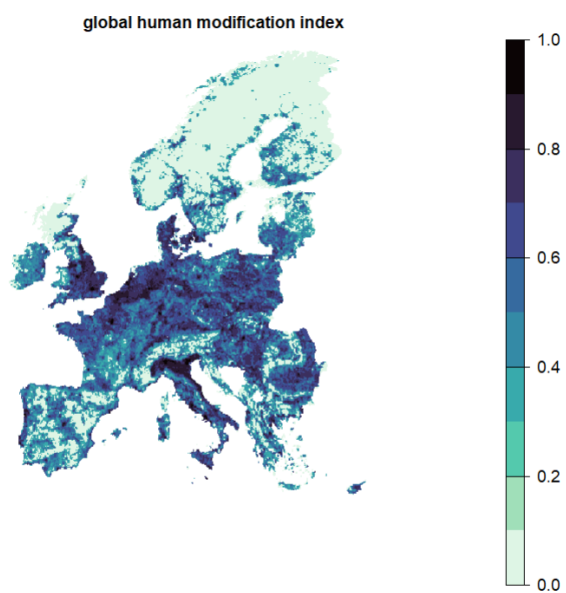


## 2.6 Vegetation quality

NDVI is often interpreted as dense and healthy vegetation, and one may be interested in selecting sites with a higher NDVI.

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

**NDVI**
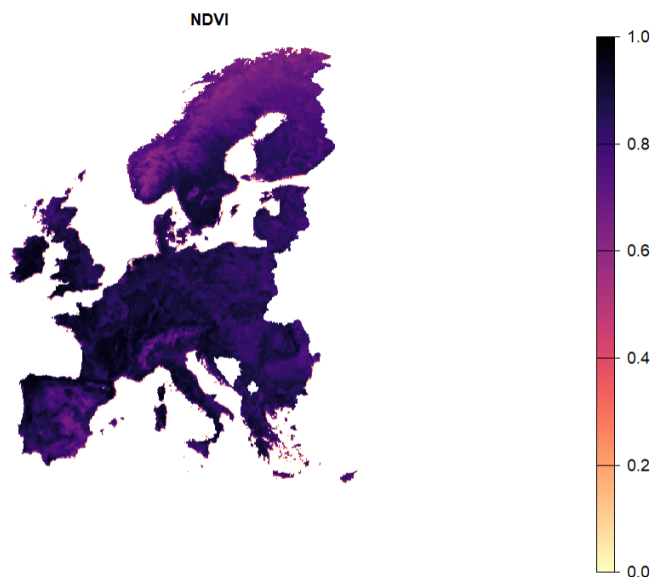
## 2.7 Other data for the prioritization

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) :

- 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),]
```

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. One typical approach for targets is to use log-linear targets Neugarten *et al.* (2024) . Another is to use the IUCN criteria to set targets based on the minimizing extinction risk Jung *et al.* (2021) .

# 3 Create basic problem

Create and solve a problem that finds the best areas for 30% protected area coverage on European land:

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

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

① target is 100% for all species distributions.
② CBC has the best performance across open solvers.
③ proportional decisions means that proportions of planning units can be selected in the solution. This typically solve faster and better than binary decisions.

# Part III

# Solving a problem

# 4 Solving a conservation planning problem

## 4.1 Create a solution

```
# solve and create the solution
s <- solve(p)
```

## 4.2 Plot the solution

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

# 5 Interpret and analyse outputs

## 5.1 Plot the solution

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

## 5.2 Calculate performance evaluation metrics

We can assess the performance of this solution for the species by assessing their representation and their target shortfall.

```
rpz_target_spp <- eval_target_coverage_summary(p, s)

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

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

## 5.3 Irreplaceability

Calculate irreplaceability of the solution using different methods.

The replacement cost scores can be calculated for small and moderate sized problems but may not be feasible for large problems (e.g., more than 100,000 planning units). For large problems, it is recommended to use the Ferrier method, which can be calculated relatively quickly for large-sized problems and it explicitly accounts for representation targets. Rarity weighted richness should only be used when neither of the other two methods can be used.

```
## to calculate importance scores using replacement cost:
# ir1 <- eval_replacement_importance(p, s)
## do not run: this only works for small problems.

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

# calculate importance scores using rarity weighted richness scores
ir3 <- eval_rare_richness_importance(p, s)


plot(c(ir1, ir2, ir3), axes = F, col = viridisLite::magma(n = 100, direction =-1))
```

# Part IV

# Adding complexity

# 6 Objective functions

## 6.1 The need for targets

## 6.2 Minimum set

## 6.3 Maximum coverage

## 6.4 Creating ranked priority maps

# 7 Adding complexity to conservation planning

## 7.1 Modify targets

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.

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

s1 <- solve(p1)

# plot map

plot(s1)
```

with loglinear targets
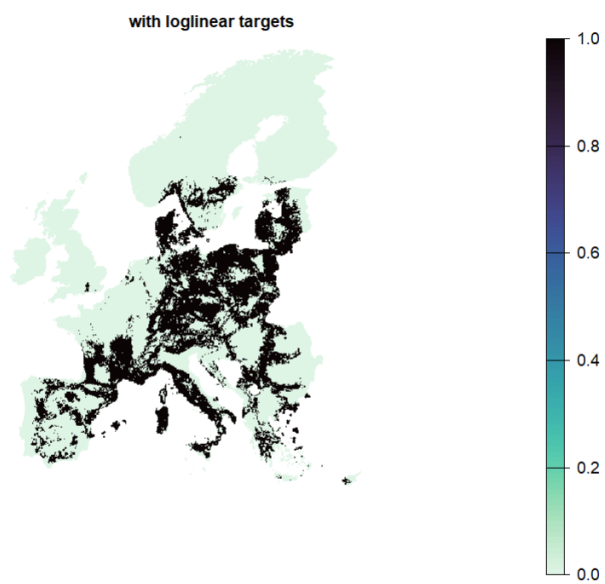
Note: targets can be informed by a combination of range size and red list status (see Jung *et al.* (2021) )

## 7.2 Add feature specific weights

```
p2 <- p1 %>%
  add_feature_weights(redlist.trees$weight)

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

with weights
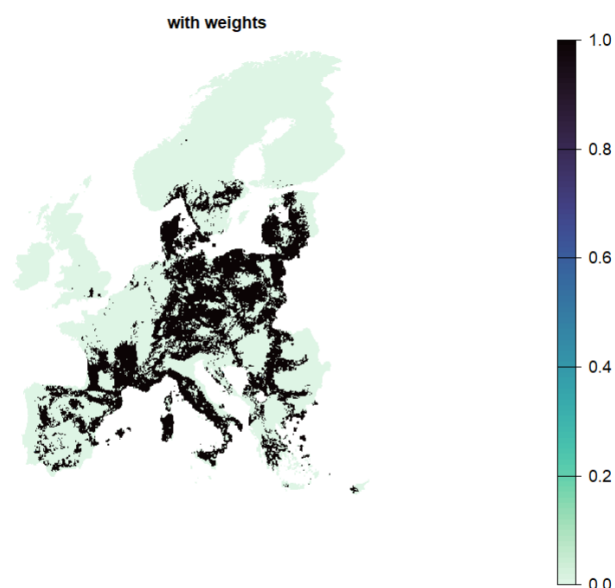
Notice how in this case the solution does not change drastically, since only a few species are listed as vulnerable.

## 7.3 Plan for future distributions under climate change

```
## create problem with future distributions as features:
p2_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_cbc_solver()%>%
  add_proportion_decisions()  ## entire grid cells (planning units) will be selected in the

s2_bis <- solve(p2_bis)
```

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)
```

average of solutions for current and future SDM
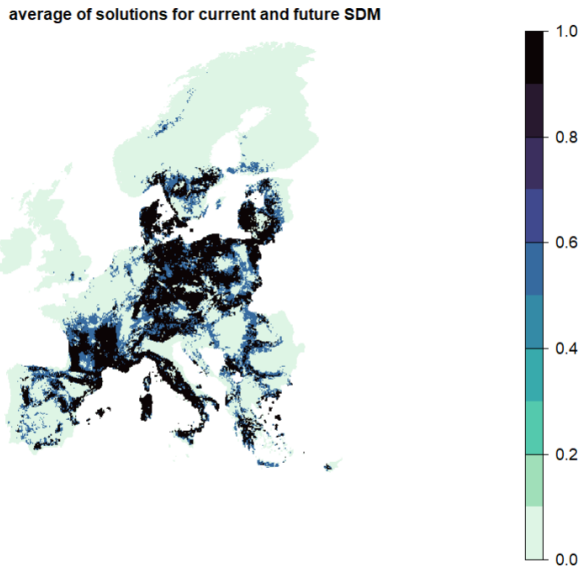
## 7.4 Add protected areas

```
### try with locked in constraints:

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 uni
pa_constraints <- data.frame(pu = cells(PA),                              ①
                             lower = unname(PA[!is.na(PA)]),              ②
```

28

```
                              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

Now, let's find top priorities for the expansion of existing protected areas but that uses projections of species distributions under 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)%>% ## to lock in proportional PA coverage p
  add_cbc_solver()%>%
  add_proportion_decisions()  ## entire grid cells (planning units) will be selected in the

s3_bis <- solve(p3_bis)
```

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
```

29

**Average solution with
current and future SDM, including PA**

**Average expansion priorities with
current and future SDM**
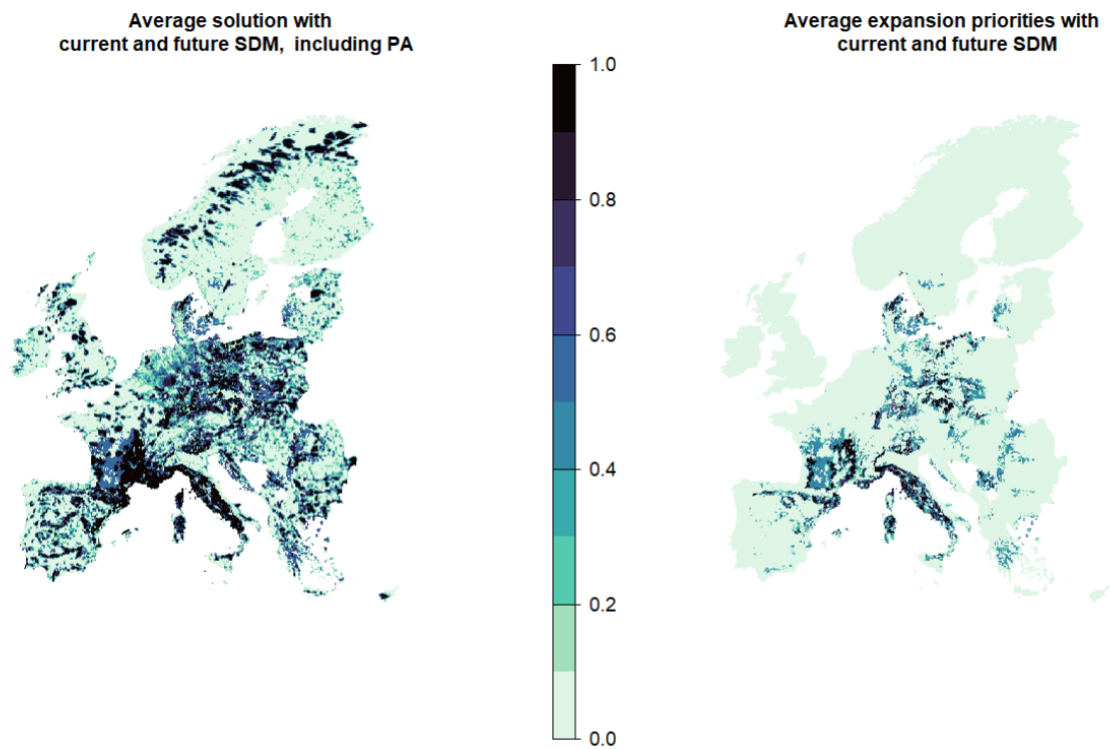
## 7.5 Add locked-out constraints

```
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...

## 7.6

## 7.7 Adding (socio-economic) costs

In prioritizr, the actual "cost" is tied to the value of the planning units, which then determines the budget. Since 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 actual 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 a penalty.

```
p5 <- p4 %>%
  add_linear_penalties(penalty = 1, data = gHM) ## note that when penalty score is set too h

s5 <- solve (p5)
```

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.

## 7.8 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.

A word of caution: 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.

## 7.9 Decision variables

So far, we solved problems as proportional decisions. Proportional decisions means that proportions of planning units can be selected in the solution. This typically solve faster and better than binary decisions.

Let's try solving the problem with a binary decision (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.

## 7.10 Modify the budget

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)
```

## 7.11

# 8 Compare and analyse different solutions

## 8.1 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)
```
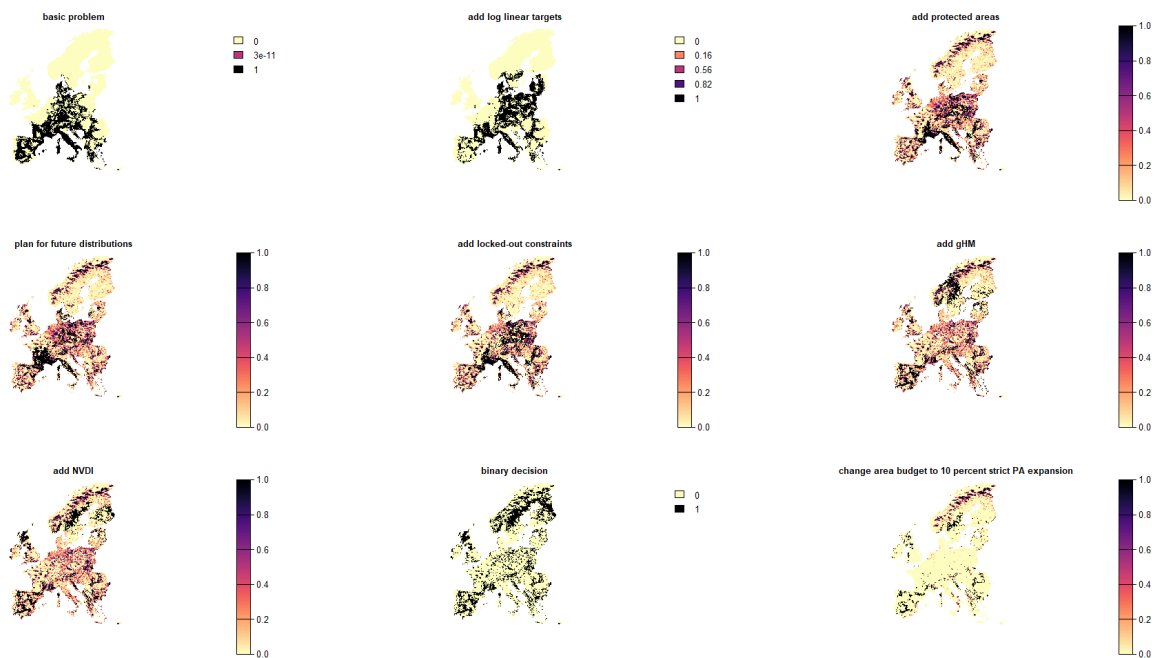
```
##### map safe bets for expansion priorities across all variations of the problems expanding
mean_s <- mean(s3, s3_bis, s4, s5, s6, s7)

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)
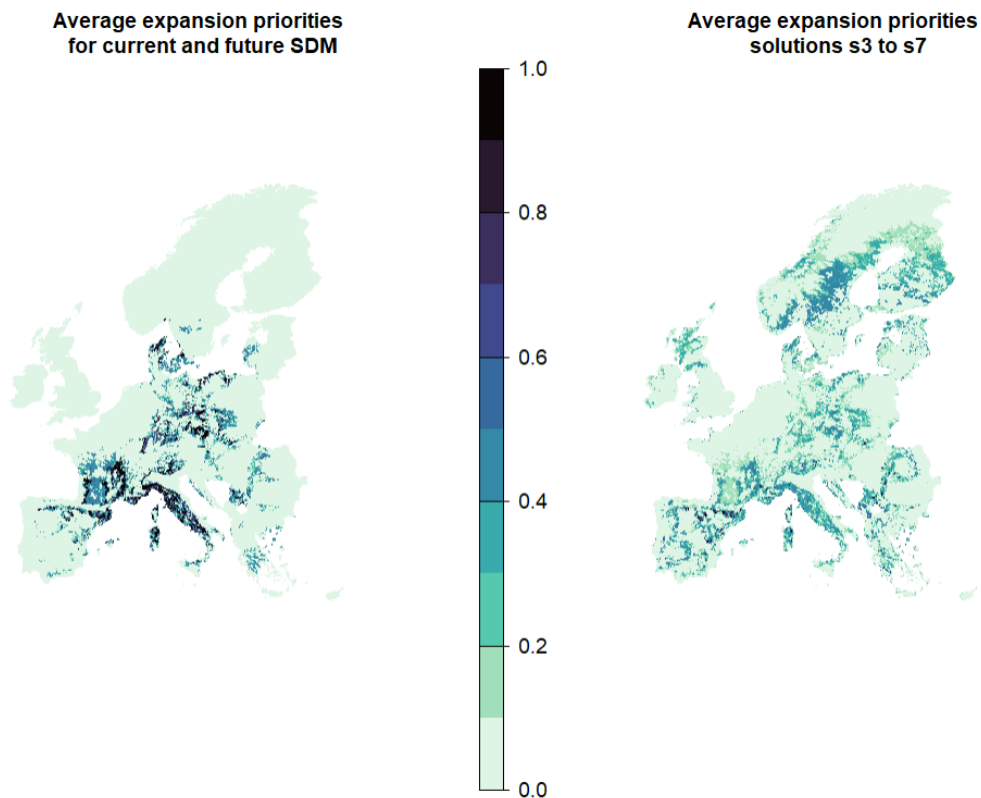


Remember that the solutions are highly dependent on methodological choices, and specifically on the input data (features, costs), constraints, and the objective function used, as well as the software.

Also note that, 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.

## 8.2 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
                          )
```
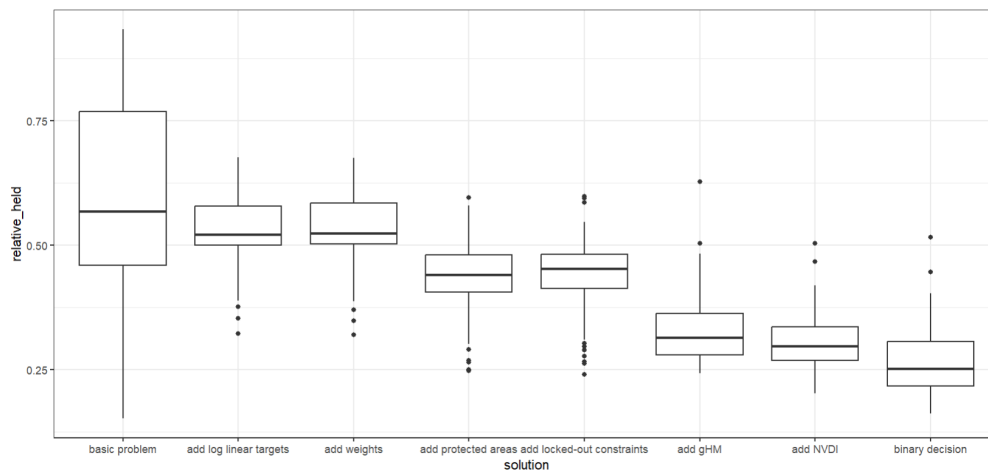
36

```
}
```

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

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



```r
# 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()
```
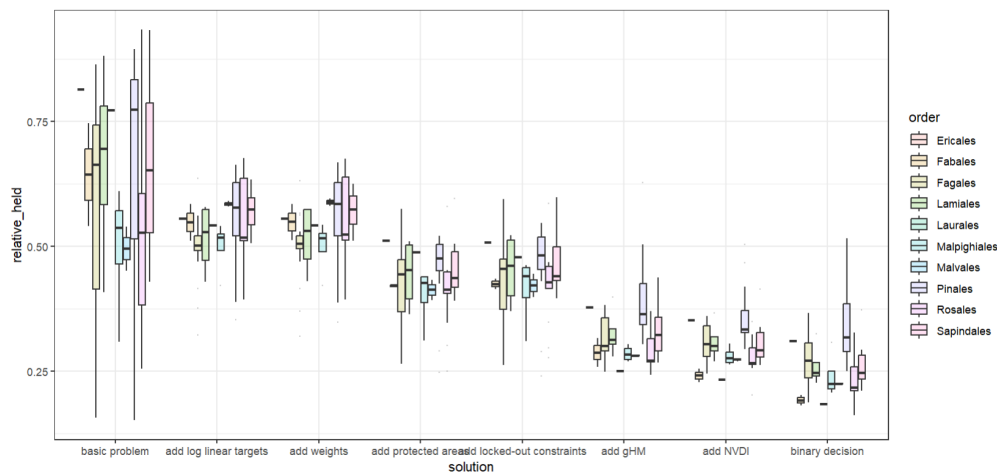
```
## group by Class instead of family
## add jitter points to see individual species representations
ggplot(scenarios_performance_species, aes(x = solution, y = relative_held)) +
  geom_boxplot(aes(fill = class), alpha = 0.2, outlier.size = 0)+
  geom_point(aes(x = solution, y = relative_held, colour = 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()
```
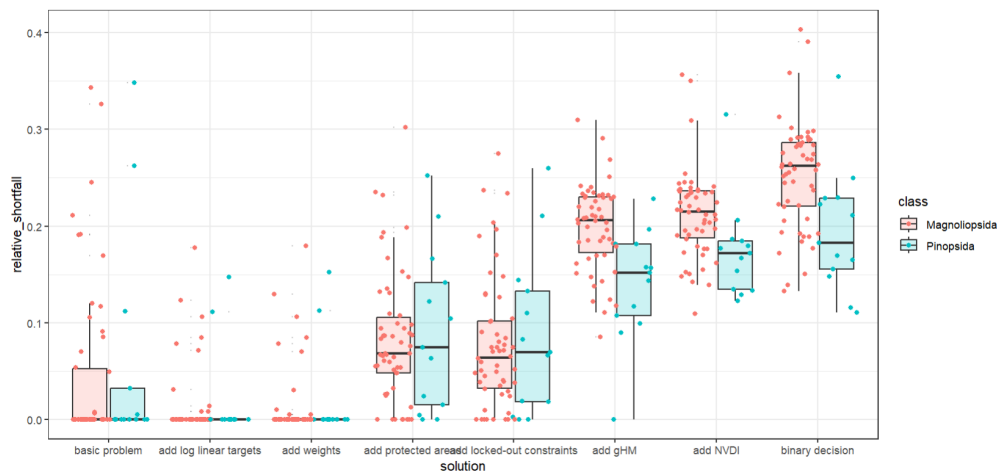
What do these two performance metrics tell us?

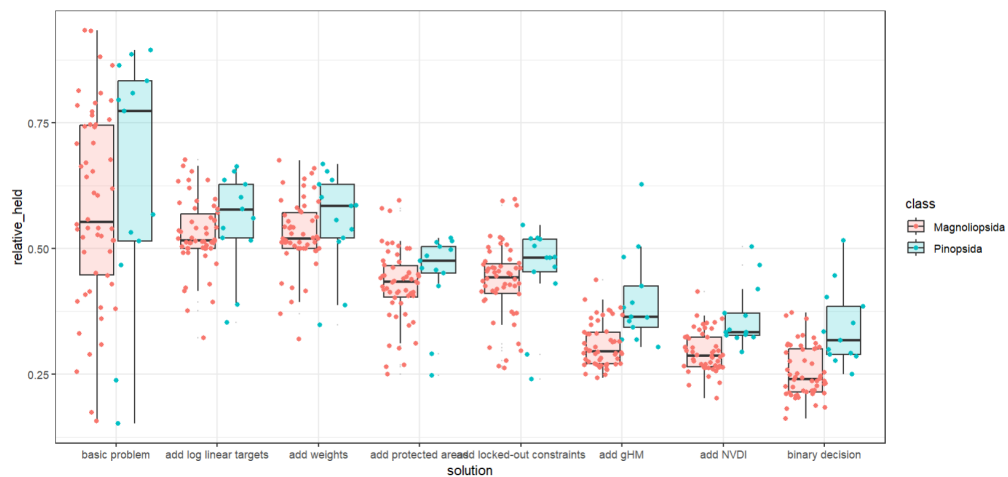## 8.3 Create a spatial ranking of conservation importance

Sometimes one may be interested in the relative ranking in the conservation value of planning units without a fixed budget. A rank map is one of the key outputs from Zonation. prioritizr does not provide such a rankmap, 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
## 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_loglinear_targets(10, 1, 10^4, 0.5) %>%
    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 =
```
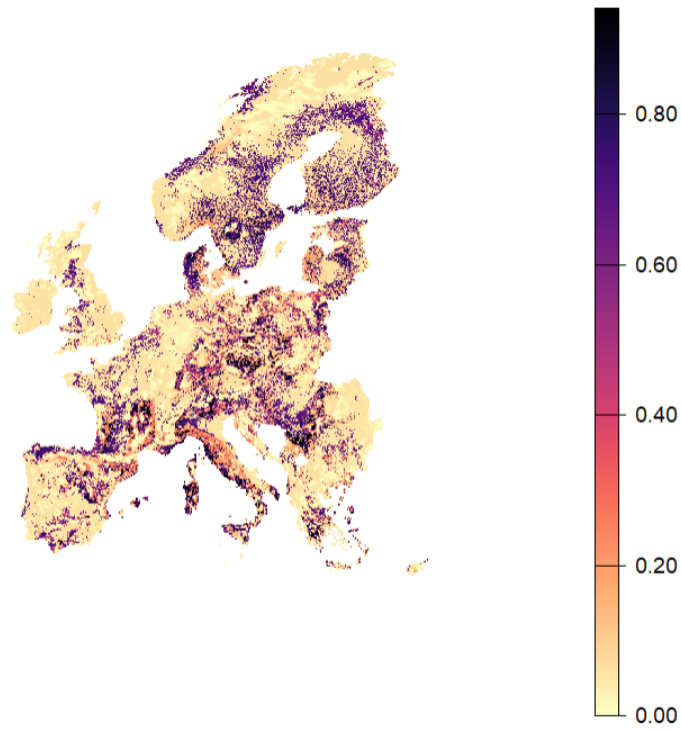
**Rank map**
**protected area expansion priorities for current SDM**

# Part V

# Advanced topics

# 9 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).

## 9.1 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.
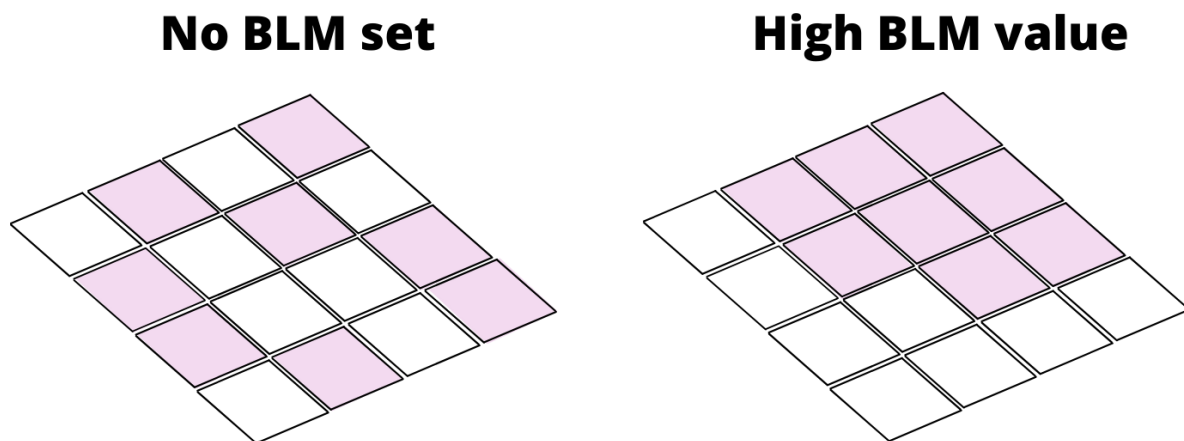


Figure 9.1: Boundary length modifier (BLM), which is effectively a penalty (Source: Marxan solutions)

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

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

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

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

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

```
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)
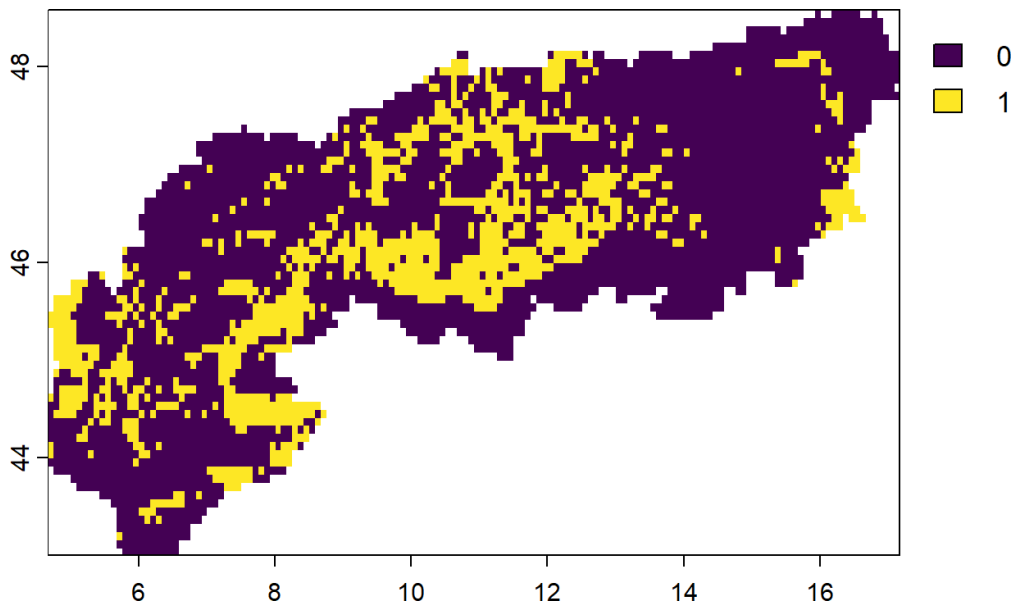


Now lets add some boundary constraints to the same problem.,

```
# 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.
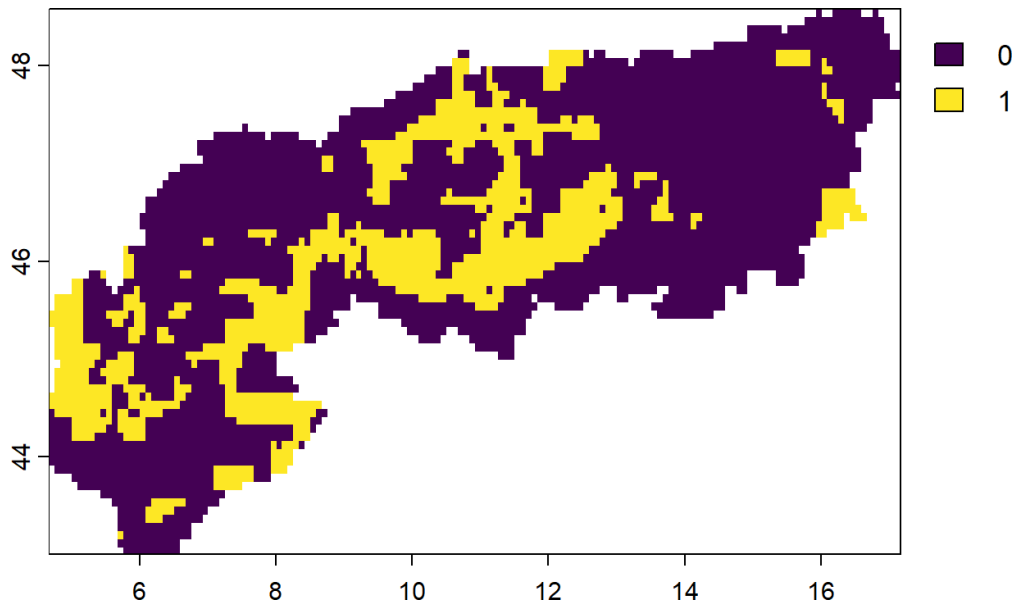


As you can see the solution is effectively more '*clumped*'. But what about the area selected? Do we need more area to get the best complementary solution here?

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

# Answer is...?
```

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

## 9.2 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.

### 9.2.1 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.
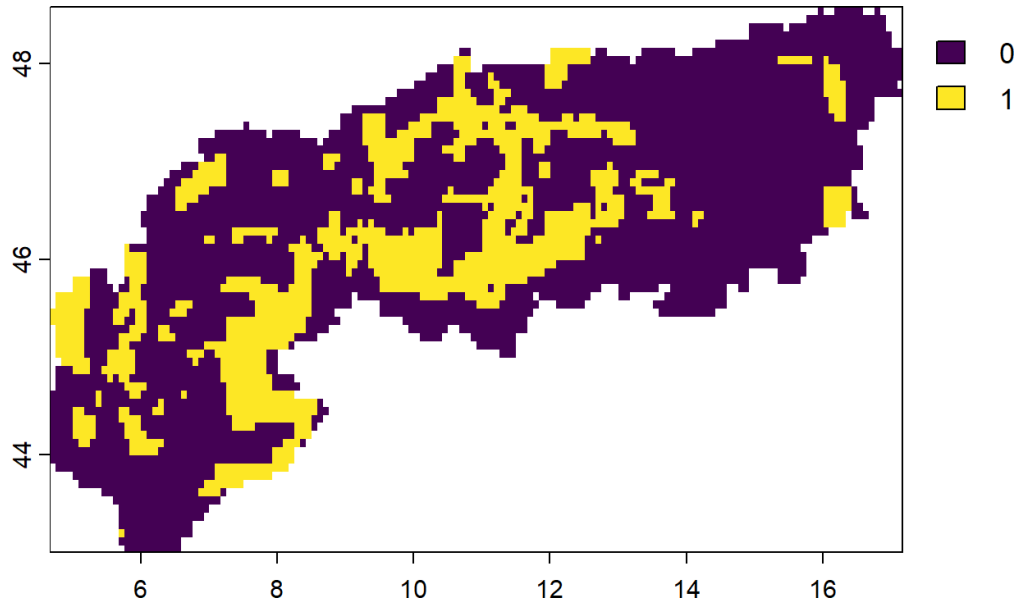


Figure 9.2: Prioritization with symmetric connectivity penalties

> 💡 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 prior-
itizing 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.
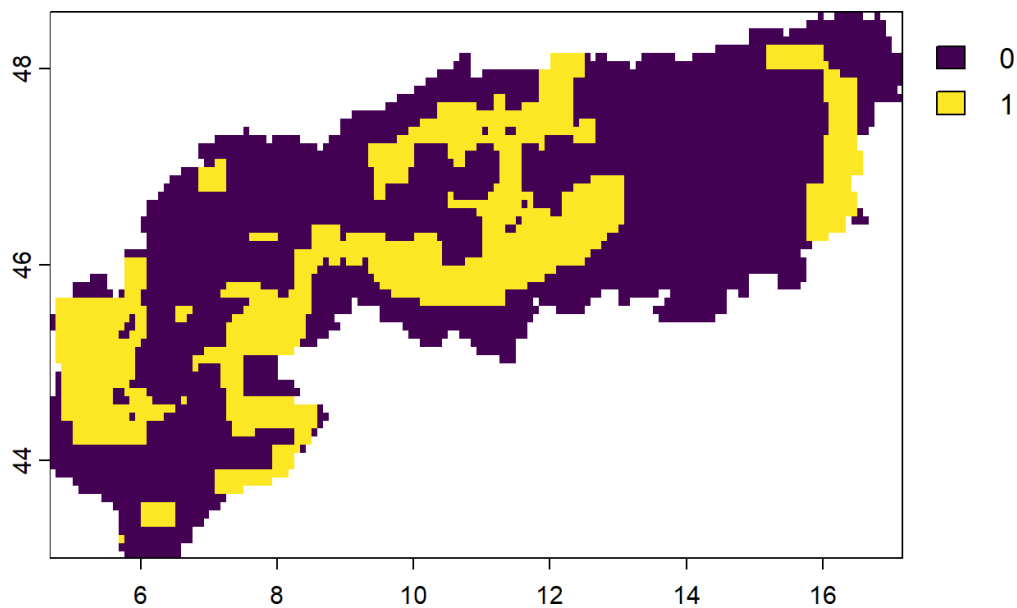


Figure 9.3: Proximity penalties(10km)

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

### 9.2.2 Asymmetric connectivity penalties

Opposed to symmetric connectivity penalties (Section 9.2.1), 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.
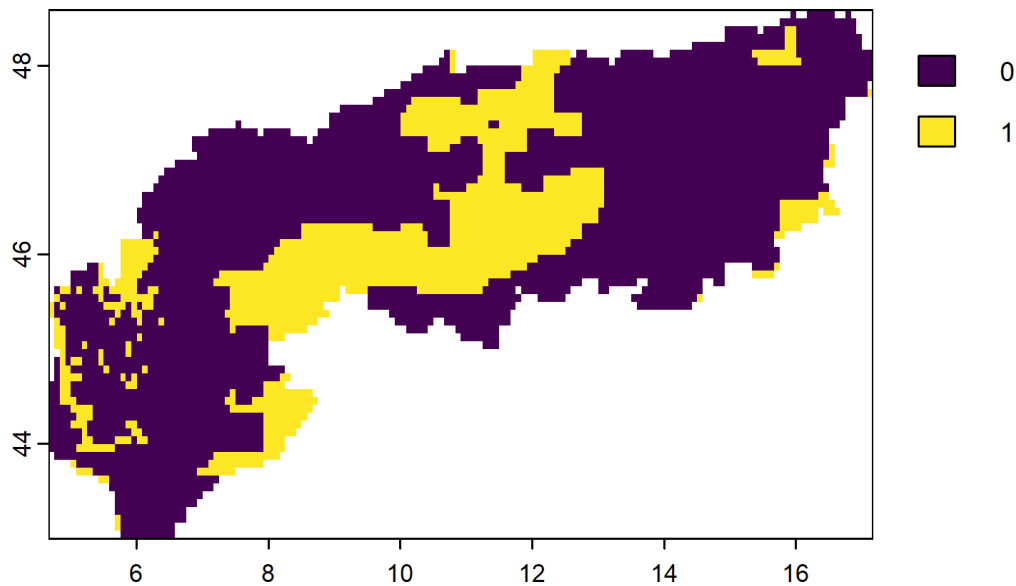
Figure 9.4: Asymmetric connectivity penalty from North to South

## 9.3 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).

### 9.3.1 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?
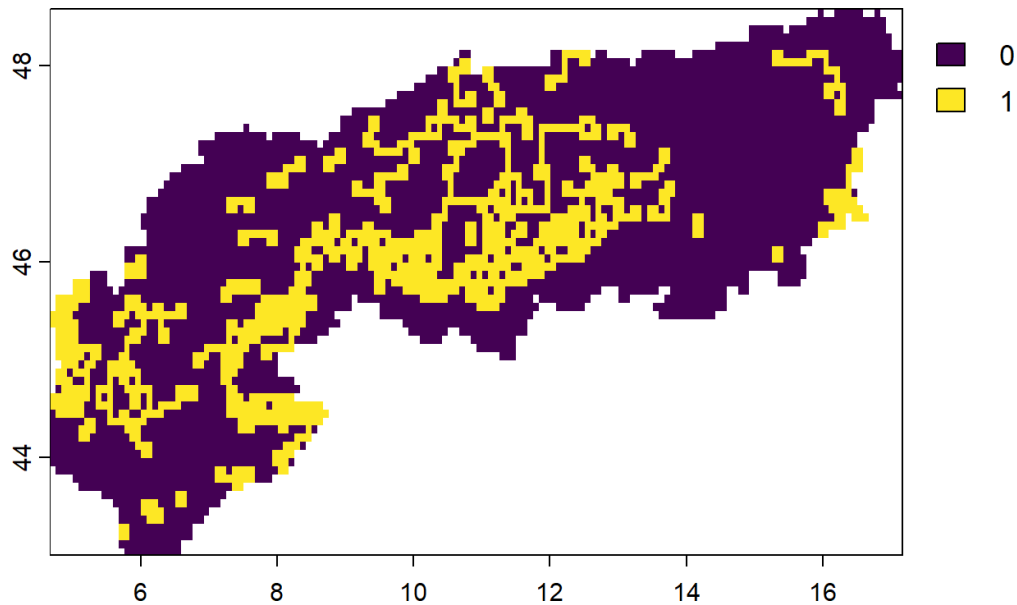


Figure 9.5: A solution with a hard connectivity constraint of having at least 2 neighbouring
PUs

## 9.3.2 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()
```

### 9.3.3 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.
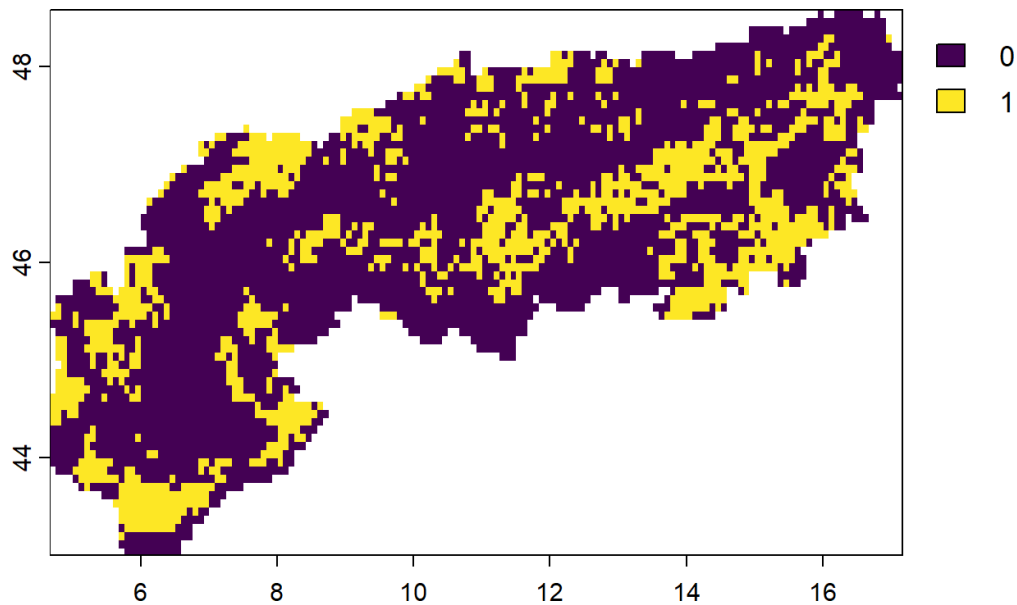
Figure 9.6: A prioritization including certain minimal amounts of greenness as constraint

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.

## 9.4 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.

```r
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.
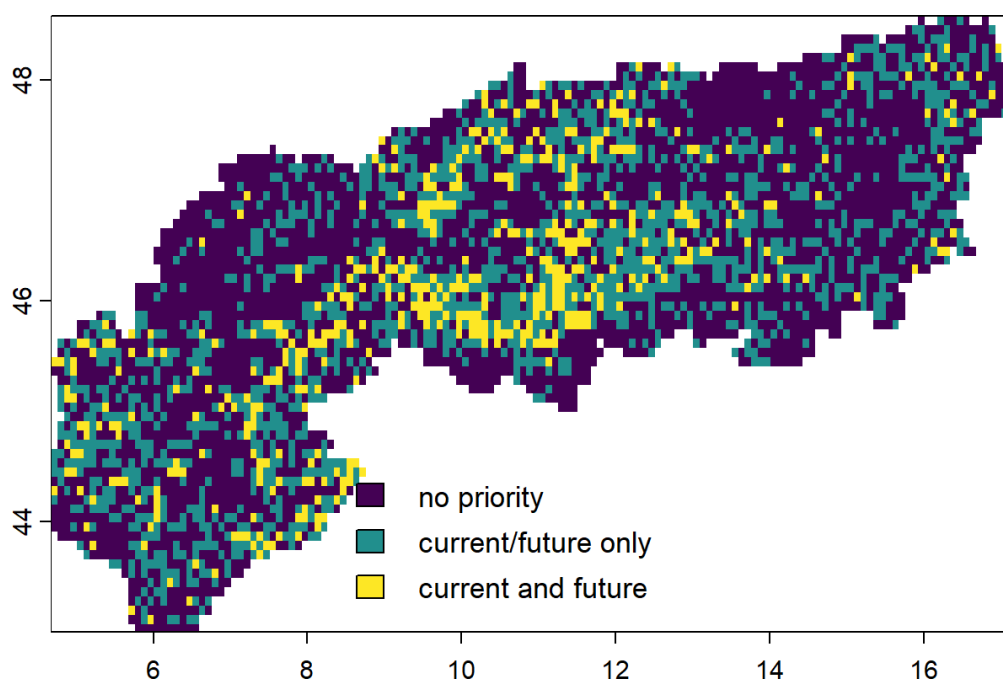② Since the decision variable is binary, we can simply sum the result.

Figure 9.7: Vertical connectivity with future projections

# 10 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.

```r
# 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.
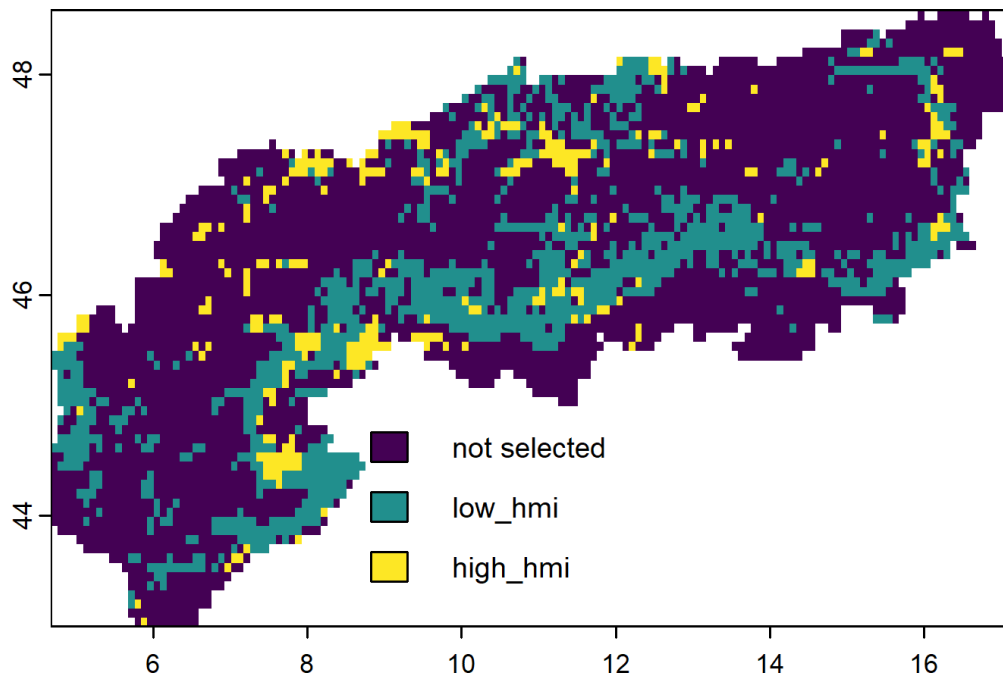
Figure 10.1: Selected features per zone

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)
```
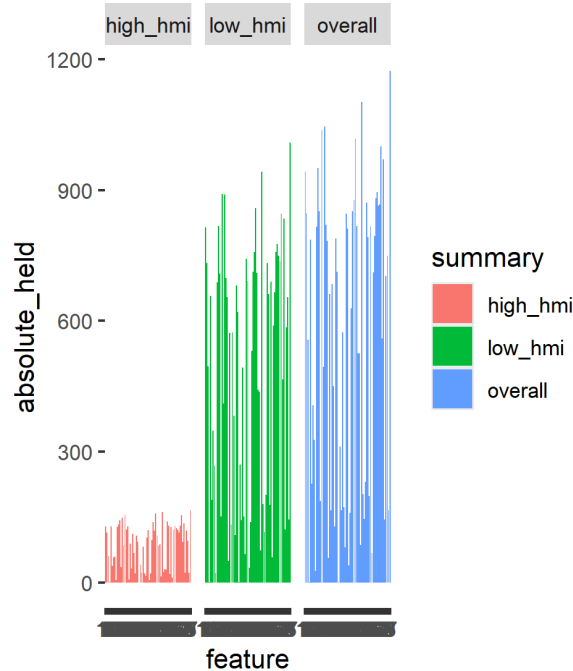
Figure 10.2: Relative amount held overall and per zone

## 10.1 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
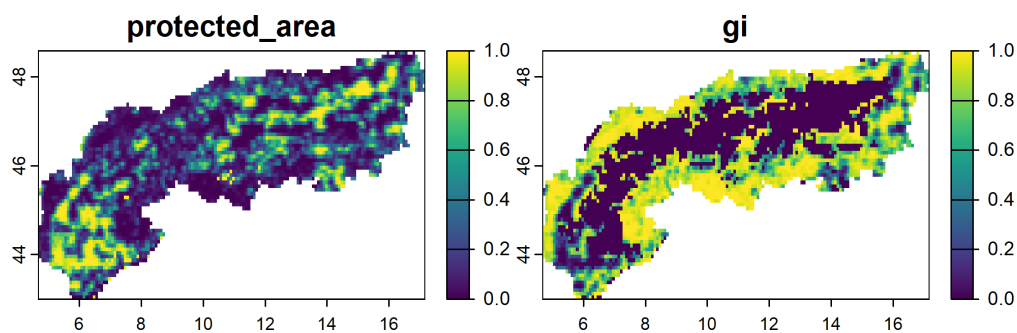
Figure 10.3: Expanding protected areas to 30% and the remainder to Green infrastructure

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 10.4: 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 10.1: A glossary of key terms used in this Training course

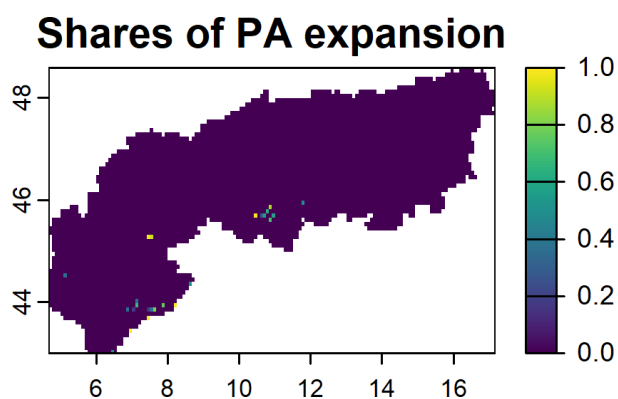| Term | Abbreviation if any | Definition |
|---|---|---|
| Boundary Length Modifier | BLM | A penalty constant added to a conservation problem that penalizes selecting isolated patches. Results in overall more compact solutions. |
| CARE | CARE | A often used abbreviation that stands for *Connectivity*, *Adequacy*, *Representation*, and *Effectiveness* 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. |
| Constrain | | A (often linear) constant or parameter that limits the selection of certain PU as part of the solution. |
| 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. |
| 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 |

| Term | Abbreviation if any | Definition |
|------|---------------------|------------|
| 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) |

# References

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.

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.

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.

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.

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

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

Neugarten, R.A., Chaplin-Kramer, R., Sharp, R.P., Schuster, R., Strimas-Mackey, M., Roehrdanz, P.R., Mulligan, M., Soesbergen, A. van, Hole, D., Kennedy, C.M. & others. (2024). Mapping the planet's critical areas for biodiversity and nature's contributions to people. *Nature communications*, *15*, 261.

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:

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

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

# 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.

> **i** 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:

```r
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.