PRIORITIZR WORKSHOP MANUAL

Jeffrey Hanson and Richard Schuster

2023-04-13

Contents

Chapter 1

Welcome!

Here you will find the manual for the prioritizr workshop. **Before you arrive at the workshop, you should make sure that you have correctly set up your computer for the workshop and you have downloaded the data from here.** Additionally, you can download a copy of the workshop slides for first day (from here) and the second day (from here).

Chapter 2

Introduction

2.1 Overview

The aim of this workshop is to help you get started with using the prioritizr R package for systematic conservation planning. It is not designed to give you a comprehensive overview and you will not become an expert after completing this workshop. Instead, we want to help you understand the core principles of conservation planning and guide you through some of the common tasks involved with generating prioritizations. In other words, we want to give you the knowledge base and confidence needed to start applying systematic conservation planning to your own work.

You are not alone in this workshop. If you are having trouble, please put your hand up and one of the instructors will help you as soon as they can. You can also ask the people sitting next to you for help too. **Most importantly, the code needed to answer the questions in this workshop are almost always located in the same section as the question. So if you are stuck, try rereading the example code and see if you can modify it to answer the question.** Please note that the first thing an instructor will ask you will be "what have you tried so far?". We can't help you if you haven't tried anything.

2.2 Setting up your computer

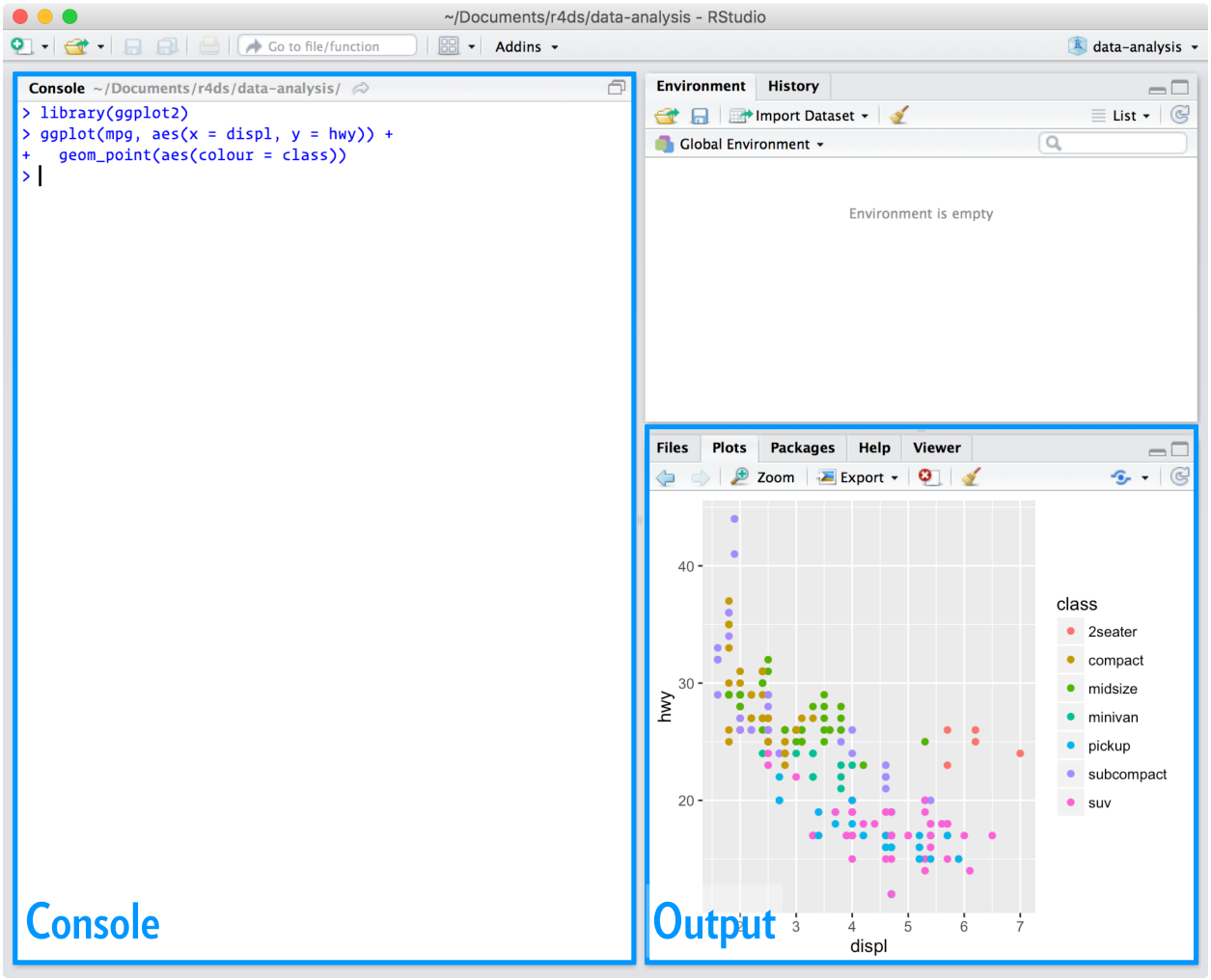
You will need to have both R and RStudio installed on your computer to complete this workshop. Although it is not imperative that you have the latest version of RStudio installed, **you will need the latest version of R installed (i.e., version 4.2.3)**. Please note that you might need administrative permissions to install these programs. After installing them, you will also need to install some R packages too.

2.2.1 R

The R statistical computing environment can be downloaded from the Comprehensive R Archive Network (CRAN). Specifically, you can download the latest version of R (version 4.2.3) from here: https://cloud.r-project.org. Please note that you will need to download the correct file for your operating system (i.e., Linux, macOS, Windows).

2.2.2 RStudio

RStudio is an integrated development environment (IDE). In other words, it is a program that is designed to make your R programming experience more enjoyable. During this workshop, you will interact with R through RStudio—meaning that you will open RStudio to code in R. You can download the latest version of RStudio here: http://www.rstudio.com/download. When you start RStudio, you will see two main parts of the interface: the *Console* and the *Output* (see below). You can type R code into the *Console* and press the enter key to run code. When you create maps or plots, they will appear in the *Output*.



2.2.3 R packages

An R package is a collection of R code and documentation that can be installed to enhance the standard R environment with additional functionality. Currently, there are over fifteen thousand R packages available on CRAN. Each of these R packages are developed to perform a specific task, such as reading Excel spreadsheets, downloading satellite imagery data, downloading and cleaning protected area data, or fitting environmental niche models. In fact, R has such a diverse ecosystem of R packages, that the question is almost always not "can I use R to . . . ?" but "what R package can I use to . . . ?". During this workshop, we will use several R packages. To install these R packages, please enter the code below in the *Console* part of the RStudio interface and press enter. Note that you will require an Internet connection and the installation process may take some time to complete.

```
install.packages("remotes")
remotes::install_cran(
  c("tidyverse", "mapview", "highs", "prioritizr"),
  upgrade = "always"
)
```

2.3 Further reading

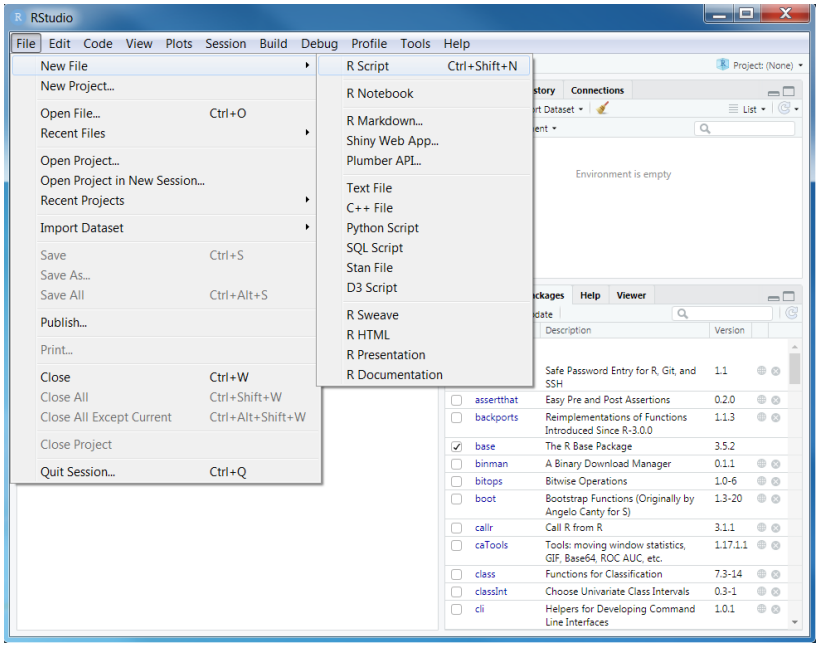
There is a wealth of resources available for learning how to use R. Although not required for this workshop, I would highly recommend that you read *R for Data Science* by Garrett Grolemund and Hadley Wickham. **This veritable trove of R goodness is freely available online.** If you spend a week going through this book then you will save months debugging and rerunning incorrect code. I would urge any and all ecologists, especially those working on Masters or PhD degrees, to read this book. I even bought this book as a Christmas present for my sister—and, yes, she was happy to receive it! For intermediate users looking to skill-up, I would recommend the *The Art of R Programming: A Tour of Statistical Software Design* by Norman Matloff and *Advanced R* by Hadley Wickham. Finally, if you wish to learn more about using R as a geospatial information system (GIS), I would recommend *Geocomputation with R* by Robin Lovelace, Jakub Nowosad, and Jannes Muenchow which is also freely available online. I also recommend *Applied Spatial Data Analysis* by Roger S. Bivand, Edzer Pebesma, and Virgilio Gómez-Rubio too.

Chapter 3

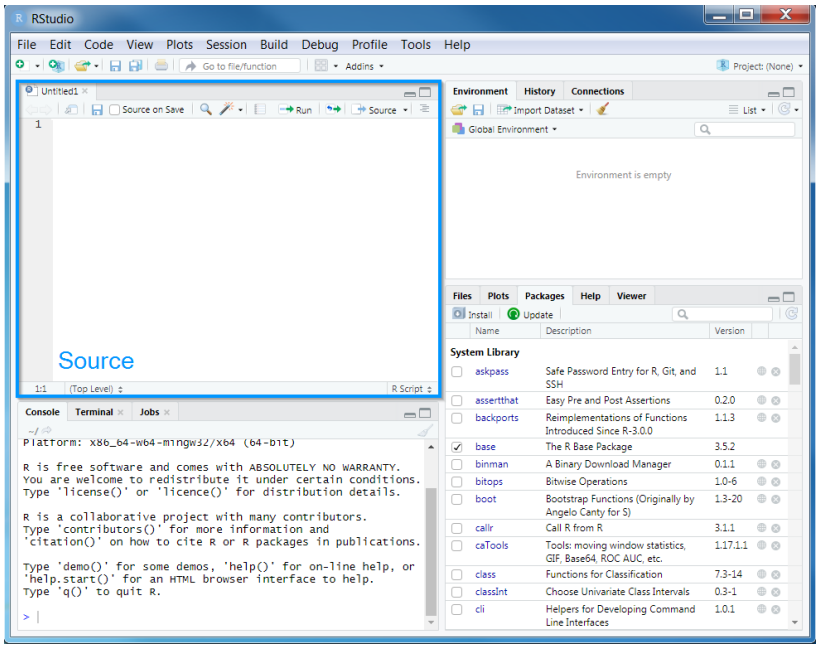
Data

3.1 Starting out

We will start by opening RStudio. Ideally, you will have already installed both R and Rstudio before the workshop. If you have not done this already, then please see the Setting up your computer section. **During this workshop, please do not copy and paste code from the workshop manual into RStudio. Instead, please write it out yourself in an R script.** When programming, you will spend a lot of time fixing coding mistakes—that is, debugging your code—so it is best to get used to making mistakes now when you have people here to help you. You can create a new R script by clicking on *File* in the RStudio menu bar, then *New File*, and then *R Script*.



After creating a new script, you will notice that a new *Source* panel has appeared. In the *Source* panel, you can type and edit code before you run it. You can run code in the *Source* panel by placing the cursor (i.e., the blinking line) on the desired line of code and pressing `Control + Enter` on your keyboard (or `CMD + Enter` if you are using an Apple computer). You can save the code in the *Source* panel by pressing `Control + s` on your keyboard (or `CMD + s` if you are using an Apple computer).



You can also make notes and write your answers to the workshop questions inside the R script. When writing notes and answers, add a `#` symbol so that the text following the `#` symbol is treated as a comment and not code. This means that you don't have to worry about highlighting specific parts of the script to avoid errors.

```
# this is a comment and R will ignore this text if you run it
# R will run the code below because it does not start with a # symbol
print("this is not a comment")
```

```
## [1] "this is not a comment"
```

```
# you can also add comments to the same line of R code too
print("this is also not a comment") # but this is a comment
```

```
## [1] "this is also not a comment"
```

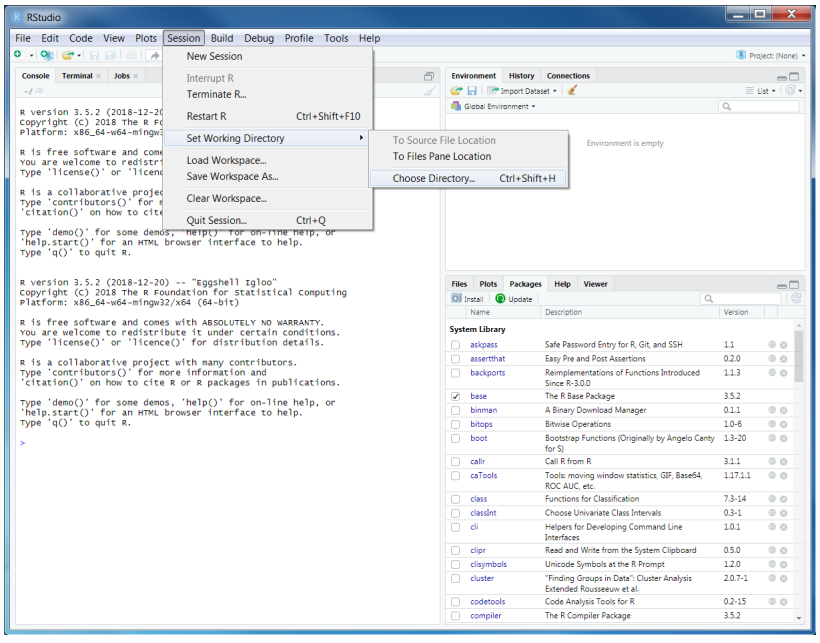
Remember to save your script regularly to ensure that you don't lose anything in the event that RStudio crashes (e.g., using `Control + s` or `CMD + s`)!

3.2 Setting up the R session

Now we will set up our R session for the workshop. Specifically, enter the following R code to attach the R packages used in this workshop.

```
# load packages
library(prioritizr) # package for conservation planning
library(tidyverse)  # package for data wrangling
library(terra)      # package for working with raster data
library(sf)         # package for working with vector data
library(highs)      # package provides HiGHS solver
library(mapview)    # package for creating interactive maps
library(units)      # package for unit conversions
library(scales)     # package for rescaling numbers

# setup printing for tables
## show all rows in tables
options(pillar.print_max = Inf)
```

You should have already downloaded the data for the prioritizr workshop. If you have not already done so, you can download it from here: https://github.com/prioritizr/workshop/raw/master/data.zip. After downloading the data, you can unzip the data into a new folder. Next, you will need to set the working directory to this new folder. To achieve this, click on the *Session* button on the RStudio menu bar, then click *Set Working Directory*, and then *Choose Directory*.



Now navigate to the folder where you unzipped the data and select *Open*. You can verify that you have correctly set the working directory using the following R code. You should see the output `TRUE` in the *Console* panel.

```
file.exists("data/pu.gpkg")
```

```
## [1] TRUE
```

3.3 Data import

Now that we have downloaded the dataset, we will need to import it into our R session. Specifically, this data was obtained from the "Introduction to Marxan" course and the Australian Government's National Vegetation Information System. It contains vector-based planning unit data (`pu.gpkg`) and the raster-based data describing the spatial distributions of 33 vegetation classes (`vegetation.tif`) in Tasmania, Australia. Please note this dataset is only provided for teaching purposes and should not be used for any real-world conservation planning. We can import the data into our R session using the following code.

```
# import planning unit data
## note that read_sf is from the sf package
pu_data <- read_sf("data/pu.gpkg")

# import vegetation data
## note that rast() is from the terra package
veg_data <- rast("data/vegetation.tif")
```

3.4 Planning unit data

The planning unit data contains spatial data describing the geometry for each planning unit and attribute data with information about each planning unit (e.g., cost values). Let's investigate the `pu_data` object. The planning unit data contains 5 columns with the following information:

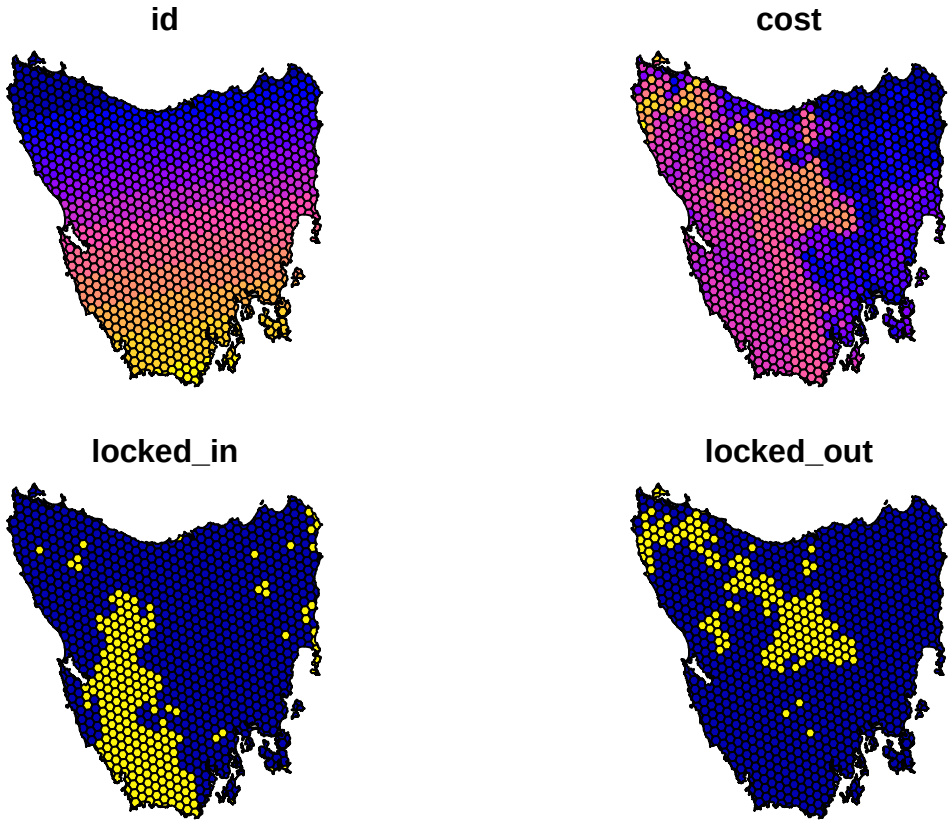
- `id`: unique identifiers for each planning unit
- `cost`: acquisition cost values for each planning unit (millions of Australian dollars).
- `locked_in`: logical values (i.e., `TRUE`/`FALSE`) indicating if planning units are covered by protected areas or not.
- `locked_out`: logical values (i.e., `TRUE`/`FALSE`) indicating if planning units cannot be managed as a protected area because they contain are too degraded.

- `geom`: spatial geometries for the planning units.

```
# print the first six rows of the planning unit data
head(pu_data)
```

```
## Simple feature collection with 6 features and 4 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: 303910.1 ymin: 5485840 xmax: 335610.4 ymax: 5502544
## Projected CRS: WGS 84 / UTM zone 55S
## # A tibble: 6 x 5
##      id  cost locked_in locked_out                                      geom
##   <int> <dbl> <lgl>     <lgl>                                <MULTIPOLYGON [m]>
## 1     1  60.2 FALSE     TRUE       (((328497 5497704, 326783.8 5500050, 326775.~
## 2     2  19.9 FALSE     FALSE      (((307121.6 5490487, 305344.4 5492917, 30538~
## 3     3  59.7 FALSE     TRUE       (((321726.1 5492382, 320111 5494593, 320127 ~
## 4     4  32.4 FALSE     FALSE      (((304314.5 5494324, 304342.2 5494287, 30432~
## 5     5  26.2 FALSE     FALSE      (((314958.5 5487057, 312336 5490646, 312339.~
## 6     6  51.3 FALSE     TRUE       (((327904.3 5491218, 326594.6 5493012, 32849~
```

```
# plot maps of the planning unit data, showing each of the columns
plot(pu_data)
```


id
cost
locked_in
locked_out

```
# print number of planning units (geometries) in the data
nrow(pu_data)
```

```
## [1] 1130
```

```
# print the highest cost value
max(pu_data$cost)
```

```
## [1] 61.92727
```

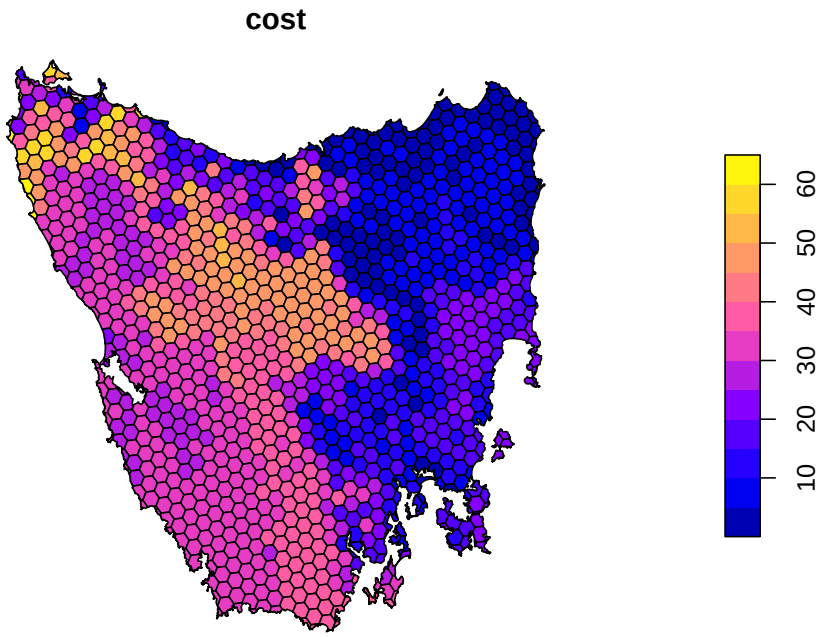
```
# print the smallest cost value
min(pu_data$cost)
```

```
## [1] 0.1924883
```

```
# print average cost value
mean(pu_data$cost)
```

```
## [1] 25.13536
```

```
# plot a map of the planning unit cost data
plot(pu_data[, "cost"])
```



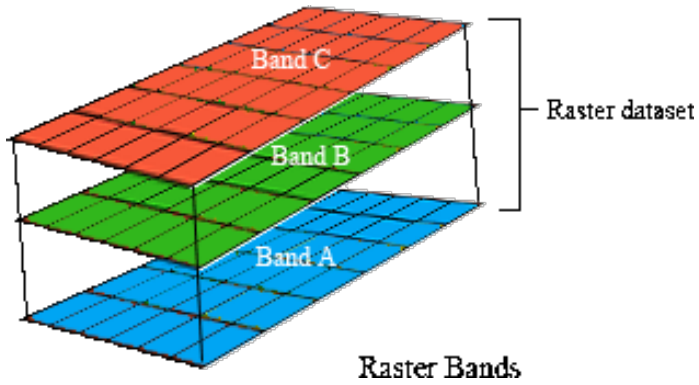

```
# plot an interactive map of the planning unit cost data
mapview(pu_data, zcol = "cost")
```

Now, you can try and answer some questions about the planning unit data.

1. How many planning units are in the planning unit data?
2. What is the highest cost value?
3. How many planning units are covered by the protected areas (hint: `sum(x)`)?
4. What is the proportion of the planning units that are covered by the protected areas (hint: `mean(x)`)?
5. How many planning units are highly degraded (hint: `sum(x)`)?
6. What is the proportion of planning units are highly degraded (hint: `mean(x)`)?
7. Can you verify that all values in the `locked_in` and `locked_out` columns are zero or one (hint: `min(x)` and `max(x)`)?.
8. Can you verify that none of the planning units are missing cost values (hint: `all(is.finite(x))`)?.
9. Can you very that none of the planning units have duplicated identifiers? (hint: `sum(duplicated(x))`)?
10. Is there a spatial pattern in the planning unit cost values (hint: use `plot(x)` to make a map).
11. Is there a spatial pattern in where most planning units are covered by protected areas (hint: use `plot(x)` to make a map).

3.5 Vegetation data

The vegetation data describes the spatial distribution of 33 vegetation classes in the study area. This data is in a raster format and so the data are organized using a square grid comprising square grid cells that are each the same size. In our case, the raster data contains multiple layers (also called "bands") and each layer has corresponds to a spatial grid with exactly the same area and has exactly the same dimensionality (i.e., number of rows, columns, and cells). In this dataset, there are 33 different regular spatial grids layered on top of each other – with each layer corresponding to a different vegetation class – and each of these layers contains a grid with 398 rows, 359 columns, and 142882 cells. Within each layer, each cell corresponds to a 1 by 1 km square. The values associated with each grid cell indicate the (one) presence or (zero) absence of a given vegetation class in the cell.



Let's explore the vegetation data.

```
# print a short summary of the data
print(veg_data)
```

```
## class       : SpatRaster
## dimensions  : 398, 359, 33  (nrow, ncol, nlyr)
## resolution  : 1000, 1000  (x, y)
## extent      : 288801.7, 647801.7, 5142976, 5540976  (xmin, xmax, ymin, ymax)
## coord. ref. : WGS 84 / UTM zone 55S (EPSG:32755)
## source      : vegetation.tif
## names       : Banks~lands, Bould~marks, Calli~lands, Cool ~orest, Eucal~hyll), Eucal~
## min values  :           0,           0,           0,           0,           0,
## max values  :           1,           1,           1,           1,           1,
```

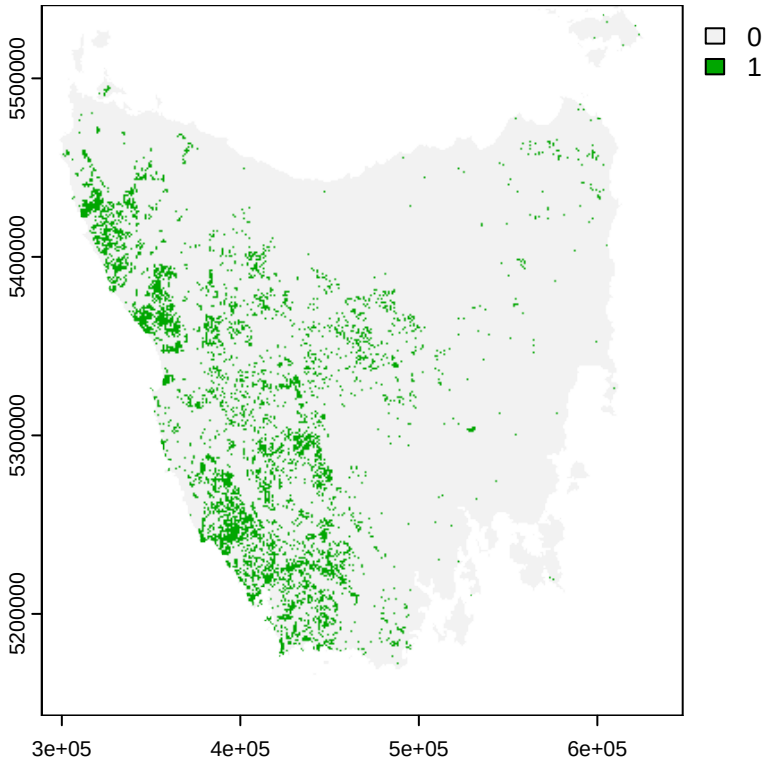
```
# print number of layers in the data
nlyr(veg_data)
```

```
## [1] 33
```

```
# print the name of each layer
names(veg_data)
```

```
##  [1] "Banksia woodlands"
##  [2] "Boulders/rock with algae, lichen or scattered plants, or alpine fjaeldmarks"
##  [3] "Callitris forests and woodlands"
##  [4] "Cool temperate rainforest"
##  [5] "Eucalyptus (+/- tall) open forest with a dense broad-leaved and/or tree-fern un
##  [6] "Eucalyptus open forests with a shrubby understorey"
##  [7] "Eucalyptus open woodlands with shrubby understorey"
##  [8] "Eucalyptus tall open forest with a fine-leaved shrubby understorey"
##  [9] "Eucalyptus tall open forests and open forests with ferns, herbs, sedges, rushes
## [10] "Eucalyptus woodlands with a shrubby understorey"
## [11] "Eucalyptus woodlands with a tussock grass understorey"
## [12] "Eucalyptus woodlands with ferns, herbs, sedges, rushes or wet tussock grassland
## [13] "Freshwater, dams, lakes, lagoons or aquatic plants"
## [14] "Heathlands"
## [15] "Leptospermum forests and woodlands"
## [16] "Low closed forest or tall closed shrublands (including Acacia, Melaleuca and Ba
## [17] "Mallee with a tussock grass understorey"
## [18] "Melaleuca open forests and woodlands"
## [19] "Melaleuca shrublands and open shrublands"
## [20] "Mixed chenopod, samphire +/- forbs"
## [21] "Naturally bare, sand, rock, claypan, mudflat"
## [22] "Other Acacia tall open shrublands and shrublands"
## [23] "Other forests and woodlands"
## [24] "Other open woodlands"
## [25] "Other shrublands"
## [26] "Other tussock grasslands"
## [27] "Regrowth or modified forests and woodlands"
## [28] "Saline or brackish sedgelands or grasslands"
## [29] "Salt lakes and lagoons"
## [30] "Sedgelands, rushs or reeds"
## [31] "Temperate tussock grasslands"
## [32] "Unclassified native vegetation"
## [33] "Wet tussock grassland with herbs, sedges or rushes, herblands or ferns"
```

```
# layers can be accessed using indices or names,
## for example the 30th class is "Sedgelands, rushs or reeds"
## and we can make a map of it using the index or the name

# plot a map of the 30th class using the index
plot(veg_data[[30]])

# plot a map of the 30th class using its layer name
plot(veg_data[["Sedgelands, rushs or reeds"]])
```




```
# plot an interactive map of the 30th class
## note that we use method = "ngb" because the data are not continuous
mapview(raster::raster(veg_data[[30]]), method = "ngb")
```

```
# print resolution on the x-axis
xres(veg_data)
```

```
## [1] 1000
```

```
# print resolution on the y-axis
yres(veg_data)
```

```
## [1] 1000
```

```
# print spatial extent of the grid, i.e., coordinates for corners
ext(veg_data)
```

```
## SpatExtent : 288801.732237428, 647801.732237428, 5142975.76801917, 5540975.76801917 (
```

```
# print a summary of the first layer
print(veg_data[[1]])
```

```
## class       : SpatRaster
## dimensions  : 398, 359, 1  (nrow, ncol, nlyr)
## resolution  : 1000, 1000  (x, y)
## extent      : 288801.7, 647801.7, 5142976, 5540976  (xmin, xmax, ymin, ymax)
## coord. ref. : WGS 84 / UTM zone 55S (EPSG:32755)
## source      : vegetation.tif
## name        : Banksia woodlands
## min value   :                 0
## max value   :                 1
```

```
# calculate the sum of all the cell values in the first layer
global(veg_data[[1]], "sum", na.rm = TRUE)
```

```
##                   sum
## Banksia woodlands   2
```

```
# calculate the maximum value of all the cell values in the first layer
global(veg_data[[1]], "max", na.rm = TRUE)
```

```
##                   max
## Banksia woodlands   1
```

```
# calculate the minimum value of all the cell values in the first layer
global(veg_data[[1]], "min", na.rm = TRUE)
```

```
##                     min
## Banksia woodlands   0
```

```
# calculate the mean value of all the cell values in the first layer
global(veg_data[[1]], "mean", na.rm = TRUE)
```

```
##                            mean
## Banksia woodlands 3.021559e-05
```

```
# calculate the maximum value in each layer
as_tibble(global(veg_data, "max", na.rm = TRUE), rownames = "feature")
```

```
## # A tibble: 33 x 2
##    feature                                                          max
##    <chr>                                                          <dbl>
##  1 Banksia woodlands                                                  1
##  2 Boulders/rock with algae, lichen or scattered plants, or alpine fjaeld~     1
##  3 Callitris forests and woodlands                                    1
##  4 Cool temperate rainforest                                          1
##  5 Eucalyptus (+/- tall) open forest with a dense broad-leaved and/or tre~     1
##  6 Eucalyptus open forests with a shrubby understorey                 1
##  7 Eucalyptus open woodlands with shrubby understorey                 1
##  8 Eucalyptus tall open forest with a fine-leaved shrubby understorey         1
##  9 Eucalyptus tall open forests and open forests with ferns, herbs, sedge~     1
## 10 Eucalyptus woodlands with a shrubby understorey                    1
## 11 Eucalyptus woodlands with a tussock grass understorey              1
## 12 Eucalyptus woodlands with ferns, herbs, sedges, rushes or wet tussock ~     1
## 13 Freshwater, dams, lakes, lagoons or aquatic plants                 1
## 14 Heathlands                                                         1
## 15 Leptospermum forests and woodlands                                 1
## 16 Low closed forest or tall closed shrublands (including Acacia, Melaleu~     1
## 17 Mallee with a tussock grass understorey                            1
## 18 Melaleuca open forests and woodlands                               1
## 19 Melaleuca shrublands and open shrublands                           1
## 20 Mixed chenopod, samphire +/- forbs                                 1
```

```
## 21 Naturally bare, sand, rock, claypan, mudflat                              1
## 22 Other Acacia tall open shrublands and shrublands                          1
## 23 Other forests and woodlands                                               1
## 24 Other open woodlands                                                      1
## 25 Other shrublands                                                          1
## 26 Other tussock grasslands                                                  1
## 27 Regrowth or modified forests and woodlands                                1
## 28 Saline or brackish sedgelands or grasslands                               1
## 29 Salt lakes and lagoons                                                    1
## 30 Sedgelands, rushs or reeds                                                1
## 31 Temperate tussock grasslands                                              1
## 32 Unclassified native vegetation                                            1
## 33 Wet tussock grassland with herbs, sedges or rushes, herblands or ferns    1
```

Now, you can try and answer some questions about the vegetation data.

1. What part of the study area is the "Temperate tussock grasslands" vegetation class found in (hint: make a map)?
2. What proportion of cells contain the "Heathland" vegetation class (hint: calculate the mean value of the cells)?
3. Which vegetation class is present in the greatest number of cells?
4. The planning unit data and the vegetation data should have the same coordinate reference system. Can you check if they are the same?

Chapter 4

Gap analysis

4.1 Introduction

Before we begin to prioritize areas for protected area establishment, we should first understand how well existing protected areas are conserving our biodiversity features (i.e., native vegetation classes in Tasmania, Australia). This step is critical: we cannot develop plans to improve conservation of biodiversity if we don't understand how well existing policies are currently conserving biodiversity! To achieve this, we can perform a "gap analysis". A gap analysis involves calculating how well each of our biodiversity features (i.e., vegetation classes in this exercise) are represented (covered) by protected areas. Next, we compare current representation by protected areas of each feature (e.g., 5% of their spatial distribution covered by protected areas) to a target threshold (e.g., 20% of their spatial distribution covered by protected areas). This target threshold denotes the minimum amount (e.g., minimum proportion of spatial distribution) that we need of each feature to be represented in the protected area system. Ideally, targets should be based on an estimate of how much area or habitat is needed for ecosystem function or species persistence [Taylor et al., 2017]. In practice, targets are generally set using simple rules of thumb (e.g., 10% or 20%), policy [Friedrichs et al., 2018], or species' geographic range size [Butchart et al., 2015, Rodrigues et al., 2004, Jung et al., 2021].

4.2 Feature abundance

Now we will perform some preliminary calculations to explore the data. First, we will calculate how much of each vegetation feature occurs inside each planning unit (i.e., the abundance of the features). To achieve this, we will use the `problem` function to create an empty conservation planning problem that only contains the planning unit and biodiversity data. We will then use the `feature_abundances` function to calculate the total amount of each feature in each planning unit.

```
# create prioritizr problem with only the data
p0 <- problem(pu_data, veg_data, cost_column = "cost")

# print empty problem,
## we can see that only the cost and feature data are defined
print(p0)
```

```
## A conservation problem (<ConservationProblem>)
## +@data
## |+@features:    "Banksia woodlands" , ... (33 total)
## |\@planning units:
## | +@data:       <sftbl_dftbldata.frame> (1130 total)
## | +@costs:      continuous values (between 0.1925 and 61.9273)
## | +@extent:     298809.5764, 5167774.5993, 613818.7743, 5502543.7119 (xmin, ymin, xma
## | \@CRS:        WGS 84 / UTM zone 55S (projected)
## +@formulation
## |+@objective:   none specified
## |+@penalties:   none specified
## |+@targets:     none specified
## |+@constraints: none specified
## |\@decisions:   binary decision
## \@optimization
##  +@portfolio:   shuffle portfolio (‘number_solutions‘ = 1, ...)
##  \@solver:      highs solver (‘gap‘ = 0.1, ‘time_limit‘ = 2147483647, ...)
## # i Use ‘summary(...)‘ to see complete formulation.
```

```
# calculate amount of each feature in each planning unit
abundance_data <- feature_abundances(p0)

# print abundance data
print(abundance_data)
```

```
## # A tibble: 33 x 3
##    feature                                absolute_abundance relative_abundance
##    <chr>                                               <dbl>              <dbl>
##  1 Banksia woodlands                                    2.00                  1
##  2 Boulders/rock with algae, lichen or sc~            140.                    1
##  3 Callitris forests and woodlands                      6.00                  1
##  4 Cool temperate rainforest                         7257.                    1
##  5 Eucalyptus (+/- tall) open forest with~           5699.                    1
```

```
##  6 Eucalyptus open forests with a shrubby~          9180.                  1
##  7 Eucalyptus open woodlands with shrubby~            38.0                 1
##  8 Eucalyptus tall open forest with a fin~          1908.                  1
##  9 Eucalyptus tall open forests and open ~           388.                  1
## 10 Eucalyptus woodlands with a shrubby un~          6145.                  1
## 11 Eucalyptus woodlands with a tussock gr~          1050.                  1
## 12 Eucalyptus woodlands with ferns, herbs~          1933.                  1
## 13 Freshwater, dams, lakes, lagoons or aq~          1883.                  1
## 14 Heathlands                                       2687.                  1
## 15 Leptospermum forests and woodlands                717.                  1
## 16 Low closed forest or tall closed shrub~          3397.                  1
## 17 Mallee with a tussock grass understorey             1                   1
## 18 Melaleuca open forests and woodlands              144.                  1
## 19 Melaleuca shrublands and open shrublan~            23.9                 1
## 20 Mixed chenopod, samphire +/- forbs                 63.3                 1
## 21 Naturally bare, sand, rock, claypan, m~           115.                  1
## 22 Other Acacia tall open shrublands and ~            23.0                 1
## 23 Other forests and woodlands                       235.                  1
## 24 Other open woodlands                              167.                  1
## 25 Other shrublands                                  234.                  1
## 26 Other tussock grasslands                           23.2                 1
## 27 Regrowth or modified forests and woodl~          568.                   1
## 28 Saline or brackish sedgelands or grass~            24.1                 1
## 29 Salt lakes and lagoons                             11.0                 1
## 30 Sedgelands, rushs or reeds                       5047.                  1
## 31 Temperate tussock grasslands                      781.                  1
## 32 Unclassified native vegetation                    118.                  1
## 33 Wet tussock grassland with herbs, sedg~           463.                  1
```

The `abundance_data` object contains three columns. The `feature` column contains the name of each feature (derived from `names(veg_data)`), the `absolute_abundance` column contains the total amount of each feature in all the planning units, and the `relative_abundance` column contains the total amount of each feature in the planning units expressed as a proportion of the total amount in the underlying raster data. Since all the raster cells containing vegetation overlap with the planning units, all of the values in the `relative_abundance` column are equal to one (meaning 100%). Now let's add a new column with the feature abundances expressed in area units (i.e., km^2).

```
# add new column with feature abundances in km^2
abundance_data$absolute_abundance_km2 <-
  (abundance_data$absolute_abundance * prod(res(veg_data))) %>%
  set_units(m^2) %>%
  set_units(km^2)
```

```
# print abundance data
print(abundance_data)
```

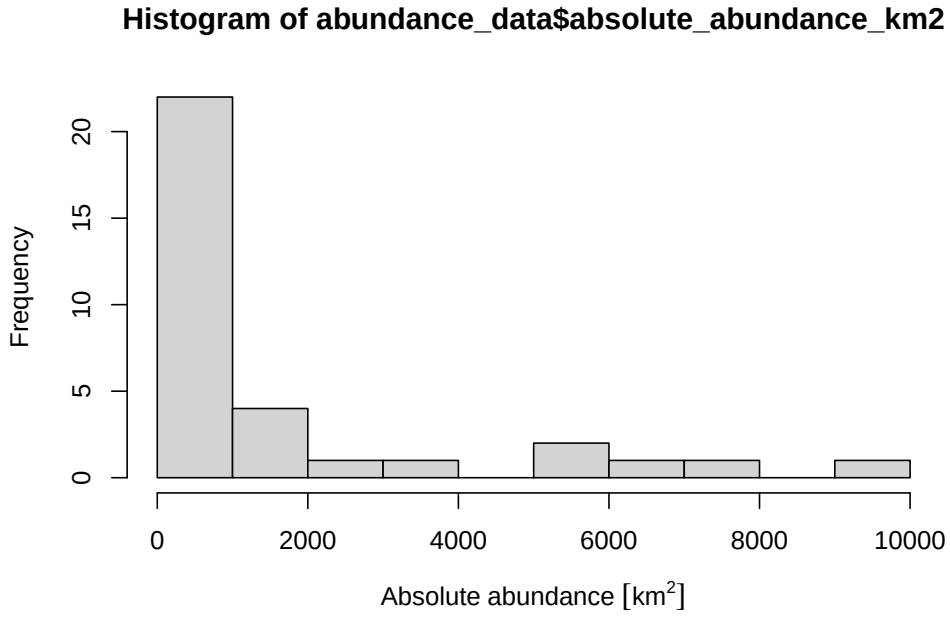
```
## # A tibble: 33 x 4
##    feature          absolute_abundance relative_abundance absolute_abundance_km2
##    <chr>                         <dbl>              <dbl>                  [km^2]
##  1 Banksia woodlan~              2.00                   1                    2.00
##  2 Boulders/rock w~            140.                     1                  140.
##  3 Callitris fores~              6.00                   1                    6.00
##  4 Cool temperate ~           7257.                     1                 7257.
##  5 Eucalyptus (+/-~           5699.                     1                 5699.
##  6 Eucalyptus open~           9180.                     1                 9180.
##  7 Eucalyptus open~             38.0                    1                   38.0
##  8 Eucalyptus tall~           1908.                     1                 1908.
##  9 Eucalyptus tall~            388.                     1                  388.
## 10 Eucalyptus wood~           6145.                     1                 6145.
## 11 Eucalyptus wood~           1050.                     1                 1050.
## 12 Eucalyptus wood~           1933.                     1                 1933.
## 13 Freshwater, dam~           1883.                     1                 1883.
## 14 Heathlands                 2687.                     1                 2687.
## 15 Leptospermum fo~            717.                     1                  717.
## 16 Low closed fore~           3397.                     1                 3397.
## 17 Mallee with a t~              1                      1                    1
## 18 Melaleuca open ~            144.                     1                  144.
## 19 Melaleuca shrub~             23.9                    1                   23.9
## 20 Mixed chenopod,~             63.3                    1                   63.3
## 21 Naturally bare,~            115.                     1                  115.
## 22 Other Acacia ta~             23.0                    1                   23.0
## 23 Other forests a~            235.                     1                  235.
## 24 Other open wood~            167.                     1                  167.
## 25 Other shrublands            234.                     1                  234.
## 26 Other tussock g~             23.2                    1                   23.2
## 27 Regrowth or mod~            568.                     1                  568.
## 28 Saline or brack~             24.1                    1                   24.1
## 29 Salt lakes and ~             11.0                    1                   11.0
## 30 Sedgelands, rus~           5047.                     1                 5047.
## 31 Temperate tusso~            781.                     1                  781.
## 32 Unclassified na~            118.                     1                  118.
## 33 Wet tussock gra~            463.                     1                  463.
```

Now let's explore the abundance data.

```
# calculate the average abundance of the features
mean(abundance_data$absolute_abundance_km2)
```

```
## 1529.413 [km^2]
```

```
# plot histogram of the features' abundances
hist(abundance_data$absolute_abundance_km2, xlab = "Absolute abundance")
```




```
# find the abundance of the feature with the largest abundance
max(abundance_data$absolute_abundance_km2)
```

```
## 9179.876 [km^2]
```

```
# find the name of the feature with the largest abundance
abundance_data$feature[which.max(abundance_data$absolute_abundance_km2)]
```

```
## [1] "Eucalyptus open forests with a shrubby understorey"
```

Now, try to answer the following questions.

1. What is the median abundance of the features (hint: `median`)?
2. What is the abundance of the feature with smallest abundance?
3. What is the name of the feature with smallest abundance?
4. What is the total abundance of all features in the planning units summed together?
5. How many features have a total abundance greater than 100 km^2 (hint: `sum(abundance_values > set_units(threshold_value, km^2))`)?

4.3 Feature representation by protected areas

After calculating the total amount of each feature in the planning units (i.e., the features' abundances), we will now calculate the amount of each feature in the planning units that are covered by protected areas (i.e., feature representation by protected areas). We can complete this task using the `feature_representation` function. This function requires (i) a conservation problem object with the planning unit and biodiversity data and also (ii) an object representing a solution to the problem (i.e an object in the same format as the planning unit data with values indicating if the planning units are selected or not).

```
# create column in planning unit data with binary values (zeros and ones)
# indicating if a planning unit is covered by protected areas or not
pu_data$pa_status <- as.numeric(pu_data$locked_in)

# calculate feature representation by protected areas
repr_data <- eval_feature_representation_summary(p0, pu_data[, "pa_status"])

# print feature representation data
print(repr_data)
```

```
## # A tibble: 33 x 5
##    summary feature                      total_amount absolute_held relative_held
##    <chr>   <chr>                               <dbl>         <dbl>         <dbl>
##  1 overall Banksia woodlands                    2.00         0.367        0.184
##  2 overall Boulders/rock with algae, l~       140.          65.5          0.466
##  3 overall Callitris forests and woodl~         6.00         0.487        0.0812
##  4 overall Cool temperate rainforest         7257.        2992.           0.412
##  5 overall Eucalyptus (+/- tall) open ~      5699.        1398.           0.245
##  6 overall Eucalyptus open forests wit~      9180.        1030.           0.112
##  7 overall Eucalyptus open woodlands w~        38.0         15.1          0.396
##  8 overall Eucalyptus tall open forest~      1908.         189.           0.0992
##  9 overall Eucalyptus tall open forest~       388.          27.4          0.0705
```

```
## 10 overall Eucalyptus woodlands with a~      6145.       1449.          0.236
## 11 overall Eucalyptus woodlands with a~      1050.         11.3         0.0107
## 12 overall Eucalyptus woodlands with f~      1933.        497.          0.257
## 13 overall Freshwater, dams, lakes, la~      1883.        585.          0.311
## 14 overall Heathlands                        2687.       1567.          0.583
## 15 overall Leptospermum forests and wo~       717.        454.          0.633
## 16 overall Low closed forest or tall c~      3397.       1141.          0.336
## 17 overall Mallee with a tussock grass~         1           0           0
## 18 overall Melaleuca open forests and ~       144.         27.4         0.191
## 19 overall Melaleuca shrublands and op~        23.9        22.2         0.930
## 20 overall Mixed chenopod, samphire +/~        63.3        30.0         0.473
## 21 overall Naturally bare, sand, rock,~       115.          3.63        0.0316
## 22 overall Other Acacia tall open shru~        23.0         4.00        0.174
## 23 overall Other forests and woodlands        235.          0.271       0.00115
## 24 overall Other open woodlands               167.         97.4         0.583
## 25 overall Other shrublands                   234.         92.6         0.395
## 26 overall Other tussock grasslands            23.2         0.0677      0.00292
## 27 overall Regrowth or modified forest~       568.          4.92        0.00866
## 28 overall Saline or brackish sedgelan~        24.1         0           0
## 29 overall Salt lakes and lagoons              11.0         0           0
## 30 overall Sedgelands, rushs or reeds        5047.       2505.          0.496
## 31 overall Temperate tussock grasslands       781.          6           0.00768
## 32 overall Unclassified native vegetat~       118.          4.93        0.0419
## 33 overall Wet tussock grassland with ~       463.         43.0         0.0928
```

Similar to the abundance data before, the `repr_data` object contains three columns. The `feature` column contains the name of each feature, the `absolute_held` column shows the total amount of each feature held in the solution (i.e., the planning units covered by protected areas), and the `relative_held` column shows the proportion of each feature held in the solution (i.e., the proportion of each feature's spatial distribution held in protected areas). Since the `absolute_held` values correspond to the number of grid cells in the `veg_data` object with overlap with protected areas, let's convert them to area units (i.e., km^2) so we can report them.

```
# add new column with the areas represented in km^2
repr_data$absolute_held_km2 <-
  (repr_data$absolute_held * prod(res(veg_data))) %>%
  set_units(m^2) %>%
  set_units(km^2)

# print representation data
print(repr_data)
```

```
## # A tibble: 33 x 6
```

```
##    summary feature    total_amount absolute_held relative_held absolute_held_km2
##    <chr>   <chr>             <dbl>         <dbl>         <dbl>            [km^2]
##  1 overall Banksia w~         2.00         0.367        0.184             0.367
##  2 overall Boulders/~       140.          65.5          0.466            65.5
##  3 overall Callitris~         6.00         0.487        0.0812            0.487
##  4 overall Cool temp~      7257.        2992.           0.412          2992.
##  5 overall Eucalyptu~      5699.        1398.           0.245          1398.
##  6 overall Eucalyptu~      9180.        1030.           0.112          1030.
##  7 overall Eucalyptu~        38.0         15.1          0.396            15.1
##  8 overall Eucalyptu~      1908.         189.           0.0992          189.
##  9 overall Eucalyptu~       388.          27.4          0.0705           27.4
## 10 overall Eucalyptu~      6145.        1449.           0.236          1449.
## 11 overall Eucalyptu~      1050.          11.3          0.0107           11.3
## 12 overall Eucalyptu~      1933.         497.           0.257          497.
## 13 overall Freshwate~      1883.         585.           0.311          585.
## 14 overall Heathlands      2687.        1567.           0.583          1567.
## 15 overall Leptosper~       717.         454.           0.633          454.
## 16 overall Low close~      3397.        1141.           0.336          1141.
## 17 overall Mallee wi~         1             0            0                0
## 18 overall Melaleuca~       144.          27.4          0.191            27.4
## 19 overall Melaleuca~        23.9         22.2          0.930            22.2
## 20 overall Mixed che~        63.3         30.0          0.473            30.0
## 21 overall Naturally~       115.           3.63         0.0316            3.63
## 22 overall Other Aca~        23.0          4.00         0.174             4.00
## 23 overall Other for~       235.           0.271        0.00115           0.271
## 24 overall Other ope~       167.          97.4          0.583            97.4
## 25 overall Other shr~       234.          92.6          0.395            92.6
## 26 overall Other tus~        23.2          0.0677       0.00292           0.0677
## 27 overall Regrowth ~       568.          4.92          0.00866           4.92
## 28 overall Saline or~        24.1         0             0                 0
## 29 overall Salt lake~        11.0         0             0                 0
## 30 overall Sedgeland~      5047.        2505.           0.496          2505.
## 31 overall Temperate~       781.           6            0.00768           6
## 32 overall Unclassif~       118.           4.93         0.0419            4.93
## 33 overall Wet tusso~       463.          43.0          0.0928           43.0
```

Now let's investigate how well the species are represented.

1. What is the average proportion of the features held in protected areas (hint: `mean(x, na.rm = TRUE)`?
2. What is the average amount of land in km^2 that features are represented by protected areas?
3. What is the name of the feature with the greatest proportionate coverage by protected areas?

4. What is the name of the feature with the greatest area coverage by protected areas?
5. Do questions two and three have the same answer? Why could this be?
6. Is there a relationship between the total abundance of a feature and how well it is represented by protected areas (hint: `plot(abundance_data$absolute_abundance, repr_data$relative_held)`)?
7. Are any features entirely missing from protected areas (hint: `sum(x == 0)`)?
8. If we set a target of 10% coverage by protected areas, how many features fail to meet this target (hint: `sum(relative_held >= target, na.rm = TRUE)`)?
9. If we set a target of 30% coverage by protected areas, how many features fail to meet this target?

Chapter 5

Spatial prioritizations

5.1 Introduction

Here we will develop prioritizations to identify priority areas for protected area establishment. Specifically, we will be using the prioritizr R package to generate prioritizations. Although other tools are also available for generating prioritizations – such as Marxan [Ardron et al., 2010], and Zonation [Moilanen et al., 2005] – it is beyond the scope of this workshop to examine them. Additionally, it is important to understand that software for generating prioritizations are decision support tools. This means that the software is designed to help you make decisions—it can't make decisions for you.

5.2 Starting out simple

To start things off, let's keep things simple. Let's create a prioritization using the minimum set formulation of the reserve selection problem [Rodrigues et al., 2008]. This formulation means that we want a solution that will meet the targets for our biodiversity features for minimum cost. Here, we will set 5% targets for each vegetation class and use the data in the `cost` column to specify acquisition costs. One advantage of prioritizr is that, unlike the Marxan decision support tool, we do not have calibrate (SPFs) to ensure the solution meets the targets. This is because – when using this formulation — prioritizr should always return solutions that meet the targets. Although we strongly recommend using Gurobi to solve problems (via `add_gurobi_solver`), we will use the HiGHS solver (via `add_highs_solver`) in this workshop since it is easier to install. This is because the Gurobi solver is much faster than the HiGHS solver (see here for installation instructions).

```
# print planning unit data
## note we use head() to show only show the first 6 rows
head(pu_data)
```

```
## Simple feature collection with 6 features and 5 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: 303910.1 ymin: 5485840 xmax: 335610.4 ymax: 5502544
## Projected CRS: WGS 84 / UTM zone 55S
## # A tibble: 6 x 6
##      id  cost locked_in locked_out                                  geom pa_status
##   <int> <dbl> <lgl>     <lgl>                       <MULTIPOLYGON [m]>     <dbl>
## 1     1  60.2 FALSE     TRUE       (((328497 5497704, 326783.8 550005~         0
## 2     2  19.9 FALSE     FALSE      (((307121.6 5490487, 305344.4 5492~         0
## 3     3  59.7 FALSE     TRUE       (((321726.1 5492382, 320111 549459~         0
## 4     4  32.4 FALSE     FALSE      (((304314.5 5494324, 304342.2 5494~         0
## 5     5  26.2 FALSE     FALSE      (((314958.5 5487057, 312336 549064~         0
## 6     6  51.3 FALSE     TRUE       (((327904.3 5491218, 326594.6 5493~         0
```

```
# create prioritization problem
p1 <-
  problem(pu_data, veg_data, cost_column = "cost") %>%
  add_min_set_objective() %>%
  add_relative_targets(0.05) %>% # 5% representation targets
  add_binary_decisions() %>%
  add_highs_solver(verbose = FALSE)

# print problem
print(p1)
```

```
## A conservation problem (<ConservationProblem>)
## +@data
## |+@features:    "Banksia woodlands" , ... (33 total)
## |\@planning units:
## | +@data:       <sftbl_dftbldata.frame> (1130 total)
## | +@costs:      continuous values (between 0.1925 and 61.9273)
## | +@extent:     298809.5764, 5167774.5993, 613818.7743, 5502543.7119 (xmin, ymin, xma
## | \@CRS:        WGS 84 / UTM zone 55S (projected)
## +@formulation
## |+@objective:   minimum set objective
```

```
## |+@penalties:   none specified
## |+@targets:     relative targets (between 0.05 and 0.05)
## |+@constraints: none specified
## |\@decisions:   binary decision
## \@optimization
##  +@portfolio:   shuffle portfolio (‘number_solutions‘ = 1, ...)
##  \@solver:      highs solver (‘gap‘ = 0.1, ‘time_limit‘ = 2147483647, ...)
## # i Use ‘summary(...)‘ to see complete formulation.
```

```
# solve problem
s1 <- solve(p1)

# print solution, the solution_1 column contains the solution values
# indicating if a planning unit is (1) selected or (0) not
## note we use head() to show only show the first 6 rows
head(s1)
```

```
## Simple feature collection with 6 features and 6 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: 303910.1 ymin: 5485840 xmax: 335610.4 ymax: 5502544
## Projected CRS: WGS 84 / UTM zone 55S
## # A tibble: 6 x 7
##      id  cost locked_in locked_out pa_status solution_1
##   <int> <dbl> <lgl>     <lgl>          <dbl>      <dbl>
## 1     1  60.2 FALSE     TRUE               0          0
## 2     2  19.9 FALSE     FALSE              0          0
## 3     3  59.7 FALSE     TRUE               0          0
## 4     4  32.4 FALSE     FALSE              0          0
## 5     5  26.2 FALSE     FALSE              0          0
## 6     6  51.3 FALSE     TRUE               0          0
## # i 1 more variable: geom <MULTIPOLYGON [m]>
```

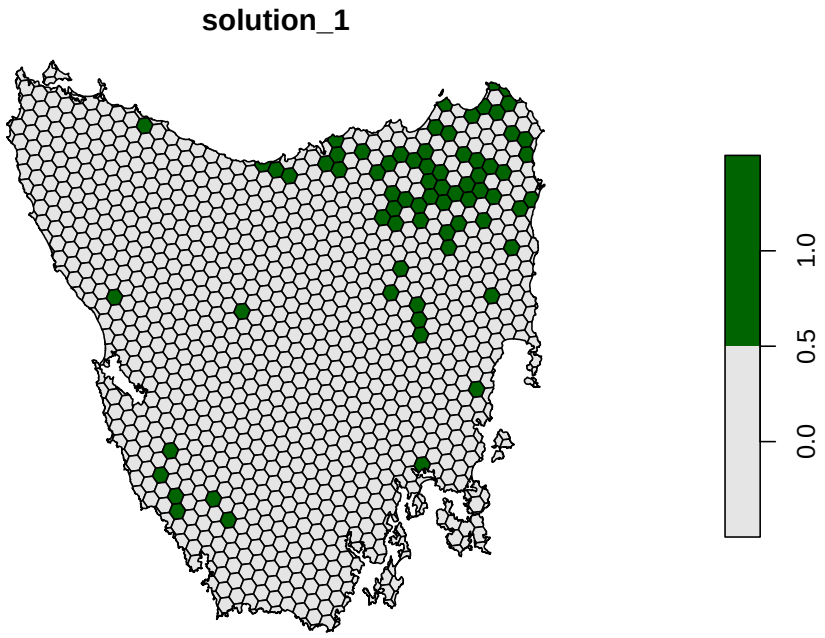
```
# calculate number of planning units selected in the prioritization
sum(s1$solution_1)
```

```
## [1] 72
```

```
# calculate total cost of the prioritization
sum(s1$solution_1 * s1$cost)
```

```
## [1] 626.5873
```

```
# plot solution
plot(s1[, "solution_1"], pal = c("grey90", "darkgreen"))
```



Now let's examine the solution.

1. How many planing units were selected in the prioritization?
2. What proportion of planning units were selected in the prioritization?
3. Is there a pattern in the spatial distribution of the priority areas?
4. Can you verify that all of the targets were met in the prioritization (hint: `feature_representation(p1, s1[, "solution_1"])`)?

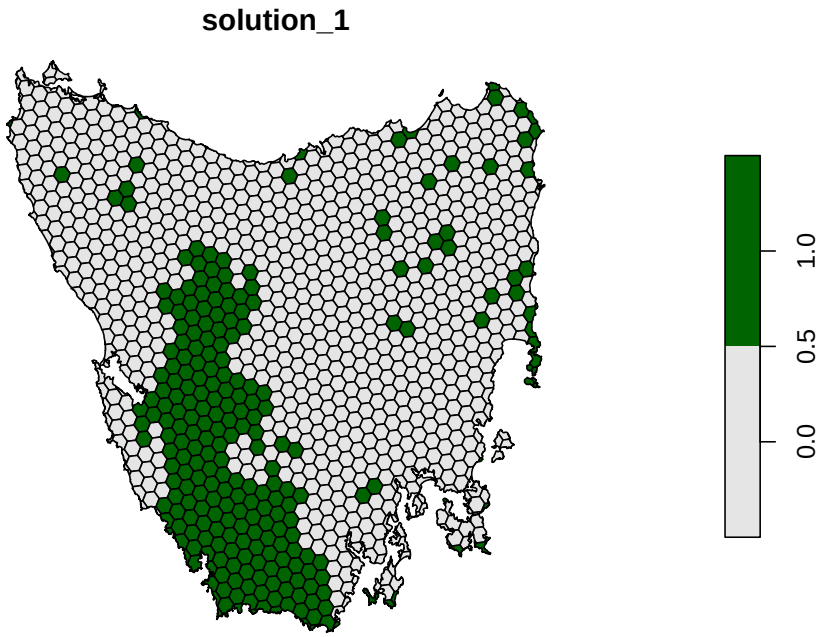
5.3 Adding complexity

Our first prioritization suffers many limitations, so let's add additional constraints to the problem to make it more useful. First, let's lock in planing units that are already by covered protected areas. If some vegetation communities are already secured inside existing protected areas, then we might not need to add as many new protected areas to the existing protected area system to meet their targets. Since our planning unit data (`pu_data`) already contains this information in the `locked_in` column, we can use this column name to specify which planning units should be locked in.

```
# create prioritization problem
p2 <-
  problem(pu_data, veg_data, cost_column = "cost") %>%
  add_min_set_objective() %>%
  add_relative_targets(0.05) %>%
  add_locked_in_constraints("locked_in") %>%
  add_binary_decisions() %>%
  add_highs_solver(verbose = FALSE)

# print problem
print(p2)
```

```
## A conservation problem (<ConservationProblem>)
## +@data
## |+@features:    "Banksia woodlands" , ... (33 total)
## |\@planning units:
## | +@data:       <sftbl_dftbldata.frame> (1130 total)
## | +@costs:      continuous values (between 0.1925 and 61.9273)
## | +@extent:     298809.5764, 5167774.5993, 613818.7743, 5502543.7119 (xmin, ymin, xma
## | \@CRS:        WGS 84 / UTM zone 55S (projected)
## +@formulation
## |+@objective:   minimum set objective
## |+@penalties:   none specified
## |+@targets:     relative targets (between 0.05 and 0.05)
## |+@constraints:
## ||\@1:          locked in constraints (257 planning units)
## |\@decisions:   binary decision
## \@optimization
##  +@portfolio:   shuffle portfolio (`number_solutions` = 1, ...)
##  \@solver:      highs solver (`gap` = 0.1, `time_limit` = 2147483647, ...)
## # i Use `summary(...)` to see complete formulation.
```

```
# solve problem
s2 <- solve(p2)

# plot solution
plot(s2[, "solution_1"], pal = c("grey90", "darkgreen"))
```



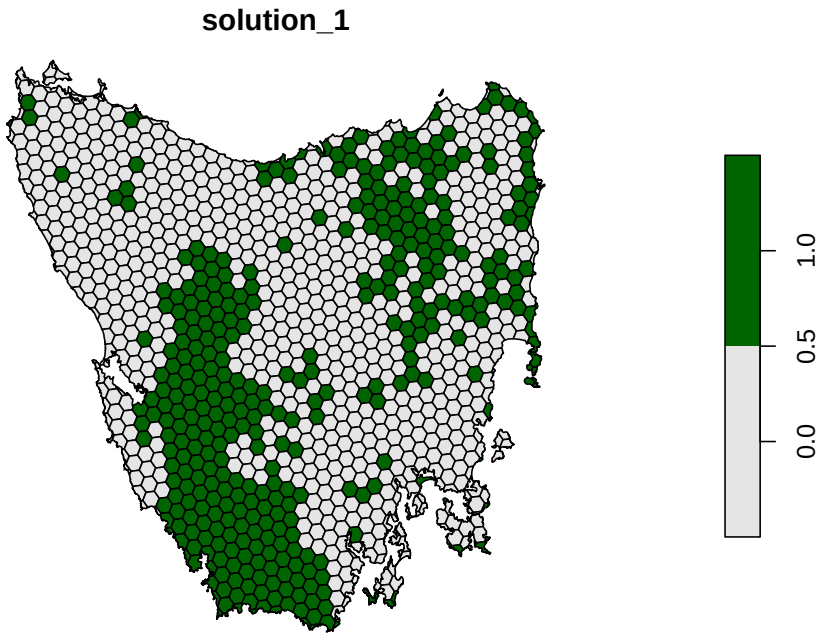
Let's pretend that we talked to an expert on the vegetation communities in our study system and they recommended that a 30% target was needed for each vegetation class. So, equipped with this information, let's set the targets to 20%.

```
# create prioritization problem
p3 <-
  problem(pu_data, veg_data, cost_column = "cost") %>%
  add_min_set_objective() %>%
  add_relative_targets(0.3) %>%
  add_locked_in_constraints("locked_in") %>%
  add_binary_decisions() %>%
  add_highs_solver(verbose = FALSE)

# print problem
print(p3)
```

```
## A conservation problem (<ConservationProblem>)
## +@data
```

```
## |+@features:    "Banksia woodlands" , ... (33 total)
## |\@planning units:
## | +@data:       <sftbl_dftbldata.frame> (1130 total)
## | +@costs:      continuous values (between 0.1925 and 61.9273)
## | +@extent:     298809.5764, 5167774.5993, 613818.7743, 5502543.7119 (xmin, ymin, xma
## | \@CRS:        WGS 84 / UTM zone 55S (projected)
## +@formulation
## |+@objective:   minimum set objective
## |+@penalties:   none specified
## |+@targets:     relative targets (between 0.3 and 0.3)
## |+@constraints:
## ||\@1:          locked in constraints (257 planning units)
## |\@decisions:   binary decision
## \@optimization
##  +@portfolio:   shuffle portfolio (‘number_solutions‘ = 1, ...)
##  \@solver:      highs solver (‘gap‘ = 0.1, ‘time_limit‘ = 2147483647, ...)
## # i Use ‘summary(...)‘ to see complete formulation.
```

```
# solve problem
s3 <- solve(p3)

# plot solution
plot(s3[, "solution_1"], pal = c("grey90", "darkgreen"))
```



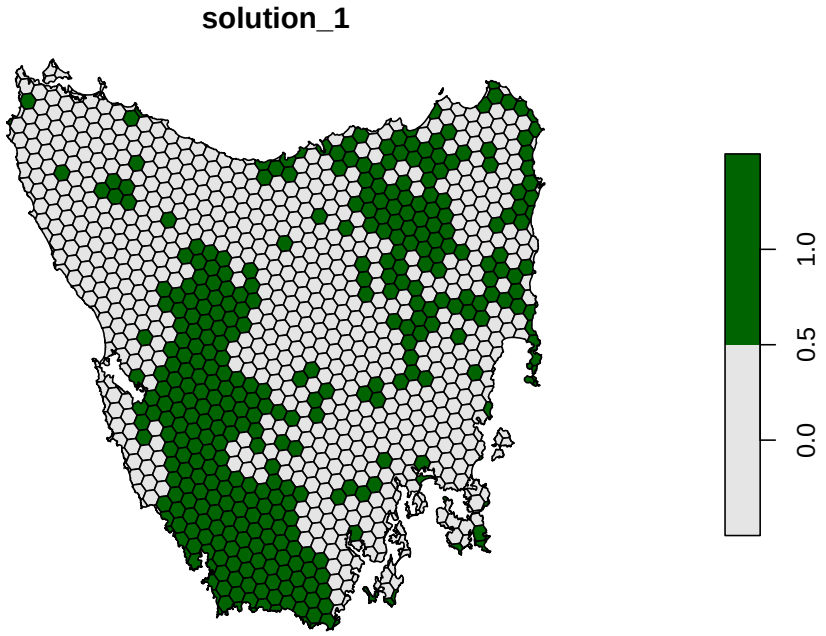
Next, let's lock out highly degraded areas. Similar to before, this data is present in our planning unit data so we can use the `locked_out` column name to achieve this.

```
# create prioritization problem
p4 <-
  problem(pu_data, veg_data, cost_column = "cost") %>%
  add_min_set_objective() %>%
  add_relative_targets(0.3) %>%
  add_locked_in_constraints("locked_in") %>%
  add_locked_out_constraints("locked_out") %>%
  add_binary_decisions() %>%
  add_highs_solver(verbose = FALSE)
```

```
# print problem
print(p4)
```

```
## A conservation problem (<ConservationProblem>)
## +@data
## |+@features:    "Banksia woodlands" , ... (33 total)
## |\@planning units:
## | +@data:       <sftbl_dftbldata.frame> (1130 total)
## | +@costs:      continuous values (between 0.1925 and 61.9273)
## | +@extent:     298809.5764, 5167774.5993, 613818.7743, 5502543.7119 (xmin, ymin, xma
## | \@CRS:        WGS 84 / UTM zone 55S (projected)
## +@formulation
## |+@objective:   minimum set objective
## |+@penalties:   none specified
## |+@targets:     relative targets (between 0.3 and 0.3)
## |+@constraints:
## ||+@1:          locked in constraints (257 planning units)
## ||\@2:          locked out constraints (165 planning units)
## |\@decisions:   binary decision
## \@optimization
##  +@portfolio:   shuffle portfolio (`number_solutions` = 1, ...)
##  \@solver:      highs solver (`gap` = 0.1, `time_limit` = 2147483647, ...)
## # i Use `summary(...)` to see complete formulation.
```

```
# solve problem
s4 <- solve(p4)

# plot solution
plot(s4[, "solution_1"],  pal = c("grey90", "darkgreen"))
```



Now, let's compare the solutions.

1. What is the cost of the planning units selected in `s2`, `s3`, and `s4`?
2. How many planning units are in `s2`, `s3`, and `s4`?
3. Do the solutions with more planning units have a greater cost? Why or why not?
4. Why does the first solution (`s1`) cost less than the second solution with protected areas locked into the solution (`s2`)?
5. Why does the third solution (`s3`) cost less than the fourth solution solution with highly degraded areas locked out (`s4`)?
6. Since planning units covered by existing protected areas have already been purchased, what is the cost for expanding the protected area system based on on the fourth prioritization (`s4`) (hint: total cost minus the cost of locked in planning units)?
7. What happens if you specify targets that exceed the total amount of vegetation in the study area and try to solve the problem? You can do this by modifying the code to make `p4` with `add_absolute_targets(1000)` instead of `add_relative_targets(0.3)` and generating a new solution.

5.4 Penalizing fragmentation

Plans for protected area systems should facilitate gene flow and dispersal between individual reserves in the system [Beger et al., 2010, Hanson et al., 2022]. However, the prioritizations we have made so far have been highly fragmented. Similar to the Marxan decision support tool, we can add penalties to our conservation planning problem to penalize fragmentation (i.e. total exposed boundary length) and we also need to set a useful penalty value when adding such penalties (akin to Marxan's boundary length multiplier value; BLM) [Beyer et al., 2016]. If we set our penalty value too low, then we will end up with a solution that is identical to the solution with no added penalties. If we set our penalty value too high, then prioritizr will take a long time to solve the problem and we will end up with a solution that contains lots of extra planning units that are not needed (since the penalty value is so high that minimizing fragmentation is more important than cost). As a rule of thumb, we generally want penalty values between 0.00001 and 0.01 but finding a useful penalty value requires calibration. The "correct" penalty value depends on the size of the planning units, the main objective values (e.g., cost values), and the effect of fragmentation on biodiversity persistence. Let's create a new problem that is similar to our previous problem (`p4`) – except that it contains boundary length penalties and a slightly higher optimality gap to reduce runtime (default is 0.1) – and solve it. Since our planning unit data is in a spatial format (i.e., vector or raster data), prioritizr can automatically calculate the boundary data for us.

```
# create prioritization problem
p5 <-
  problem(pu_data, veg_data, cost_column = "cost") %>%
  add_min_set_objective() %>%
  add_boundary_penalties(penalty = 0.001) %>%
  add_relative_targets(0.3) %>%
  add_locked_in_constraints("locked_in") %>%
  add_locked_out_constraints("locked_out") %>%
  add_binary_decisions() %>%
  add_highs_solver(verbose = FALSE)

# print problem
print(p5)
```

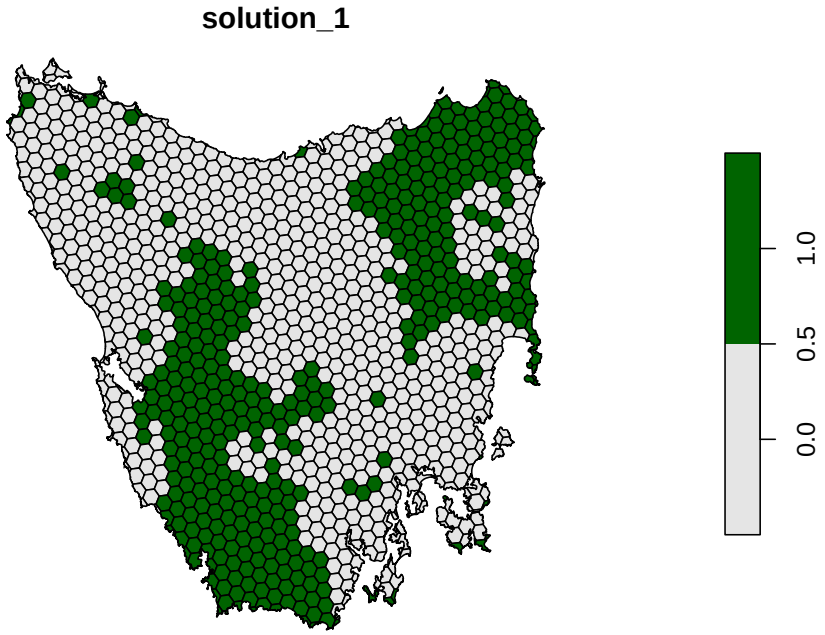
```
## A conservation problem (<ConservationProblem>)
## +@data
## |+@features:    "Banksia woodlands" , ... (33 total)
## |\@planning units:
## | +@data:       <sftbl_dftbldata.frame> (1130 total)
## | +@costs:      continuous values (between 0.1925 and 61.9273)
## | +@extent:     298809.5764, 5167774.5993, 613818.7743, 5502543.7119 (xmin, ymin, xma
## | \@CRS:        WGS 84 / UTM zone 55S (projected)
```

```
## +@formulation
## |+@objective:   minimum set objective
## |+@penalties:
## ||\@1:         boundary penalties (‘penalty‘ = 0.001, ‘edge_factor‘ = 0.5, ...)
## |+@targets:    relative targets (between 0.3 and 0.3)
## |+@constraints:
## ||+@1:         locked in constraints (257 planning units)
## ||\@2:         locked out constraints (165 planning units)
## |\@decisions:  binary decision
## \@optimization
##  +@portfolio:  shuffle portfolio (‘number_solutions‘ = 1, ...)
##  \@solver:     highs solver (‘gap‘ = 0.1, ‘time_limit‘ = 2147483647, ...)
## # i Use ‘summary(...)‘ to see complete formulation.
```

```
# solve problem,
s5 <- solve(p5)

# print solution
## note we use head() to show only show the first 6 rows
head(s5)
```

```
## Simple feature collection with 6 features and 6 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: 303910.1 ymin: 5485840 xmax: 335610.4 ymax: 5502544
## Projected CRS: WGS 84 / UTM zone 55S
## # A tibble: 6 x 7
##      id  cost locked_in locked_out pa_status solution_1
##   <int> <dbl> <lgl>     <lgl>          <dbl>      <dbl>
## 1     1  60.2 FALSE     TRUE               0          0
## 2     2  19.9 FALSE     FALSE              0          0
## 3     3  59.7 FALSE     TRUE               0          0
## 4     4  32.4 FALSE     FALSE              0          0
## 5     5  26.2 FALSE     FALSE              0          0
## 6     6  51.3 FALSE     TRUE               0          0
## # i 1 more variable: geom <MULTIPOLYGON [m]>
```

```
# plot solution
plot(s5[, "solution_1"], pal = c("grey90", "darkgreen"))
```



Now let's compare the solutions to the problems with (`s5`) and without (`s4`) the boundary length penalties.

1. What is the cost the fourth (`s4`) and fifth (`s5`) solutions? Why does the fifth solution (`s5`) cost more than the fourth (`s4`) solution?
2. Try setting the penalty value to 0.000000001 (i.e. `1e-9`) instead of 0.0005. What is the cost of the solution now? Is it different from the fourth solution (`s4`) (hint: try plotting the solutions to visualize them)? Is this is a useful penalty value? Why?
3. Try setting the penalty value to 0.5. What is the cost of the solution now? Is it different from the fourth solution (`s4`) (hint: try plotting the solutions to visualize them)? Is this a useful penalty value? Why?

5.5 Budget limited prioritizations

In the real-world, the funding available for conservation is often very limited. As a consequence, decision makers often need prioritizations where the total cost of priority areas does not exceed a budget. In our fourth prioritization (`s4`), we found that we would need to spend an additional $1334 million AUD to ensure that each vegetation community is adequately represented in the protected area system. But what if the funds available for establishing new protected areas were limited to $100 million AUD? In this case, we need a "budget limited prioritization". Budget limited prioritizations aim to maximize some measure of conservation benefit subject to a budget (e.g., number of species with at least one occurrence in the protected area system, or phylogenetic diversity). Let's create a prioritization that

aims to minimize the target shortfalls as much as possible across all features whilst keeping within a pre-specified budget [following Jung et al., 2021].

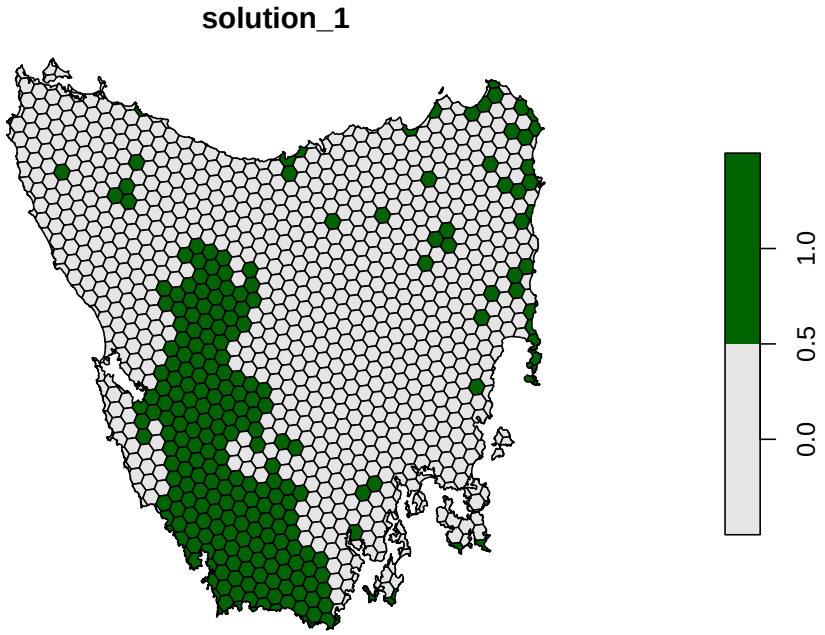
```
# funds for additional land acquisition (same units as cost data)
funds <- 100

# calculate the total budget for the prioritization
budget <- funds + sum(s4$cost * s4$locked_in)
print(budget)
```

```
## [1] 8575.56
```

```
# create prioritization problem
p6 <-
  problem(pu_data, veg_data, cost_column = "cost") %>%
  add_min_shortfall_objective(budget) %>%
  add_relative_targets(0.3) %>%
  add_locked_in_constraints("locked_in") %>%
  add_locked_out_constraints("locked_out") %>%
  add_binary_decisions() %>%
  add_highs_solver(verbose = FALSE)

# print problem
print(p6)
```

```
## A conservation problem (<ConservationProblem>)
## +@data
## |+@features:    "Banksia woodlands" , ... (33 total)
## |\@planning units:
## | +@data:       <sftbl_dftbldata.frame> (1130 total)
## | +@costs:      continuous values (between 0.1925 and 61.9273)
## | +@extent:     298809.5764, 5167774.5993, 613818.7743, 5502543.7119 (xmin, ymin, xma
## | \@CRS:        WGS 84 / UTM zone 55S (projected)
## +@formulation
## |+@objective:   minimum shortfall objective (‘budget‘ = 8575.5601)
## |+@penalties:   none specified
## |+@targets:     relative targets (between 0.3 and 0.3)
## |+@constraints:
## ||+@1:          locked in constraints (257 planning units)
## ||\@2:          locked out constraints (165 planning units)
## |\@decisions:   binary decision
## \@optimization
##  +@portfolio:   shuffle portfolio (‘number_solutions‘ = 1, ...)
##  \@solver:      highs solver (‘gap‘ = 0.1, ‘time_limit‘ = 2147483647, ...)
## # i Use ‘summary(...)‘ to see complete formulation.
```

```
# solve problem
s6 <- solve(p6)

# plot solution
plot(s6[, "solution_1"], pal = c("grey90", "darkgreen"))
```




```
# calculate feature representation
r6 <- eval_feature_representation_summary(p6, s6[, "solution_1"])

# calculate number of features with targets met
sum(r6$relative_held >= 0.3, na.rm = TRUE)
```

```
## [1] 15
```

```
# calculate average proportion of each feature represented by solution
mean(r6$relative_held, na.rm = TRUE)
```

```
## [1] 0.3238168
```

```
# find out which features have their targets met
print(r6$feature[r6$relative_held >= 0.3])
```

```
##  [1] "Boulders/rock with algae, lichen or scattered plants, or alpine fjaeldmarks"
##  [2] "Cool temperate rainforest"
##  [3] "Eucalyptus open woodlands with shrubby understorey"
```

```
##  [4] "Eucalyptus tall open forests and open forests with ferns, herbs, sedges, rushes
##  [5] "Freshwater, dams, lakes, lagoons or aquatic plants"
##  [6] "Heathlands"
##  [7] "Leptospermum forests and woodlands"
##  [8] "Low closed forest or tall closed shrublands (including Acacia, Melaleuca and Ba
##  [9] "Mallee with a tussock grass understorey"
## [10] "Melaleuca shrublands and open shrublands"
## [11] "Mixed chenopod, samphire +/- forbs"
## [12] "Other Acacia tall open shrublands and shrublands"
## [13] "Other open woodlands"
## [14] "Other shrublands"
## [15] "Sedgelands, rushs or reeds"
```

We can also add weights to specify that it is more important to meet the targets for certain features and less important for other features. A common approach for weighting features is to assign a greater importance to features with smaller spatial distributions. The rationale behind this weighting method is that features with smaller spatial distributions are at greater risk of extinction. So, let's calculate some weights for our vegetation communities and see how weighting the features changes our prioritization.

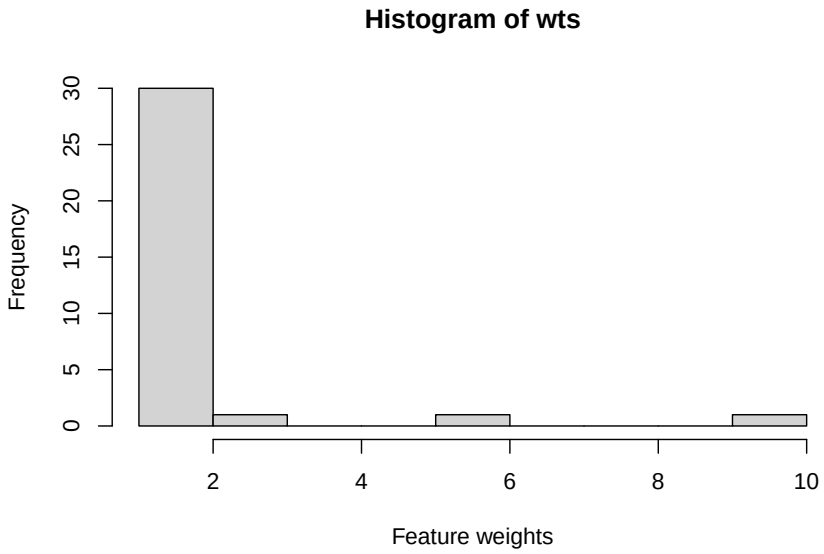
```
# calculate weights as the log inverse number of grid cells that each vegetation
# class occupies, rescaled between 1 and 100
wts <- 1 / global(veg_data, "sum", na.rm = TRUE)[[1]]
wts <- scales::rescale(wts, to = c(1, 10))

# print the name of the feature with smallest weight
names(veg_data)[which.min(wts)]
```

```
## [1] "Eucalyptus open forests with a shrubby understorey"
```

```
# print the name of the feature with greatest weight
names(veg_data)[which.max(wts)]
```

```
## [1] "Mallee with a tussock grass understorey"
```

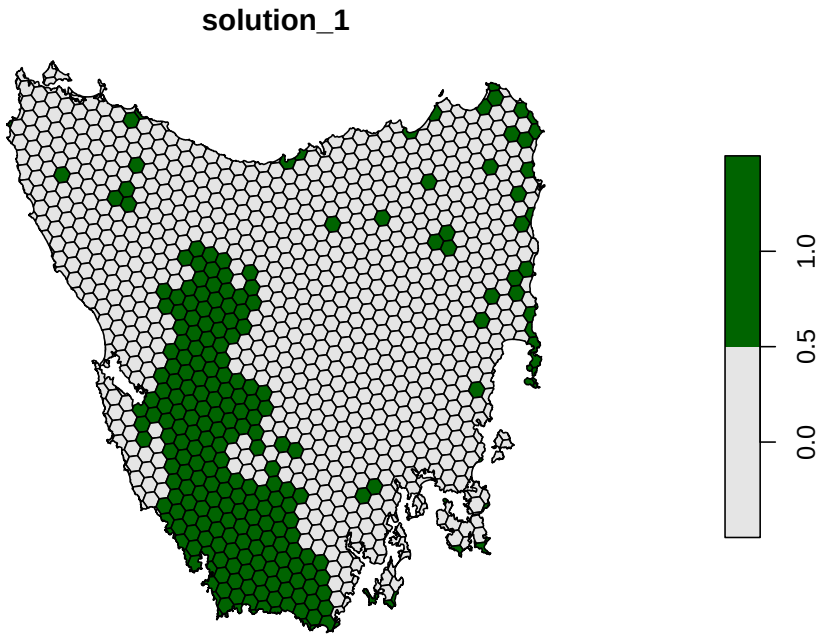
```
# plot histogram of weights
hist(wts, xlab = "Feature weights")

# create prioritization problem with weights
p7 <-
  problem(pu_data, veg_data, cost_column = "cost") %>%
  add_min_shortfall_objective(budget) %>%
  add_relative_targets(0.3) %>%
  add_feature_weights(wts) %>%
  add_locked_in_constraints("locked_in") %>%
  add_locked_out_constraints("locked_out") %>%
  add_binary_decisions() %>%
  add_highs_solver(verbose = FALSE)

# print problem
print(p7)
```




```
## A conservation problem (<ConservationProblem>)
## +@data
## |+@features:    "Banksia woodlands" , ... (33 total)
## |\@planning units:
## | +@data:       <sftbl_dftbldata.frame> (1130 total)
## | +@costs:      continuous values (between 0.1925 and 61.9273)
## | +@extent:     298809.5764, 5167774.5993, 613818.7743, 5502543.7119 (xmin, ymin, xma
## | \@CRS:        WGS 84 / UTM zone 55S (projected)
## +@formulation
## |+@objective:   minimum shortfall objective (‘budget‘ = 8575.5601)
## |+@penalties:
```

```
## ||\@1:          feature weights (‘weights‘ = asymmetric continuous values (non-zero v
## |+@targets:     relative targets (between 0.3 and 0.3)
## |+@constraints:
## ||+@1:          locked in constraints (257 planning units)
## ||\@2:          locked out constraints (165 planning units)
## |\@decisions:   binary decision
## \@optimization
##  +@portfolio:   shuffle portfolio (‘number_solutions‘ = 1, ...)
##  \@solver:      highs solver (‘gap‘ = 0.1, ‘time_limit‘ = 2147483647, ...)
## # i Use ‘summary(...)‘ to see complete formulation.
```

```
# solve problem
s7 <- solve(p7)

# plot solution
plot(s7[, "solution_1"], pal = c("grey90", "darkgreen"))
```




```
# calculate feature representation
r7 <- eval_feature_representation_summary(p7, s7[, "solution_1"])

# calculate number of features with targets met
sum(r7$relative_held >= 0.3, na.rm = TRUE)
```

```
## [1] 15
```

```
# calculate average proportion of each feature represented by solution
mean(r6$relative_held, na.rm = TRUE)
```

```
## [1] 0.3238168
```

```
# find out which features have their targets met when we add weights
print(r7$feature[r7$relative_held >= 0.3])
```

```
##  [1] "Banksia woodlands"
##  [2] "Boulders/rock with algae, lichen or scattered plants, or alpine fjaeldmarks"
##  [3] "Cool temperate rainforest"
##  [4] "Eucalyptus open woodlands with shrubby understorey"
##  [5] "Freshwater, dams, lakes, lagoons or aquatic plants"
##  [6] "Heathlands"
##  [7] "Leptospermum forests and woodlands"
##  [8] "Low closed forest or tall closed shrublands (including Acacia, Melaleuca and Ba
##  [9] "Mallee with a tussock grass understorey"
## [10] "Melaleuca shrublands and open shrublands"
## [11] "Mixed chenopod, samphire +/- forbs"
## [12] "Other Acacia tall open shrublands and shrublands"
## [13] "Other open woodlands"
## [14] "Other shrublands"
## [15] "Sedgelands, rushs or reeds"
```

1. What is the name of the feature with the smallest weight?
2. What is the cost of the sixth (`s6`) and seventh (`s7`) solutions?
3. Does there seem to be a big difference in which planning units were selected in the sixth (`s6`) and seventh (`s7`) solutions?
4. Is there a difference between which features are adequately represented in the sixth (`s6`) and seventh (`s7`) solutions? If so, what is the difference?

Chapter 6

Importance

6.1 Introduction

Systematic conservation planning involves identifying priority areas for conservation actions [Margules and Pressey, 2000]. As we saw in the previous section, we can generate a spatial prioritization that optimizes a particular objective, given a set of constraints, to identify a set of priority areas for management. This information is very useful because it provides a complete and cost-effective plan for achieving our conservation goals. However, when we just look at the priority areas in a spatial prioritization, we don't necessarily know which priority areas – among all the priority areas in the spatial prioritization – are more or less important for conservation. For example, if we generated a spatial prioritization based on threatened and non-threatened species, it would be useful to know which priority areas are necessary to protect because they contain species that are not found anywhere else in the study area. To obtain this information, we can calculate importance scores for the planning units selected in the prioritization. This information can be useful for scheduling implementation of conservation plans and finding compromises for stakeholder discussions [Pressey, 1999].

6.2 Quantifying irreplaceability

To keep things simple, let's start by creating a new conservation planning problem and solving it to generate a spatial prioritization. This will be very similar to one of the prioritizations that we generated in the previous section. Specifically, we will use the minium set objective, 30% representation targets, locked in, locked out constraints, and binary decisions.

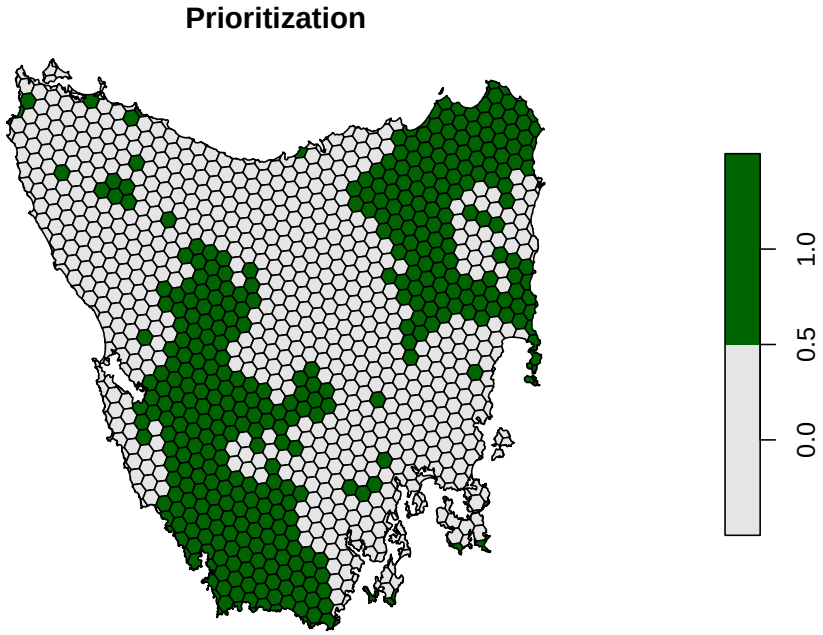
```
# create prioritization problem
p8 <-
  problem(pu_data, veg_data, cost_column = "cost") %>%
  add_min_set_objective() %>%
  add_boundary_penalties(penalty = 0.001) %>%
  add_relative_targets(0.3) %>%
  add_locked_in_constraints("locked_in") %>%
  add_locked_out_constraints("locked_out") %>%
  add_binary_decisions() %>%
  add_highs_solver(verbose = FALSE)

# print problem
print(p8)
```

```
## A conservation problem (<ConservationProblem>)
## +@data
## |+@features:    "Banksia woodlands" , ... (33 total)
## |\@planning units:
## | +@data:       <sftbl_dftbldata.frame> (1130 total)
## | +@costs:      continuous values (between 0.1925 and 61.9273)
## | +@extent:     298809.5764, 5167774.5993, 613818.7743, 5502543.7119 (xmin, ymin, xma
## | \@CRS:        WGS 84 / UTM zone 55S (projected)
## +@formulation
## |+@objective:   minimum set objective
## |+@penalties:
## ||\@1:          boundary penalties (‘penalty‘ = 0.001, ‘edge_factor‘ = 0.5, ...)
## |+@targets:     relative targets (between 0.3 and 0.3)
## |+@constraints:
## ||+@1:          locked in constraints (257 planning units)
## ||\@2:          locked out constraints (165 planning units)
## |\@decisions:   binary decision
## \@optimization
##  +@portfolio:   shuffle portfolio (‘number_solutions‘ = 1, ...)
##  \@solver:      highs solver (‘gap‘ = 0.1, ‘time_limit‘ = 2147483647, ...)
## # i Use ‘summary(...)‘ to see complete formulation.
```

```
# solve problem,
s8 <- solve(p8)

# print solution
## note we use head() to show only show the first 6 rows
head(s8)
```

```
## Simple feature collection with 6 features and 6 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: 303910.1 ymin: 5485840 xmax: 335610.4 ymax: 5502544
## Projected CRS: WGS 84 / UTM zone 55S
## # A tibble: 6 x 7
##      id  cost locked_in locked_out pa_status solution_1
##   <int> <dbl> <lgl>     <lgl>          <dbl>      <dbl>
## 1     1  60.2 FALSE     TRUE               0          0
## 2     2  19.9 FALSE     FALSE              0          0
## 3     3  59.7 FALSE     TRUE               0          0
## 4     4  32.4 FALSE     FALSE              0          0
## 5     5  26.2 FALSE     FALSE              0          0
## 6     6  51.3 FALSE     TRUE               0          0
## # i 1 more variable: geom <MULTIPOLYGON [m]>
```

```
# plot solution
plot(
  s8[, "solution_1"], main = "Prioritization",
  pal = c("grey90", "darkgreen")
)
```



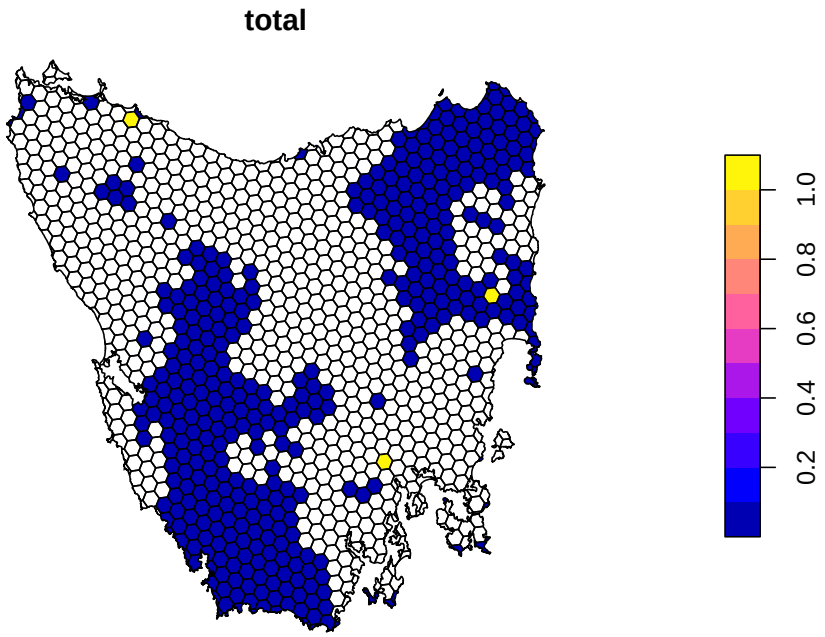
Now, we will calculate importance scores. Specifically, we will calculate importance scores based on irreplaceability metric developed by Ferrier *et al.* [2000]. These scores describe how important each planning unit is for meeting the representation targets. Briefly, the metric

calculates a score for each feature separately – so we can tell which planning units are more important for particular features – and a total score describing the overall importance each planning unit has meeting all the targets. Although the disadvantage of this method is that it does not account for planning unit costs [c.f., the replacement cost metric, Cabeza and Moilanen, 2006], it is useful because it accounts for the representation targets and can be calculated relatively quickly for problems with many planning units and features.

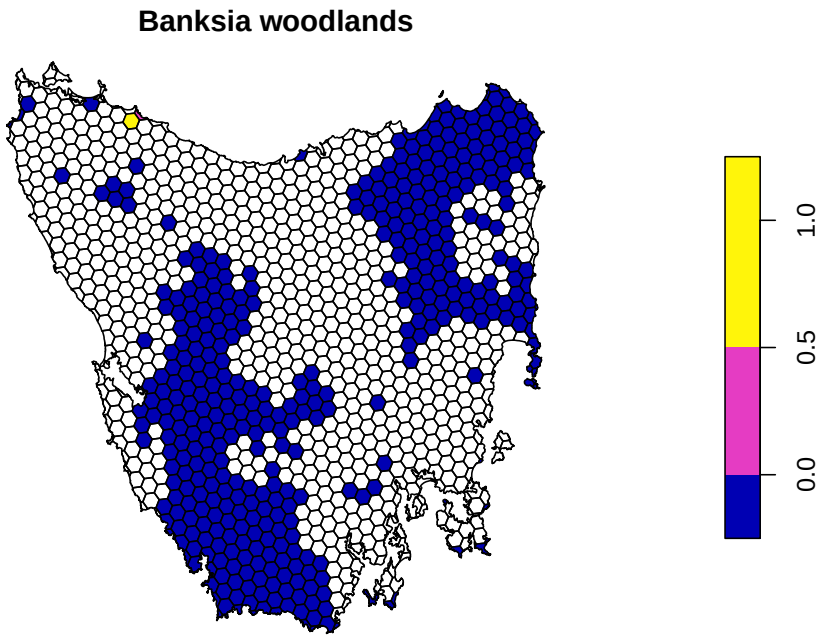
```
# calculate Ferrier scores
i8 <- eval_ferrier_importance(p8, s8[, "solution_1"])

# set NA values for planning units not selected in solution
i8 <-
  i8 %>%
  mutate_at(
    c("total", names(veg_data)),
    function(x) x * if_else(s8$solution_1 > 0.5, 1, NA_real_)
  )


# print scores
## note we use head() to show only show the first 6 rows
head(i8)
```

```
## Simple feature collection with 6 features and 34 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: 303910.1 ymin: 5485840 xmax: 335610.4 ymax: 5502544
## Projected CRS: WGS 84 / UTM zone 55S
## # A tibble: 6 x 35
##   `Banksia woodlands` Boulders/rock with algae, lichen ~1 Callitris forests an~2
##                 <dbl>                               <dbl>                  <dbl>
## 1                  NA                                  NA                     NA
## 2                  NA                                  NA                     NA
## 3                  NA                                  NA                     NA
## 4                  NA                                  NA                     NA
## 5                  NA                                  NA                     NA
## 6                  NA                                  NA                     NA
## # i abbreviated names:
## #   1: `Boulders/rock with algae, lichen or scattered plants, or alpine fjaeldmarks`,
## #   2: `Callitris forests and woodlands`
## # i 32 more variables: `Cool temperate rainforest` <dbl>,
## #   `Eucalyptus (+/- tall) open forest with a dense broad-leaved and/or tree-fern und
## #   `Eucalyptus open forests with a shrubby understorey` <dbl>,
## #   `Eucalyptus open woodlands with shrubby understorey` <dbl>, ...
```

```
# plot total scores across all features
plot(i8[, "total"])
```




```
# plot scores for first feature
plot(i8[, names(veg_data)[1]])
```



```
# plot scores for second feature
plot(i8[, names(veg_data)[1]])
```

Here we can see that some planning units in the prioritization have much higher importance scores than other planning units. If you're familiar with Marxan, the importance scores here convey a similar concept to the selection frequency. However, the advantage with this approach is that you don't need to generate tens of thousands of solutions in order to evaluate the relative importance of different planning units. Additionally, you can see which planning units are more, or less, important for particular features. This can be useful to help understand why certain planning units were selected by the prioritization.

1. Which parts of the study area have the highest importance values?
2. How do the total importance values change when you decrease the targets from 30% to 10%?

Chapter 7

Answers

This chapter contains the answers to the questions presented in the earlier chapters. The answers are provided here so you can check if your answers are correct.

7.1 Data

7.1.1 Planning unit data

1. `nrow(pu_data)`
2. `max(pu_data$cost)`
3. `sum(pu_data$locked_in)`
4. `mean(pu_data$locked_in)`
5. `sum(pu_data$locked_out)`
6. `mean(pu_data$locked_out)`
7. `assert_that(min(c(pu_data$locked_in, pu_data$locked_out)) == 0)`
   `assert_that(max(c(pu_data$locked_in, pu_data$locked_out)) == 1)`
8. `all(is.finite(pu_data$cost))`
9. `assert_that(sum(duplicated(pu_data$id)) == 0)`
10. Yes, the eastern side of Tasmania is generally much cheaper than the western side.
11. Yes, most planning units covered by protected areas are located in the south-western side of Tasmania.

7.1.2 Vegetation data

1. North-eastern quarter of Tasmania

2. `cellStats(veg_data[["Heathland"]], "mean")`
3. `names(veg_data)[which.max(global(veg_data, "sum", na.rm = TRUE)[[1]])]`
4. Yes, they are the same.

7.2 Gap analysis

7.2.1 Feature abundance

1. `median(abundance_data$absolute_abundance_km2)`
2. `min(abundance_data$absolute_abundance_km2)`
3. `abundance_data$feature[which.min(abundance_data$absolute_abundance_km2)]`
4. `sum(abundance_data$absolute_abundance_km2)`
5. `sum(abundance_data$absolute_abundance_km2 > set_units(100, km^2))`

7.2.2 Feature representation by protected areas

1. `mean(repr_data$relative_held, na.rm = TRUE)`
2. `mean(repr_data$absolute_held_km2, na.rm = TRUE)`
3. `repr_data$feature[which.max(repr_data$relative_held)]`
4. `repr_data$feature[which.max(repr_data$absolute_held)]`
5. No, just because a vegetation class is widespread does not necessarily mean that it has the greatest overlap with protected areas. In fact, due to biases in the establishment of protected areas this can often be the case.
6. Yes, the largest protected areas tend to have the great representation (broadly speaking).
   `plot(abundance_data$absolute_abundance, repr_data$relative_held)`
7. `sum(repr_data$absolute_held == 0)`
8. `sum(repr_data$relative_held > 0.1, na.rm = TRUE)`
9. `sum(repr_data$relative_held > 0.3, na.rm = TRUE)`

7.3 Spatial prioritizations

7.3.1 Starting out simple

1. `sum(s1$solution_1)`
2. `mean(s1$solution_1)`
3. Yes, the planning units are generally spread out across most of the study area and they are not biased towards specific areas.
4. `all(feature_representation(p1, s1[, "solution_1"])$relative_held >= 0.3)`

7.3.2 Adding complexity

1. `sum(s2$cost * s2$solution_1)`,
   `sum(s3$cost * s3$solution_1)`,
   `sum(s4$cost * s4$solution_1)`

2. `sum(s2$solution_1)`,
   `sum(s3$solution_1)`,
   `sum(s4$solution_1)`
3. No, just because a solution a solution has more planning units does not mean that it will cost less.
4. This is because the planning units covered by existing protected areas have a non-zero cost and locking in these planning units introduces inefficiencies into the solution. This is very common in real-world conservation prioritizations because existing protected areas are often in places that do little to benefit biodiversity [Fuller et al., 2010].
5. This is because some of the planning units that are highly degraded – based on just the planning unit costs and vegetation data – provide cost-efficient opportunities for meeting the targets and excluding them from the reserve selection process means that other more costly planning units are needed to meet the targets.
6. `sum(s4$cost * s4$solution_1) - sum(s4$cost * s4$locked_in)`
7. We get an error message stating the the problem is infeasible because there is no valid solution—even if we selected all the planning units the study area we would still not meet the targets.

7.3.3 Penalizing fragmentation

1. The cost of the fourth solution is `sum(s4$solution_1 * s4$cost)` and the cost of the fifth solution is `sum(s5$solution_1 * s5$cost)`. The fifth solution (`s5`) costs more than the fourth solution (`s4`) because we have added penalties to the conservation planning problem to indicate that we are willing to accept a slightly more costly solution if it means that we can reduce fragmentation.
2. The solution is now nearly identical to the fourth solution (`s4`) and so has nearly the same cost. This penalty value is too low and is not useful because it does not reduce the fragmentation in our solution.
3. The solution now contains a lot of extra planning units that are not needed to meet our targets. In fact, nearly every planning unit in the study is now selected. This penalty value is too high and is not useful.

7.3.4 Budget limited prioritizations

1. `names(veg_data)[which.min(wts)]`
2. `sum(s6$cost * s6$solution_1)`,
   `sum(s7$cost * s7$solution_1)`
3. No, the sixth (`s6`) and seventh (`s7`) solutions both share many of the same selected planning units and there does not appear to be an obvious difference in the spatial location of the planning units which they do not share.
4. Yes. Both solutions contain adequately represent these features:
   `r6$feature[r6$relative_held > 0.3 & r7$relative_held > 0.3]`.
   The sixth (`s6`) adequately represents these features too:
   `r6$feature[r6$relative_held > 0.3 & !r7$relative_held > 0.3]`.
   The seventh (`s7`) adequately represents these features too:
   `r7$feature[r7$relative_held > 0.3 & !r6$relative_held > 0.3]`

7.4 Importance

7.4.1 Quantifying irreplaceability

1. There are 3 planning units with much higher irreplaceability values than the other planning units. These are found in the north-west, north-east, and south-east parts of Tasmania.

2. There are fewer planning units with high irreplaceability values, because the targets are lower, there are less irreplaceable planning units. In other words, there are more possible combinations of planning units available for meeting the targets.

Chapter 8

Acknowledgements

Many thanks to Icons8 for providing the icons used in this manual and to Yihui Xie for developing the bookdown R package that underpins this manual. We also thank Garrett Grolemund and Hadley Wickham for creating one of the Rstudio screenshots used in this manual that was originally a part of their *R for Data Science* book.

Chapter 9

Session information

```
# print session information
sessionInfo()
```

```
## R version 4.2.3 (2023-03-15)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 20.04.6 LTS
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.9.0
## LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.9.0
##
## locale:
##  [1] LC_CTYPE=C.UTF-8       LC_NUMERIC=C           LC_TIME=C.UTF-8
##  [4] LC_COLLATE=C.UTF-8     LC_MONETARY=C.UTF-8    LC_MESSAGES=C.UTF-8
##  [7] LC_PAPER=C.UTF-8       LC_NAME=C              LC_ADDRESS=C
## [10] LC_TELEPHONE=C         LC_MEASUREMENT=C.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] scales_1.2.1     units_0.8-1      mapview_2.11.0   highs_0.1-6
##  [5] sf_1.0-12        terra_1.7-23     lubridate_1.9.2  forcats_1.0.0
##  [9] stringr_1.5.0    dplyr_1.1.1      purrr_1.0.1      readr_2.1.4
## [13] tidyr_1.3.0      tibble_3.2.1     ggplot2_3.4.2    tidyverse_2.0.0
## [17] prioritizr_8.0.1
##
## loaded via a namespace (and not attached):
```

```
##  [1] Rcpp_1.0.10         ape_5.7-1           lattice_0.20-45
##  [4] png_0.1-8           class_7.3-21        assertthat_0.2.1
##  [7] digest_0.6.31       utf8_1.2.3          R6_2.5.1
## [10] backports_1.4.1     stats4_4.2.3        evaluate_0.20
## [13] e1071_1.7-13        pillar_1.9.0        rlang_1.1.0
## [16] rstudioapi_0.14     raster_3.6-20       Matrix_1.5-3
## [19] checkmate_2.1.0     rmarkdown_2.21      exactextractr_0.9.1
## [22] webshot_0.5.4       htmlwidgets_1.6.2   munsell_0.5.0
## [25] proxy_0.4-27        compiler_4.2.3      xfun_0.38
## [28] pkgconfig_2.0.3     base64enc_0.1-3     htmltools_0.5.5
## [31] tidyselect_1.2.0    bookdown_0.33.2     codetools_0.2-19
## [34] fansi_1.0.4         tzdb_0.3.0          withr_2.5.0
## [37] grid_4.2.3          satellite_1.0.4     nlme_3.1-162
## [40] gtable_0.3.3        lifecycle_1.0.3     DBI_1.1.3
## [43] magrittr_2.0.3      KernSmooth_2.23-20  cli_3.6.1
## [46] stringi_1.7.12      leaflet_2.1.2       sp_1.6-0
## [49] generics_0.1.3      vctrs_0.6.1         tools_4.2.3
## [52] leafem_0.2.0        glue_1.6.2          hms_1.1.3
## [55] crosstalk_1.2.0     parallel_4.2.3      fastmap_1.1.1
## [58] yaml_2.3.7          timechange_0.2.0    colorspace_2.1-0
## [61] classInt_0.4-9      knitr_1.42
```

Bibliography

Jeff A Ardron, Hugh P Possingham, and Carissa J Klein. *Marxan Good Practices Handbook* . Pacific Marine Analysis and Research Association, Victoria, BC, Canada, (2nd version) edition, 2010.

Maria Beger, Simon Linke, Matt Watts, Eddie Game, Eric Treml, Ian Ball, and Hugh P Possingham. Incorporating asymmetric connectivity into spatial decision making for conservation. *Conservation Letters*, 3(5):359–368, 2010.

Hawthorne L. Beyer, Yann Dujardin, Matthew E. Watts, and Hugh P. Possingham. Solving conservation planning problems with integer linear programming. *Ecological Modelling*, 328:14–22, 2016.

Stuart H.M. Butchart, Martin Clarke, Robert J. Smith, Rachel E. Sykes, Jörn P.W. Scharlemann, Mike Harfoot, Graeme M. Buchanan, Ariadne Angulo, Andrew Balmford, Bastian Bertzky, Thomas M. Brooks, Kent E. Carpenter, Mia T. Comeros-Raynal, John Cornell, G. Francesco Ficetola, Lincoln D.C. Fishpool, Richard A. Fuller, Jonas Geldmann, Heather Harwell, Craig Hilton-Taylor, Michael Hoffmann, Ackbar Joolia, Lucas Joppa, Naomi Kingston, Ian May, Amy Milam, Beth Polidoro, Gina Ralph, Nadia Richman, Carlo Rondinini, Daniel B. Segan, Benjamin Skolnik, Mark D. Spalding, Simon N. Stuart, Andy Symes, Joseph Taylor, Piero Visconti, James E.M. Watson, Louisa Wood, and Neil D. Burgess. Shortfalls and solutions for meeting national and global conservation area targets. *Conservation Letters*, 8(5):329–337, 2015.

Mar Cabeza and Atte Moilanen. Replacement cost: A practical measure of site value for cost-effective reserve planning. *Biological Conservation*, 132(3):336–342, October 2006. doi: 10.1016/j.biocon.2006.04.025. URL https://doi.org/10.1016/j.biocon.2006.04.025.

Simon Ferrier, Robert L. Pressey, and Thomas W. Barrett. A new predictor of the irreplaceability of areas for achieving a conservation goal, its application to real-world planning, and a research agenda for further refinement. *Biological Conservation*, 93(3):303–325, May 2000. doi: 10.1016/s0006-3207(99)00149-4. URL https://doi.org/10.1016/s0006-3207(99)00149-4.

Martin Friedrichs, Virgilio Hermoso, Vanessa Bremerich, and Simone D. Langhans. Evaluation of habitat protection under the european natura 2000 conservation network – the example for germany. *PLOS ONE*, 13(12):e0208264, 2018.

Richard A Fuller, Eve McDonald-Madden, Kerrie A Wilson, Josie Carwardine, Hedley S Grantham, James EM Watson, Carissa J Klein, David C Green, and Hugh P Possingham. Replacing underperforming protected areas achieves better conservation outcomes. *Nature*, 466(7304):365, 2010.

Jeffrey O. Hanson, Jaimie Vincent, Richard Schuster, Lenore Fahrig, Angela Brennan, Amanda E. Martin, Josie S. Hughes, Richard Pither, and Joseph R. Bennett. A comparison of approaches for including connectivity in systematic conservation planning. *Journal of Applied Ecology*, 59(10):2507–2519, 2022.

Martin Jung, Andy Arnell, Xavier de Lamo, Shaenandhoa García-Rangel, Matthew Lewis, Jennifer Mark, Cory Merow, Lera Miles, Ian Ondo, Samuel Pironon, Corinna Ravilious, Malin Rivers, Dmitry Schepaschenko, Oliver Tallowin, Arnout van Soesbergen, Rafaël Govaerts, Bradley L. Boyle, Brian J. Enquist, Xiao Feng, Rachael Gallagher, Brian Maitner, Shai Meiri, Mark Mulligan, Gali Ofer, Uri Roll, Jeffrey O. Hanson, Walter Jetz, Moreno Di Marco, Jennifer McGowan, D. Scott Rinnan, Jeffrey D. Sachs, Myroslava Lesiv, Vanessa M. Adams, Samuel C. Andrew, Joseph R. Burger, Lee Hannah, Pablo A. Marquet, James K. McCarthy, Naia Morueta-Holme, Erica A. Newman, Daniel S. Park, Patrick R. Roehrdanz, Jens-Christian Svenning, Cyrille Violle, Jan J. Wieringa, Graham Wynne, Steffen Fritz, Bernardo B. N. Strassburg, Michael Obersteiner, Valerie Kapos, Neil Burgess, Guido Schmidt-Traub, and Piero Visconti. Areas of global importance for conserving terrestrial biodiversity, carbon and water. *Nature Ecology and Evolution*, 5 (11):1499–1509, 2021.

C. R. Margules and R. L. Pressey. Systematic conservation planning. *Nature*, 405(6783): 243–253, 2000.

Atte Moilanen, Aldina M.A Franco, Regan I Early, Richard Fox, Brendan Wintle, and Chris D Thomas. Prioritizing multiple-use landscapes for conservation: methods for large multi-species planning problems. *Proceedings of the Royal Society B: Biological Sciences*, 272(1575):1885–1891, August 2005. doi: 10.1098/rspb.2005.3164. URL https://doi.org/10.1098/rspb.2005.3164.

R L Pressey. Applications of irreplaceability analysis to planning and management problems. *Parks*, 9(1):42–51, 1999.

Ana S. Rodrigues, J. Orestes Cerdeira, and Kevin J. Gaston. Flexibility, efficiency, and accountability: adapting reserve selection algorithms to more complex conservation problems. *Ecography*, 23(5):565–574, 2008.

Ana S. L. Rodrigues, H. Resit Akçakaya, Sandy J. Andelman, Mohamed I. Bakarr, Luigi Boitani, Thomas M. Brooks, Janice S. Chanson, Lincoln D. C. Fishpool, Gustavo A. B. Da Fonseca, Kevin J. Gaston, Michael Hoffmann, Pablo A. Marquet, John D. Pilgrim, Robert L. Pressey, Jan Schipper, Wes Sechrest, Simon N. Stuart, Les G. Underhill, Robert W. Waller, Matthew E. J. Watts, and Xie Yan. Global gap analysis: priority regions for expanding the global protected-area network. *BioScience*, 54(12):1092–1100, 2004.

Chris Taylor, Natasha Cadenhead, David B. Lindenmayer, and Brendan A. Wintle. Improving the design of a conservation reserve for a critically endangered species. *PLOS ONE*, 12(1):e0169629, 2017.