Lab Tutorial: Making Conservation Planning Decisions in Antarctic

Wen Wen/Mike Bode/Kerrie Wilson

2023-04-26

Outline

- Introduction ~ 5 min
- Data ~ 15 min
- Gap Analysis ~ 15 min
- Prioritization ~ 20 min
- Conclusion ~ 5 min

Learning Objectives

- To understand the core principles of conservation planning decisions.
- To provide the knowledge base to conduct conservation planning analyses.

Data Use Restriction

THIS LAB TUTORIAL DATA IS INTENDED ONLY FOR TRAINING PURPOSES AND NOT FOR FURTHER SCIENTIFIC PUBLICATION OR ANY LEGAL CLAIMS. THE DATA HEREIN HAS BEEN OBTAINED FROM SOURCES BELIEVED TO BE RELIABLE, BUT ITS ACCURACY AND COMPLETENESS, AND THE OPINIONS BASED THEREON, ARE NOT GUARANTEED.

1. Introduction

In this tutorial, you are playing the role of a conservation planner, for which you have been tasked by the Antarctic Scientific Committee to evaluate the existing system of protected areas in the Australian Antarctic Territory (AAT), which are in Antarctic Conservation Biogeographic Regions (ACBR's) 7 and 16. In the analysis process, you will be using the prioritizr R package as a decision support tool for spatial conservation prioritization. Your task is to find new conservation areas that will protect the species of interest, while minimizing the overall cost.

1.1 RMarkdown (.Rmd) & Tutorial Data

The data for this tutorial has been provided in this google drive link. After downloading this LabTutorial.zip, extract the files into a folder and titled "LabTutorial" folder.

1.2 Setting up your computer

You will need the latest stable version of the R program installed (i.e., version 4.2.2) and RStudio program. Please ensure you have administrative rights while installing these programs. After successfully installing the R program and RStudio, you will also need to install some of the following R packages (section 1.3). Open the RStudio application and then click Menu 'File' and Open Project… \LabTutorial\ConsPlan_Antarctic.Rproj (Index.Rmd will be loaded automatically on the panel source)

1.3 R packages

```
install.packages(
  c("sf", "sp", "rgeos", "rgdal", "raster", "units", "prioritizr",
    "tidyverse", "terra", "mapview", "assertthat", "data.table", "gridExtra",
    "Rsymphony"),
  type = "binary", repos="https://cran.rstudio.com")
```

1.4 Load packages

```
library(prioritizr)
```

```
## Warning: package 'prioritizr' was built under R version 4.2.3
```

```
library(sf)
```

```
## Warning: package 'sf' was built under R version 4.2.3

## Linking to GEOS 3.9.3, GDAL 3.5.2, PROJ 8.2.1; sf_use_s2() is TRUE
```

```
library(sp)
```

```
## Warning: package 'sp' was built under R version 4.2.3
```

```
library(raster)
```

```
## Warning: package 'raster' was built under R version 4.2.3
```

```
library(rgeos)
```

```
## Warning: package 'rgeos' was built under R version 4.2.3

## rgeos version: 0.6-2, (SVN revision 693)
##  GEOS runtime version: 3.9.3-CAPI-1.14.3
##  Please note that rgeos will be retired during 2023,
## plan transition to sf functions using GEOS at your earliest convenience.
##  GEOS using OverlayNG
##  Linking to sp version: 1.6-0
##  Polygon checking: TRUE
```

```
library(assertthat)
```

```
## Warning: package 'assertthat' was built under R version 4.2.3
```

```
library(units)
```

```
## Warning: package 'units' was built under R version 4.2.3

## udunits database from C:/Users/wwen/AppData/Local/R/win-library/4.2/units/share/udunits/udunits2.xml
```

```
library(data.table)
```

```
##
## Attaching package: 'data.table'

## The following object is masked from 'package:raster':
##
##     shift
```

```
library(gridExtra)
```

```
## Warning: package 'gridExtra' was built under R version 4.2.3
```

```
library(terra)
```

```
## Warning: package 'terra' was built under R version 4.2.3

## terra 1.7.18

##
## Attaching package: 'terra'

## The following object is masked from 'package:data.table':
##
##     shift

## The following object is masked from 'package:assertthat':
##
##     noNA
```

```
library(mapview)
```

```
## Warning: package 'mapview' was built under R version 4.2.3
```

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.2.3

## Warning: package 'ggplot2' was built under R version 4.2.3

## Warning: package 'dplyr' was built under R version 4.2.3

## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.1     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::between()     masks data.table::between()
## x dplyr::combine()     masks gridExtra::combine()
## x tidyr::extract()     masks terra::extract(), raster::extract()
## x dplyr::filter()      masks stats::filter()
## x dplyr::first()       masks data.table::first()
## x tibble::has_name()   masks assertthat::has_name()
## x lubridate::hour()    masks data.table::hour()
## x lubridate::isoweek() masks data.table::isoweek()
## x dplyr::lag()         masks stats::lag()
## x dplyr::last()        masks data.table::last()
## x lubridate::mday()    masks data.table::mday()
## x lubridate::minute()  masks data.table::minute()
## x lubridate::month()   masks data.table::month()
## x lubridate::quarter() masks data.table::quarter()
## x lubridate::second()  masks data.table::second()
## x dplyr::select()      masks raster::select()
## x dplyr::symdiff()     masks rgeos::symdiff()
## x purrr::transpose()   masks data.table::transpose()
## x lubridate::wday()    masks data.table::wday()
## x lubridate::week()    masks data.table::week()
## x lubridate::yday()    masks data.table::yday()
## x lubridate::year()    masks data.table::year()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```
library(Rsymphony)
```

1.5 Check your current working directory

```
getwd()
```

```
## [1] "C:/LabTutorial/ConsPlan_Antarctic"
```

2. Data

2.1 Import planning unit layer & coastline

```
planning_unit <- st_read("pu_layer_antarctic.shp")
```

```
## Reading layer `pu_layer_antarctic' from data source
##   `C:\LabTutorial\ConsPlan_Antarctic\pu_layer_antarctic.shp'
##   using driver `ESRI Shapefile'
## Simple feature collection with 6806 features and 6 fields
## Geometry type: POLYGON
## Dimension:     XY
## Bounding box:  xmin: 1633365 ymin: 406417.2 xmax: 2463365 ymax: 1226417
## Projected CRS: WGS_1984_Stereographic_South_Pole
```

```
coastline <- st_read("coastline.shp")
```

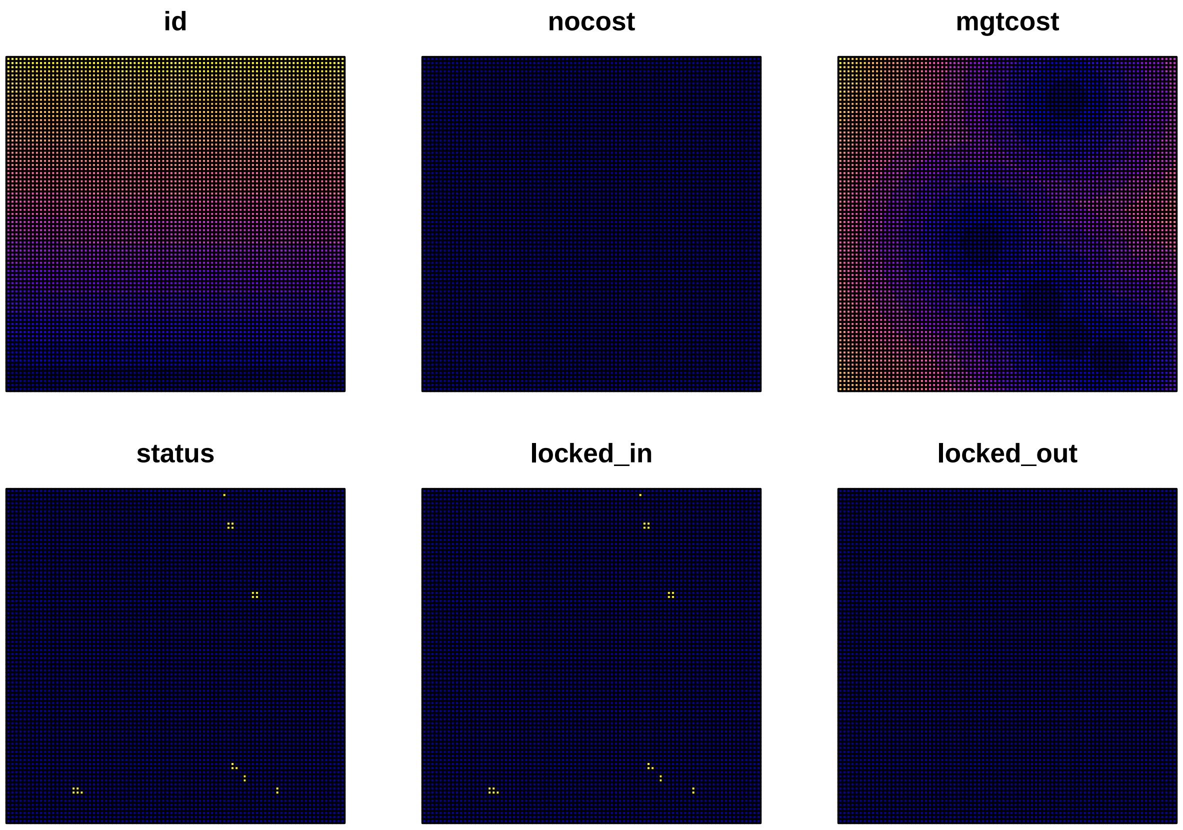
```
## Reading layer `coastline' from data source
##   `C:\LabTutorial\ConsPlan_Antarctic\coastline.shp' using driver `ESRI Shapefile'
## Simple feature collection with 1338 features and 2 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -5791904 ymin: -5791904 xmax: 5791904 ymax: 5791904
## Projected CRS: WGS 84 / Antarctic Polar Stereographic
```

2.2 Format column in planning unit data (TRUE or FALSE)

```
planning_unit$locked_in <- as.logical(planning_unit$locked_in)
planning_unit$locked_out <- as.logical(planning_unit$locked_out)
```

2.3 Plot the planning unit data

```
plot(planning_unit)
```



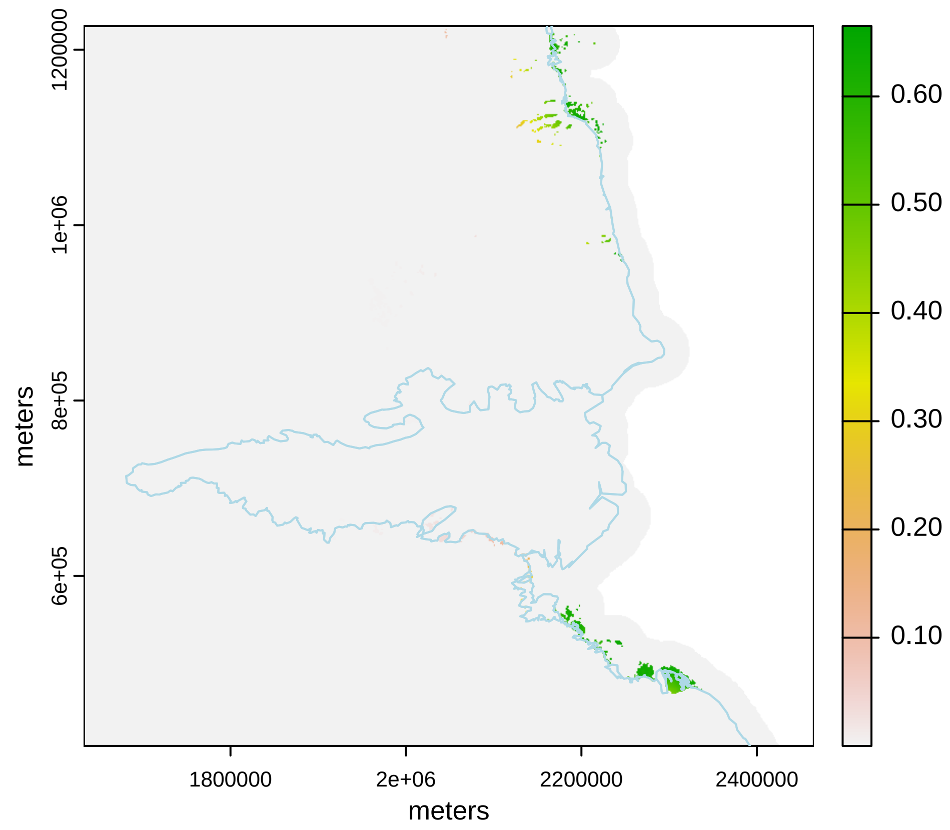
2.4 Conservation Target Data

2.4.1 South Polar Skua (*Stercorarius maccormicki*)

```
skua <- terra::rast("skua2_raster.tif")
print(skua)
```

```
## class       : SpatRaster
## dimensions  : 821, 832, 1  (nrow, ncol, nlyr)
## resolution  : 1000, 1000  (x, y)
## extent      : 1632858, 2464858, 406074.1, 1227074  (xmin, xmax, ymin, ymax)
## coord. ref. : WGS_1984_Stereographic_South_Pole
## source      : skua2_raster.tif
## name        : skua2_raster
```

```
plot(skua, xlab = "meters", ylab = "meters")
plot(st_geometry(coastline), add = TRUE, border = 'lightblue')
```

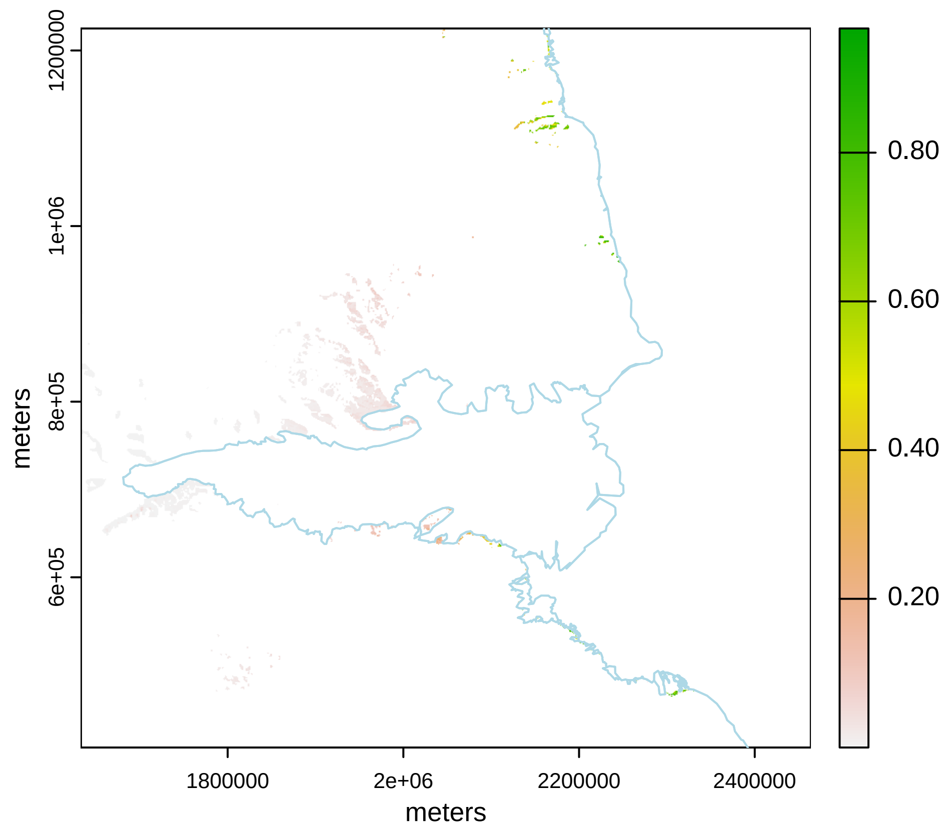


2.4.2 Moss (*Bryum pseudotriquetrum*)

```
bryum <- terra::rast("bryum2_raster.tif")
print(bryum)
```

```
## class       : SpatRaster
## dimensions  : 819, 831, 1  (nrow, ncol, nlyr)
## resolution  : 1000, 1000  (x, y)
## extent      : 1632858, 2463858, 406074.1, 1225074  (xmin, xmax, ymin, ymax)
## coord. ref. : WGS_1984_Stereographic_South_Pole
## source      : bryum2_raster.tif
## name        : bryum2_raster
```

```
plot(bryum, xlab = "meters", ylab = "meters")
plot(st_geometry(coastline), add = TRUE, border = 'lightblue')
```

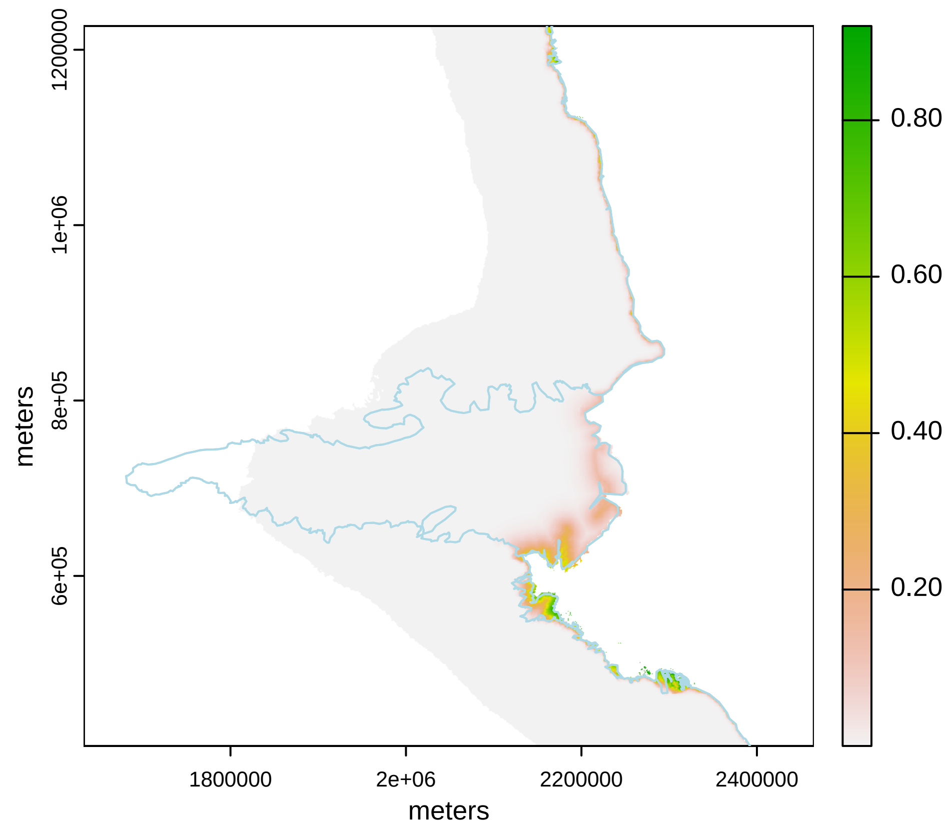


2.4.3 Lichen (*Usnea antarctica*)

```r
usnea <- terra::rast("usnea_raster.tif")
print(usnea)
```

```
## class       : SpatRaster
## dimensions  : 821, 832, 1  (nrow, ncol, nlyr)
## resolution  : 1000, 1000  (x, y)
## extent      : 1632858, 2464858, 406074.1, 1227074  (xmin, xmax, ymin, ymax)
## coord. ref. : WGS_1984_Stereographic_South_Pole
## source      : usnea_raster.tif
## name        : usnea_raster
```

```r
plot(usnea, xlab = "meters", ylab = "meters")
plot(st_geometry(coastline), add = TRUE, border = 'lightblue')
```

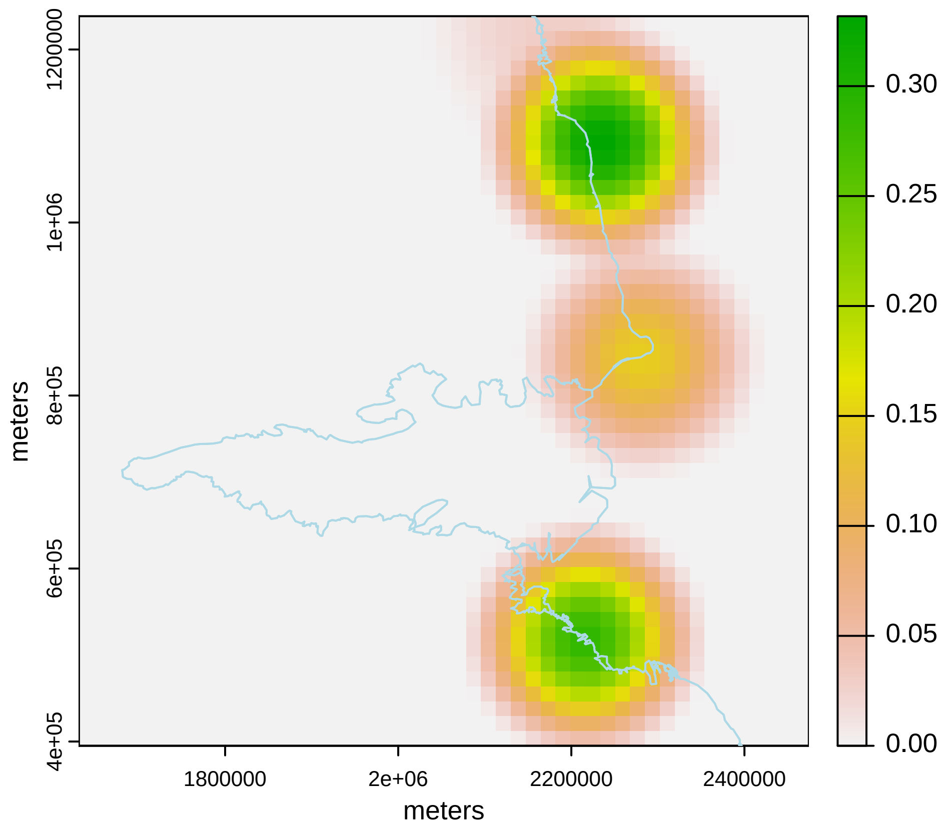


2.4.4 Emperor Penguin (*Aptenodytes forsteri*)

```
emperor <- terra::rast("emperor_raster.tif")
print(emperor)
```

```
## class       : SpatRaster
## dimensions  : 49, 49, 1  (nrow, ncol, nlyr)
## resolution  : 17212.84, 17212.84  (x, y)
## extent      : 1630967, 2474396, 395078.9, 1238508  (xmin, xmax, ymin, ymax)
## coord. ref. : WGS 84 / Antarctic Polar Stereographic (EPSG:3031)
## source      : emperor_raster.tif
## name        : emperor_raster
```

```
plot(emperor, xlab = "meters", ylab = "meters")
plot(st_geometry(coastline), add = TRUE, border = 'lightblue')
```



2.4.4 Resample species data

```r
emperor_resample <- terra::resample(emperor, skua, method="bilinear")
print(emperor_resample)
```

```
## class       : SpatRaster
## dimensions  : 821, 832, 1  (nrow, ncol, nlyr)
## resolution  : 1000, 1000  (x, y)
## extent      : 1632858, 2464858, 406074.1, 1227074  (xmin, xmax, ymin, ymax)
## coord. ref. : WGS 84 / Antarctic Polar Stereographic (EPSG:3031)
## source(s)   : memory
## name        : emperor_raster
## min value   :      0.0000000
## max value   :      0.3315811
```

```r
bryum_resample <- terra::resample(bryum, skua, method="bilinear")
print(bryum_resample)
```

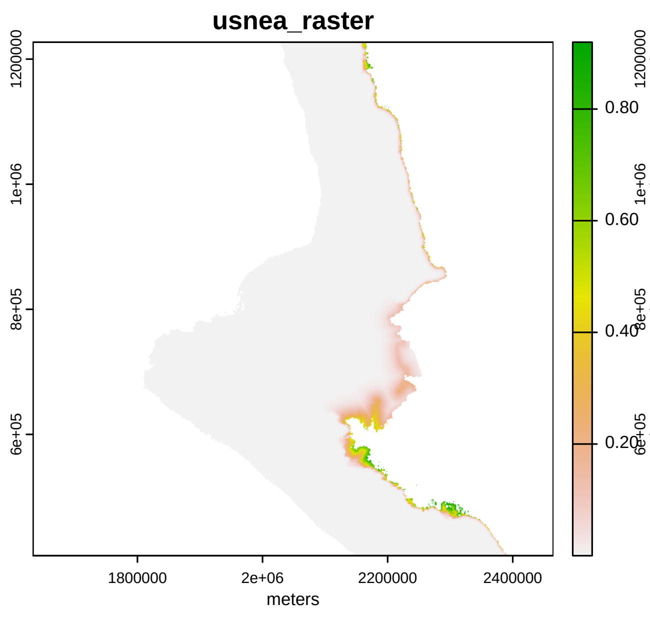
```
## class       : SpatRaster
## dimensions  : 821, 832, 1  (nrow, ncol, nlyr)
## resolution  : 1000, 1000  (x, y)
## extent      : 1632858, 2464858, 406074.1, 1227074  (xmin, xmax, ymin, ymax)
## coord. ref. : WGS_1984_Stereographic_South_Pole
## source(s)   : memory
## name        : bryum2_raster
## min value   :  2.254863e-13
## max value   :  9.670199e-01
```

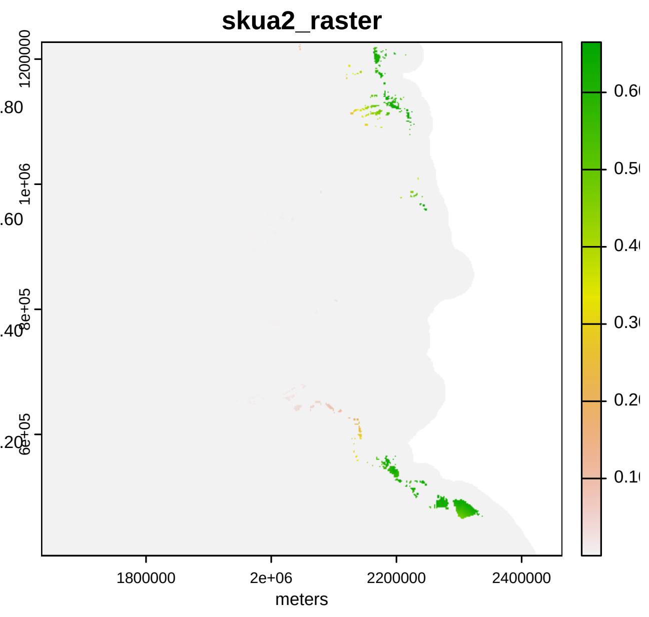
2.4.5 Combine all species data into single conservation target layer

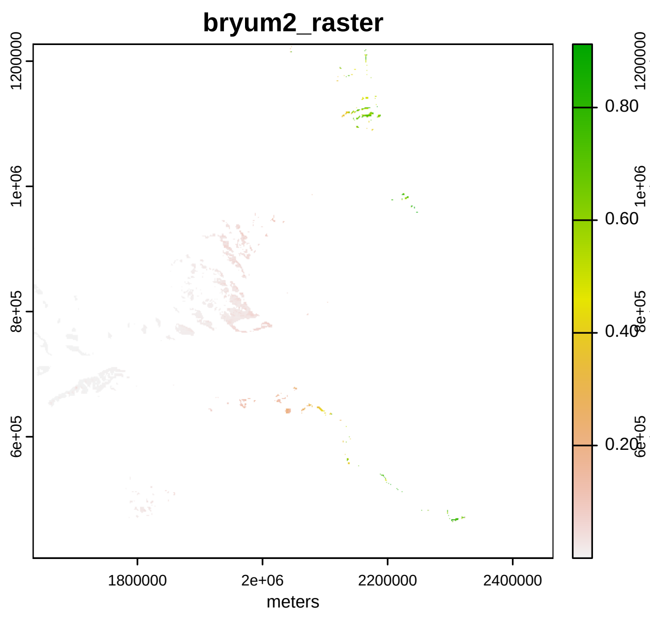
```
species <- c(usnea, skua, bryum_resample, emperor_resample)
print(species)
```

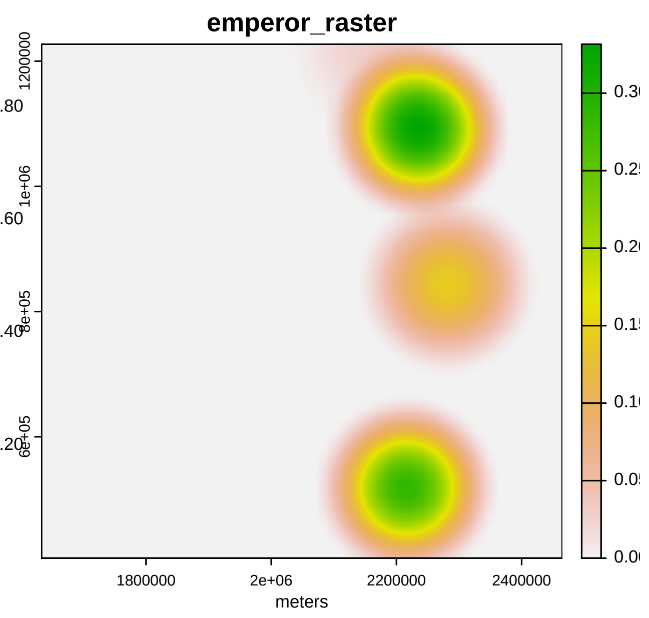
```
## class       : SpatRaster
## dimensions  : 821, 832, 4  (nrow, ncol, nlyr)
## resolution  : 1000, 1000  (x, y)
## extent      : 1632858, 2464858, 406074.1, 1227074  (xmin, xmax, ymin, ymax)
## coord. ref. : WGS_1984_Stereographic_South_Pole
## sources     : usnea_raster.tif
##               skua2_raster.tif
##               memory
##               memory
## names       : usnea_raster, skua2_raster, bryum2_raster, emperor_raster
## min values  :           ? ,           ? ,  2.254863e-13,      0.0000000
## max values  :           ? ,           ? ,  9.670199e-01,      0.3315811
```

```
plot(species, xlab = "meters", ylab = "meters")
```


usnea_raster
1200000
1e+06
8e+05
6e+05
1800000
2e+06
2200000
2400000
meters
0.80
0.60
0.40
0.20


skua2_raster
1200000
1e+06
8e+05
6e+05
1800000
2e+06
2200000
2400000
meters
0.6
0.5
0.4
0.3
0.2
0.1


bryum2_raster
1200000
1e+06
8e+05
6e+05
1800000
2e+06
2200000
2400000
meters
0.80
0.60
0.40
0.20


emperor_raster
1200000
1e+06
8e+05
6e+05
1800000
2e+06
2200000
2400000
meters
0.3
0.2
0.2
0.1
0.1
0.0
0.0

3. Feature representation

We will try to identify how well species targets are represented by existing ASPAs. Let's create prioritizr problem with only the data.

3.1 Null problem

```
p0 <- problem(planning_unit, species, cost_column = "nocost") %>%
      add_rsymphony_solver()
summary(p0)
```

```
## A conservation problem (<ConservationProblem>)
## +@data
## |+@features:    "usnea_raster", "skua2_raster", "bryum2_raster", "emperor_raster" , ... (4 total)
## |\@planning units:
## | +@data:       <sfdata.frame> (6806 total)
## | +@costs:      binary values ("0" and "1")
## | +@extent:     1633364.7876, 406417.1677, 2463364.7876, 1226417.1677 (xmin, ymin, xmax, ymax)
## | \@CRS:        WGS_1984_Stereographic_South_Pole
## \@formulation
##  +@objective:   none specified
##  +@penalties:   none specified
##  +@targets:     none specified
##  +@constraints: none specified
##  +@decisions:   binary decision
##  +@portfolio:   shuffle portfolio
##  |+@              `number_solutions` = 1
##  |+@              `threads` = 1
##  |\@              `remove_duplicates` = TRUE
##  \@solver:      rsymphony solver
##   +@              `gap` = 0.1
##   +@              `time_limit` = 2147483647
##   +@              `first_feasible` = FALSE
##   \@              `verbose` = TRUE
```

We will create a column in planning unit layer with binary values (zeros or ones) and this will indicate whether a planning unit is protected or not.

3.2 Evaluate representation data

```
planning_unit$aspa_status <- as.numeric(planning_unit$locked_in)
repr_data <- eval_feature_representation_summary(p0, planning_unit[, "aspa_status"] )
print(repr_data)
```

```
## # A tibble: 4 x 5
##   summary feature         total_amount absolute_held relative_held
##   <chr>   <chr>                  <dbl>         <dbl>         <dbl>
## 1 overall usnea_raster           3070.         245.         0.0798
## 2 overall skua2_raster           1665.         365.         0.219
## 3 overall bryum2_raster           656.          51.7        0.0788
## 4 overall emperor_raster        18434.         266.         0.0144
```

We are going to add new column with the areas represented in km2 and then print the representative data.

3.3 Convert the unit area into square km

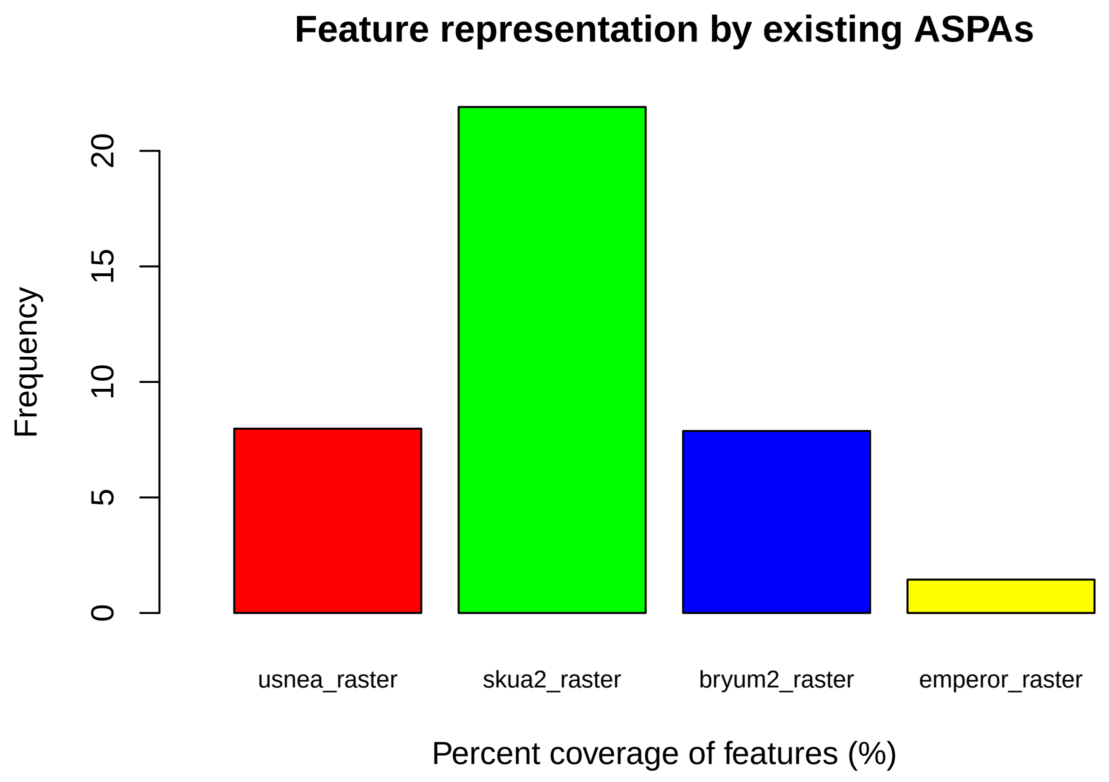
```
repr_data$absolute_held_km2 <-
    (repr_data$absolute_held * prod(res(species))) %>%
 set_units(m^2) %>%
 set_units(km^2)

print(repr_data)
```

```
## # A tibble: 4 x 6
##   summary feature     total_amount absolute_held relative_held absolute_held_km2
##   <chr>   <chr>              <dbl>         <dbl>         <dbl>            [km^2]
## 1 overall usnea_rast~        3070.         245.         0.0798             245.
## 2 overall skua2_rast~        1665.         365.         0.219              365.
## 3 overall bryum2_ras~         656.          51.7        0.0788              51.7
## 4 overall emperor_ra~       18434.         266.         0.0144             266.
```

Let's plot a bar chart of feature representation by existing ASPAs.

3.4 Bar chart of feature representation

```
barplot(repr_data$relative_held * 100,
     main = "Feature representation by existing ASPAs",
     xlim = c(0,5),
     col = c("red", "green", "blue", "yellow"),
     names.arg = repr_data$feature,
     cex.names = 0.75,
     xlab = "Percent coverage of features (%)",
     ylab = "Frequency"
)
```



4. Prioritization

We will consider two prioritization scenarios in this exercise as shown in below table.

Table 1: **Prioritization Scenarios**

	Relative Target	Lock In (Existing ASPA)	Cost
Scenario I	10%	No	No Cost/Equal
Scenario II	30%	Yes	Management Cost

4.1 Prioritization 1st Scenario (Equal Cost + No Lock In + 10% Target)

Here we will start to create a problem for prioritization and print it.

4.1.1 Create Problem

```
p1 <- problem(planning_unit, species, cost_column = "nocost") %>%
      add_min_set_objective() %>%
      add_relative_targets(0.10) %>%
      add_binary_decisions() %>%
      add_rsymphony_solver()
summary(p1)
```

```
## A conservation problem (<ConservationProblem>)
## +@data
## |+@features:     "usnea_raster", "skua2_raster", "bryum2_raster", "emperor_raster" , ... (4 total)
## |\@planning units:
## | +@data:        <sfdata.frame> (6806 total)
## | +@costs:       binary values ("0" and "1")
## | +@extent:      1633364.7876, 406417.1677, 2463364.7876, 1226417.1677 (xmin, ymin, xmax, ymax)
## | \@CRS:         WGS_1984_Stereographic_South_Pole
## \@formulation
##  +@objective:    minimum set objective
##  +@penalties:    none specified
##  +@targets:      relative targets (between 0.1 and 0.1)
##  +@constraints:  none specified
##  +@decisions:    binary decision
##  +@portfolio:    shuffle portfolio
##  |+@              `number_solutions` = 1
##  |+@              `threads` = 1
##  |\@              `remove_duplicates` = TRUE
##  \@solver:       rsymphony solver
##   +@              `gap` = 0.1
##   +@              `time_limit` = 2147483647
##   +@              `first_feasible` = FALSE
##   \@              `verbose` = TRUE
```

Let's solve the problem based on the first scenario and then print the result.

4.1.2 Solve Problem

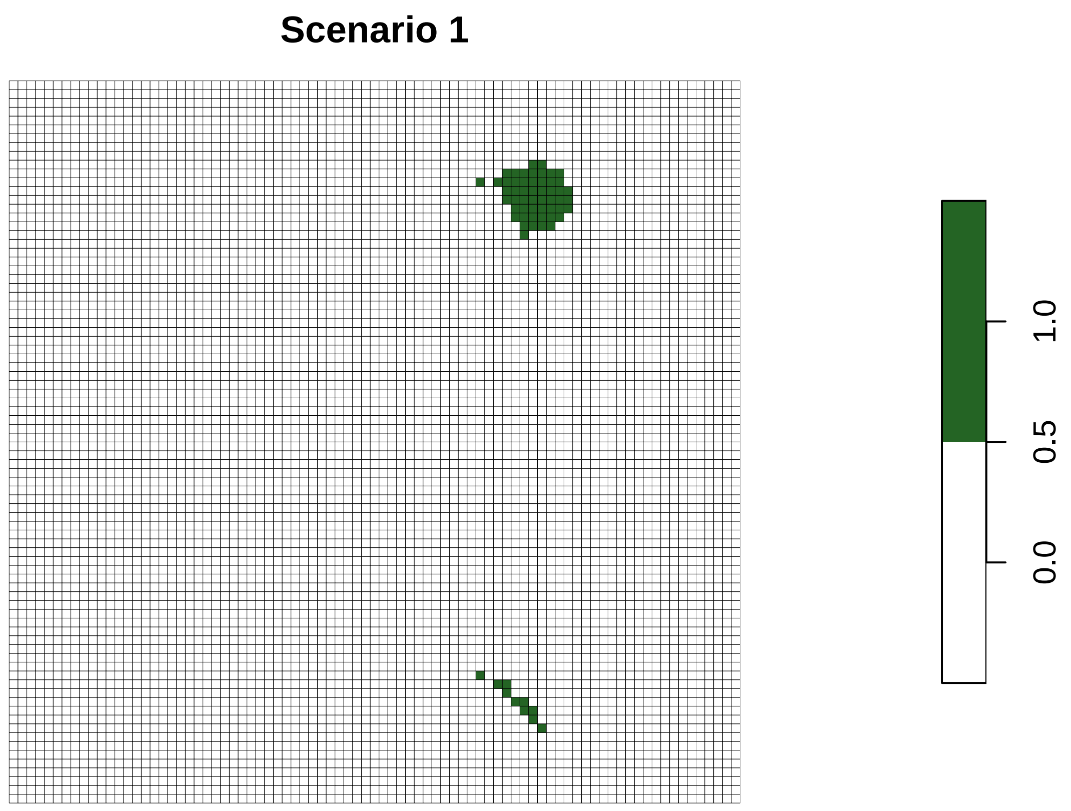
```
s1 <- solve(p1)
print(s1)
```

```
## Simple feature collection with 6806 features and 8 fields
## Geometry type: POLYGON
## Dimension:     XY
## Bounding box:  xmin: 1633365 ymin: 406417.2 xmax: 2463365 ymax: 1226417
## Projected CRS: WGS_1984_Stereographic_South_Pole
## # A tibble: 6,806 x 9
##        id nocost mgtcost status locked_in locked_out aspa_status solution_1
##     <dbl>  <dbl>   <dbl>  <dbl> <lgl>     <lgl>            <dbl>      <dbl>
##  1      0      1  500851      0 FALSE     FALSE                0          0
##  2      1      1  493959      0 FALSE     FALSE                0          0
##  3      2      1  487171      0 FALSE     FALSE                0          0
##  4      3      1  480494      0 FALSE     FALSE                0          0
##  5      4      1  473932      0 FALSE     FALSE                0          0
##  6      5      1  467489      0 FALSE     FALSE                0          0
##  7      6       1  461170       0 FALSE      FALSE                     0           0
##  8      7       1  454981       0 FALSE      FALSE                     0           0
##  9      8       1  448926       0 FALSE      FALSE                     0           0
## 10      9       1  443010       0 FALSE      FALSE                     0           0
## # i 6,796 more rows
## # i 1 more variable: geometry <POLYGON [m]>
```

After solving the problem you can visualize the selected planning units using the following markdown code.

4.1.3 Plot Solution Scenario 1

```
plot(st_as_sf(s1["solution_1"]),  main = "Scenario 1", pal = c("white", "#246424"), lwd = 0.01)
```



We will calculate the feature representation statistics based on the prioritization analysis for the first scenarios.

4.1.4 Evaluate Target

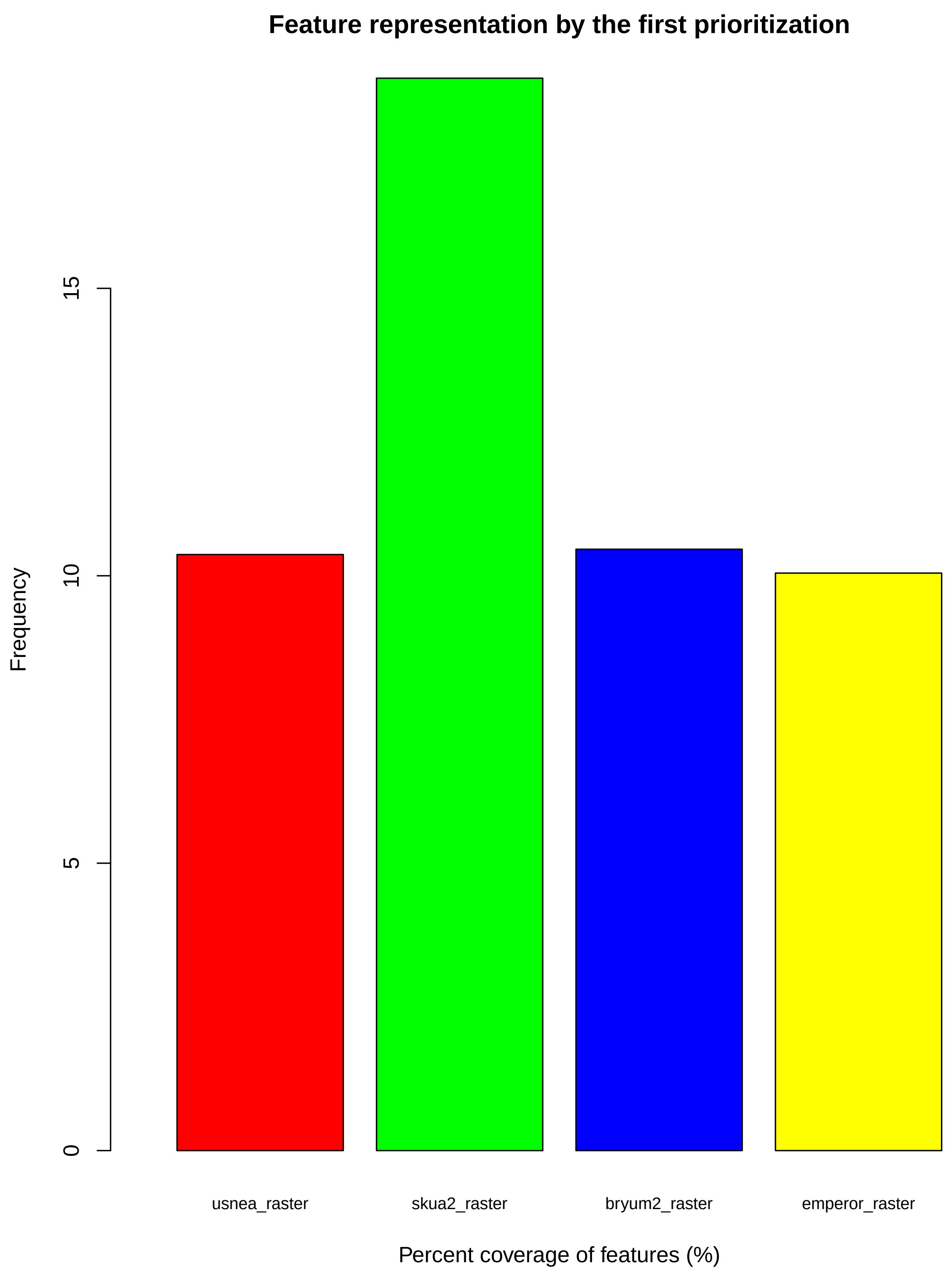
```
tc_s1 <- eval_target_coverage_summary(p1, s1[, "solution_1"])
print(tc_s1)
```

```
## # A tibble: 4 x 9
##   feature    met   total_amount absolute_target absolute_held absolute_shortfall
##   <chr>      <lgl>        <dbl>           <dbl>         <dbl>              <dbl>
## 1 usnea_ras~ TRUE         3070.            307.          318.                  0
## 2 skua2_ras~ TRUE         1665.            166.          311.                  0
## 3 bryum2_ra~ TRUE          656.             65.6          68.7                 0
## 4 emperor_r~ TRUE        18434.           1843.         1852.                  0
## # i 3 more variables: relative_target <dbl>, relative_held <dbl>,
## #   relative_shortfall <dbl>
```

As you can see on the summary table all species have met the minimum requirements that have been specified in the problem. Next, let's visualize the histogram for this first scenario so we can see the feature representation by this first prioritization.

4.1.5 Create Barplot

```
barplot(
  tc_s1$relative_held * 100,
  main = "Feature representation by the first prioritization",
  xlim = c(0, 5),
  col = c("red","green", "blue", "yellow"),
  names.arg = tc_s1$feature,
  cex.names = 0.75,
  xlab = "Percent coverage of features (%)",
  ylab = "Frequency"
)
```



4.2 Prioritization 2nd Scenario (Mgt Cost + Lock In + 30% Target)

First we will define and print the prioritization problem.

4.2.1 Create Problem

```
p2 <- problem(planning_unit, species, cost_column = "mgtcost") %>%
      add_min_set_objective() %>%
      add_relative_targets(0.30) %>%
      add_locked_in_constraints("locked_in") %>%
      add_binary_decisions() %>%
      add_rsymphony_solver()
summary(p2)
```

```
## A conservation problem (<ConservationProblem>)
## +@data
## |+@features:     "usnea_raster", "skua2_raster", "bryum2_raster", "emperor_raster" , ... (4 total)
## |\@planning units:
## | +@data:        <sfdata.frame> (6806 total)
## | +@costs:       continuous values (between 2689 and 560391)
## | +@extent:      1633364.7876, 406417.1677, 2463364.7876, 1226417.1677 (xmin, ymin, xmax, ymax)
## | \@CRS:         WGS_1984_Stereographic_South_Pole
## \@formulation
##  +@objective:    minimum set objective
##  +@penalties:    none specified
##  +@targets:      relative targets (between 0.3 and 0.3)
##  +@constraints:
##  |\@             locked in constraints (21 planning units)
##  +@decisions:    binary decision
##  +@portfolio:    shuffle portfolio
##  |+@               `number_solutions` = 1
##  |+@               `threads` = 1
##  |\@               `remove_duplicates` = TRUE
##  \@solver:       rsymphony solver
##   +@               `gap` = 0.1
##   +@               `time_limit` = 2147483647
##   +@               `first_feasible` = FALSE
##   \@               `verbose` = TRUE
```

Let's solve the problem based on the second scenario and then print the result.

4.2.2 Solve Problem

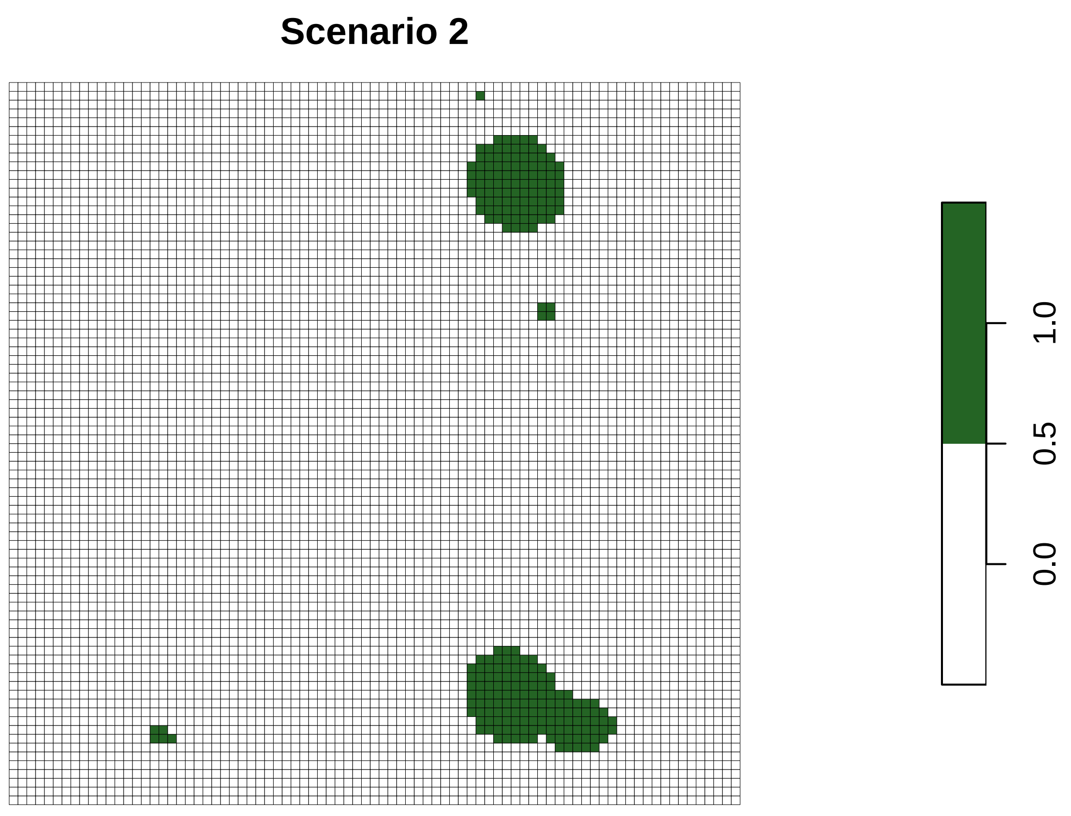
```
s2 <- solve(p2)
print(s2)
```

```
## Simple feature collection with 6806 features and 8 fields
## Geometry type: POLYGON
## Dimension:     XY
## Bounding box:  xmin: 1633365 ymin: 406417.2 xmax: 2463365 ymax: 1226417
## Projected CRS: WGS_1984_Stereographic_South_Pole
## # A tibble: 6,806 x 9
##       id nocost mgtcost status locked_in locked_out aspa_status solution_1
##    <dbl>  <dbl>   <dbl>  <dbl> <lgl>     <lgl>            <dbl>      <dbl>
##  1     0      1  500851      0 FALSE     FALSE                0          0
##  2     1      1  493959      0 FALSE     FALSE                0          0
##  3     2      1  487171      0 FALSE     FALSE                0          0
##  4     3      1  480494      0 FALSE     FALSE                0          0
##  5     4      1  473932      0 FALSE     FALSE                0          0
##  6     5      1  467489      0 FALSE     FALSE                0          0
##  7     6      1  461170      0 FALSE     FALSE                0          0
##  8     7      1  454981      0 FALSE     FALSE                0          0
##  9     8      1  448926      0 FALSE     FALSE                0          0
## 10     9      1  443010      0 FALSE     FALSE                0          0
## # i 6,796 more rows
## # i 1 more variable: geometry <POLYGON [m]>
```

Next, we will try to plot the solution on the map from this scenario to visualize the selected planning units and non-selected areas. Now, we can compare it and analyze the selection process based on two scenarios for the new ASPAs.

4.2.3 Plot Solution Scenario 2

```
plot(st_as_sf(s2["solution_1"]), main = "Scenario 2", pal = c("white", "#246424"), lwd = 0.01)
```



We will also calculate the feature representation based on the second prioritization and we can identify how each feature has met the requirements.

4.2.4 Evaluate Target

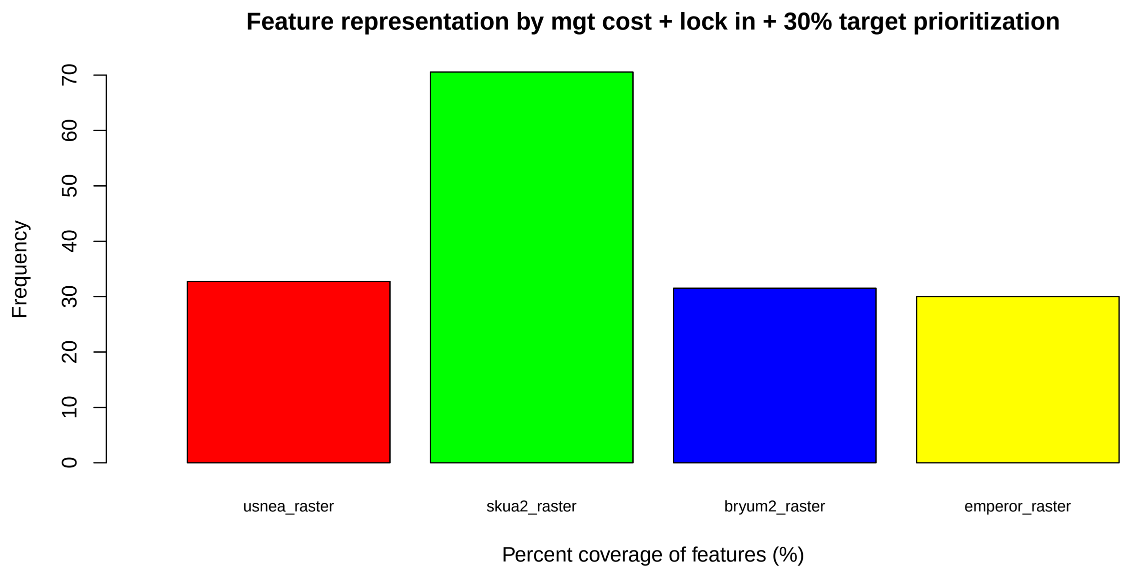
```r
tc_s2 <- eval_target_coverage_summary(p2, s2[, "solution_1"])
print(tc_s2)
```

```
## # A tibble: 4 x 9
##   feature    met   total_amount absolute_target absolute_held absolute_shortfall
##   <chr>      <lgl>        <dbl>           <dbl>         <dbl>              <dbl>
## 1 usnea_ras~ TRUE         3070.            921.         1006.                  0
## 2 skua2_ras~ TRUE         1665.            499.         1175.                  0
## 3 bryum2_ra~ TRUE          656.            197.          207.                  0
## 4 emperor_r~ TRUE        18434.           5530.         5531.                  0
## # i 3 more variables: relative_target <dbl>, relative_held <dbl>,
## #   relative_shortfall <dbl>
```

Let's plot the histogram for feature representation based on the second prioritization scenario. We will see that the feature targets are well represented in this second scenario.

4.2.5 Create Barplot

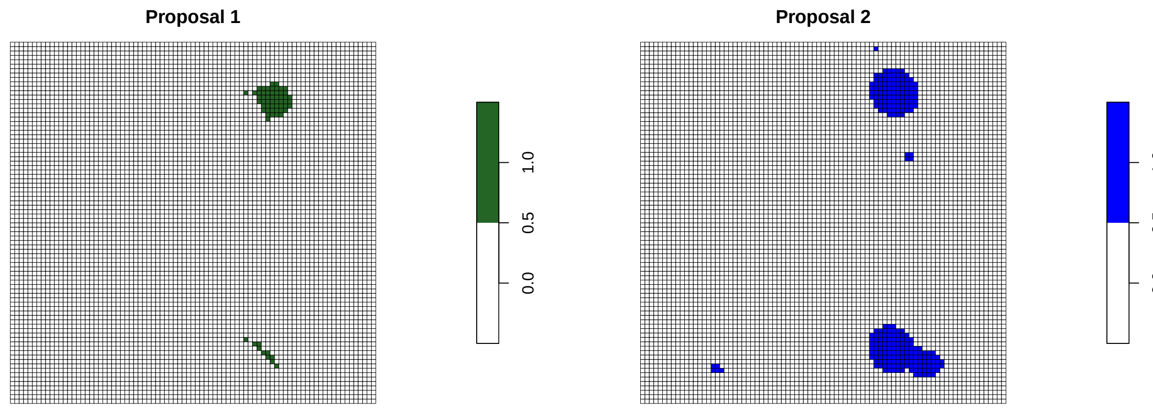
```r
barplot(
  tc_s2$relative_held * 100,
  main = "Feature representation by mgt cost + lock in + 30% target prioritization",
  xlim = c(0, 5),
  col = c("red","green", "blue", "yellow"),
  names.arg = tc_s2$feature,
  cex.names = 0.75,
  xlab = "Percent coverage of features (%)",
  ylab = "Frequency"
)
```



4.3 Protected Areas Proposal (Scenario 1 + Scenario 2)

Here we will combine two prioritization results based on the two scenarios.

```
par(mar = c(4, 4, .1, .1))
plot(st_as_sf(s1["solution_1"]), pal = c("white", "#246424"), main = "Proposal 1", lwd = 0.01)
plot(st_as_sf(s2["solution_1"]), pal = c("white", "blue"), main = "Proposal 2", lwd = 0.01)
```



5. Conclusion

Congratulation! You have completed the lab tutorial on conservation planning.

6. Acknowledgements

Many thanks to the prioritizr team for providing the library package in this tutorial and to Isabella Todd and Aleks Terauds who are willing to share the species distribution model datasets for this exercise. We also thank the Australian Antarctic Data Centre for the Antarctic Specially Protected Areas shapefile and penguin colony data and the Securing Antarctic Environmental Future (SAEF) program which facilitated the implementation of this training.