PRIORITIZR WORKSHOP MANUAL

Jeffrey O. Hanson

2019-09-30

Contents

Chapter 1

Welcome!

Here you will find the manual for the prioritizr module of the *Spatial Conservation Prioritization: Concepts, Methods and Application* workshop held at CIBIO-InBIO, Vairão, Portugal. **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. We cannot guarantee a reliable internet connection during the workshop, and so you may be unable to complete the workshop if you have not set up your computer beforehand.**

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. Phrased provocatively, 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. If you are having trouble with using the prioritizr R package, please consult the package website: https://prioritizr.net. It has plenty of examples and tutorials. Finally, please note that the first thing an instructor will ask you will (probably) be "what have you tried so far?". We can't help you if you haven't tried anything.

2.2 Setting up your computer

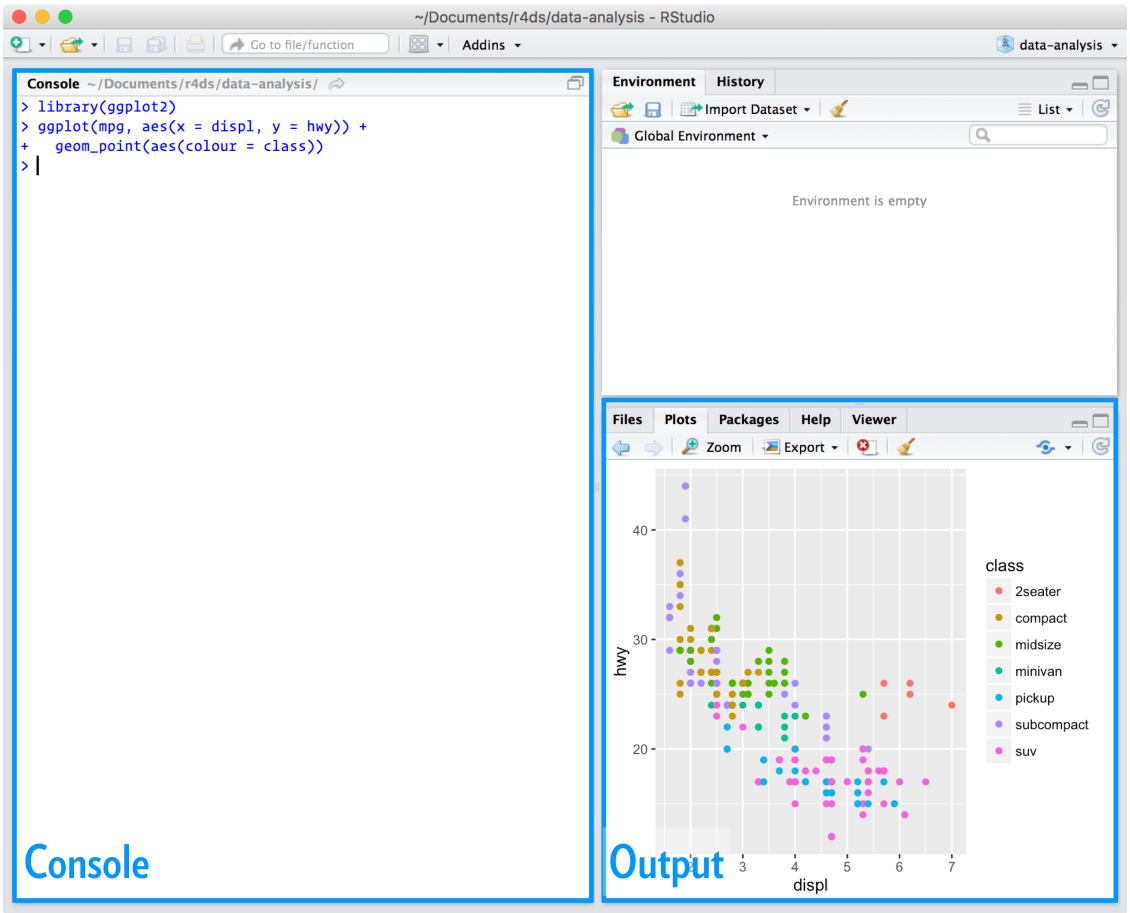
You will need to have both R and RStudio installed on your computer to complete this workshop. Although it is not imperative that you have the latest version of RStudio installed, **you will need the latest version of R installed (i.e. version 3.6.1)**. Please note that you meed need administrative permissions to complete install these programs. After installing them, you will also need to install various R packages too.

2.2.1 R

The R statistical computing environment can be downloaded from the Comprehensive R Archive Network (CRAN). You can download the latest version of R (version 3.6.1) 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, Mac OSX, 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 when you want 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:



You can type R code into the *Console* part of the interface and press the enter key to run the code.

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 ten thousand R packages available on CRAN. Each of these R packages (mostly) aim to serve a specific need, 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 (generally) not "can I use R to do …?" but "what R package can I use to …?". During this workshop, we will use various R packages. To install these R packages, please run enter the code below in the *Console* part of the RStudio interface and press enter. Please note that you will require an internet connection to install the packages and the installation process may take a while to complete.

```
install.packages(c("sf", "tidyverse", "sp", "rgeos", "rgdal", "raster",
                   "prioritizr", "prioritizrdata" "Rsymphony", "mapview"))
```

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

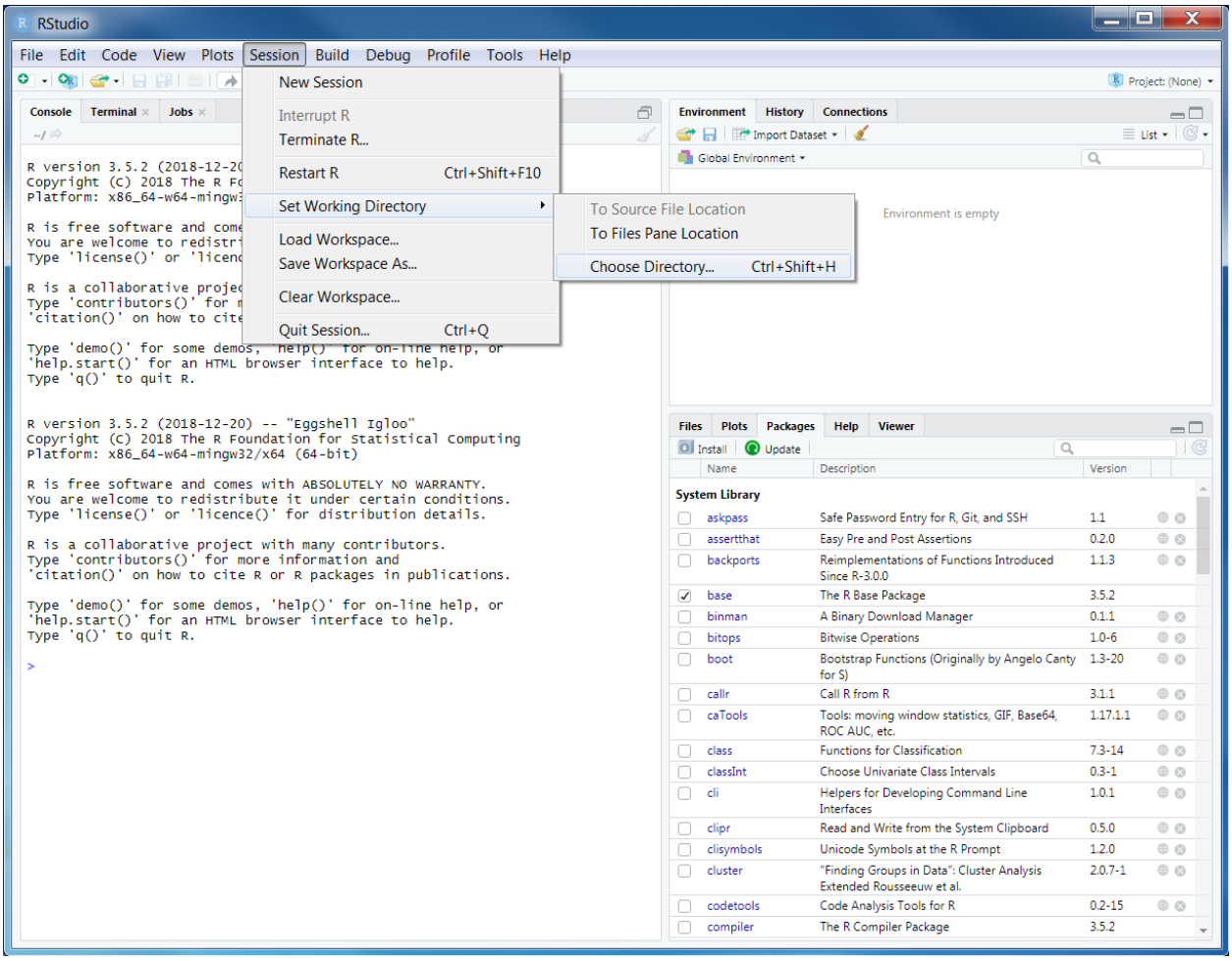
Data

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. After opening RStudio, enter the following R code to attach the R packages we will use in this workshop.

```
# load packages
library(tidyverse)
library(prioritizr)
library(rgdal)
library(raster)
library(mapview)
```

You should have already downloaded the data for the prioritizr module of this workshop. If you have not already done so, you can download it from here: https://github.com/prioritizr/cibio-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`.

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

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

3.2 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 was originally a subset of a larger spatial prioritization project performed under contract to Australia's Department of Environment and Water Resources. It contains vector-based planning

unit data (`pu.shp`)and the raster-based data describing the spatial distributions of 62 vegetation classes (`vegetation.tif`) in Tasmania, Australia. We can import the data into our R session using the following code.

```
# import planning unit data
pu_data <- readOGR("data/pu.shp")
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "/home/travis/build/prioritizr/cibio-workshop/data/pu.shp", layer: "pu"
## with 1130 features
## It has 5 fields
```

```
# import vegetation data
veg_data <- stack("data/vegetation.tif")
```

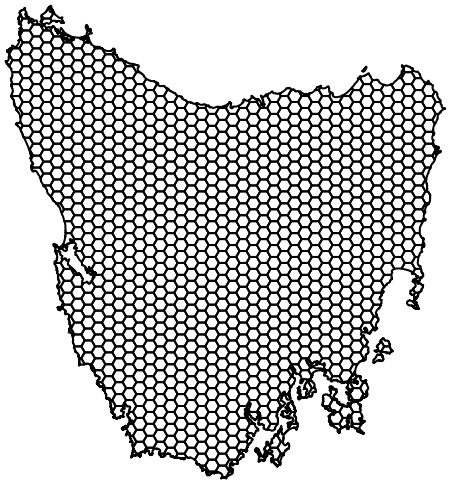
3.3 Planning unit data

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 attribute data contains 5 columns with contain the following information: * `id`: unique identifiers for each planning unit * `cost`: acquisition cost values for each planning unit * `status`: status information for each planning unit (only relevant with Marxan) * `locked_in`: binary values (i.e. one or zero) indicating if planning units are covered by protected areas or not. * `locked_out`: binary values (i.e. one or zero) indicating if planning units cannot be managed as a protected area because they contain too much anthropologically altered land.

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

```
## class       : SpatialPolygonsDataFrame
## features    : 1130
## extent      : 1080623, 1399989, -4840595, -4497092  (xmin, xmax, ymin, ymax)
## crs         : +proj=aea +lat_1=-18 +lat_2=-36 +lat_0=0 +lon_0=132 +x_0=0 +y_0=0 +ellp
## variables   : 5
## names       :   id,              cost, status, locked_in, locked_out
## min values  :    1, 0.192488262910798,      0,         0,          0
## max values  : 1130,  61.9272727272727,      2,         1,          0
```

```
# plot the planning unit data
plot(pu_data)
```



```
# plot an interactive map of the planning unit data
mapview(pu_data)
```

```
# print the structure of object
str(pu_data, max.level = 2)
```

```
## Formal class 'SpatialPolygonsDataFrame' [package "sp"] with 5 slots
##   ..@ data       :'data.frame':  1130 obs. of  5 variables:
##   ..@ polygons   :List of 1130
##   ..@ plotOrder  : int [1:1130] 217 973 506 645 705 975 253 271 704 889 ...
##   ..@ bbox       : num [1:2, 1:2] 1080623 -4840595 1399989 -4497092
##   .. ..- attr(*, "dimnames")=List of 2
##   ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot
```

```
# print the class of the object
class(pu_data)
```

```
## [1] "SpatialPolygonsDataFrame"
## attr(,"package")
## [1] "sp"
```

```
# print the slots of the object
slotNames(pu_data)
```

```
## [1] "data"        "polygons"    "plotOrder"   "bbox"        "proj4string"
```

```
# print the geometry for the 80th planning unit
pu_data@polygons[[80]]
```

```
## An object of class "Polygons"
## Slot "Polygons":
## [[1]]
## An object of class "Polygon"
## Slot "labpt":
## [1]  1289177 -4558185
##
## Slot "area":
## [1] 1060361
##
## Slot "hole":
## [1] FALSE
##
## Slot "ringDir":
## [1] 1
##
## Slot "coords":
##          [,1]     [,2]
##  [1,] 1288123 -4558431
##  [2,] 1287877 -4558005
##  [3,] 1288177 -4558019
##  [4,] 1288278 -4558054
##  [5,] 1288834 -4558038
##  [6,] 1289026 -4557929
##  [7,] 1289168 -4557928
##  [8,] 1289350 -4557790
##  [9,] 1289517 -4557744
## [10,] 1289618 -4557773
```

```
## [11,] 1289836 -4557965
## [12,] 1290000 -4557984
## [13,] 1290025 -4557987
## [14,] 1290144 -4558168
## [15,] 1290460 -4558431
## [16,] 1288123 -4558431
##
##
##
## Slot "plotOrder":
## [1] 1
##
## Slot "labpt":
## [1]  1289177 -4558185
##
## Slot "ID":
## [1] "79"
##
## Slot "area":
## [1] 1060361
```

```
# print the coordinate reference system
print(pu_data@proj4string)
```

```
## CRS arguments:
##  +proj=aea +lat_1=-18 +lat_2=-36 +lat_0=0 +lon_0=132 +x_0=0 +y_0=0
## +ellps=GRS80 +units=m +no_defs
```

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

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

```
# print the first six rows in the attribute data
head(pu_data@data)
```

```
##   id     cost status locked_in locked_out
## 0  1 60.24638      0         0          0
## 1  2 19.86301      0         0          0
## 2  3 59.68051      0         0          0
## 3  4 32.41614      0         0          0
## 4  5 26.17706      0         0          0
## 5  6 51.26218      0         0          0
```

```
# print the first six values in the cost column of the attribute data
head(pu_data$cost)
```

```
## [1] 60.24638 19.86301 59.68051 32.41614 26.17706 51.26218
```

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

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

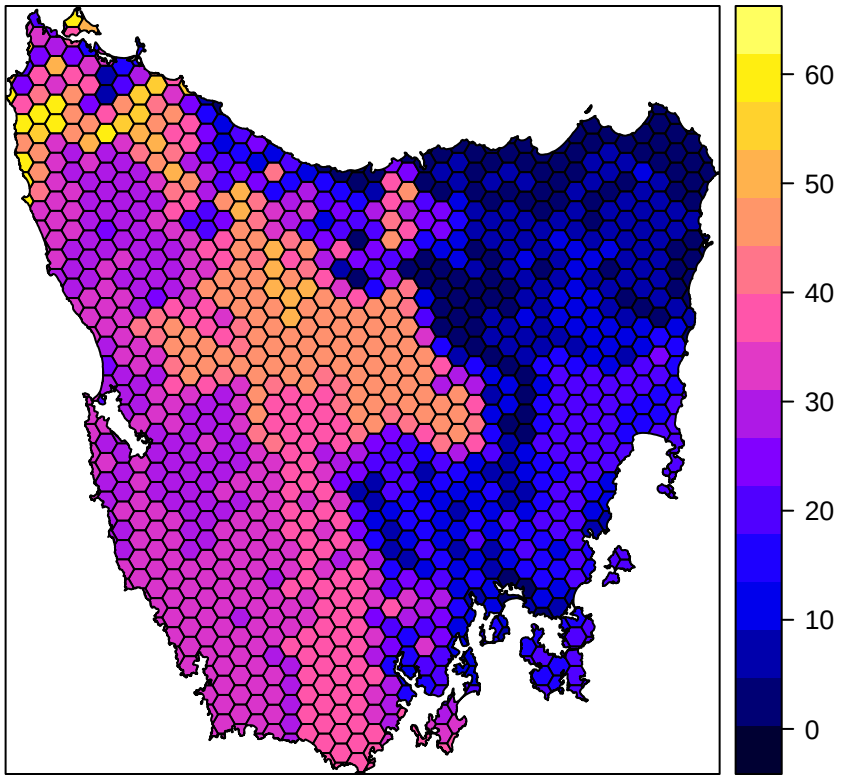
```
# print the smallest cost value
min(pu_data$cost)
```

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

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

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

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



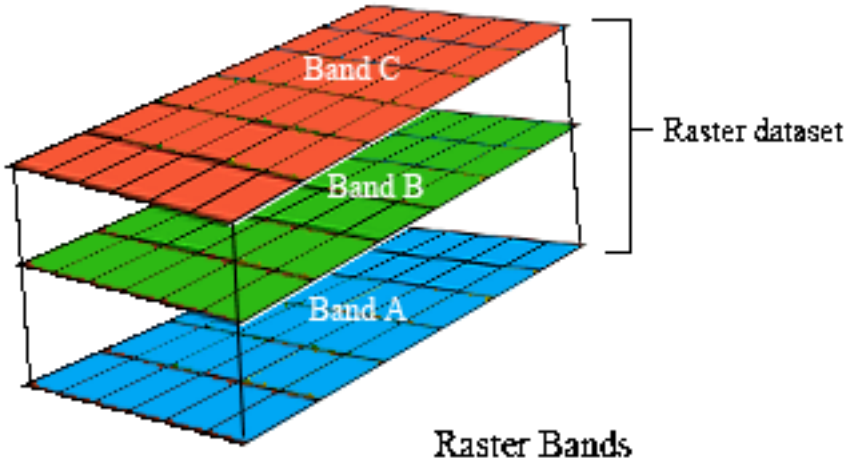

```
# plot an interactive map of the planning unit cost data
mapview(pu_data, zcol = "cost")
```

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 dominated by anthropologically altered land (hint: `sum(x)`)?
6. What is the proportion of planning units dominated by anthropologically altered land (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: make a map).
11. Is there a spatial pattern in where most planning units are covered by protected areas (hint: make a map of `locked_in`).

3.4 Vegetation data

The vegetation data describes the spatial distribution of 62 vegetation classes in the study area. This data is in a raster format and so the data are organized using a regular spatial grid with square grid cells. In our case, our 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 62 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 343 columns, 320 rows, and 109760 cells. Within each layer, each cell corresponds to a 1 by 1 km square. The values associated with each grid cell contain values (i.e. one or zero) indicating the presence or absence of a given vegetation class in the cell.

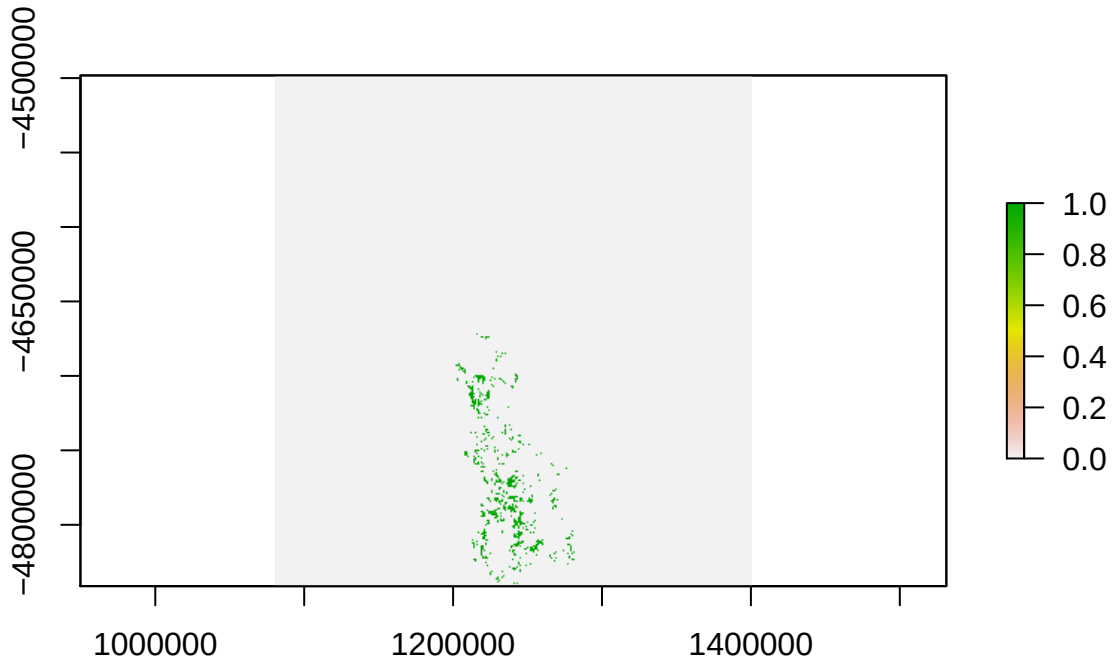


Let's explore the vegetation data.

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

```
## class      : RasterStack
## dimensions : 343, 320, 109760, 62  (nrow, ncol, ncell, nlayers)
## resolution : 1000, 1000  (x, y)
## extent     : 1080496, 1400496, -4841217, -4498217  (xmin, xmax, ymin, ymax)
## crs        : +proj=aea +lat_1=-18 +lat_2=-36 +lat_0=0 +lon_0=132 +x_0=0 +y_0=0 +ellps
## names      : vegetation.1, vegetation.2, vegetation.3, vegetation.4, vegetation.5, ve
## min values :            0,            0,            0,            0,            0,
## max values :            1,            1,            1,            1,            1,
```

```
# plot a map of the 36th vegetation class
plot(veg_data[[36]])
```




```
# plot an interactive map of the 36th vegetation class
mapview(veg_data[[36]])
```

```
# print number of rows in the data
nrow(veg_data)
```

```
## [1] 343
```

```
# print number of columns  in the data
ncol(veg_data)
```

```
## [1] 320
```

```
# print number of cells in the data
ncell(veg_data)
```

```
## [1] 109760
```

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

```
## [1] 62
```

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

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

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

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

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

```
## class      : Extent
## xmin       : 1080496
## xmax       : 1400496
## ymin       : -4841217
## ymax       : -4498217
```

```r
# print the coordinate reference system
print(veg_data@crs)
```

```
## CRS arguments:
##  +proj=aea +lat_1=-18 +lat_2=-36 +lat_0=0 +lon_0=132 +x_0=0 +y_0=0
## +ellps=GRS80 +units=m +no_defs
```

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

```
## class      : RasterLayer
## band       : 1  (of  62  bands)
## dimensions : 343, 320, 109760  (nrow, ncol, ncell)
## resolution : 1000, 1000  (x, y)
## extent     : 1080496, 1400496, -4841217, -4498217  (xmin, xmax, ymin, ymax)
## crs        : +proj=aea +lat_1=-18 +lat_2=-36 +lat_0=0 +lon_0=132 +x_0=0 +y_0=0 +ellps
## source     : /home/travis/build/prioritizr/cibio-workshop/data/vegetation.tif
```

```
## names       : vegetation.1
## values      : 0, 1  (min, max)
```

```
# print the value in the 800th cell in the first layer of the stack
print(veg_data[[1]][800])
```

```
##
## 0
```

```
# print the value of the cell located in the 30th row and the 60th column of
# the first layer
print(veg_data[[1]][30, 60])
```

```
##
## 0
```

```
# calculate the sum of all the cell values in the first layer
cellStats(veg_data[[1]], "sum")
```

```
## [1] 36
```

```
# calculate the maximum value of all the cell values in the first layer
cellStats(veg_data[[1]], "max")
```

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

```
# calculate the minimum value of all the cell values in the first layer
cellStats(veg_data[[1]], "min")
```

```
## [1] 0
```

```
# calculate the mean value of all the cell values in the first layer
cellStats(veg_data[[1]], "mean")
```

```
## [1] 0.0003279883
```

```
# calculate the maximum value in each layer
data.frame(max = cellStats(veg_data, "max"))
```

```
##    max
## 1    1
```

```
## 2     1
## 3     1
## 4     1
## 5     1
## 6     1
## 7     1
## 8     1
## 9     1
## 10    1
## 11    1
## 12    1
## 13    1
## 14    1
## 15    1
## 16    1
## 17    1
## 18    1
## 19    1
## 20    1
## 21    1
## 22    1
## 23    1
## 24    1
## 25    1
## 26    1
## 27    1
## 28    1
## 29    1
## 30    1
## 31    1
## 32    1
## 33    1
## 34    1
## 35    1
## 36    1
## 37    1
## 38    1
## 39    1
## 40    1
## 41    1
## 42    1
## 43    1
## 44    1
## 45    1
## 46    1
```

```
## 47   1
## 48   1
## 49   1
## 50   1
## 51   1
## 52   1
## 53   1
## 54   1
## 55   1
## 56   1
## 57   1
## 58   1
## 59   1
## 60   1
## 61   1
## 62   1
```

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

1. What part of the study area is the 51st vegetation class found in (hint: make a map)?
2. How many rows does the 23rd layer contain?
3. Is the third vegetation class present at the 400th cell?
4. How many cells contain the 56th vegetation class?
5. What proportion of cells contain the 12th vegetation class?
6. Which vegetation class is present in the greatest number of cells?
7. Make an new object by summing together all the layer values into a single grid (i.e. `sum_veg_data <- sum(veg_data)`) and make a map showing this data (i.e. `plot(sum_veg_data)`). What does it mean if cells contain zeros in this new object? What reasons could there be to explain why some cells in this new object contain zeros?

Chapter 4

Gap analysis

TODO.

Chapter 5

Spatial prioritizations

TODO.

Chapter 6

Irreplaceability

TODO.

Chapter 7

Acknowledgements

Many thanks to Icons8 for providing the icons used in this manual.