EcoCommons running species distribution models in R

Step 1, set up your workspace, download and clean occurrence data

Author details: EcoCommons Platform Contact details: comms@ecocommons.org.au This script is the product of EcoCommons platform.

please cite:

EcoCommons Australia 2022. EcoCommons Australia – the platform of choice for ecological and environmental modelling, EcoCommons, Griffith University, Nathan, Queensland. Available: https://data\T1\ textendashexplorer.app.ecocommons.org.au/ (Accessed: Date[e.g., January 19, 2022]). https://doi.org/10.47486/PL108

Date: March 2022

Script and data info: The script and data in this file draws on several sources including

1. The ATLAS of Living Australia (ALA), script modified from script written by Jenna Wraith, ALA Training & Outreach Coordinator

Stevenson M, Westgate M, Newman P (2022). *galah: Atlas of Living Australia (ALA) Data and Resources in R*. R package version 1.3.1, <URL: https://CRAN.R-project.org/package=galah>.

See for vingnettes on how to use the package: https://atlasoflivingaustralia.github.io/galah/

2. Global Biodiversity Information Facility (GBIF)

source of base code which has been modified here: https://www.r-bloggers.com/2021/03/downloading-and-cleaning-gbif-data-with-r/

Chamberlain S, Barve V, Mcglinn D, Oldoni D, Desmet P, Geffert L, Ram K (2022). *rgbif: Interface to the Global Biodiversity Information Facility API*. R package version 3.6.0, <URL: https://CRAN.R-project.org/package=rgbif>.

Setting up your workspace

First, if you are new to using R, we strongly suggest you visit this website and fo through that material before trying this: https://datacarpentry.org/R-ecology-lesson/

You should be able to run all this code in EcoCommons' coding cloud using either a jupyter notebook (R Kernal) or by importing this notebook into R studio as an R Markdown, i.e. https://rmarkdown.rstudio.com/docs/reference/convert_ipynb.html

Below is code to install and load the needed packages, and to set up your working directories and subfolders

Install packages if needed, check first to see if they are installed, if they are not run the required line of code for the needed package

install.packages("galah") install.packages("rgbif") install.packages("maps") install.packages("tidyr") install.packages("jpeg") install.packages("raster") install.packages("rgeos") install.packages("sp")

```
# load all required packages
# Use the install packages line above for any packages not already installed.
# install.packages("tidyr") # installs the tidyr package
# library(tidyr) #loads that package into the current session

library(galah)
```

```
## Warning: package 'galah' was built under R version 4.1.2
```

```
library(rgbif)
```

```
## Warning: package 'rgbif' was built under R version 4.1.2
```

```
library(maps)
library(tidyr)
```

```
## Warning: package 'tidyr' was built under R version 4.1.2
```

```
library(jpeg)
library(raster)
```

```
## Warning: package 'raster' was built under R version 4.1.2

## Loading required package: sp
```

```
library(rgeos)
```

```
## rgeos version: 0.5-9, (SVN revision 684)
##  GEOS runtime version: 3.8.1-CAPI-1.13.3
##  Please note that rgeos will be retired by the end of 2023,
## plan transition to sf functions using GEOS at your earliest convenience.
##  Linking to sp version: 1.4-6
##  Polygon checking: TRUE
```

```
library(sp)
library(knitr)
library(rmarkdown)
```

```
## Warning: package 'rmarkdown' was built under R version 4.1.2
```

```
#confirm which packages are loaded
(.packages())
```

```
##  [1] "rmarkdown" "knitr"     "rgeos"     "raster"    "sp"        "jpeg"
##  [7] "tidyr"     "maps"      "rgbif"     "galah"     "stats"     "graphics"
## [13] "grDevices" "utils"     "datasets"  "methods"   "base"
```

```r
# identify your working directory

getwd()
```

```
## [1] "/Users/s2992269/Documents/Use_cases/SDM_in_R/R_SDM"
```

```r
direct<- "/Users/s2992269/Documents/Use_cases"
folder <- "/SDM_in_R"

#this sets your working director for all subsequent chunks of code in your R Markdown script
knitr::opts_knit$set(root.dir = paste0(direct,folder))

# if you are not working R Markdown, simply use this instead
# setwd(paste0(direct,folder))
```

getwd() setwd("")

```r
# double check your working directory
getwd()
```

```
## [1] "/Users/s2992269/Documents/Use_cases/SDM_in_R"
```

create a variety of folders in your working directory, if statement ignores this if the folder exists already

```r
raw_data_folder <- paste0(getwd(),"/raw_data")
  if (!file.exists(raw_data_folder)) {
    dir.create(raw_data_folder)}

data_folder     <- paste0(getwd(),"/data")
  if (!file.exists(data_folder)) {
    dir.create(data_folder)}

results_folder  <- paste0(getwd(),"/results")
  if (!file.exists(results_folder)) {
    dir.create(results_folder)}

results_folder  <- paste0(getwd(),"/results1")
  if (!file.exists(results_folder)) {
    dir.create(results_folder)}

results_folder  <- paste0(getwd(),"/results2")
  if (!file.exists(results_folder)) {
    dir.create(results_folder)}

results_folder  <- paste0(getwd(),"/results3")
  if (!file.exists(results_folder)) {
    dir.create(results_folder)}
```

```
results_folder  <- paste0(getwd(),"/results4")
  if (!file.exists(results_folder)) {
    dir.create(results_folder)}

results_folder  <- paste0(getwd(),"/results_brt")
  if (!file.exists(results_folder)) {
    dir.create(results_folder)}

results_folder  <- paste0(getwd(),"/results_glm")
  if (!file.exists(results_folder)) {
    dir.create(results_folder)}

results_folder  <- paste0(getwd(),"/results_gam")
  if (!file.exists(results_folder)) {
    dir.create(results_folder)}

scripts_folder  <- paste0(getwd(),"/scripts")
    if (!file.exists(scripts_folder)) {
        dir.create(scripts_folder)}

raw_data_folder <- paste0(getwd(),"/predictors")
  if (!file.exists(raw_data_folder)) {
    dir.create(raw_data_folder)}

raw_data_folder <- paste0(getwd(),"/predictors_future")
  if (!file.exists(raw_data_folder)) {
    dir.create(raw_data_folder)}
```

```
# Set galah_config by adding your email
# galah_config(email = "your-email@email.com") # This email needs to be registered with ALA
# you can register here: https://auth.ala.org.au/userdetails/registration/createAccount

galah_config(email = "r.clemens@griffith.edu.au") # again put your email in here
```

```
## These configuration options will only be saved for this session.
##     Set ‘preserve = TRUE‘ to preserve them for future sessions.
```

```
# select an ATLAS
## These configuration options will only be saved for this session
## Set preserve = TRUE to preserve them for future sessions

show_all_atlases()
```

```
## # A tibble: 6 x 3
##   atlas     taxonomy_source taxonomy_info
##   <chr>     <chr>           <chr>
## 1 Australia ALA             https://bie.ala.org.au/
## 2 Austria   GBIF            https://www.gbif.org/dataset/d7dddbf4-2cf0-4f39-9b2~
## 3 Guatemala GBIF            https://www.gbif.org/dataset/d7dddbf4-2cf0-4f39-9b2~
## 4 Spain     GBIF            https://www.gbif.org/dataset/d7dddbf4-2cf0-4f39-9b2~
## 5 Sweden    GBIF            https://www.gbif.org/dataset/d7dddbf4-2cf0-4f39-9b2~
## 6 UK        NBN             https://www.nhm.ac.uk/our-science/data/uk-species.h~
```

```r
# we will be using the Australian ATLAS data

galah_config(atlas="Australia")
```

```
## These configuration options will only be saved for this session.
##     Set 'preserve = TRUE' to preserve them for future sessions.
```

```r
# show all the fields or columns of data within in the ALA

show_all_fields()
```

```
## # A tibble: 788 x 4
##    id                   description            type   link
##    <chr>                <chr>                  <chr>  <chr>
##  1 acceptedNameUsage    Accepted name          fields <NA>
##  2 acceptedNameUsageID  Accepted name          fields <NA>
##  3 accessRights         Access rights          fields <NA>
##  4 assertionUserId      Assertions by user     fields <NA>
##  5 assertions           Record issues          fields <NA>
##  6 associatedMedia      Associated Media       fields <NA>
##  7 associatedOccurrences Associated Occurrences fields <NA>
##  8 associatedReferences Associated References  fields <NA>
##  9 associatedSequences  Associated Sequences   fields <NA>
## 10 associatedTaxa       Associated Taxa        fields <NA>
## # ... with 778 more rows
```

```r
# find the kinds of data in each field replace the field name of interest in quotes

search_fields("australian states and territories")
```

```
## # A tibble: 2 x 4
##   id     description                                                type  link
##   <chr>  <chr>                                                      <chr> <chr>
## 1 cl2013 ASGS Australian States and Territories Australian Statisti~ laye~ http~
## 2 cl22   Australian States and Territories Australian States and Te~ laye~ http~
```

```r
search_fields("coordinateUncertaintyinMeters")
```

```
## # A tibble: 2 x 4
##   id                               description                     type  link
##   <chr>                            <chr>                           <chr> <chr>
## 1 coordinateUncertaintyInMeters    Coordinate uncertainty (in metr~ fiel~ <NA>
## 2 raw_coordinateUncertaintyInMeters Coordinate uncertainty (unproce~ fiel~ <NA>
```

```r
search_fields("occurrenceID")
```

```
## # A tibble: 1 x 4
##   id           description   type   link
##   <chr>        <chr>         <chr>  <chr>
## 1 occurrenceID Occurrence ID fields <NA>
```

```
# look at the field values

search_field_values("cl22")
```

```
## # A tibble: 11 x 2
##    field category
##    <chr> <chr>
##  1 cl22  New South Wales
##  2 cl22  Victoria
##  3 cl22  Queensland
##  4 cl22  South Australia
##  5 cl22  Western Australia
##  6 cl22  Northern Territory
##  7 cl22  Australian Capital Territory
##  8 cl22  Tasmania
##  9 cl22  Unknown1
## 10 cl22  Ashmore and Cartier Islands
## 11 cl22  Coral Sea Islands
```

```
# in this example we are looking at frogs
search_taxa("Amphibia")
```

```
## # A tibble: 1 x 10
##   search_term scientific_name taxon_concept_id   rank  match_type kingdom phylum
##   <chr>       <chr>           <chr>              <chr> <chr>      <chr>   <chr>
## 1 Amphibia    AMPHIBIA        urn:lsid:biodiver~ class exactMatch Animal~ Chord~
## # ... with 3 more variables: class <chr>, vernacular_name <chr>, issues <chr>
```

```
search_taxa(genus = "Limnodynastes")
```

```
## # A tibble: 1 x 13
##   search_term scientific_name scientific_name~ taxon_concept_id rank  match_type
##   <chr>       <chr>           <chr>            <chr>            <chr> <chr>
## 1 Limnodynas~ Limnodynastes   Fitzinger, 1843  urn:lsid:biodiv~ genus exactMatch
## # ... with 7 more variables: kingdom <chr>, phylum <chr>, class <chr>,
## #   order <chr>, family <chr>, genus <chr>, issues <chr>
```

```
search_taxa(species = "Limnodynastes peroni")
```

```
## # A tibble: 1 x 15
##   search_term scientific_name scientific_name~ taxon_concept_id rank  match_type
##   <chr>       <chr>           <chr>            <chr>            <chr> <chr>
## 1 Limnodynas~ Limnodynastes ~ (Duméril & Bibr~ urn:lsid:biodiv~ spec~ exactMatch
## # ... with 9 more variables: kingdom <chr>, phylum <chr>, class <chr>,
## #   order <chr>, family <chr>, genus <chr>, species <chr>,
## #   vernacular_name <chr>, issues <chr>
```

download some occurence data

the galah_call function starts a query, then galah_identify selects the taxa you are interested in, galah_filter selects records for specified columns and atlas_occurrences retrieves the specified occurrence records

```r
# look for records submitted by FrogID

search_field_values("datasetName")  # note you can see more rows, click lower right below
```

```
## Warning: This field has 8365 possible values. Only the first 20 will be returned.
## i Change ‘limit‘ to return more values.
```

```
## # A tibble: 20 x 2
##    field       category
##    <chr>       <chr>
##  1 datasetName EOD - eBird Observation Dataset
##  2 datasetName DPIE Data from Scientific Licences dataset
##  3 datasetName iNaturalist research-grade observations
##  4 datasetName Australian Bird & Bat Banding Scheme
##  5 datasetName iNaturalist observations
##  6 datasetName DPIE Default Sightings
##  7 datasetName Australia's Virtual Herbarium
##  8 datasetName Wild Count Fauna
##  9 datasetName Miscellaneous Fauna Data
## 10 datasetName Wildlife Rehab Database
## 11 datasetName FrogID
## 12 datasetName Land Assessment Branch
## 13 datasetName State Forests Biodata
## 14 datasetName Royal Botanic Gardens Herbarium Specimen Register
## 15 datasetName Herbarium (north)
## 16 datasetName Fauna Survey
## 17 datasetName AWC-NSW Parks Partnership Projects
## 18 datasetName DENR Weed Management Branch
## 19 datasetName GAP_EAST Vegetation Survey
## 20 datasetName Biodiversity Conservation (north)
```

```r
# first check the number of records your query will return, there are 67,000+ records in ALA

galah_call() %>%
  galah_identify("Limnodynastes peroni")%>%
  atlas_counts()
```

```
## # A tibble: 1 x 1
##   count
##   <int>
## 1 67248
```

```r
# Then filter those records so only those records from "FrogID" are returned for this species
# & remove records with a coordinate uncertainty greater than 100m, there are 36,000+ records from the
galah_call() %>%
  galah_identify("Limnodynastes peroni")%>%
  galah_filter(datasetName == "FrogID")%>%
  galah_filter(stateProvince == "Queensland")%>%
  galah_filter(coordinateUncertaintyInMeters < 100)%>%
  atlas_counts()
```

```
## # A tibble: 1 x 1
```

```
##   count
##   <int>
## 1  4372
```

```
# select the occurrence records, galah_select returns wanted columns
# We could also filter by year, but FrogID data is all pretty recent
# Often you will want to ensure you are not including really old records in your data
# i.e. galah_filter(year >= 2020)

LiPe <- galah_call() %>%
  galah_identify("Limnodynastes peroni")%>%
  galah_filter(datasetName == "FrogID")%>%
  galah_filter(coordinateUncertaintyInMeters < 100)%>%
  galah_filter(stateProvince == "Queensland")%>%
  galah_select("datasetName")%>%
  atlas_occurrences()
```

```
## This query will return 4372 records

##   |                                                                              |
```

```
# get familiar with data - this will return the column names or fields

head(LiPe)
```

```
## # A tibble: 6 x 8
##   datasetName decimalLatitude decimalLongitude eventDate          scientificName
##   <chr>                 <dbl>            <dbl> <chr>              <chr>
## 1 FrogID                -27.7             153. 2020-03-12T13:00:~ Limnodynastes~
## 2 FrogID                -16.9             146. 2019-06-28T14:00:~ Limnodynastes~
## 3 FrogID                -26.5             153. 2018-06-08T14:00:~ Limnodynastes~
## 4 FrogID                -27.4             153. 2020-06-13T14:00:~ Limnodynastes~
## 5 FrogID                -27.5             153. 2018-09-21T14:00:~ Limnodynastes~
## 6 FrogID                -27.5             153. 2019-09-22T14:00:~ Limnodynastes~
## # ... with 3 more variables: taxonConceptID <chr>, recordID <chr>,
## #   dataResourceName <chr>
```

```
# generate a summary of those data

names(LiPe)
```

```
## [1] "datasetName"      "decimalLatitude"  "decimalLongitude" "eventDate"
## [5] "scientificName"   "taxonConceptID"   "recordID"         "dataResourceName"
```

write.csv(LiPe, "raw_data/Limnodynastes peroni.csv")

```
summary(LiPe)
```

```
##  datasetName        decimalLatitude  decimalLongitude  eventDate
##  Length:4372        Min.   :-28.92   Min.   :145.3    Length:4372
##  Class :character   1st Qu.:-27.53   1st Qu.:152.7    Class :character
```

```
##  Mode  :character   Median :-26.73   Median :152.9    Mode  :character
##                     Mean   :-25.75   Mean   :152.0
##                     3rd Qu.:-26.11   3rd Qu.:153.0
##                     Max.   :-16.26   Max.   :153.5
##  scientificName     taxonConceptID       recordID         dataResourceName
##  Length:4372        Length:4372        Length:4372        Length:4372
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
```

drop columns not needed in analyses

Note you will often want to filter on additional fields not shown here

```
LiPe<-LiPe[,c(2:5)]
```

LiPe<-LiPe[,c("decimalLatitude","decimalLongitude","eventDate","scientificName")]

```
# look at the top rows of the new dataset

head(LiPe)
```

```
## # A tibble: 6 x 4
##   decimalLatitude decimalLongitude eventDate           scientificName
##             <dbl>            <dbl> <chr>               <chr>
## 1           -27.7             153. 2020-03-12T13:00:00Z Limnodynastes peronii
## 2           -16.9             146. 2019-06-28T14:00:00Z Limnodynastes peronii
## 3           -26.5             153. 2018-06-08T14:00:00Z Limnodynastes peronii
## 4           -27.4             153. 2020-06-13T14:00:00Z Limnodynastes peronii
## 5           -27.5             153. 2018-09-21T14:00:00Z Limnodynastes peronii
## 6           -27.5             153. 2019-09-22T14:00:00Z Limnodynastes peronii
```

```
# this is the earliest date, 2017 is fairly recent, and we want to make sure our predictor variables ar

min(LiPe$eventDate)
```

```
## [1] "2017-11-09T13:00:00Z"
```

```
#This is the most recent date, 2020
max(LiPe$eventDate)
```

```
## [1] "2020-11-08T13:00:00Z"
```

```
# save your filtered data to another folder, again you would often filter on other fields as well

write.csv(LiPe, "data/Limnodynastes peroni.csv")
```

double check the directory within the chunks is correct note the directory outside the code chunks might be different

```
getwd()
```

```
## [1] "/Users/s2992269/Documents/Use_cases/SDM_in_R"
```

```
# read in data and overwrite LiPe
LiPe <- read.csv(paste0(getwd(),"/data/Limnodynastes peroni.csv"))
```

```
# FrogID records are pretty clean, but often when you map data you will see odd locations
map("world", xlim = range(LiPe$decimalLongitude),
    ylim = range(LiPe$decimalLatitude))
points(LiPe[ , c("decimalLongitude", "decimalLatitude")], pch = ".")
```

When you look at these records mapped, you can see that most records are concentrated where people are near the bigger cities of eastern Australiathis kind of sampling bias is common in Australian occurrence data. It takes so much more effort to sample in remote locations There are a few things you can do to reduce the problem of bias, you can break up your study area into areas near cities, and more remote areas (we will not do that here) You can reduce the number of records close to one another. This reduces spatial autocorrelation, and is a good step for these kinds of records before we show how to do spatial - thinning (reducing how close records are together) we need a base raster layer

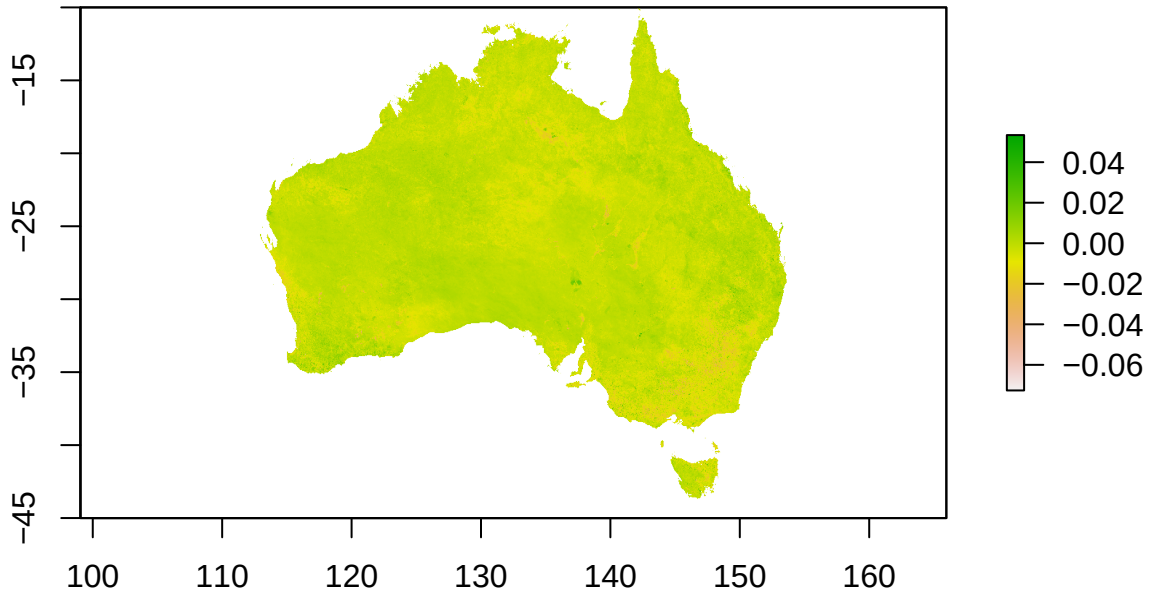
In all the modelling we will show here, each environmental predictor needs to cover the same area (have the same extent), have grid cells that are the same size (same resolution), and use the same coordinate system to define a method for turning a spherical planet earth into a flat map (same coordinate system & map projection = Coordinate Reference System - CRS)

A base layer is one with a CRS, extent and resolution that all other layers will be turned into. Generally it is a good idea to use the largest possible extent, i.e. you often get better predictions when you include the entire range of your species, and when you use the finest (smallest grid cells) possible. Smaller grid cells only help prediction when the values vary from one grid cell to the next grid cell. I recommed choosing the variable that is most closely related to the ecology of your species and has the smallest resolution (grid cell size). Which variable do you think will predict best where your species are found?

```
# this is Enhanced Vegetation Index (EVI) which captures greenness index while correcting ndvi issues, 
# https://api.data-ingester.app.ecocommons.org.au/api/data/34ab5ea6-650f-503f-9446-88d0ae9effe1/downloa

EVI<-raster("raw_data/ndlc-2004-250m_trend-evi-mean.tif")
```

```
plot(EVI)
```




```
EVI
```

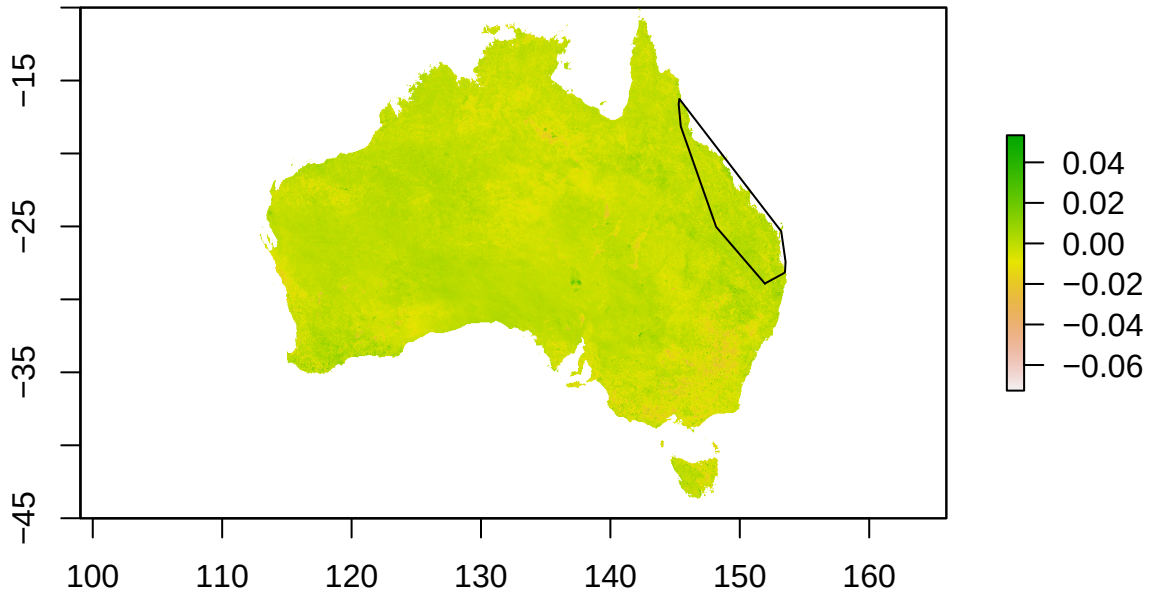
```
## class      : RasterLayer
## dimensions : 14902, 19161, 285537222  (nrow, ncol, ncell)
## resolution : 0.002349, 0.002349  (x, y)
## extent     : 110, 155.0092, -45.0048, -10  (xmin, xmax, ymin, ymax)
## crs        : +proj=longlat +datum=WGS84 +no_defs
## source     : ndlc-2004-250m_trend-evi-mean.tif
## names      : ndlc.2004.250m_trend.evi.mean
## values     : -0.09511322, 0.08792239  (min, max)
```

```
LiPe_pts <- SpatialPoints(coords = cbind(LiPe$decimalLongitude, LiPe$decimalLatitude),CRS(as.character(
```

```
require(rgeos)
LiPe_convH<- rgeos::gConvexHull(LiPe_pts)
```

```
# check if your convex hull looks correct by plotting
# useful information on working with spatial data in R https://cengel.github.io/R-spatial/intro.html

plot(EVI)
lines(LiPe_convH)
```



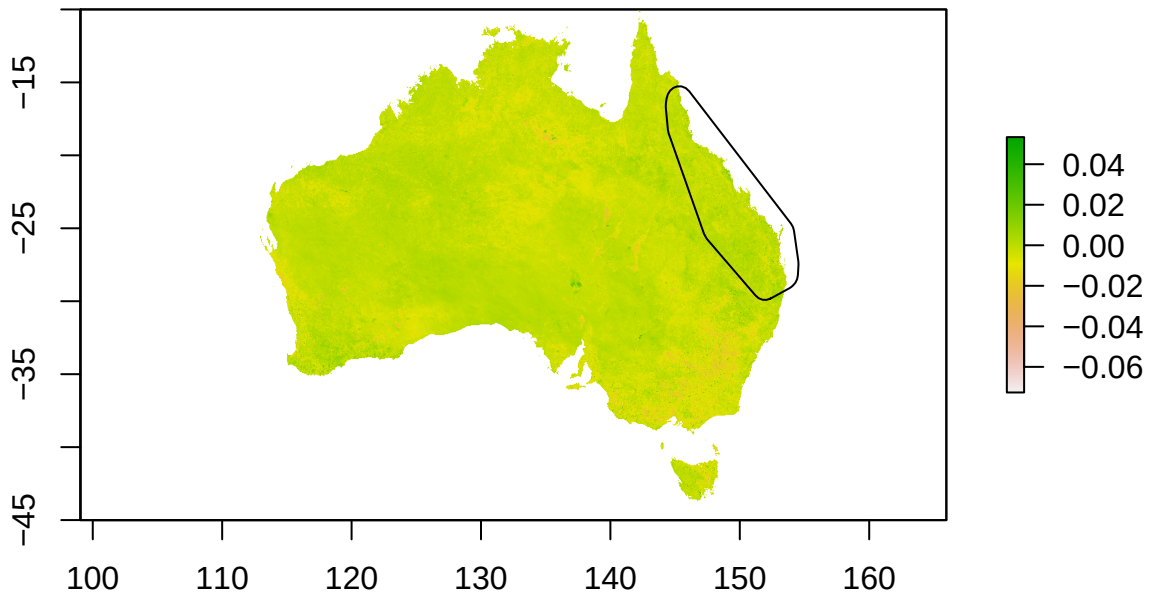

```
#increase the size of your extent (study area to beyone the extent of your point data)
# our CRS is in decimal degrees so 1 degree ~ 111km larger

LiPe_extent <- rgeos::gBuffer(LiPe_convH, width = 1)
```

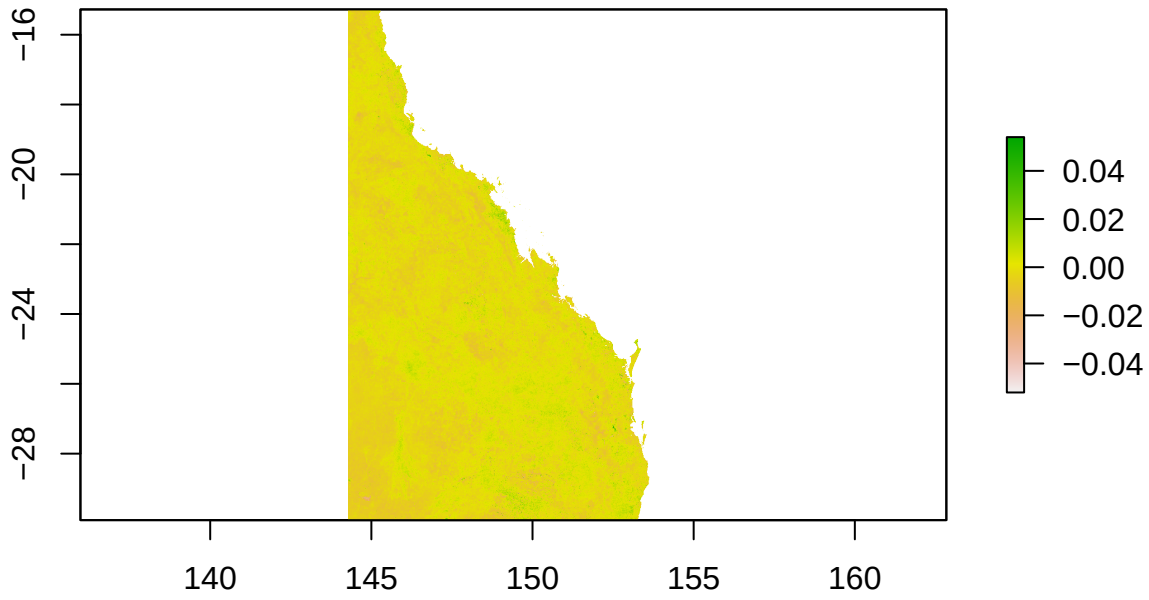
```
## Warning in rgeos::gBuffer(LiPe_convH, width = 1): Spatial object is not
## projected; GEOS expects planar coordinates
```

```
# check if your convex hull or extent is now larger

plot(EVI)
lines(LiPe_extent)
```



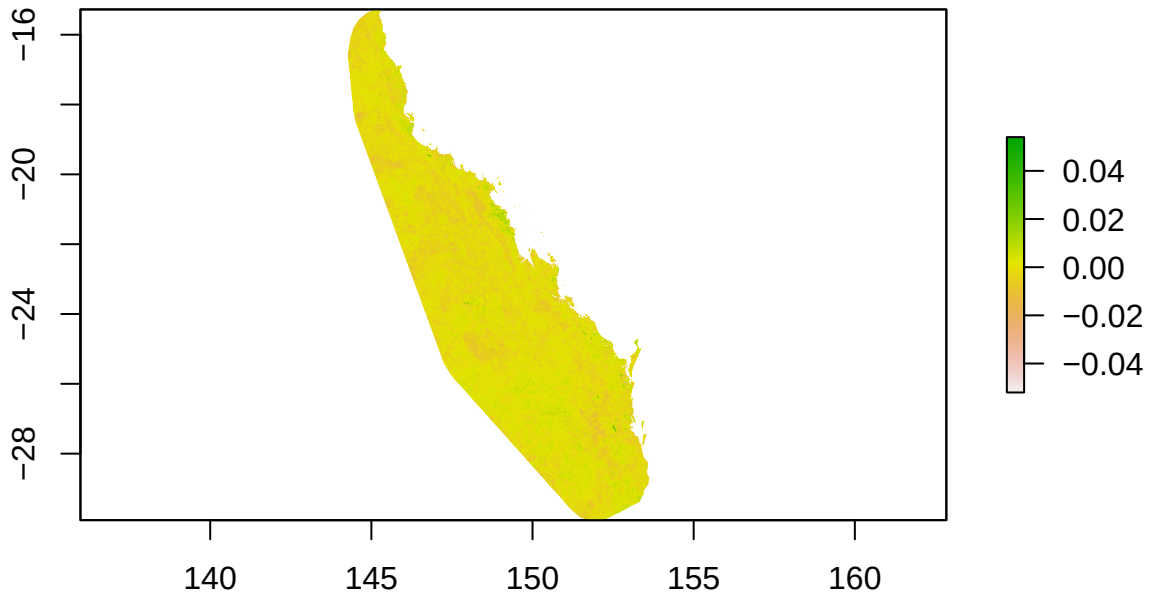

```
EVI_LiPe<- raster::crop(EVI,LiPe_extent)
plot(EVI_LiPe)
```




```
EVI_LiPe
```

```
## class      : RasterLayer
## dimensions : 6228, 4366, 27191448  (nrow, ncol, ncell)
## resolution : 0.002349, 0.002349  (x, y)
## extent     : 144.279, 154.5347, -29.90543, -15.27585  (xmin, xmax, ymin, ymax)
## crs        : +proj=longlat +datum=WGS84 +no_defs
## source     : memory
## names      : ndlc.2004.250m_trend.evi.mean
## values     : -0.07367065, 0.06311681  (min, max)
```

```
EVI_LiPe_mask <- raster::mask(EVI_LiPe,LiPe_extent)
plot(EVI_LiPe_mask)
```

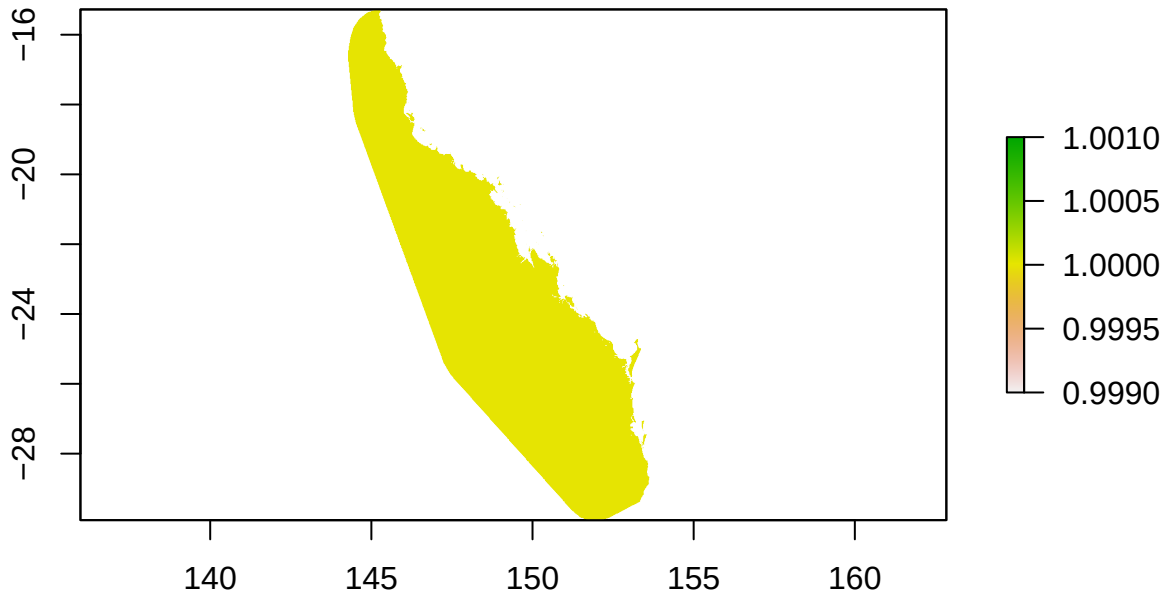



```
# create your reference raster, all other rasters will be set to this resolution, extent and CRS, divid
```

```
base <- EVI_LiPe_mask/EVI_LiPe_mask
```

```
crs(base)<-crs(EVI)
```

```
plot(base)
```




```
writeRaster(base,"data/base_LiPe.asc", overwrite=TRUE)
```

```
getwd()
```

```
## [1] "/Users/s2992269/Documents/Use_cases/SDM_in_R"
```

run these lines if you are coming back to the script to load your base layer

require(raster) base <- raster("data/base_LiPe.asc") #check that it looks correct plot(base)

spatially thin occurrence records so that only one record is selected randomly from within a Grid cell that is 4 times larger than "base"

```
#aggregate reduce resolution (make grid cells larger) (factor = 4) notice grid cells are now 4 times la
large_base <- aggregate(base, fact=4)
res(base)
```

```
## [1] 0.002349 0.002349
```

```
res(large_base)
```

```
## [1] 0.009396 0.009396
```

```
LiPe <- read.csv("data/Limnodynastes peroni.csv")
LiPe_pts <- sp::SpatialPoints(coords = cbind(LiPe$decimalLongitude, LiPe$decimalLatitude),CRS(as.charac

cell_no<- raster::extract(large_base,LiPe_pts,cellnumbers=TRUE)
head(cell_no)
```

```
##        cells layer
## [1,] 1446728     1
## [2,]  182521     1
## [3,] 1299311     1
## [4,] 1404135     1
## [5,] 1421616     1
## [6,] 1419428     1
```

```
LiPe_cells<- cbind(LiPe,cell_no)

head(LiPe_cells)
```

```
##   X decimalLatitude decimalLongitude            eventDate        scientificName
## 1 1       -27.72192         152.9221 2020-03-12T13:00:00Z Limnodynastes peronii
## 2 2       -16.85200         145.7490 2019-06-28T14:00:00Z Limnodynastes peronii
## 3 3       -26.45079         152.9461 2018-06-08T14:00:00Z Limnodynastes peronii
## 4 4       -27.35331         152.8712 2020-06-13T14:00:00Z Limnodynastes peronii
## 5 5       -27.50329         152.9568 2018-09-21T14:00:00Z Limnodynastes peronii
## 6 6       -27.48866         152.9152 2019-09-22T14:00:00Z Limnodynastes peronii
##     cells layer
## 1 1446728     1
## 2  182521     1
## 3 1299311     1
## 4 1404135     1
## 5 1421616     1
## 6 1419428     1
```

```
require(dplyr)
```

```
## Loading required package: dplyr
```

```
## Warning: package 'dplyr' was built under R version 4.1.2
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:rgeos':
##
##     intersect, setdiff, union
```

```
## The following objects are masked from 'package:raster':
##
##     intersect, select, union

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

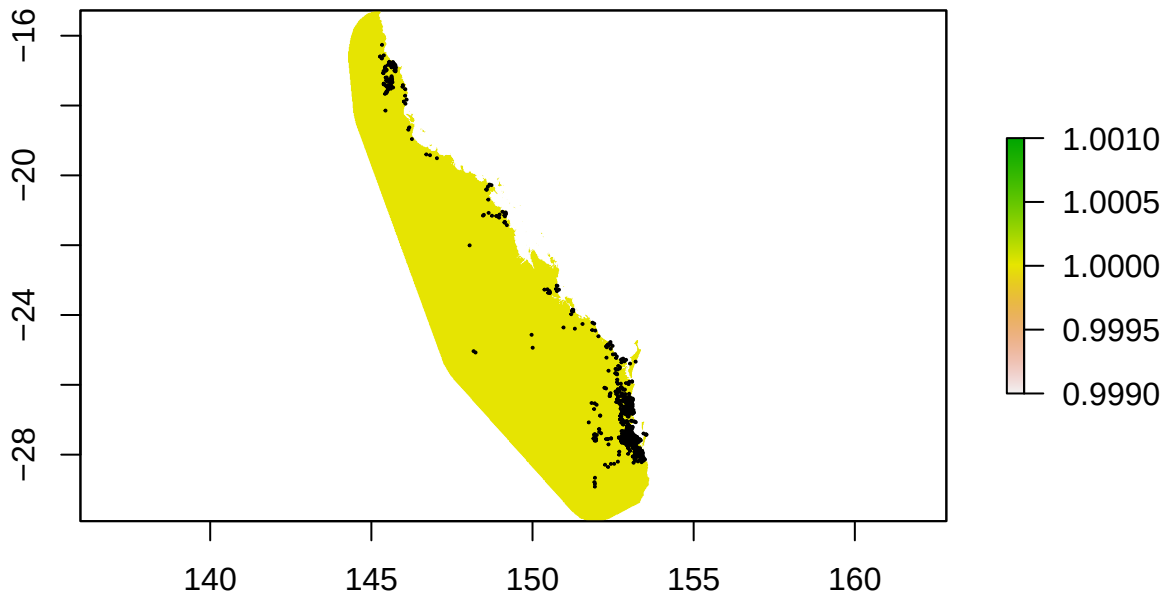
```
LiPe_thinned <- LiPe_cells %>%
  group_by(cells) %>%
  slice_sample(n = 1)

length(LiPe_thinned$cells)
```

```
## [1] 1006
```

```
# plot to look at results & write results to file

LiPe_pts2 <- SpatialPoints(coords = cbind(LiPe_thinned$decimalLongitude, LiPe_thinned$decimalLatitude),C
plot(base)
points(LiPe_pts2,pch=20,cex=0.2)
```



```
write.csv(LiPe_thinned,"data/LiPe_thinned.csv")
```

download all FrogID survey records to capture survey effort, spatial sampling bias, and to zero-fill

```
galah_call() %>%
  galah_identify("Amphibia")%>%
  galah_filter(datasetName == "FrogID")%>%
  galah_filter(coordinateUncertaintyInMeters < 100)%>%
  galah_filter(stateProvince == "Queensland")%>%
  galah_select("datasetName")%>%
  atlas_counts()
```

```
## # A tibble: 1 x 1
##   count
##   <int>
## 1 35457
```

```
frogs <- galah_call() %>%
  galah_identify("Amphibia")%>%
  galah_filter(datasetName == "FrogID")%>%
  galah_filter(coordinateUncertaintyInMeters < 100)%>%
  galah_filter(stateProvince == "Queensland")%>%
  galah_select(datasetName,occurrenceID)%>%
  atlas_occurrences()
```

```
## This query will return 35457 records
```

```
##   |                                                                      |
```

```
head(frogs)
```

```
## # A tibble: 6 x 9
##   datasetName occurrenceID decimalLatitude decimalLongitude eventDate
##   <chr>       <chr>                  <dbl>            <dbl> <chr>
## 1 FrogID      102079                 -27.4             153. 2020-03-20T13:00:00Z
## 2 FrogID      20748                  -17.0             145. 2018-03-08T13:00:00Z
## 3 FrogID      50925                  -12.8             143. 2018-12-29T13:00:00Z
## 4 FrogID      260356                 -26.4             153. 2020-09-20T14:00:00Z
## 5 FrogID      228744                 -26.8             153. 2018-12-19T13:00:00Z
## 6 FrogID      165818                 -27.6             153. 2018-09-27T14:00:00Z
## # ... with 4 more variables: scientificName <chr>, taxonConceptID <chr>,
## #   recordID <chr>, dataResourceName <chr>
```

```
length(frogs$occurrenceID)
```

```
## [1] 35457
```

```
frogs$unique_visit<- paste0(frogs$decimalLatitude,frogs$decimalLongitude,frogs$eventDate)
```

```
length(unique(frogs$unique_visit))
```

```
## [1] 17789
```

```
length(unique(frogs$eventDate))
```

```
## [1] 1026
```

```
frogs$visitID <- as.numeric(as.factor(frogs$unique_visit))
```

```
length(unique(frogs$visitID))
```

```
## [1] 17789
```

```
head(frogs$visitID, 100)
```

```
##   [1] 13564  3334   193  9040 11533 15099  4707 13215  5048 15579  9383  2481
##  [13]  7488 17066  8931 12823 11304  9842 10142  1450 14260  3349  6997   481
##  [25] 14526  4622  1450   183  7382 12638  6888 13158 17498  2077   772 13100
##  [37] 10281  9585  3696  2633 16232 12827  4740  9054 14059 15511  3439  2696
##  [49] 10399  3805  6892    23  8270  9144 10216 12824 15472  4707 15368  9469
##  [61]  8270 13491  6268 15387 15966 15040   449 13285  8279 11563 11178  1639
##  [73] 14830  6106 11428  8781 10707  1995  5858  2855   581  8704  3617    94
##  [85] 17366 13964 13108  9135  3153 16797  8852  8395 10027 13146 12133  9135
##  [97] 13855 12951  9288 14594
```

```
head(frogs)
```

```
## # A tibble: 6 x 11
##   datasetName occurrenceID decimalLatitude decimalLongitude eventDate
##   <chr>       <chr>                  <dbl>            <dbl> <chr>
## 1 FrogID      102079                 -27.4             153. 2020-03-20T13:00:00Z
## 2 FrogID      20748                  -17.0             145. 2018-03-08T13:00:00Z
## 3 FrogID      50925                  -12.8             143. 2018-12-29T13:00:00Z
## 4 FrogID      260356                 -26.4             153. 2020-09-20T14:00:00Z
## 5 FrogID      228744                 -26.8             153. 2018-12-19T13:00:00Z
## 6 FrogID      165818                 -27.6             153. 2018-09-27T14:00:00Z
## # ... with 6 more variables: scientificName <chr>, taxonConceptID <chr>,
## #   recordID <chr>, dataResourceName <chr>, unique_visit <chr>, visitID <dbl>
```

```
names(frogs)
```

```
##  [1] "datasetName"      "occurrenceID"     "decimalLatitude"  "decimalLongitude"
##  [5] "eventDate"        "scientificName"   "taxonConceptID"   "recordID"
##  [9] "dataResourceName" "unique_visit"     "visitID"
```

```
frogs2<-frogs[,c(1,3:6,11)]
```

```
head(frogs2)
```

```
## # A tibble: 6 x 6
##   datasetName decimalLatitude decimalLongitude eventDate  scientificName visitID
##   <chr>                 <dbl>            <dbl> <chr>      <chr>            <dbl>
## 1 FrogID                -27.4             153. 2020-03-2~ Litoria fallax   13564
## 2 FrogID                -17.0             145. 2018-03-0~ Litoria rubel~    3334
## 3 FrogID                -12.8             143. 2018-12-2~ Litoria infra~     193
## 4 FrogID                -26.4             153. 2020-09-2~ Litoria fallax    9040
## 5 FrogID                -26.8             153. 2018-12-1~ Mixophyes fas~   11533
## 6 FrogID                -27.6             153. 2018-09-2~ Adelotus brev~   15099
```

```
getwd()
```

```
## [1] "/Users/s2992269/Documents/Use_cases/SDM_in_R"
```

```
write.csv(frogs2,"raw_data/FrogID_all_ALA_Mar2022.csv")
```

```
require(dplyr)
names(frogs2)
```

```
## [1] "datasetName"      "decimalLatitude"  "decimalLongitude" "eventDate"
## [5] "scientificName"   "visitID"
```

```
#Richness on each visit
frogs3 <- frogs2 %>%
  group_by(decimalLatitude, decimalLongitude, visitID) %>%
  summarise(no_spp = length(unique(scientificName)))
```

```
## `summarise()` has grouped output by 'decimalLatitude', 'decimalLongitude'. You
## can override using the `.groups` argument.
```

```
head(frogs3)
```

```
## # A tibble: 6 x 4
## # Groups:   decimalLatitude, decimalLongitude [6]
##   decimalLatitude decimalLongitude visitID no_spp
##             <dbl>            <dbl>   <dbl>  <int>
## 1           -28.9             152.   17789      4
## 2           -28.9             152.   17788      4
## 3           -28.9             152.   17786      1
## 4           -28.9             152.   17787      1
## 5           -28.9             152.   17785      1
## 6           -28.9             152.   17784      1
```

```
length(frogs3$decimalLatitude)
```

```
## [1] 17789
```

```
summary(frogs3$no_spp)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   1.000   1.818   2.000  12.000
```

```
# calculate the number of visits to the same lat long location
frogs4 <- frogs3 %>%
  group_by(decimalLatitude, decimalLongitude) %>%
  summarise(no_visits = length(visitID))
```

```
## `summarise()` has grouped output by 'decimalLatitude'. You can override using
## the `.groups` argument.
```

```
length(frogs4$decimalLatitude)
```

```
## [1] 13429
```

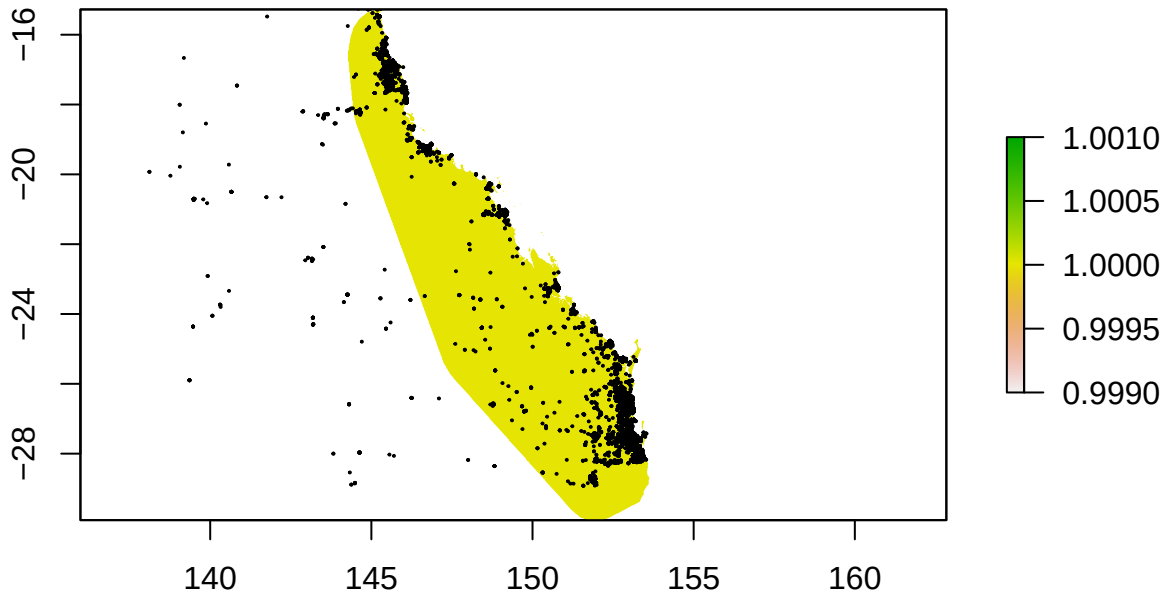
```
summary(frogs4$no_visits)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   1.000   1.325   1.000 161.000
```

```
# note there were 448 visits to one lat long location, but most locations only had one visit
```

```
require(sp)
visits_pts<-SpatialPoints(coords = cbind(frogs4$decimalLongitude, frogs4$decimalLatitude),CRS(as.charac
```

```
#woops we need to include only those points within our study area
plot(base)
points(visits_pts,pch=20,cex=0.2)
```




```
# This creates a raster that totals the total number of visits at each
# Lat / long location
b1 <- rasterize(visits_pts, base, frogs4$no_visits, fun=sum, background=0)
b1
```

```
## class      : RasterLayer
## dimensions : 6228, 4366, 27191448  (nrow, ncol, ncell)
## resolution : 0.002349, 0.002349  (x, y)
## extent     : 144.279, 154.5347, -29.90543, -15.27585  (xmin, xmax, ymin, ymax)
## crs        : +proj=longlat +datum=WGS84 +no_defs
## source     : memory
## names      : layer
## values     : 0, 296  (min, max)
```

This is one way to generate a bias file, simply make a layer where the value in each cell = the number of surveys done in that cell. Maxent will not work with values of zero in the bias file, so below we add 1 to each cell value in our study area (all values in base = 1). Here you can see that there are too few points to show up on a map. If unable to add the bias layer directly into Maxent (args = c("biaslayer = bias")) - this method does not work for me and unfortunately means that when using R it is not using the FACTORBIASOUT argument, but a work around is to account for bias within Maxent by using your bias grid to determine the probability of sampling that grid cell to generate your background. Here most of the values in the grid are the same value of 1 which in our case means no sampling was done there. So in this example we have a number of sites with an increased probability of being selected as a background point in Maxent, but most of the study area has the same low probability of being selected as background. (Some people might choose to use this)

```
numb_visits_bias <- b1+base
```

Below we highlight an alternative method to generate a bias grid. If we assume areas close to areas where surveys have been conducted

are more likely to have been surveyed, so there is a general spatial sampling bias, we may want to identify those spatial locations

where data was more likely to have come from. you can do this kind of thing with a focal function (see step 2 script),

but furthe below we generate a smoothed spatial surface where the probability of an area being sampled is a function of distance and

sampling intensity. Using a Kernel density function is one way to do this.

```
bb <- bbox(b1)
```

```
visit_locations<- raster::extract(b1, visits_pts, cellnumbers=TRUE)
```

```
# these are the points that only occur in the study area
visit_locations2 <- as.data.frame(na.omit(visit_locations))
```

```
cellID <- unique(visit_locations2$cells)
```

```
xy_visits <- xyFromCell(b1,cell = cellID)
```

```
lg<-c(bb[1,1],bb[1,2])
lt<-c(bb[2,1],bb[2,2])
coordss<-cbind(lg,lt)
v_xy2<-rbind(xy_visits,coordss)
```

For some of these functions kde2d, and xyFromCell I had to increase my local R-environ memory size Step 1: Open terminal,

Step 2:

cd ~ touch .Renviron open .Renviron Step 3: Save the following as the first line of .Renviron:

R_MAX_VSIZE=100Gb

```
require(MASS)
```

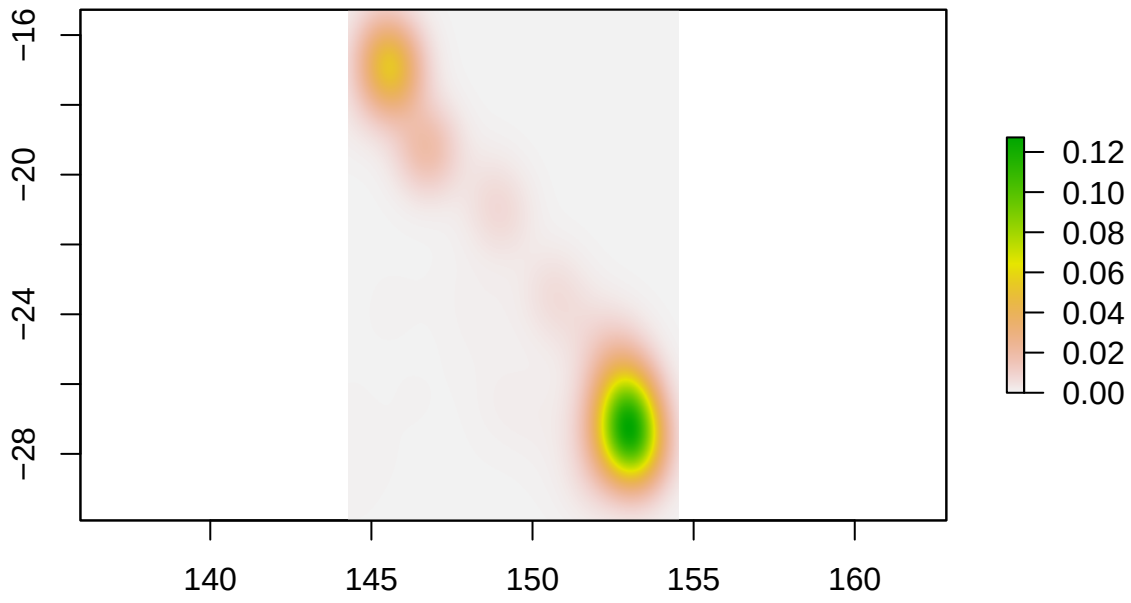
```
## Loading required package: MASS
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

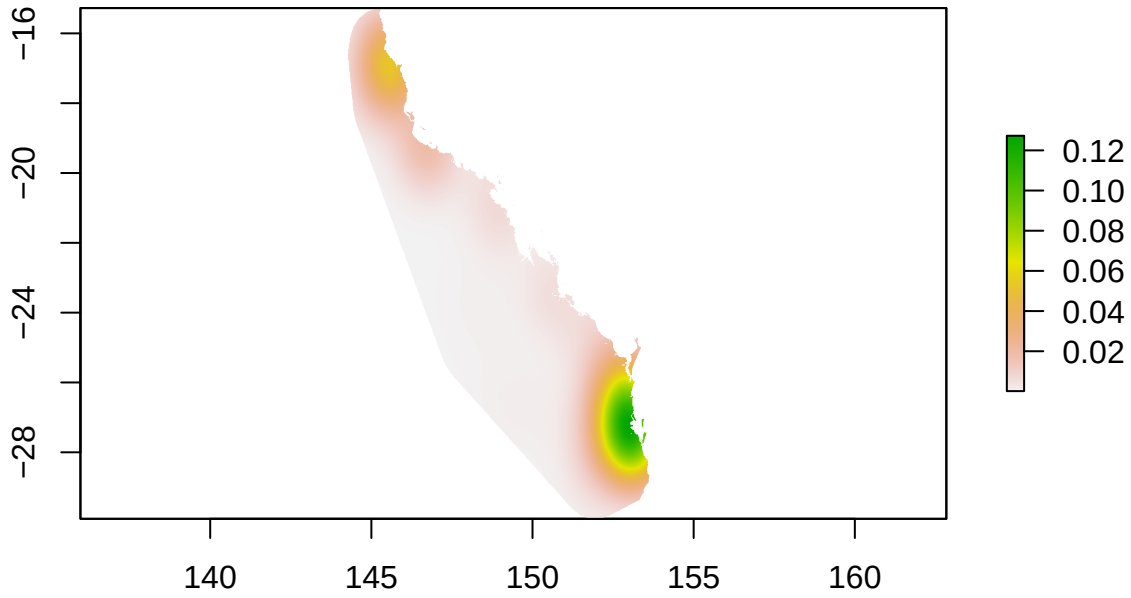
```
## The following objects are masked from 'package:raster':
##
##     area, select
```

```
dens <- kde2d(v_xy2[,1], v_xy2[,2], n = c(nrow(b1), ncol(b1)))
b5 <- raster(dens)
plot(b5)
```




```
b6<-projectRaster(b5,base)
```

```
b7<-mask(b6,base)
plot(b7)
```




```
writeRaster(b7,"data/Bias_LiPe_kd.asc",overwrite=TRUE)
```

Now lets clean up our global environment, and get rid some of these big files

```
rm(b5)
rm(b6)
rm(EVI)
rm(EVI_LiPe)
rm(EVI_LiPe_mask)
rm(frogs2)
rm(frogs3)
rm(visit_locations)
```

Zero filling

In order to identify locations that were surveyed but where LiPe was not detected.

The FrogID project attempts to identify all frog species that were heard during a period of sound recording. While there is a possibility that frogs were present but not detected it is correct to assume that if the target species was not recorded on a certain date and time at a specific lat/long the there was a zero detection.

You can then make some assumptions around the number of surveys needed to be sure there were no frogs detected, while understanding that the spatial resolution of the modelling you are doing may include many frog habitats some of which may not have been surveyed. Still this method of generating pseudo-absences is at least putting zeros in areas where surveys were done, but the species was not detected. If you relax the assumptions further, then species for which counts are usually inclusive of all the species detected (like bird lists), those cells where 10 (the number of surveys will vary by species dataset etc) surveys have been done but which did not detect the target species can be assumed to have zero of that species.

The raster layer we created and named b1 has the number of surveys at all the locations where frogs were surveyed. First, we are going to check how many cells have a value of 3 or higher. So is there enough data to produce pseudo-absence zeros if we assume that when three visits were done in a grid cell we should have identified our target species (here striped marsh frog). In a perfect world we may have needed 10 surveys at each of the habitats within each of the grid cells in order to be certain that the frog is truly absent from this location. Here we have some evidence that the frog was not present, and we have the B1 raster which captures survey effort which is a useful covariate in occupancy modelling (something we won't go into here).

```
# reclassify raster with number of surveys to 1 if more than 2 surveys were done, and zero otherwise

m <- c(0, 2.9, 0,  2.9, 3247, 1)
reclass <- matrix(m, ncol= 3, byrow= TRUE)
rc <- reclassify(b1, reclass)
```

```
visits_3or_more <- mask(rc,base)
# since all values are 1 or zero and ones are where there were three or more visits, cellStats gives us
freq(visits_3or_more)
```

```
##      value    count
## [1,]     0  8787328
## [2,]     1     1294
## [3,]    NA 18402826
```

```
#create a vector of 1's of the same length as the LiPe_pres
LiPe_pres<- rep(1,length(LiPe$decimalLatitude))

# Then create a raster with a value of 1 for each gridcell where a LiPe was recorded
LiPe_pres_raster<-rasterize(LiPe_pts,base,LiPe_pres, fun = min, background=0)

LiPe_pres_raster2<-mask(LiPe_pres_raster,base)
```

```
#How many grid cells of the pre-thinned data had LIPE recorded in them
freq(LiPe_pres_raster2)
```

```
##      value    count
## [1,]     0  8787342
## [2,]     1     1280
## [3,]    NA 18402826
```

```
# confirm that your two raters are the same extent, CRS, resolution etc
compareRaster(LiPe_pres_raster2,visits_3or_more,extent=TRUE, rowcol=TRUE, crs=TRUE, res=TRUE, orig=TRUE
```

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

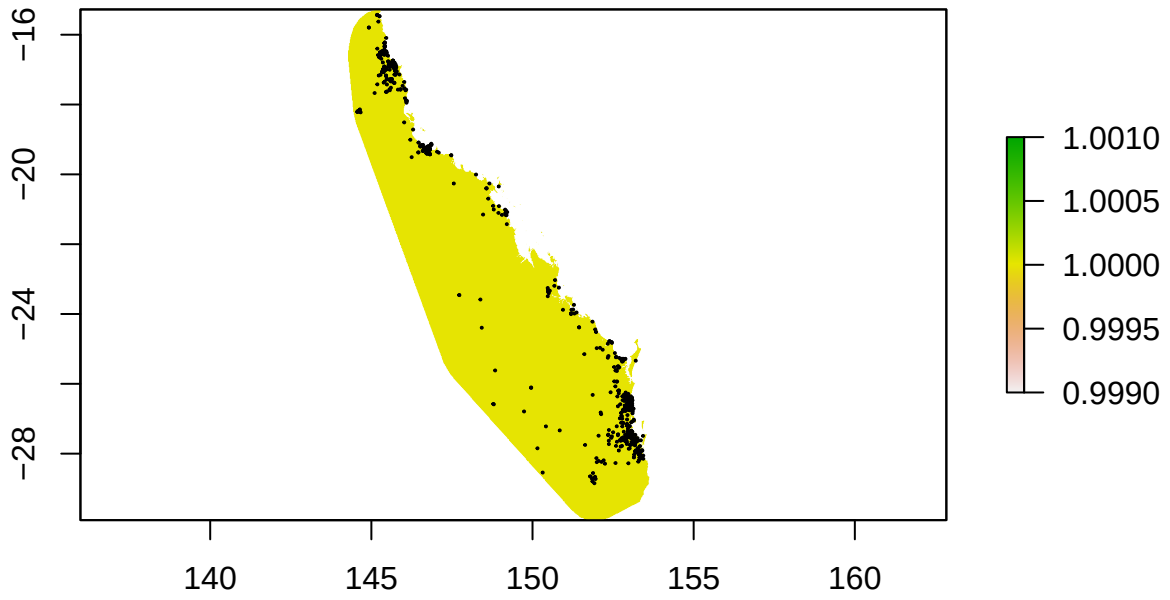
```
Zero_LiPe <- visits_3or_more - LiPe_pres_raster2
# note there are 3179 locations with a value of -1 indicating the number of locations where LiPe was th
freq(Zero_LiPe)
```

```
##      value    count
## [1,]    -1      695
## [2,]     0  8787218
## [3,]     1      709
## [4,]    NA 18402826
```

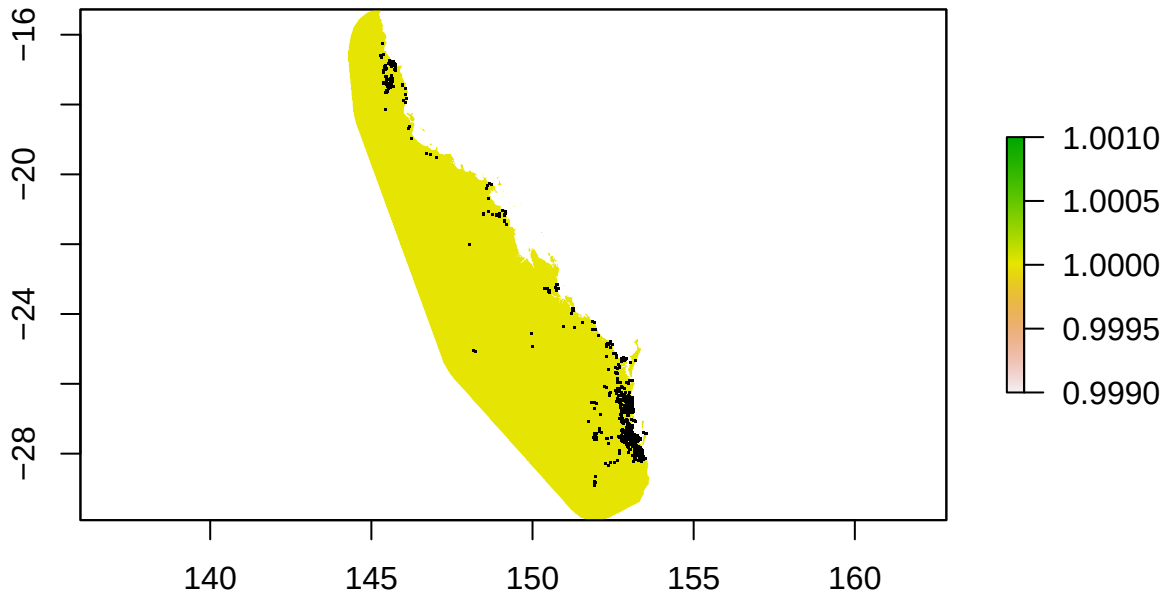
```
#reclassify so a zero raster has the value of 1 for all LiPe zeros, and NA or 0 for all other values
m <- c(-2, 0.1, 0,  0.1, 2, 1)
reclass2 <- matrix(m, ncol= 3, byrow= TRUE)
rc2 <- reclassify(Zero_LiPe, reclass2)
Zero_LiPe2<-mask(rc2,base)
freq(Zero_LiPe2)
```

```
##      value    count
## [1,]     0  8787913
## [2,]     1      709
## [3,]    NA 18402826
```

```
# extract the cell numbers from the 0 grid where the value ==1
cell_vals_0<-Which(Zero_LiPe2 ==1,cells=TRUE)
# these are the lat / longs for locations where at least three surveys were done, but zero LiPe were de
xy_zero_LiPe <- xyFromCell(Zero_LiPe2,cell = cell_vals_0)
#plot the absence locations
plot(base)
points(xy_zero_LiPe, pch=20,cex=0.2)
```




```
# plot the presence locations
plot(base)
points(LiPe[ , c("decimalLongitude", "decimalLatitude")], pch = ".")
```



we will use these pseudo absences for BRT & GLM later

```
LiPe_zeros<-as.data.frame(xy_zero_LiPe)
colnames(LiPe_zeros)<-c("Long","Lat")
LiPe_zeros$spp<-"Limnodynastes_peroni"
LiPe_zeros$pres<- 0
write.csv(LiPe_zeros,"data/LiPe_zero_locations_3vis.csv")

writeRaster(Zero_LiPe2,"data/LiPe_absences1_3vis.asc",overwrite=TRUE)

zeros_3visit <- read.csv("data/LiPe_zero_locations_3vis.csv")
```

```
# reclassify raster with number of surveys to 1 if more than 1 surveys was done, and zero otherwise

m <- c(0, 0.9, 0,  0.9, 3247, 1)
reclass3 <- matrix(m, ncol= 3, byrow= TRUE)
rc3 <- reclassify(b1, reclass3)
freq(rc3)
```

```
##      value    count
## [1,]     0 27186845
## [2,]     1     4603
```

```
# It is a good idea to check your work as you go.  Plotting data, looking at freq are some ways to do t
```

```
t1<-as.data.frame(freq(b1))
sum(t1$count)-t1[1,2]
```

```
## [1] 4603
```

```
# sure enough freq of rc3 returns 29161 cells with a value of 1, and the sum of cell counts with values

# Why am I showing this?  Because my first attempt at generating the rc3 raster did not have equal numb
```

```
visits_1or_more <- mask(rc3,base)
freq(visits_1or_more)
```

```
##      value    count
## [1,]     0  8784080
## [2,]     1     4542
## [3,]    NA 18402826
```

```
# confirm that your two raters are the same extent, CRS, resolution etc
compareRaster(LiPe_pres_raster2,visits_1or_more,extent=TRUE, rowcol=TRUE, crs=TRUE, res=TRUE, orig=TRUE
```

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

```
Zero_LiPe1visit <- visits_1or_more - LiPe_pres_raster2
freq(Zero_LiPe1visit)
```

```
##      value    count
## [1,]     0  8785360
## [2,]     1     3262
## [3,]    NA 18402826
```

```
# here we have 22134 grid cells where another frog(s) was recorded but not our target spp.  We only nee
```

```
cell_vals_0_1 <- Which(Zero_LiPe1visit==1,cells=TRUE)

xy_zero_LiPe_1vis <- as.data.frame(xyFromCell(Zero_LiPe2,cell = cell_vals_0_1))

colnames(xy_zero_LiPe_1vis)<-c("Long","Lat")
xy_zero_LiPe_1vis$spp<-"Limnodynastes_peroni"
xy_zero_LiPe_1vis$pres<- 0
write.csv(xy_zero_LiPe_1vis,"data/LiPe_zero_locations_1vis.csv")

LiPe_pts2 <- SpatialPoints(coords = cbind(xy_zero_LiPe_1vis$Long, xy_zero_LiPe_1vis$Lat),CRS(as.charact
cell_no2<- as.data.frame(raster::extract(large_base,LiPe_pts2,cellnumbers=TRUE))

head(cell_no2)
```

```
##   cells layer
## 1    96     1
## 2    98     1
## 3 12093     1
## 4 17568     1
## 5 17569     1
## 6 18661     1
```

notice here I did not need to use a subsequent colnames function

```
LiPe_cells2<- as.data.frame(cbind(Long = xy_zero_LiPe_1vis$Long, Lat = xy_zero_LiPe_1vis$Lat,cell_no = c

head(LiPe_cells2)
```

```
##       Long       Lat cell_no
## 1 145.1798 -15.28173      96
## 2 145.1939 -15.28173      98
## 3 145.0389 -15.38038   12093
## 4 145.1798 -15.43206   17568
## 5 145.1821 -15.43206   17569
## 6 145.1821 -15.43676   18661
```

```
require(dplyr)
LiPe_zeros_thinned_1vis <- LiPe_cells2 %>%
  group_by(cell_no) %>%
  slice_sample(n = 1)

length(LiPe_zeros_thinned_1vis$cell_no)
```

```
## [1] 2404
```

note this in this object there is still more, than 10,000 areas where frog surveys were done, but no LiPe were recorded. We might see how many cells we would have if there were two or more visits, and then thin that result, but we will try to use these and the points from the bias layer to fit our models. There are trade-offs with every modelling decision, some are important for your result, some do not really impact the result. Your decisions on what is best for your model depends on your question, the available data, the ecology of your species, and an understanding of what has worked or is important according to the literature for your species and your kind of question.

```
LiPe_zeros_thinned_1vis$spp<-"Limnodynastes_peroni"
LiPe_zeros_thinned_1vis$pres<- 0
head(LiPe_zeros_thinned_1vis)
```

```
## # A tibble: 6 x 5
## # Groups:   cell_no [6]
##    Long   Lat cell_no spp                  pres
##   <dbl> <dbl>   <dbl> <chr>               <dbl>
```

```
## 1  145. -15.3      96 Limnodynastes_peroni     0
## 2  145. -15.3      98 Limnodynastes_peroni     0
## 3  145. -15.4   12093 Limnodynastes_peroni     0
## 4  145. -15.4   17568 Limnodynastes_peroni     0
## 5  145. -15.4   17569 Limnodynastes_peroni     0
## 6  145. -15.4   18661 Limnodynastes_peroni     0
```

```
write.csv(LiPe_zeros_thinned_1vis,"data/LiPe_zero_locations_thinned_1vis.csv")
```

note if you just use head(LiPe_zeros_thinned_1vis), you won't see all the decimal points in Long and Lat, print.data.frame shows all the decimal

```
print.data.frame(head(LiPe_zeros_thinned_1vis))
```

```
##       Long       Lat cell_no                  spp pres
## 1 145.1798 -15.28173      96 Limnodynastes_peroni    0
## 2 145.1939 -15.28173      98 Limnodynastes_peroni    0
## 3 145.0389 -15.38038   12093 Limnodynastes_peroni    0
## 4 145.1798 -15.43206   17568 Limnodynastes_peroni    0
## 5 145.1821 -15.43206   17569 Limnodynastes_peroni    0
## 6 145.1821 -15.43676   18661 Limnodynastes_peroni    0
```

It is often a good idea to break your data into randomly selected training and testing data. Often you would randomly remove 20% or more of your data and withold that from the model building process. Once your model was finsihed using your training data, you would then test your model with this training data. If you have very little data, bootstrapping can be used to see if removal of a small percentage of data repeatedly changes results. This gives you a good understanding of the confidence intervals around your results and can reduce the impact of outliers on your final result. Cross-validation requires more data. Ideally for cross-validation you break your data into 10 folds (subsets) and compare results between folds, or summarise variability between folds. Some people select folds spatially, with each fold an independent spatial block of data, but random folds seem to work just as well or better. If you can afford to set aside 20% + of your data, withholding a test data set is good practice. Ideally, you will test your model with completely independent data.