AI for the Environment: from AI to Ecological Models

Week 9: morning workshop

Rory Gibb & Ella Browning

07/03/2023

Morning workshop: Spatial data processing, analysis and modelling in R

Today we'll be exploring and analysing some camera trap data from the Masai Mara collected as part of the Biome Health project - see the lecture slides for a general summary of the data and the project. R provides an excellent set of packages for fast and reproducible GIS functionality, which we'll be using in this morning's workshop. Many of these functions for spatial data work are contained within the simple features (*sf*) package. There is a nice tutorial here: https://learning.nceas.ucsb.edu/2019-11-RRCourse/spatial-vector-analysis-using-sf.html

The goal of this morning's session is to familiarise you with with several of these tools for data processing, analysis and visualisation, and there will be code snippets with short exercises interspersed, along with some larger extension exercises at the end if you have time. All the data and environmental layers you'll need for the workshop are in the GitHub, in the "9_AIToEcologicalModels" folder. Please download the whole folder and set this as your working directory, then all the materials you will need are contained within the "data" subfolder.

```r
# dependencies
library(dplyr); library(magrittr); library(terra); library(rgdal); library(sf);
library(ggplot2); library(lme4); library(rstudioapi);library(MetBrewer); library(tibble)

## -- NB -- ##
# Ensure the most up-to-date version of 'terra' is installed

# automatically set file path (or if this doesn't work, manually set your
# working directory to the folder "9_AItoEcologicalModels")
PATH = dirname(rstudioapi::getSourceEditorContext()$path)
setwd(PATH)
```

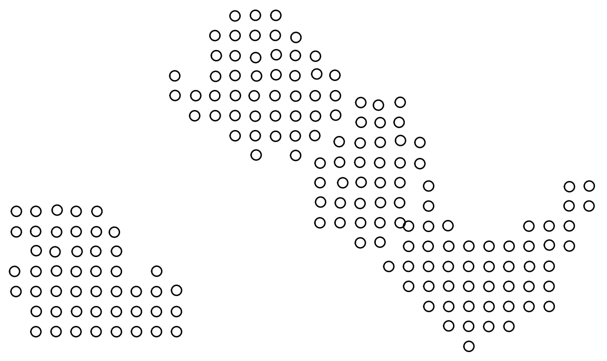
Point, polygon data and mapping in '*sf*'

Let's begin by looking at our camera trap survey locations - these are examples of point data (xy coordinates). We'll read them in, look at them and turn them into an SF (simple features) object for easy spatial handling. Pay attention to the units that the coordinates are stored within (for example, metres versus degrees long-lat); when we are working with several spatial objects, we need to ensure their coordinate reference systems are harmonised so each is comparable.

```r
#read locations
locs = read.csv("./data/kenya/survey/bh_camera_locations.csv")

# use 'head()' and plot() to look at the data
# What are the units of the XY coordinates?

# convert to an sf object by providing the XY columns as coordinates
# initially we create an object in the WGS84 projection with a lat-lon units (CRS = 4326)
locs = sf::st_as_sf(x = locs,
                    coords = c("Longitude", "Latitude"),
                    crs = 4326)

# for such a small area it's much easier to work in metres
# reproject using st_transform to a metres projection
locs = sf::st_transform(locs, crs = "+proj=utm +zone=36 +south +datum=WGS84 +units=m +no_defs")
```

Now we have a "simple feature collection" with 150 points, 1 per sampling site. SF objects can be subsetted and worked with in dplyr like a data frame/tibble, but also have an additional "geometry" column that stores the geographic information (here, lat-lon point locations). We also specified a coordinate reference system when we align this to other spatial data we need to ensure these are aligned.

Use *head()* to take a look at the 'sf' object and how it is formatted. *Note*: this provides information about the spatial extent, units and CRS. To plot the geometry you can call *plot(sf_object$geometry)*



The camera trap locations above were point data, now let's read some polygon data - these are the boundaries of the nature conservancies in the Masai Mara, and of the overall study area. Polygon data are stored as shapefiles. We'll read in these shapefiles, transform them to the same CRS as the locations to ensure everything is harmonised, and plot.

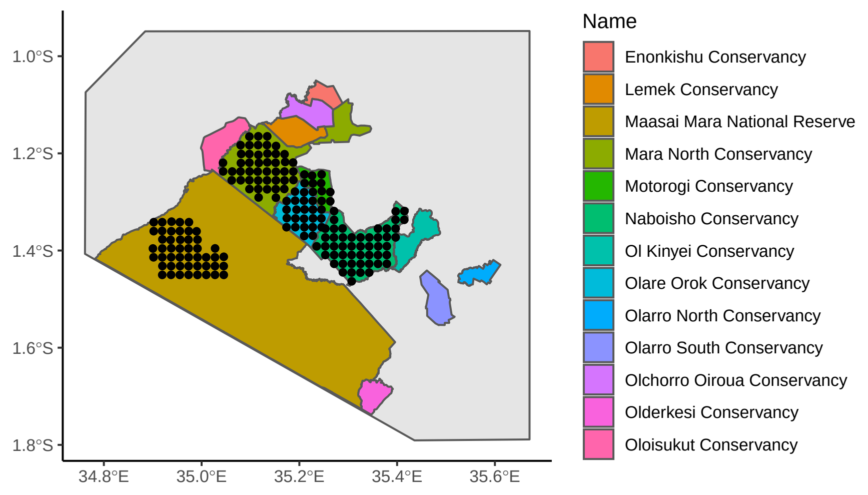
```r
# overall study area (stored as an ESRI shapefile)
study_area = sf::st_read("./data/kenya/survey/shapefiles/Study_area.shp") %>%
  sf::st_transform(crs = sf::st_crs(locs))

# nature conservancies and protected areas that the sampling was taking place within
```

```
conservancies = sf::st_read("./data/kenya/survey/shapefiles/Protected_areas.shp") %>%
  sf::st_transform(crs = sf::st_crs(locs))

# take a look at these by calling head(); what do you notice about the geometry type?
# plot the geometry using plot()
```

sf is really easy to interface with ggplot for visualisation and mapping. We can call *geom_sf()* to plot sf objects, and pass them usual ggplot aesthetics.

Let's create a map of sampling sites over space and how they correspond to the different nature conservancies in the area.

```
# pass ggplot the study area, conservancies and the locations
# ensuring everything has a harmonised CRS means everything is mapped correctly
ggplot() +
  geom_sf(data=study_area, fill="grey90") +
  geom_sf(data=conservancies, aes(fill=Name)) +
  geom_sf(data=locs, color="black") +
  theme_classic()
```



Nice! So we can see where our camera traps were located and how the study was designed spatially - this is a great start.

Mapping a metric of interest over space

During camera trap studies, not every camera trap is operational for the same period of time. It depends on when they were installed, whether there were any malfunction days, and whether they were damaged in situ (e.g. by wildlife). So when we analyse camera trap data, we need to know the distribution of sampling effort - when, and for how long, each camera trap was sampling for.

In the 'data' folder is a CSV of sampling effort per camera; values of NA mean the camera was not installed; 0 means installed but not actively sampling; and 1 means actively sampling. Let's read in the effort csv and map the distribution of effort over space.

```
# read csv of effort and take a look by calling head()
effort = read.csv("./data/kenya/survey/bh_camera_samplingeffort.csv")

# use dplyr's filter, group_by and summarise functions
# to keep only dates when sampling was active
# then sum the number of days of sampling per site
ef = effort %>%
  dplyr::filter(effort_class == 1) %>%
  dplyr::group_by(CT_site) %>%
  dplyr::summarise(n_days_sampled = n_distinct(Date))

# use the 'hist()' function to plot a histogram of sampling effort across sites -
# what do you notice?

# combine with locations data using left_join (this merges data frames based on
# shared columns), and map over space
# use ggplot's aes() function to map the point size to the effort metrics
# what do you notice about the spatial distribution of effort?
locs = locs %>% dplyr::left_join(ef)
ggplot() +
  geom_sf(data=study_area, fill="grey90") +
  geom_sf(data=conservancies, aes(fill=Name)) +
  geom_sf(data=locs, color="black", aes(size=n_days_sampled), alpha=0.5) +
  theme_classic()
```

It's very easy to run all kinds of spatial operations using *sf*. As one example, we might be interested in understanding the distances between our sampling locations, i.e. how far apart were the cameras? *sf::st_distance()* allows us to generate a pairwise matrix of great circle distances between our points, which we could then use to visualise the distances between any given camera and all the others.

In the solutions we've included an example of this, so take a look if you'd like to explore sf functionality further.

Processing, exploring and mapping species detection data for a study species

In the data folder is a CSV containing the tagged camera trap images data for a selection of species identified in the study area. This data frame contains records of individual images, which were tagged to species level - in this case, they were manually tagged by human uers, but could in other instances be automatically classified using an AI classifier. It also contains information about the site, date, species and other key metrics.

If we want to look at how our species are distributed over space, we can process these to site level metrics and map them, just like we did with the sampling effort above. First, let's read in and explore the data.

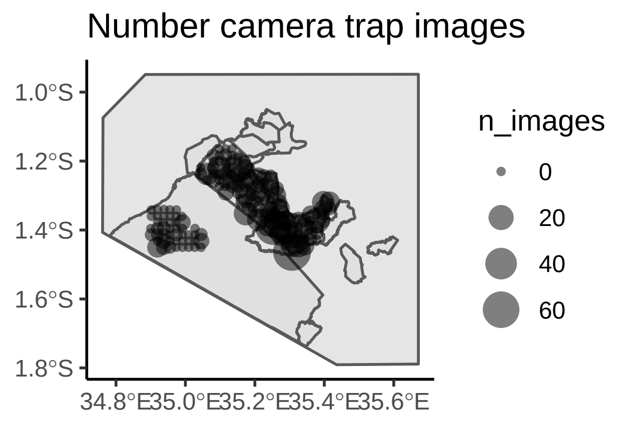
```
# camera trapimages data
ctd = read.csv("./data/kenya/survey/bh_camera_images_mara.csv")

 # explore the data frame using the head(), summary() and table() functions
# to find out what the data contain
# (e.g. how many observations per site/species?)

# it's good practice to always convert temporal data columns to a time format R
# can recognise
# convert Date column into Date class, matching format to how the string
# is written in the CSV
ctd$Date = as.Date(ctd$Date, format="%Y-%m-%d")

# the table function, gives a quick way to summarise observations per column
# here we tabulate species by date to look at the number of images, per species, per day
as.data.frame(table(ctd$Species, ctd$Date)) %>%
  dplyr::mutate(Date = as.Date(Var2)) %>%
  ggplot() +
  geom_line(aes(Date, Freq)) +
  theme_minimal() +
  facet_wrap(~Var1)
```

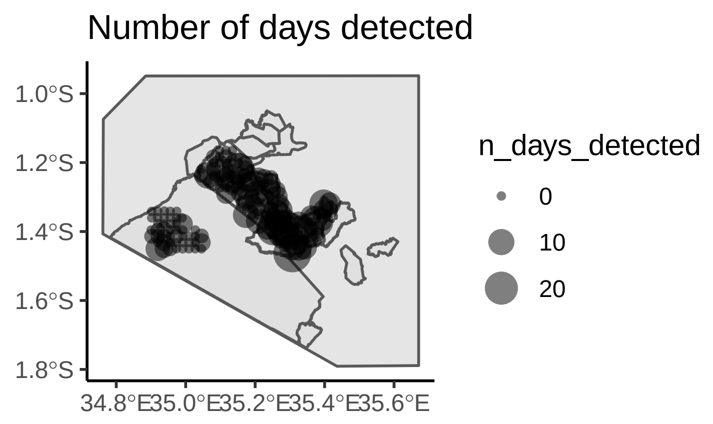
Now let's look more closely at a focal study species, the hare (*Lepus capensis*, the Cape hare). We'll use the same principles as above to map the distribution of our species detections over space, looking both at the total number of images, and the number of days in which a species was detected.

```
# species of interest
sp = "hare"

# filter camera trap data to species of interest
# group by site, then calculate the number of images and the number of unique days
sp_det = ctd %>%
  dplyr::filter(Species == sp) %>%
  dplyr::group_by(CT_site) %>%
  dplyr::summarise(n_images = n_distinct(image),
                   n_days_detected = n_distinct(Date),
                   Species = head(Species, 1))

# take a look at the data using head() and plot histograms of number of days
# detected and number of images
# What do you notice - why might they look different?
```

Exercise 1

- Combine the species detection data with the locations of the camera traps sf_object using *left_join()*. (Note: if there are unmatched rows between the two dataframes being combined, dplyr automatically sets the values to those for NA - you'll need to change these to zeroes so our data contain non-detections). Modify the code above to map these two metrics over space. How similar are the spatial patterns of number of images versus number of days detected? Why might they be more different, and which is a more suitable measure of site occupancy?



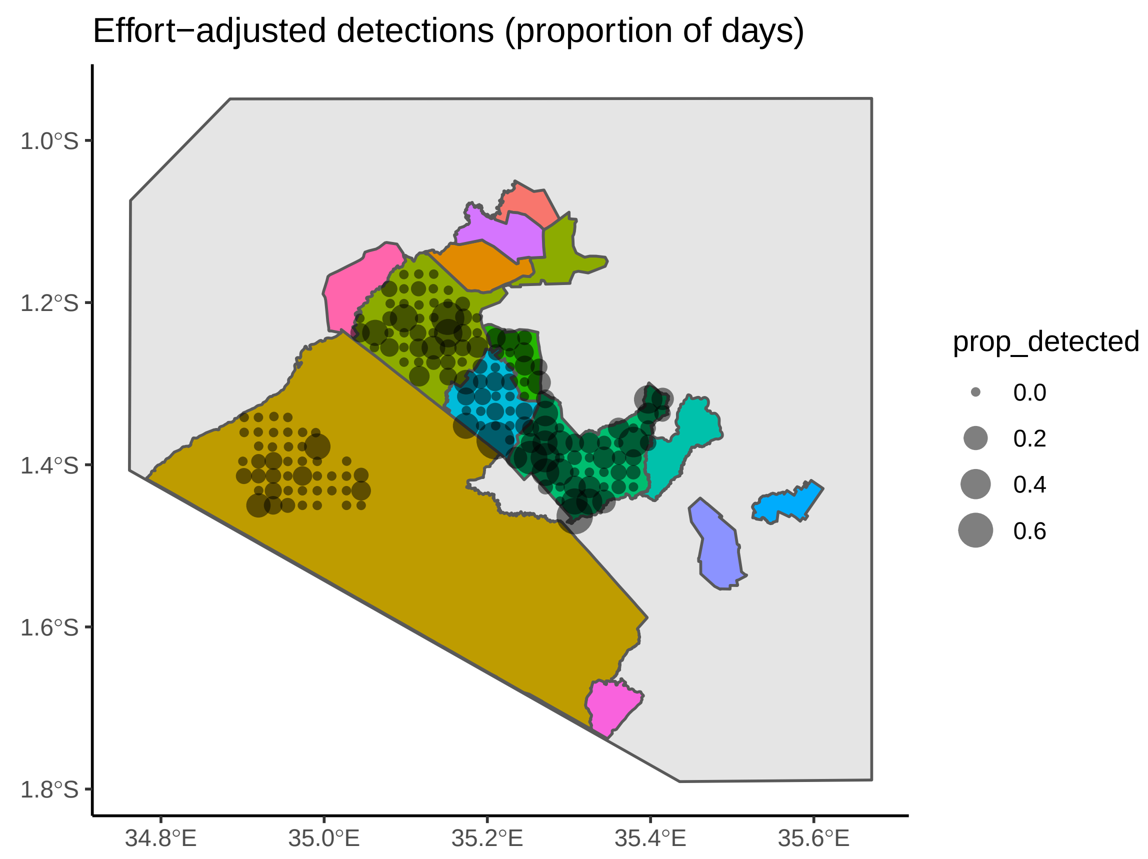


From our work so far, what do you think would be an appropriate measure to use if we were to use a model to analyse the occupancy of this species over space? Would number of days in which a species was detected be an appropriate measure to use by itself? If not, why not? *Hint: what was the distribution of sampling effort like across cameras?*

We need to account for the fact that effort differed between the cameras! Some were operational for very few days, so the number of days detected is not directly comparable between these and cameras that were active for several weeks.

Exercise 2

- Adjust for effort by calculating the proportion of sampled days in which the species was detected in each location, and then map this metric over space. Does this effort-corrected measure of occupancy look different to the metrics that did not account for effort?



Mapping and visualising environmental raster data using *terra*

One of the nice things abut geolocated data is that it's easy to align our species detections with associated environmental information from raster data.

For this, we'll mainly use the "terra" package (and sometimes its predecessor, "raster"; there is a nice introduction and code examples here: https://rspatial.org/pkg/index.html). A raster is essentially a big matrix with geographical information associated with each grid cell. Like sf objects, it has a coordinate reference system that links each cell's XY coordinates with a geographic location. Just like other spatial objects, these follow a particular map projection to map coordinate values to a location on the spherical earth. As in this workshop, these are stored in standardised formats such as GeoTiffs. We'll start by reading, plotting and exploring some rasters of habitat class across the Masai Mara.

```
# read in habitat raster (saved as a GeoTiff file)
hab = terra::rast("./data/kenya/environment/habitat/habitatfinal.tif")

# call "hab" in the console to take a look at the raster properties
# you can look at the CRS details by calling crs(hab) - what units is the projection in?

# try plotting using the plot() function - do you see anything?

# call "hab" in the console and take a look
# what properties does this object have? what do you think they mean?
hab
```

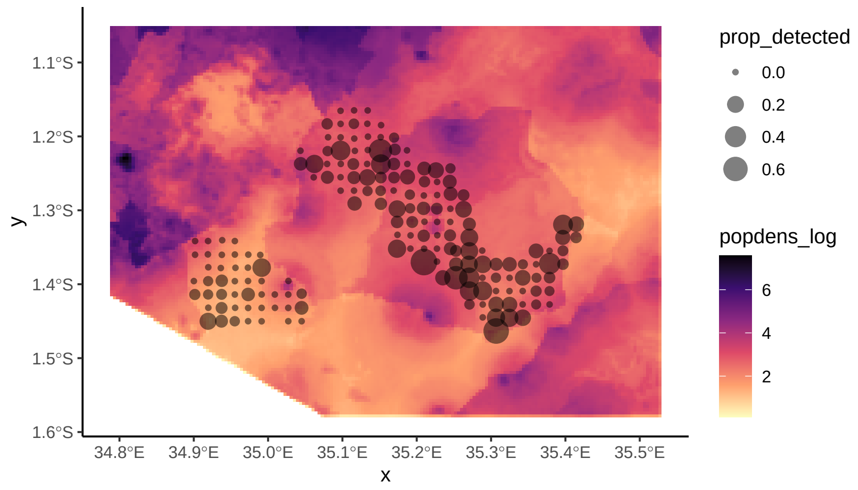
```r
# the raster has numeric values for missing data = 300 and 500
# so we need to replace those with NA and plot again
terra::values(hab)[ terra::values(hab) > 200 ] = NA
terra::plot(hab)
```

We need to ensure the coordinate reference systems are harmonised with our *sf* spatial objects, which we do by reprojecting the raster into the same reference CRS. This effectively overlays our current raster onto a raster projected into the new crs, and then populates the new raster with values. We need to choose a method for this; because our raster is categorical (habitat classes) we use the nearest neighbour distance. If it was continuous, it might be more appropriate to use an interpolation method.

```r
# reproject raster then crop to our study area
hab = terra::project(hab, y=crs(locs), method="near")
hab = terra::crop(hab, terra::ext(locs)+10000)

# add this to our ggplot (make a dataframe from the raster, then call geom_raster())
hab_df = as.data.frame(hab, xy=TRUE)
ggplot() +
  geom_raster(data = hab_df, aes(x, y, fill=factor(habitatfinal))) +
  geom_sf(data=locs, color="black", aes(size=prop_detected), alpha=0.6) +
  scale_fill_discrete(type = as.vector(MetBrewer::met.brewer(name="Archambault", n=6)),
                      name="Habitat") +
  theme_classic()
```

The terra package (and its predecessor raster) also provides efficient ways to manipulate and do grid cell-wise calculations across rasters. Let's look at an example for some human population data for Kenya. We'll read in high-resolution population per grid-cell estimates from WorldPop, which are at ~100x100m grid cell resolution, aggregate to a coarser scale (~500m) then produce a raster of log population density, to visualise how this metric varies across our study area. Here we'll read in some population data, aggregate the raster to coarser resolution, then calculate population density (in persons per square kilometre) rather than total population.

```r
# population raster
pop = terra::rast("./data/kenya/environment/population/worldpop_ppp_mara.tif")
plot(pop)

# this is at very fine scale (~100m), so let's aggregate to a coarser (500m) raster
# aggregates by a factor of 5 (i.e. 25 cells per aggregated cell)
# and sums the population across all the pixels in the aggregation group
pop = terra::aggregate(pop, fact=5, fun="sum", na.rm=TRUE)
plot(pop)

# calculate population density
area_ras = terra::cellSize(pop) # creates raster with cell size in metres squared
area_ras = area_ras / 10^6 # divide by 10^6 to convert to km^2
popdens = pop / area_ras  # calc pop dens
popdens = log(popdens+1) # log transform for ease of use
names(popdens) = "popdens_log" # rename to population density

# plot and overlay our sampling locations
# how does the population density vary across the sampling area, and at its margins?
pop_df = as.data.frame(popdens, xy=TRUE)
ggplot() +
  geom_raster(data = pop_df, aes(x, y, fill=popdens_log)) +
  geom_sf(data=locs, color="black", aes(size=prop_detected), alpha=0.5) +
  scale_fill_viridis_c(option="magma", direction=-1) +
  theme_classic()
```



Extracting environmental values at sampling locations

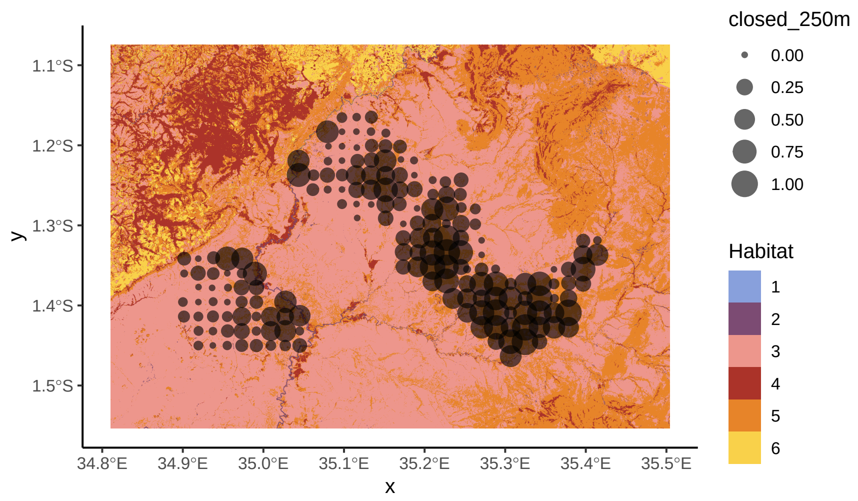
As we have geographical locations of our camera traps (XY point coordinates), and georeferenced raster data in the same projection, it is now straightforward to overlay our sampling locations on the raster and extract the local environmental information. This provides an easy means to access additional geographical covariates for asking ecological questions.

First, let's extract information on population density in the grid cell of each sampling site - here, we're only extracting the value in the specific grid cell that the camera trap falls within.

```r
# ensure projections match (transform the locs CRS to the same as the population raster)
locs_reproj = sf::st_transform(locs, crs=crs(popdens))

# use "extract" function to extract value in the grid cell that each point falls in
locs_pd = terra::extract(popdens, locs_reproj)

# add the extracted values to the "locs" dataframe
locs$popdens_log = locs_pd$popdens_log

# plot a histogram and map this metric over space
# how is population distributed across the study area?
```

Next, we'll extract information about the habitat surrounding each sampling site. Our habitat raster ('hab') is at fine resolution (30m), so extracting the value only in the grid cell the point falls in might be less useful to describing the overall habitat conditions in surrounding area. To deal with this, we can create a buffer around each point (a circular polygon with a user-defined radius, e.g. 250 metres) and then average the habitat conditions within that local area. When we are developing our modelling pipeline, we can think about what size of buffer might be most relevant and how this maps onto our species. Experiment with the code below to see what it looks like when we change the width of the buffer around each camera trap. *Note:* the units of "dist" in st_buffer are the units of its coordinate reference system, which might be metres, or might be degrees long/lat - make sure you check this when you are doing your own projects, and plot the buffers to make sure they're doing what you think!

```r
# plot the habitat raster - it has 6 non-NA classes
plot(hab)

# this table describes the class mapping, so we can easily extract what we need
# by cross-referencing this to the raster
hab_classes = data.frame(class = 1:6,
                         type = c("no data", "water", "open", "closed",
                                  "semi-closed", "agriculture"))

# visualise individual habitat types: plot raster where class_of_interest == TRUE
plot(hab == 6) # agriculture
plot(hab %in% 4:5) # closed or semi-closed

# create buffers around each camera trap using st_buffer
# this is set to 250m but) try different radius sizes and plot them
locs_buf = sf::st_buffer(locs, dist=250)
plot(locs_buf$geometry)

# extract the proportion of closed/semi-closed habitat in the buffer
locs_ext = terra::extract(hab %in% 4:5, # closed/semi-closed
                          locs_buf, # buffer sf
                          fun = "mean") # mean of 1s and 0s gives the proportion cover

# the "extract" function gets computationally expensive quickly for large datasets
# the "exactextractr" package provides a fast and flexible function to do this quickly
# feel free to experiment with this
# locs_ext2 = exactextractr::exact_extract(hab %in% 4:5,
#                                          locs_buf,
#                                          fun = 'mean')
# plot(locs_ext$habitatfinal, locs_ext2)

# add into our locations sf
locs$closed_250m = locs_ext$habitatfinal

# scatterplots against response variable and other covariates
ggplot(locs) + geom_point(aes(popdens_log, closed_250m))
ggplot(locs) + geom_point(aes(popdens_log, prop_detected))
ggplot(locs) + geom_point(aes(closed_250m, prop_detected))

# map
ggplot() +
  geom_raster(data = hab_df, aes(x, y, fill=factor(habitatfinal))) +
  geom_sf(data=locs, color="black", aes(size=closed_250m), alpha=0.6) +
  scale_fill_discrete(type = as.vector(MetBrewer::met.brewer(name="Archambault", n=6)),
                      name="Habitat") +
  theme_classic()
```



Exercise 3.

Extract and visualise the distribution of *open* habitat in a 250m buffer around each camera. Use scatter plots to explore its relationship to closed habitat, to population density, and to our species of interest.

- What do you notice? How might this impact how we later analyse the data?

Extension exercises

If you have time, here are some other things to try.

Exercise 4.

Try extracting the same habitat types with a larger buffer (e.g. 1km or 2km). How different do the values look? * Why might this matter, and if you were analysing these data to answer an ecological question, how might you choose an appropriate buffer size?

Exercise 5

The folder "./data/kenya/environment/chelsa_climatology/" contains a mean climatology (annual mean temperature, averaged betwen 1979 and 2010) for our study location, from CHELSA. Read in the raster, plot for the study area, and overlay points. Extract the temperature values at each location and plot them.

- What do you notice? How useful do you think this covariate might be in explaining the distribution of species across our study area?

Exercise 6

The species detections dataframe also contains data for several other species. By modifying the code above, explore the distribution of total number of images, number days detected, and proportion days detected for other species, using histograms, scatterplots and mapping.

- Are there differences between species in the relationship between number of images, number of days detected, and proportion days detected? How do you think these could relate to the sampling design, and to the ecology of each species?