Data Science for Ecology, Climate Change and Health: spatial data and ecological modelling - SOLUTIONS

Ella Browning & Rory Gibb

22/01/2025

Part 1: 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 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 Finally, we'll use the camera trap and environmental data in some generalised linear models for a preliminary analysis of the drivers of species distributions.

The goal of this 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.

```
# dependencies
library(dplyr); library(magrittr); library(terra); 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)
```

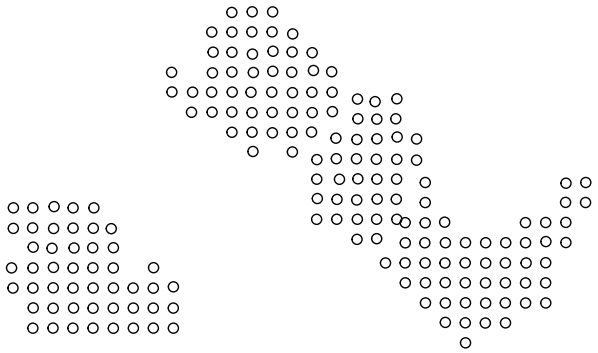
Point, polygon data and mapping in '*sf*'

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.

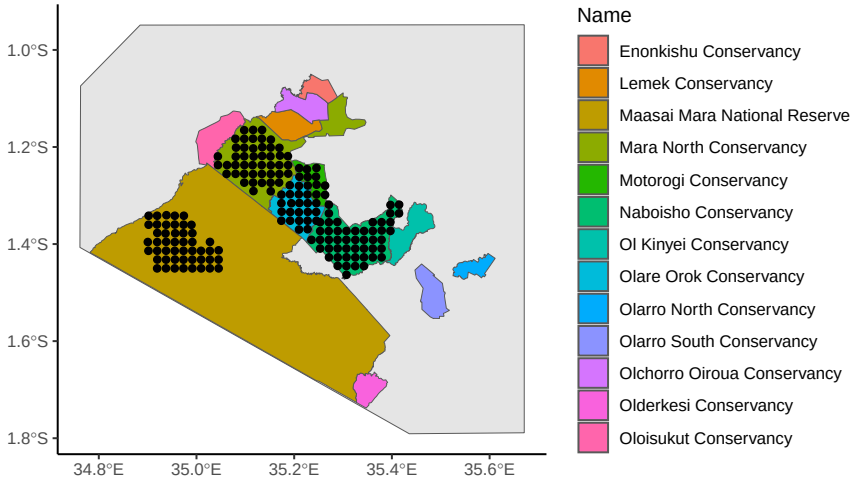
```r
# overall study area (stored as an ESRI shapefile)
study_area = sf::st_read("./data/kenya/survey/shapefiles/Study_area.shp") %>%
  sf::st_transform(crs = sf::st_crs(locs))

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

```r
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.

```r
# 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.

```r
# 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.

```r
# generate pairwise distances matrix (in metres, as this is the CRS unit)
dists = sf::st_distance(locs)

# convert to km
dists = dists / 1000

# what's the maximum distance between any pair of cameras - nearly 60km
max(dists)

# let's look at distances for an arbitrary camera trap, say 13
dist13 = as.vector(dists[ , 13])

# we can calculate the distance to the closest and furthest other sampling
# location (excluding itself which has distance zero)
# betwen 1.9 and 32.1 km
range(dist13[ dist13 != 0])

# add into our locs data sf object using "mutate", and map
locs %>%
  dplyr::mutate(dist_from_13 = as.vector(dist13)) %>%
  ggplot() +
  geom_sf(aes(color=dist_from_13, size=dist_from_13)) +
  theme_classic() +
```

```
  scale_color_viridis_c(end=0.8)
```

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.

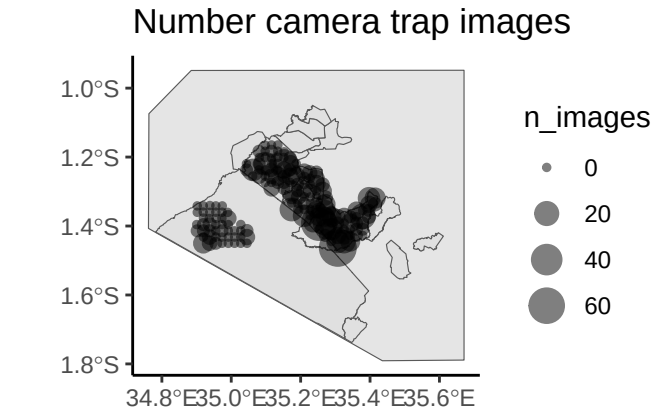
```
# species of interest
sp = "hare"

# 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
```

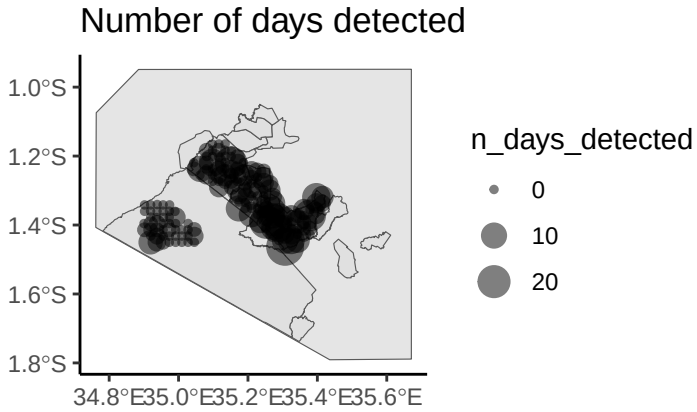
```
# detected and number of images
# What do you notice - why might they look different?
```

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?

```
# use left_join to combine locs with the species detection data
locs = locs %>% dplyr::left_join(sp_det)

# some locations that were sampled will not show up in the image data (if there were no images)
# make sure these are set to zeroes (rather than NAs) in the combined data
locs = locs %>%
  dplyr::mutate(
    n_images = replace(n_images, is.na(n_images), 0),
    n_days_detected = replace(n_days_detected, is.na(n_days_detected), 0)
  )

# map both metrics over space
p1 = ggplot() +
  geom_sf(data=study_area, fill="grey90") +
  geom_sf(data=conservancies, fill="grey88") +
  geom_sf(data=locs, color="black", aes(size=n_images), alpha=0.5) +
  theme_classic() +
  ggtitle("Number camera trap images")
p2 = ggplot() +
  geom_sf(data=study_area, fill="grey90") +
  geom_sf(data=conservancies, fill="grey88") +
  geom_sf(data=locs, color="black", aes(size=n_days_detected), alpha=0.5) +
  theme_classic() +
  ggtitle("Number of days detected")

gridExtra::grid.arrange(p1, p2, nrow=2)
```





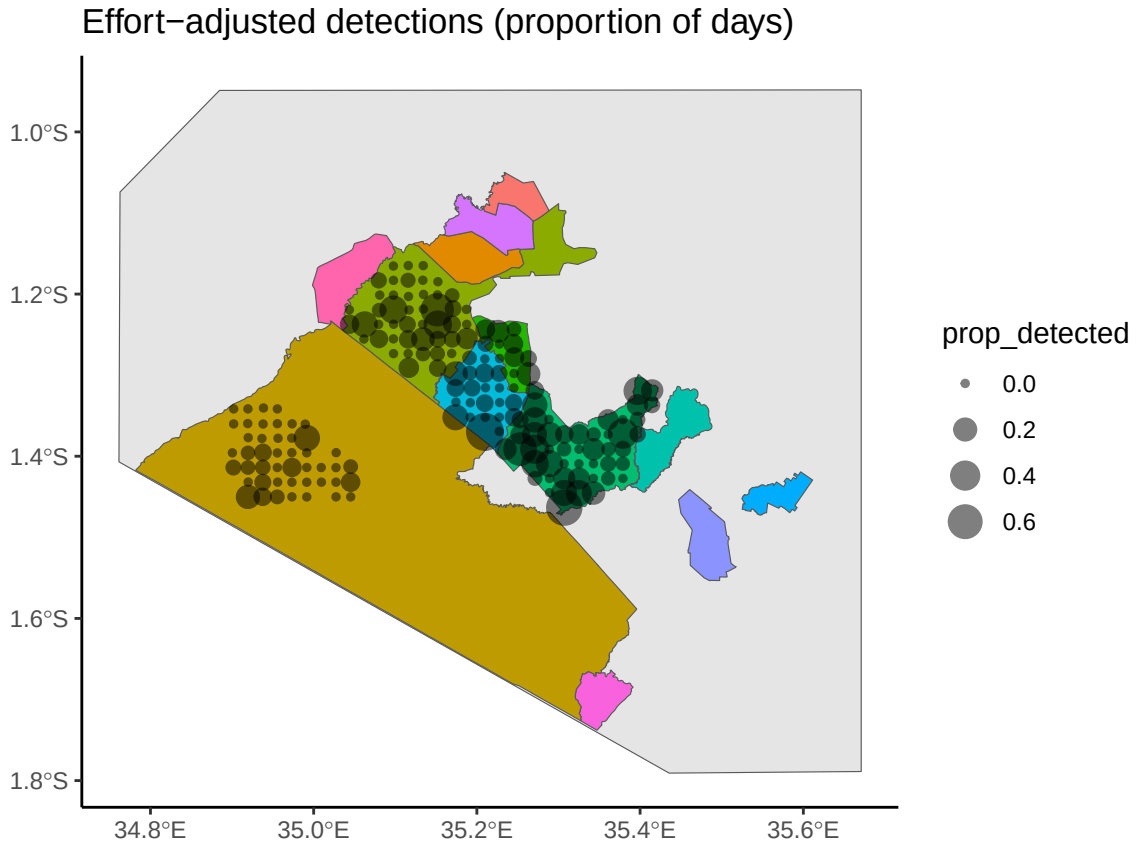
From our work so far, what do you think would be an appropriate measure to use if we were to use a model to analyse the occupancy of this species over space? Would number of days in which a species was detected be an appropriate measure to use by itself? If not, why not? *Hint: what was the distribution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

```
# calculate proportion detected
locs$prop_detected = locs$n_days_detected / locs$n_days_sampled

# plot over space
ggplot() +
  geom_sf(data=study_area, fill="grey90") +
  geom_sf(data=conservancies, aes(fill=Name)) +
  scale_fill_discrete(guide="none") +
  geom_sf(data=locs, color="black", aes(size=prop_detected), alpha=0.5) +
  theme_classic() +
  ggtitle("Effort-adjusted detections (proportion of days)")
```



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
```

```
# 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)
```

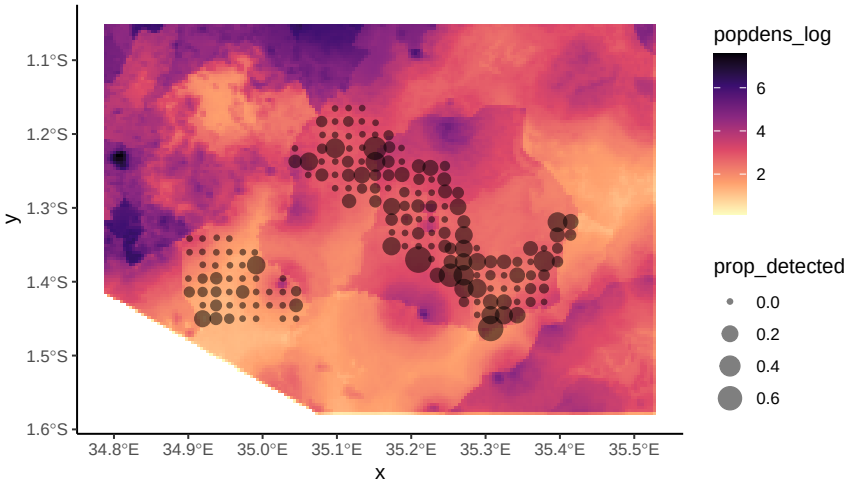
We need to ensure the coordinate reference systems are harmonised with our *sf* spatial objects, which we do by reprojecting the raster into the same reference CRS. This effectively overlays our current raster onto a raster projected into the new crs, and then populates the new raster with values. We 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.

```
# 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.

```
# 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.

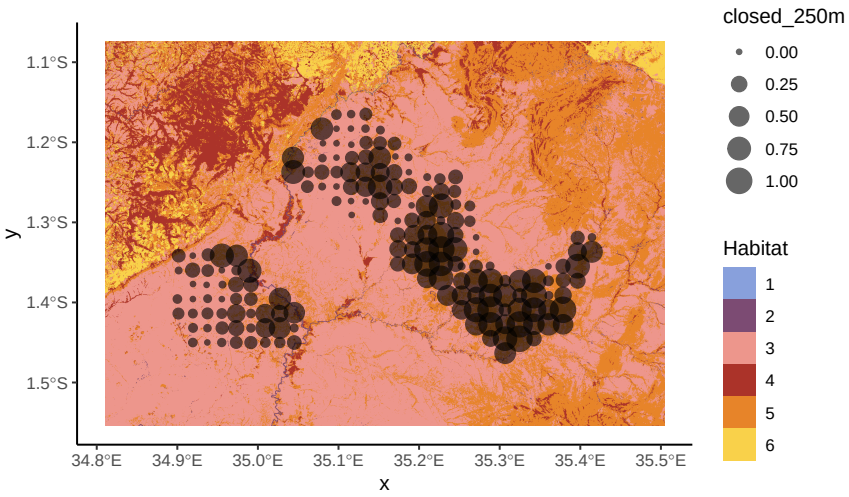
First, let's extract information on population density in the grid cell of each sampling site - here, we're only extracting the value in the specific grid cell that the camera trap falls within.

```
# 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!

```
# 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()
```

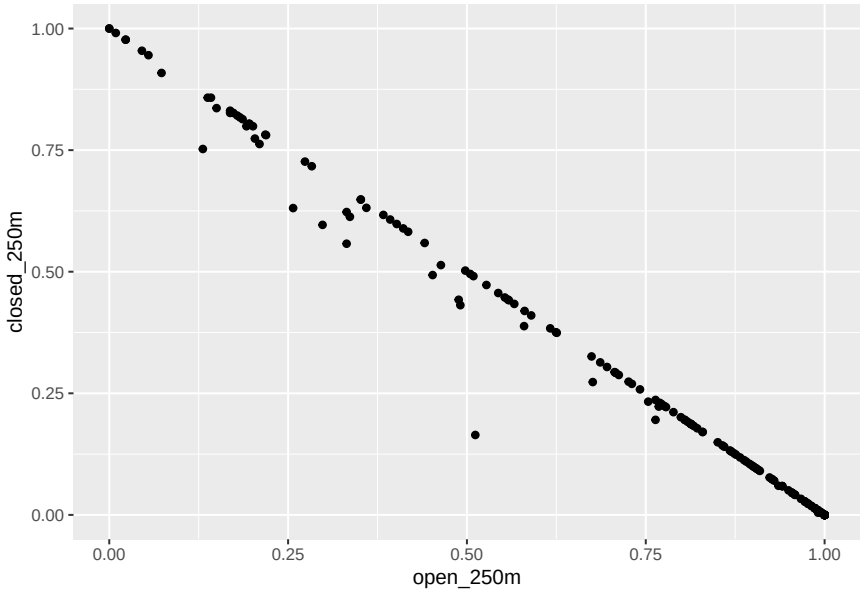


Exercise 3.

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?

```
# creates a TRUE/FALSE (1/0) raster for specific habitat type
locs_ext = terra::extract(hab %in% 3,
                          locs_buf, # buffers
                          fun = "mean") # mean of 1s and 0s gives the proportion cover

locs$open_250m = locs_ext$habitatfinal

# almost perfectly collinear! why?
# what does this mean for inferring ecological relationships with habitat structure?
ggplot(locs) + geom_point(aes(open_250m, closed_250m))
```



Part 1 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?

Part 2: Drivers of species occurrence across the Masai Mara

Now, we're going to do some preliminary investigation into the distribution and drivers of occupancy for our study species (the Cape hare, *Lepus capensis*) in relation to anthropogenic and environmental factors across the Masai Mara, using the camera trap data and spatial data we processed and extracted above. We will explore fitting and evaluating some generalised linear models to infer ecological drivers.

Defining our research question

Let's start with a broad question: what is the relationship between level of anthropogenic pressure and spatial occupancy of our focal species? We can define anthropogenic pressure in many ways, but here, let's focus on livestock pressure. We know that one of the major anthropogenic activities in the Mara ecosystem is livestock grazing, which influences vegetation structure and community composition. Hares generally inhabit grassland and pastoral ecosystems, so **if habitat suitability for hares is shaped by pastoral activity we might expect a positive relationship with livestock**. Alternatively, **if areas with very high levels of livestock activity are too frequently disturbed to provide amenable habitat, we might expect hare occurrence to decline where livestock pressure is highest**. So here we have two alternative, plausible hypotheses.

We might also need to account for other factors that may covary with our drivers of interest and also affect hare presence; here, we'll look at habitat type (proportion of agricultural and closed habitat) and distance to the nearest water body, as well as the conservancy in which the cameras were located.

Environmental and anthropogenic covariates at camera trap locations

Above, we used the spatial locations of the camera traps along with raster data to extract information about the environmental factors around each camera trap. We have now combined all of these together into a full dataframe of site-level covariates for modelling. *In the solutions*, you can see the full code block that we used to produce this dataframe from the raw data sources, but for this exercise we will just read in the final dataframe for our analyses.

The data contain several site-level covariates: proportion of closed/semi-closed habitat or agriculture land use within a 250m radius, distance to the nearest water source, population density, conservancy, and livestock pressure. Livestock pressure was estimated from the tagged camera trap data (*see the code in the solutions*), and defined as the proportion of surveyed days in which livestock were detected.

```
# read in required data (n=178 camera traps)
# create an SF object with the location geometry and site-level covariates
locs = read.csv("./data/kenya/survey/bh_camera_locations.csv") %>%
  sf::st_as_sf(coords = c("Longitude", "Latitude"), crs = 4326) %>%
  sf::st_transform(locs, crs = "+proj=utm +zone=36 +south +datum=WGS84 +units=m +no_defs") %>%
  dplyr::filter(CT_site != "MT34")

# add coordinates columns for XY locations
locs = cbind(locs, sf::st_coordinates(locs))

# covariates for each camera trap
covars = read.csv("./data/kenya/data_processed/bh_site_covariates_spatial.csv") %>%
  dplyr::select(-Conservancy)

# combine and only keep locations where camera sampled for > 0 days (n=175)
locs = locs %>%
  dplyr::left_join(covars) %>%
  dplyr::filter(n_days_sampled > 0)
```

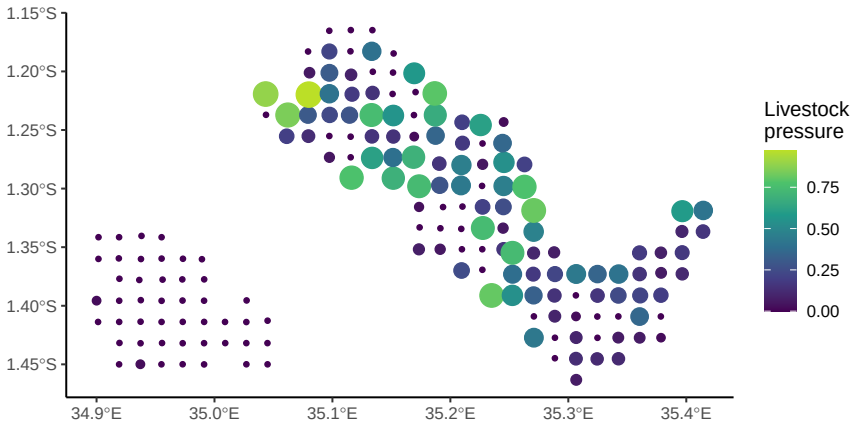
Exercise 6

- Call head() to familiarise yourself with the data structure of *locs*.

- Explore the distribution of our covariates of interest using histograms and scatterplots, and use *ggplot()* to plot boxplots of covariates of interest across different conservancies (see code below). Will all of these covariates be suitable to include in a model?
- Explore mapping anthropogenic and environmental factors - how do these vary across the study area? We should remember any spatial differences between conservancies later when we are running our analyses.

```
# create a longitudinal dataframe of conservancy vs each covariate
locs_longdf = locs %>%
  sf::st_drop_geometry() %>%
  tidyr::pivot_longer(cols = c("closed_lc", "agri_lc",
                               "distance_to_water", "popdens_log", "livestock_pressure"),
                      names_to="covariate", values_to="value")

# boxplots of covariates across each conservancy - what do you notice?
ggplot(locs_longdf) +
  geom_boxplot(aes(factor(Conservancy), value, group=Conservancy, fill=Conservancy)) +
  theme_minimal() +
  facet_wrap(~covariate, scales="free_y") +
  # sets the x-axis text to print at an angle and not overlapping the plot
  theme(axis.text.x = element_text(angle = 45, hjust = 0.8))
```

```
# map livestock pressure across the study area
# change this to look at other covariates!
locs %>%
  ggplot() +
  geom_sf(aes(size=livestock_pressure, color=livestock_pressure)) +
  theme_classic() +
  scale_color_viridis_c(end=0.9, name="Livestock\npressure") +
  scale_size(guide="none")
```



Combining environmental covariates with species detections to create a modelling dataframe

Now let's incorporate survey data for our species of interest (*Lepus capensis*) from the camera trap images. As the day is our unit of sampling, we'll calculate the number of days in which the species was detected, and also the proportion of surveyed days.

```
# specify our species of interest
spp = "hare"

# number of days with hares observed per camera trap
ctd = read.csv("./data/kenya/survey/bh_camera_images_mara.csv") %>%
  dplyr::filter(CT_site %in% locs$CT_site) %>%
  dplyr::mutate(Date = as.Date(Date, format="%Y-%m-%d")) %>%
  dplyr::filter(Species == spp) %>%
  dplyr::group_by(CT_site) %>%
  dplyr::summarise(n_days_detected = n_distinct(Date))

# add to "locs" and replace autofill NAs
locs = locs %>%
  dplyr::left_join(ctd) %>%
  dplyr::mutate(n_days_detected = replace(n_days_detected, is.na(n_days_detected), 0))

# calculate proportion detected and ensure ranges between 0 and 1
locs$prop_detected = locs$n_days_detected / locs$n_days_sampled
range(locs$prop_detected)

# quick viz
ggplot(locs) +
  geom_sf(aes(size=prop_detected, color=prop_detected), alpha=0.8) +
  theme_classic() +
  scale_color_viridis_c(name="Hare\noccupancy\n(proportion\nsampled\ndays)") +
  scale_size(guide="none")
```

Exercise 7

- Plot scatterplots of the relationship between our response variable (proportion hare detections, *"prop_det"*) and our covariates of interest, in particular livestock activity, closed habitat, distance to water and conservancy. Do you see any obvious evidence of relationships in the raw plots?

```
# exercise 2 solution

# visualise
ggplot(locs) + geom_point(aes(livestock_pressure, prop_detected)) + theme_classic()
ggplot(locs) + geom_point(aes(closed_lc, prop_detected)) + theme_classic()
ggplot(locs) + geom_point(aes(distance_to_water, prop_detected)) + theme_classic()
ggplot(locs) + geom_jitter(aes(Conservancy, prop_detected)) + theme_classic()
```

Fitting logistic (binomial) regression models to estimate probability of hare occupancy

Let's investigate our research question using generalised linear models. Our response variable is binomial - i.e. the species either was detected or not detected, during each sampling window. Each camera trap sampled for a particular number of nights ("trials") with a particular number of detections ("successes"), and we are interested in how our covariates affect the *probability* of success. We model this using **logistic regression** with a binomial likelihood and logit link function, where we estimate the linear effects of covariates $X_1 : X_n$ on the log odds of hare occurrence.

The model would be formulated as:

$Y_i \sim Binom(n_i, p_i)$

$log(p_i/(1-p_i)) = \beta_0 + X\beta$

where Y_i is the proportion of successful outcomes (hare detections), n_i is the number of trials (the number of sampled days), and p_i is the probability of success, which we are estimating as a function of covariates. β is a vector of slope parameters, and X is a matrix of covariates.

Our covariates are all on different scales of magnitude to each other, so slope estimates are difficult to compare between covariates. To deal with this we centre and scale covariates - subtract the mean and divide by the standard deviation. This way, slope parameters always describe the change in Y for 1 standard deviation change in X, regardless of what units X was measured in.

```
# scale linear covariates for comparability and save as new variables "_s"
# (denoting scaled)
locs$closed_lc_s = scale(locs$closed_lc)
locs$open_lc_s = scale(locs$open_250m)
locs$livestock_pressure_s = scale(locs$livestock_pressure)
locs$distance_to_water_s = scale(locs$distance_to_water)
locs$popdens_log_s = scale(locs$popdens_log)

# plot histograms of our scaled covariates
# what do you notice in comparison to the unscaled versions?
```

We use the *glm()* function to fit a generalised linear model, defining this formula as "Y ~ covariate + covariate + ...", specifying a binomial likelihood. Since the conservancies showed markedly different levels of livestock activity and habitat factors, we include conservancy as a covariate *a priori* in the model, to account for these coarse spatial differences across the study area.

```
# logistic regression model with livestock and conservancy
# response is the proportion detections
# "weights" argument provides the model with number of trials
m1 = glm(prop_detected ~ livestock_pressure_s + Conservancy,
         family=binomial(link="logit"),
         weights = n_days_sampled,
         data=locs)

# summary information
summary(m1)
```

- Call *summary()* on the model to see summary information including residuals and fitted parameters (coefficients table). What does the slope estimate suggest about the relationship with livestock pressure?

The model summary also shows some goodness-of-fit statistics based on the log-likelihood:

- **Residual deviance**, a measure of how much of the total variation in the observed data is explained by the model (*lower values = more variation explained = better model*). A significant reduction from the null deviance supports the inclusion of covariates in the model.
- **AIC (Akaike Information Criterion)** accounts for the improvement in log-likelihood provided by including more parameters, while penalising overfitting to the data (*lower values = better fitting model*).

We should also look at the distribution of residuals (the unexplained error not accounted for by the model) to check our model is adhering to assumptions. These plots get tricky to interpret visually for GLMs because different likelihoods make different assumptions about how error is distributed (see: https://bookdown.org/ltupper/340f21_notes/deviance-and-residuals.html). Plotting the model shows the Pearson residuals; generally these should be evenly distributed around the fitted line (shown in red).

```
# AIC
AIC(m1)

# plot model
plot(m1)
```

Exercise 8

What other factors might be influencing hare occurrence and also covary with livestock occurrence? It may be important to include these too, in case they explain some of this relationship.

- Fit another model called *m2*, also including closed habitat and distance to water as covariates.
- Call *summary()* to examine the model. Compare the AIC and residual deviance; does adding these covariates improve the model?

```
# include additional covariates
m2 = glm(prop_detected ~ livestock_pressure_s + Conservancy + distance_to_water_s + closed_lc_s,
         family=binomial(link="logit"),
         weights = n_days_sampled,
         data=locs)

# summary
summary(m2)
```

- The function below extracts the fitted parameter estimates, calculates the 95% confidence intervals and visulises them. Use this to plot the slope estimates for *m2*. What does the model suggest about the relationship between hare occupancy, livestock and habitat metrics? What about how hare occupancy differs between conservancies?

```
# Function to plot coefficients and confidence intervals.
# The intercept is often at a different scale to the slope parameters
# so creates a separate sub-plot for the intercept).

plotFixedEffects = function(model){

  plot_df = coef(summary(model)) %>% # extract parameters table from model
```

```
    as.data.frame() %>% # convert to df
    tibble::rownames_to_column(var="param") %>% # make column "param" from row names
    # classify param as either Intercept or Slope
    dplyr::mutate(param_type = ifelse(param == "(Intercept)", "Intercept", "Slope")) %>%
    dplyr::rename("se"=3) # rename std error variable because easier to work with

  plot = ggplot(plot_df) +
    geom_point(aes(param, Estimate), size=3) + # point estimate
    # 95% confidence interval (1.96 * standard error)
    geom_linerange(aes(param, ymin=Estimate-(1.96*se), ymax=Estimate+(1.96*se))) +
    geom_hline(yintercept=0, lty=2) + # horizontal line marking zero (i.e. no effect)
    theme_minimal() +
    facet_wrap(~param_type, scales="free") + # split plot by parameter type
    theme(axis.text = element_text(size=12),
          axis.title = element_text(size=12),
          strip.text = element_text(size=14)) +
    xlab("Parameter") + ylab("Estimate (95% confidence interval)") +
    coord_flip() # flip so the plot is horizontal

  return(plot)
}
```

Note that conservancy is a 4-level categorical variable, so the model is estimating how the intercept differs between each category (i.e. are hare detections on average higher or lower in each conservancy?). One of the categories is incorporated into the intercept (the base factor), and the 3 other parameters estimate the log odds difference in hare occurrence between the base factor and each of the other conservancies.

Excercise 9

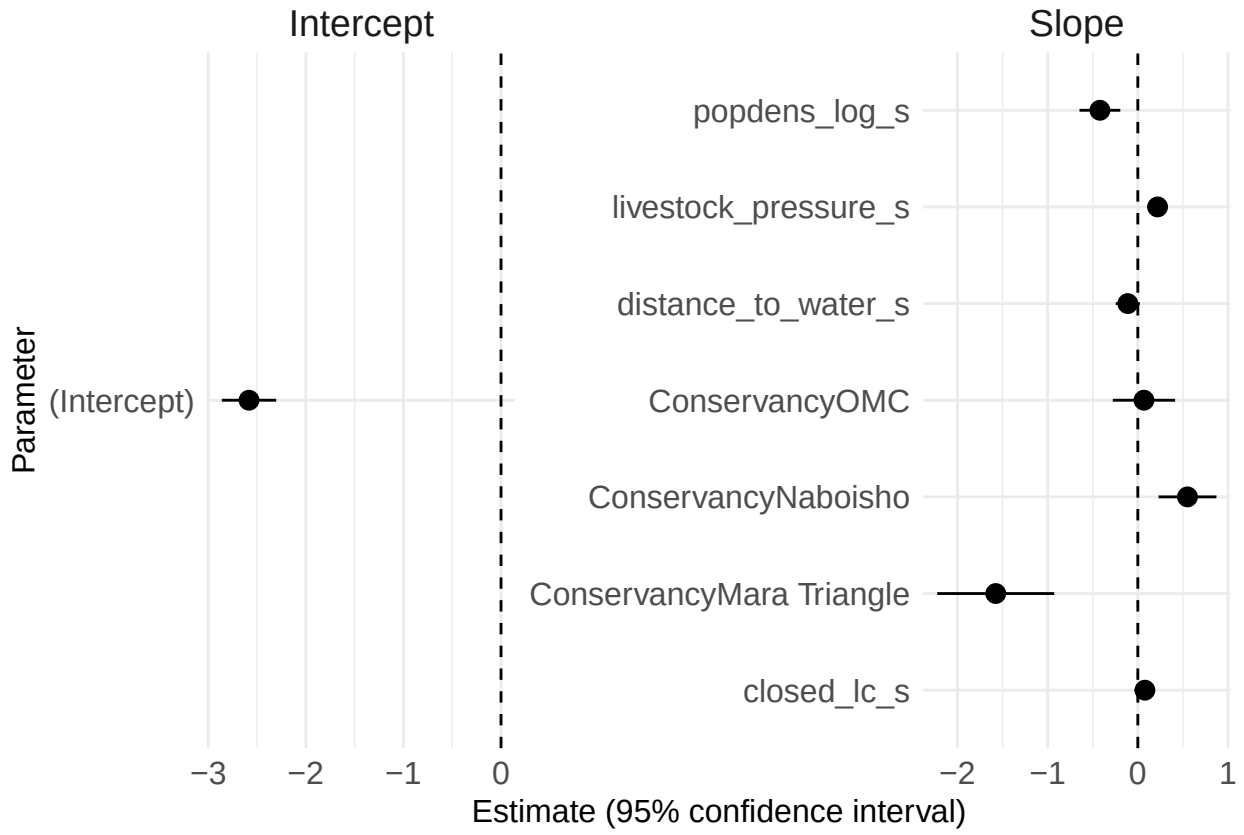
Explore the impact of the other covariates extracted above and included in your *locs* dataframe on hare occupancy. Use AIC and the *plot_fixed_effects* function to assess model fit and compare with models 1 & 2. Which covariates best explain the data using this modelling approach?

```
# include additional covariates
m3 = glm(prop_detected ~ livestock_pressure_s + Conservancy + distance_to_water_s + closed_lc_s + popde
         family=binomial(link="logit"),
         weights = n_days_sampled,
         data=locs)

# summary
summary(m3)
```

```
##
## Call:
## glm(formula = prop_detected ~ livestock_pressure_s + Conservancy +
##     distance_to_water_s + closed_lc_s + popdens_log_s, family = binomial(link = "logit"),
##     data = locs, weights = n_days_sampled)
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -2.58246    0.14201 -18.185  < 2e-16 ***
## livestock_pressure_s      0.21889    0.05423   4.036 5.43e-05 ***
```

```
## ConservancyMara Triangle -1.57577    0.33088  -4.762 1.91e-06 ***
## ConservancyNaboisho       0.55045    0.16390   3.358 0.000784 ***
## ConservancyOMC            0.06785    0.17612   0.385 0.700048
## distance_to_water_s      -0.11139    0.06825  -1.632 0.102680
## closed_lc_s               0.07849    0.05471   1.435 0.151423
## popdens_log_s            -0.42056    0.11535  -3.646 0.000267 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 943.31  on 174  degrees of freedom
## Residual deviance: 748.48  on 167  degrees of freedom
## AIC: 1021.4
##
## Number of Fisher Scoring iterations: 5
```

```
plotFixedEffects(m3)
```




```
AIC(m2);AIC(m3)
```

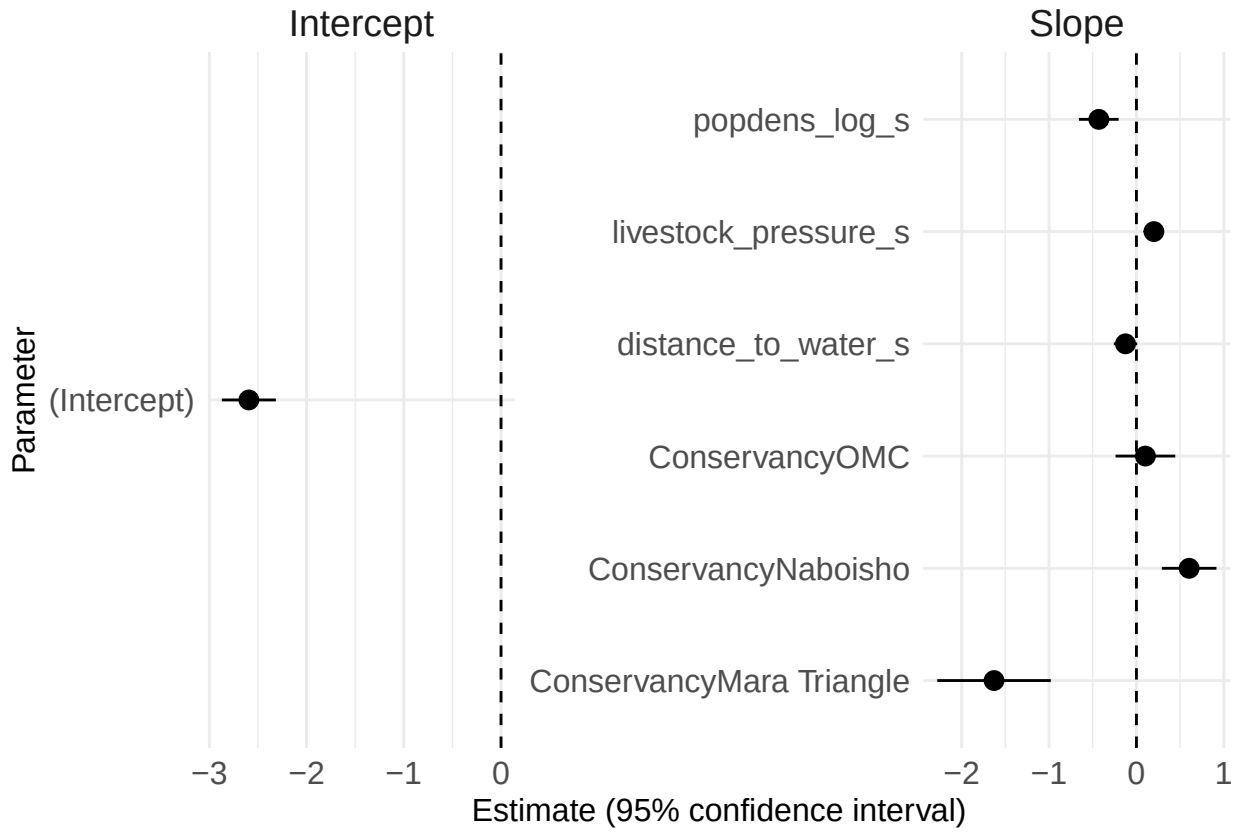
```
## [1] 1033.67
```

```
## [1] 1021.378
```

```r
m4 = glm(prop_detected ~ livestock_pressure_s + Conservancy + distance_to_water_s + popdens_log_s,
         family=binomial(link="logit"),
         weights = n_days_sampled,
         data=locs)

# summary
summary(m4)
```

```
##
## Call:
## glm(formula = prop_detected ~ livestock_pressure_s + Conservancy +
##     distance_to_water_s + popdens_log_s, family = binomial(link = "logit"),
##     data = locs, weights = n_days_sampled)
##
## Coefficients:
##                           Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -2.59337    0.14175 -18.295  < 2e-16 ***
## livestock_pressure_s      0.19990    0.05251   3.807 0.000141 ***
## ConservancyMara Triangle -1.62931    0.33132  -4.918 8.76e-07 ***
## ConservancyNaboisho       0.60312    0.15932   3.786 0.000153 ***
## ConservancyOMC            0.10267    0.17418   0.589 0.555565
## distance_to_water_s      -0.12504    0.06819  -1.834 0.066687 .
## popdens_log_s            -0.43028    0.11599  -3.710 0.000207 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 943.31  on 174  degrees of freedom
## Residual deviance: 750.52  on 168  degrees of freedom
## AIC: 1021.4
##
## Number of Fisher Scoring iterations: 5
```

```r
plotFixedEffects(m4)
```




```
AIC(m2);AIC(m3);AIC(m4)
```

```
## [1] 1033.67
```

```
## [1] 1021.378
```

```
## [1] 1021.42
```

Part 2 Extension exercise

Exercise 10

There are several other wildlife species included in the camera trap tagged images data frame. Try developing your own models and hypotheses based on these other species data and the available environmental and social covariates.

- How different are your findings for different species? Are there any consistent patterns?