ICCB Species distribution modelling

Scott Forrest and Charlotte Patterson

2025-06-14

In this script we are fitting species distribution models to the data of koalas (Phascolarctos cinereus) in the South-East Queensland (SEQ) region under current environmental conditions, which we will predict into future environmental conditions. Our modelling protocol follows these steps: 1. Load the koala presence data and environmental covariates. 2. Sample background points. 3. Select environmental covariates based on expert knowledge. 4. Exploration of the koala presence and background data. 5. Fit a Generalised Linear Model (GLM) to the data based on current climatic conditions. 7. Evaluate the model performance with spatial block cross validation. 8. Predict the model into future environmental conditions. 9. Plot model uncertainty.

Table of contents

We wrote this script drawing on some of the following resources:

- Ecocommons Notebooks https://www.ecocommons.org.au/notebooks/
- Damaris Zurell's SDM Intro https://damariszurell.github.io/SDM-Intro/

Import packages

```
# general R functions
library(tidyverse)
library(RColorBrewer)

# for downloading species data
library(galah)

# for general spatial operations
library(terra)
library(sf)
library(tidyterra)

# for SDM analyses
library(predicts)
library(blockCV)
library(ecospat)
library(usdm)
library(precrec)
library(corrplot)

# Set the random seed for reproducibility
set.seed(123)
```

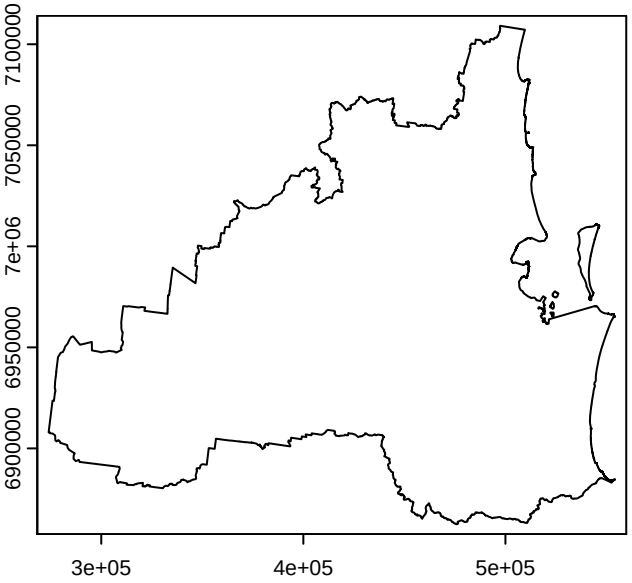
Load South East Queensland (SEQ) boundary

We start by defining our study area, which is the South East Queensland (SEQ) region. In a previous session we used the Local Government Areas (LGA) shapefile to define the extent of SEQ.

What we'll load is a polygon of our boundary which becomes our model domain.

For our coordinate reference system throughout, we'll be using GDA2020 / MGA zone 56, which is identified by the EPSG code 7856.

```
SEQ_extent.vect <- vect("Data/Environmental_variables/seq_boundary.gpkg")

# Define an sf object as well (used later)
SEQ_extent <- st_as_sf(SEQ_extent.vect,
                       coords = c("x", "y"),
                       crs = 7856)

plot(SEQ_extent.vect)
```



Step 1. Load the koala presence data and environmental covariates.

First, we're going to get some koala occurrence data. We will use the `galah` package to access the Atlas of Living Australia (ALA) data. The ALA is a great resource for Australian biodiversity data, and the `galah` package provides a convenient interface to access it.

We won't have time to explore these, but there are plenty of options for those using data outside of Australia. For example, the `rgbif` package provides access to the Global Biodiversity Information Facility (GBIF) data, and the `spocc` package provides access to a range of biodiversity data sources.

To cite the `galah` package

```
# Package citation
citation(package = "galah")
```

```
To cite galah in publications use:

  Westgate M, Kellie D, Stevenson M, Newman P (2025). _galah:
  Biodiversity Data from the GBIF Node Network_. R package version
  2.1.1, <https://CRAN.R-project.org/package=galah>.

A BibTeX entry for LaTeX users is

  @Manual{,
    title = {galah: Biodiversity Data from the GBIF Node Network},
    author = {Martin Westgate and Dax Kellie and Matilda Stevenson and Peggy Newman},
    year = {2025},
    note = {R package version 2.1.1},
    url = {https://CRAN.R-project.org/package=galah},
  }
```

Atlas of Living Australia using "galah" package

To access records from the ALA, you will need to be registered.

There are two registration options:

1. If you are affiliated with an Australian institution, you should be able to register for ALA through your institution. Find your institution in the list of institutions when you select 'Login' on the ALA website. Follow the prompts to enter your institution email and password. If your institution is not listed, you will need to register for an ALA account.
2. If you are not affiliated with an Australian institution, you will need to register for an ALA account. You can do this by selecting 'Register' on the ALA website. Follow the prompts to enter your email and password.

Once you're registered, you can set up your galah access configuration. This is a one-time setup that allows you to access the ALA data through the `galah` package. For now, we'll just load the pre-downloaded data, but you should use your own email address once you've registered for an ALA account.

```
galah_config(atlas = "ALA",
             email = "your@email.com", # replace with your email address
             password = "your_password" # replace with your password
             )
```

Download koala data from ALA

We're using a species name call with a state filter to identify all Presence-Only occurrence records of koala for Queensland.

To prevent calling from the ALA every time we run this script, we have saved the data to a CSV file. You can uncomment the code below to download the data again if you need to, or change it to your species and area of interest.

```
# koala_occurrences <- galah_call() %>%
#   galah_identify("Phascolarctos cinereus") %>%
#   galah_filter(
#     stateProvince == "Queensland",
#     occurrenceStatus == "PRESENT"
#   ) %>%
#   atlas_occurrences()

# save the csv
# write_csv(koala_occurrences, "Data/Biological_records/koala_occurrences.csv")

koala_occurrences <- read_csv("Data/Biological_records/koala_occurrences.csv")
```

The download using `galah` takes a minute or so, and we end up with a dataframe of 93456 records of koala for Queensland! You can see in the `dataResourceName` column that the data come from a variety of sources.

```
nrow(koala_occurrences)
```

```
[1] 93544
```

```
names(koala_occurrences)
```

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

```
head(koala_occurrences)
```

```
# A tibble: 6 x 8
  recordID        scientificName taxonConceptID decimalLatitude decimalLongitude
  <chr>           <chr>          <chr>                    <dbl>             <dbl>
1 00005e72-9aa0-~ Phascolarctos~ https://biodi~           -27.6              153.
2 00020a44-e8a1-~ Phascolarctos~ https://biodi~           -27.3              153.
3 00027b0c-4074-~ Phascolarctos~ https://biodi~           -27.3              153.
4 0002f47b-9da5-~ Phascolarctos~ https://biodi~           -27.5              153.
5 0003b563-803b-~ Phascolarctos~ https://biodi~           -19.1              147.
6 0003d509-b53a-~ Phascolarctos~ https://biodi~           -27.5              153.
# i 3 more variables: eventDate <dttm>, occurrenceStatus <chr>,
#   dataResourceName <chr>
```

Count the number of records per data resource

```
koala_occurrences %>% filter(occurrenceStatus == "PRESENT") %>%
  group_by(dataResourceName) %>%
  summarise(n = n()) %>%
  arrange(desc(n))
```

```
# A tibble: 42 x 2
   dataResourceName                                                    n
   <chr>                                                           <int>
 1 WildNet - Queensland Wildlife Data                              51920
 2 iNaturalist Australia                                           13789
 3 City of Gold Coast Koala Sightings                              12907
 4 Redlands Coast Koala Watch                                       4129
 5 Koala Count                                                      1827
 6 Koala Mapping Mackay, Whitsundays & Central Queensland Areas     1596
 7 OWAD Environment detection dogs Koala records                    1374
 8 ALA species sightings and OzAtlas                                1039
 9 Logan City Council Species Sightings                              841
10 Redland City Council Citizen Science Portal                       717
# i 32 more rows
```

Cleaning koala data

Not all data are created equal, and it's always important to spend some time examining and cleaning your species occurrence data before using it in a model. This is particularly relevant for us because we're using records collated into a database from a bunch of different sources. We don't have time to explore this today, but here's a few examples of things you might check at this stage and some functions / packages available to help:

- When downloading data from the ALA or directly from GBIF, you can filter for specific fields. For example, you can filter for records with a specific coordinate precision or quality, or remove those records older than a certain date. You can use the `galah_filter()` function for this.
- Check for duplicates in the data. You can use the `dplyr` package with functions like `distinct()`.
- Common issues like spatial outliers, taxonomic errors or coordinate errors are tested for with packages like: `spocc`, `CoordinateCleaner`, and `bdc`.
- Usually you would check that the CRS for the occurrences is the same as the shapefile. However, we know that the 'galah' package operates in WGS84.

```
# Check for missing values in decimalLongitude and decimalLatitude
paste0("Number of NAs in 'longitude' ", sum(is.na(koala_occurrences$decimalLongitude)))
```

```
[1] "Number of NAs in 'longitude' 215"
```

```
paste0("Number of NAs in 'latitude' ", sum(is.na(koala_occurrences$decimalLatitude)))
```

```
[1] "Number of NAs in 'latitude' 215"
```

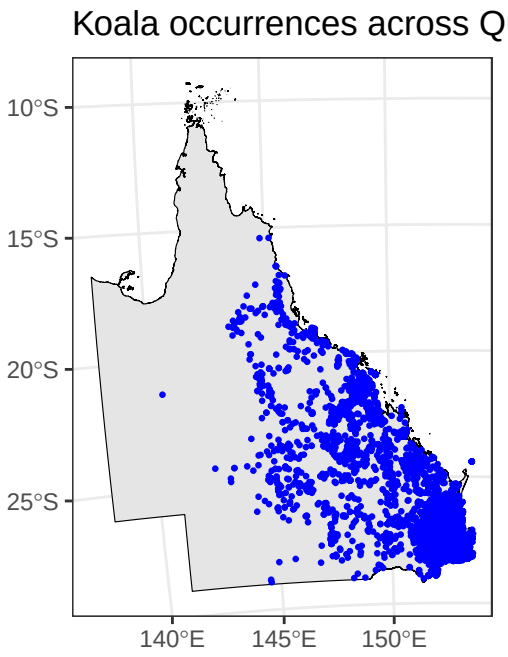
```
# Remove NAs and project data into the CRS that we want to use
koala_occ_sf <- koala_occurrences %>%
  tidyr::drop_na(decimalLongitude, decimalLatitude) %>% # remove NA values
  st_as_sf(coords = c("decimalLongitude", "decimalLatitude"),
           crs = 4326) %>%
  st_transform(7856) # Transform to the same CRS as SEQ_extent
```

Plot the koala data across Queensland

```
# Load a shapefile for Queensland just for plotting
qld_shp = st_read('Data/Environmental_variables/QLD_State_Mask.shp') %>%
  st_transform(7856) # Transform to the same CRS as SEQ_extent
```

```
Reading layer `QLD_State_Mask' from data source
  `/Users/scottforrest/Library/CloudStorage/OneDrive-QueenslandUniversityofTechnology/PhD -
  using driver `ESRI Shapefile'
Simple feature collection with 1 feature and 1 field
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: 137.9946 ymin: -29.17927 xmax: 153.5519 ymax: -9.219566
Geodetic CRS:  WGS 84
```

```
# Create a map using ggplot2
ggplot() +
  geom_sf(data = qld_shp, color = "black") +
  geom_sf(data = koala_occ_sf,
             color = "blue", size = 0.5) + # Add points for occurrences
  ggtitle("Koala occurrences across Queensland") +             # Add title
  theme_bw()
```



Filter by SEQ region

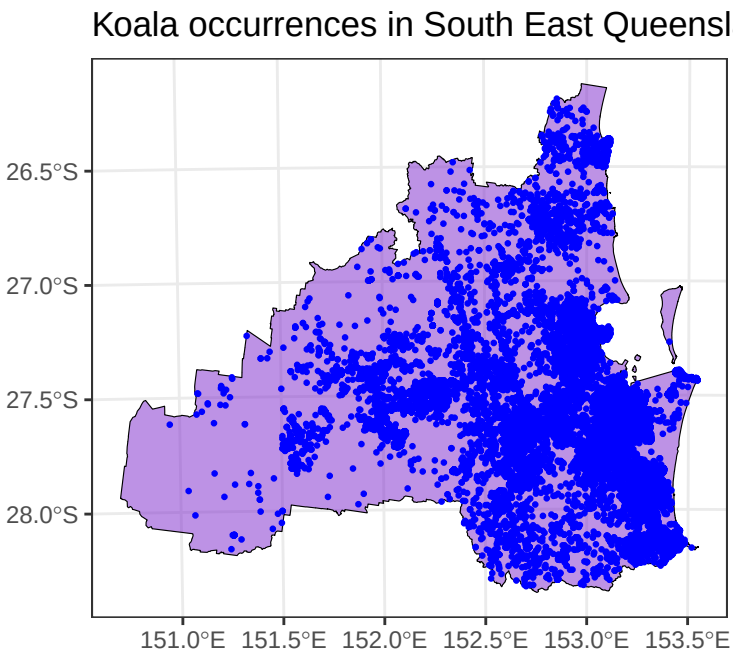
This takes a minute or two.

```
koala_occ_sf <- koala_occ_sf %>%
  st_intersection(SEQ_extent) %>% # Mask to extent
  distinct() # drop any duplicated records
```

Plot the filtered koala data

```
ggplot() +
  geom_sf(data = SEQ_extent, fill = "purple3", alpha = 0.5, color = "black", size = 0.2) +
  geom_sf(data = koala_occ_sf,                                  # Add koala presence location
          aes(geometry = geometry),
             color = "blue", size = 0.5) +                      # Add points for occurrences
```

```
  ggtitle("Koala occurrences in South East Queensland") +      # Add title
  theme_bw()
```



Step 2. Sample background points

Choosing background points to sample the availability of different environmental conditions is an important step in presence-only modelling. These points are contrasted against environmental conditions where your species was found (the presences) to help the model learn what conditions are suitable for the species. Background selection is a critical step in presence-only SDMs. Choices reflect your understanding of your study species. There's lots of good discussion about approaches to background selection in the literature, and we recommend reading some of these papers to understand the implications of your choices.

For this tutorial, we will use random sampling of background points across the SEQ region to keep it simple.

A few other approaches include:

- **Buffering**: Create a buffer around the presence points and sample points within that buffer. Figure from Velazco et al..

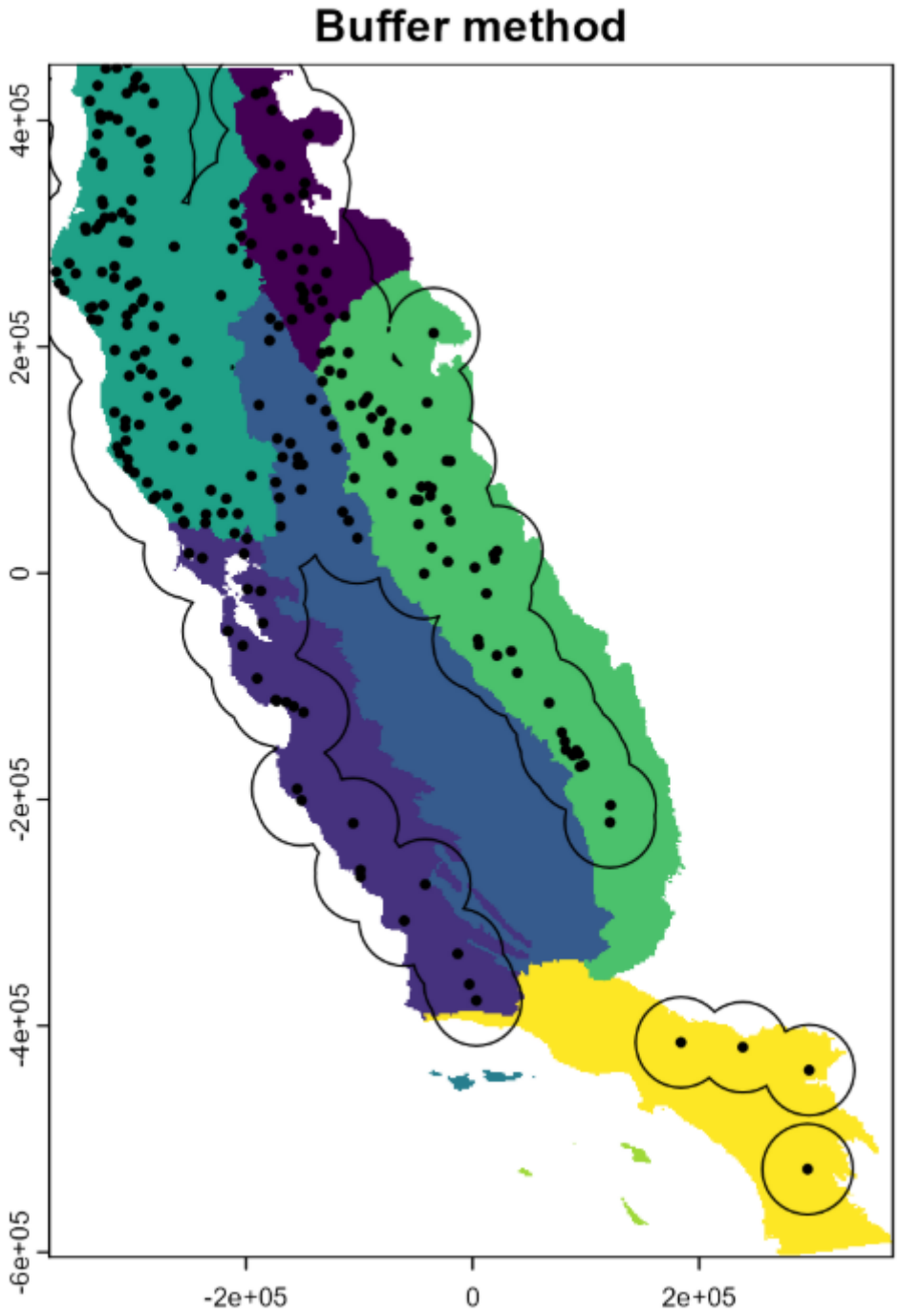


Figure 1: Buffered background locations

- **Minimum convex hull**: Create a minimum convex hull around the presence points and sample points within that hull. Figure from Velazco et al..

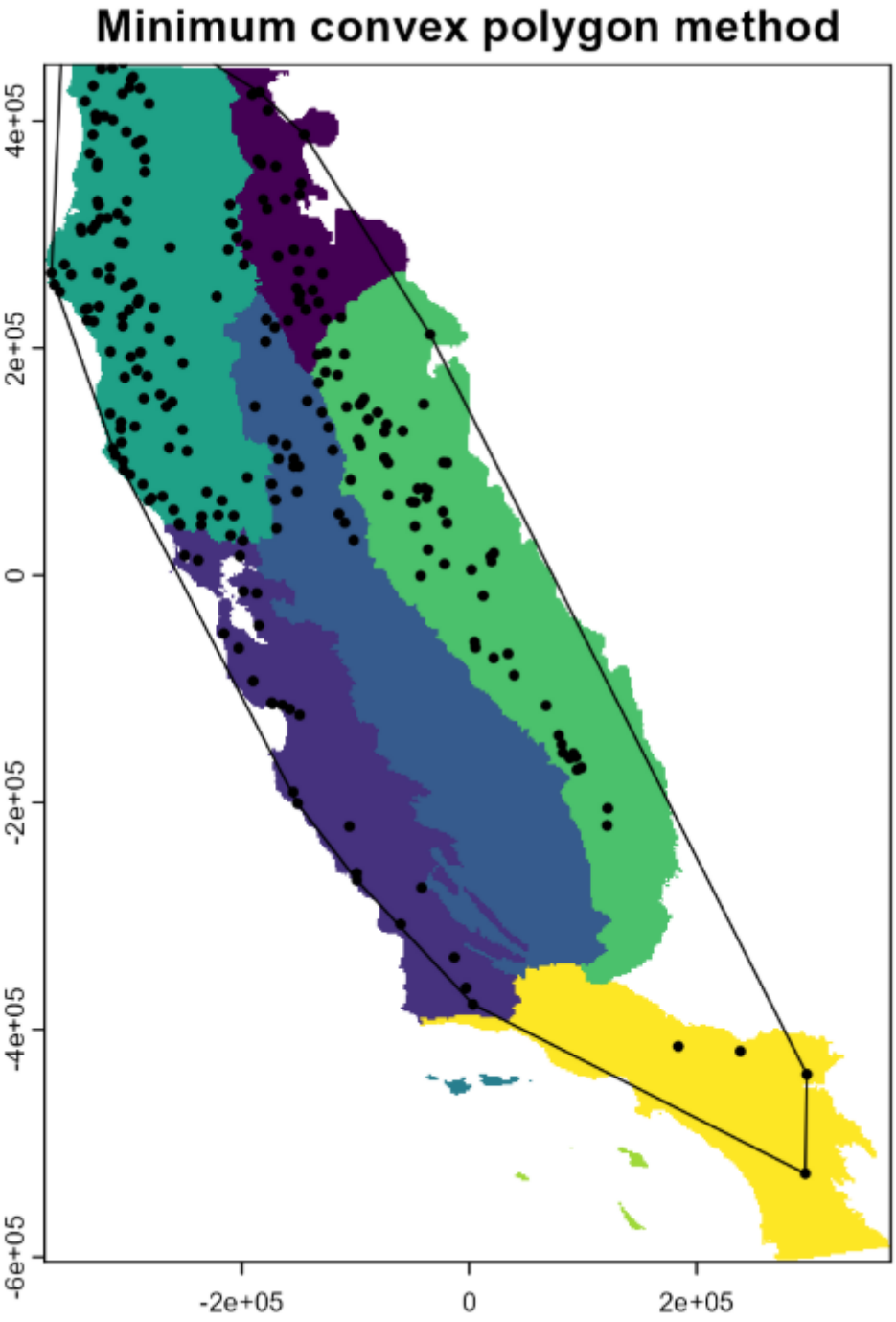


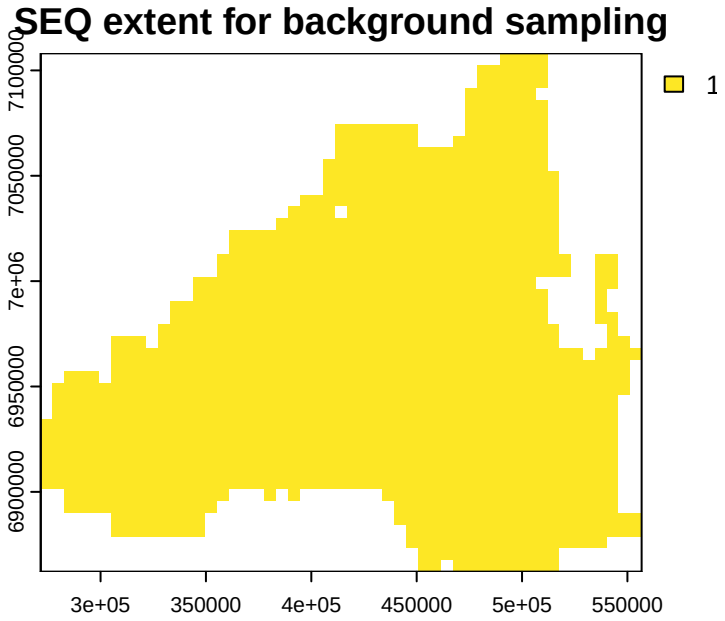
Figure 2: Minimum convex hull background locations

Random background sampling

We need to load a template grid for creating background points. This matches the grid our covariates are on, which we'll load soon.

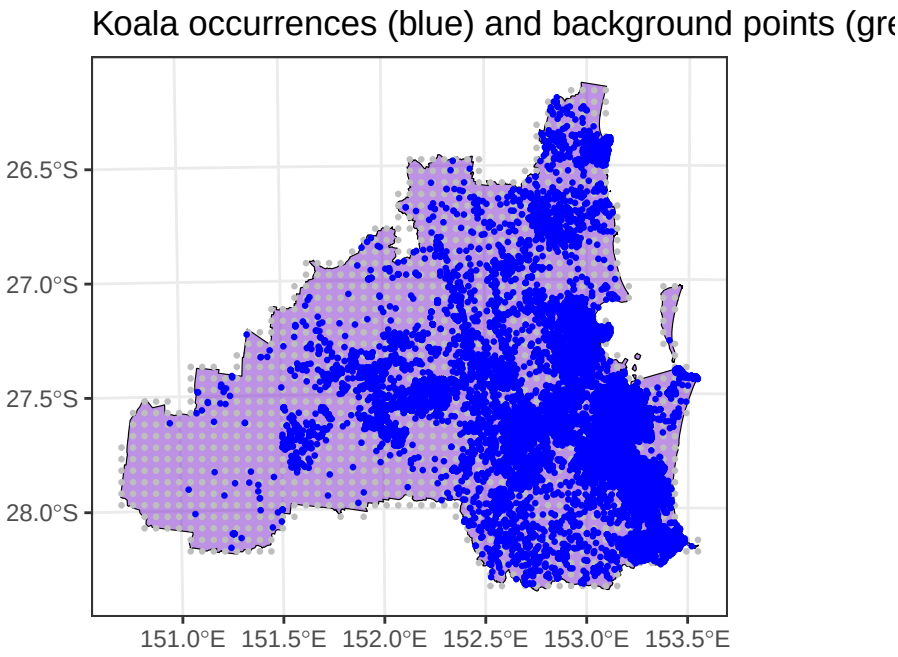
We're using a function from the `predicts` package for this.

```
domain <- rast("Data/Environmental_variables/SEQ_current_bioclim.tif")

# Make a blank raster of all 1s of the same dimensions as our covariate grid
domain <- domain[[1]]
domain <- ifel(is.na(domain), NA, 1) # Set all values to 1 except for NA values (which are o

names(domain) <- "SEQ_extent"

plot(domain, main = "SEQ extent for background sampling")
```




```
# Set the location and number of background points
# The mask means that any NA (not SEQ) locations do not include background points
background <- predicts::backgroundSample(mask = domain,
                                         n = 2500)

# Convert to terra SpatVector object
background <- terra::vect(background[,1:2], crs = "EPSG:7856")

# Convert background points (SpatVector) to data frame
background_df <- as.data.frame(geom(background))
```

```
koala_occ.vect <- vect(koala_occ_sf)

# Plot the presences (blue) and the background points (grey)
ggplot() +
  geom_sf(data = SEQ_extent, fill = "purple3", alpha = 0.5, color = "black", size = 0.2) +
  geom_spatvector(data = background,                              # Add koala presence location
             color = "gray", cex = 0.5) +                # Add points for occurrences
  geom_spatvector(data = koala_occ.vect,
                  aes(geometry = geometry),
                  color = "blue", cex = 0.5) + # Add background points
  ggtitle("Koala occurrences (blue) and background points (grey) in South East Queensland")
  theme_bw()
```



Combine koala presence and background points as dataframes

```
# Make a dataframe of just x, y and presence
koala_occ_df <- koala_occ_sf %>%
  dplyr::mutate(x = sf::st_coordinates(.)[,1],
                y = sf::st_coordinates(.)[,2]) %>%
                st_drop_geometry() %>%
  dplyr::select(x,y) %>%
  mutate(Presence = 1)

head(koala_occ_df)
```

```
# A tibble: 6 x 3
        x        y Presence
    <dbl>    <dbl>    <dbl>
1 522861. 6951030.        1
2 500099. 6982060.        1
3 495931. 6981622.        1
4 527834. 6955310.        1
5 510776. 6955199.        1
6 527933. 6955294.        1
```

```
background_df <- background %>%
  as.data.frame(geom = "XY") %>%
  dplyr::select(x,y) %>%
  mutate(Presence = 0)

head(background_df)
```

```
         x       y Presence
1 492450.8 7105286        0
2 498041.7 7105286        0
3 503632.6 7105286        0
4 509223.5 7105286        0
5 481268.9 7099695        0
6 486859.8 7099695        0
```

```
# Combine to one dataframe
pr_bg <- rbind(koala_occ_df, background_df)
```

Step 3. Environmental covariate selection

First, we load the rasters describing the current environmental conditions. We did some pre-formatting of these rasters so they match the koala data in projection and extent.

Layers were made available to us by the EcoCommons team and were created by Toombs and Ma (2025):

Toombs, N., and Ma S., 2025, A High-Resolution Dataset of 19 Bioclimatic Indices over Australia, Climate Projections and Services – Queensland Treasury, Brisbane, Queensland. [https://longpaddock.qld.gov.au/qld-future-climate/data-info/tern/]

```r
covs_current <- rast("Data/Environmental_variables/SEQ_current_bioclim.tif")


# Define the BIOCLIM names for the raster layers
layer_names <- c(
  "BIO1_Annual_Mean_Temp",
  "BIO2_Mean_Diurnal_Temp_Range",
  "BIO3_Isothermality",
  "BIO4_Temperature_Seasonality",
  "BIO5_Max_Temp_Warmest_Month",
  "BIO6_Min_Temp_Coldest_Month",
  "BIO7_Temperature_Annual_Range",
  "BIO8_Mean_Temp_Wettest_Quarter",
  "BIO9_Mean_Temp_Driest_Quarter",
  "BIO10_Mean_Temp_Warmest_Quarter",
  "BIO11_Mean_Temp_Coldest_Quarter",
  "BIO12_Annual_Precipitation",
  "BIO13_Precip_Wettest_Month",
  "BIO14_Precip_Driest_Month",
  "BIO15_Precip_Seasonality",
  "BIO16_Precip_Wettest_Quarter",
  "BIO17_Precip_Driest_Quarter",
  "BIO18_Precip_Warmest_Quarter",
  "BIO19_Precip_Coldest_Quarter")

names(covs_current) <- layer_names
```

One approach: Narrow down potential covariates based on ecological or expert knowledge

For this example, we had advice from scientists at CSIRO who conducted an expert elicitation to gather a set of potential covariates that are likely to be important for koalas.

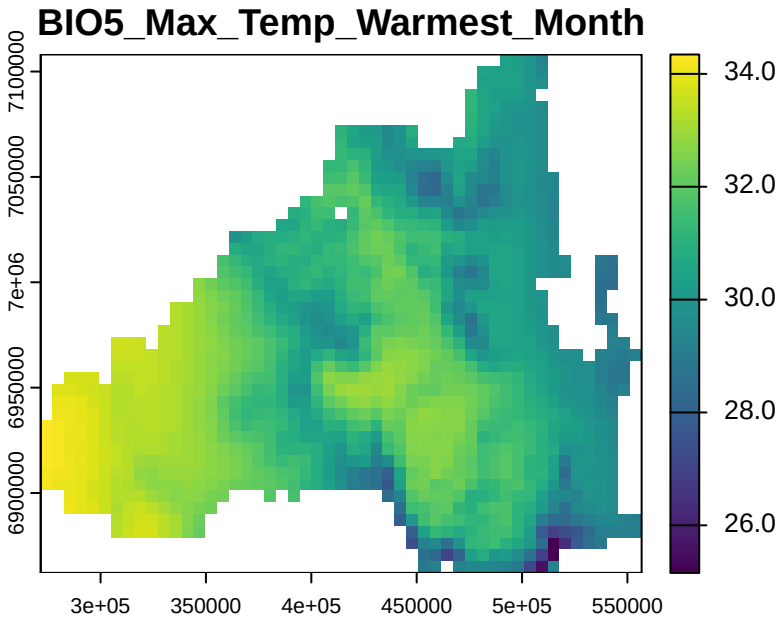
This is an example of one approach for deciding what covariates are candidates for inclusion in your model. We use this expert knowledge to filter out the key bioclim variables.

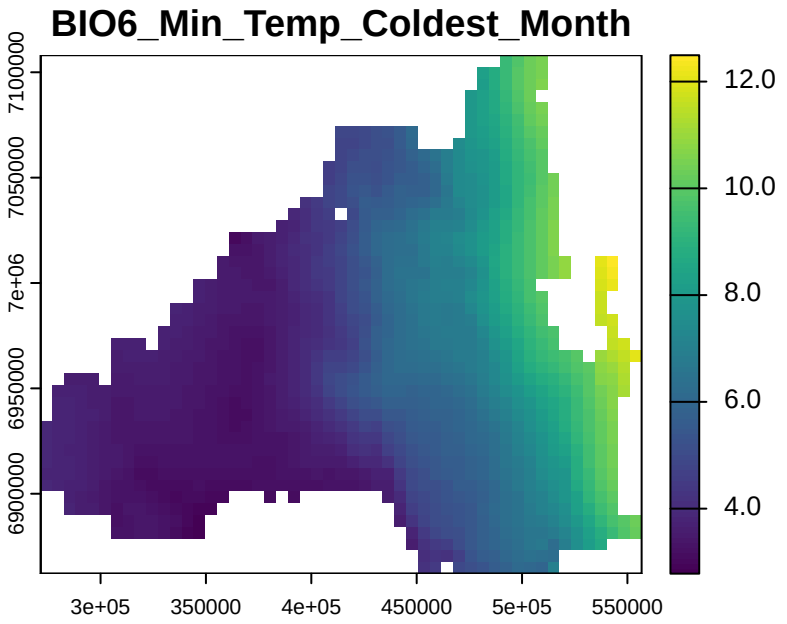
We select the following:

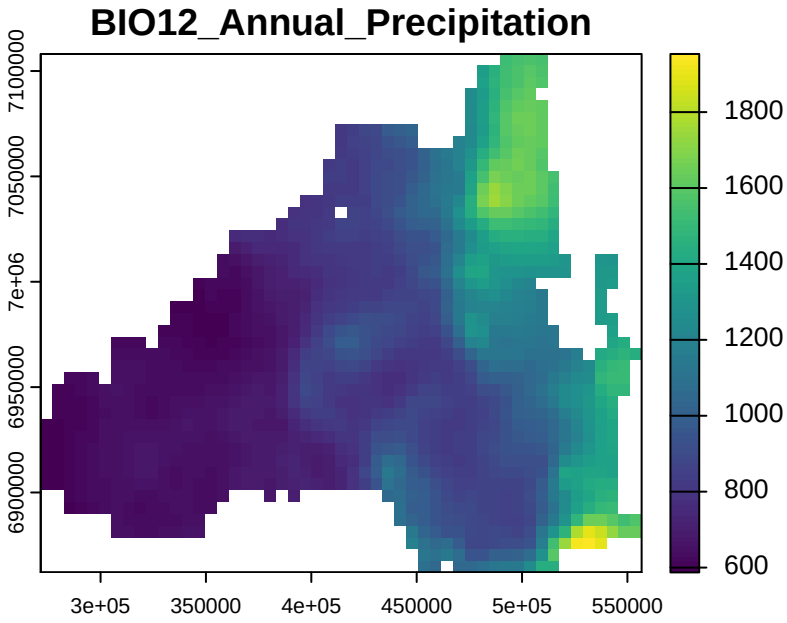
- Bio5 : Max temp of the warmest month (mainly for the northern populations)
- Bio6 : Min temp of the coldest month (mainly for southern populations, which essentially excludes alpine regions)
- Bio12 : Annual Precipitation
- Bio15 : Precipitation seasonality (coefficient of variation).

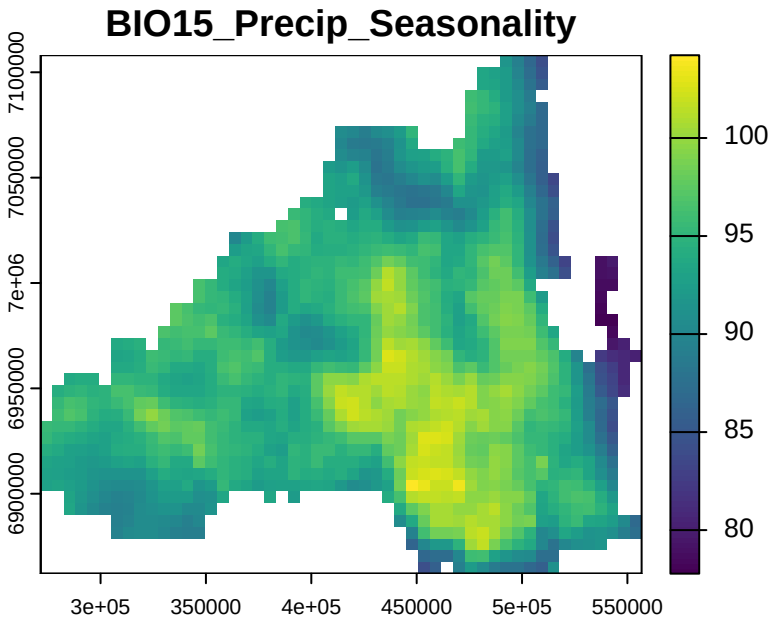
Plotting our four covariates from expert elicitation

```
# select only the covariates we are using by their names
covs_current_expert <- subset(covs_current,
                              names(covs_current) %in% c("BIO5_Max_Temp_Warmest_Month",
                                                         "BIO6_Min_Temp_Coldest_Month",
                                                         "BIO12_Annual_Precipitation",
                                                         "BIO15_Precip_Seasonality"))

for(i in 1:nlyr(covs_current_expert)) {
  plot(covs_current_expert[[i]], main = names(covs_current_expert)[[i]])
}
```




BIO6_Min_Temp_Coldest_Month
7100000
7050000
7e+06
6950000
6900000
3e+05
350000
4e+05
450000
5e+05
550000
12.0
10.0
8.0
6.0
4.0


BIO12_Annual_Precipitation
7100000
7050000
7e+06
6950000
6900000
3e+05
350000
4e+05
450000
5e+05
550000
1800
1600
1400
1200
1000
800
600



Covariate to account for sampling bias

As these data were collected from a wide number of sources, we have to consider whether any of the data, such as that which was opportunistically collected (Kéry et al. 2010), might be biased towards areas of higher human activity. This is a common issue in species occurrence data and is typically called sampling bias or preferential sampling, and occurs because areas that are more accessible and/or have higher human activity are more likely to be opportunistically sampled (Conn, Thorson, and Johnson 2017).

This means that if there are environmental variables that happen correlated with human activity (which is often the case, humans also select for favourable environmental conditions), it will appear that these variables are more influential than they actually are, because we associating that environmental covariate with 'more' presence records.

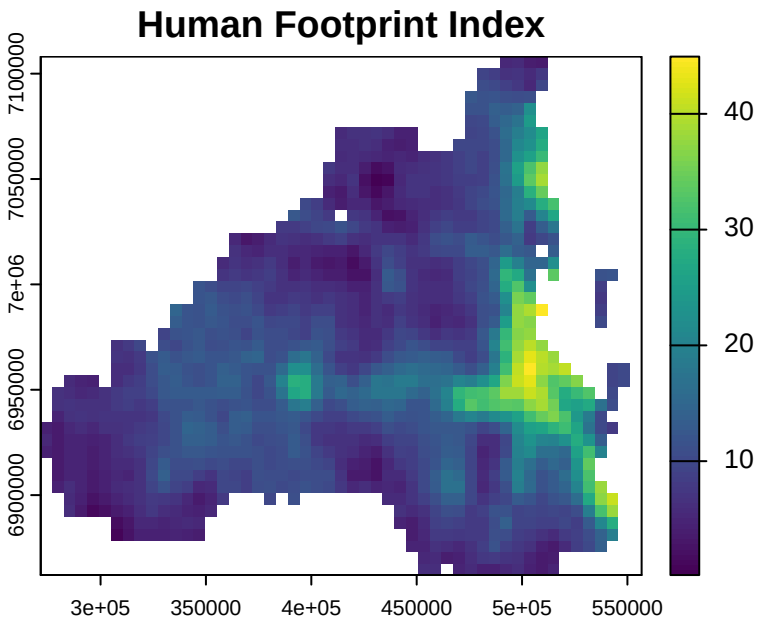
There are several different ways to account for this when modelling (Pennino et al. 2019; Mäkinen, Merow, and Jetz 2024; Dubos et al. 2022). Some of the common approaches include:

- Sampling pseudo-absences more points where there is higher sampling effort.
- Using a covariate to attempt to capture the sampling bias, such as human population density or distance to roads.
- Using a model or model structure that explicitly accounts for sampling bias, such as including a spatial random effect or using an integrated model to include presence-only and presence-absence data (Foster et al. 2024).

In our case, we will use a covariate for the Global Human Footprint Index (HFP) , which is a measure of human impact on the environment. This is a raster layer with continuous (percentage values) that we will use to account for sampling bias in our model.

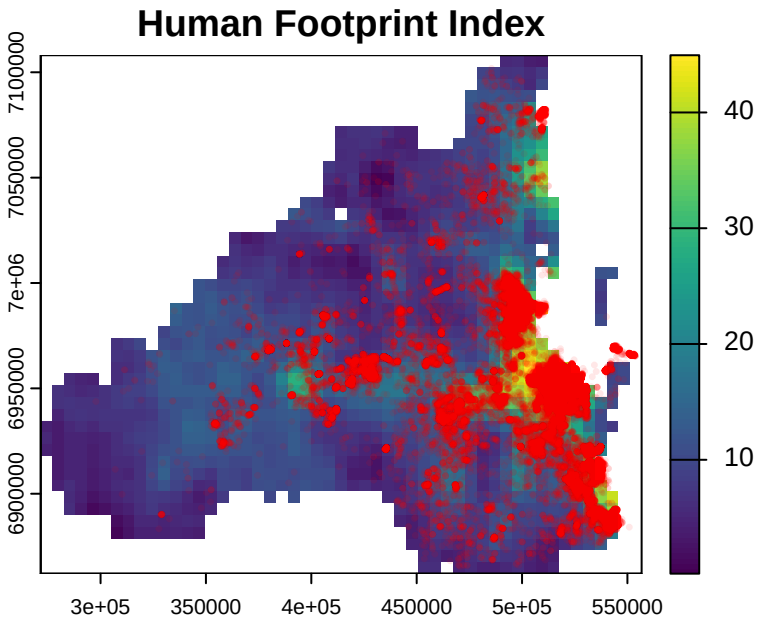
You will be able to see Brisbane as the largest area of human footprint, with Moreton Bay visible to the right, followed by the Gold Coast to the south, the Sunshine Coast to the north, and Toowoomba to the west (about 2 hours drive).

```
human_footprint <- rast("Data/Environmental_variables/cost_hfp2013.tif")
names(human_footprint) <- "human_footprint"

# Update CRS to EPSG:7856 (GDA2020 / MGA zone 56)
human_footprint <- terra::project(human_footprint, "EPSG:7856")

# Resample to match the extent and resolution of the covariates
human_footprint <- resample(human_footprint, covs_current_expert, method = "bilinear")

# Mask to SEQ extent
human_footprint <- terra::mask(human_footprint, covs_current_expert[[1]])

plot(human_footprint, main = "Human Footprint Index")
```



Although this isn't a formal comparison, we can see that this variable is correlated with the presence of koalas.

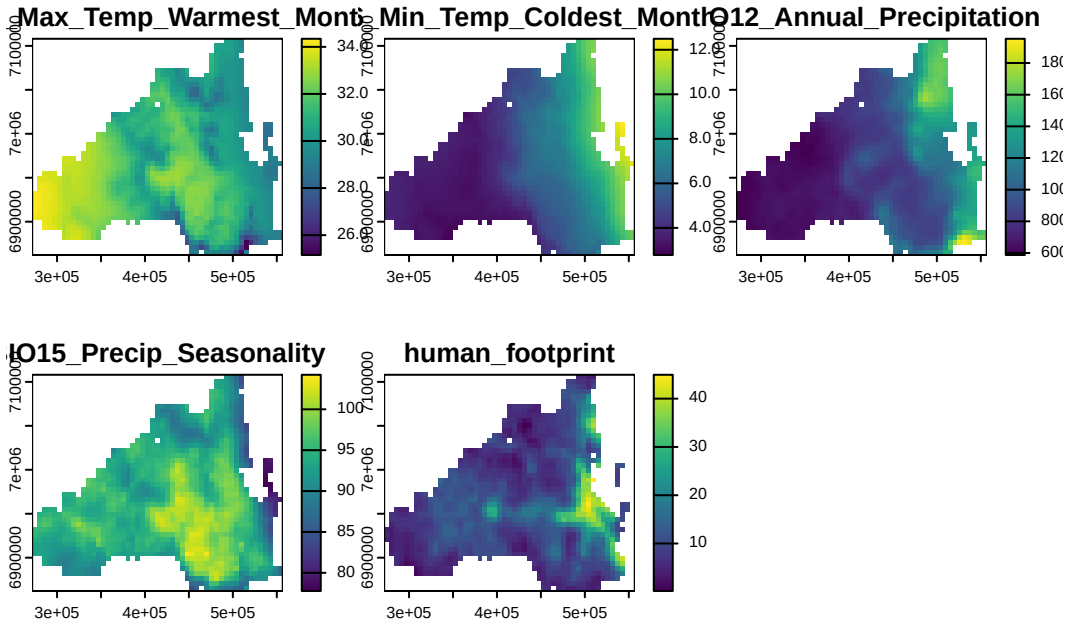
```
plot(human_footprint, main = "Human Footprint Index")
points(koala_occ_sf, col = "red", alpha = 0.1, pch = 20, cex = 0.5)
```



You would want to put some more thought into your own study and the best way to account for sampling bias, but in our case we will try using the human footprint index as a covariate in our model, and assess how our predictions change.

Add the human footprint layer to our covariates

```
covs_current_expert_HF <- c(covs_current_expert, human_footprint)
plot(covs_current_expert_HF)
```



Extract environmental covariate values from presence and background locations (training locations)

This will create a dataframe where each row is a koala presence or background location and each column is a value for our four bioclimatic variables at that location.

We use a function from `terra` for this.

```
# this will give us the covariate values for each presence and background point
PB_covs <- terra::extract(
  x = covs_current_expert_HF,   # Raster with environmental variables
  y = pr_bg[, c("x", "y")],  # Dataframe with x, y and presence
  ID = FALSE                 # Don't return an ID column for each location
)

train_PB_covs <- cbind(pr_bg, PB_covs) # Combine the presence column with the covariate valu

# drop any NAs
train_PB_covs <- train_PB_covs %>% drop_na()

head(train_PB_covs)
```

```
         x       y Presence BIO5_Max_Temp_Warmest_Month
1 522861.4 6951030        1                     29.86404
2 500098.7 6982060        1                     30.44493
3 495930.5 6981622        1                     30.44493
4 527833.8 6955310        1                     29.71574
5 510776.4 6955199        1                     30.43131
6 527933.4 6955294        1                     29.71574
  BIO6_Min_Temp_Coldest_Month BIO12_Annual_Precipitation
1                    9.045586                    1231.603
2                    8.930631                    1174.339
3                    8.930631                    1174.339
4                    9.618237                    1231.261
5                    8.725674                    1076.065
6                    9.618237                    1231.261
  BIO15_Precip_Seasonality human_footprint
1                 95.31358        27.52487
2                 99.00175        37.59798
3                 99.00175        37.59798
4                 91.62077        36.49710
5                 98.15495        37.08822
6                 91.62077        36.49710
```

Thin the koala presence points (for tutorial only)

We now thin the presences to reduce the number of points to a manageable size for plotting and modelling. *This is just for the purpose of this tutorial*, and is done here to make the tutorial run faster and to make the plots clearer. There are some reasons you would want to thin (such as bias correction), but you would want to carefully consider whether this is appropriate for your modelling.

We only thin the presence points as the background points are already limited by the number of cells in the SEQ region.

```
train_PB_covs_pres <- train_PB_covs %>% filter(Presence == 1)
train_PB_covs_bg <- train_PB_covs %>% filter(Presence == 0)

# Thin the presences - 10000 presence points
train_PB_covs_pres_thinned <- train_PB_covs_pres[sample(nrow(train_PB_covs_pres), 10000), ]

# Combine back into both presence and background
train_PB_covs_thinned <- rbind(train_PB_covs_pres_thinned, train_PB_covs_bg, make.row.names
```

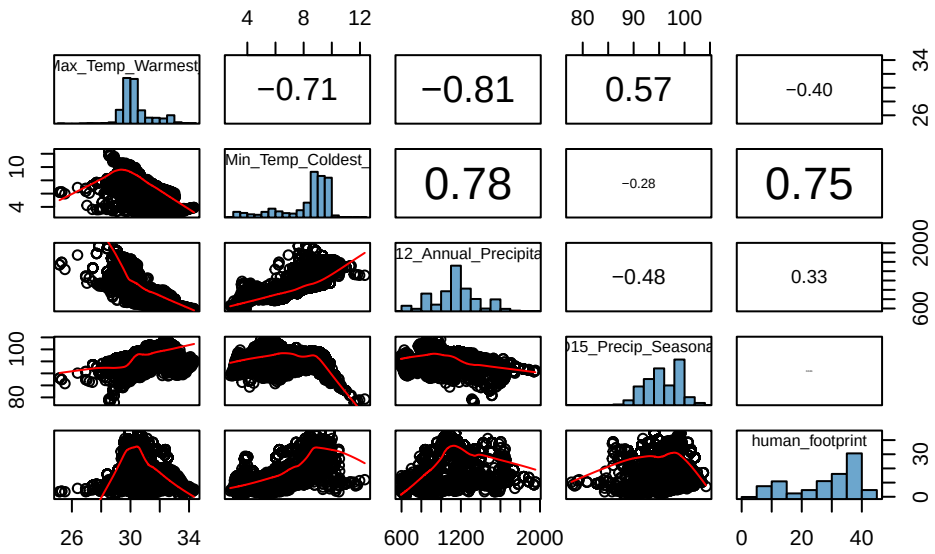
Check correlation and multicollinearity of covariates

Although we've narrowed down our covariates already based on discussion with experts, there's a chance that our remaining variables might be correlated or collinear. This would cause issues with models, particularly because we're hoping to predict our model into the future. We're going to inspect our covariates for signs of correlation and multicolinearity.

There are several different methods for creating correlation plots, we just show two.

Correlation plot from the ecospat package

```
# use the dataframe we just created, dropping the x, y, and presence columns (1-3)
ecospat::ecospat.cor.plot(train_PB_covs_thinned[, -1:-3])
```



What the correlation test plot suggests is that we have some correlated covariates. Particularly our Bio6 and Bio12.

Often a rule of thumb of ~ 0.7 correlation is used to decide what a 'too correlated' covariate pair are (REF). In practice, it's important to think carefully before dropping covariates.

Variance Inflation Factor (VIF)

If you find corrplot is hard for you to use to make decisions, we can also look at the Variance Inflation Factor (VIF). VIF is another statistical measure used to detect multicollinearity in a set of explanatory (independent) variables in a regression model.

Interpretation:

- VIF = 1: No correlation
- VIF > 1 and <= 5: Moderate correlation; may not require corrective action.
- VIF > 5: Indicates high correlation. Multicollinearity may be problematic, and further investigation is recommended.
- VIF > 10: Strong multicollinearity. The variable is highly collinear with others, and steps should be taken to address this.

```
usdm::vifstep(covs_current_expert_HF) # Variance Inflation Factor and test for multicollinea
```

```
No variable from the 5 input variables has collinearity problem.

The linear correlation coefficients ranges between:
min correlation ( BIO15_Precip_Seasonality ~ BIO6_Min_Temp_Coldest_Month ):  -0.1420642
max correlation ( BIO12_Annual_Precipitation ~ BIO6_Min_Temp_Coldest_Month ):  0.8762638
```

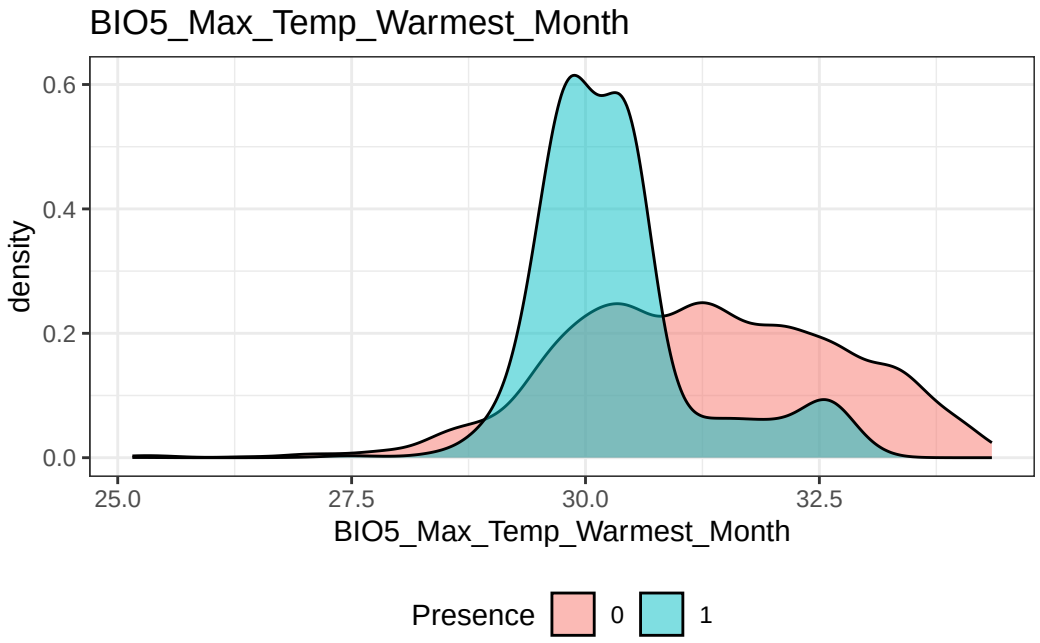
```
---------- VIFs of the remained variables --------
                   Variables      VIF
1 BIO5_Max_Temp_Warmest_Month 2.658798
2 BIO6_Min_Temp_Coldest_Month 5.916991
3  BIO12_Annual_Precipitation 7.552088
4    BIO15_Precip_Seasonality 1.330505
5             human_footprint 1.551557
```

Step 4. Exploration of the koala presence and background data

It is good practice to assess where in the environmental space the presence and background points are located. This can help to identify if there are any potential issues with the data, such as a lack of background points in certain areas of environmental space. It's also a good way to have a first look at any patterns in the species data and the environmental covariates.

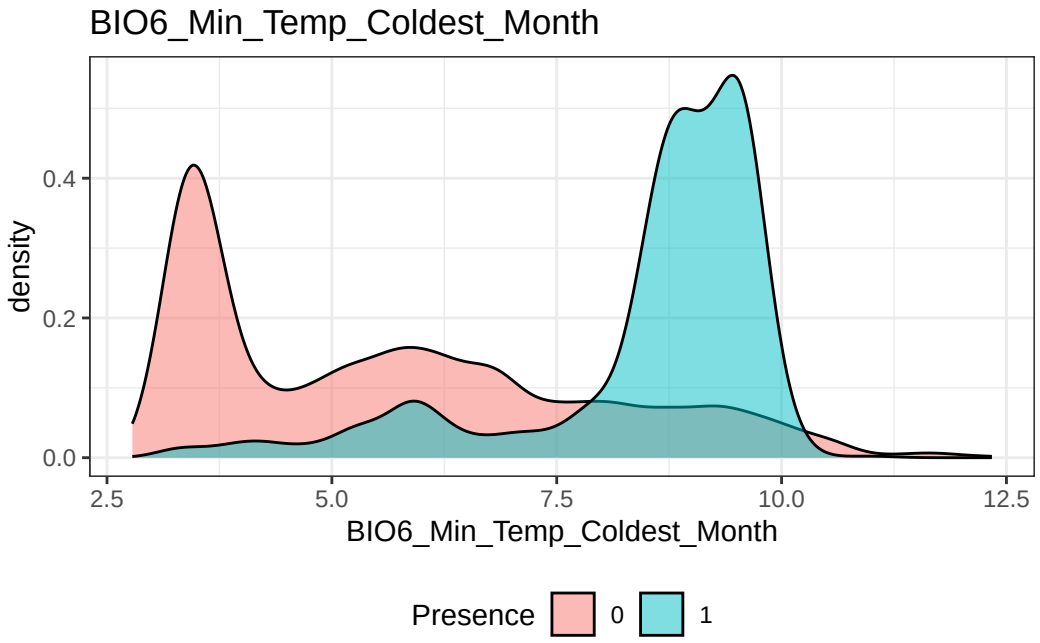
Bio5 Max. temperature warmest month

```
ggplot() +
  geom_density(data = train_PB_covs_thinned,
               aes(x = .data[["BIO5_Max_Temp_Warmest_Month"]], fill = as.factor(Presence)),
               alpha = 0.5,
               bw = 0.25) + # we can try different bandwidth values to smooth the densities
  theme_bw() +
  labs(title = "BIO5_Max_Temp_Warmest_Month",
       fill = "Presence") +
  theme(legend.position = "bottom")
```



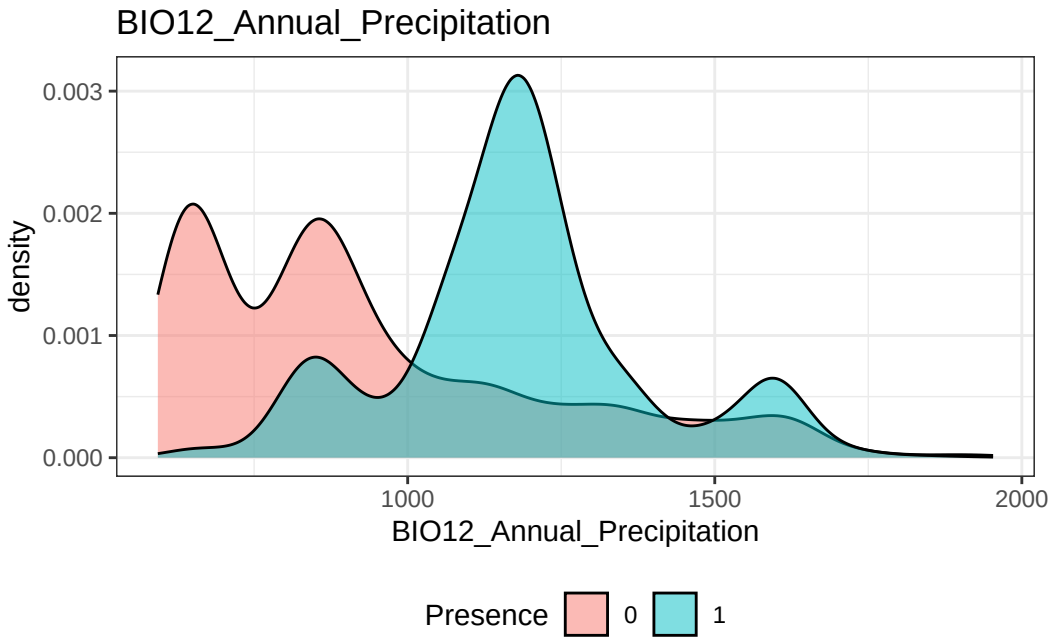
Bio6 Min. temperature coldest month

```
ggplot() +
          geom_density(data = train_PB_covs_thinned,
                       aes(x = .data[["BIO6_Min_Temp_Coldest_Month"]], fill = as.factor(Pres
               alpha = 0.5,
               bw = 0.25) + # we can try different bandwidth values to smooth the densities
    theme_bw() +
    labs(title = "BIO6_Min_Temp_Coldest_Month",
       fill = "Presence") +
  theme(legend.position = "bottom")
```



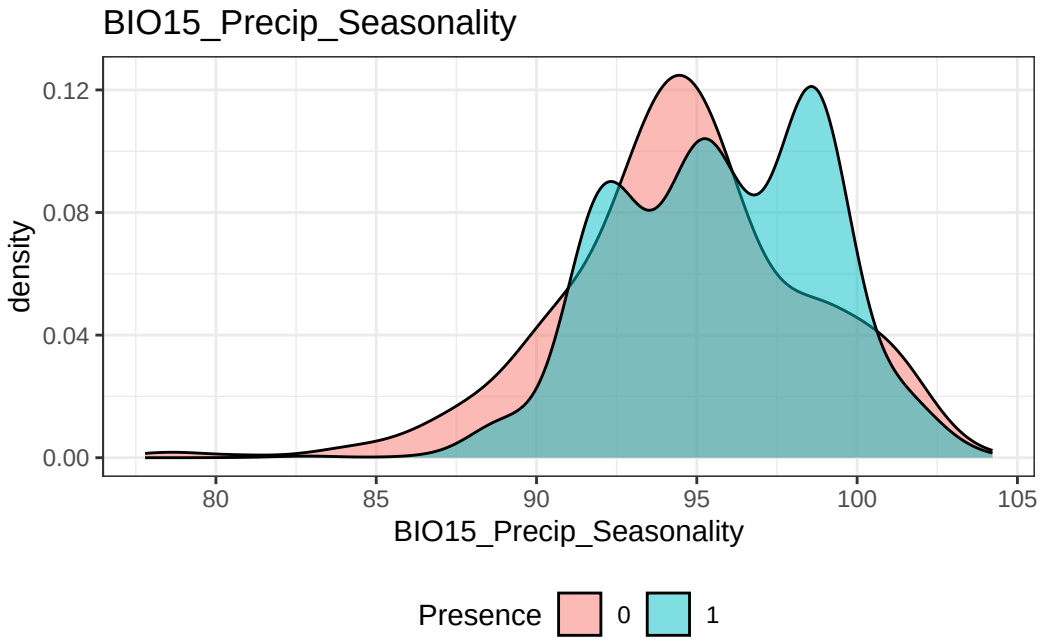
Bio12 Annual precipitation

```
ggplot() +
          geom_density(data = train_PB_covs_thinned,
                       aes(x = .data[["BIO12_Annual_Precipitation"]], fill = as.factor(Prese
               alpha = 0.5,
               bw = 50) + # we can try different bandwidth values to smooth the densities
    theme_bw() +
    labs(title = "BIO12_Annual_Precipitation",
       fill = "Presence") +
  theme(legend.position = "bottom")
```



Bio15 Precipitation seasonality

```
ggplot() +
          geom_density(data = train_PB_covs_thinned,
                       aes(x = .data[["BIO15_Precip_Seasonality"]], fill = as.factor(Presenc
               alpha = 0.5,
               bw = 1) + # we can try different bandwidth values to smooth the densities
    theme_bw() +
    labs(title = "BIO15_Precip_Seasonality",
       fill = "Presence") +
  theme(legend.position = "bottom")
```



Scaling the covariates

It is common to scale the covariates before fitting a model. This is because some models, such as Generalised Linear Models (GLMs), can be sensitive to the scale of the covariates. Scaling the covariates can also make the coefficients more comparable, as they are now on a similar scale to each other.

A typical scaling approach is to subtract the mean and divide by the standard deviation of each covariate, which is also known as z-scaling or z-scoring. After this operation, all covariates will have a mean of 0 and a standard deviation of 1.

We can use a base R function to scale the covariates, and the default is to centre (subtract the mean) and scale (divide by the standard deviation).

```
# pull out just the covariate values from the dataframe
covariate_values <- train_PB_covs_thinned[, -1:-3]

# scale the covariates - returns a matrix
scaled_covariates <- base::scale(covariate_values,
                                 center = TRUE,
                                 scale = TRUE)

# convert back to a dataframe (and remove row names)
scaled_covariates_df <- data.frame(scaled_covariates)

head(scaled_covariates_df)
```

```
  BIO5_Max_Temp_Warmest_Month BIO6_Min_Temp_Coldest_Month
1                  -0.4240962                   0.7580190
2                   2.2654126                  -1.3338958
3                  -0.2564907                   0.2921544
4                  -0.5739359                   0.7246740
5                  -0.5739359                   0.7246740
6                  -0.8459409                   0.9088325
  BIO12_Annual_Precipitation BIO15_Precip_Seasonality human_footprint
1                 0.64547060              -1.31413549       0.3263228
2                -1.35466130               1.71903964      -1.1614318
3                 0.06260560               0.04455667       0.4769760
4                -0.08143464              -0.24231847       1.1427425
5                -0.08143464              -0.24231847       1.1427425
6                 1.55487796              -0.76480060       0.9352724
```

```r
# Add the presence column back to the scaled covariates
train_PB_covs_thinned_scaled <- cbind(train_PB_covs_thinned[, 1:3], scaled_covariates_df)
```

The scaling operation also returns the particular scaling values that were used, which is helpful for generating predictions later on. We can access these using the attributes of the returned dataframe. We only need the scaling factor, as we will rescale the coefficients before generating predictions.

```r
# Get the attributes of the scaled covariates
scaled_attributes <- attributes(scaled_covariates)
scaled_attributes$`scaled:scale` # the SD/scaling factor of each covariate
```

```
BIO5_Max_Temp_Warmest_Month BIO6_Min_Temp_Coldest_Month
                  0.9930861                   1.7614327
 BIO12_Annual_Precipitation    BIO15_Precip_Seasonality
                230.8653893                   3.2804452
            human_footprint
                 11.1656717
```

Step 5. Fit a model: A generalised linear model (GLM)

```r
# Make a folder to save outputs
dir.create("Outputs/GLM_outputs", showWarnings = F)
```

Null model

Null model: no explanatory variables or predictors are included.

It is always helpful to create a null model as a benchmark to assess how the inclusion of explanatory variables improves the model.

```
# Fit a null model with only the intercept
null_model <- glm(Presence ~ 1,
                  data = train_PB_covs_thinned_scaled,
                  family = binomial(link = "logit"))

# Check the model results
summary(null_model)
```

```
Call:
glm(formula = Presence ~ 1, family = binomial(link = "logit"),
    data = train_PB_covs_thinned_scaled)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  2.10046    0.03028   69.37   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 7734  on 11223  degrees of freedom
Residual deviance: 7734  on 11223  degrees of freedom
AIC: 7736

Number of Fisher Scoring iterations: 4
```

GLM - expert variables with linear terms

In this model, we include linear terms for the covariates. You can also do things like add quadratic terms to account for non-linear relationships between the predictors and the response variable. This increases the complexity of the model and allows for more flexibility in fitting the data.

You can also try fitting the model with and without the scaled covariates. The z- and p-values of each covariate should be the same (but not for the intercept and centering the covariates affects that), as well as the AIC.

```r
glm_model <- glm(Presence ~
                   BIO5_Max_Temp_Warmest_Month +
                   BIO6_Min_Temp_Coldest_Month +
                   BIO12_Annual_Precipitation +
                   BIO15_Precip_Seasonality +
                   human_footprint,
                 data=train_PB_covs_thinned_scaled,
                 family = binomial(link = "logit"))

# Check the model results
summary(glm_model)
```

```
Call:
glm(formula = Presence ~ BIO5_Max_Temp_Warmest_Month + BIO6_Min_Temp_Coldest_Month +
    BIO12_Annual_Precipitation + BIO15_Precip_Seasonality + human_footprint,
    family = binomial(link = "logit"), data = train_PB_covs_thinned_scaled)

Coefficients:
                            Estimate Std. Error z value Pr(>|z|)
(Intercept)                  3.50134    0.07444  47.033  < 2e-16 ***
BIO5_Max_Temp_Warmest_Month -0.47195    0.05629  -8.385  < 2e-16 ***
BIO6_Min_Temp_Coldest_Month  0.36938    0.07578   4.874 1.09e-06 ***
BIO12_Annual_Precipitation  -0.04255    0.07955  -0.535    0.593
BIO15_Precip_Seasonality     0.74946    0.04546  16.486  < 2e-16 ***
human_footprint              1.41789    0.06829  20.764  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 7734.0  on 11223  degrees of freedom
Residual deviance: 4454.4  on 11218  degrees of freedom
AIC: 4466.4

Number of Fisher Scoring iterations: 7
```

Model effect evaluation

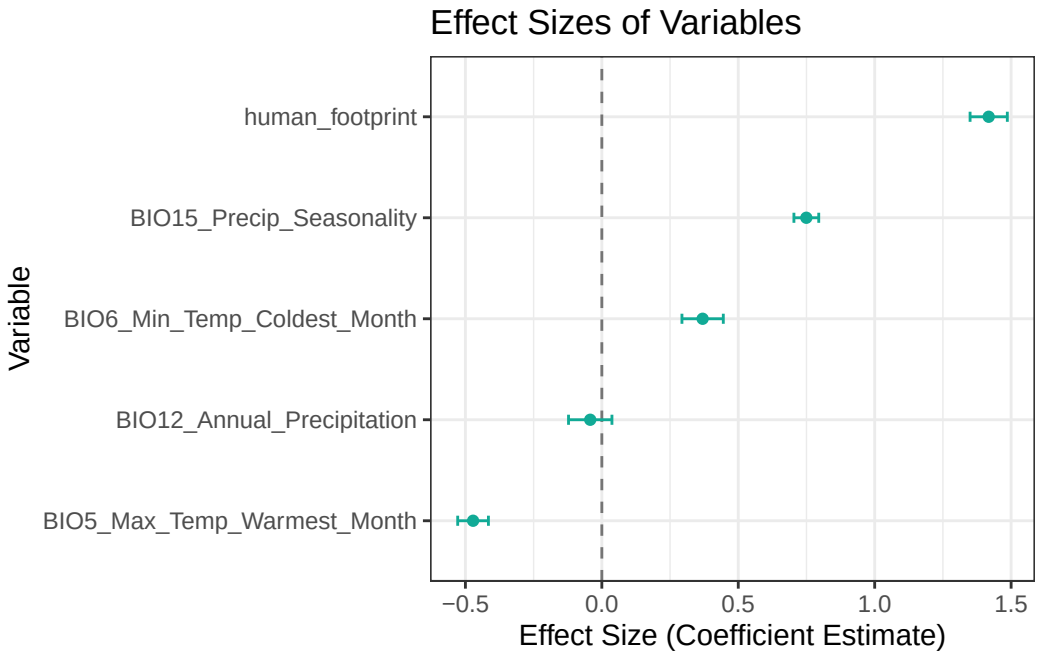
Here we use a function presented in an EcoCommons Australia notebook to evaluate the model performance. The notebook can be found on their GitHub: https://github.com/EcoCommonsAustralia/n

```r
# Function to plot effect size graph
plot_effect_size <- function(glm_model) {
  # Check if required libraries are installed
  if (!requireNamespace("ggplot2", quietly = TRUE)) {
    stop("Please install the 'ggplot2' package to use this function.")
  }
  library(ggplot2)

  # Extract effect sizes (coefficients) from the model
  coefs <- summary(glm_model)$coefficients
  effect_sizes <- data.frame(
    Variable = rownames(coefs)[-1],  # Exclude the intercept
    Effect_Size = coefs[-1, "Estimate"],
    Std_Error = coefs[-1, "Std. Error"]
  )

  # Sort by effect size
  effect_sizes <- effect_sizes[order(-abs(effect_sizes$Effect_Size)), ]

  # Plot the effect sizes with error bars
  ggplot(effect_sizes, aes(x = reorder(Variable, Effect_Size), y = Effect_Size)) +
    geom_hline(yintercept = 0, linetype = "dashed", color = "black", alpha = 0.5) +
    geom_point(stat = "identity", colour = "#11aa96") +
    geom_errorbar(aes(ymin = Effect_Size - Std_Error, ymax = Effect_Size + Std_Error),
                  colour = "#11aa96", width = 0.1) +
    coord_flip() +
    labs(
      title = "Effect Sizes of Variables",
      x = "Variable",
      y = "Effect Size (Coefficient Estimate)"
    ) +
    theme_bw()
}
```

Run the function on the covariates used in the model.

Interestingly, the human footprint index has the largest effect size, followed by the annual precipitation and the minimum temperature of the coldest month, both of which were positive. The maximum temperature of the warmest month has a negative coefficient, which suggests that koalas are less likely to be found in areas with higher temperatures during the warmest month.

```r
plot_effect_size(glm_model)
```



Response curves

Again, we can use a function from the EcoCommons notebook to plot the response curves from the model.

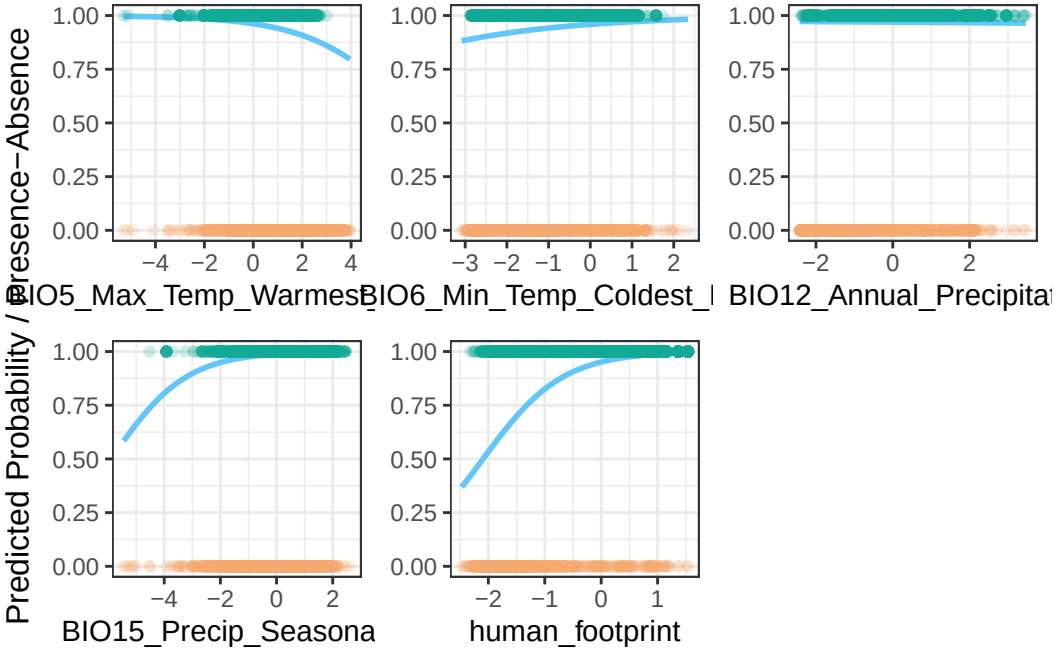
```
plot_species_response <- function(glm_model, predictors, data) {
  # Check if required libraries are installed
  if (!requireNamespace("ggplot2", quietly = TRUE) || !requireNamespace("gridExtra", quietly
    stop("Please install the 'ggplot2' and 'gridExtra' packages to use this function.")
  }
  library(ggplot2)
  library(gridExtra)

  # Create empty list to store response plots
  response_plots <- list()

  # Loop through each predictor variable
  for (predictor in predictors) {
    # Create new data frame to vary predictor while keeping others constant
    pred_range <- seq(
      min(data[[predictor]], na.rm = TRUE),
      max(data[[predictor]], na.rm = TRUE),
      length.out = 100
    )
    const_data <- data[1, , drop = FALSE]  # Use first row to keep other predictors constant
```

```r
  response_data <- const_data[rep(1, 100), ]  # Duplicate the row
  response_data[[predictor]] <- pred_range

  # Predict probabilities
  predicted_response <- predict(glm_model, newdata = response_data, type = "response")

  # Create data frame for plotting
  plot_data <- data.frame(
    Predictor_Value = pred_range,
    Predicted_Probability = predicted_response
  )

  # Add presence and absence data
  presence_absence_data <- data.frame(
    Predictor_Value = data[[predictor]],
    Presence_Absence = data$Presence
  )

  # Generate the response plot
  p <- ggplot() +

    geom_line(data = plot_data,
              aes(x = Predictor_Value, y = Predicted_Probability),
              color = "#61c6fa", linewidth = 1) +

    geom_point(data = presence_absence_data[presence_absence_data$Presence_Absence == 1, ],
               aes(x = Predictor_Value, y = Presence_Absence),
               color = "#11aa96", alpha = 0.2) +

    geom_point(data = presence_absence_data[presence_absence_data$Presence_Absence == 0, ],
               aes(x = Predictor_Value, y = Presence_Absence),
               color = "#f6aa70", alpha = 0.2) +

    labs(x = predictor, y = NULL) +
    theme_bw() +
    theme(axis.title.y = element_blank())

  # Store the plot in the list
  response_plots[[predictor]] <- p
}

# Arrange all plots in one combined plot with a single shared y-axis label
grid.arrange(
  grobs = response_plots,
```

```
    ncol = 3,
    left = "Predicted Probability / Presence-Absence"
  )
}
```

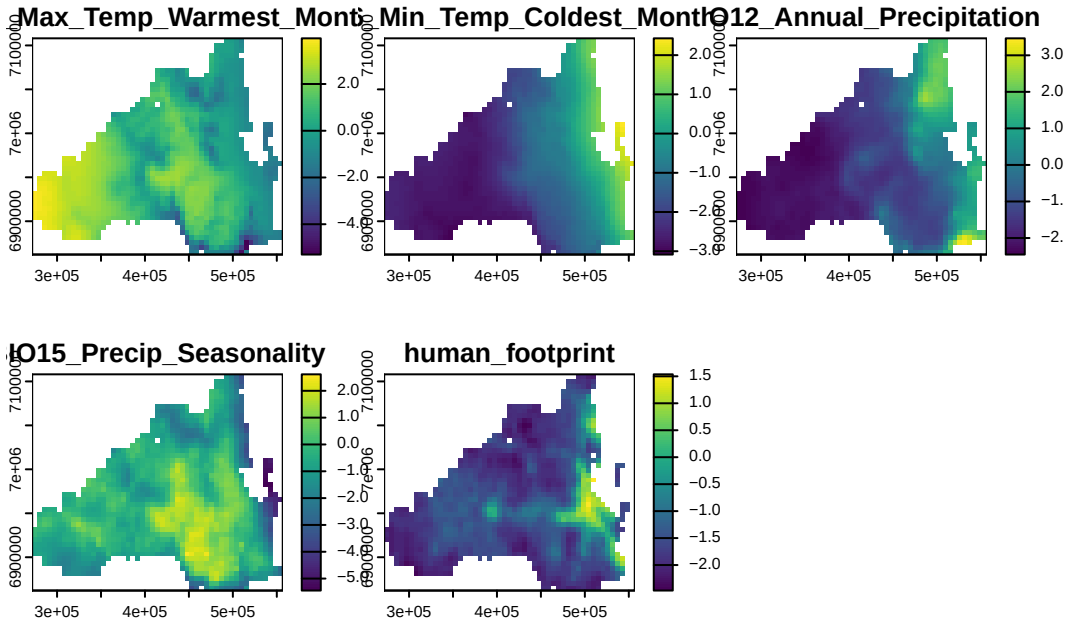
Plot the response curves

```
# get the names of the covariates in the model (drop the intercept)
predictors <- names(glm_model$coefficients)[-1]

# Plot the response curves for the predictors in the model
plot_species_response(glm_model, predictors, train_PB_covs_thinned_scaled)
```



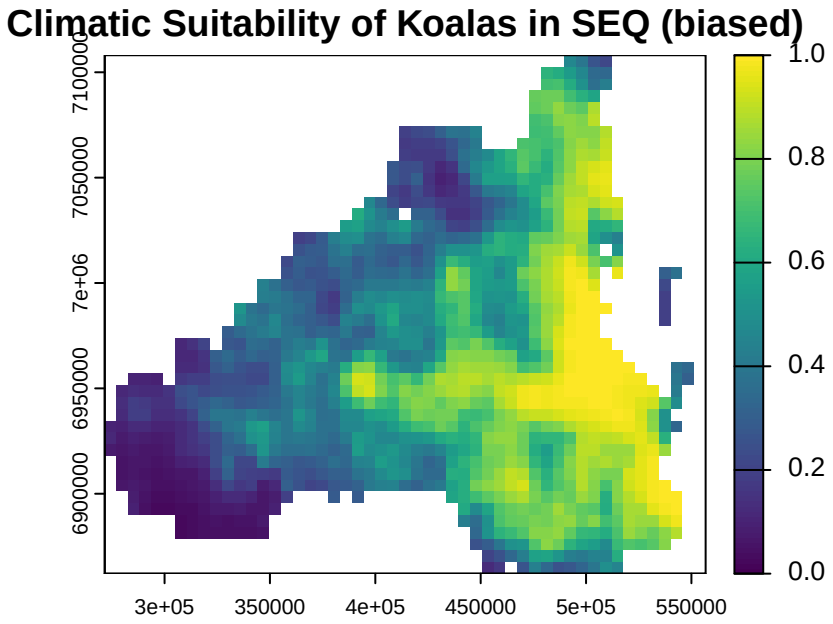
Scale the raster layers with the same scaling as the covariates

```
# Scale your prediction rasters using the same scaling parameters
covs_current_expert_HF_scaled <- scale(covs_current_expert_HF,
                                       center = attr(scaled_covariates, "scaled:center"),
                                       scale = attr(scaled_covariates, "scaled:scale"))

# plot to check the scaling worked
plot(covs_current_expert_HF_scaled)
```



GLM predictions to current environment

```
# Predict the presence probability across the entire raster extent
predicted_current_biased <- predicts::predict(covs_current_expert_HF_scaled, glm_model, type

# Plot the species distribution raster
plot(
  predicted_current_biased,
  range = c(0, 1),  # Set min and max values for the color scale
  main = "Climatic Suitability of Koalas in SEQ (biased)"
)
```

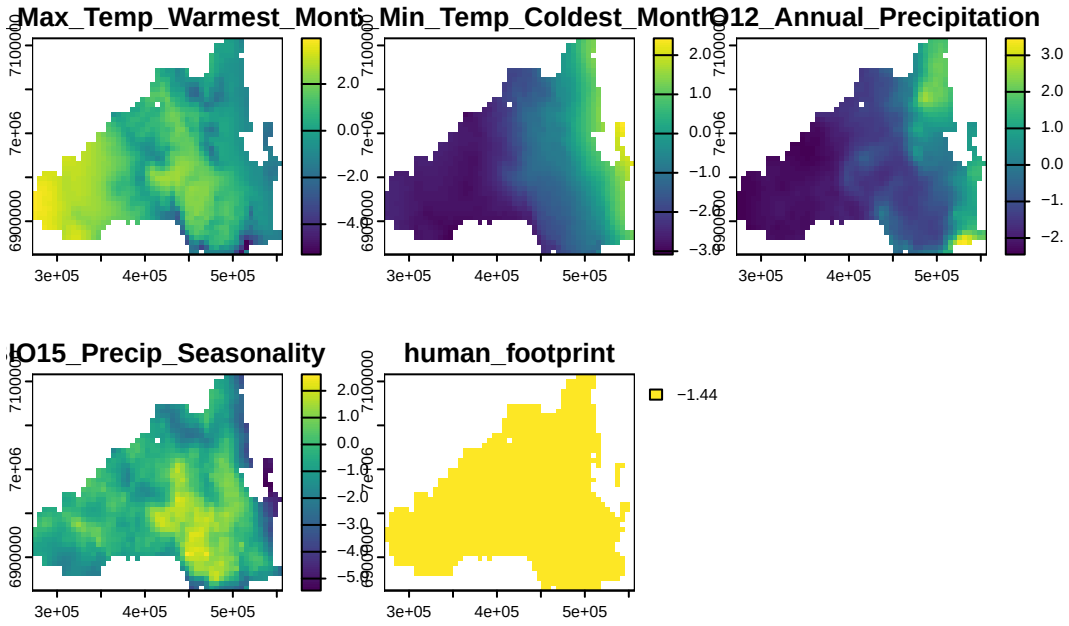


Accounting for the sampling bias

What we did above did not account for the sampling bias, as the predictions we generated maintained the bias process. We need to instead set the human footprint index to a constant value across the entire extent, which represents if the process that produces the bias is the same everywhere.

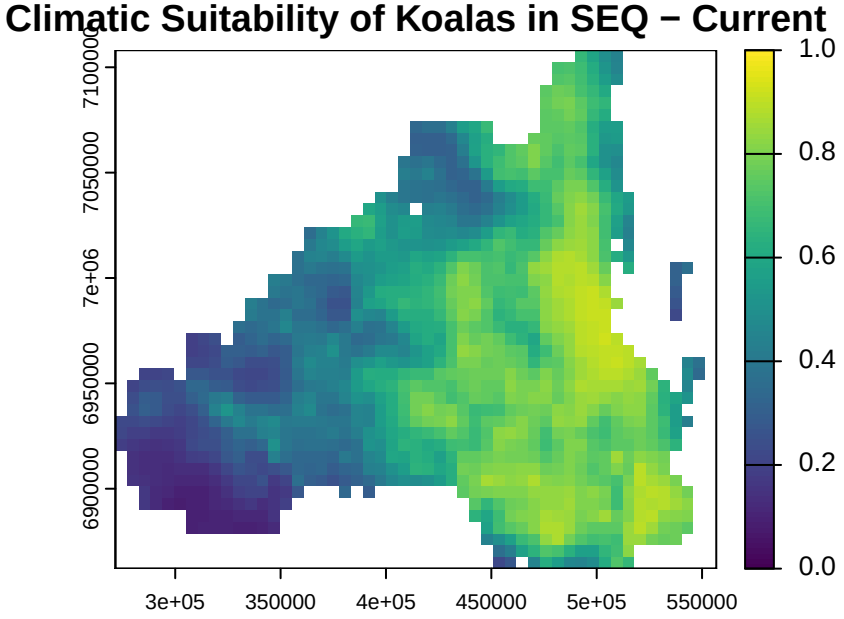
The approach that we've taken is not perfect, and would require some thinking and investigation, but it illustrates the potential impact of sampling bias and how you might account for it.

```
# pull out the scaled human footprint index layer
constant_HF <- covs_current_expert_HF_scaled[[5]]

# Set the human footprint index to a constant value across the entire extent
# Mean value of the human footprint index
constant_HF[] <- mean(constant_HF[], na.rm = TRUE)
# mask to the extent of the covariates
constant_HF <- terra::mask(constant_HF, covs_current_expert_HF_scaled[[5]])

# add the constant human footprint back into the raster layers
covs_current_expert_constHF_scaled <- c(covs_current_expert_HF_scaled[[1:4]], constant_HF)

# check the new human footprint layer
plot(covs_current_expert_constHF_scaled)
```



Generate predictions with the constant human footprint layer

```
# Predict the presence probability across the entire raster extent
predicted_current <- predicts::predict(covs_current_expert_constHF_scaled, glm_model, type =

# Plot the species distribution raster
plot(
  predicted_current,
  range = c(0, 1),  # Set min and max values for the color scale
  main = "Climatic Suitability of Koalas in SEQ - Current"
)
```




```
# Save the plot as a png
png(filename="Outputs/GLM_outputs/predicted_current.png",
    width = 150, height = 130, units = "mm", res = 600)
plot(predicted_current,
  range = c(0, 1),  # Set min and max values for the color scale
  main = "Climatic Suitability of Koalas in SEQ - Current")
dev.off()
```

```
pdf
  2
```

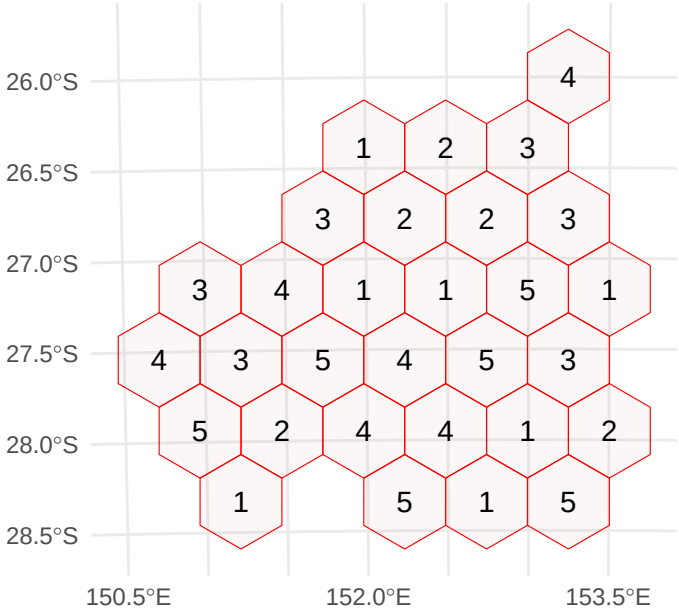
```
# Write the raster to file
writeRaster(predicted_current, "Outputs/GLM_outputs/predicted_current.tif", overwrite = TRUE
```

Model evaluation with spatial block cross-validation

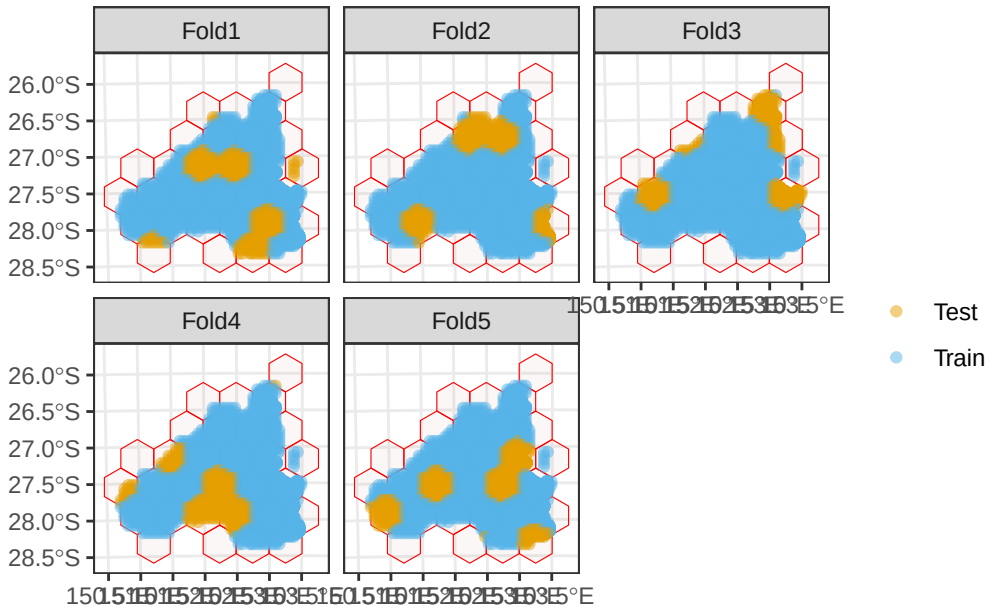
```
# Convert training data to sf
train_PB_covs_thinned_sf <- st_as_sf(train_PB_covs_thinned[, c("x", "y", "Presence")], coord

# Generate spatial blocks
spblock <- cv_spatial(x = train_PB_covs_thinned_sf,
                      column = "Presence",
                      r = NULL,
                      size = 50000, # Size of the blocks in metres
```

```
                        k = 5,
                        hexagon = TRUE,
                        selection = "random",
                        iteration = 100, # to find evenly-dispersed folds
                        biomod2 = FALSE,
                        progress = FALSE)
```

```
  train_0 train_1 test_0 test_1
1     957    9310    267    690
2     987    8385    237   1615
3    1024    5324    200   4676
4     981    9428    243    572
5     947    7553    277   2447
```




```
cv_plot(cv = spblock,
        x = train_PB_covs_thinned_sf,
        points_alpha = 0.5,
        nrow = 2)
```




```
# Extract the folds to save
spfolds <- spblock$folds_list

# We now have a list of 5 folds, where the first object is the training data, and the second
str(spfolds)
```

```
List of 5
 $ :List of 2
  ..$ : int [1:10267] 10734 10639 10595 10687 10593 10829 10685 10594 10780 10638 ...
  ..$ : int [1:957] 5755 11165 11107 11136 11162 11159 11164 11134 11163 11133 ...
 $ :List of 2
  ..$ : int [1:9372] 10734 10639 10595 10687 10593 10829 10685 10594 10780 10638 ...
  ..$ : int [1:1852] 255 6283 4673 1817 11032 961 2481 10987 4341 11028 ...
 $ :List of 2
  ..$ : int [1:6348] 10734 10639 10595 10687 10593 10829 10685 10594 10780 10638 ...
  ..$ : int [1:4876] 10478 10743 10600 10838 10744 10517 8438 10697 10552 10555 ...
 $ :List of 2
  ..$ : int [1:10409] 11024 10975 10972 10974 11103 10928 11017 10877 11102 11022 ...
  ..$ : int [1:815] 10734 10639 10595 10687 10593 10829 10685 10594 10780 10638 ...
 $ :List of 2
  ..$ : int [1:8500] 10734 10639 10595 10687 10593 10829 10685 10594 10780 10638 ...
  ..$ : int [1:2724] 11024 10975 10972 10974 11103 10928 11017 10877 11102 11022 ...
```

Run the model for every fold and evaluate

Model evaluation - metrics

Typically, it helps to evaluate your model with several metrics that describe different features of model performance and prediction. Here, we define a function to feed in a model prediction and calculate several evaluation metrics.

The metrics are:

Area under the receiver operating characteristic curve (AUC ROC)

- Higher values of this (closer to 1) suggest a model is good at distinguishing presence points from the background.

Continuous boyce index

- Higher values of this (closer to 1) suggest a model is good at predicting higher suitability at spots where there were presences.

```
# Start a dataframe to save results
eval_df <- data.frame(fold = numeric(),
                      ROC = numeric(),
                      boyce = numeric())

for(f in seq_along(spfolds)) {

  # Subset the training and testing data (spatial cross validation) (for the fth fold)

  train_PB_covs_scv <- train_PB_covs_thinned[spfolds[[f]][[1]], ]
  test_PB_covs_scv <- train_PB_covs_thinned[spfolds[[f]][[2]], ]

  glm_model_fold <- glm(Presence ~
                          BIO5_Max_Temp_Warmest_Month +
                          BIO6_Min_Temp_Coldest_Month +
                          BIO12_Annual_Precipitation +
                          BIO15_Precip_Seasonality +
                          human_footprint,
                        data=train_PB_covs_scv,
                        family = binomial(link = "logit"))

    # Predict to the testing data of fold f
  test_PB_covs_scv$pred <- predict(glm_model_fold, newdata = test_PB_covs_scv, type = "respo

  # Evaluate prediction on test set
  ROC = precrec::auc(precrec::evalmod(scores = test_PB_covs_scv$pred, labels = test_PB_covs_
```

```
  boyce = ecospat::ecospat.boyce(fit = test_PB_covs_scv$pred,
                                 obs = test_PB_covs_scv$pred[which(test_PB_covs_scv$Presence
                                 nclass = 0, # Calculate continuous index
                                 method = "pearson",
                                 PEplot = F)[["cor"]]

  # Add results to dataframe
  eval_df <- eval_df %>% add_row(fold = f, ROC = ROC, boyce = boyce)

}
```

Summarise the evaluation metrics

```
# Mean AUC & boyce
eval_df %>%
  summarise(mean_AUC = mean(ROC),
            mean_boyce = mean(boyce),
            sd_AUC = sd(ROC),
            sd_boyce = sd(boyce))
```

```
   mean_AUC mean_boyce     sd_AUC   sd_boyce
1 0.8423407     0.8362 0.09709689 0.07477433
```

Make predictions to future climates

Load future environmental data

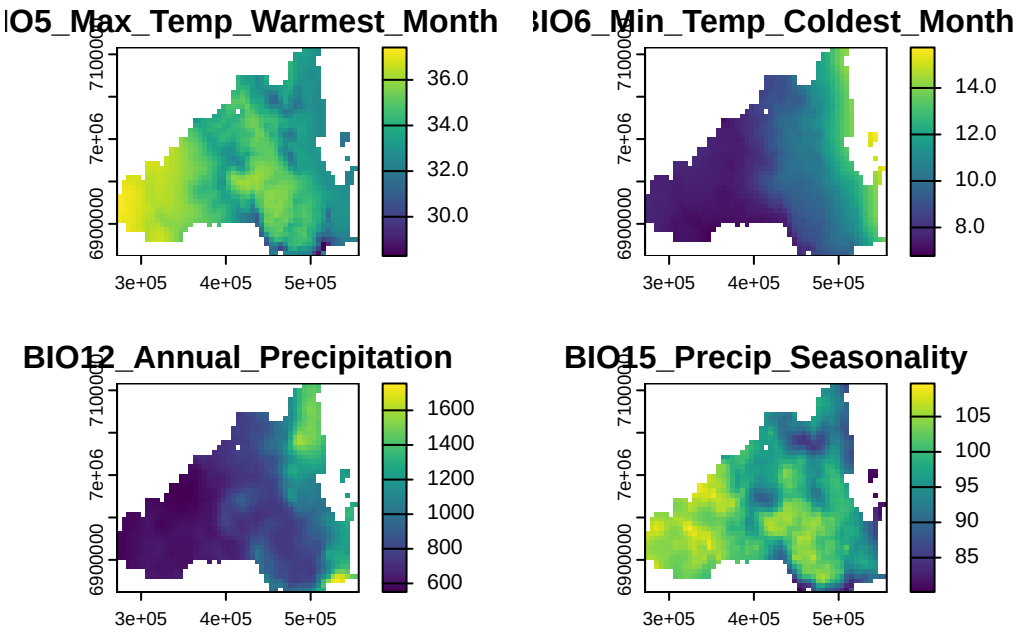
```
covs_future_SSP370 <- rast("Data/Environmental_variables/SEQ_future_bioclim.2090.SSP370.tif"
names(covs_future_SSP370) <- layer_names
covs_future_SSP370
```

```
class       : SpatRaster
dimensions  : 44, 51, 19  (nrow, ncol, nlyr)
resolution  : 5590.925, 5590.925  (x, y)
extent      : 271609.2, 556746.4, 6862081, 7108082  (xmin, xmax, ymin, ymax)
coord. ref. : GDA2020 / MGA zone 56 (EPSG:7856)
source      : SEQ_future_bioclim.2090.SSP370.tif
names       : BIO1_~_Temp, BIO2_~Range, BIO3_~ality, BIO4_~ality, BIO5_~Month, BIO6_~Month,
min values  :    19.07255,     6.59230,    40.85448,    325.5495,    28.28100,    6.774293,
max values  :    24.52122,    14.53494,    50.96117,    530.5290,    37.41868,   15.740561,
```

```
covs_future_SSP370 <- terra::mask(covs_future_SSP370, covs_current) # Crop to SEQ extent usi

covs_future_SSP370_expert <- subset(covs_future_SSP370,
                                    names(covs_future_SSP370) %in% c("BIO5_Max_Temp_Warmest_
                                                                     "BIO6_Min_Temp_Coldest_
                                                                     "BIO12_Annual_Precipita
                                                                     "BIO15_Precip_Seasonali
```

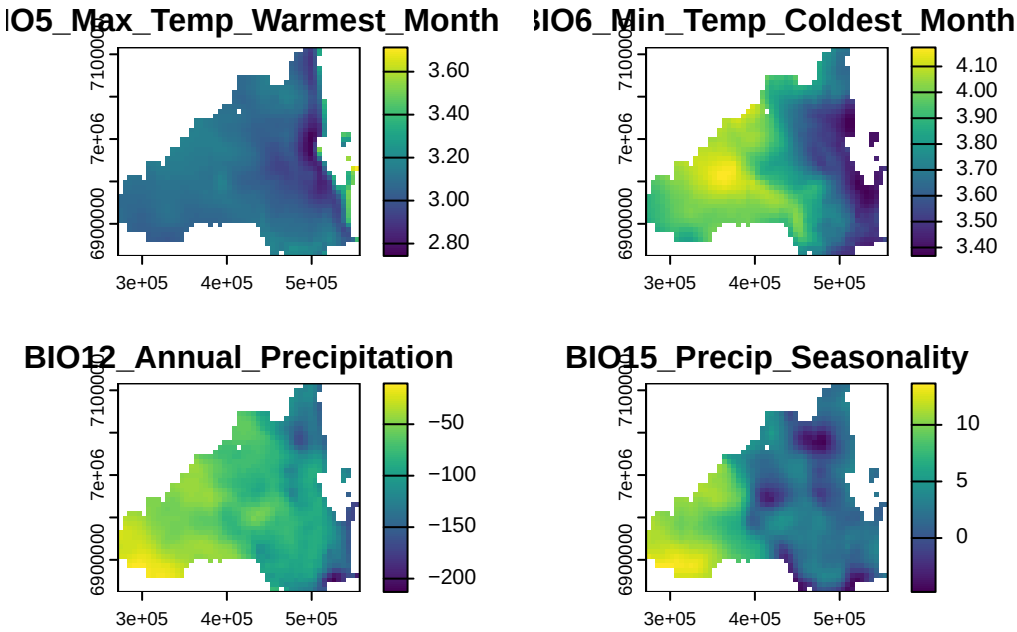
Plot the future rasters

```
plot(covs_future_SSP370_expert)
```



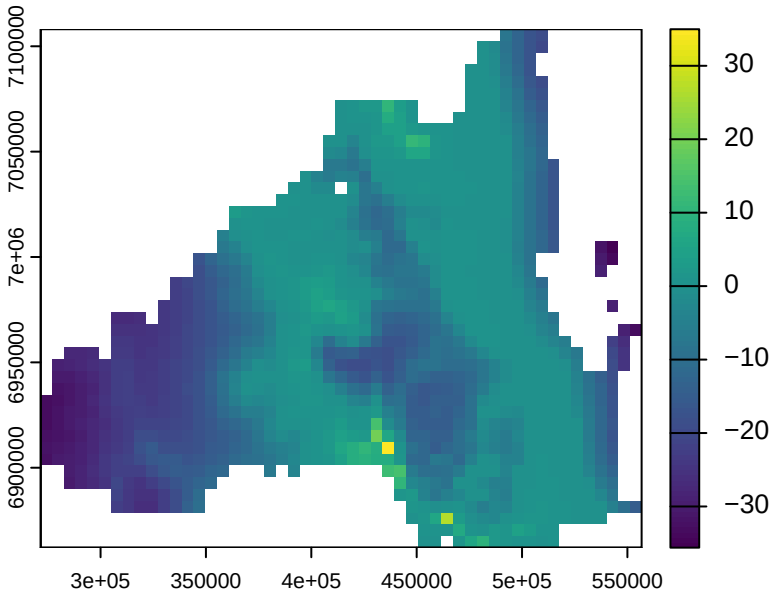
Compare the current and future rasters

```
plot(covs_future_SSP370_expert - covs_current_expert)
```

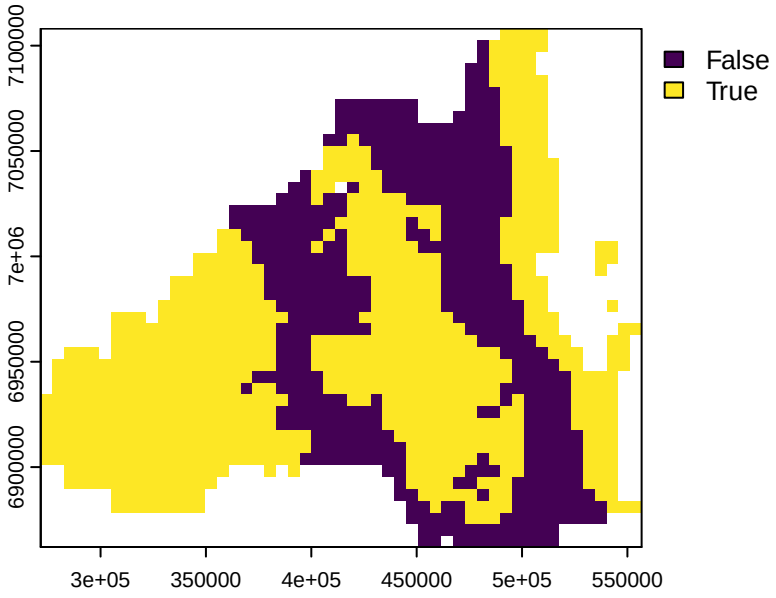


Test the environmental distance between current data and future conditions

```
mess <- predicts::mess(covs_future_SSP370_expert,
                       train_PB_covs_thinned[, c("BIO5_Max_Temp_Warmest_Month",
                                                 "BIO6_Min_Temp_Coldest_Month",
                                                 "BIO12_Annual_Precipitation",
                                                 "BIO15_Precip_Seasonality")])

plot(mess)
```



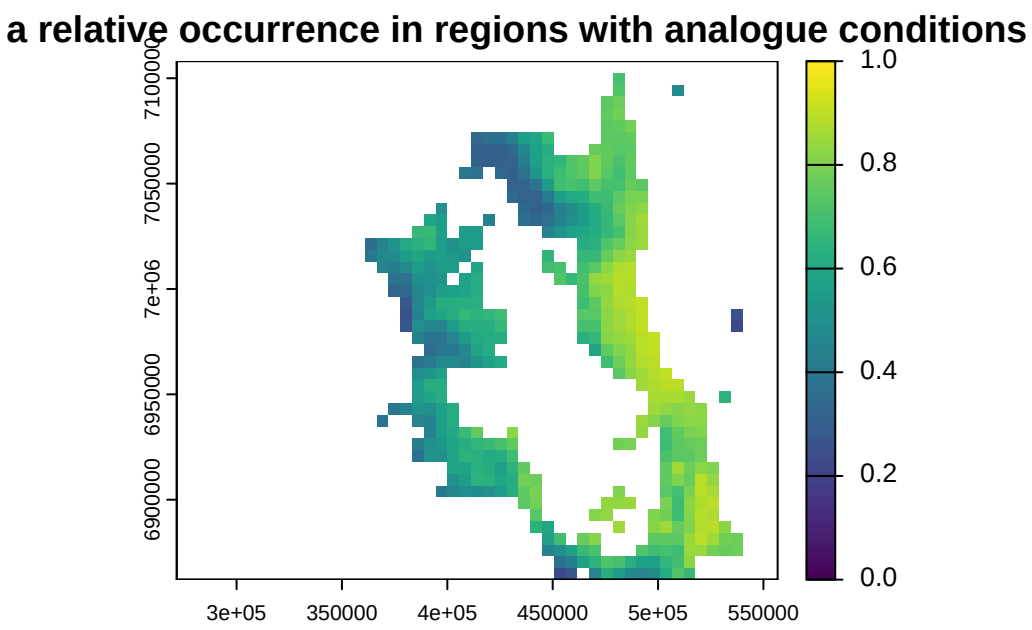

```
r_mess_mask <- mess < 0
plot(r_mess_mask)
```



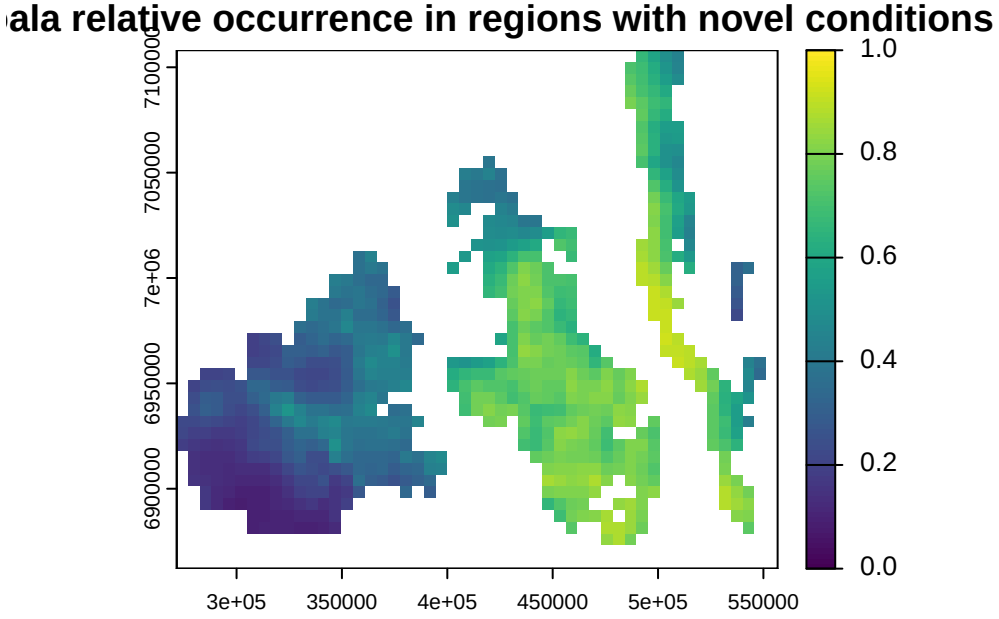
Test which areas you might mask out because they are 'novel' in environmental space and therefore require model extrapolation.

```
analog_fut <- predicted_current

values(analog_fut)[values(mess)<0] <- NA
```

```r
plot(analog_fut,
     range = c(0, 1),  # Set min and max values for the color scale
     main = "Koala relative occurrence in regions with analogue conditions")
```




```r
novel_fut <- predicted_current

values(novel_fut)[values(mess)>0] <- NA

plot(novel_fut,
     range = c(0, 1),  # Set min and max values for the color scale
     main = "Koala relative occurrence in regions with novel conditions")
```



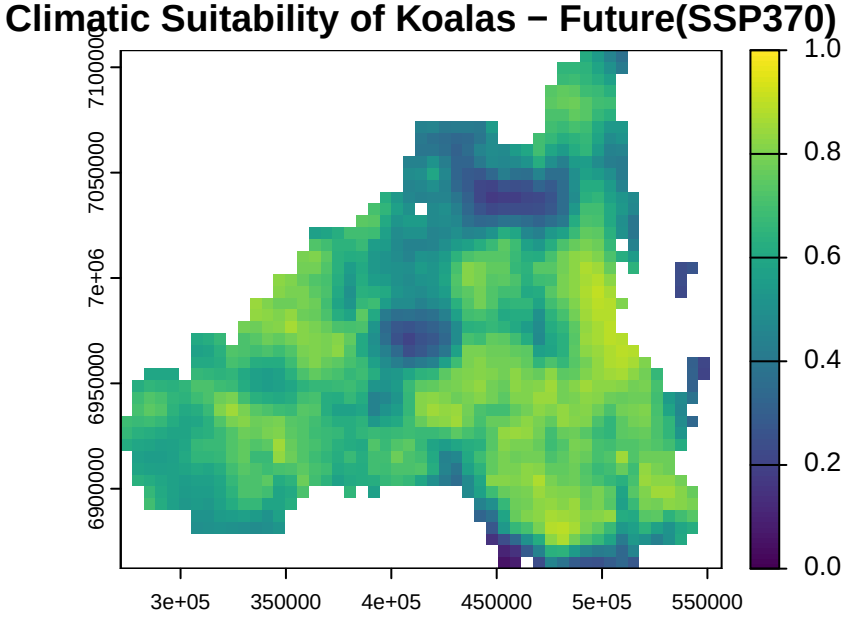
GLM future predictions

Scale future layers and add human footprint

```
# Scale the future covariates using the same scaling parameters as the training data
covs_future_SSP370_expert_scaled <- scale(covs_future_SSP370_expert,
                                          center = attr(scaled_covariates, "scaled:center")[-5],
                                          scale = attr(scaled_covariates, "scaled:scale")[-5])

# Add the human footprint layer to the future covariates
covs_future_SSP370_expert_HF_scaled <- c(covs_future_SSP370_expert_scaled, constant_HF)
```

Make projections

```
# Predict the presence probability across the entire raster extent
predicted_future370 <- predict(covs_future_SSP370_expert_HF_scaled, glm_model, type = "respo

# Plot the species distribution raster
plot(
  predicted_future370,
  range = c(0, 1),  # Set min and max values for the color scale
  main = "Climatic Suitability of Koalas - Future(SSP370)"
)
```




```
# Save the plot as a png
png(filename="Outputs/GLM_outputs/predicted_future370.png",
    width = 150, height = 130, units = "mm", res = 600)
plot(predicted_future370,
  range = c(0, 1),  # Set min and max values for the color scale
  main = "Climatic Suitability of Koalas - Future(SSP370)")
dev.off()
```

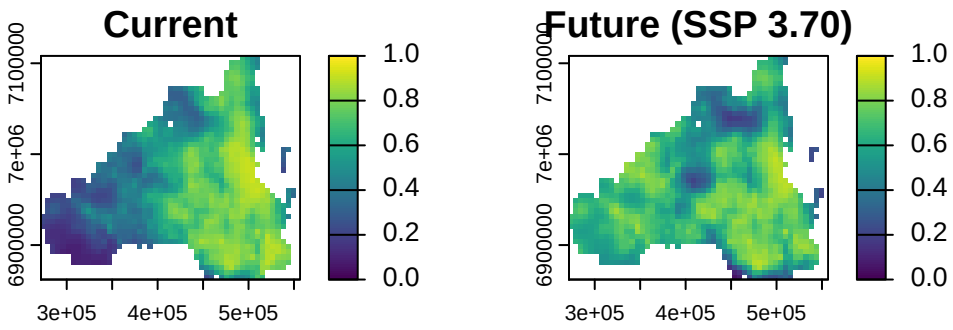
```
pdf
  2
```

```
# Write the raster to file
writeRaster(predicted_future370, "Outputs/GLM_outputs/predicted_future370.tif", overwrite =
```

Show the predictions side-by-side

```
par(mfrow = c(1, 2))

plot(
  predicted_current,
  range = c(0, 1),
  main = "Current"
)
```

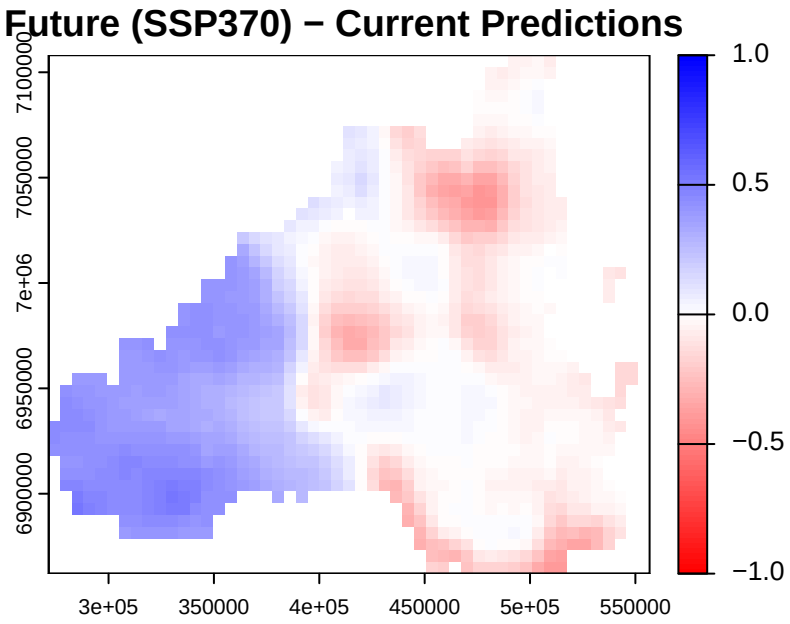
```r
plot(
  predicted_future370,
  range = c(0, 1),
  main = "Future (SSP 3.70)"
)
```



Plot the different between current and future

Red is **less** suitable in the Future SSP370 scenario, and blue is **more** suitable in the Future SSP370 scenario.

```r
# return to single plotting
par(mfrow = c(1, 1))

# Create symmetric breaks centered at 0
breaks <- seq(-1, 1, length.out = 101)

# Create the color palette
cols <- colorRampPalette(c("red", "white", "blue"))(100)


plot(predicted_future370 - predicted_current,
     main = "Future (SSP370) - Current Predictions ",
     col = cols,
     range = c(-1, 1))
```




```
# Save the plot as a png
png(filename="Outputs/GLM_outputs/future370-current.png",
    width = 150, height = 130, units = "mm", res = 600)
plot(predicted_future370 - predicted_current,
     main = "Future (SSP370) - Current Predictions ",
     col = cols,
     range = c(-1, 1))
dev.off()
```

```
pdf
  2
```

```
# Write the raster to file
writeRaster(predicted_future370 - predicted_current,
            "Outputs/GLM_outputs/future370-current.tif", overwrite = TRUE)
```

Presenting predictions with uncertainty

There are many sources of model uncertainty that should be explored and ideally, presented alongside model predictions.

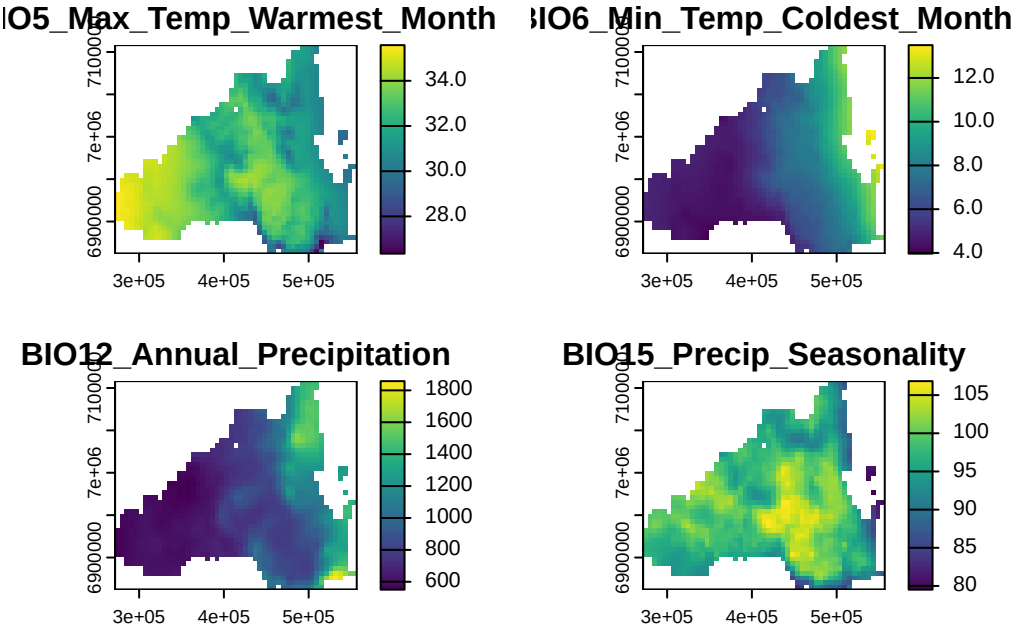
One that we'll focus on here is climate scenario uncertainty. We do so by fitting a second model to future climate data from a lower emission shared socioeconomic path scenario (SSP 1.26).

Load future environmental data (SSP 1.26)

```
covs_future_SSP126 <- rast("Data/Environmental_variables/SEQ_future_bioclim.2090.SSP126.tif"
names(covs_future_SSP126) <- layer_names
covs_future_SSP126
```

```
class       : SpatRaster
dimensions  : 44, 51, 19  (nrow, ncol, nlyr)
resolution  : 5590.925, 5590.925  (x, y)
extent      : 271609.2, 556746.4, 6862081, 7108082  (xmin, xmax, ymin, ymax)
coord. ref. : GDA2020 / MGA zone 56 (EPSG:7856)
source      : SEQ_future_bioclim.2090.SSP126.tif
names       : BIO1_~_Temp, BIO2_~Range, BIO3_~ality, BIO4_~ality, BIO5_~Month, BIO6_~Month,
min values  :    17.11955,    6.654293,    41.56141,    322.8842,    26.35068,     3.96662,
max values  :    22.42196,   14.853741,    51.27921,    546.6645,    35.57761,    13.50076,
```

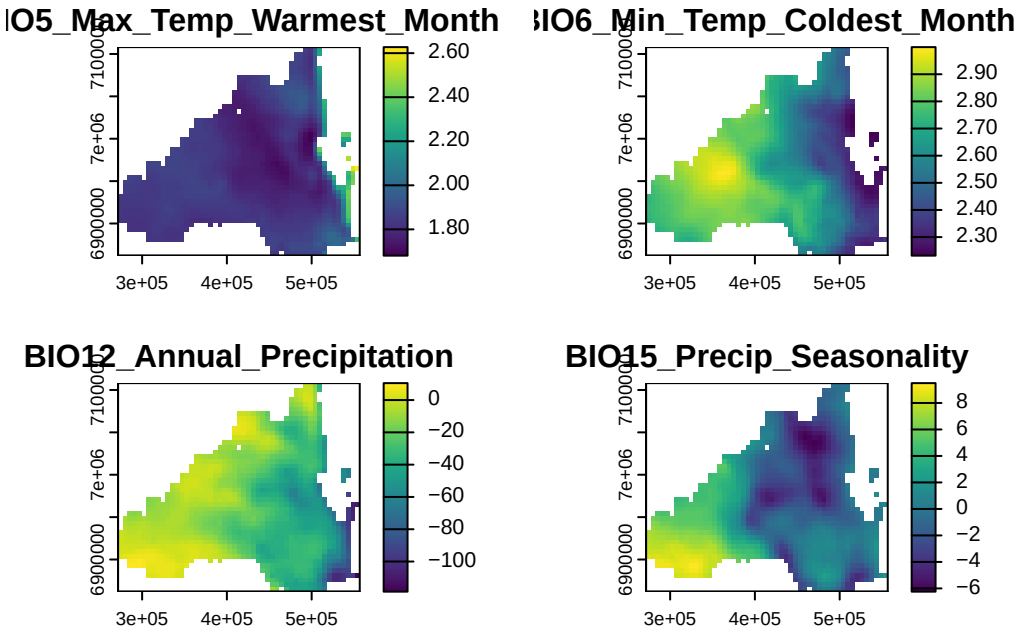
```
covs_future_SSP126_expert <- subset(covs_future_SSP126, names(covs_future_SSP126) %in% c("BI
                                                                                         "B
                                                                                         "B
                                                                                         "B


plot(covs_future_SSP126_expert) # to plot the layers
```



Plot the difference between the two future scenarios

This is SSP370 - SSP126, so positive values indicate that the SSP370 scenario has higher values than the SSP126 scenario.

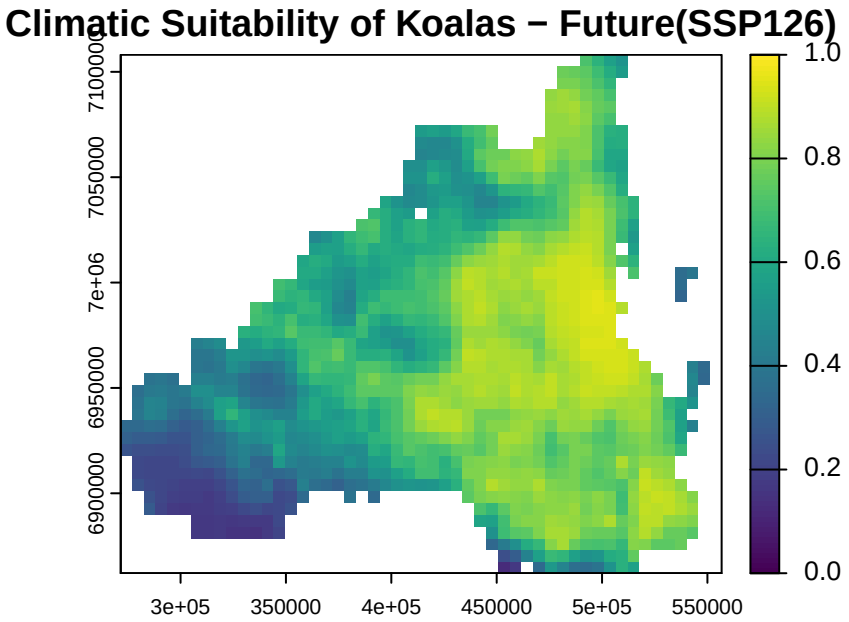
```
terra::plot(covs_future_SSP370_expert - covs_future_SSP126_expert)
```



GLM future predictions (SSP 1.26)

```
# Scale the future covariates using the same scaling parameters as the training data
covs_future_SSP126_expert_scaled <- scale(covs_future_SSP126_expert,
                                          center = attr(scaled_covariates, "scaled:center")[-5],
                                          scale = attr(scaled_covariates, "scaled:scale")[-5])

# Add the human footprint layer to the future covariates
covs_future_SSP126_expert_HF_scaled <- c(covs_future_SSP126_expert_scaled, constant_HF)

# Predict the presence probability across the entire raster extent
predicted_future126 <- predict(covs_future_SSP126_expert_HF_scaled, glm_model, type = "respo

# Plot the species distribution raster
plot(
  predicted_future126,
  range = c(0, 1),  # Set min and max values for the color scale
```

```
  main = "Climatic Suitability of Koalas - Future(SSP126)"
)
```




```
# Save the plot as a png
png(filename="Outputs/GLM_outputs/predicted_future126.png",
    width = 150, height = 130, units = "mm", res = 600)
plot(predicted_future126,
  range = c(0, 1),  # Set min and max values for the color scale
  main = "Climatic Suitability of Koalas - Future(SSP126)")
dev.off()
```

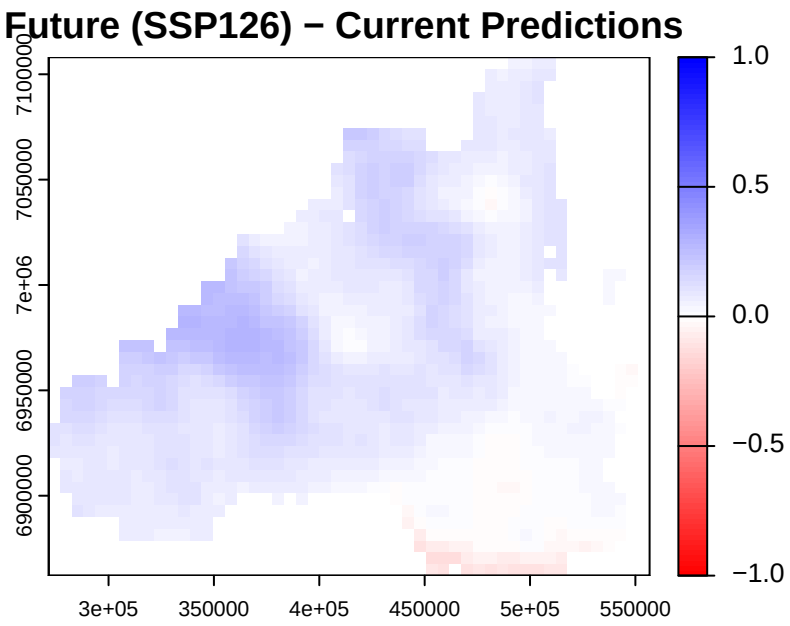
```
pdf
  2
```

```
# Write the raster to file
writeRaster(predicted_future370, "Outputs/GLM_outputs/predicted_future126.tif", overwrite =
```

Plot the different between current and future (SSP126)

Red is **less** suitable in the Future SSP126 scenario, and blue is **more** suitable in the Future SSP126 scenario.

```r
# return to single plotting
par(mfrow = c(1, 1))

# Create symmetric breaks centered at 0
breaks <- seq(-1, 1, length.out = 101)

# Create the color palette
cols <- colorRampPalette(c("red", "white", "blue"))(100)


plot(predicted_future126 - predicted_current,
     main = "Future (SSP126) - Current Predictions ",
     col = cols,
     range = c(-1, 1))
```



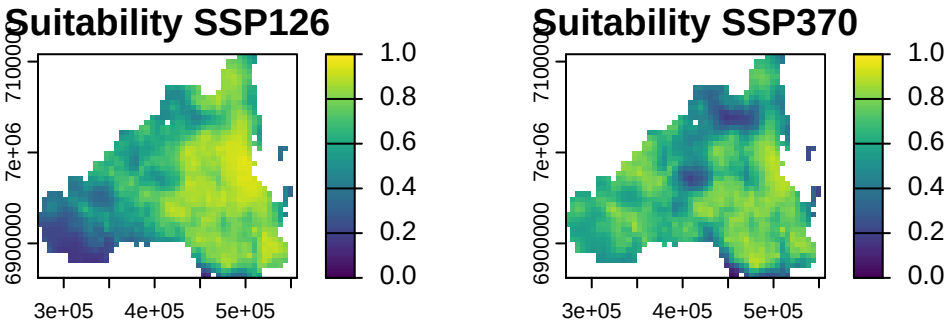

```r
# Save the plot as a png
png(filename="Outputs/GLM_outputs/future126-current.png",
    width = 150, height = 130, units = "mm", res = 600)
plot(predicted_future126 - predicted_current,
     main = "Future (SSP126) - Current Predictions ",
     col = cols,
     range = c(-1, 1))
dev.off()
```

```
pdf
  2
```

```r
# Write the raster to file
writeRaster(predicted_future126 - predicted_current,
            "Outputs/GLM_outputs/future126-current.tif", overwrite = TRUE)
```

Compare the two future predictions

```r
par(mfrow = c(1, 2))

# Plot the predicted suitability
plot(
  predicted_future126,
  range = c(0,1),
  main = "Suitability SSP126"
)

plot(
  predicted_future370,
  range = c(0,1),
  main = "Suitability SSP370"
)
```

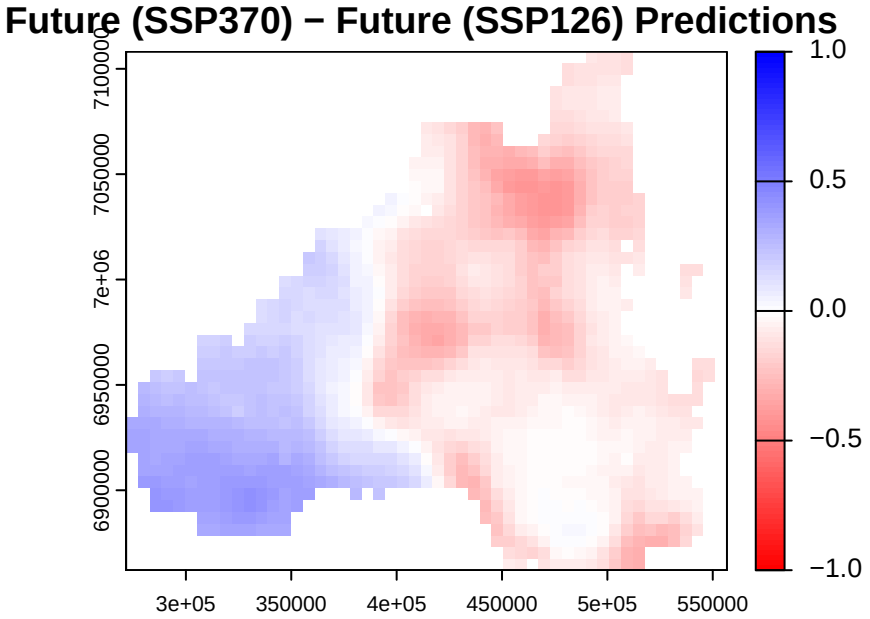


Plot the different between the two future predictions

Red is **less** suitable in the Future SSP370 scenario than the SSP126 scenario, and blue is **more** suitable in Future SSP370 scenario than the SSP126 scenario.

```r
# return to single plotting
par(mfrow = c(1, 1))

# Create symmetric breaks centered at 0
breaks <- seq(-1, 1, length.out = 101)
```

```r
# Create the color palette
cols <- colorRampPalette(c("red", "white", "blue"))(100)


plot(predicted_future370 - predicted_future126,
     main = "Future (SSP370) - Future (SSP126) Predictions ",
     col = cols,
     range = c(-1, 1))
```




```r
# Save the plot as a png
png(filename="Outputs/GLM_outputs/future370-future126.png",
    width = 150, height = 130, units = "mm", res = 600)
plot(predicted_future370 - predicted_future126,
     main = "Future (SSP370) - Future (SSP126) Predictions ",
     col = cols,
     range = c(-1, 1))
dev.off()
```

```
pdf
  2
```

```r
# Write the raster to file
writeRaster(predicted_future370 - predicted_future126,
            "Outputs/GLM_outputs/future370-future126.tif", overwrite = TRUE)
```

Model uncertainty

Another element of uncertainty that can be represented is model uncertainty, or the standard error around the coefficient estimates.

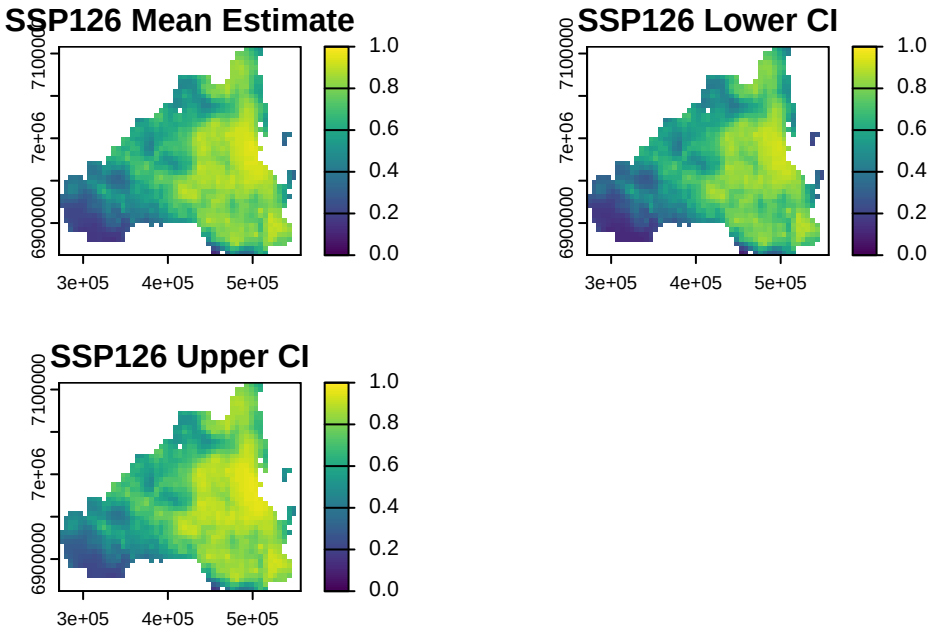
```
# Extract standard errors of coefficients
coef_se <- summary(glm_model)$coefficients[, "Std. Error"]
print(coef_se)
```

```
               (Intercept) BIO5_Max_Temp_Warmest_Month
                0.07444492                  0.05628503
BIO6_Min_Temp_Coldest_Month  BIO12_Annual_Precipitation
                0.07578009                  0.07954993
   BIO15_Precip_Seasonality             human_footprint
                0.04546156                  0.06828756
```

Projecting with uncertainty

```
covs_df <- as.data.frame(covs_future_SSP126_expert_HF_scaled, na.rm = FALSE)

pred_link <- predict(glm_model, newdata = covs_df, type = "link", se.fit = TRUE)

# Linear predictor (eta)
eta <- pred_link$fit
se_eta <- pred_link$se.fit

# Confidence intervals (95%)
z <- 1.96
eta_lower <- eta - z * se_eta
eta_upper <- eta + z * se_eta

# Transform back to response scale
linkinv <- glm_model$family$linkinv
predicted <- linkinv(eta)
lower_ci <- linkinv(eta_lower)
upper_ci <- linkinv(eta_upper)


# Add to covs_df
covs_df$predicted <- predicted
covs_df$lower_ci <- lower_ci
covs_df$upper_ci <- upper_ci
```

```r
predicted_r <- setValues(rast(covs_future_SSP126_expert_HF_scaled, nlyr = 1), predicted)
lower_ci_r <- setValues(rast(covs_future_SSP126_expert_HF_scaled, nlyr = 1), lower_ci)
upper_ci_r <- setValues(rast(covs_future_SSP126_expert_HF_scaled, nlyr = 1), upper_ci)

# Step 2: Name the layers
names(predicted_r) <- "SSP126 Mean Estimate"
names(lower_ci_r) <- "SSP126 Lower CI"
names(upper_ci_r) <- "SSP126 Upper CI"

prediction_w_uncertainty <- c(predicted_r, lower_ci_r, upper_ci_r)

plot(prediction_w_uncertainty, range = c(0, 1))
```



References

Conn, Paul B, James T Thorson, and Devin S Johnson. 2017. "Confronting Preferential Sampling When Analysing Population Distributions: Diagnosis and Model-based Triage." *Methods in Ecology and Evolution* 8 (11): 1535–46. https://doi.org/10.1111/2041-210x.12803.

Dubos, Nicolas, Clémentine Préau, Maxime Lenormand, Guillaume Papuga, Sophie Monsarrat, Pierre Denelle, Marine Le Louarn, et al. 2022. "Assessing the Effect of Sample Bias Correction in Species Distribution Models." *Ecological Indicators* 145 (109487): 109487. https://doi.org/10.1016/j.ecolind.2022.109487.

Foster, Scott D, David Peel, Geoffrey R Hosack, Andrew Hoskins, David J Mitchell, Kirstin Proft, Wen-Hsi Yang, David E Uribe-Rivera, and Jens G Froese. 2024. "'RISDM': Species Distribution Modelling from Multiple Data Sources in R." *Ecography* 2024 (6). https://doi.org/10.1111/ecog.06964.

Kéry, Marc, J Andrew Royle, Hans Schmid, Michael Schaub, Bernard Volet, Guido Häfliger, and Niklaus Zbinden. 2010. "Site-Occupancy Distribution Modeling to Correct Population-Trend Estimates Derived from Opportunistic Observations: Distribution Trends from Opportunistic Observations." *Conservation Biology: The Journal of the Society for Conservation Biology* 24 (5): 1388–97. https://doi.org/10.1111/j.1523-1739.2010.01479.x.

Mäkinen, Jussi, Cory Merow, and Walter Jetz. 2024. "Integrated Species Distribution Models to Account for Sampling Biases and Improve Range-wide Occurrence Predictions." *Global Ecology and Biogeography: A Journal of Macroecology* 33 (3): 356–70. https://doi.org/10.1111/geb.13792.

Pennino, Maria Grazia, Iosu Paradinas, Janine B Illian, Facundo Muñoz, José María Bellido, Antonio López-Quílez, and David Conesa. 2019. "Accounting for Preferential Sampling in Species Distribution Models." *Ecology and Evolution* 9 (1): 653–63. https://doi.org/10.1002/ece3.4789.

Session information

```
sessionInfo()
```

```
R version 4.5.0 (2025-04-11)
Platform: aarch64-apple-darwin20
Running under: macOS Sequoia 15.5

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/lib/libRlapack.dylib;

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: Australia/Brisbane
tzcode source: internal

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
 [1] gridExtra_2.3      corrplot_0.95      precrec_0.14.5     usdm_2.1-7
```

```
 [5] ecospat_4.1.2      blockCV_3.1-5      predicts_0.1-19    tidyterra_0.7.2
 [9] sf_1.0-20          terra_1.8-50       galah_2.1.1        RColorBrewer_1.1-3
[13] lubridate_1.9.4    forcats_1.0.0      stringr_1.5.1      dplyr_1.1.4
[17] purrr_1.0.4        readr_2.1.5        tidyr_1.3.1        tibble_3.2.1
[21] ggplot2_3.5.2      tidyverse_2.0.0

loaded via a namespace (and not attached):
 [1] gtable_0.3.6       xfun_0.52          raster_3.6-32      httr2_1.1.2
 [5] lattice_0.22-6     tzdb_0.5.0         vctrs_0.6.5        tools_4.5.0
 [9] generics_0.1.3     parallel_4.5.0     proxy_0.4-27       pkgconfig_2.0.3
[13] KernSmooth_2.23-26 data.table_1.17.0  assertthat_0.2.1   lifecycle_1.0.4
[17] compiler_4.5.0     farver_2.1.2       tinytex_0.57       codetools_0.2-20
[21] rrapply_1.2.7      htmltools_0.5.8.1  potions_0.2.0      class_7.3-23
[25] yaml_2.3.10        pillar_1.10.2      crayon_1.5.3       classInt_0.4-11
[29] iterators_1.0.14   foreach_1.5.2      tidyselect_1.2.1   digest_0.6.37
[33] stringi_1.8.7      labeling_0.4.3     fastmap_1.2.0      grid_4.5.0
[37] cli_3.6.5          magrittr_2.0.3     utf8_1.2.5         dichromat_2.0-0.1
[41] e1071_1.7-16       withr_3.0.2        scales_1.4.0       rappdirs_0.3.3
[45] bit64_4.6.0-1      sp_2.2-0           timechange_0.3.0   rmarkdown_2.29
[49] lobstr_1.1.2       bit_4.6.0          hms_1.1.3          evaluate_1.0.3
[53] knitr_1.50         rlang_1.1.6        Rcpp_1.0.14        glue_1.8.0
[57] DBI_1.2.3          vroom_1.6.5        rstudioapi_0.17.1  jsonlite_2.0.0
[61] R6_2.6.1           units_0.8-7
```