

Source code for *Using citizen science to parse climatic and landcover influences on bird occupancy within a tropical biodiversity hotspot*

Vijay Ramesh Pratik R. Gupte Morgan W. Tingley VV Robin Ruth DeFries

2020-12-25

Contents

1	Introduction	2
1.1	Attribution	2
1.2	Data access	2
1.3	Data processing	2
1.4	Main Text Figure 1	2
2	Preparing eBird Data	4
2.1	Prepare libraries and data sources	4
2.2	Filter data	4
2.3	Process filtered data	5
2.4	Spatial filter	5
2.5	Handle presence data	6
2.6	Add decimal time	7
3	Preparing Environmental Predictors	8
3.1	Prepare libraries	8
3.2	Prepare spatial extent	8
3.3	Prepare terrain rasters	9
3.4	Prepare CHELSA rasters	9
3.5	Resample landcover from 10m to 1km	9
3.6	Resample other rasters to 1km	10
3.7	Temperature and rainfall in relation to elevation	11
3.8	Land cover type in relation to elevation	11
3.9	Main Text Figure 2	12
4	Preparing Observer Expertise Scores	12
4.1	Prepare libraries	12
4.2	Prepare data	14
4.3	Spatially explicit filter on checklists	15
4.4	Prepare species of interest	15
4.5	Prepare checklists for observer score	16
4.6	Get landcover	16
4.7	Filter checklist data	17
4.8	Model observer expertise	17
5	Examining Spatial Sampling Bias	18
5.1	Prepare libraries	19
5.2	Prepare data for processing	19
5.3	Main Text Figure 3	21

41	5.4	Distance to nearest neighbouring site	22
42	5.5	Main Text Figure 3	22
43	6	Adding Covariates to Checklist Data	22
44	6.1	Prepare libraries and data	22
45	6.2	Spatial subsampling	24
46	6.3	Temporal subsampling	25
47	6.4	Add checklist calibration index	25
48	6.5	Add climatic and landscape covariates	26
49	6.6	Spatial buffers around selected checklists	27
50	6.7	Spatial buffer-wide covariates	27
51	7	Modelling Species Occupancy	29
52	7.1	Load dataframe and scale covariates	29
53	7.2	Running a null model	31
54	7.3	Identifying covariates necessary to model the detection process	32
55	7.4	Land Cover and Climate	35
56	7.5	Goodness-of-fit tests	38
57	8	Visualizing Occupancy Predictor Effects	39
58	8.1	Prepare libraries	39
59	8.2	Load species list	39
60	8.3	Show AIC weight importance	40
61	8.4	Prepare model coefficient data	42
62	8.5	Get predictor effects	43
63	8.6	Main Text Figure 4	44

64 1 Introduction

65 This is the readable version containing analysis that models associations between environmental predictors (climate and
66 landcover) and citizen science observations of birds across the Nilgiri and Anamalai Hills of the Western Ghats Biodiversity
67 Hotspot.

68 Methods and format are derived from Strimas-Mackey et al..

69 1.1 Attribution

70 Please contact the following in case of interest in the project.

- 71 • Vijay Ramesh (lead author)
- 72 – PhD student, Columbia University
- 73 • Pratik Gupte (repo maintainer)
- 74 – PhD student, University of Groningen

75 1.2 Data access

76 The data used in this work are available from eBird.

77 1.3 Data processing

78 The data processing for this project is described in the following sections. Navigate through them using the links in the
79 sidebar.

80 1.4 Main Text Figure 1

81 Figure prepared in QGIS 3.10.

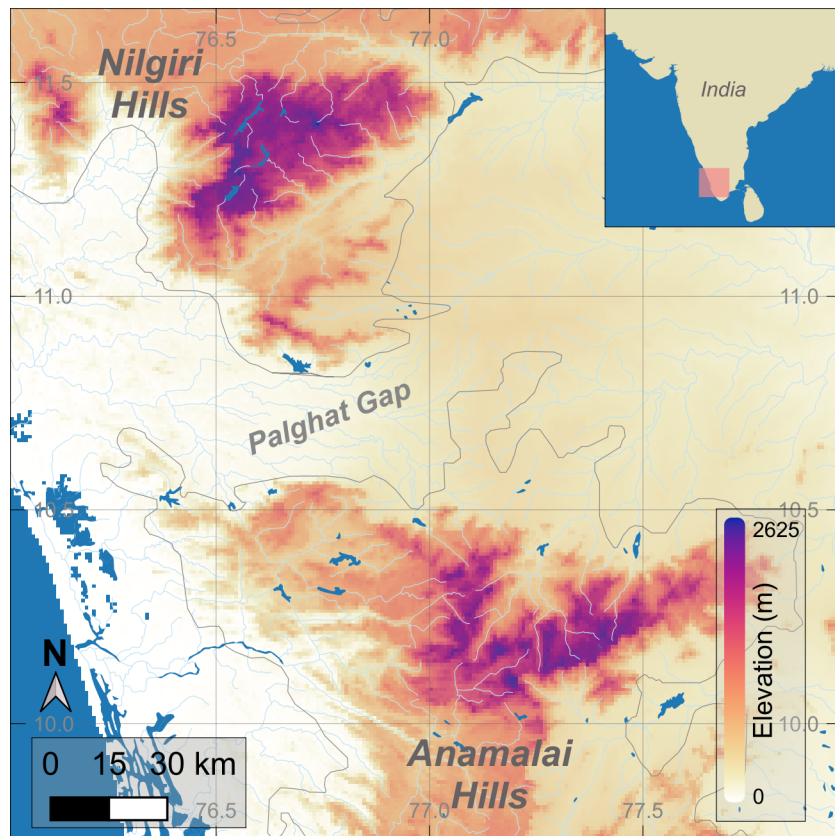


Figure 1: A shaded relief of the study area - the Nilgiri and the Anamalai hills are shown in this figure. This map was made using the SRTM digital elevation model at a spatial resolution of 1km and data from Natural Earth were used to outline boundaries of water bodies.

2 Preparing eBird Data

2.1 Prepare libraries and data sources

Here, we will load the necessary libraries required for preparing the eBird data. Please download the latest versions of the eBird Basic Dataset (for India) and the eBird Sampling dataset from <https://ebird.org/data/download>

```
# load libraries
library(tidyverse)
library(readr)
library(sf)
library(auk)
library(readxl)

# custom sum function
sum.no.na <- function(x) {
  sum(x, na.rm = T)
}

# set file paths for auk functions
# To use these two datasets, please download the latest versions from https://ebird.org/data/download and set the f
f_in_ebd <- file.path("data/ebd_IN_relApr-2020.txt")
f_in_sampling <- file.path("data/ebd_sampling_relApr-2020.txt")
```

2.2 Filter data

Insert the list of species that we will be analyzing in this study. We initially chose those species that occurred in at least 5% of all checklists across 50% of the 25 x 25 km cells from where they have been reported, resulting in a total of 93 species. To arrive at this final list of species, we carried out further pre-processing which can be found in Section 1 of the Supplementary material.

```
# add species of interest
specieslist <- read.csv("data/species_list.csv")
speciesOfInterest <- as.character(specieslist$scientific_name)

Here, we set broad spatial filters for the states of Kerala, Tamil Nadu and Karnataka and keep only those checklists for our list of species that were reported between 1st Jan 2013 and 31st Dec 2019.

# run filters using auk packages
ebd_filters <- auk_ebd(f_in_ebd, f_in_sampling) %>%
  auk_species(speciesOfInterest) %>%
  auk_country(country = "IN") %>%
  auk_state(c("IN-KL", "IN-TN", "IN-KA")) %>% # Restricting geography to TamilNadu, Kerala & Karnataka
  auk_date(c("2013-01-01", "2019-12-31")) %>%
  auk_complete()

# check filters
ebd_filters
```

Below code need not be run if it has been filtered once already and the above path leads to the right dataset. NB: This is a computation heavy process, run with caution.

```
# specify output location and perform filter
f_out_ebd <- "data/01_ebird-filtered-EBD-westernGhats.txt"
f_out_sampling <- "data/01_ebird-filtered-sampling-westernGhats.txt"
```

```

1 ebd_filtered <- auk_filter(ebd_filters,
2   file = f_out_ebd,
3   file_sampling = f_out_sampling, overwrite = TRUE
4 )

```

95 2.3 Process filtered data

96 The data has been filtered above using the auk functions. We will now work with the filtered checklist observations (Please
 97 note that we have not yet spatially filtered the checklists to the confines of our study area, which is the Nilgiris and the
 98 Anamalai hills. This step is carried out further on).

```

1 # read in the data
2 ebd <- read_ebd(f_out_ebd)

```

99 eBird checklists only suggest whether a species was reported at a particular location. To arrive at absence data, we use a
 100 process known as zero-filling (Johnston et al., 2019), wherein a new dataframe is created with a 0 marked for each checklist
 101 when the bird was not observed.

```

1 # fill zeroes
2 zf <- auk_zerofill(f_out_ebd, f_out_sampling)
3 new_zf <- collapse_zerofill(zf)

```

102 Let us now choose specific columns necessary for further analysis.

```

1 # choose columns of interest
2 columnsOfInterest <- c(
3   "checklist_id", "scientific_name", "common_name",
4   "observation_count", "locality", "locality_id",
5   "locality_type", "latitude", "longitude",
6   "observation_date", "time_observations_started",
7   "observer_id", "sampling_event_identifier",
8   "protocol_type", "duration_minutes",
9   "effort_distance_km", "effort_area_ha",
10  "number_observers", "species_observed",
11  "reviewed")
12
13 # make list of presence and absence data and choose cols of interest
14 data <- list(ebd, new_zf) %>%
15   map(function(x) {
16     x %>% select(one_of(columnsOfInterest))
17   })
18
19 # remove zerofills to save working memory
20 rm(zf, new_zf)
21 gc()
22
23 # check for presences and absence in absences df, remove essentially the presences df which may lead to erroneous a
24 data[[2]] <- data[[2]] %>% filter(species_observed == F)

```

103 2.4 Spatial filter

104 A spatial filter is now supplied to further restrict our list of observations to the confines of the Nilgiris and the Anamalai
 105 hills of the Western Ghats biodiversity hotspot.

```

1 # load shapefile of the study area
2 library(sf)
3 hills <- st_read("data/spatial/hillsShapefile/Nil_Ana_Pal.shp")

```

```

4
5 # write a prelim filter by bounding box
6 box <- st_bbox(hills)
7
8 # get data spatial coordinates
9 dataLocs <- data %>%
10   map(function(x) {
11     select(x, longitude, latitude) %>%
12       filter(between(longitude, box["xmin"], box["xmax"]) &
13         between(latitude, box["ymin"], box["ymax"]))
14   }) %>%
15   bind_rows() %>%
16   distinct() %>%
17   st_as_sf(coords = c("longitude", "latitude")) %>%
18   st_set_crs(4326) %>%
19   st_intersection(hills)
20
21 # get simplified data and drop geometry
22 dataLocs <- mutate(dataLocs, spatialKeep = T) %>%
23   bind_cols(., as_tibble(st_coordinates(dataLocs))) %>%
24   st_drop_geometry()
25
26 # bind to data and then filter
27 data <- data %>%
28   map(function(x) {
29     left_join(x, dataLocs, by = c("longitude" = "X", "latitude" = "Y")) %>%
30     filter(spatialKeep == T) %>%
31     select(-Id, -spatialKeep)
32   })
33
106 Save temporary data created so far.
34
35 # save a temp data file
36 save(data, file = "data/01_data_temp.rdata")

```

2.5 Handle presence data

Further pre-processing is required in the case of many checklists where species abundance is often unknown and an 'X' is denoted in such cases. Here, we convert all 'X' notations to a 1, suggesting a presence (as we are not concerned with abundance data in this analysis). We also removed those checklists where the duration in minutes is either not recorded or listed as zero. Lastly, we added an sampling effort based filter following Johnston et al., 2019, wherein we considered only those checklists with duration in minutes is less than 300 and distance in kilometres traveled is less than 5km. Lastly, we excluded those group checklists where the number of observers was greater than 10.

```

37 # in the first set, replace X, for presences, with 1
38 data[[1]] <- data[[1]] %>%
39   mutate(observation_count = ifelse(observation_count == "X",
40     "1", observation_count
41   ))
42
43 # remove records where duration is 0
44 data <- map(data, function(x) filter(x, duration_minutes > 0))
45
46 # group data by site and sampling event identifier
47 # then, summarise relevant variables as the sum
48 dataGrouped <- map(data, function(x) {

```

```

13 x %>%
14   group_by(sampling_event_identfier) %>%
15   summarise_at(
16     vars(
17       duration_minutes, effort_distance_km,
18       effort_area_ha
19     ),
20     list(sum.no.na)
21   )
22 })
23
24 # bind rows combining data frames, and filter
25 dataGrouped <- bind_rows(dataGrouped) %>%
26   filter(
27     duration_minutes <= 300,
28     effort_distance_km <= 5,
29     effort_area_ha <= 500
30   )
31
32 # get data identifiers, such as sampling identifier etc
33 dataConstants <- data %>%
34   bind_rows() %>%
35   select(
36     sampling_event_identfier, time_observations_started,
37     locality, locality_type, locality_id,
38     observer_id, observation_date, scientific_name,
39     observation_count, protocol_type, number_observers,
40     longitude, latitude
41   )
42
43 # join the summarised data with the identifiers,
44 # using sampling_event_identfier as the key
45 dataGrouped <- left_join(dataGrouped, dataConstants,
46   by = "sampling_event_identfier"
47 )
48
49 # remove checklists or seis with more than 10 observers
50 count(dataGrouped, number_observers > 10) # count how many have 10+ obs
51 dataGrouped <- filter(dataGrouped, number_observers <= 10)

```

114 2.6 Add decimal time

115 We added a column where time is denoted in decimal hours since midnight.

```

1 # assign present or not, and get time in decimal hours since midnight
2 library(lubridate)
3 time_to_decimal <- function(x) {
4   x <- hms(x, quiet = TRUE)
5   hour(x) + minute(x) / 60 + second(x) / 3600
6 }
7
8 # will cause issues if using time obs started as a linear effect and not quadratic
9 dataGrouped <- mutate(dataGrouped,
10   pres_abs = observation_count >= 1,
11   decimalTime = time_to_decimal(time_observations_started)

```

```

12 )
13
14 # check class of dataGrouped, make sure not sf
15 assertthat::assert_that(!"sf" %in% class(dataGrouped))
116 The above data is saved to a file.
1
17 # save a temp data file
18 save(dataGrouped, file = "data/01_data_prelim_processing.rdata")

```

117 3 Preparing Environmental Predictors

118 In this script, we processed climatic and landscape predictors for occupancy modeling.

119 All climatic data was obtained from <https://chelsa-climate.org/bioclim/> All landscape data was derived from a high resolution Sentinel-2 composite image that was classified using Google Earth Engine. This code can be accessed in Section 2 of the Supplementary Material and here: <https://code.earthengine.google.com/ec69fc4ffad32a532b25202009243d42>.

122 The goal here is to resample all rasters so that they have the same resolution of 1km cells. We also tested for spatial autocorrelation among climatic predictors, which can be found in Section 3 of the Supplementary Material.

124 3.1 Prepare libraries

125 We load some common libraries for raster processing and define a custom mode function.

```

1
2 # load libs
3 library(raster)
4 library(stringi)
5 library(glue)
6 library(gdalUtils)
7 library(purrr)
8
9 # prep mode function to aggregate
10 funcMode <- function(x, na.rm = T) {
11   ux <- unique(x)
12   ux[which.max(tabulate(match(x, ux)))]
13 }
14
15 # a basic test
16 assertthat::assert_that(funcMode(c(2, 2, 2, 2, 3, 3, 3, 4)) == as.character(2),
17   msg = "problem in the mode function"
18 ) # works

```

126 3.2 Prepare spatial extent

127 We prepare a 30km buffer around the boundary of the study area. This buffer will be used to mask the landscape rasters. The buffer procedure is done on data transformed to the UTM 43N CRS to avoid distortions.

```

1 # load hills
2 library(sf)
3 hills <- st_read("data/spatial/hillsShapefile/Nil_Ana_Pal.shp")
4 hills <- st_transform(hills, 32643)
5 buffer <- st_buffer(hills, 3e4) %>%
6   st_transform(4326)

```


3.3 Prepare terrain rasters

We prepare the elevation data which is an SRTM raster layer, and derive the slope and aspect from it after cropping it to the extent of the study site buffer. Please download the latest version of the SRTM raster layer from <https://www.worldclim.org/data/worldclim21.html>

```
1 # load elevation and crop to hills size, then mask by hills
2 alt <- raster("data/spatial/Elevation/alt") # this layer is not added to github as a result of its large size and c
3 alt.hills <- crop(alt, as(buffer, "Spatial"))
4 rm(alt)
5 gc()
6
7 # get slope and aspect
8 slopeData <- terrain(x = alt.hills, opt = c("slope", "aspect"))
9 elevData <- raster::stack(alt.hills, slopeData)
10 rm(alt.hills)
11 gc()
```

3.4 Prepare CHELSA rasters

We prepare the CHELSA rasters for annual temperature and annual precipitation in the same way, reading them in, cropping them to the study site buffer extent, and handling the temperature layer values which we divide by 10. The CHELSA rasters can be downloaded from <https://chelsa-climate.org/bioclim/>

```
1 # list chelsa files
2 # the chelsa data for Annual mean temperature and annual precipitation can be downloaded from the aforementioned li
3 chelsaFiles <- list.files("data/chelsa/",
4   full.names = TRUE,
5   pattern = "*.tif"
6 )
7
8 # gather chelsa rasters
9 chelsaData <- purrr::map(chelsaFiles, function(chr) {
10   a <- raster(chr)
11   crs(a) <- crs(elevData)
12   a <- crop(a, as(buffer, "Spatial"))
13   return(a)
14 })
15
16 # divide temperature by 10
17 chelsaData[[1]] <- chelsaData[[1]] / 10
18
19 # stack chelsa data
20 chelsaData <- raster::stack(chelsaData)
```

We stack the terrain and climatic rasters.

```
1 # stack rasters for efficient reprojection later
2 env_data <- stack(elevData, chelsaData)
```

3.5 Resample landcover from 10m to 1km

We read in a land cover classified image and resample that using the mode function to a 1km resolution. Please note that the resampling process need not be carried out as it has been done already and the resampled raster can be loaded with the subsequent code chunk.

The classified image can be obtained from: <https://code.earthengine.google.com/ec69fc4ffad32a532b25202009243d42>.

```

1 # read in landcover raster location
2 landcover <- "data/landUseClassification/classifiedImage-UTM.tif"
3
4 # get extent
5 e <- bbox(raster(landcover))
6
7 # init resolution
8 res_init <- res(raster(landcover))
9
10 # res to transform to 1000m
11 res_final <- res_init * 100
12
13 # use gdalutils gdalwarp for resampling transform
14 # to 1km from 10m
15 gdalUtils::gdalwarp(
16   srcfile = landcover,
17   dstfile = "data/landUseClassification/lc_01000m.tif",
18   tr = c(res_final), r = "mode", te = c(e)
19 )

```

143 We compare the frequency of landcover classes between the original 10m and the resampled 1km raster to be certain that
 144 the resampling has not resulted in drastic misrepresentation of the frequency of any landcover type. This comparison is
 145 made using the figure below.

146 3.6 Resample other rasters to 1km

147 We now resample all other rasters to a resolution of 1km.

148 3.6.1 Read in resampled landcover

149 Here, we read in the 1km landcover raster and set 0 to NA.

```

1 lc_data <- raster("data/landUseClassification/lc_01000m.tif")
2 lc_data[lc_data == 0] <- NA

```

150 3.6.2 Reproject environmental data using landcover as a template

```

1
2 # resample to the corresponding landcover data
3 env_data_resamp <- projectRaster(
4   from = env_data, to = lc_data,
5   crs = crs(lc_data), res = res(lc_data)
6 )
7
8 # export as raster stack
9 land_stack <- stack(env_data_resamp, lc_data)
10
11 # get names
12 land_names <- glue('data/spatial/landscape_resamp{c("01")}km.tif')
13
14 # write to file
15 writeRaster(land_stack, filename = as.character(land_names), overwrite = TRUE)

```

3.7 Temperature and rainfall in relation to elevation

3.7.1 Prepare libraries and CI function

```
1 # load libs
2 library(dplyr)
3 library(tidyr)
4
5 library(scales)
6 library(ggplot2)
7
8 # get ci func
9 ci <- function(x) {
10   qnorm(0.975) * sd(x, na.rm = T) / sqrt(length(x))
11 }
```

3.7.2 Load resampled environmental rasters

```
1 # read landscape prepare for plotting
2 landscape <- stack("data/spatial/landscape_resamp01_km.tif")
3
4 # get proper names
5 elev_names <- c("elev", "slope", "aspect")
6 chelsa_names <- c("bio_01", "bio_12")
7
8 names(landscape) <- as.character(glue('{c(elev_names, chelsa_names, "landcover")}'))
9
10 # make duplicate stack
11 land_data <- landscape[[c("elev", chelsa_names)]]
12
13 # convert to list
14 land_data <- as.list(land_data)
15
16 # map get values over the stack
17 land_data <- purrr::map(land_data, getValues)
18 names(land_data) <- c("elev", chelsa_names)
19
20 # conver to dataframe and round to 100m
21 land_data <- bind_cols(land_data)
22 land_data <- drop_na(land_data) %>%
23   mutate(elev_round = plyr::round_any(elev, 200)) %>%
24   dplyr::select(-elev) %>%
25   pivot_longer(
26     cols = contains("bio"),
27     names_to = "clim_var"
28   ) %>%
29   group_by(elev_round, clim_var) %>%
30   summarise_all(.funs = list(~ mean(.), ~ ci(.)))
```

Figure code is hidden in versions rendered as HTML or PDF.

3.8 Land cover type in relation to elevation

```
1 # get data from landscape rasters
2 lc_elev <- tibble(
3   elev = getValues(landscape[["elev"]]),
```

```

4   landcover = getValues(landscape[["landcover"]])
5 )
6
7 # process data for proportions
8 lc_elev <- lc_elev %>%
9   filter(!is.na(landcover), !is.na(elev)) %>%
10  mutate(elev = plyr::round_any(elev, 100)) %>%
11  count(elev, landcover) %>%
12  group_by(elev) %>%
13  mutate(prop = n / sum(n))
14
15 # fill out lc elev
16 lc_elev_canon <- crossing(
17   elev = unique(lc_elev$elev),
18   landcover = unique(lc_elev$landcover)
19 )
20
21 # bind with lcelev
22 lc_elev <- full_join(lc_elev, lc_elev_canon)
23
24 # convert NA to zero
25 lc_elev <- replace_na(lc_elev, replace = list(n = 0, prop = 0))

```

Figure code is hidden in versions rendered as HTML and PDF.

3.9 Main Text Figure 2

4 Preparing Observer Expertise Scores

Differences in local avifaunal expertise among citizen scientists can lead to biased species detection when compared with data collected by a consistent set of trained observers (van Strien et al., 2013). Including observer expertise as a detection covariate in occupancy models using eBird data can help account for this variation (Johnston et al., 2018). Observer-specific expertise in local avifauna was calculated following (Kelling et al., 2015) as the normalized predicted number of species reported by an observer after 60 minutes of sampling across the most common land cover type within the study area. This score was calculated by examining checklists from anonymized observers across the study area. We modified (Kelling et al., 2015) formulation by including only observations of the 93 species of interest in our calculations. An observer with a higher number of species of interest reported within 60 minutes would have a higher observer-specific expertise score, with respect to the study area.

Plots with respect to how observer expertise varied over time (2013-2019) for the list of species considered in this study (across the study area alone) can be accessed in Section 5 of the Supplementary Material.

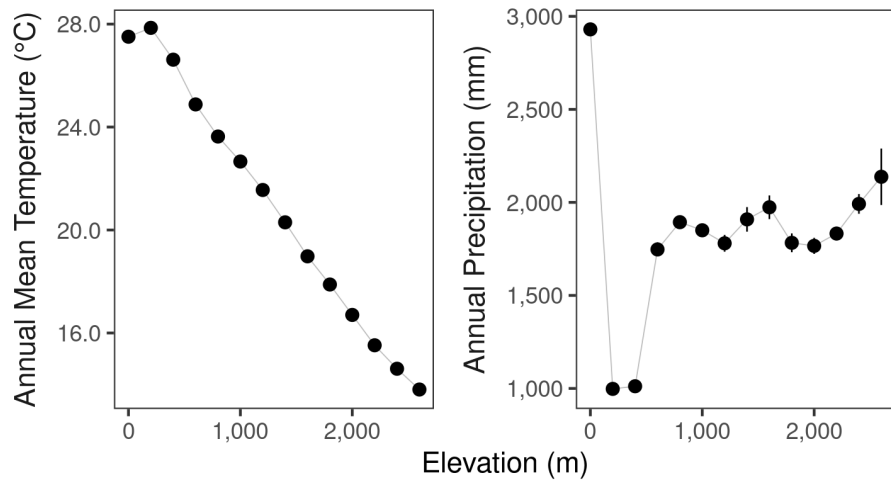
4.1 Prepare libraries

```

1
2 # load libs
3 library(data.table)
4 library(readxl)
5 library(magrittr)
6 library(stringr)
7 library(dplyr)
8 library(tidyr)
9 library(auk)
10
11 # get decimal time function

```

(a)



(b)

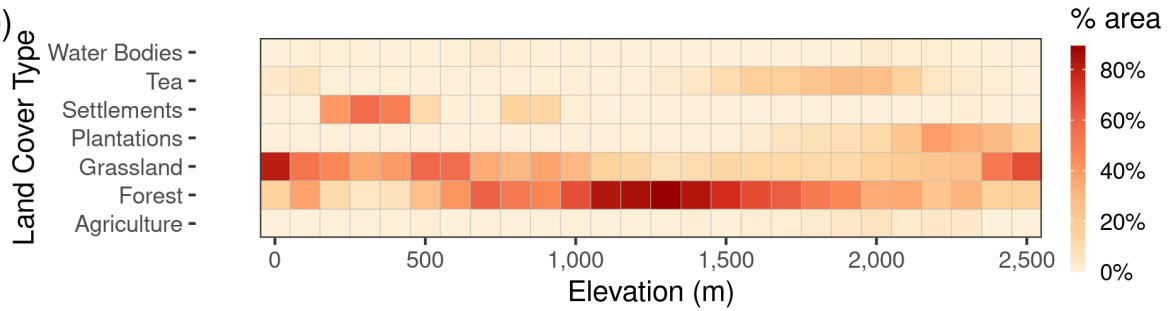


Figure 2: Annual Mean Temperature varied from ~28C in the plains to <14C at higher elevations. Annual precipitation increased at lower elevations (in the plains) to ~3000mm and ranged between 1500mm and 2200mm at mid- and high elevations across the Nilgiri and the Anamalai hills. (b) The proportion of land cover types varied across the study area as shown in this panel (1 = agriculture; 2 = forests; 3 = grasslands; 4 = plantations; 5 = settlements; 6 = tea; 7 = water bodies).

```

12 library(lubridate)
13 time_to_decimal <- function(x) {
14   x <- lubridate::hms(x, quiet = TRUE)
15   lubridate::hour(x) + lubridate::minute(x) / 60 + lubridate::second(x) / 3600
16 }

```

171 4.2 Prepare data

172 Here, we go through the data preparation process again because we might want to assess observer expertise over a larger
 173 area than the study site.

```

1 # Read in shapefile of study area to subset by bounding box
2 library(sf)
3 wg <- st_read("data/spatial/hillsShapefile/Nil_Ana_Pal.shp") %>%
4   st_transform(32643)
5
6 # set file paths for auk functions
7 f_in_ebd <- file.path("data/01_ebird-filtered-EBD-westernGhats.txt")
8 f_in_sampling <- file.path("data/01_ebird-filtered-sampling-westernGhats.txt")
9
10 # run filters using auk packages
11 ebd_filters <- auk_ebd(f_in_ebd, f_in_sampling) %>%
12   auk_country(country = "IN") %>%
13   auk_state(c("IN-KL", "IN-TN", "IN-KA")) %>%
14   # Restricting geography to TamilNadu, Kerala & Karnataka
15   auk_date(c("2013-01-01", "2019-12-31")) %>%
16   auk_complete()
17
18 # check filters
19 ebd_filters
20
21 # specify output location and perform filter
22 f_out_ebd <- "data/ebird_for_expertise.txt"
23 f_out_sampling <- "data/ebird_sampling_for_expertise.txt"
24
25 ebd_filtered <- auk_filter(ebd_filters,
26   file = f_out_ebd,
27   file_sampling = f_out_sampling, overwrite = TRUE
28 )

```

174 Load in the filtered data and columns of interest.

```

1 ## Process filtered data
2 # read in the data
3 ebd <- fread(f_out_ebd)
4 names <- names(ebd) %>%
5   stringr::str_to_lower() %>%
6   stringr::str_replace_all(" ", "_")
7
8 setnames(ebd, names)
9 # choose columns of interest
10 columnsOfInterest <- c(
11   "checklist_id", "scientific_name", "observation_count",
12   "locality", "locality_id", "locality_type", "latitude",
13   "longitude", "observation_date",
14   "time_observations_started", "observer_id",

```

```

15   "sampling_event_identifler", "protocol_type",
16   "duration_minutes", "effort_distance_km", "effort_area_ha",
17   "number_observers", "species_observed", "reviewed"
18 )
19
20 ebd <- setDF(ebd) %>%
21   as_tibble() %>%
22   dplyr::select(one_of(columnsOfInterest))
23
24 setDT(ebd)

```

175 4.3 Spatially explicit filter on checklists

```

1  # get checklist locations
2  ebd_locs <- ebd[, .(longitude, latitude)]
3  ebd_locs <- setDF(ebd_locs) %>% distinct()
4  ebd_locs <- st_as_sf(ebd_locs,
5    coords = c("longitude", "latitude")
6  ) %>%
7    `st_crs`<-`(4326) %>%
8    bind_cols(as_tibble(st_coordinates(.))) %>%
9    st_transform(32643) %>%
10   mutate(id = 1:nrow(.))
11
12 # check whether to include
13 to_keep <- unlist(st_contains(wg, ebd_locs))
14
15 # filter locs
16 ebd_locs <- filter(ebd_locs, id %in% to_keep) %>%
17   bind_cols(as_tibble(st_coordinates(st_as_sf(.)))) %>%
18   st_drop_geometry()
19
20 ebd <- ebd[longitude %in% ebd_locs$X & latitude %in% ebd_locs$Y, ]

```

176 4.4 Prepare species of interest

```

1  # read in species list
2  specieslist <- read.csv("data/species_list.csv")
3
4  # set species of interest
5  soi <- specieslist$scientific_name
6
7  ebdSpSum <- ebd[, .(
8    nSp = .N,
9    totSoiSeen = length(intersect(scientific_name, soi))
10  ),
11  by = list(sampling_event_identifler)
12  ]
13
14 # write to file and link with checklist id later
15 fwrite(ebdSpSum, file = "data/03_data-nspp-per-chk.csv")

```

4.5 Prepare checklists for observer score

```
# 1. add new columns of decimal time and julian date
ebd[, `:=`(
  decimalTime = time_to_decimal(time_observations_started),
  julianDate = yday(as.POSIXct(observation_date))
)]

ebdEffChk <- setDF(ebd) %>%
  mutate(year = year(observation_date)) %>%
  distinct(
    sampling_event_identifider, observer_id,
    year,
    duration_minutes, effort_distance_km, effort_area_ha,
    longitude, latitude,
    locality, locality_id,
    decimalTime, julianDate, number_observers
  ) %>%
  # drop rows with NAs in cols used in the model
  tidyr::drop_na(
    sampling_event_identifider, observer_id,
    duration_minutes, decimalTime, julianDate
  ) %>%
  # drop years below 2013
  filter(year >= 2013)

# 3. join to covariates and remove large groups (> 10)
ebdChkSummary <- inner_join(ebdEffChk, ebdSpSum)

# remove ebird data
rm(ebd)
gc()
```

4.6 Get landcover

Read in land cover type data resampled at 1km resolution.

```
# read in 1km landcover and set 0 to NA
library(raster)
landcover <- raster::raster("data/landUseClassification/lc_01000m.tif")
landcover[landcover == 0] <- NA

# get locs in utm coords
locs <- distinct(
  ebdChkSummary, sampling_event_identifider, longitude, latitude,
  locality, locality_id
)
locs <- st_as_sf(locs, coords = c("longitude", "latitude")) %>%
  `st_crs<-`(4326) %>%
  st_transform(32643) %>%
  st_coordinates()

# get for unique points
landcoverVec <- raster::extract(
```



```

18   x = landcover,
19   y = locs
20 )
21
22 # assign to df and overwrite
23 setDT(ebdChkSummary)[, landcover := landcoverVec]

```

180 4.7 Filter checklist data

```

1  # change names for easy handling
2  setnames(ebdChkSummary, c(
3    "sei", "observer", "year", "duration", "distance",
4    "area", "longitude", "latitude", "locality",
5    "locality_id", "decimalTime",
6    "julianDate", "nObs", "nSp", "nSoi", "landcover"
7  ))
8
9  # count data points per observer
10 obscount <- count(ebdChkSummary, observer) %>%
11   filter(n >= 10)
12
13 # make factor variables and remove obs not in obscount
14 # also remove 0 durations
15 ebdChkSummary <- ebdChkSummary %>%
16   mutate(
17     distance = ifelse(is.na(distance), 0, distance),
18     duration = if_else(is.na(duration), 0.0, as.double(duration))
19   ) %>%
20   filter(
21     observer %in% obscount$observer,
22     duration > 0,
23     duration <= 300,
24     nSoi >= 0,
25     distance <= 5,
26     !is.na(nSoi)
27   ) %>%
28   mutate(
29     landcover = as.factor(landcover),
30     observer = as.factor(observer)
31   ) %>%
32   drop_na(landcover)
33
34
35 # save to file for later reuse
36 fwrite(ebdChkSummary, file = "data/03_data-covars-perChklist.csv")

```

181 4.8 Model observer expertise

182 Our observer expertise model aims to include the random intercept effect of observer identity, with a random slope effect
 183 of duration. This models the different rate of species accumulation by different observers, as well as their different starting
 184 points.

```

1  # uses either a subset or all data
2  library(lmerTest)
3

```

```

4 # here we specify a glmm with random effects for observer
5 # time is considered a fixed log predictor and a random slope
6 modObsExp <- glmer(nSoi ~ sqrt(duration) +
7   landcover +
8   sqrt(decimalTime) +
9   I((sqrt(decimalTime))^2) +
10  log(julianDate) +
11  I((log(julianDate)^2)) +
12  (1 | observer) + (0 + duration | observer),
13  data = ebdChkSummary, family = "poisson"
14 )

1 # make dir if absent
2 if (!dir.exists("data/modOutput")) {
3   dir.create("data/modOutput")
4 }

5
6 # write model output to text file
7 {
8   writeLines(R.utils::captureOutput(list(Sys.time(), summary(modObsExp))),
9     con = "data/modOutput/03_model-output-expertise.txt"
10  )
11 }

1 # make df with means
2 observer <- unique(ebdChkSummary$observer)
3
4 # predict at 60 mins on the most common landcover
5 dfPredict <- ebdChkSummary %>%
6   summarise_at(vars(duration, decimalTime, julianDate), list(~ mean(.))) %>%
7   mutate(duration = 60, landcover = as.factor(6)) %>%
8   tidyr::crossing(observer)
9
10 # run predict from model on it
11 dfPredict <- mutate(dfPredict,
12   score = predict(modObsExp,
13     newdata = dfPredict,
14     type = "response",
15     allow.new.levels = TRUE
16   )
17 ) %>%
18   mutate(score = scales::rescale(score))

1 fwrite(dfPredict %>% dplyr::select(observer, score),
2   file = "data/03_data-obsExpertise-score.csv"
3 )

```

185 5 Examining Spatial Sampling Bias

186 The goal of this section is to determine how far each checklist location is from the nearest road, and how far each site is from
 187 its nearest neighbour. This involves finding the pairwise distance between a large number of unique checklist locations to
 188 a vast number of roads, as well as to each other.

189 Parts of this section are thus implemented in Python, using the R-Python interface package, reticulate.

5.1 Prepare libraries

```
# load libraries
library(reticulate)
library(sf)
library(dplyr)
library(scales)
library(readr)
library(purrr)

library(ggplot2)
library(ggthemes)
library(ggspatial)
library(scico)

# round any function
round_any <- function(x, accuracy = 20000) {
  round(x / accuracy) * accuracy
}

# ci function
ci <- function(x) {
  qnorm(0.975) * sd(x, na.rm = TRUE) / sqrt(length(x))
}

# set python path
use_python("/usr/bin/python3")
```

Importing python libraries. These libraries need to be installed before use.

```
# import classic python libs
import itertools
from operator import itemgetter
import numpy as np
import matplotlib.pyplot as plt
import math

# libs for dataframes
import pandas as pd

# import libs for geodata
from shapely.ops import nearest_points
import geopandas as gpd
import rasterio

# import ckdtee
from scipy.spatial import cKDTree
from shapely.geometry import Point, MultiPoint, LineString, MultiLineString
```

5.2 Prepare data for processing

First we read in the roads shapefile, which is obtained from the Open Street Map database. Then we read in the checklist covariates, and extract the unique coordinate pairs. All data are reprojected to be in the UTM 43N coordinate system.

We define a custom Python function to separate multi-feature geometries (here, roads which are in parts) into single feature geometries. Then we define a function to use the K-dimensional trees method from scipy to find the distance between two geometries, here, the distance between the locations and the nearest road. We define another function to find the distance

198 between checklist locations and all other checklist locations.
199 We use these functions to find the distance between each checklist location and the nearest road and the next nearest site.

200 5.2.1 Python functions and distance calculations

```
1  # read in roads shapefile
2  roads = gpd.read_file("data/spatial/roads_studysite_2019/roads_studysite_2019.shp")
3  roads.head()
4
5  # read in checklist covariates for conversion to gpd
6  # get unique coordinates, assign them to the df
7  # convert df to geo-df
8  chkCovars = pd.read_csv("data/03_data-covars-perChklist.csv")
9  unique_locs = chkCovars.drop_duplicates(subset=['longitude',
10                                             'latitude'])[['longitude', 'latitude']]
11  unique_locs['coordId'] = np.arange(1, unique_locs.shape[0]+1)
12  chkCovars = chkCovars.merge(unique_locs, on=['longitude', 'latitude'])
13
14  unique_locs = gpd.GeoDataFrame(
15  unique_locs,
16  geometry = gpd.points_from_xy(unique_locs.longitude, unique_locs.latitude))
17  unique_locs.crs = {'init': 'epsg:4326'}
18
19  # reproject spatial to 43n epsg 32643
20
21  roads = roads.to_crs({'init': 'epsg:32643'})
22  unique_locs = unique_locs.to_crs({'init': 'epsg:32643'})
23
24  # function to simplify multilinestrings
25  def simplify_roads(complex_roads):
26      simpleRoads = []
27      for i in range(len(complex_roads.geometry)):
28          feature = complex_roads.geometry.iloc[i]
29          if feature.geom_type == "LineString":
30              simpleRoads.append(feature)
31          elif feature.geom_type == "MultiLineString":
32              for road_level2 in feature:
33                  simpleRoads.append(road_level2)
34      return simpleRoads
35
36  # function to use ckdttrees to find the nearest road
37  def ckdnearest(gdfA, gdfB):
38      A = np.concatenate(
39      [np.array(geom.coords) for geom in gdfA.geometry.to_list()])
40      simplified_features = simplify_roads(gdfB)
41      B = [np.array(geom.coords) for geom in simplified_features]
42      B = np.concatenate(B)
43      ckd_tree = cKDTree(B)
44      dist, idx = ckd_tree.query(A, k=1)
45      return dist
46
47  # function to use ckdttrees for nearest other checklist point
48  def ckdnearest_point(gdfA, gdfB):
49      A = np.concatenate(
```

```

50     [np.array(geom.coords) for geom in gdfA.geometry.to_list()])
51     #simplified_features = simplify_roads(gdfB)
52     B = np.concatenate(
53     [np.array(geom.coords) for geom in gdfB.geometry.to_list()])
54     #B = np.concatenate(B)
55     ckd_tree = cKDTree(B)
56     dist, idx = ckd_tree.query(A, k=[2])
57     return dist
58
59 # get distance to nearest road
60 unique_locs['dist_road'] = ckdnearest(unique_locs, roads)
61
62 # get distance to nearest other site
63 unique_locs['nnb'] = ckdnearest_point(unique_locs, unique_locs)
64
65 # write to file
66 unique_locs = pd.DataFrame(unique_locs.drop(columns='geometry'))
67 unique_locs['dist_road'] = unique_locs['dist_road']
68 unique_locs['nnb'] = unique_locs['nnb']
69 unique_locs.to_csv(path_or_buf = "data/locs_dist_to_road.csv", index=False)
70
71 # merge unique locs with chkCovars
72 chkCovars = chkCovars.merge(unique_locs, on=['latitude',
73                                     'longitude', 'coordId'])

```

201 5.2.2 Spatially explicit filter on checklists

202 We filter the checklists by the boundary of the study area. This is *not* the extent.

```

1 # extract data from python
2 chkCovars <- py$chkCovars
3 chkCovars <- st_as_sf(chkCovars, coords = c("longitude", "latitude")) %>%
4   `st_crs<-`(4326) %>%
5   st_transform(32643)
6
7 # read wg
8 wg <- st_read("data/spatial/hillsShapefile/Nil_Ana_Pal.shp") %>%
9   st_transform(32643)
10
11 # spatial subset
12 chkCovars <- chkCovars %>%
13   mutate(id = 1:nrow(.)) %>%
14   filter(id %in% unlist(st_contains(wg, chkCovars)))

```

203 5.3 Main Text Figure 3

204 5.3.1 Prepare histogram of distance to roads

205 Figure code is hidden in versions rendered as HTML or PDF.

206 5.3.2 Table: Distance to roads

```

1 # write the mean and ci95 to file
2 chkCovars %>%
3   st_drop_geometry() %>%
4   select(dist_road, nnb) %>%

```

```

5   tidyr::pivot_longer(
6     cols = c("dist_road", "nnb"),
7     names_to = "variable"
8   ) %>%
9   group_by(variable) %>%
10  summarise_at(
11    vars(value),
12    list(~ mean(.), ~ sd(.), ~ min(.), ~ max(.))
13  ) %>%
14  write_csv("data/results/distance_roads_sites.csv")

```

207 5.4 Distance to nearest neighbouring site

```

1   # get unique locations
2   locs <- py$unique_locs

```

208 Figure code is hidden in versions rendered as HTML and PDF.

209 5.5 Main Text Figure 3

```

1   roads <- st_read("data/spatial/roads_studysite_2019/roads_studysite_2019.shp") %>%
2     st_transform(32643)
3   points <- chkCovars %>%
4     bind_cols(as_tibble(st_coordinates(.))) %>%
5     st_drop_geometry() %>%
6     mutate(X = round_any(X, 2500), Y = round_any(Y, 2500))
7
8   points <- count(points, X, Y)
9
10  # add land
11  library(rnaturalearth)
12  land <- ne_countries(
13    scale = 50, type = "countries", continent = "asia",
14    country = "india",
15    returnclass = c("sf")
16  ) %>%
17    st_transform(32643)
18
19  bbox <- st_bbox(wg)

```

210 Figure code is hidden in versions rendered as HTML and PDF.

211 6 Adding Covariates to Checklist Data

212 In this section, we prepare a final list of covariates, after taking into account spatial sampling bias (examined in the previous
213 section), temporal bias and observer expertise scores.

214 6.1 Prepare libraries and data

```

1
2   # load libs
3   library(dplyr)
4   library(readr)
5   library(stringr)
6   library(purrr)

```

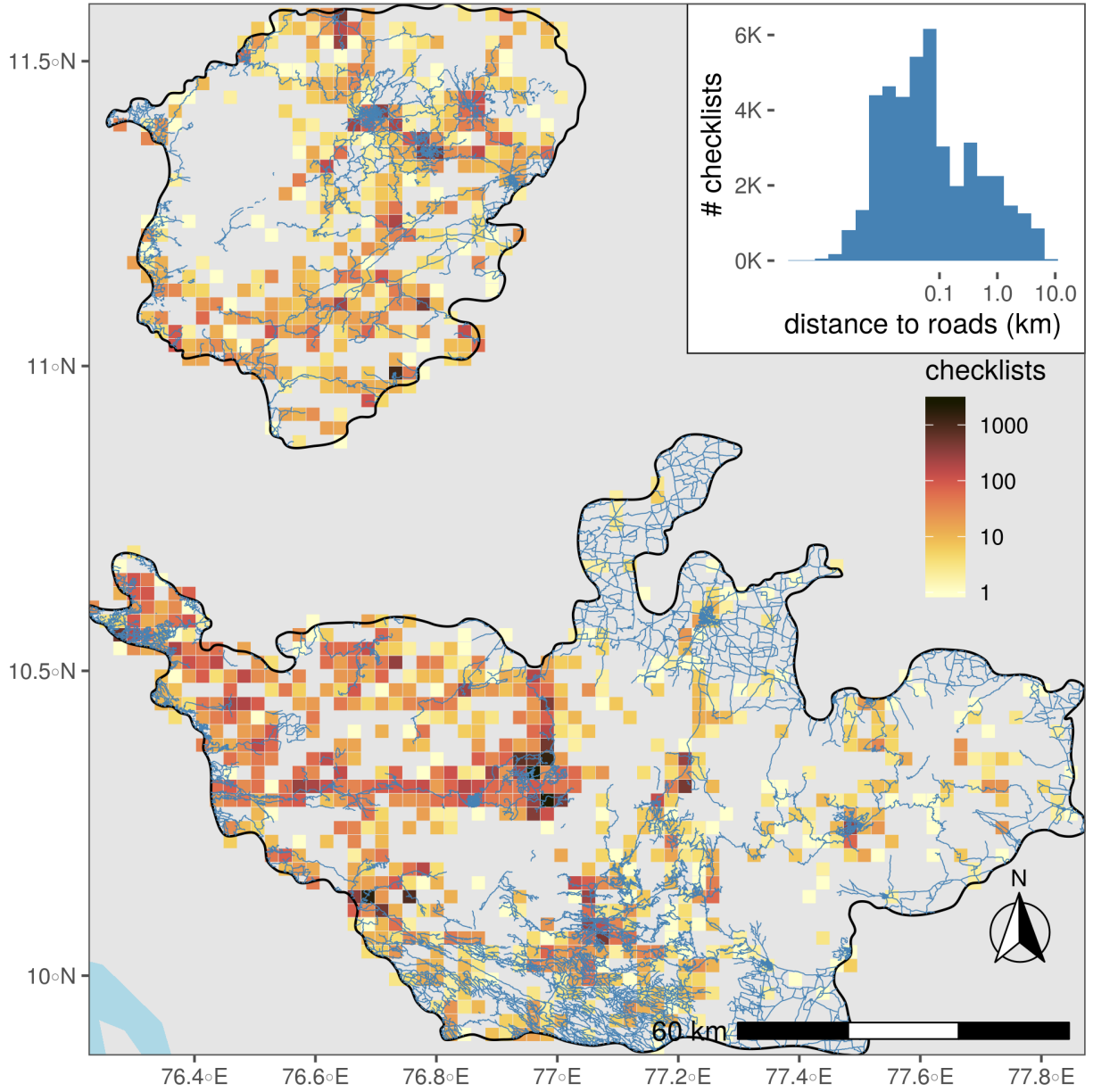


Figure 3: Spatial sampling bias of eBird observations across the Nilgiri and the Anamalai hills. A large proportion of localities/sites were next to roads and were on average only ~300m from another locality/site. Each cell here is 2.5km x 2.5km

```

7 library(raster)
8 library(glue)
9 library(velox)
10 library(tidyr)
11 library(sf)
12
13 # load saved data object
14 load("data/01_ebird_data_prelim_processing.rdata")

```

215 6.2 Spatial subsampling

216 Sampling bias can be introduced into citizen science due to the often ad-hoc nature of data collection (Sullivan et al.,
217 2014). For eBird, this translates into checklists reported when convenient, rather than at regular or random points in time
218 and space, leading to non-independence in the data if observations are spatio-temporally clustered (Johnston et al., 2019).
219 Spatio-temporal autocorrelation in the data can be reduced by sub-sampling at an appropriate spatial resolution, and by
220 avoiding temporal clustering. We estimated two simple measures of spatial clustering: the distance from each site to the
221 nearest road (road data from OpenStreetMap; (OpenStreetMap contributors, 2017), and the nearest-neighbor distance for
222 each site. Sites were strongly tied to roads (mean distance to road \pm SD = 390.77 \pm 859.15 m; range = 0.28 m – 7.64 km)
223 and were on average only 297 m away from another site (SD = 553 m; range = 0.14 m – 12.85 km) (Figure 3). This analysis
224 was done in the previous section.

225 Here, to further reduce spatial autocorrelation, we divided the study area into a grid of 1km wide square cells and picked
226 checklists from one site at random within each grid cell.

227 Prior to running this analysis, we checked how many checklists/data would be retained given a particular value of distance
228 to account for spatial independence. This analysis can be accessed in Section 6 of the Supplementary Material. We show
229 that over 80% of checklists are retained with a distance cutoff of 1km. In addition, a number of thinning approaches were
230 tested to determine which method retained the highest proportion of points, while accounting for sampling effort (time and
231 distance). This analysis can be accessed in Section 7 of the Supplementary Material.

```

1 # grid based spatial thinning
2 gridsize <- 1000 # grid size in metres
3 effort_distance_max <- 1000 # removing checklists with this distance
4
5 # make grids across the study site
6 hills <- st_read("data/spatial/hillsShapefile/Nil_Ana_Pal.shp") %>%
7   st_transform(32643)
8 grid <- st_make_grid(hills, cellsize = gridsize)
9
10 # split data by species
11 data_spatial_thin <- split(x = dataGrouped, f = dataGrouped$scientific_name)
12
13 # spatial thinning on each species retains
14 # site with maximum visits per grid cell
15 data_spatial_thin <- map(data_spatial_thin, function(df) {
16
17   # count visits per locality
18   df <- group_by(df, locality) %>%
19     mutate(tot_effort = length(sampling_event_identfier)) %>%
20     ungroup()
21
22   # remove sites with distances above spatial independence
23   df <- df %>%
24     filter(effort_distance_km <= effort_distance_max) %>%
25     st_as_sf(coords = c("longitude", "latitude")) %>%
26     `st_crs`-<`(4326) %>%

```



```

27     st_transform(32643) %>%
28     mutate(coordId = 1:nrow(.)) %>%
29     bind_cols(as_tibble(st_coordinates(.)))
30
31   # whcih cell has which coords
32   grid_contents <- st_contains(grid, df) %>%
33     as_tibble() %>%
34     rename(cell = row.id, coordId = col.id)
35
36   # what's the max point in each grid
37   points_max <- left_join(df %>% st_drop_geometry(),
38     grid_contents,
39     by = "coordId"
40   ) %>%
41     group_by(cell) %>%
42     filter(tot_effort == max(tot_effort))
43
44   return(points_max)
45 })
46
47 # remove old data
48 rm(dataGrouped)

```

232 6.3 Temporal subsampling

233 Additionally, from each selected site, we randomly selected a maximum of 10 checklists, which reduced temporal autocor-
 234 relation.

```

1   # subsample data for random 10 observations
2   dataSubsample <- map(data_spatial_thin, function(df) {
3     df <- ungroup(df)
4     df_to_locality <- split(x = df, f = df$locality)
5     df_samples <- map_if(
6       .x = df_to_locality,
7       .p = function(x) {
8         nrow(x) > 10
9       },
10      .f = function(x) sample_n(x, 10, replace = FALSE)
11    )
12
13    return(bind_rows(df_samples))
14  })
15
16 # bind all rows for data frame
17 dataSubsample <- bind_rows(dataSubsample)
18
19 # remove previous data
20 rm(data_spatial_thin)

```

235 6.4 Add checklist calibration index

236 Load the CCI computed in the previous section. The CCI was the lone observer's expertise score for single-observer
 237 checklists, and the highest expertise score among observers for group checklists.

```

1 # read in obs score and extract numbers
2 expertiseScore <- read_csv("data/03_data-obsExpertise-score.csv") %>%
3   mutate(numObserver = str_extract(observer, "\\d+")) %>%
4   dplyr::select(-observer)
5
6 # group seis consist of multiple observers
7 # in this case, seis need to have the highest expertise observer score
8 # as the associated covariate
9
10 # get unique observers per sei
11 dataSeiScore <- distinct(
12   dataSubsample, sampling_event_identifider,
13   observer_id
14 ) %>%
15   # make list column of observers
16   mutate(observers = str_split(observer_id, ",")) %>%
17   unnest(cols = c(observers)) %>%
18   # add numeric observer id
19   mutate(numObserver = str_extract(observers, "\\d+")) %>%
20   # now get distinct sei and observer id numeric
21   distinct(sampling_event_identifider, numObserver)
22
23 # now add expertise score to sei
24 dataSeiScore <- left_join(dataSeiScore, expertiseScore,
25   by = "numObserver"
26 ) %>%
27   # get max expertise score per sei
28   group_by(sampling_event_identifider) %>%
29   summarise(expertise = max(score))
30
31 # add to dataCovar
32 dataSubsample <- left_join(dataSubsample, dataSeiScore,
33   by = "sampling_event_identifider"
34 )
35
36 # remove data without expertise score
37 dataSubsample <- filter(dataSubsample, !is.na(expertise))

```

238 6.5 Add climatic and landscape covariates

239 Reload climate and land cover predictors prepared previously.

```

1
2 # list landscape covariate stacks
3 landscape_files <- "data/spatial/landscape_resamp01_km.tif"
4
5 # read in as stacks
6 landscape_data <- stack(landscape_files)
7
8 # get proper names
9 elev_names <- c("elev", "slope", "aspect")
10 chelsa_names <- c("bio1", "bio12")
11
12 names(landscape_data) <- as.character(glue('{c(elev_names, chelsa_names, "landcover")}'))

```

6.6 Spatial buffers around selected checklists

Every checklist on eBird is associated with a latitude and longitude. However, the coordinates entered by an observer may not accurately depict the location at which a species was detected. This can occur for two reasons: first, traveling checklists are often associated with a single location along the route travelled by observers; and second, checklist locations could be assigned to a ‘hotspot’ – a location that is marked by eBird as being frequented by multiple observers. In many cases, an observation might be assigned to a hotspot even though the observation was not made at the precise location of the hotspot (Praveen J, 2017). (Johnston et al., 2019) showed that a large proportion of observations occurred within a 3km grid, even for those checklists up to 5km in length. Hence to adjust for spatial precision, we considered a minimum radius of 2.5km around each unique locality when sampling environmental covariate values.

```
1 # assign neighbourhood radius in m
2 sample_radius <- 2.5 * 1e3
3
4 # get distinct points and make buffer
5 ebird_buff <- dataSubsample %>%
6   ungroup() %>%
7   distinct(X, Y) %>%
8   mutate(id = 1:nrow(.)) %>%
9   crossing(sample_radius) %>%
10  arrange(id) %>%
11  group_by(sample_radius) %>%
12  nest() %>%
13  ungroup()
14
15
16 # convert to spatial features
17 ebird_buff <- mutate(ebird_buff,
18   data = map2(
19     data, sample_radius,
20     function(df, rd) {
21       df_sf <- st_as_sf(df, coords = c("X", "Y"), crs = 32643) %>%
22         # add long lat
23         bind_cols(as_tibble(st_coordinates(.))) %>%
24         # rename(longitude = X, latitude = Y) %>%
25         # # transform to modis projection
26         # st_transform(crs = 32643) %>%
27         # buffer to create neighborhood around each point
28         st_buffer(dist = rd)
29     }
30   )
31 )
```

6.7 Spatial buffer-wide covariates

6.7.1 Mean climatic covariates

All climatic covariates are sampled by considering the mean values within a 2.5km radius as discussed above and prefixed “am_”.

```
1 # get area mean for all preds except landcover, which is the last one
2 env_area_mean <- purrr::map(ebird_buff$data, function(df) {
3   stk <- landscape_data[[-dim(landscape_data)[3]]] # removing landcover here
4   velstk <- velox(stk)
5   dextr <- velstk$extract(
6     sp = df, df = TRUE,
```

```

7     fun = function(x) mean(x, na.rm = T)
8   )
9
10  # assign names for joining
11  names(dextr) <- c("id", names(stk))
12  return(as_tibble(dextr))
13 })
14
15 # join to buffer data
16 ebird_buff <- ebird_buff %>%
17   mutate(data = map2(data, env_area_mean, inner_join, by = "id"))

```

253 6.7.2 Proportions of land cover type

254 All land cover covariates were sampled by considering the proportion of each land cover type within a 2.5km radius.

```

1  # get the last element of each stack from the list
2  # this is the landcover at that resolution
3  lc_area_prop <- purrr::map(ebird_buff$data, function(df) {
4    lc <- landscape_data[[dim(landscape_data)[3]]] # accessing landcover here
5    lc_velox <- velox(lc)
6    lc_vals <- lc_velox$extract(sp = df, df = TRUE)
7    names(lc_vals) <- c("id", "lc")
8
9    # get landcover proportions
10   lc_prop <- count(lc_vals, id, lc) %>%
11     group_by(id) %>%
12     mutate(
13       lc = glue('lc_{str_pad(lc, 2, pad = "0")}'),
14       prop = n / sum(n)
15     ) %>%
16     dplyr::select(-n) %>%
17     tidyr::pivot_wider(
18       names_from = lc,
19       values_from = prop,
20       values_fill = list(prop = 0)
21     ) %>%
22     ungroup()
23
24   return(lc_prop)
25 })
26
27 # join to data
28 ebird_buff <- ebird_buff %>%
29   mutate(data = map2(data, lc_area_prop, inner_join, by = "id"))

```

255 6.7.3 Link environmental covariates to checklists

```

1  # duplicate scale data
2  data_at_scale <- ebird_buff
3
4  # join the full data to landscape samples at each scale
5  data_at_scale$data <- map(data_at_scale$data, function(df) {
6    df <- st_drop_geometry(df)
7    df <- inner_join(dataSubsample, df, by = c("X", "Y"))

```

```

8     return(df)
9 })
256 Save data to file.

1 # write to file
2 pmap(data_at_scale, function(sample_radius, data) {
3     write_csv(data, path = glue('data/04_data-covars-{str_pad(sample_radius/1e3, 2, pad = "0")}km.csv'))
4     message(glue('export done: data/04_data-covars-{str_pad(sample_radius/1e3, 2, pad = "0")}km.csv'))
5 })

```

257 7 Modelling Species Occupancy

258 7.0.1 Load necessary libraries

```

1 # Load libraries
2 library(auk)
3 library(lubridate)
4 library(sf)
5 library(unmarked)
6 library(raster)
7 library(ebirdst)
8 library(MuMIn)
9 library(AICcmodavg)
10 library(fields)
11 library(tidyverse)
12 library(doParallel)
13 library(snow)
14 library(openxlsx)
15 library(data.table)
16 library(dplyr)
17 library(ecodist)
18
19 # Source necessary functions
20 source("code/fun_screen_cor.R")
21 source("code/fun_model_estimate_collection.r")

```

259 7.1 Load dataframe and scale covariates

260 Here, we load the required dataframe that contains 10 random visits to a site and environmental covariates that were prepared
261 at a spatial scale of 2.5 sq.km. We also scaled all covariates (mean around 0 and standard deviation of 1). Next, we ensured
262 that only Traveling and Stationary checklists were considered. Even though stationary counts have no distance traveled,
263 we defaulted all stationary accounts to an effective distance of 100m, which we consider the average maximum detection
264 radius for most bird species in our area. Following this, we excluded predictors with a Pearson coefficient >0.5 which
265 resulted in the removal of grasslands as it was highly negatively correlated with forests ($r = -0.77$).

266 Please note that species-specific plots of probabilities of occupancy as a function of environmental data can be accessed in
267 Section 8 of the Supplementary Material.

```

1 # Load in the prepared dataframe that contains 10 random visits to each site
2 dat <- fread("data/04_data-covars-2.5km.csv", header = T)
3 setDF(dat)
4 head(dat)
5
6 # Some more pre-processing to get the right data structures
7

```

```

8 # Ensuring that only Traveling and Stationary checklists were considered
9 names(dat)
10 dat <- dat %>% filter(protocol_type %in% c("Traveling", "Stationary"))
11
12 # We take all stationary counts and give them a distance of 100 m (so 0.1 km),
13 # as that's approximately the max normal hearing distance for people doing point counts.
14 dat <- dat %>%
15   mutate(effort_distance_km = replace(
16     effort_distance_km,
17     which(effort_distance_km == 0 &
18       protocol_type == "Stationary"),
19     0.1
20   ))
21
22 # Converting time observations started to numeric and adding it as a new column
23 # This new column will be minute_observations_started
24 dat <- dat %>%
25   mutate(min_obs_started = strtoi(as.difftime(time_observations_started,
26     format = "%H:%M:%S", units = "mins"
27   )))
28
29 # Adding the julian date to the dataframe
30 dat <- dat %>% mutate(julian_date = lubridate::yday(dat$observation_date))
31
32 # Removing other unnecessary columns from the dataframe and creating a clean one without the rest
33 names(dat)
34 dat <- dat[, -c(1, 4, 5, 16, 18, 21, 23, 24, 25, 26, 28:37, 39:45, 47)]
35
36 # Rename column names:
37 names(dat) <- c(
38   "duration_minutes", "effort_distance_km", "locality",
39   "locality_type", "locality_id", "observer_id",
40   "observation_date", "scientific_name", "observation_count",
41   "protocol_type", "number_observers", "pres_abs", "tot_effort",
42   "longitude", "latitude", "expertise", "bio_1.y", "bio_12.y",
43   "lc_02.y", "lc_06.y", "lc_01.y", "lc_07.y", "lc_04.y",
44   "lc_05.y", "min_obs_started", "julian_date"
45 )
46
47 dat.1 <- dat %>%
48   mutate(
49     year = year(observation_date),
50     pres_abs = as.integer(pres_abs)
51   ) # occupancy modeling requires an integer response
52
53 # Dividing Annual Mean Temperature by 10 to arrive at accurate values of temperature
54 dat.1$bio_1.y <- dat.1$bio_1.y / 10
55
56 # Scaling detection and occupancy covariates
57 dat.scaled <- dat.1
58 dat.scaled[, c(1, 2, 11, 16:25)] <- scale(dat.scaled[, c(1, 2, 11, 16:25)]) # Scaling and standardizing detection and occupancy covariates
59 fwrite(dat.scaled, file = "data/05_scaled-covars-2.5km.csv")
60
61 # Reload the scaled covariate data

```

```

62 dat.scaled <- fread("data/05_scaled-covars-2.5km.csv", header = T)
63 setDF(dat.scaled)
64 head(dat.scaled)
65
66 # Ensure observation_date column is in the right format
67 dat.scaled$observation_date <- format(
68   as.Date(
69     dat.scaled$observation_date,
70     "%m/%d/%Y"
71   ),
72   "%Y-%m-%d"
73 )
74
75 # Testing for correlations before running further analyses
76 # Most are uncorrelated since we decided to keep only 2 climatic and 6 land cover predictors
77 source("code/screen_cor.R")
78 names(dat.scaled)
79 screen.cor(dat.scaled[, c(1, 2, 11, 16:25)], threshold = 0.5)

```

268 7.2 Running a null model

```

1 # All null models are stored in lists below
2 all_null <- list()
3
4 # Add a progress bar for the loop
5 pb <- txtProgressBar(
6   min = 0,
7   max = length(unique(dat.scaled$scientific_name)),
8   style = 3
9 ) # text based bar
10
11 for (i in 1:length(unique(dat.scaled$scientific_name))) {
12   data <- dat.scaled %>%
13     filter(dat.scaled$scientific_name == unique(dat.scaled$scientific_name)[i])
14
15   # Preparing data for the unmarked model
16   occ <- filter_repeat_visits(data,
17     min_obs = 1, max_obs = 10,
18     annual_closure = FALSE,
19     n_days = 2600, # 7 years is considered a period of closure
20     date_var = "observation_date",
21     site_vars = c("locality_id")
22   )
23
24   obs_covs <- c(
25     "min_obs_started",
26     "duration_minutes",
27     "effort_distance_km",
28     "number_observers",
29     "protocol_type",
30     "expertise",
31     "julian_date"
32   )
33

```

```

34 # format for unmarked
35 occ_wide <- format_unmarked_occu(occ,
36   site_id = "site",
37   response = "pres_abs",
38   site_covs = c("locality_id", "lc_01.y", "lc_02.y", "lc_04.y",
39   "lc_05.y", "lc_06.y", "lc_07.y", "bio_1.y", "bio_12.y"),
40   obs_covs = obs_covs
41 )
42
43 # Convert this dataframe of observations into an unmarked object to start fitting occupancy models
44 occ_um <- formatWide(occ_wide, type = "unmarkedFrameOccu")
45
46 # Set up the model
47 all_null[[i]] <- occu(~1 ~ 1, data = occ_um)
48 names(all_null)[i] <- unique(dat.scaled$scientific_name)[i]
49 setTxtProgressBar(pb, i)
50 }
51 close(pb)
52
53 # Store all the model outputs for each species
54 capture.output(all_null, file = "data\\results\\null_models.csv")

```

269 7.3 Identifying covariates necessary to model the detection process

270 Here, we use the unmarked package in R (Fiske and Chandler 2019) to identify detection level covariates that are important
 271 for each species.

```

1
2 # All models are stored in lists below
3 det_dred <- list()
4
5 # Subsetting those models whose deltaAIC<2 (Burnham et al., 2011)
6 top_det <- list()
7
8 # Getting model averaged coefficients and relative importance scores
9 det_avg <- list()
10 det_imp <- list()
11
12 # Getting model estimates
13 det_modelEst <- list()
14
15 # Add a progress bar for the loop
16 pb <- txtProgressBar(min = 0,
17   max = length(unique(dat.scaled$scientific_name)), style = 3) # text based bar
18
19 for (i in 1:length(unique(dat.scaled$scientific_name))) {
20   data <- dat.scaled %>%
21     filter(dat.scaled$scientific_name == unique(dat.scaled$scientific_name)[i])
22
23   # Preparing data for the unmarked model
24   occ <- filter_repeat_visits(data,
25     min_obs = 1, max_obs = 10,
26     annual_closure = FALSE,
27     n_days = 2600, # 6 years is considered a period of closure
28     date_var = "observation_date",

```



```

29   site_vars = c("locality_id")
30 )
31
32 obs_covs <- c(
33   "min_obs_started",
34   "duration_minutes",
35   "effort_distance_km",
36   "number_observers",
37   "protocol_type",
38   "expertise",
39   "julian_date"
40 )
41
42 # format for unmarked
43 occ_wide <- format_unmarked_occu(occ,
44   site_id = "site",
45   response = "pres_abs",
46   site_covs = c("locality_id", "lc_01.y", "lc_02.y", "lc_04.y",
47     "lc_05.y", "lc_06.y", "lc_07.y", "bio_1.y", "bio_12.y"),
48   obs_covs = obs_covs
49 )
50
51 # Convert this dataframe of observations into an unmarked object to start fitting occupancy models
52 occ_um <- formatWide(occ_wide, type = "unmarkedFrameOccu")
53
54 # Fit a global model with all detection level covariates
55 global_mod <- occu(~ min_obs_started +
56   julian_date +
57   duration_minutes +
58   effort_distance_km +
59   number_observers +
60   protocol_type +
61   expertise ~ 1, data = occ_um)
62
63 # Set up the cluster
64 clusterType <- if (length(find.package("snow", quiet = TRUE))) "SOCK" else "PSOCK"
65 clust <- try(makeCluster(getOption("cl.cores", 6), type = clusterType))
66
67 clusterEvalQ(clust, library(unmarked))
68 clusterExport(clust, "occ_um")
69
70 det_dred[[i]] <- pdredge(global_mod, clust)
71 names(det_dred)[i] <- unique(dat.scaled$scientific_name)[i]
72
73 # Get the top models, which we'll define as those with deltaAICc < 2
74 top_det[[i]] <- get.models(det_dred[[i]], subset = delta < 2, cluster = clust)
75 names(top_det)[i] <- unique(dat.scaled$scientific_name)[i]
76
77 # Obtaining model averaged coefficients
78 if (length(top_det[[i]]) > 1) {
79   a <- model.avg(top_det[[i]], fit = TRUE)
80   det_avg[[i]] <- as.data.frame(a$coefficients)
81   names(det_avg)[i] <- unique(dat.scaled$scientific_name)[i]
82

```

```

83   det_modelEst[[i]] <- data.frame(
84     Coefficient = coefTable(a, full = T)[, 1],
85     SE = coefTable(a, full = T)[, 2],
86     lowerCI = confint(a)[, 1],
87     upperCI = confint(a)[, 2],
88     z_value = (summary(a)$coefmat.full)[, 3],
89     Pr_z = (summary(a)$coefmat.full)[, 4]
90   )
91
92   names(det_modelEst)[i] <- unique(dat.scaled$scientific_name)[i]
93
94   det_imp[[i]] <- as.data.frame(MuMIn::importance(a))
95   names(det_imp)[i] <- unique(dat.scaled$scientific_name)[i]
96 } else {
97   det_avg[[i]] <- as.data.frame(unmarked::coef(top_det[[i]][[1]]))
98   names(det_avg)[i] <- unique(dat.scaled$scientific_name)[i]
99
100   lowDet <- data.frame(lowerCI = confint(top_det[[i]][[1]], type = "det")[, 1])
101   upDet <- data.frame(upperCI = confint(top_det[[i]][[1]], type = "det")[, 2])
102   zDet <- data.frame(summary(top_det[[i]][[1]])$det[, 3])
103   Pr_zDet <- data.frame(summary(top_det[[i]][[1]])$det[, 4])
104
105   Coefficient <- coefTable(top_det[[i]][[1]])[, 1]
106   SE <- coefTable(top_det[[i]][[1]])[, 2]
107
108   det_modelEst[[i]] <- data.frame(
109     Coefficient = Coefficient[2:9],
110     SE = SE[2:9],
111     lowerCI = lowDet,
112     upperCI = upDet,
113     z_value = zDet,
114     Pr_z = Pr_zDet
115   )
116
117   names(det_modelEst)[i] <- unique(dat.scaled$scientific_name)[i]
118 }
119 setTxtProgressBar(pb, i)
120 stopCluster(clust)
121 }
122 close(pb)
123
124 ## Storing output from the above models in excel sheets
125
126 # 1. Store all the model outputs for each species (variable: det_dred() - see above)
127 write.xlsx(det_dred, file = "data\\results\\det-dred.xlsx")
128
129 # 2. Store all the model averaged outputs for each species and the relative importance score
130 write.xlsx(det_avg, file = "data\\results\\det-avg.xlsx", rowNames = T, colNames = T)
131 write.xlsx(det_imp, file = "data\\results\\det-imp.xlsx", rowNames = T, colNames = T)
132
133 write.xlsx(det_modelEst, file = "data\\results\\det-modelEst.xlsx", rowNames = T, colNames = T)

```

7.4 Land Cover and Climate

Occupancy models estimate the probability of occurrence of a given species while controlling for the probability of detection and allow us to model the factors affecting occurrence and detection independently (Johnston et al., 2018; Mackenzie et al., 2018). The flexible eBird observation process contributes to the largest source of variation in the likelihood of detecting a particular species (Johnston et al., 2019); hence, we included seven covariates that influence the probability of detection for each checklist: ordinal day of year, duration of observation, distance travelled, protocol type, time observations started, number of observers and the checklist calibration index (CCI).

Using a multi-model information-theoretic approach, we tested how strongly our occurrence data fit our candidate set of environmental covariates (K.P. Burnham & Anderson, 2002). We fitted single-species occupancy models for each species, to simultaneously estimate a probability of detection (p) and a probability of occupancy (ψ) (Fiske & Chandler, 2011; Mackenzie et al., 2002). For each species, we fit 256 models, each with a unique combination of the eight (climate and land cover) occupancy covariates and all seven detection covariates (Appendix S5).

Across the 256 models tested for each species, the model with highest support was determined using AICc scores. However, across the majority of the species, no single model had overwhelming support. Hence, for each species, we examined those models which had $\Delta AICc < 2$, as these top models were considered to explain a large proportion of the association between the species-specific probability of occupancy and environmental drivers (Kenneth P. Burnham et al., 2011; Elsen et al., 2017). Using these restricted model sets for each species; we created a model-averaged coefficient estimate for each predictor and assessed its direction and significance (Bartoń, 2009). We considered a predictor to be significantly associated with occupancy if the range of the 95% confidence interval around the model-averaged coefficient did not contain zero. Next, we obtained a measure of relative importance of climatic and landscape predictors by calculating cumulative variable importance scores. These scores were calculated by obtaining the sum of model weights (AIC weights) across all models (including the top models) for each predictor across all species.

```
# All models are stored in lists below
lc_clim <- list()

# Subsetting those models whose deltaAIC<2 (Burnham et al., 2011)
top_lc_clim <- list()

# Getting model averaged coefficients and relative importance scores
lc_clim_avg <- list()
lc_clim_imp <- list()

# Storing Model estimates
lc_clim_modelEst <- list()

# Add a progress bar for the loop
pb <- txtProgressBar(min = 0, max = length(unique(dat.scaled$scientific_name)), style = 3) # text based bar

for (i in 1:length(unique(dat.scaled$scientific_name))) {
  data <- dat.scaled %>% filter(dat.scaled$scientific_name == unique(dat.scaled$scientific_name)[1])

  # Preparing data for the unmarked model
  occ <- filter_repeat_visits(data,
    min_obs = 1, max_obs = 10,
    annual_closure = FALSE,
    n_days = 2600, # 6 years is considered a period of closure
    date_var = "observation_date",
    site_vars = c("locality_id")
  )

  obs_covs <- c(
    "min_obs_started",
```

```

31     "duration_minutes",
32     "effort_distance_km",
33     "number_observers",
34     "protocol_type",
35     "expertise",
36     "julian_date"
37 )
38
39 # format for unmarked
40 occ_wide <- format_unmarked_occu(occ,
41     site_id = "site",
42     response = "pres_abs",
43     site_covs = c("locality_id", "lc_01.y", "lc_02.y", "lc_04.y", "lc_05.y",
44         "lc_06.y", "lc_07.y", "bio_1.y", "bio_12.y"),
45     obs_covs = obs_covs
46 )
47
48 # Convert this dataframe of observations into an unmarked object to start fitting occupancy models
49 occ_um <- formatWide(occ_wide, type = "unmarkedFrameOccu")
50
51 model_lc_clim <- occu(~ min_obs_started +
52     julian_date +
53     duration_minutes +
54     effort_distance_km +
55     number_observers +
56     protocol_type +
57     expertise ~ lc_01.y + lc_02.y + lc_04.y +
58     lc_05.y + lc_06.y + lc_07.y + bio_1.y + bio_12.y, data = occ_um)
59
60 # Set up the cluster
61 clusterType <- if (length(find.package("snow", quiet = TRUE))) "SOCK" else "PSOCK"
62 clust <- try(makeCluster(getOption("cl.cores", 6), type = clusterType))
63
64 clusterEvalQ(clust, library(unmarked))
65 clusterExport(clust, "occ_um")
66
67 # Detection terms are fixed
68 det_terms <- c(
69     "p(duration_minutes)", "p(effort_distance_km)", "p(expertise)",
70     "p(julian_date)", "p(min_obs_started)",
71     "p(number_observers)", "p(protocol_type)"
72 )
73
74 lc_clim[[i]] <- pdredge(model_lc_clim, clust, fixed = det_terms)
75 names(lc_clim)[i] <- unique(dat.scaled$scientific_name)[i]
76
77 # Identifying top subset of models based on deltaAIC scores being less than 2 (Burnham et al., 2011)
78 top_lc_clim[[i]] <- get.models(lc_clim[[i]], subset = delta < 2, cluster = clust)
79
80 names(top_lc_clim)[i] <- unique(dat.scaled$scientific_name)[i]
81
82 # Obtaining model averaged coefficients for both candidate model subsets
83 if (length(top_lc_clim[[i]]) > 1) {
84     a <- model.avg(top_lc_clim[[i]], fit = TRUE)

```

```

85     lc_clim_avg[[i]] <- as.data.frame(a$coefficients)
86     names(lc_clim_avg)[i] <- unique(dat.scaled$scientific_name)[i]
87
88     lc_clim_modelEst[[i]] <- data.frame(
89       Coefficient = coefTable(a, full = T)[, 1],
90       SE = coefTable(a, full = T)[, 2],
91       lowerCI = confint(a)[, 1],
92       upperCI = confint(a)[, 2],
93       z_value = (summary(a)$coefmat.full)[, 3],
94       Pr_z = (summary(a)$coefmat.full)[, 4]
95     )
96
97     names(lc_clim_modelEst)[i] <- unique(dat.scaled$scientific_name)[i]
98
99     lc_clim_imp[[i]] <- as.data.frame(MuMIn::importance(a))
100    names(lc_clim_imp)[i] <- unique(dat.scaled$scientific_name)[i]
101  } else {
102    lc_clim_avg[[i]] <- as.data.frame(unmarked::coef(top_lc_clim[[i]][[1]]))
103    names(lc_clim_avg)[i] <- unique(dat.scaled$scientific_name)[i]
104
105    lowSt <- data.frame(lowerCI = confint(top_lc_clim[[i]][[1]]), type = "state")[, 1]
106    lowDet <- data.frame(lowerCI = confint(top_lc_clim[[i]][[1]]), type = "det")[, 1]
107    upSt <- data.frame(upperCI = confint(top_lc_clim[[i]][[1]]), type = "state")[, 2]
108    upDet <- data.frame(upperCI = confint(top_lc_clim[[i]][[1]]), type = "det")[, 2]
109    zSt <- data.frame(z_value = summary(top_lc_clim[[i]][[1]])$state[, 3])
110    zDet <- data.frame(z_value = summary(top_lc_clim[[i]][[1]])$det[, 3])
111    Pr_zSt <- data.frame(Pr_z = summary(top_lc_clim[[i]][[1]])$state[, 4])
112    Pr_zDet <- data.frame(Pr_z = summary(top_lc_clim[[i]][[1]])$det[, 4])
113
114    lc_clim_modelEst[[i]] <- data.frame(
115      Coefficient = coefTable(top_lc_clim[[i]][[1]]), [, 1],
116      SE = coefTable(top_lc_clim[[i]][[1]]), [, 2],
117      lowerCI = rbind(lowSt, lowDet),
118      upperCI = rbind(upSt, upDet),
119      z_value = rbind(zSt, zDet),
120      Pr_z = rbind(Pr_zSt, Pr_zDet)
121    )
122
123    names(lc_clim_modelEst)[i] <- unique(dat.scaled$scientific_name)[i]
124  }
125  setTxtProgressBar(pb, i)
126  stopCluster(clust)
127 }
128 close(pb)
129
130 # 1. Store all the model outputs for each species (for both landcover and climate)
131 write.xlsx(lc_clim, file = "data\\results\\lc-clim.xlsx")
132
133 # 2. Store all the model averaged outputs for each species and relative importance scores
134 write.xlsx(lc_clim_avg, file = "data\\results\\lc-clim-avg.xlsx", rowNames = T, colNames = T)
135 write.xlsx(lc_clim_imp, file = "data\\results\\lc-clim-imp.xlsx", rowNames = T, colNames = T)
136
137 # 3. Store all model estimates
138 write.xlsx(lc_clim_modelEst, file = "data\\results\\lc-clim-modelEst.xlsx", rowNames = T, colNames = T)

```

7.5 Goodness-of-fit tests

Adequate model fit was assessed using a chi-square goodness-of-fit test using 5000 parametric bootstrap simulations on a global model that included all occupancy and detection covariates (MacKenzie & Bailey, 2004).

```
goodness_of_fit <- data.frame()

# Add a progress bar for the loop
pb <- txtProgressBar(min = 0, max = length(unique(dat.scaled$scientific_name)), style = 3) # text based bar

for (i in 1:length(unique(dat.scaled$scientific_name))) {
  data <- dat.scaled %>% filter(dat.scaled$scientific_name == unique(dat.scaled$scientific_name)[i])

  # Preparing data for the unmarked model
  occ <- filter_repeat_visits(data,
    min_obs = 1, max_obs = 10,
    annual_closure = FALSE,
    n_days = 2600, # 6 years is considered a period of closure
    date_var = "observation_date",
    site_vars = c("locality_id")
  )

  obs_covs <- c(
    "min_obs_started",
    "duration_minutes",
    "effort_distance_km",
    "number_observers",
    "protocol_type",
    "expertise",
    "julian_date"
  )

  # format for unmarked
  occ_wide <- format_unmarked_occu(occ,
    site_id = "site",
    response = "pres_abs",
    site_covs = c("locality_id", "lc_01.y", "lc_02.y", "lc_04.y", "lc_05.y", "lc_06.y", "lc_07.y", "bio_1.y", "bio_12.y"),
    obs_covs = obs_covs
  )

  # Convert this dataframe of observations into an unmarked object to start fitting occupancy models
  occ_um <- formatWide(occ_wide, type = "unmarkedFrameOccu")

  model_lc_clim <- occu(~ min_obs_started +
    julian_date +
    duration_minutes +
    effort_distance_km +
    number_observers +
    protocol_type +
    expertise ~ lc_01.y + lc_02.y + lc_04.y +
    lc_05.y + lc_06.y + lc_07.y + bio_1.y + bio_12.y, data = occ_um)

  occ_gof <- mb.gof.test(model_lc_clim, nsim = 5000, plot.hist = FALSE)

  p.value <- occ_gof$p.value
```

```

51 c.hat <- occ_gof$c.hat.est
52 scientific_name <- unique(data$scientific_name)
53
54 a <- data.frame(scientific_name, p.value, c.hat)
55
56 goodness_of_fit <- rbind(a, goodness_of_fit)
57
58 setTxtProgressBar(pb, i)
59 }
60 close(pb)
61
62 write.csv(goodness_of_fit, "data\\results\\05_goodness-of-fit-2.5km.csv")

```

297 8 Visualizing Occupancy Predictor Effects

298 In this section, we will visualize the cumulative AIC weights and the magnitude and direction of species-specific probability
 299 of occupancy.

300 To get cumulative AIC weights, we first obtained a measure of relative importance of climatic and landscape predictors by
 301 calculating cumulative variable importance scores. These scores were calculated by obtaining the sum of model weights
 302 (AIC weights) across all models (including the top models) for each predictor across all species. We then calculated the
 303 mean cumulative variable importance score and a standard deviation for each predictor (K.P. Burnham & Anderson, 2002).

304 8.1 Prepare libraries

```

1 # to load data
2 library(readxl)
3
4 # to handle data
5 library(dplyr)
6 library(readr)
7 library(forcats)
8 library(tidyr)
9 library(purrr)
10 library(stringr)
11 library(data.table)
12
13 # to wrangle models
14 source("code/fun_model_estimate_collection.r")
15 source("code/fun_make_resp_data.r")
16
17 # nice tables
18 library(knitr)
19 library(kableExtra)
20
21 # plotting
22 library(ggplot2)
23 library(patchwork)
24 source("code/fun_plot_interaction.r")

```

305 8.2 Load species list

```

1 # list of species
2 species <- read_csv("data/species_list.csv")

```

```
3 list_of_species <- as.character(species$scientific_name)
```

306 8.3 Show AIC weight importance

307 8.3.1 Read in AIC weight data

```
1 # which files to read
2 file_names <- c("data/results/lc-clim-imp.xlsx")
3
4 # read in sheets by species
5 model_imp <- map(file_names, function(f) {
6   md_list <- map(list_of_species, function(sn) {
7
8     # some sheets are not found
9
10    tryCatch({
11
12      readxl::read_excel(f, sheet = sn) %>%
13        `colnames<-`(c("predictor", "AIC_weight")) %>%
14        filter(str_detect(predictor, "psi")) %>%
15        mutate(predictor = stringr::str_extract(predictor,
16          pattern = stringr::regex("\\((.*?)\\)"),
17          predictor = stringr::str_replace_all(predictor, "[/(/)]", ""),
18          predictor = stringr::str_remove(predictor, "\\y"))
19
20    },
21    error = function(e) {
22      message(as.character(e))
23    }
24  )
25 })
26 names(md_list) <- list_of_species
27
28 return(md_list)
29 })
```

308 8.3.2 Prepare cumulative AIC weight data

```
1 # assign scale - minimum spatial scale at which the analysis was carried out to account for observer effort
2 names(model_imp) <- c("2.5km")
3 model_imp <- imap(model_imp, function(.x, .y) {
4   .x <- bind_rows(.x)
5   .x$scale <- .y
6   return(.x)
7 })
8
9 # bind rows
10 model_imp <- map(model_imp, bind_rows) %>%
11   bind_rows()
12
13 # convert to numeric
14 model_imp$AIC_weight <- as.numeric(model_imp$AIC_weight)
15 model_imp$scale <- as.factor(model_imp$scale)
16 levels(model_imp$scale) <- c("2.5km")
17
```



```

18 # Let's get a summary of cumulative variable importance
19 model_imp <-group_by(model_imp, predictor) %>%
20   summarise(mean_AIC = mean(AIC_weight),
21             sd_AIC=sd(AIC_weight),
22             min_AIC = min(AIC_weight),
23             max_AIC = max(AIC_weight),
24             med_AIC = median(AIC_weight))
25
26 # write to file
27 write_csv(model_imp,
28           file = "data/results/cumulative_AIC_weights.csv")
29
309 Read data back in.
31
32 # read data and make factor
33 model_imp <- read_csv("data/results/cumulative_AIC_weights.csv")
34 model_imp$predictor <- as_factor(model_imp$predictor)
35
36 # make nice names
37 predictor_name <- tibble(predictor = levels(model_imp$predictor),
38                           pred_name = c("Annual Mean Temperature (°C)",
39                                           "Annual Precipitation (mm)",
40                                           "% Agriculture", "% Forests",
41                                           "% Plantations", "% Settlements",
42                                           "% Tea", "% Water Bodies"))
43
44 # rename predictor
45 model_imp <- left_join(model_imp, predictor_name)
46
310 Prepare figure for cumulative AIC weight. Figure code is hidden in versions rendered as HTML and PDF.
47
48 fig_aic <-
49   ggplot(model_imp)+
50   geom_pointrange(aes(x = reorder(predictor, mean_AIC),
51                       y = mean_AIC,
52                       ymin = mean_AIC - sd_AIC,
53                       ymax = mean_AIC + sd_AIC))+
54   geom_text(aes(x = predictor,
55                 y = 0.2,
56                 label = pred_name),
57             angle = 0,
58             hjust = "inward",
59             vjust = 2)+
60   # scale_y_continuous(breaks = seq(45, 75, 10))+
61   scale_x_discrete(labels = NULL)+
62   # scale_color_brewer(palette = "RdBu", values = c(0.5, 1))+
63   coord_flip(
64     # ylim = c(45, 75)
65   )+
66   theme_test()+
67   theme(legend.position = "none")+
68   labs(x = "Predictor",
69        y = "Cumulative AIC weight")
70
71 ggsave(fig_aic,
72        filename = "figs/fig_aic_weight.png",

```

```

26     device = png(),
27     dpi = 300,
28     width = 79, height = 120, units = "mm")

```

311 8.4 Prepare model coefficient data

312 For each species, we examined those models which had $\Delta AICc < 2$, as these top models were considered to explain a large
313 proportion of the association between the species-specific probability of occupancy and environmental drivers (Kenneth
314 P. Burnham et al., 2011; Elsen et al., 2017). Using these restricted model sets for each species; we created a model-
315 averaged coefficient estimate for each predictor and assessed its direction and significance (Bartoń, 2009). We considered
316 a predictor to be significantly associated with occupancy if the range of the 95% confidence interval around the model-
317 averaged coefficient did not contain zero.

```

1  file_read <- c("data/results/lc-clim-modelEst.xlsx")
2
3  # read data as list column
4  model_est <- map(file_read, function(fr) {
5    md_list <- map(list_of_species, function(sn) {
6      readxl::read_excel(fr, sheet = sn)
7    })
8    names(md_list) <- list_of_species
9    return(md_list)
10 })
11
12 # prepare model data
13 scales = c("2.5km")
14 model_data <- tibble(scale = scales,
15                      scientific_name = list_of_species) %>%
16   arrange(desc(scale))
17
18 # rename model data components and separate predictors
19 names <- c("predictor", "coefficient", "se", "ci_lower",
20           "ci_higher", "z_value", "p_value")
21
22 # get data for plotting:
23 model_est <- map(model_est, function(l) {
24   map(l, function(df) {
25     colnames(df) <- names
26     df <- separate_interaction_terms(df)
27     df <- make_response_data(df)
28     return(df)
29   })
30 })
31
32 # add names and scales
33 model_est <- map(model_est, function(l) {
34   imap(l, function(.x, .y) {
35     mutate(.x, scientific_name = .y)
36   })
37 })
38
39 # add names to model estimates
40 names(model_est) <- scales
41 model_est <- imap(model_est, function(.x, .y) {
42   bind_rows(.x) %>%

```

```

43     mutate(scale = .y)
44   })
45
46   # remove modulators
47   model_est <- bind_rows(model_est) %>%
48     select(-matches("modulator"))
49
50   # join data to species name
51   model_data <- model_data %>%
52     left_join(model_est)
53
54   # Keep only those predictors whose p-values are significant:
55   model_data <- model_data %>%
56     filter(p_value < 0.05)
57
318   Export predictor effects.
59
60   # get predictor effect data
61   data_predictor_effect <- distinct(model_data,
62                                     scientific_name,
63                                     se,
64                                     predictor, coefficient)
65
66   # write to file
67   write_csv(data_predictor_effect,
68             path = "data/results/data_predictor_effect.csv")
69
319   Export model data.
71
72   model_data_to_file <- model_data %>%
73     select(predictor, data,
74            scientific_name, scale) %>%
75     unnest(cols = "data")
76
77   # remove .y
78   model_data_to_file <- model_data_to_file %>%
79     mutate(predictor = str_remove(predictor, "\\..y"))
80
81   write_csv(model_data_to_file,
82             "data/results/data_occupancy_predictors.csv")
83
320   Read in data after clearing R session.
85
86   # read from file
87   model_data <- read_csv("data/results/data_predictor_effect.csv")
88
321   Fix predictor name.
89
90   # remove .y from predictors
91   model_data <- model_data %>%
92     mutate_at(.vars = c("predictor"), .funs = function(x){
93       stringr::str_remove(x, ".y")
94     })
95

```

322 8.5 Get predictor effects

```

96   # is the coeff positive? how many positive per scale per predictor per axis of split?
97   data_predictor <- mutate(model_data,

```

```

3           direction = coefficient > 0) %>%
4   count(predictor,
5         direction) %>%
6   mutate(mag = n * (if_else(direction, 1, -1)))
7
8   # wrangle data to get nice bars
9   data_predictor <- data_predictor%>%
10    select(-n) %>%
11    drop_na(direction) %>%
12    mutate(direction = ifelse(direction, "positive", "negative")) %>%
13    pivot_wider(values_from = "mag", names_from = "direction") %>%
14    mutate_at(vars(positive, negative),
15              ~if_else(is.na(.), 0, .))
16
17   data_predictor_long <- data_predictor %>%
18     pivot_longer(cols = c("negative", "positive"),
19                 names_to = "effect",
20                 values_to = "magnitude")
21
22   # write
23   write_csv(data_predictor_long,
24             path = "data/results/data_predictor_direction_nSpecies.csv")

```

323 Prepare data to determine the direction (positive or negative) of the effect of each predictor. How many species are affected
324 in either direction?

```

1   # join with predictor names and relative AIC
2   data_predictor_long <- left_join(data_predictor_long, model_imp)

```

325 Prepare figure of the number of species affected in each direction. Figure code is hidden in versions rendered as HTML
326 and PDF.

327 8.6 Main Text Figure 4

328 Figure code is hidden in versions rendered as HTML and PDF.

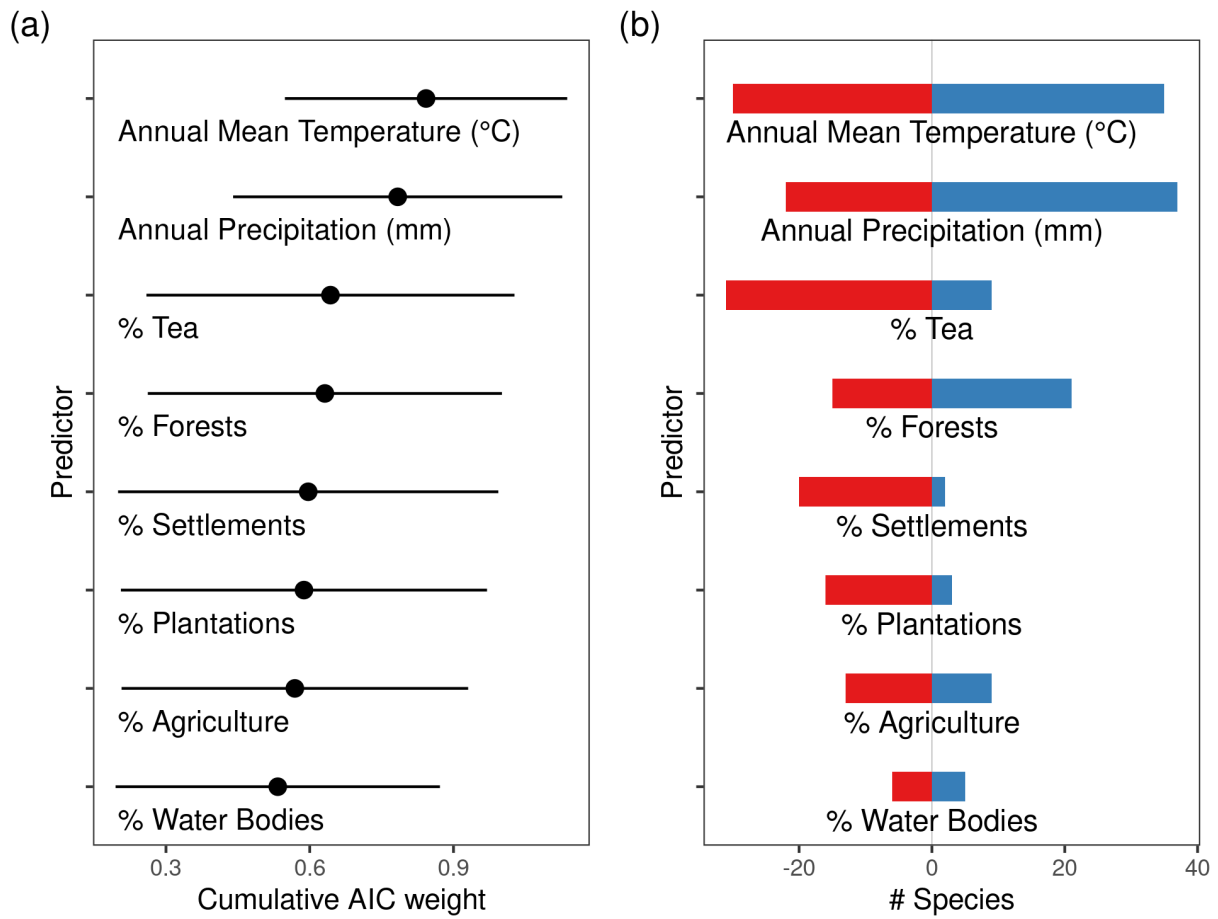


Figure 4: (a) Cumulative AIC weights suggest that climatic predictors have higher relative importance when compared to landscape predictors. (b) The direction of association between species-specific probability of occupancy and climatic and landscape is shown here. While climatic predictors were both positively and negatively associated with the probability of occupancy for a number of species, human-associated land cover types were largely negatively associated with species-specific probability of occupancy.