Human Influence in SDMs: Literature Review (Part IV)

Veronica F. Frans (email: verofrans@gmail.com)

February 1, 2024

Contents

1 Summary

This is the fourth R script of the literature review and synthesis for the article entitled, "Gaps and opportunities in modeling human influence on species distributions in the Anthropocene," by Veronica F. Frans and Jianguo Liu.

Here, in Part IV of the synthesis, we use the CSV files of the systematic review and data cleanup from Part I-III to assess the study area scales and get a global context for human predictor use in SDMs. We also summarize SDM algorithms across studies and do a final cleanup of the dataset for export as Supporting Information.

Thus, the following is accomplished:

(1) Summary of study area scales across articles
(2) Maps of study countries for all articles, taxa, domain, and study focus
(3) Maps with frequency of studies across spatial scales
(4) Maps of first published years of human predictor use across spatial scales
(5) Maps showing frequency of predictors across spatial scales
(6) Cleanup and summary of SDM algorithms used across articles
(7) Export of CSV file for the systematic review (supplementary material for publication)

The next script (Part V) uses the predictor list and the resulting CSV file of the systematic review to summarize predictor use in relation to United Nations Sustainable Development Goals (SDGs).

2 R Setup

We are using R version 4.3.0 (R Core Team 2023).

2.1 Libraries

Load libraries

```
# load libraries
  library("dplyr")          # for table manipulations
  library("scales")         # for scales and formatting
  library("kableExtra")     # for table viewing in Rmarkdown
  library("tidyr")          # for table manipulations
  library("plyr")           # for table manipulations
  library("tidyverse")      # for graphics/table management
  library("ggplot2")        # for graphics
  library("RColorBrewer")   # for graphics
  library("ggforce")        # for graphics (speeds up ggplot)
  library("ggalluvial")     # for graphics
  library("ggbreak")        # for graphics
  library("patchwork")      # for graphics
  library("classInt")       # for graphics
  library("biscale")        # for graphics
  library("raster")         # for mapping
  library("rgdal")          # for mapping
  library("sp")             # for mapping
  library("ggmap")          # for mapping and graphics
  library("maps")           # for mapping
```

```
  library("tmap")           # for mapping
  library("plotfunctions") # for data visualization
  library("svglite")        # for saving graphics in svg format
  library("countrycode")    # for country name edits
```

2.2 Directories

The primary directory is the folder where the `hum_sdm_litrv_r.Rproj` is stored.

```
# create image folder and its directory
  dir.create(paste0("images"))
  image.dir <- paste0("images\\")

# create data folder and its directory
  dir.create(paste0("data"))
  data.dir <- paste0("data\\")
```

2.3 Load data

Upload the data table from the abstract screening and review, and subset to only the articles that are accepted. We will also need a few saved CSV files created in Part II.

```
# full article screening and review table
  rev.df <- read.csv(paste0(data.dir,"hum_sdm_lit_review_RAW.csv"),
                             header=T, sep=",")

# subset of only accepted articles (after year 2000)
  yes.df <- rev.df[(rev.df$relevant=="yes"),]
  yes.df <- yes.df[(yes.df$year>=2000),]

# study domain, taxa, and focus list of counts
  domtaxfoc.df <- read.csv(paste0(data.dir,"domain_taxa_focus_count_papers.csv"),
                            header=T, sep=",")

# predictor list of counts
  preds.list.export <- read.csv(paste0(data.dir,"predictor_list_summary_FINAL.csv"),
                                 header=T, sep=",")

# study domain, taxa, focus, and predictor list (very long)
  prdotf.list.long <- read.csv(paste0(data.dir,"predictor_domain_taxa_focus_long.csv"),
                                header=T, sep=",")

# ambiguous predictor list
  amb.df <- read.csv(paste0(data.dir,"ambiguous_predictor_dataframe.csv"),
                                header=T, sep=",")
```

3 Study area scale

Make a pie chart of the study area scales across articles.

3.1 Table setup

```
# edit country and multi-country scale names
  yes.df$study_area_scale[yes.df$study_area_scale == "country" ] <- "national"
  yes.df$study_area_scale[yes.df$study_area_scale == "multi-country" ] <- "multinational"

# subset table
  study.area.df <- subset(yes.df, select=c("uid","study_area_scale","study_area_country"))

# edit a typo
  study.area.df$study_area_scale[
                          study.area.df$study_area_scale=="contintental"] <- "continental"

# set scale as factor
  study.area.df$study_area_scale <- as.factor(as.character(study.area.df$study_area_scale))

# get summary
  summary(study.area.df)
```

```
##       uid             study_area_scale study_area_country
##  Min.   :    2   continental  : 49     Length:1441
##  1st Qu.: 2885   global       : 42     Class :character
##  Median : 5968   local        :556     Mode  :character
##  Mean   : 6050   multinational:193
##  3rd Qu.: 9082   national     :238
##  Max.   :12484   regional     :363
```

3.2 Table: study area scales

```
# Get a count of scale for each paper
  study.scales.list <- ddply(study.area.df, .(study_area_scale),
                             summarize,
                             count_studies=length(study_area_scale),
                             count_papers=length(unique(uid)),
                             percent_papers=count_papers/length(unique(study.area.df$uid))*100
                             )

# Sort by most frequent
  study.scales.list <- study.scales.list[order(-study.scales.list$count_papers),]

# adjust levels
  study.scales.list$study_area_scale <- factor(study.scales.list$study_area_scale,
                                       levels=c("local","regional","national",
                                                "multinational", "continental","global"))

# add column for positioning labels on pie chart
  study.scales.list <- study.scales.list %>%
                       group_by(study_area_scale) %>%
                       mutate(pos = (cumsum(count_papers)-count_papers)/2)

# View table
```

```
  kableExtra::kbl(study.scales.list,booktabs=T, longtable=T) %>%
    kable_styling(latex_options = c("striped","repeat_header"))
```

study_area_scale	count_studies	count_papers	percent_papers	pos
local	556	553	38.698391	0.0
regional	363	362	25.332400	276.5
national	238	238	16.655003	457.5
multinational	193	193	13.505948	576.5
continental	49	46	3.219034	673.0
global	42	37	2.589223	696.0

3.3 Pie chart

Blank theme from http://www.sthda.com/english/wiki/ggplot2-pie-chart-quick-start-guide-r-software-and-data-visualization:

```
# create blank theme
  blank_theme <- theme_minimal()+
                 theme(
                 axis.title.x = element_blank(),
                 axis.title.y = element_blank(),
                 panel.border = element_blank(),
                 panel.grid=element_blank(),
                 axis.ticks = element_blank(),
                 plot.title=element_text(size=14, face="bold")
                 )
```

Make pie chart.

```
  require('scales')

# colorblind-friendly colors
  sc.cols <- c('#882255','#0077BB','#44AA99','#117733','#999933','#DDCC77')

# label positions
  sc.pos <- c(300,200,120,50,25,10)
  sc.pos.x <- c(1.3,1,.9,1.5,1.35,1.5)
  sc.pos.y <- c(10,25,-15,8,5,8)

# barchart
  sc.bar <- ggplot(data=study.scales.list) +
            aes(x="", y=count_papers, fill=study_area_scale)+
            geom_bar(width = 1, stat = "identity") +
            scale_fill_manual(values=sc.cols) +
            geom_text(aes(x = 1.1, y = sc.pos,label = count_papers), size=5)+
            geom_text(aes(x = sc.pos.x,
                          y = sc.pos.y+sc.pos,
                          label = study_area_scale), size=6)

# convert to pie
```

```
  sc.pie <- sc.bar + coord_polar(theta="y", start=0)
  sc.pie <- sc.pie + blank_theme # activate for blank theme function

# pie labeled
  sc.lab <- sc.pie +
            theme(plot.margin = unit(c(-2,0,0,0), "lines"))

# save
  ggsave(plot=sc.lab, filename = paste0(image.dir,'study_area_scale_pie.png'),
         height = 5, width = 5, units = 'in', dpi = 600)
  ggsave(plot=sc.lab, filename = paste0(image.dir,'study_area_scale_pie.svg'),
         height = 5, width = 5, units = 'in')
```

This figure will be edited later into the overall map of study area countries compared to human footprint, which was done using ArcPro.

4 Edit study area country names

Next, we refine the list of countries for the study areas. The study areas were recorded based on the materials and methods sections of the articles, or extracted from visually inspecting any maps that were provided by the authors as reference. Here, the list of countries are converted using the `countrycode` package. Note that some country names did not fully carry over, but to the best of our abilities, we edited and made further matches for mapping.

4.1 Country name edits

Make new rows for each country in the study area dataset.

```
# Separate countries into multiple rows
  study.area.df2 <- separate_rows(study.area.df,study_area_country,sep="; ",
                                  convert = TRUE)

# Change to factor
  study.area.df2$study_area_country <- as.factor(study.area.df2$study_area_country)
```

Edit country names using the `countrycode` package.

```
# Function to correct misspelled country names
  edit_country_names <- function(country){
                                        countrycode(country, "country.name", "country.name")
                                          }

# Apply the correction function to the 'country' column (this takes time!)
  study.area.df2$country_edit <- sapply(study.area.df2$study_area_country,
                                        edit_country_names)

# Print the corrected data
  levels(as.factor(study.area.df2$country_edit))
```

```
##   [1] "Afghanistan"              "Albania"
```

```
##   [3] "Algeria"                  "Andorra"
##   [5] "Angola"                   "Anguilla"
##   [7] "Antigua & Barbuda"        "Argentina"
##   [9] "Armenia"                  "Australia"
##  [11] "Austria"                  "Azerbaijan"
##  [13] "Bahamas"                  "Bahrain"
##  [15] "Bangladesh"               "Barbados"
##  [17] "Belarus"                  "Belgium"
##  [19] "Belize"                   "Benin"
##  [21] "Bhutan"                   "Bolivia"
##  [23] "Bosnia & Herzegovina"     "Botswana"
##  [25] "Brazil"                   "British Virgin Islands"
##  [27] "Brunei"                   "Bulgaria"
##  [29] "Burkina Faso"             "Burundi"
##  [31] "Cambodia"                 "Cameroon"
##  [33] "Canada"                   "Caribbean Netherlands"
##  [35] "Central African Republic" "Chad"
##  [37] "Chile"                    "China"
##  [39] "Colombia"                 "Comoros"
##  [41] "Congo - Brazzaville"      "Congo - Kinshasa"
##  [43] "Costa Rica"               "Côte d'Ivoire"
##  [45] "Croatia"                  "Cuba"
##  [47] "Cyprus"                   "Czechia"
##  [49] "Denmark"                  "Djibouti"
##  [51] "Dominica"                 "Dominican Republic"
##  [53] "Ecuador"                  "Egypt"
##  [55] "El Salvador"              "Equatorial Guinea"
##  [57] "Eritrea"                  "Estonia"
##  [59] "Eswatini"                 "Ethiopia"
##  [61] "Falkland Islands"         "Finland"
##  [63] "France"                   "French Guiana"
##  [65] "Gabon"                    "Gambia"
##  [67] "Georgia"                  "Germany"
##  [69] "Ghana"                    "Gibraltar"
##  [71] "Greece"                   "Grenada"
##  [73] "Guadeloupe"               "Guam"
##  [75] "Guatemala"                "Guinea"
##  [77] "Guinea-Bissau"            "Guyana"
##  [79] "Haiti"                    "Honduras"
##  [81] "Hong Kong SAR China"      "Hungary"
##  [83] "Iceland"                  "India"
##  [85] "Indonesia"                "Iran"
##  [87] "Iraq"                     "Ireland"
##  [89] "Israel"                   "Italy"
##  [91] "Jamaica"                  "Japan"
##  [93] "Jordan"                   "Kazakhstan"
##  [95] "Kenya"                    "Kosovo"
##  [97] "Kuwait"                   "Kyrgyzstan"
##  [99] "Laos"                     "Latvia"
## [101] "Lebanon"                  "Lesotho"
## [103] "Liberia"                  "Libya"
## [105] "Liechtenstein"            "Lithuania"
## [107] "Luxembourg"               "Macao SAR China"
## [109] "Madagascar"               "Malawi"
```

```
## [111] "Malaysia"                 "Maldives"
## [113] "Mali"                     "Malta"
## [115] "Martinique"               "Mauritania"
## [117] "Mexico"                   "Moldova"
## [119] "Monaco"                   "Mongolia"
## [121] "Montenegro"               "Montserrat"
## [123] "Morocco"                  "Mozambique"
## [125] "Myanmar (Burma)"          "Namibia"
## [127] "Nepal"                    "Netherlands"
## [129] "New Zealand"              "Nicaragua"
## [131] "Niger"                    "Nigeria"
## [133] "North Korea"              "North Macedonia"
## [135] "Norway"                   "Oman"
## [137] "Pakistan"                 "Palestinian Territories"
## [139] "Panama"                   "Papua New Guinea"
## [141] "Paraguay"                 "Peru"
## [143] "Philippines"              "Poland"
## [145] "Portugal"                 "Puerto Rico"
## [147] "Qatar"                    "Romania"
## [149] "Russia"                   "Rwanda"
## [151] "São Tomé & Príncipe"      "Saudi Arabia"
## [153] "Senegal"                  "Serbia"
## [155] "Sierra Leone"             "Singapore"
## [157] "Sint Maarten"             "Slovakia"
## [159] "Slovenia"                 "Somalia"
## [161] "South Africa"             "South Korea"
## [163] "South Sudan"              "Spain"
## [165] "Sri Lanka"                "St. Barthélemy"
## [167] "St. Kitts & Nevis"        "St. Lucia"
## [169] "St. Vincent & Grenadines" "Sudan"
## [171] "Suriname"                 "Sweden"
## [173] "Switzerland"              "Syria"
## [175] "Taiwan"                   "Tajikistan"
## [177] "Tanzania"                 "Thailand"
## [179] "Timor-Leste"              "Togo"
## [181] "Trinidad & Tobago"        "Tunisia"
## [183] "Turkey"                   "Turkmenistan"
## [185] "U.S. Virgin Islands"      "Uganda"
## [187] "Ukraine"                  "United Arab Emirates"
## [189] "United States"            "Uruguay"
## [191] "Uzbekistan"               "Venezuela"
## [193] "Vietnam"                  "Yemen"
## [195] "Zambia"                   "Zimbabwe"
```

Edit remaining country names.

```
# subset data to edit
  edit.list <- study.area.df2[is.na(study.area.df2$country_edit),]

# reset factor levels
  edit.list$study_area_country <- droplevels(edit.list$study_area_country)

# display here (note that we are ignoring continental and global studies in the edits)
  #summary(as.factor(edit.list$study_area_country))
```

```r
# Create a vector of patterns to search and replace (search on left, replace on right)
  patterns <- c("Azores" = "Portugal",
                "Borneo" = "Indonesia",
                "Burunei" = "Brunei",
                "Columbia" = "Colombia",
                "England" = "United Kingdom",
                "Gemany" = "Germany",
                "Gerogia" = "Georgia",
                "Java" = "Indonesia",
                "Krgyzstan" = "Kyrgyzstan",
                "Lybia" = "Libya",
                "Mauritiana" = "Mauritania",
                "Northern Ireland" = "United Kingdom",
                "Paracel Islands" = "China",
                "Phillipines" = "Philippines",
                "Saint Martin" = "St Maarten",
                "Saipan" = "United States",
                "Scotland" = "United Kingdom",
                "Spartly Island" = "Philippines",
                "Sumatra" = "Indonesia",
                "Sundan" = "Sudan",
                "Uraguay" = "Uruguay",
                "Wales" = "United Kingdom"
  )

# for-loop of edits
  for (pattern in names(patterns)) {
    edit.list <- data.frame(lapply(edit.list, function(x) {
      gsub(pattern, patterns[pattern], x)
    }))
  }

# Apply the correction function to the 'country' column (this takes time!)
  edit.list$country_edit <- sapply(edit.list$study_area_country,
                                   edit_country_names)

# Print the corrected data
  levels(as.factor(edit.list$country_edit))
```

```
##  [1] "Brunei"         "China"          "Colombia"       "Georgia"
##  [5] "Germany"        "Indonesia"      "Kyrgyzstan"     "Libya"
##  [9] "Mauritania"     "Philippines"    "Portugal"       "Sint Maarten"
## [13] "Sudan"          "United Kingdom" "United States"  "Uruguay"
```

Merge corrected data with the original table.

```r
# manual edit
  study.area.df2$country_edit[study.area.df2$study_area_country=="Oceania"] <- "Oceania"

# format columns and rows
  study.area.df2$uid <- as.character(study.area.df2$uid)
  study.area.all <- study.area.df2[!is.na(study.area.df2$country_edit),]
```

```
# combine rows
  study.area.all <- rbind(study.area.all,edit.list)

# remove any duplicated country names per uid
  study.area.all <- study.area.all[!duplicated(study.area.all[c('uid','country_edit')]),]
```

Next, we generate the ISO for country names and append to `country.list` for use in mapping.

```
# extract ISO2 code for each country
  study.area.all$iso2 <- countrycode(study.area.all$country_edit, "country.name", "iso2c")

# extract ISO3 code for each country
  study.area.all$iso3 <- countrycode(study.area.all$country_edit, "country.name", "iso3c")
```

Delete an entry for a region that is not assigned as a country under this package.

```
# delete an entry
  study.area.all <- study.area.all[!(study.area.all$study_area_country=="Kosovo"),]
```

Get final summary of countries studied.

```
# get count of countries studied
  paste(length(unique(study.area.all$country_edit,na.rm=T)),
        "countries have used human predictors in SDMs")
```

```
## [1] "198 countries have used human predictors in SDMs"
```

4.2 Merge and export country names to literature review CSV

Compress country names and ISO codes to the existing edited systematic review dataframe of accepted articles. Using this full table will help to map summaries of study areas across taxa, domain, and study focus levels. We will also merge the publication years to this dataset.

```
# # make new column called region and copy over continental and global scale names (original column)
  study.area.all$region <- study.area.all$country_edit
  study.area.all$region[is.na(study.area.all$region)] <- as.character(
    study.area.all$study_area_country[is.na(study.area.all$region)])

# drop columns
  study.area.all <- subset(study.area.all, select = -c(study_area_country,country_edit))

# set uid to integer
  study.area.all$uid <- as.integer(as.character(study.area.all$uid))

# left join by paper ID (uid)
  full.list.long <- left_join(prdotf.list.long, study.area.all, by='uid',
                              relationship = "many-to-many")
  full.list.long <- distinct(full.list.long, .keep_all = TRUE) # ensure no repeated rows

# add years of publication via left-join
```

```
  year.df <- subset(yes.df, select = c('uid','year'))

# left join by paper ID (uid)
  full.list.long <- left_join(full.list.long, year.df, by='uid')
  full.list.long <- distinct(full.list.long, .keep_all = TRUE) # ensure no repeated rows

# drop columns with names ending in ".x" and ".y"
  columns_to_drop <- grepl("\\.x$|\\.y$", names(full.list.long))

# Remove the identified columns
  full.list.long <- full.list.long[, !columns_to_drop]

 # save as a CSV
  write.csv(full.list.long,
            paste0(data.dir,"scale_taxa_domain_focus_predictor_countries.csv"),
            row.names = FALSE)
```

4.3 Subset by continent and global scales

Make subsets of the different study area scales for the summary maps.

```
# reread to remove rownames
  full.list.long <- read.csv(paste0(data.dir,"scale_taxa_domain_focus_predictor_countries.csv"),
                             header=T, sep=",")

# Make continental, country, and global lists
  study.continent <- full.list.long[grep("continental",full.list.long$study_area_scale),]
  study.country <- full.list.long[-grep("continental",full.list.long$study_area_scale),]
  study.country <- study.country[-grep("global",study.country$study_area_scale),]
  study.global <- full.list.long[grep("global",full.list.long$study_area_scale),]
```

5 Study country map

Here, we map the numbers of articles per country by first making summary tables of counts.

```
# Get a count of countries
  country.list <- ddply(study.country, .(region,iso3,iso2),
                        summarize,
                        # number of articles
                        n_papers=length(unique(uid)),
                        # percent of articles
                        perc_papers=length(unique(uid))/length(unique(full.list.long$uid))
                        )
```

Next, we match the country names using a worldmap within the tmap package.

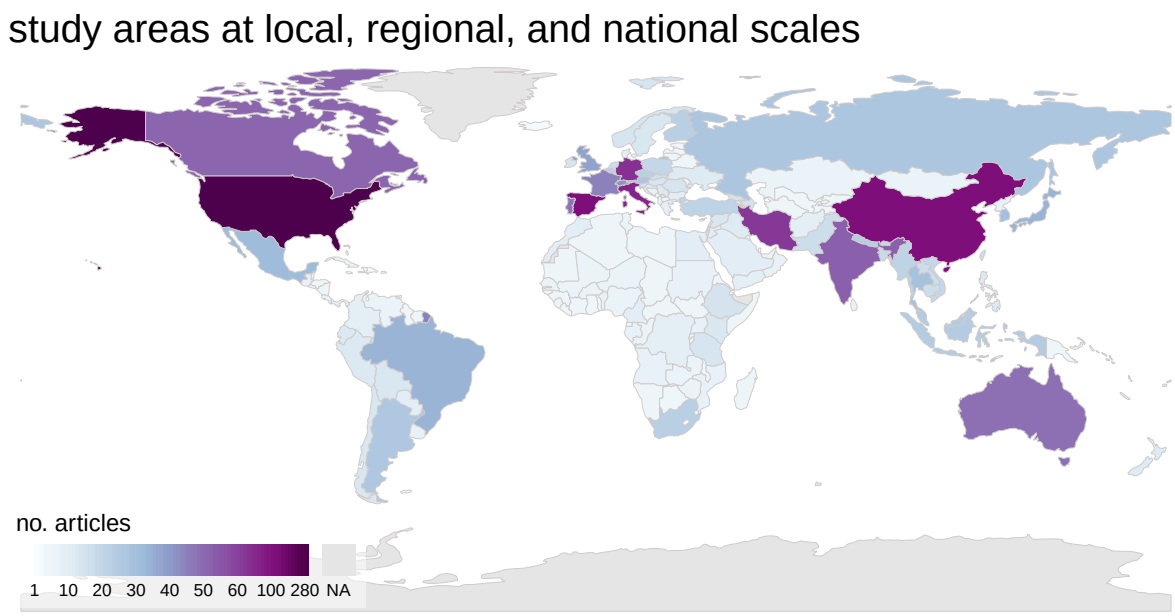
```
# match country names to world map using names
  data(World)
  country.list$name <- country.list$region
  country_papers <- left_join(World, country.list, by = "name")
```

```
# rejoin for mismatch via ISO3
  missing_countries <- anti_join(country.list, World, by = "name")
  missing_countries$iso_a3 <- missing_countries$iso3
  all_papers <- left_join(country_papers,missing_countries, by = "iso_a3")%>%
                          mutate(name = name.x,
                                 region = coalesce(region.x, region.y),
                                 iso2 = coalesce(iso2.x, iso2.y),
                                 iso3 = coalesce(iso3.x, iso3.y),
                                 region = coalesce(region.x, region.y),
                                 n_papers = coalesce(n_papers.x,n_papers.y),
                                 perc_papers = coalesce(perc_papers.x, perc_papers.y)
                                 )

# drop columns with names ending in ".x" and ".y"
  columns_to_drop <- grepl("\\.x$|\\.y$", names(all_papers))

# remove the identified columns
  all_papers <- all_papers[, !columns_to_drop]

# get number of countries for which names and ISO3 matched
  paste(nrow(all_papers[!is.na(all_papers$region),]),
        "out of",length(unique(country.list$region)),
        "counry names were matchable to the world map")
```

```
## [1] "166 out of 196 counry names were matchable to the world map"
```

```
# set mode to plotting
  tmap_mode("plot")
  #tmap_mode("view")  # set to viewing for interactive map

# plot using `tmap`
  cty_ints <- classIntervals(all_papers$n_papers, 9, style = "jenks")
  country.map <- tm_shape(all_papers) +
                 tm_borders("#d0cbcb", alpha = 1, lwd = 0.6) +
                 tm_polygons("n_papers", style = "cont",
                             breaks = c(1,10,20,30,40,50,60,100,280), # manual breaks
                             #breaks = pretty(c(1, 280), n = 9),
                             #breaks = unique(as.integer(cty_ints$brks)),
                             palette = brewer.pal(n = 9, name = "BuPu"),
                             colorNA = 'grey90',
                             textNA = "NA",
                             title="no. articles",
                             legend.is.portrait=FALSE) +
               tm_layout(main.title = 'study areas at local, regional, and national scales',
                         main.title.position = c('left','top'),
                         legend.position = c('left','bottom'),
                         legend.bg.color = 'white',
                         legend.bg.alpha = 0.5,
                         frame = FALSE)

# save figure
  tmap_save(country.map, filename = paste0(image.dir,"study_area_articles_per_country.png"),
            height = 6, width = 8, units = 'in', dpi = 600)
```

```
  tmap_save(country.map, filename = paste0(image.dir,"study_area_articles_per_country.svg"),
            height = 6, width = 8, units = 'in')

# show here
  country.map
```



Save map as shapefile, for use in the ArcPro version of this visualized map.

```
# save as shapefile
  sf::st_write(all_papers, paste0(data.dir,"all_papers_poly.shp"))
```

6 Study domain map

Here, we map the numbers of articles per country by first making summary tables of counts.

```
# Get a count of countries
  country.domain <- ddply(study.country, .(region, iso3, iso2),
                          summarize,
                          # number of articles
                          n_papers = length(unique(uid)),
                          # percent of articles
                          perc_papers = length(unique(uid))/length(unique(full.list.long$uid)),
                          # count of unique uid when domain is "freshwater"
                          n_freshwater = length(unique(uid[domain == "freshwater"])),
                          # count of unique uid when domain is "marine"
                          n_marine = length(unique(uid[domain == "marine"])),
```

```
                          # count of unique uid when domain is "terrestrial"
                          n_terrestrial = length(unique(uid[domain == "terrestrial"]))
)
```

Next, we match the country names using a worldmap within the tmap package. We make two matching attempts via two data entries: country name and ISO3 code.

```
# match country names to world map using names
  data(World)
  country.domain$name <- country.domain$region
  domain_papers <- left_join(World, country.domain, by = "name")

# rejoin for mismatch via ISO3
  missing_countries <- anti_join(country.domain, World, by = "name")
  missing_countries$iso_a3 <- missing_countries$iso3
  all_domain <- left_join(domain_papers,missing_countries, by = "iso_a3") %>%
                          mutate(name = name.x,
                                 region = coalesce(region.x, region.y),
                                 iso2 = coalesce(iso2.x, iso2.y),
                                 iso3 = coalesce(iso3.x, iso3.y),
                                 region = coalesce(region.x, region.y),
                                 n_papers = coalesce(n_papers.x, n_papers.y),
                                 perc_papers = coalesce(perc_papers.x, perc_papers.y),
                                 n_freshwater = coalesce(n_freshwater.x, n_freshwater.y),
                                 n_marine = coalesce(n_marine.x, n_marine.y),
                                 n_terrestrial = coalesce(n_terrestrial.x, n_terrestrial.y)
                                 )

# drop columns with names ending in ".x" and ".y"
  columns_to_drop <- grepl("\\.x$|\\.y$", names(all_domain))

# Remove the identified columns
  all_domain <- all_domain[, !columns_to_drop]

# convert counts of 0 for each domain to NA
  all_domain$n_freshwater <- ifelse(all_domain$n_freshwater == 0, NA, all_domain$n_freshwater)
  all_domain$n_marine <- ifelse(all_domain$n_marine == 0, NA, all_domain$n_marine)
  all_domain$n_terrestrial <- ifelse(all_domain$n_terrestrial == 0, NA, all_domain$n_terrestrial)

# get number of countries for which names and ISO3 matched
  paste(nrow(all_domain[!is.na(all_domain$region),]),
        "out of",length(unique(country.domain$region)),
        "country names were matchable to the world map")
```

```
## [1] "166 out of 196 country names were matchable to the world map"
```

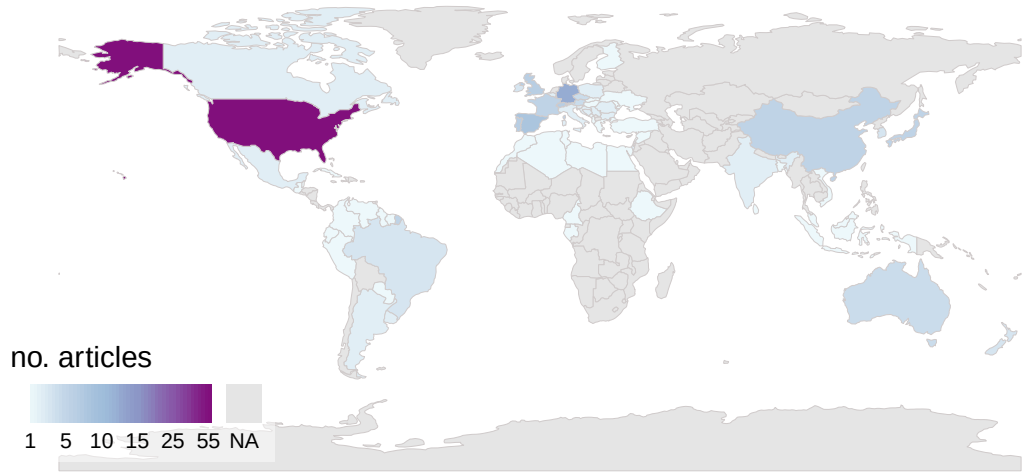
Make a multi-plot of maps.

```
# set mode to plotting
  tmap_mode("plot")
  #tmap_mode("view")  # set to viewing for interactive map

# plot using `tmap`
```

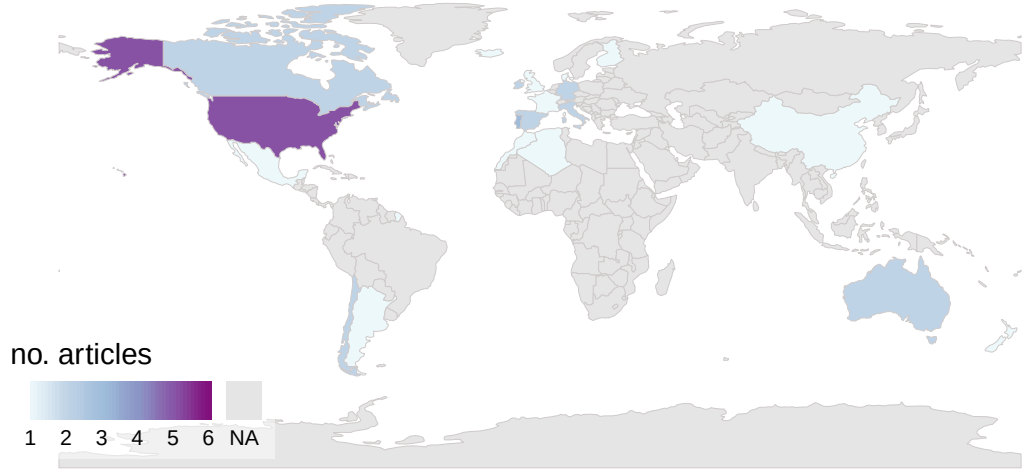
```
# freshwater
  fr_ints <- classIntervals(all_domain$n_freshwater, 6, style = "jenks")
  fresh.map <- tm_shape(all_domain) +
               tm_borders("#d0cbcb", alpha = 1, lwd = 0.5) +
               tm_polygons("n_freshwater", style = "cont",
                           breaks = c(1,5,10,15,25,55), # manual breaks
                           palette = brewer.pal(n = 6, name = "BuPu"),
                           colorNA = 'grey90',
                           textNA = "NA",
                           title="no. articles",
                           legend.is.portrait=FALSE) +
               tm_layout(main.title = 'freshwater',
                         main.title.position = c('left','top'),
                         legend.position = c('left','bottom'),
                         legend.bg.color = 'white',
                         legend.bg.alpha = 0.5,
                         frame = FALSE)
# marine
  ma_ints <- classIntervals(all_domain$n_marine, 5, style = "jenks")
  mar.map <- tm_shape(all_domain) +
             tm_borders("#d0cbcb", alpha = 1, lwd = 0.5) +
             tm_polygons("n_marine", style = "cont",
                         breaks = c(1,2,3,4,5,6),  # manual breaks
                         #breaks = unique(as.integer(ma_ints$brks)),
                         palette = brewer.pal(n = 6, name = "BuPu"),
                         colorNA = 'grey90',
                         textNA = "NA",
                         title="no. articles",
                         legend.is.portrait=FALSE) +
             tm_layout(main.title = 'marine',
                       main.title.position = c('left','top'),
                       legend.position = c('left','bottom'),
                       legend.bg.color = 'white',
                       legend.bg.alpha = 0.5,
                       frame = FALSE)
# terrestrial
  te_ints <- classIntervals(all_domain$n_terrestrial, 5, style = "jenks")
  terr.map <- tm_shape(all_domain)  +
              tm_borders("#d0cbcb", alpha = 1, lwd = 0.5) +
              tm_polygons("n_terrestrial", style = "cont",
                          breaks = c(1,10,30,50,100,230), # manual breaks
                          #breaks = unique(as.integer(te_ints$brks)),
                          palette = brewer.pal(n = 6, name = "BuPu"),
                          colorNA = 'grey90',
                            textNA = "NA",
                          title="no. articles",
                          legend.is.portrait=FALSE) +
              tm_layout(main.title = 'terrestrial',
                        main.title.position = c('left','top'),
                        legend.position = c('left','bottom'),
                        legend.bg.color = 'white',
                        legend.bg.alpha = 0.5,
                        frame = FALSE)
```

```
# compile all together
  domain.fig <- tmap_arrange(fresh.map, mar.map, terr.map, nrow = 3)

# save figure
  tmap_save(domain.fig, filename = paste0(image.dir,"domain_articles_per_country.png"),
            height = 9, width = 6, units = 'in', dpi = 600)
  tmap_save(domain.fig, filename = paste0(image.dir,"domain_articles_per_country.svg"),
            height = 9, width = 6, units = 'in')

# show here
  domain.fig
```

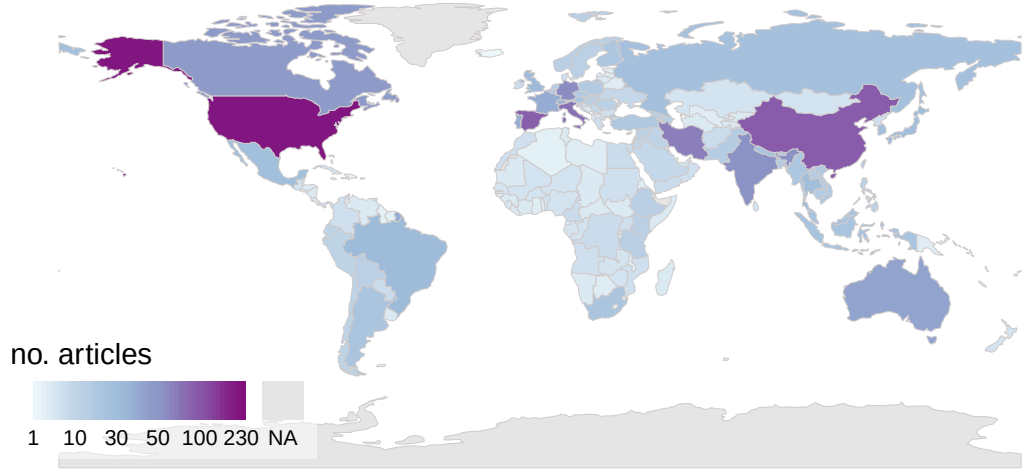
freshwater



marine



terrestrial



7 Study taxa map

Here, we map the numbers of articles per country by first making summary tables of counts.

```
# Get a count of countries
  country.taxa <- ddply(study.country, .(region, iso3, iso2),
                        summarize,
                        # number of articles
                        n_papers = length(unique(uid)),
                        # percent of articles
                        perc_papers = length(unique(uid))/length(unique(full.list.long$uid)),
                        # count of unique uid for each taxa
                        n_amphibians = length(unique(uid[taxa == "amphibians"])),
                        n_birds = length(unique(uid[taxa == "birds"])),
                        n_fish = length(unique(uid[taxa == "fish"])),
                        n_plants = length(unique(uid[taxa == "herbaceous plants"])),
                        n_inverts = length(unique(uid[taxa == "invertebrates"])),
                        n_mammals = length(unique(uid[taxa == "mammals"])),
                        n_micro = length(unique(uid[taxa == "microorganisms"])),
                        n_reptiles = length(unique(uid[taxa == "reptiles"])),
                        n_trees = length(unique(uid[taxa == "trees/shrubs"]))
                        )
```

Next, we match the country names using a worldmap within the tmap package. We make two matching attempts via two data entries: country name and ISO3 code.

```
# match country names to world map using names
  data(World)
  country.taxa$name <- country.taxa$region
  taxa_papers <- left_join(World, country.taxa, by = "name")

# rejoin for mismatch via ISO3
  missing_countries <- anti_join(country.taxa, World, by = "name")
  missing_countries$iso_a3 <- missing_countries$iso3
  all_taxa <- left_join(taxa_papers,missing_countries, by = "iso_a3")%>%
                        mutate(name = name.x,
                               region = coalesce(region.x, region.y),
                               iso2 = coalesce(iso2.x, iso2.y),
                               iso3 = coalesce(iso3.x, iso3.y),
                               region = coalesce(region.x, region.y),
                               n_papers = coalesce(n_papers.x, n_papers.y),
                               perc_papers = coalesce(perc_papers.x, perc_papers.y),
                               n_amphibians = coalesce(n_amphibians.x, n_amphibians.y),
                               n_birds = coalesce(n_birds.x, n_birds.y),
                               n_fish = coalesce(n_fish.x,n_fish.y),
                               n_plants = coalesce(n_plants.x,n_plants.y),
                               n_inverts = coalesce(n_inverts.x,n_inverts.y),
                               n_mammals = coalesce(n_mammals.x,n_mammals.y),
                               n_micro = coalesce(n_micro.x,n_micro.y),
                               n_reptiles = coalesce(n_reptiles.x,n_reptiles.y),
                               n_trees = coalesce(n_trees.x,n_trees.y)
                               )
```

```
# drop columns with names ending in ".x" and ".y"
  columns_to_drop <- grepl("\\.x$|\\.y$", names(all_taxa))

# Remove the identified columns
  all_taxa <- all_taxa[, !columns_to_drop]

# convert counts of 0 for each taxa to NA
  all_taxa$n_amphibians <- ifelse(all_taxa$n_amphibians == 0, NA, all_taxa$n_amphibians)
  all_taxa$n_birds <- ifelse(all_taxa$n_birds == 0, NA, all_taxa$n_birds)
  all_taxa$n_fish <- ifelse(all_taxa$n_fish == 0, NA, all_taxa$n_fish)
  all_taxa$n_plants <- ifelse(all_taxa$n_plants == 0, NA, all_taxa$n_plants)
  all_taxa$n_inverts <- ifelse(all_taxa$n_inverts == 0, NA, all_taxa$n_inverts)
  all_taxa$n_mammals <- ifelse(all_taxa$n_mammals == 0, NA, all_taxa$n_mammals)
  all_taxa$n_micro <- ifelse(all_taxa$n_micro == 0, NA, all_taxa$n_micro)
  all_taxa$n_reptiles <- ifelse(all_taxa$n_reptiles == 0, NA, all_taxa$n_reptiles)
  all_taxa$n_trees <- ifelse(all_taxa$n_trees == 0, NA, all_taxa$n_trees)

# get number of countries for which names and ISO3 matched
  paste(nrow(all_taxa[!is.na(all_taxa$region),]),
        "out of",length(unique(country.taxa$region)),
        "country names were matchable to the world map")
```

```
## [1] "166 out of 196 country names were matchable to the world map"
```

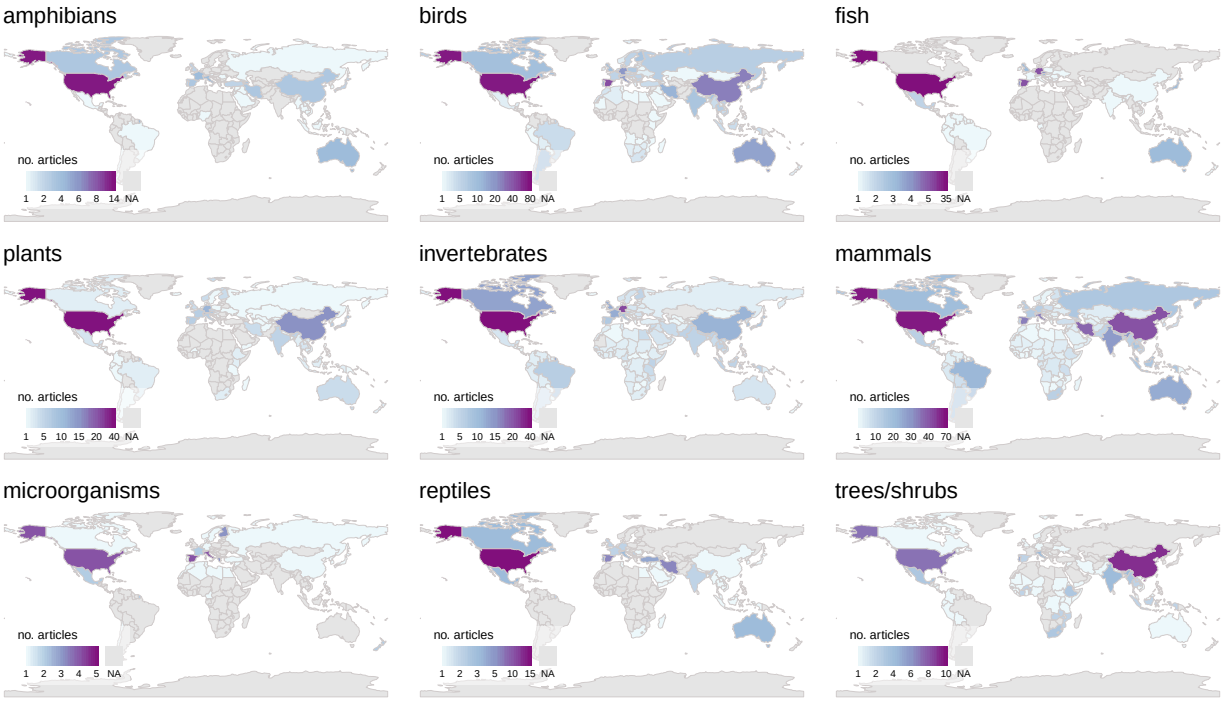
Next, we make a function for maps to look similar to each other, with the exception of unique legends, using manual breaks.

```
# create a mapping style function
  require(tmap)
  require(RColorBrewer)
  require(classInt)

  cat_map <- function(data, variable, title, manualbreaks) {
                      map <- tm_shape(data) +
                             tm_borders("#d0cbcb", alpha = 1, lwd = 0.5) +
                             tm_polygons(variable, style = "cont",
                                         breaks = manualbreaks, # manual breaks
                                         palette = brewer.pal(n = length(manualbreaks),
                                                              name = "BuPu"),
                                         colorNA = 'grey90',
                                         textNA = "NA",
                                         title="no. articles",
                                         legend.is.portrait=FALSE) +
                             tm_layout(main.title = title,
                                       main.title.position = c('left','top'),
                                       main.title.size = 0.75,
                                       legend.position = c('left','bottom'),
                                       legend.bg.color = 'white',
                                       legend.bg.alpha = 0.5,
                                       legend.title.size = 1,
                                       legend.height = 0.3,
                                       legend.width = 0.75,
                                       legend.text.size = 0.75,
```

```
                                        frame = FALSE)
                    return(map)
                   }
```

And then we produce maps using the above function.

```
# make maps
  amph.map <- cat_map(all_taxa, "n_amphibians", "amphibians",c(1,2,4,6,8,14))
  bird.map <- cat_map(all_taxa, "n_birds", "birds",c(1,5,10,20,40,80))
  fish.map <- cat_map(all_taxa, "n_fish", "fish",c(1,2,3,4,5,35))
  plnt.map <- cat_map(all_taxa, "n_plants", "plants",c(1,5,10,15,20,40))
  invt.map <- cat_map(all_taxa, "n_inverts", "invertebrates",c(1,5,10,15,20,40))
  mamm.map <- cat_map(all_taxa, "n_mammals", "mammals",c(1,10,20,30,40,70))
  micr.map <- cat_map(all_taxa, "n_micro", "microorganisms",c(1,2,3,4,5))
  rept.map <- cat_map(all_taxa, "n_reptiles", "reptiles",c(1,2,3,5,10,15))
  tree.map <- cat_map(all_taxa, "n_trees", "trees/shrubs",c(1,2,4,6,8,10))

# compile all together
  taxa.fig <- tmap_arrange(amph.map, bird.map, fish.map,
                           plnt.map, invt.map, mamm.map,
                           micr.map, rept.map, tree.map,
                           ncol = 3)

# save figure
  tmap_save(taxa.fig, filename = paste0(image.dir,"taxa_articles_per_country.png"),
            height = 4, width = 7, units = 'in',
            dpi = 600)
  tmap_save(taxa.fig, filename = paste0(image.dir,"taxa_articles_per_country.svg"),
            height = 4, width = 7, units = 'in')

# show here
  taxa.fig
```



8 Study focus map

Here, we map the numbers of articles per country by first making summary tables of counts.

```
# Get a count of countries
  country.focus <- ddply(study.country, .(region, iso3, iso2),
                    summarize,
                    # number of articles
                    n_papers = length(unique(uid)),
                    # percent of articles
                    perc_papers = length(unique(uid))/length(unique(full.list.long$uid)),
                    # count of unique uid for each study_focus
                    n_conf = length(unique(uid[study_focus == "conflict/collisions"])),
                    n_cons = length(unique(uid[study_focus == "conservation"])),
                    n_dist = length(unique(uid[study_focus == "disturbance/habitat change"])),
                    n_expl = length(unique(uid[study_focus == "exploratory"])),
                    n_food = length(unique(uid[study_focus == "food/economics"])),
                    n_heal = length(unique(uid[study_focus == "human health/safety"])),
                    n_inva = length(unique(uid[study_focus == "invasions"])),
                    n_rest = length(unique(uid[study_focus == "reintroduction/restoration"]))
                        )
```

Next, we match the country names using a worldmap within the tmap package. We make two matching attempts via two data entries: country name and ISO3 code.

```
# match country names to world map using names
  data(World)
  country.focus$name <- country.focus$region
  focus_papers <- left_join(World, country.focus, by = "name")
```

```r
# rejoin for mismatch via ISO3
  missing_countries <- anti_join(country.focus, World, by = "name")
  missing_countries$iso_a3 <- missing_countries$iso3
  all_focus <- left_join(focus_papers,missing_countries, by = "iso_a3")%>%
                        mutate(name = name.x,
                               region = coalesce(region.x, region.y),
                               iso2 = coalesce(iso2.x, iso2.y),
                               iso3 = coalesce(iso3.x, iso3.y),
                               region = coalesce(region.x, region.y),
                               n_papers = coalesce(n_papers.x, n_papers.y),
                               perc_papers = coalesce(perc_papers.x, perc_papers.y),
                               n_conf = coalesce(n_conf.x, n_conf.y),
                               n_cons = coalesce(n_cons.x, n_cons.y),
                               n_dist = coalesce(n_dist.x,n_dist.y),
                               n_expl = coalesce(n_expl.x,n_expl.y),
                               n_food = coalesce(n_food.x,n_food.y),
                               n_heal = coalesce(n_heal.x,n_heal.y),
                               n_inva = coalesce(n_inva.x,n_inva.y),
                               n_rest = coalesce(n_rest.x,n_rest.y)
                               )

# drop columns with names ending in ".x" and ".y"
  columns_to_drop <- grepl("\\.x$|\\.y$", names(all_focus))

# Remove the identified columns
  all_focus <- all_focus[, !columns_to_drop]

# convert counts of 0 for each focus to NA
  all_focus$n_conf <- ifelse(all_focus$n_conf == 0, NA, all_focus$n_conf)
  all_focus$n_cons <- ifelse(all_focus$n_cons == 0, NA, all_focus$n_cons)
  all_focus$n_dist <- ifelse(all_focus$n_dist == 0, NA, all_focus$n_dist)
  all_focus$n_expl <- ifelse(all_focus$n_expl == 0, NA, all_focus$n_expl)
  all_focus$n_food <- ifelse(all_focus$n_food == 0, NA, all_focus$n_food)
  all_focus$n_heal <- ifelse(all_focus$n_heal == 0, NA, all_focus$n_heal)
  all_focus$n_inva <- ifelse(all_focus$n_inva == 0, NA, all_focus$n_inva)
  all_focus$n_rest <- ifelse(all_focus$n_rest == 0, NA, all_focus$n_rest)

# get number of countries for which names and ISO3 matched
  paste(nrow(all_focus[!is.na(all_focus$region),]),
        "out of",length(unique(country.focus$region)),
        "country names were matchable to the world map")
```

```
## [1] "166 out of 196 country names were matchable to the world map"
```

Next, we make a function for maps to look similar to each other, with the exception of unique legends, using manual breaks.

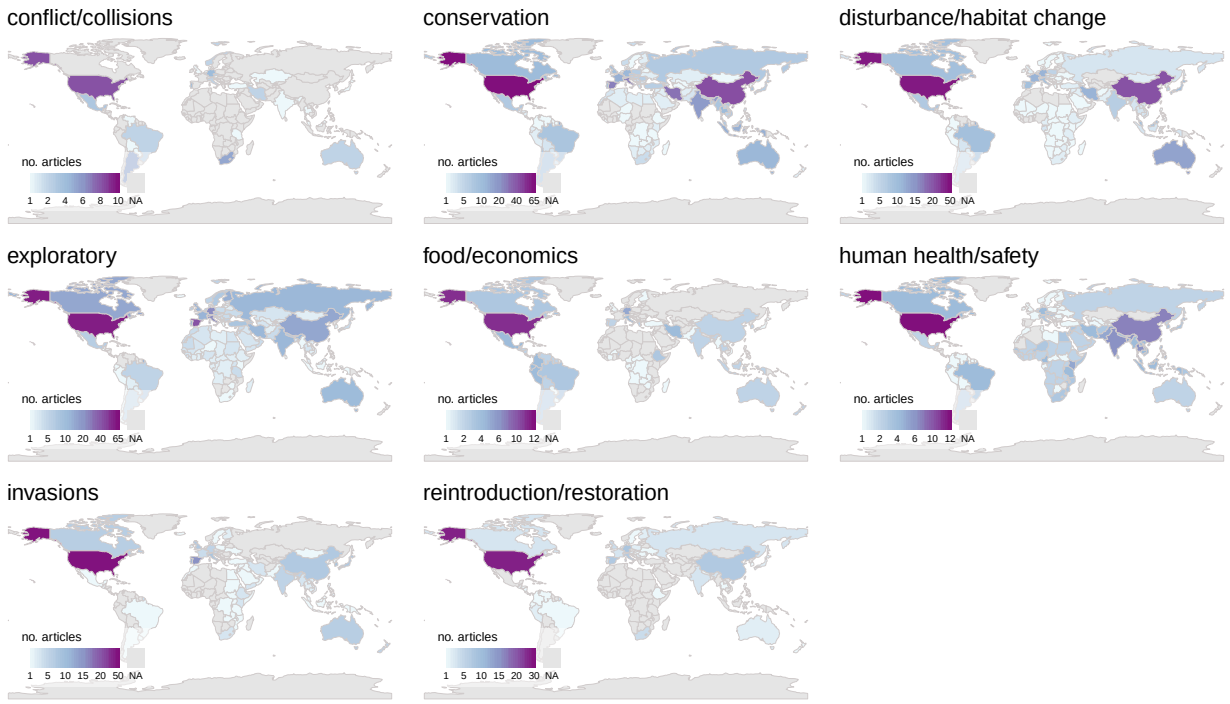
```r
# create a mapping style function
  require(tmap)
  require(RColorBrewer)
  require(classInt)
```

```
  cat_map <- function(data, variable, title, manualbreaks) {
                      map <- tm_shape(data) +
                             tm_borders("#d0cbcb", alpha = 1, lwd = 0.5) +
                             tm_polygons(variable, style = "cont",
                                         breaks = manualbreaks, # manual breaks
                                         palette = brewer.pal(n = length(manualbreaks),
                                                              name = "BuPu"),
                                         colorNA = 'grey90',
                                         textNA = "NA",
                                         title="no. articles",
                                         legend.is.portrait=FALSE) +
                             tm_layout(main.title = title,
                                       main.title.position = c('left','top'),
                                       main.title.size = 0.75,
                                       legend.position = c('left','bottom'),
                                       legend.bg.color = 'white',
                                       legend.bg.alpha = 0.5,
                                       legend.title.size = 1,
                                       legend.height = 0.3,
                                       legend.width = 0.75,
                                       legend.text.size = 0.75,
                                       frame = FALSE)
                      return(map)
                     }
```

Make a multi-plot of maps.

```
# example code to get ideas for manual breaks
  #pretty(c(1,max(all_focus$n_conf,na.rm = TRUE)),n=6)

# make maps
  conf.map <- cat_map(all_focus, "n_conf", "conflict/collisions",c(1,2,4,6,8,10))
  cons.map <- cat_map(all_focus, "n_cons", "conservation",c(1,5,10,20,40,65))
  dist.map <- cat_map(all_focus, "n_dist", "disturbance/habitat change",c(1,5,10,15,20,50))
  expl.map <- cat_map(all_focus, "n_expl", "exploratory",c(1,5,10,20,40,65))
  food.map <- cat_map(all_focus, "n_food", "food/economics",c(1,2,4,6,10,12))
  heal.map <- cat_map(all_focus, "n_heal", "human health/safety",c(1,2,4,6,10,12))
  inva.map <- cat_map(all_focus, "n_inva", "invasions",c(1,5,10,15,20,50))
  rest.map <- cat_map(all_focus, "n_rest", "reintroduction/restoration",c(1,5,10,15,20,30))

# compile all together
  focus.fig <- tmap_arrange(conf.map, cons.map, dist.map,
                            expl.map, food.map, heal.map,
                            inva.map, rest.map,
                            ncol = 3)

# save figure
  tmap_save(focus.fig, filename = paste0(image.dir,"focus_articles_per_country.png"),
            height = 4, width = 7, units = 'in',
            dpi = 600)
  tmap_save(focus.fig, filename = paste0(image.dir,"focus_articles_per_country.svg"),
            height = 4, width = 7, units = 'in')
```

```
# show here
  focus.fig
```



9 Study scale map

Here, we map the numbers of articles per country by first making summary tables of counts.

```
# Get a count of countries
  country.scale <- ddply(study.country, .(region, iso3, iso2),
                         summarize,
                         # number of articles
                         n_papers = length(unique(uid)),
                         # percent of articles
                         perc_papers = length(unique(uid))/length(unique(full.list.long$uid)),
                         # count of unique uid for each study_area_scale
                         n_loc = length(unique(uid[study_area_scale == "local"])),
                         n_reg = length(unique(uid[study_area_scale == "regional"])),
                         n_ntl = length(unique(uid[study_area_scale == "national"])),
                         n_mul = length(unique(uid[study_area_scale == "multinational"]))
                         )
```

Next, we match the country names using a worldmap within the tmap package. We make two matching attempts via two data entries: country name and ISO3 code.

```
# match country names to world map using names
  data(World)
  country.scale$name <- country.scale$region
  scale_papers <- left_join(World, country.scale, by = "name")
```

```
# rejoin for mismatch via ISO3
  missing_countries <- anti_join(country.scale, World, by = "name")
  missing_countries$iso_a3 <- missing_countries$iso3
  all_scale <- left_join(scale_papers,missing_countries, by = "iso_a3")%>%
                         mutate(name = name.x,
                                region = coalesce(region.x, region.y),
                                iso2 = coalesce(iso2.x, iso2.y),
                                iso3 = coalesce(iso3.x, iso3.y),
                                region = coalesce(region.x, region.y),
                                n_papers = coalesce(n_papers.x, n_papers.y),
                                perc_papers = coalesce(perc_papers.x, perc_papers.y),
                                n_loc = coalesce(n_loc.x, n_loc.y),
                                n_reg = coalesce(n_reg.x, n_reg.y),
                                n_ntl = coalesce(n_ntl.x,n_ntl.y),
                                n_mul = coalesce(n_mul.x,n_mul.y)
                                )

# drop columns with names ending in ".x" and ".y"
  columns_to_drop <- grepl("\\.x$|\\.y$", names(all_scale))

# Remove the identified columns
  all_scale <- all_scale[, !columns_to_drop]

# convert counts of 0 for each focus to NA
  all_scale$n_loc <- ifelse(all_scale$n_loc == 0, NA, all_scale$n_loc)
  all_scale$n_reg <- ifelse(all_scale$n_reg == 0, NA, all_scale$n_reg)
  all_scale$n_ntl <- ifelse(all_scale$n_ntl == 0, NA, all_scale$n_ntl)
  all_scale$n_mul <- ifelse(all_scale$n_mul == 0, NA, all_scale$n_mul)

# get number of countries for which names and ISO3 matched
  paste(nrow(all_scale[!is.na(all_scale$region),]),
        "out of",length(unique(country.scale$region)),
        "country names were matchable to the world map")
```

```
## [1] "166 out of 196 country names were matchable to the world map"
```

Next, we make the continental scale map.

```
# Get a count of continents
  continents <- ddply(study.continent, .(region),
                      summarize,
                      # number of articles
                      n_papers = length(unique(uid)),
                      # percent of articles
                      perc_papers = length(unique(uid))/length(unique(full.list.long$uid)),
                      # count of unique uid for each study_area_scale
                      n_con = length(unique(uid[study_area_scale == "continental"]))
                      )

# add a row with NA for Antarctica
  continents[nrow(continents) + 1,] <- c("Antarctica",NA,NA,NA,NA)
```

```r
# change structure
  continents$n_papers <- as.integer(continents$n_papers)
  continents$perc_papers <- as.numeric(continents$perc_papers)
  continents$n_con <- as.integer(continents$n_con)

# view table
  kableExtra::kbl(continents, booktabs = T,longtable = T) %>%
              kable_styling(latex_options = c("striped","repeat_header"))
```

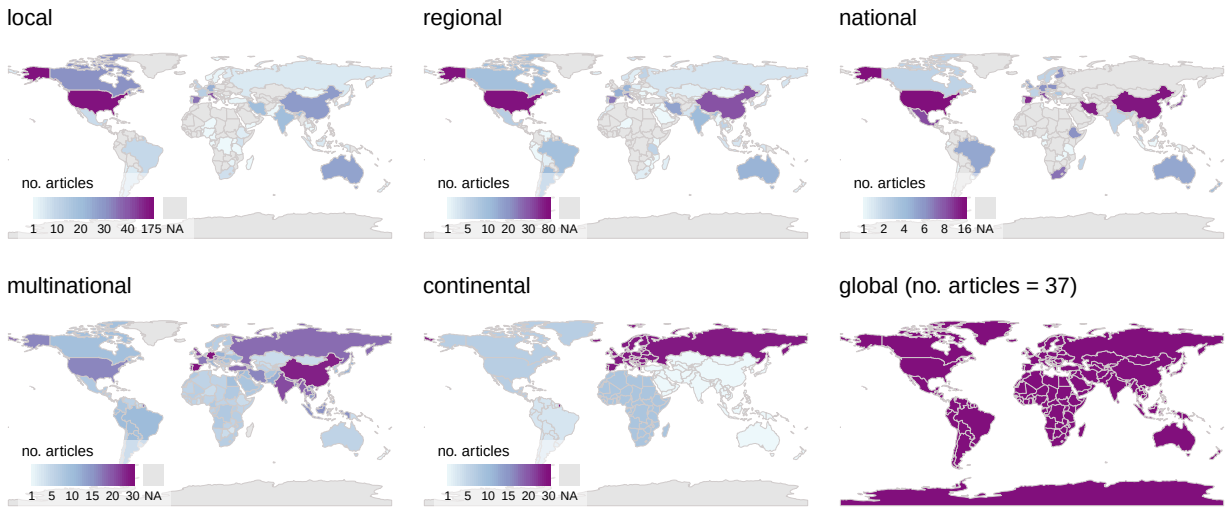
region	n_papers	perc_papers	n_con
Africa	8	0.0055983	8
Asia	1	0.0006998	1
Europe	28	0.0195941	28
North America	6	0.0041987	6
Oceania	1	0.0006998	1
South America	3	0.0020994	3
Antarctica	NA	NA	NA

Next, we make a function for maps to look similar to each other, with the exception of unique legends, using manual breaks.

```r
# create a mapping style function
  require(tmap)
  require(RColorBrewer)
  require(classInt)

  cat_map <- function(data, variable, title, manualbreaks) {
                      map <- tm_shape(data) +
                             tm_borders("#d0cbcb", alpha = 1, lwd = 0.5) +
                             tm_polygons(variable, style = "cont",
                                         breaks = manualbreaks, # manual breaks
                                         palette = brewer.pal(n = length(manualbreaks),
                                                              name = "BuPu"),
                                         colorNA = 'grey90',
                                         textNA = "NA",
                                         title="no. articles",
                                         legend.is.portrait=FALSE) +
                             tm_layout(main.title = title,
                                       main.title.position = c('left','top'),
                                       main.title.size = 0.75,
                                       legend.position = c('left','bottom'),
                                       legend.bg.color = 'white',
                                       legend.bg.alpha = 0.5,
                                       legend.title.size = 1,
                                       legend.height = 0.3,
                                       legend.width = 0.75,
                                       legend.text.size = 0.75,
                                       frame = FALSE)
                      return(map)
                     }
```

Make a multi-plot of maps.

```
# example code to get ideas for manual breaks
  #pretty(c(1,max(all_scale$n_loc,na.rm = TRUE)),n=6)

# make maps
  loc.map <- cat_map(all_scale, "n_loc", "local", c(1,10,20,30,40,175))
  reg.map <- cat_map(all_scale, "n_reg", "regional", c(1,5,10,20,30,80))
  ntl.map <- cat_map(all_scale, "n_ntl", "national", c(1,2,4,6,8,16))
  mul.map <- cat_map(all_scale, "n_mul", "multinational", c(1,5,10,15,20,30))

# continental scale map
  # Join data to continent polygons
    continents$continent <- continents$region
    contmap <- merge(World, continents, by = "continent")

  # Create the map
    con.map <- cat_map(contmap, "n_papers", "continental", c(1,5,10,15,20,30))

# global scale map (single color) with custom legend
  glb.map <- tm_shape(World) +
              tm_borders("#d0cbcb", alpha = 1, lwd = 0.5) +
              tm_fill(col='#810f7c',
                      title="no. articles",
                      legend.is.portrait=FALSE) +
              tm_layout(main.title = paste0("global (no. articles = ",
                                            length(unique(study.global$uid)),")"),
                      main.title.position = c('left','top'),
                      main.title.size = 0.75,
                      legend.position = c('left','bottom'),
                      legend.bg.color = 'white',
                      legend.bg.alpha = 0.5,
                      legend.title.size = 1,
                      legend.height = 0.3,
                      legend.width = 0.75,
                      legend.text.size = 0.75,
                      frame = FALSE)

# compile all together
  scale.fig <- tmap_arrange(loc.map, reg.map,
                            ntl.map, mul.map,
                            con.map,
                            glb.map,
                            ncol = 3)

# save figure
  tmap_save(scale.fig, filename = paste0(image.dir,"scale_articles_per_country.png"),
            height = 3, width = 7, units = 'in',
            dpi = 600)
  tmap_save(scale.fig, filename = paste0(image.dir,"scale_articles_per_country.svg"),
            height = 3, width = 7, units = 'in')

# show here
  scale.fig
```



10 Mapping first published years of human predictor use by scale

Here, we map the years for the first time human predictors are used in SDMs around the world at various spatial scales.

```
# Get a count of countries
  country.yrs <- ddply(study.country, .(region, iso3, iso2),
                       summarize,
                       # first year of publication by scale
                       yr_loc = min(year[study_area_scale == "local"]),
                       yr_reg = min(year[study_area_scale == "regional"]),
                       yr_ntl = min(year[study_area_scale == "national"]),
                       yr_mul = min(year[study_area_scale == "multinational"])
                       )
```

Next, we match the country names using a worldmap within the tmap package. We make two matching attempts via two data entries: country name and ISO3 code.

```
# match country names to world map using names
  data(World)
  country.yrs$name <- country.yrs$region
  year_papers <- left_join(World, country.yrs, by = "name")


# rejoin for mismatch via ISO3
  missing_countries <- anti_join(country.yrs, World, by = "name")
  missing_countries$iso_a3 <- missing_countries$iso3
  yrs_scale <- left_join(year_papers,missing_countries, by = "iso_a3")%>%
                       mutate(name = name.x,
                              region = coalesce(region.x, region.y),
                              iso2 = coalesce(iso2.x, iso2.y),
                              iso3 = coalesce(iso3.x, iso3.y),
                              region = coalesce(region.x, region.y),
                              yr_loc = coalesce(yr_loc.x, yr_loc.y),
```

```
                                yr_reg = coalesce(yr_reg.x, yr_reg.y),
                                yr_ntl = coalesce(yr_ntl.x,yr_ntl.y),
                                yr_mul = coalesce(yr_mul.x,yr_mul.y)
                                )

# drop columns with names ending in ".x" and ".y"
  columns_to_drop <- grepl("\\.x$|\\.y$", names(yrs_scale))

# Remove the identified columns
  yrs_scale <- yrs_scale[, !columns_to_drop]

# convert counts of 0 for each focus to NA
  yrs_scale$yr_loc <- ifelse(yrs_scale$yr_loc == 0, NA, yrs_scale$yr_loc)
  yrs_scale$yr_reg <- ifelse(yrs_scale$yr_reg == 0, NA, yrs_scale$yr_reg)
  yrs_scale$yr_ntl <- ifelse(yrs_scale$yr_ntl == 0, NA, yrs_scale$yr_ntl)
  yrs_scale$yr_mul <- ifelse(yrs_scale$yr_mul == 0, NA, yrs_scale$yr_mul)

# get number of countries for which names and ISO3 matched
  paste(nrow(yrs_scale[!is.na(yrs_scale$region),]),
        "out of",length(unique(country.yrs$region)),
        "country names were matchable to the world map")
```

```
## [1] "166 out of 196 country names were matchable to the world map"
```

Next, we make the continental scale map.

```
# Get a count of continents
  continent.yrs <- ddply(study.continent, .(region),
                         summarize,
                         ## first year of publication by scale
                         yr_con = min(year[study_area_scale == "continental"])
                         )

# add a row with NA for Antarctica
  continent.yrs[nrow(continent.yrs) + 1,] <- c("Antarctica",NA)

# change structure
  continent.yrs$yr_con <- as.integer(continent.yrs$yr_con)

# view table
  kableExtra::kbl(continent.yrs, booktabs = T,longtable = T) %>%
              kable_styling(latex_options = c("striped","repeat_header"))
```

region	yr_con
Africa	2007
Asia	2018
Europe	2010
North America	2014
Oceania	2018
South America	2017
Antarctica	NA

(continued)

region	yr_con

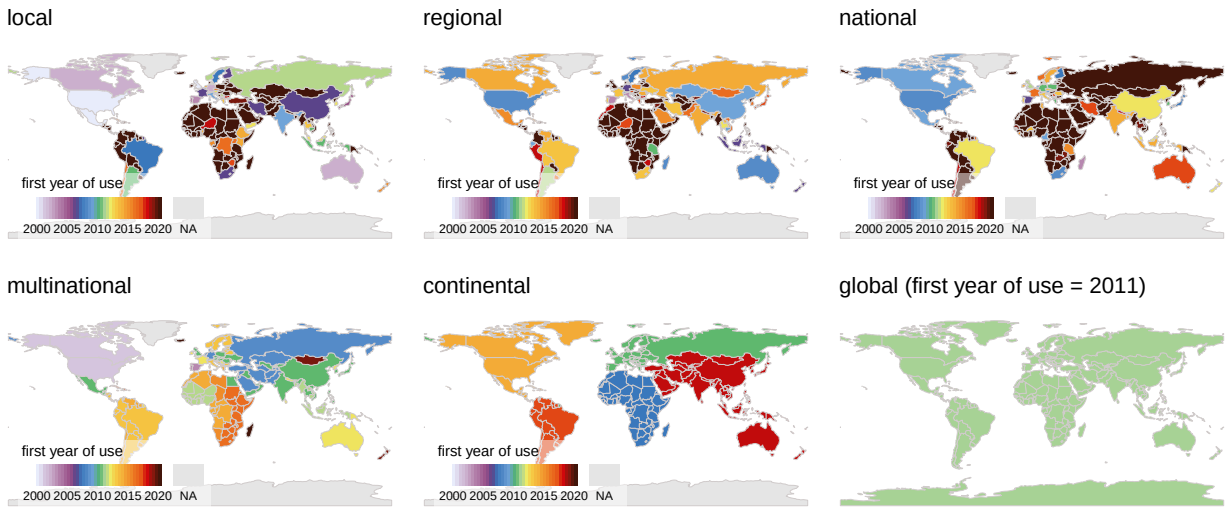
Next, we make a function for maps to look similar to each other, with the exception of unique legends, using manual breaks.

```
# create a mapping style function
  require(tmap)
  require(RColorBrewer)
  require(classInt)

# assign color (colorblind-friendly palette from Paul Tol)
  col.yrs <- colorRampPalette(c('#E8ECFB', '#D9CCE3', '#D1BBD7', '#CAACCB', '#BA8DB4',
                                '#AE76A3', '#AA6F9E', '#994F88', '#882E72', '#1965B0',
                                '#437DBF', '#5289C7', '#6195CF', '#7BAFDE', '#4EB265',
                                '#90C987', '#CAE0AB', '#F7F056', '#F7CB45', '#F6C141',
                                '#F4A736', '#F1932D', '#EE8026', '#E8601C', '#E65518',
                                '#DC050C', '#A5170E', '#72190E', '#42150A'))
  col.yrs <- col.yrs(21)

# custom map function
  cat_map <- function(data, variable, title, manualbreaks) {
                      map <- tm_shape(data) +
                             tm_borders("#d0cbcb", alpha = 1, lwd = 0.5) +
                             tm_polygons(variable, style = "cont",
                                         breaks = manualbreaks, # manual breaks
                                         palette = col.yrs,
                                         colorNA = 'grey90',
                                         textNA = "NA",
                                         title="first year of use",
                                         legend.is.portrait=FALSE,
                                         legend.show = TRUE) +
                             tm_layout(main.title = title,
                                       main.title.position = c('left','top'),
                                       main.title.size = 0.75,
                                       legend.position = c('left','bottom'),
                                       legend.bg.color = 'white',
                                       legend.bg.alpha = 0.5,
                                       legend.title.size = 1,
                                       legend.height = 0.3,
                                       legend.width = 0.75,
                                       legend.text.size = 0.75,
                                       legend.format=list(fun=function(x) formatC(x,
                                                                  digits=0, format="d")),
                                       frame = FALSE)
                      return(map)
                     }
```

Make a multi-plot of maps.

```
# example code to get ideas for manual breaks
  #pretty(c(1,max(yrs_scale$n_loc,na.rm = TRUE)),n=6)
```

```
# make maps
  loc.yr.map <- cat_map(yrs_scale, "yr_loc", "local", seq(2000,2021,5))
  reg.yr.map <- cat_map(yrs_scale, "yr_reg", "regional", seq(2000,2021,5))
  ntl.yr.map <- cat_map(yrs_scale, "yr_ntl", "national", seq(2000,2021,5))
  mul.yr.map <- cat_map(yrs_scale, "yr_mul", "multinational", seq(2000,2021,5))

# continental scale map
  # Join data to continent polygons
    continent.yrs$continent <- continent.yrs$region
    contyrs <- merge(World, continent.yrs, by = "continent")

  # Create the map
    con.yr.map <- cat_map(contyrs, "yr_con", "continental", seq(2000,2021,5))

# activate to get hex for global scale map if value is between two schemes
  # pal.glb <- colorRampPalette(colors = c('#90C987','#F7F056'))(3)
  # scales::show_col(pal.glb)

# global scale map (single color) with custom legend
  glb.yr.map <- tm_shape(World) +
                tm_borders("#d0cbcb", alpha = 1, lwd = 0.5) +
                tm_fill(col="#A7D295",
                        title="first year of use",
                        legend.is.portrait=FALSE) +
                tm_layout(main.title = paste0("global (first year of use = ",
                                              min(study.global$year),")"),
                          main.title.position = c('left','top'),
                          main.title.size = 0.75,
                          legend.position = c('left','bottom'),
                          legend.bg.color = 'white',
                          legend.bg.alpha = 0.5,
                          legend.title.size = 1,
                          legend.height = 0.3,
                          legend.width = 0.75,
                          legend.text.size = 0.75,
                          frame = FALSE)

# compile all together
  years.fig <- tmap_arrange(loc.yr.map, reg.yr.map,
                            ntl.yr.map, mul.yr.map,
                            con.yr.map,
                            glb.yr.map,
                            ncol = 3)

# save figure
  tmap_save(years.fig, filename = paste0(image.dir,"scale_years_map.png"),
            height = 3, width = 7, units = 'in',
            dpi = 600)
  tmap_save(years.fig, filename = paste0(image.dir,"scale_years_map.svg"),
            height = 3, width = 7, units = 'in')

# show here
  years.fig
```



11 Mapping frequency of predictors by study scale

Here, we map the number of unique predictors per country per spatial scale.

```
# Get a count of countries
  country.prd <- ddply(study.country, .(region, iso3, iso2),
                           summarize,
                           # count of unique predictors for study_area_scale
                           pr_loc = length(unique(
                                           predictor[study_area_scale == "local"])),
                           pr_reg = length(unique(
                                           predictor[study_area_scale == "regional"])),
                           pr_ntl = length(unique(
                                           predictor[study_area_scale == "national"])),
                           pr_mul = length(unique(
                                           predictor[study_area_scale == "multinational"]))
                           )
```

Next, we match the country names using a worldmap within the tmap package. We make two matching attempts via two data entries: country name and ISO3 code.

```
# match country names to world map using names
  data(World)
  country.prd$name <- country.prd$region
  pred_papers <- left_join(World, country.prd, by = "name")


# rejoin for mismatch via ISO3
  missing_countries <- anti_join(country.prd, World, by = "name")
  missing_countries$iso_a3 <- missing_countries$iso3
  prd_scale <- left_join(pred_papers,missing_countries, by = "iso_a3")%>%
                           mutate(name = name.x,
                                  region = coalesce(region.x, region.y),
                                  iso2 = coalesce(iso2.x, iso2.y),
```

```
                                iso3 = coalesce(iso3.x, iso3.y),
                                region = coalesce(region.x, region.y),
                                pr_loc = coalesce(pr_loc.x, pr_loc.y),
                                pr_reg = coalesce(pr_reg.x, pr_reg.y),
                                pr_ntl = coalesce(pr_ntl.x,pr_ntl.y),
                                pr_mul = coalesce(pr_mul.x,pr_mul.y)
                                )

# drop columns with names ending in ".x" and ".y"
  columns_to_drop <- grepl("\\.x$|\\.y$", names(prd_scale))

# Remove the identified columns
  prd_scale <- prd_scale[, !columns_to_drop]

# convert counts of 0 for each focus to NA
  prd_scale$pr_loc <- ifelse(prd_scale$pr_loc == 0, NA, prd_scale$pr_loc)
  prd_scale$pr_reg <- ifelse(prd_scale$pr_reg == 0, NA, prd_scale$pr_reg)
  prd_scale$pr_ntl <- ifelse(prd_scale$pr_ntl == 0, NA, prd_scale$pr_ntl)
  prd_scale$pr_mul <- ifelse(prd_scale$pr_mul == 0, NA, prd_scale$pr_mul)

# get number of countries for which names and ISO3 matched
  paste(nrow(prd_scale[!is.na(prd_scale$region),]),
        "out of",length(unique(country.prd$region)),
        "country names were matchable to the world map")
```

```
## [1] "166 out of 196 country names were matchable to the world map"
```

Next, we make the continental scale map.

```
# Get a count of continents
  continent.prd <- ddply(study.continent, .(region),
                         summarize,
                         # count of unique predictors for study_area_scale
                         pr_con = length(unique(
                                         predictor[study_area_scale == "continental"]))
                         )

# add a row with NA for Antarctica
  continent.prd[nrow(continent.prd) + 1,] <- c("Antarctica",NA)

# change structure
  continent.prd$pr_con <- as.integer(continent.prd$pr_con)

# view table
  kableExtra::kbl(continent.prd, booktabs = T,longtable = T) %>%
              kable_styling(latex_options = c("striped","repeat_header"))
```

region	pr_con
Africa	9
Asia	4
Europe	56

(continued)

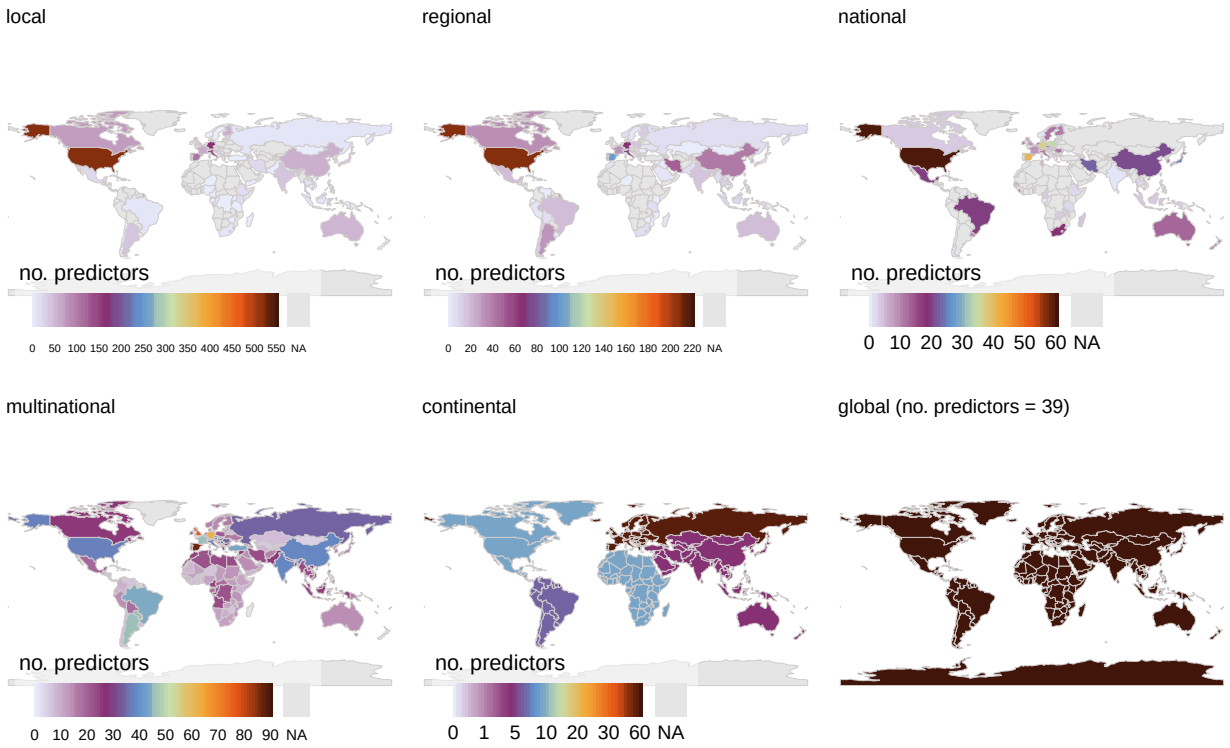
region	pr_con
North America	9
Oceania	4
South America	6
Antarctica	NA

Next, we make a function for maps to look similar to each other, with the exception of unique legends, using manual breaks.

```
# create a mapping style function
  require(tmap)
  require(RColorBrewer)
  require(classInt)

# assign color (colorblind-friendly palette from Paul Tol)
  col.prd <- colorRampPalette(c('#E8ECFB', '#D9CCE3', '#D1BBD7', '#CAACCB', '#BA8DB4',
                                '#AE76A3', '#AA6F9E', '#994F88', '#882E72', '#1965B0',
                                '#437DBF', '#5289C7', '#6195CF', '#7BAFDE', '#4EB265',
                                '#90C987', '#CAE0AB', '#F7F056', '#F7CB45', '#F6C141',
                                '#F4A736', '#F1932D', '#EE8026', '#E8601C', '#E65518',
                                '#DC050C', '#A5170E', '#72190E', '#42150A'))
  col.prd <- col.prd(8)

# map function
  cat_map <- function(data, variable, title, manualbreaks) {
                      map <- tm_shape(data) +
                             tm_borders("#d0cbcb", alpha = 1, lwd = 0.5) +
                             tm_polygons(variable, style = "cont",
                                         breaks = manualbreaks,
                                         palette = col.prd,
                                         colorNA = 'grey90',
                                         textNA = "NA",
                                         title="no. predictors",
                                         legend.is.portrait=FALSE) +
                             tm_layout(main.title = title,
                                       main.title.position = c('left','top'),
                                       main.title.size = 0.75,
                                       legend.position = c('left','bottom'),
                                       legend.bg.color = 'white',
                                       legend.bg.alpha = 0.5,
                                       legend.title.size = 1,
                                       legend.height = 0.3,
                                       legend.width = 0.75,
                                       legend.text.size = 0.75,
                                       legend.format=list(fun=function(x) formatC(x,
                                                          digits=0, format="d")),
                                       frame = FALSE)
                       return(map)
                      }
```

Make a multi-plot of maps.

```
# make maps
  loc.pr.map <- cat_map(prd_scale, "pr_loc", "local",
                        pretty(c(1,max(prd_scale$pr_loc,na.rm = TRUE)),n=8))
  reg.pr.map <- cat_map(prd_scale, "pr_reg", "regional",
                        pretty(c(1,max(prd_scale$pr_reg,na.rm = TRUE)),n=8))
  ntl.pr.map <- cat_map(prd_scale, "pr_ntl", "national",
                        pretty(c(1,max(prd_scale$pr_ntl,na.rm = TRUE)),n=8))
  mul.pr.map <- cat_map(prd_scale, "pr_mul", "multinational",
                        pretty(c(1,max(prd_scale$pr_mul,na.rm = TRUE)),n=8))

# continental scale map
  # Join data to continent polygons
    continent.prd$continent <- continent.prd$region
    contprd <- merge(World, continent.prd, by = "continent")

  # Create the map
    con.pr.map <- cat_map(contprd, "pr_con", "continental", c(0,1,5,10,20,30,60))

# global scale map (single color) with custom legend
  glb.pr.map <- tm_shape(World) +
                tm_borders("#d0cbcb", alpha = 1, lwd = 0.5) +
                tm_fill(col='#42150A',
                        title="first year of use",
                        legend.is.portrait=FALSE) +
                tm_layout(main.title = paste0("global (no. predictors = ",
                                              length(unique(study.global$predictor)),")"),
                          main.title.position = c('left','top'),
                          main.title.size = 0.75,
                          legend.position = c('left','bottom'),
                          legend.bg.color = 'white',
                          legend.bg.alpha = 0.5,
                          legend.title.size = 1,
                          legend.height = 0.3,
                          legend.width = 0.75,
                          legend.text.size = 0.75,
                          frame = FALSE)

# compile all together
  preds.fig <- tmap_arrange(loc.pr.map, reg.pr.map,
                            ntl.pr.map, mul.pr.map,
                            con.pr.map,
                            glb.pr.map,
                            ncol = 3)

# save figure
  tmap_save(preds.fig, filename = paste0(image.dir,"scale_predictors_map.png"),
            height = 3, width = 7, units = 'in',
            dpi = 600)
  tmap_save(preds.fig, filename = paste0(image.dir,"scale_predictors_map.svg"),
            height = 3, width = 7, units = 'in')

# show here
  preds.fig
```



12 SDM algorithm use across articles

Here, we edit and summarize SDM algorithm use. First, we make a subset dataframe.

```
# Extract table and predictors from relevant papers
  sdm.df <- subset(yes.df,
                   select = c("uid","SDM_algorithm","SDM_algorithm_ensembles"))
```

Edit algorithm names.

```
# expand rows
  sdm.df <- separate_rows(sdm.df, SDM_algorithm_ensembles, sep="; ",convert = TRUE)

# Create a vector of patterns to search and replace (search on left, replace on right)
  patterns <- c(
                "maxent" = "Maxent",
                "MAXENT" = "Maxent",
                "Maxnet" = "Maxent", #maxnet is open-access Maxent
                "favorability_model" = "favorability_function",
                "Bayesian_inference" = "hierarchical_model",
                "GLMM" = "hierarchical_model",
                "GAMM" = "hierarchical_model",
                "n-dimensional_hypervolume" = "hierarchical_model",
                "regression_tree" = "BRT",
                "bagged_decision_tree" = "BRT",
                "recursive_partitioning" = "CART",
```

```
                "TreeNet" = "GBM",
                "linear_regression" = "GLM",
                "multiplicative_regression"= "GLM",
                "probit_regression"= "GLM",
                "Mahalnobis_distance" = "Mahalanobis_distance",
                "Mahalanobis_Distance" = "Mahalanobis_distance",
                "Maxlike" = "MaxLike",
                "multiplicative_regression" = "GLM",
                "Domain" = "DOMAIN",
                "Gower_distance" = "DOMAIN",
                "BIOCLIM" = "SRE",
                "SER" = "SRE",
                "occupancy_model" = "Occupancy_model",
                "hierarchical_model" = "Hierarchical_model",
                "favorability_function" = "Favorability_function",
                "Favorability_function" = "Favorability",
                "logistic_regression" = "Logistic_regression",
                "Logistic_regression" = "GLM",
                "RSF" = "GLM",
                "CART" = "CTA",
                "Occupancy_model" = "Hierarchical_model",
                "MaxLike" = "Hierarchical_model",
                "Hierarchical_model" = "Hierarchical",
                "MDA" = "DA",
                "FDA" = "DA",
                "Penrose_distance" = "Mahalanobis_distance",
                "Mahalanobis_distance" = "Mahalanobis"
                )

# for-loop of edits
  for (pattern in names(patterns)) {
    sdm.df <- data.frame(lapply(sdm.df, function(x) {
      gsub(pattern, patterns[pattern], x)
    }))
  }

# remove underscores
  sdm.df <- data.frame(lapply(sdm.df, function(x) {gsub("_"," ", x)}))

# convert blank to NA
  sdm.df$SDM_algorithm_ensembles[sdm.df$SDM_algorithm_ensembles==''] <- NA

# remove duplicates
  sdm.df <- sdm.df[-duplicated(sdm.df),]

# get new counts
  length(unique(sdm.df$SDM_algorithm));
```

```
## [1] 18
```

```
  length(unique(sdm.df$SDM_algorithm_ensembles))
```

```
## [1] 18
```

Get summaries

```
# summary of SDM algorithms
  summary(as.factor(sdm.df$SDM_algorithm))
```

```
##          ANN          BRT          CTA           DA         ENFA     ensemble
##            4           55            7            1           39         1179
## Favorability          GAM         GARP          GBM          GLM Hierarchical
##            8           38           12            5          251           77
##  Mahalanobis         MARS       Maxent     multiple           RF          SVM
##           14            3          580          281           39            2
```

```
# summary of SDM ensembles
  summary(as.factor(sdm.df$SDM_algorithm_ensembles))
```

```
##          ANN          BRT          CTA           DA       DOMAIN         ENFA
##           94           57           86           75           11           21
##          GAM         GARP          GBM          GLM Hierarchical  Mahalanobis
##          134           21          142          216           10           16
##         MARS       Maxent           RF          SRE          SVM         NA's
##          112          194          186           61           24         1135
```

12.1 Summary table of SDM algorithm use

First, make a new table with a column for all algorithms

```
# make subsets
  sdm.single.df <- subset(sdm.df,
                          select = c("uid","SDM_algorithm"))
  sdm.ensemb.df <- subset(sdm.df,
                          select = c("uid","SDM_algorithm_ensembles"))

# rename columns
  colnames(sdm.single.df)[2] <- "SDM"
  colnames(sdm.ensemb.df)[2] <- "SDM"

# remove NA's, multiples, and ensembles
  sdm.single.df <- sdm.single.df[!is.na(sdm.single.df$SDM),]
  sdm.single.df <- sdm.single.df[!sdm.single.df$SDM=='multiple',]
  sdm.single.df <- sdm.single.df[!sdm.single.df$SDM=='ensemble',]
  sdm.ensemb.df <- sdm.ensemb.df[!is.na(sdm.ensemb.df$SDM),]

# rbind
  sdm.all.df <- rbind(sdm.single.df,sdm.ensemb.df)

# remove duplicates
  sdm.all.df <- sdm.all.df[-duplicated(sdm.all.df),]

# manually add one row that is missing
  uid18.df <- sdm.single.df[sdm.single.df$uid==18,]
  sdm.all.df <- rbind(uid18.df,sdm.all.df)
```

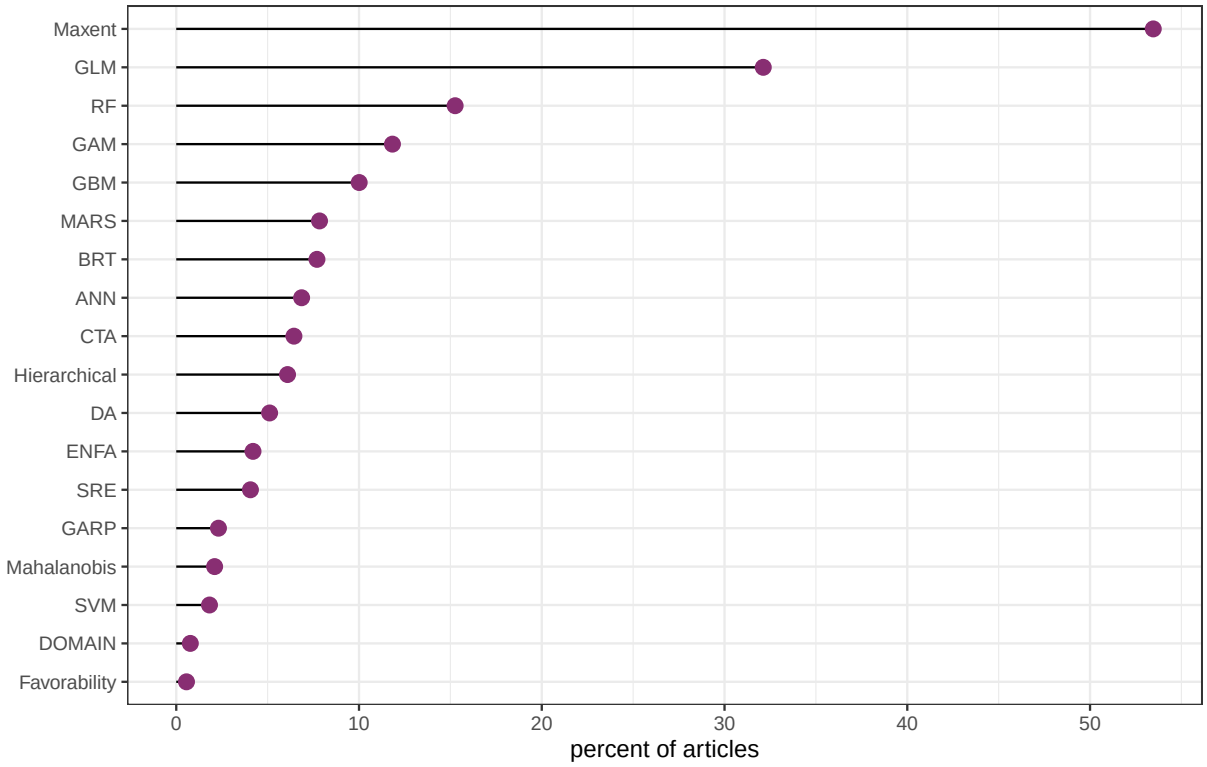
```r
# ensure number of articles matches accepted articles
  #anti_join(yes.df,sdm.all.df,by='uid')
  length(unique(sdm.all.df$uid)) == length(unique(yes.df$uid))
```

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

12.2 Plot of SDM algorithm use

Next, get sum of papers per algorithm and visualize as a bar plot (% of articles)

```r
# get summary
  sdm.sums <- ddply(sdm.all.df, .(SDM),
                    summarize,
                    # number of articles
                    n_papers = length(unique(uid)),
                    # percent of articles
                    perc_papers = round(length(unique(uid))/length(unique(yes.df$uid)),4)
                    )

# make lollipop plot, ranked from most- to least- used SDM
  sdm.fig <- sdm.sums %>%
                arrange(perc_papers) %>%    # sort by value
                mutate(SDM=factor(SDM, levels=SDM)) %>%   # update the factor levels
                ggplot(aes(x=SDM, y=perc_papers*100)) +
                  geom_segment(aes(xend=SDM, yend=0)) +
                  geom_point(size=3, color="#882E72") +
                  ylim(0,50) +
                  scale_y_continuous(breaks = seq(0,50,10)) +
                  coord_flip() +
                  theme_bw() +
                  xlab("")+ ylab("percent of articles")

# save
  ggsave(plot=sdm.fig, filename = paste0(image.dir,'SDM_algorithm_use.png'),
         height = 4, width = 7, units = 'in', dpi = 600)
  ggsave(plot=sdm.fig, filename = paste0(image.dir,'SDM_algorithm_use.svg'),
         height = 4, width = 7, units = 'in')

# view here
  sdm.fig
```



13 Final dataset for export

Here, we prepare the final dataset for the article's appendix. We'll be using the following items for compiling:

- rev.df (list of all articles)
- yes.df (list of accepted articles)
- preds.list.export (list of predictors, including edited domain, taxa, and study focus names)
- study.area.all (table of edited study area names)
- amb.df (table of counts of ambiguous predictors)

First, we will work with all the relevant articles, then we will add the semi-relevant and not relevant ones to the list.

```
# extract columns of interest from yes.df
  final.df.yes <- subset(yes.df,
                         select=c('uid','year','title','author','journal',
                                  'relevant',
                                  'time','hum_time','time_start','time_end',
                                  'future_time_start','future_time_end',
                                  'ttl_species',
                                  'num_env_preds','num_hum_preds',
                                  'worldclim','qual_eval'
                                  ))
  final.df.yes$uid <- as.character(final.df.yes$uid)

# collapse predictor list items
```

```
  final.df.vars <-  ddply(prdotf.list.long, .(uid,study_focus),
                          summarize,
                          # list of domains studied
                          domain=paste(unique(domain),collapse="; "),
                          # list of taxa studied
                          taxa=paste(unique(taxa),collapse="; "),
                          # list of predictors used
                          hum_preds=paste(unique(predictor),collapse="; "),
                          # list of data types used
                          hum_pred_type=paste(unique(study_focus),collapse="; "),
                          # list of data categories
                          hum_pred_cat=paste(unique(category),collapse="; "))
  final.df.vars$uid <- as.character(final.df.vars$uid)

# collapse study area names
  final.df.areas <-  ddply(study.area.all, .(uid),
                           summarize,
                           # list of scales
                           study_area_scale=paste(unique(study_area_scale),collapse="; "),
                           # list of countries
                           study_area_country=paste(unique(region),collapse="; ")
                           )
  final.df.areas$uid <- as.character(final.df.areas$uid)

# collapse SDM algorithm names
  final.df.sdms <-  ddply(sdm.df, .(uid),
                          summarize,
                          # list of scales
                          SDM_algorithm=paste(unique(SDM_algorithm),collapse="; "),
                          # list of countries
                          SDM_algorithm_ensembles=paste(unique(SDM_algorithm_ensembles),
                                                        collapse="; ")
                         )
  final.df.sdms$uid <- as.character(final.df.sdms$uid)

# count of ambiguous predictors and indicator column
  final.df.amb <- data.frame(uid=amb.df$uid,amb_pred='yes',ttl_amb_pred=amb.df$amb_count)
  final.df.amb <- final.df.amb[!duplicated(final.df.amb$uid),]
  final.df.amb$uid <- as.character(final.df.amb$uid)

# left joins
  final.df <- left_join(final.df.yes,final.df.areas, by='uid')
  final.df <- left_join(final.df,final.df.vars, by='uid')
  final.df <- left_join(final.df,final.df.amb, by='uid')
  final.df <- left_join(final.df,final.df.sdms, by='uid')

# edit empty ambiguous field
  final.df$amb_pred[is.na(final.df$amb_pred)] <- 'no'

# remove duplicated rows
  final.df <- final.df[!duplicated(final.df),]

# preview dataframe
```

```
head(final.df)
```

```
##   uid year
## 1   2 2021
## 2  18 2021
## 3  25 2021
## 4  27 2021
## 5  45 2021
## 6  53 2021
##
## 1                                                    'THE BEST OF TWO WORLDS'-COMBINING CLASSIFIER FUSIO
## 2 A DISTRIBUTION MODEL FOR GLOSSINA BREVIPALPIS AND GLOSSINA AUSTENI IN SOUTHERN MOZAMBIQUE, ESWATIN
## 3                                                                                        A HIERARCHICAL F
## 4                                                                                A HYBRID CORRELATIVE-MECHA
## 5                                                                                      A PRAGMATI
## 6                                                                  A SEASCAPE APPROACH FOR GUIDING EFFECTI
##
## 1                                                                                                  Mouta
## 2 de Beer, Chantel J.;Dicko, Ahmadou H.;Ntshangase, Jerome;Moyaba, Percy;Taioe, Moeti O.;Mulandane,
## 3
## 4  McClure, Meredith L.;Haase, Catherine G.;Hranac, Carter Reed;Hayman, David T. S.;Dickson, Brett G
## 5
## 6
##                              journal relevant         time     hum_time
## 1                       REMOTE SENSING      yes present-only present-only
## 2    PLOS NEGLECTED TROPICAL DISEASES      yes present-only present-only
## 3               ECOLOGICAL MODELLING      yes present-only present-only
## 4             JOURNAL OF BIOGEOGRAPHY      yes present-only present-only
## 5      JOURNAL OF WILDLIFE MANAGEMENT      yes present-only present-only
## 6 ELEMENTA-SCIENCE OF THE ANTHROPOCENE      yes present-only present-only
##   time_start time_end future_time_start future_time_end ttl_species
## 1        UNK      UNK              <NA>            <NA>           1
## 2       2009     2019              <NA>            <NA>           2
## 3       1950     2020              <NA>            <NA>          25
## 4       1948     2021              <NA>            <NA>           5
## 5       2001     2012              <NA>            <NA>           1
## 6       2015     2015              <NA>            <NA>           1
##   num_env_preds num_hum_preds worldclim qual_eval study_area_scale
## 1            14            6        no      <NA>            local
## 2            23            3        no      <NA>    multinational
## 3             9            3       yes      <NA>            local
## 4            11            3        no      <NA>    multinational
## 5             3            2        no      <NA>            local
## 6             7            1        no      <NA>         regional
##                   study_area_country                study_focus      domain
## 1                           Portugal                  invasions terrestrial
## 2 South Africa; Mozambique; Eswatini        human health/safety terrestrial
## 3                            Germany             food/economics terrestrial
## 4              United States; Canada               conservation terrestrial
## 5                      United States                exploratory  freshwater
## 6                             Canada reintroduction/restoration      marine
##           taxa
## 1  trees/shrubs
```

```
## 2 invertebrates
## 3 invertebrates
## 4       mammals
## 5       mammals
## 6 invertebrates
##
## 1 artificial urban areas percent; crops annual percent; highways length total; production forest euca
## 2
## 3
## 4
## 5
## 6
##                hum_pred_type
## 1                  invasions
## 2         human health/safety
## 3              food/economics
## 4                conservation
## 5                 exploratory
## 6 reintroduction/restoration
##                                                                  hum_pred_cat
## 1 infrastructure; food/agriculture; transportation; energy/raw materials
## 2             socio-economic; food/agriculture; management/interventions
## 3                                        infrastructure; food/agriculture
## 4                                         energy/raw materials; pollution
## 5                                        food/agriculture; transportation
## 6                                                management/interventions
##   amb_pred ttl_amb_pred SDM_algorithm            SDM_algorithm_ensembles
## 1       no           NA      multiple RF; CTA; GLM; DA; ANN; Maxent; GAM
## 2       no           NA        Maxent                                 NA
## 3       no           NA      multiple                       GLM; RF; DA
## 4       no           NA           BRT                                 NA
## 5       no           NA  Hierarchical                                 NA
## 6       no           NA  Hierarchical                                 NA
```

Next are the semi-relevant and non-relevant articles to append as new rows.

```
# semi-relevant articles
  final.df.semi <- rev.df[rev.df$relevant=='semi',]
  final.df.semi <- subset(final.df.semi,
                          select=c('uid','year','title','author','journal','relevant'))

# non-relevant articles
  final.df.no <- rev.df[rev.df$relevant=='no',]
  final.df.no <- subset(final.df.no,
                          select=c('uid','year','title','author','journal','relevant'))

# unknown articles
  final.df.unk <- rev.df[rev.df$relevant=='UNK',]
  final.df.unk <- subset(final.df.unk,
                          select=c('uid','year','title','author','journal','relevant'))

# bind rows
  final.df$uid <- as.integer(final.df$uid)
  final.df <- bind_rows(final.df,final.df.semi, final.df.no,final.df.unk)
```

Double-check the number of articles, confirming that they match original input:

```
# check number of articles to match original
  length(unique(final.df$uid)) == length(unique(rev.df$uid))
```

```
## [1] FALSE
```

Next, we re-organize the columns and rename a few of them.

```
# relocate
# use colname = newcolname notation for renaming
  final.df <- relocate(final.df,
                       study_focus, study_area_scale, study_area_country,
                       .before = time)
  final.df <- relocate(final.df,
                       taxa,
                       .before = ttl_species)
  final.df <- relocate(final.df,
                       domain = domain,SDM_algorithm,SDM_algorithm_ensembles,
                       .after = ttl_species)
  final.df <- relocate(final.df,
                       hum_preds, hum_pred_type, hum_pred_cat,
                       hum_amb_preds = amb_pred,
                       num_env_preds, num_hum_preds,
                       num_amb_preds = ttl_amb_pred,
                       hum_pred_cat,
                       .after = SDM_algorithm_ensembles)

  colnames(final.df)
```

```
##  [1] "uid"                    "year"
##  [3] "title"                  "author"
##  [5] "journal"                "relevant"
##  [7] "study_focus"            "study_area_scale"
##  [9] "study_area_country"     "time"
## [11] "hum_time"               "time_start"
## [13] "time_end"               "future_time_start"
## [15] "future_time_end"        "taxa"
## [17] "ttl_species"            "domain"
## [19] "SDM_algorithm"          "SDM_algorithm_ensembles"
## [21] "hum_preds"              "hum_pred_type"
## [23] "hum_pred_cat"           "hum_amb_preds"
## [25] "num_env_preds"          "num_hum_preds"
## [27] "num_amb_preds"          "worldclim"
## [29] "qual_eval"
```

Change all blank fields to NA.

```
# change NAs to blank
  final.df[final.df == ''] <- NA
  final.df$SDM_algorithm_ensembles[final.df$SDM_algorithm_ensembles == 'NA'] <- NA
```

Save the table as a CSV.

```
# save
  write.csv(final.df,paste0(data.dir,"lit_review_dataframe_FINAL.csv"),
            row.names = FALSE)
```

14 Save

```
# save progress
  save.image("SDMs_human_lit_review_IV.RData")
```

THIS IS THE END OF THE SCRIPT.

See "Human Influence in SDMs: Literature Review (Part V)" for next steps.
