CURE: Florida Plants and Climate Change

ML Gaynor

Spring 2020

Contents

Course information

In Spring 2020, Johanna Jantzen and I taught Course-Based Undergraduate Research (CURE) sections with Drs. Pam and Doug Soltis. The material that follows was developed for this course.

Set-up

Install packages needed for these scripts. Versions used are indicated when packages are loaded. Note: Sourcing function .R scripts will Functions packages.

```
install.packages("devtools")
install.packages("tidyverse")
# installs ggplot2, dplyr, tidyr, readr, purr, tibble, stringr, and forcats
install.packages(c("plyr", "spocc",
                   "ridigbio"))
install.packages(c("lubridate", "CoordinateCleaner",
                   "raster", "spatstat"))
install.packages(c("rgdal", "gtools",
                   "alphahull", "rgeos",
                   "sp"))
install.packages(c("caret", "factoextra",
                   "FactoMineR", "agricolae"))

install.packages("RColorBrewer")
install.packages("ENMeval")


library(devtools)
install_github("vqv/ggbiplot")
install_github("danlwarren/ENMTools")
```

CURE Project

According to the Main and Lohrer, 2018 there are 28 plant species residing in Archbold Biological Station listed as Endangered or Threatened. Of these 28 species, 25 are classified as Angiosperms. Each student was assigned 5 species to learn about and focus their independent projects on.

R script	Description
01_OccurrenceDataDownload.R	Download specimen records for a list of synonymns
02_OccurrenceDataCleaningStep1.R	Identify specimen needing to be georeferenced or requested directly from the collection
03_OccurrenceDataCleaningStep2.R	Combined raw, georeferenced, and requested specimen records. Clean occurrence data
04_ClimateLayers.R	Trim climatic layer for training and for projection
05_ClimaticNiche.R	Investigate climatic niche
06_EcologicalNicheModeling.R	Examine ENM generated in Maxent
07_ENMProcessing.R	Niche breath, niche overlap, and geographic overlap

01-OccurrenceDataDownload.R

Downloading occurrence data for CURE plants. ML Gaynor

I really wanted to download data from 'idigbio', 'gbif', and 'bison' and keep specific columns:

Column	Description
name	scientific name
basis	basis of record
date	event data
institutionID	institution ID
collectionCode	collection code
collectionID	collection ID
country	country
county	county
state	stateprovince
locality	locality or verbartim locality
Latitude	decimal latitude
Longitude	decimal longitude
ID	idigbio = uuid, gbif = key, bison = occurrenceID
coordinateUncertaintyInMeters	coordinate uncertainty in meters
habitat	habitat
prov	indicates who provided the data: gbif, bison, or idigbio
spocc.latitude	
spocc.longitude	

Load source

```
source("functions/DownloadingDataMore.R")
library(ggplot2)
```

Make synonym lists

Here I only show 5 species as examples, however synonymn list were collected for all 25 species.

```
Warea_carteri <- c("Warea carteri")
Lechea_cernua <- c("Lechea cernua")
Hypericum_cumulicola <- c("Hypericum cumulicola",
                          "Sanidophyllum cumulicola")
Hypericum_edisonianum <- c("Hypericum edisonianum",
                           "Ascyrum edisonianum")
Prunus_geniculata <- c("Prunus geniculata")
```

Download occurrence data

Note file management

```
spocc_combine(Warea_carteri,
              "data/raw/Warea_carteri_raw_021620_Andrade.csv")
spocc_combine(Lechea_cernua,
```

```
              "data/raw/Lechea_cernua_raw_021620_Andrade.csv" )
spocc_combine(Hypericum_cumulicola,
              "data/raw/Hypericum_cumulicola_raw_021620_Andrade.csv" )
spocc_combine(Hypericum_edisonianum,
              "data/raw/Hypericum_edisonianum_raw_021620_Andrade.csv")
spocc_combine(Prunus_geniculata,
              "data/raw/Prunus_geniculata_raw_021620_Andrade.csv" )
```

What records did we obtain?

Read in all records

```
list_of_files <- list.files("data/raw/", pattern = "*.csv",
                            full.names = TRUE, recursive = FALSE)
readfiles <- c()
for (i in 1:length(list_of_files)){
  readfiles[[i]] <- read.csv(list_of_files[i])
}
occurrence_data <- bind_rows(readfiles)
```

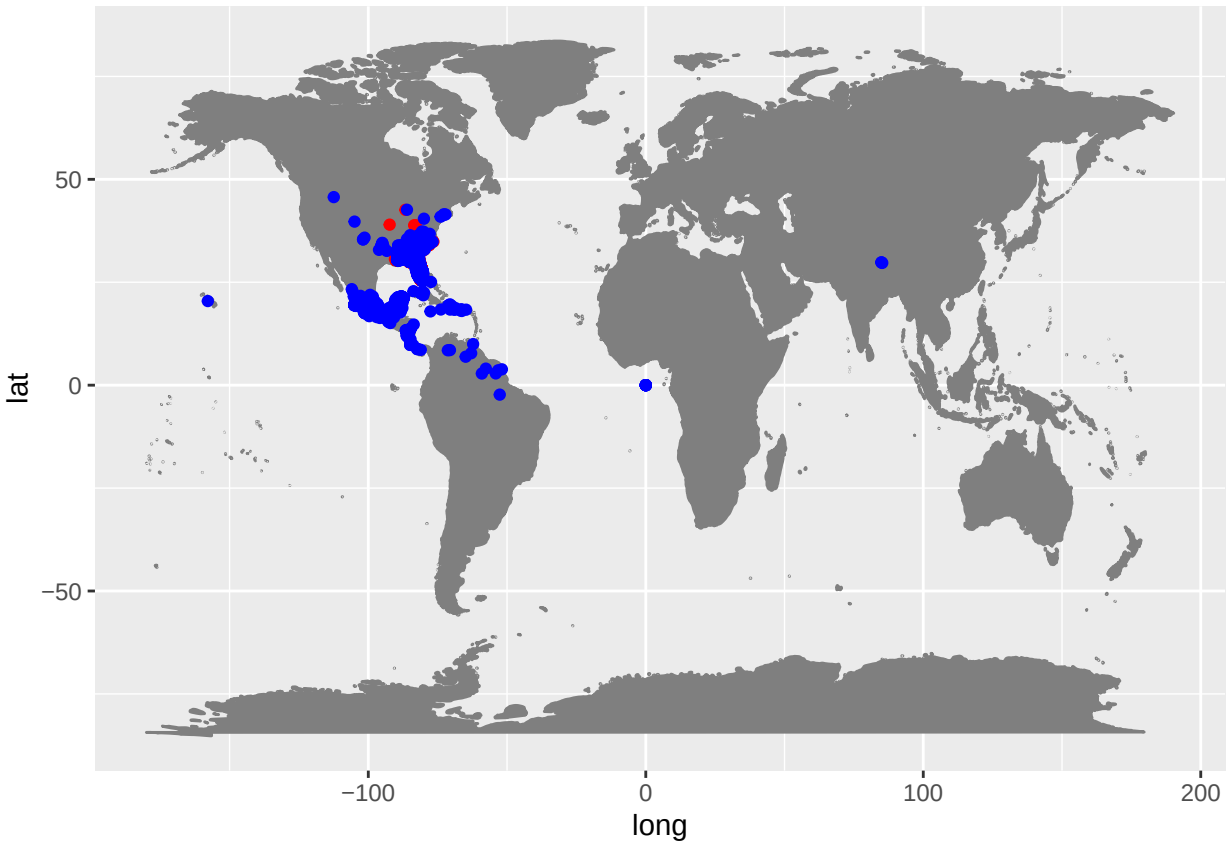
Count number of specimen obtained

```
dflist <- c()
rawcounts <- c()
for (i in 1:25){
  df <- readfiles[[i]]
  dflist[i] <- as.character(df$name[1])
  rawcounts[i] <- nrow(df)
}
count <- data_frame(species = dflist, raw_count = rawcounts)
count
```

species	raw_count
asclepias curtissii	394
calamintha ashei	236
dicerandra frutescens	456
encyclia tampensis	1068
eriogonum floridanum	376
eryngium cuneifolium	202
garberia heterophylla	833
hartwrightia floridana	499
hypericum cumulicola	500
hypericum edisonianum	322
lechea cernua	893
liatris ohlingerae	445
lilium catesbaei	1514
nolina brittoniana	600
coleataenia abscissa	280
paronychia chartacea	843
pinguicula lutea	1584
platanthera ciliaris	2440
polygonella basiramia	793
polygonella myriophylla	585
prunus geniculata	648
tillandsia dressleri	1206
tillandsia fasciculata	1480
tillandsia utriculata subsp. utriculata	761
warea carteri	328

Plot raw records

```
mapWorld <- borders("world", colour="gray50", fill="gray50")
messy_map <- ggplot()+
               mapWorld+
               geom_point(occurrence_data,
                          mapping = aes(x = spocc.longitude,
                                        y = spocc.latitude),
                          col = "red") +
               geom_point(occurrence_data,
                          mapping = aes(x = Longitude,
                                        y = Latitude),
                          col = "blue")
messy_map
```


50
0
−50
lat
−100
0
100
200
long

02_OccurrenceDataCleaningStep1.R

Identify raw records that need to be requested from collection managers or those that need to be georeferenced.

Load source

```
library(dplyr)
source("functions/MoreCleaningFunctions.R")
```

Create starting dataframes

Here we are using *readraw* which is a function to read in the raw occurrence records.

```
list <- list.files("data/raw/", pattern = "*Andrade.csv", full.names = TRUE)
clist <- lapply(list, readraw)
cdf <- bind_rows(clist)
```

Identify flagged specimen

needed_records is a function I wrote to identify records with flags.

```
cneeded <- needed_records(cdf)
```

Fix synonymns

First, make list of each synoymn.

```
Warea_carteri <- c("warea carteri")
Lechea_cernua <- c("lechea cernua")
Hypericum_cumulicola <- c("hypericum cumulicola", "sanidophyllum cumulicola")
Hypericum_edisonianum <- c("hypericum edisonianum", "ascyrum edisonianum")
Prunus_geniculata <- c("prunus geniculata")
```

Then, use ifelse statements to modify the names

```
cneeded_mod <- cneeded %>%
  mutate(new_name = ifelse(scientificname %in% Warea_carteri, "Warea carteri",
                           ifelse(scientificname %in% Lechea_cernua, "Lechea cernua",
                                  ifelse(scientificname %in% Hypericum_cumulicola, "Hypericum cumulicol
                                         ifelse(scientificname %in% Hypericum_edisonianum, "Hypericum e
                                                ifelse(scientificname %in% Prunus_geniculata, "Prunus g
                                  ))))
cneeded_count <- cneeded_mod %>%
                 group_by(new_name, `data.dwc:institutionCode`) %>%
                 dplyr::summarize(count = n())
cneeded_count
```

new_name	data.dwc:institutionCode	count
Hypericum cumulicola	ARCH	11
Hypericum cumulicola	CLEMS	1
Hypericum cumulicola	CM	1
Hypericum cumulicola	FLAS	56
Hypericum cumulicola	GA	5
Hypericum cumulicola	IND	1
Hypericum cumulicola	NCSC	1
Hypericum cumulicola	NCU	10
Hypericum edisonianum	ARCH	12
Hypericum edisonianum	FLAS	21
Hypericum edisonianum	GA	6
Hypericum edisonianum	IND	4
Hypericum edisonianum	NCSC	2
Hypericum edisonianum	NCU	3
Hypericum edisonianum	SEL	2
Lechea cernua	ARCH	5
Lechea cernua	FLAS	56
Lechea cernua	SEL	1
Prunus geniculata	ARCH	6
Prunus geniculata	FLAS	88
Prunus geniculata	GA	13
Prunus geniculata	ID	1
Prunus geniculata	LSU	6
Prunus geniculata	NCU	3
Warea carteri	ARCH	10
Warea carteri	FLAS	27
Warea carteri	GA	4
Warea carteri	LSU	1
Warea carteri	NCU	5

What herbariums do I need to contact?

1. A word document that contains:

- Each herbarium name and Collections Manager contact information (name, email, position title). Find on the NYBG herbarium index website.
- Number of specimen records you are needing and for which species (SpeciesA 20, SpeciesB 3)

2. A **single csv file for each herbarium** containing all records you are requesting - **DO NOT send me a csv for each species**. Column names should match the information_needed colnames above. You can contruct this manually or in R.

Quick way to make the csv for each species

```
inames <- unique(cneeded_count$`data.dwc:institutionCode`)
path <- "data/raw/request/"
end <- "_Andrade.csv"
for (i in 1:length(inames)){
  code <- inames[i]
  need <- cneeded_mod %>%
          filter(`data.dwc:institutionCode` == code)
  file_name1 <- paste(path, code, sep = "")
```

```
  file_name2 <- paste(file_name1, end, sep = "")
  write.csv(need, file_name2)
}
```

Create dataframe for georeferencing

Anything with the locality string avaliable!

```
for_georeferencing <- need_to_georeference(cdf)
```

```
for_georeferencing_mod <- for_georeferencing %>%
  mutate(new_name = ifelse(scientificname %in% Warea_carteri, "Warea carteri",
                           ifelse(scientificname %in% Lechea_cernua, "Lechea cernua",
                                  ifelse(scientificname %in% Hypericum_cumulicola, "Hypericum cumulicol
                                         ifelse(scientificname %in% Hypericum_edisonianum, "Hypericum e
                                                ifelse(scientificname %in% Prunus_geniculata, "Prunus g
                                  ))))
```

03_OccurrenceDataCleaningStep2.R

Load Packages

```
library(dplyr)
library(ggplot2)
library(ridigbio)
library(lubridate)
library(CoordinateCleaner)
library(raster)
library(spatstat)
source("functions/MoreCleaningFunctions.R")
```

Create starting data file

Load raw data

Change the path and pattern for your data.

```
rawfiles <- list.files(path = "data/raw",
                       pattern = "*Andrade.csv", full.names = TRUE)
raw_records <- lapply(rawfiles, readraw)
```

Load georeferenced data

```
geofiles <- list.files(path = "data/raw/georeference/Andrade_georeferenced/",
                       pattern = "*.csv", full.names = TRUE)
geo_records <- lapply(geofiles, readgeoref)
```

Make sure the order of the raw data files and the georeference files match - if you named these correctly they should be in the same order

Combinded raw and georeferenced data

```
# Make empty list to store the output of the four loop
combined <- c()
for (i in 1:5){
  ## Subset the ith dataframe in raw_records
  occurrence_records <- raw_records[[i]]
  ## Subset the ith dataframe in geo_records
  georeference_records <- geo_records[[i]]
  georeference_records <- matchColClasses(occurrence_records, georeference_records)
  occurrence_records <- left_join(occurrence_records, georeference_records)
  combined[[i]] <- occurrence_records
}
```

Load recieved data

This data is from FLAS (UF's herbarium) - Use of FLAS data in any product should be credited with "University of Florida Herbarium (FLAS), Florida Museum of Natural History". We will also need to provide FLAS with information regarding how these data are used (publication title, etc.). Exact localities of endangered species are not supposed to be published (they can be included in analysis for publication, but

the lat/long should not be relased).
More on those data are on this page: https://www.floridamuseum.ufl.edu/herbarium/cat/tech/exportinfo.htm

Export FLAS data for georeferencing

Load full FLAS file

```
FLASfiles <- list.files(path = "data/raw/request/recieved/Andrade/",
                        pattern = "*.csv", full.names = TRUE)
FLAS_records <- lapply(FLASfiles, readflas)
```

Export data for georeferencing

This includes all observations, even those with Lat/Long.

```
## Create list of names
FLASfilesB <- list.files(path = "data/raw/request/recieved/Andrade/",
                        pattern = "*.csv", full.names = FALSE)
name_FLAS <- gsub(".csv", "", FLASfilesB)

## Set path and file-ending to create file name for write.csv
path <- "data/raw/georeference/recieved/"
end <- "_georeference.csv"

## for loop
for (i in 1:5){
      name <- name_FLAS[i]
      df <- FLAS_records[[i]]
      df_ready <- flas_needed(FLAS_df = df)
      part <- paste(path, name, sep="")
      full <- paste(part, end, sep = "")
      write.csv(df_ready, full, row.names = FALSE)
}
```

After georeferencing FLAS

Load georeferenced recieved files

```
FLASfiles <- list.files(path = "data/raw/georeference/recieved/",
                        pattern = "*.csv", full.names = TRUE)
FLAS_records <- lapply(FLASfiles, readflasfiltered)
```

Join georeferenced + raw with FLAS

```
combined_final <- c()
for (i in 1:5){
  combined_records <- combined[[i]]
  FLAS_records_loop <- FLAS_records[[i]]
  FLAS_records_loop <- matchColClasses(combined_records, FLAS_records_loop)
  combined_records <- full_join(combined_records, FLAS_records_loop)
  combined_final[[i]] <- combined_records
}
```

Seperate species

```
Hypericum_cumulicola_df <- combined_final[[1]]
Hypericum_edisonianum_df <- combined_final[[2]]
Lechea_cernua_df <- combined_final[[3]]
Prunus_geniculata_df <- combined_final[[4]]
Warea_carteri_df <- combined_final[[5]]
```

Repeat the following steps for each species

Inspect the dataframe

```
print(head(Hypericum_cumulicola_df))
```

```
##                    name             basis       date
## 1 hypericum cumulicola PreservedSpecimen       <NA>
## 2 hypericum cumulicola PreservedSpecimen       <NA>
## 3 hypericum cumulicola PreservedSpecimen 1986-09-30
## 4 hypericum cumulicola PreservedSpecimen 1947-10-12
## 5 hypericum cumulicola PreservedSpecimen       <NA>
## 6 hypericum cumulicola PreservedSpecimen       <NA>
##                     institutionID collectionCode
## 1                            <NA>           <NA>
## 2                            <NA>           <NA>
## 3 university of florida herbarium           <NA>
## 4   urn:lsid:biocol.org:col:15638           herb
## 5                            <NA>           <NA>
## 6                            <NA>           <NA>
##                           collectionID       country           county   state
## 1 17f2d0fa-39a6-4465-8055-1d6fc12eeda2 united states          de soto florida
## 2 565b6f19-288f-4614-a4c9-b09448e96547 united states        highlands florida
## 3                                 <NA> united states        highlands florida
## 4                                 <NA> united states    highlands co. florida
## 5 274b5332-1247-4374-b124-c819b814cd6e united states        highlands florida
## 6        urn:lsid:biocol.org:col:15610 united states highlands county florida
##   locality Latitude Longitude                                   ID
## 1     <NA>       NA        NA 017841eb-8995-449a-bd9c-791e13e1d509
## 2     <NA>       NA        NA 01c11192-5b71-4b52-b725-692fd75e0e7a
## 3     <NA>       NA        NA 01ee2556-3b1b-4c91-b7b9-cc7f87c5b45b
## 4     <NA>       NA        NA 032ae393-5378-4a42-be48-cf1425afbada
## 5     <NA>       NA        NA 035f0baa-dde9-4adb-9726-8aa1b28c2ba6
## 6     <NA>       NA        NA 03877043-426d-4a69-914e-deb46021219b
##   coordinateUncertaintyInMeters
## 1                          <NA>
## 2                          <NA>
## 3                          <NA>
## 4                          <NA>
## 5                          <NA>
## 6                          <NA>
##                                                                                                infor
## 1 field values redacted: eventDate, month, day, startDayOfYear, verbatimEventDate, recordNumber, loca
## 2
## 3                                                            location information not given for enda
```

```
## 4
## 5
## 6                                                                                  field values reda
##                                                                                                 habi
## 1                                                                                                  <N
## 2                                                                                                  <N
## 3                                                                                                  <N
## 4 Bare white sand of sand scrub in association with Ceratiola and Dentoceras [Polygonella myriophylla
## 5                                                                                                  <N
## 6                                                                                                  <N
##      prov spocc.latitude spocc.longitude spocc.prov spocc.date
## 1 idigbio             NA              NA    idigbio 1920-03-17
## 2 idigbio             NA              NA    idigbio       <NA>
## 3 idigbio             NA              NA    idigbio 1986-09-30
## 4 idigbio             NA              NA    idigbio 1947-10-12
## 5 idigbio             NA              NA    idigbio       <NA>
## 6 idigbio             NA              NA    idigbio       <NA>
##             spocc.name uuid data.dwc.catalogNumber data.dwc.institutionCode
## 1 hypericum cumulicola <NA>                   <NA>                     <NA>
## 2 hypericum cumulicola <NA>                   <NA>                     <NA>
## 3 hypericum cumulicola <NA>                   <NA>                     <NA>
## 4 hypericum cumulicola <NA>                   <NA>                     <NA>
## 5 hypericum cumulicola <NA>                   <NA>                     <NA>
## 6 hypericum cumulicola <NA>                   <NA>                     <NA>
##   data
## 1 <NA>
## 2 <NA>
## 3 <NA>
## 4 <NA>
## 5 <NA>
## 6 <NA>
```

Fix taxon name

Check name column to ensure all namees are referring to your species.

1. What names are included in your species dataframe?

```r
unique(Hypericum_cumulicola_df$name)
```

```
## [1] "hypericum cumulicola"
## [2] "sanidophyllum cumulicola"
## [3] "Hypericum cumulicola (Small) P.Adams"
## [4] "Sanidophyllum cumulicola Small"
## [5] "Hypericum cumulicola"
```

2. Were each of these names listed on your synoymn list?

Create list of synonyms or accepted names

Include any string on the list above that you would accept as

```r
Hypericum_cumulicola <- c("hypericum cumulicola", "sanidophyllum cumulicola",
                          "Hypericum cumulicola (Small) P.Adams",
                          "Sanidophyllum cumulicola Small" )
```

Mutate and Filter

```
Hypericum_cumulicola_df <- Hypericum_cumulicola_df %>%
                           mutate(correct_name = ifelse(name
                                                        %in% Hypericum_cumulicola,
                                                        "Hypericum cumulicola", 0)) %>%
                           filter(correct_name != 0)
```

Fix the data formatting

Date

```
unique(Hypericum_cumulicola_df$date)
```

```
##   [1] NA           "1986-09-30" "1947-10-12" "1979-09-29" "1962-01-14"
##   [6] "1920-12-13" "1934-09-04" "1987-11-20" "1987-11-12" "1987-03-27"
##  [11] "1987-03-18" "1986-09-19" "1934-09-05" "1986-05-08" "1987-09-11"
##  [16] "1949-07"    "1987-01-18" "1961-09-30" "1986-07-29" "1987-05-16"
##  [21] "1979-09-30" "1987-12-10" "1987-02-28" "1964-06-08" "1979-07-26"
##  [26] "2001-04-26" "1986-07-01" "1960-10-07" "1987-11-23" "1966-07-27"
##  [31] "1978-10-20" "1987-04-04" "1986-08-06" "1959-07-13" "1978-10-08"
##  [36] "1960-04-25" "1984-09-02" "2001-08-08" "1987-03-02" "1990-11-13"
##  [41] "2005-06-29" "1986-09-24" "1987-03-01" "1960-08-01" "1987-03-04"
##  [46] "1980-11-01" "1979-01-16" "1988-04-25" "1990-11-09" "1986-10-14"
##  [51] "1987-02-16" "1987-03-03" "1986-06-30" "1986-02-18" "1984-07-30"
##  [56] "1981-12-18" "1959-11-28" "1986-03-04" "1959-11-27" "1986-09-29"
##  [61] "1986-09-25" "1961-08-23" "1980-11-14" "1979-03-21" "1979-08-20"
##  [66] "1981-09-05" "2002-11-25" "1987-01-19" "2015-10-28" "2015-11-16"
##  [71] "2015-11-18" "2015-11-17" "2014-04-28" "2014-05-15" "2014-05-14"
##  [76] "2014-11-07" "2012-09-27" "2012-09-19" "2012-10-12" "2012-10-16"
##  [81] "2002-09-10" "1999-01-01" "1998-09-23" "1998-09-17" "1998-09-11"
##  [86] "1998-10-20" "1994-01-23" "1991-05-24" "1991-10-19" "1990-01-01"
##  [91] "1989-02-17" "1987-03-10" "1987-04-30" "1987-04-21" "1987-05-04"
##  [96] "1987-06-04" "1987-08-18" "1987-11-19" "1987-12-09" "1986-08-21"
## [101] "1986-10-16" "1986-10-15" "1983-08-09" "1983-08-23" "1983-09-06"
## [106] "1983-09-09" "1981-09-16" "1981-10-03" "1981-01-01" "1976-07-26"
## [111] "1966-07-10" "1964-04-05" "1961-01-01" "1960-01-01" "1959-01-01"
## [116] "1949-07-01" "1946-01-01" "1934-08-17" "1934-01-01" "1925-05-19"
## [121] "1924-07-17" "1922-08-30" "1922-08-31" "1922-12-24" "1921-04-25"
## [126] "1921-05-23" "1920-04-18" "1920-01-01" "1919-05-01"
```

Date fixing

Here we are using the package lubridate.

```
Hypericum_cumulicola_df$date <- ymd(Hypericum_cumulicola_df$date)
```

```
## Warning: 2 failed to parse.
```

```
Hypericum_cumulicola_df <- Hypericum_cumulicola_df %>%
  dplyr::mutate(
  year = lubridate::year(date),
  month = lubridate::month(date),
  day = lubridate::day(date))
```

Lat/Long cleaning

Before we start cleaning latitude and longitude, how many occurrence records did we have?

```
nrow(Hypericum_cumulicola_df)
```

```
## [1] 500
```

Merge

Mergingthe two latitude columns and the two longitude columns

```
Hypericum_cumulicola_df$Latitude <- coalesce(Hypericum_cumulicola_df$Latitude,
                                             Hypericum_cumulicola_df$spocc.latitude)
Hypericum_cumulicola_df$Longitude <- coalesce(Hypericum_cumulicola_df$Longitude,
                                              Hypericum_cumulicola_df$spocc.longitude)
```

Filter

Filter to only have occurrence records with latitude and longitude.

```
Hypericum_cumulicola_df <- Hypericum_cumulicola_df %>%
                      filter(!is.na(Longitude)) %>%
                      filter(!is.na(Latitude))
nrow(Hypericum_cumulicola_df)
```

```
## [1] 113
```

WOW, look at how many occurrence points we got rid of!

Precision

```
Hypericum_cumulicola_df$Latitude <- round(
  Hypericum_cumulicola_df$Latitude, digits = 2)
Hypericum_cumulicola_df$Longitude <- round(
  Hypericum_cumulicola_df$Longitude, digits = 2)
```

Remove Institution Points

What is different about these two commands? see more: https://ropensci.github.io/CoordinateCleaner/articles/Quickstart_Flagging_problematic_coordinates_in_a_nutshell.html

Also what institutions do they cover? https://ropensci.github.io/CoordinateCleaner/articles/Background_the_institutions_database.html

```
occurrence_records <- cc_inst(Hypericum_cumulicola_df,
                                          lon = "Longitude",
                                          lat = "Latitude",
                                          species = "correct_name")

occurrence_records2 <- clean_coordinates(Hypericum_cumulicola_df,
                                          lon = "Longitude",
                                          lat = "Latitude",
                                          species = "correct_name")
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "/private/var/folders/w6/wvtg_zh17qxb27rnsrz_x0v40000gp/T/Rtmp8yrcTG", layer: "ne_50m_land"
```

```
## with 1420 features
## It has 3 fields
## Integer64 fields read as strings:  scalerank
```

```
nrow(occurrence_records)
```

```
## [1] 113
```

If you find points you want to remove in occurrence_records2:

```
occurrence_records <- occurrence_records2 %>%
                      filter(.summary != FALSE)
nrow(occurrence_records)
```

FOR THIS PROJECT - REMOVE ONLY INST POINTS

```
Hypericum_cumulicola_df <- cc_inst(Hypericum_cumulicola_df,
                                           lon = "Longitude",
                                           lat = "Latitude",
                                           species = "correct_name")
```

```
## Testing biodiversity institutions
```

```
## Removed 0 records.
```

Select columns to continue with:

```
Hypericum_cumulicola_df_ready <- Hypericum_cumulicola_df %>%
                      dplyr::select(name = correct_name,
                             Latitude = Latitude,
                             Longitude = Longitude,
                             ID = ID,
                             basis = basis,
                             coordinateUncertaintyInMeters =
                               coordinateUncertaintyInMeters,
                             year = year,
                             month = month,
                             day = day,
                             prov = prov)
```

Remove duplicate latitude and longitude

```
Hypericum_cumulicola_df_ready <- distinct(Hypericum_cumulicola_df_ready, Latitude, Longitude, .keep_all
nrow(Hypericum_cumulicola_df_ready)
```

```
## [1] 4
```

Only one point per pixel

```
bio1 <- raster("data/climatic_layers/raw/bio1.bil")
rasterResolution <- max(res(bio1))

while(min(nndist(Hypericum_cumulicola_df_ready[,2:3])) < rasterResolution){
  nnD <- nndist(Hypericum_cumulicola_df_ready[,2:3])
  Hypericum_cumulicola_df_ready <- Hypericum_cumulicola_df_ready[-(which(min(nnD) == nnD)[1]),]
  }
nrow(Hypericum_cumulicola_df_ready)
```

```
## [1] 4
```

Write as CSV

```
write.csv(Hypericum_cumulicola_df_ready,
          "data/cleaned_occurrence/full/
          Hypericum_cumulicola_cleaned_031620_Gaynor.csv")
```

Write for maxent

```
Hypericum_cumulicola_df_ready_maxent <- Hypericum_cumulicola_df_ready %>%
                                    dplyr::select(name, Latitude, Longitude)
```

```
write.csv(Hypericum_cumulicola_df_ready_maxent,
          "data/cleaned_occurrence/maxent/
           Hypericum_cumulicola_cleaned_031620_Gaynor.csv")
```

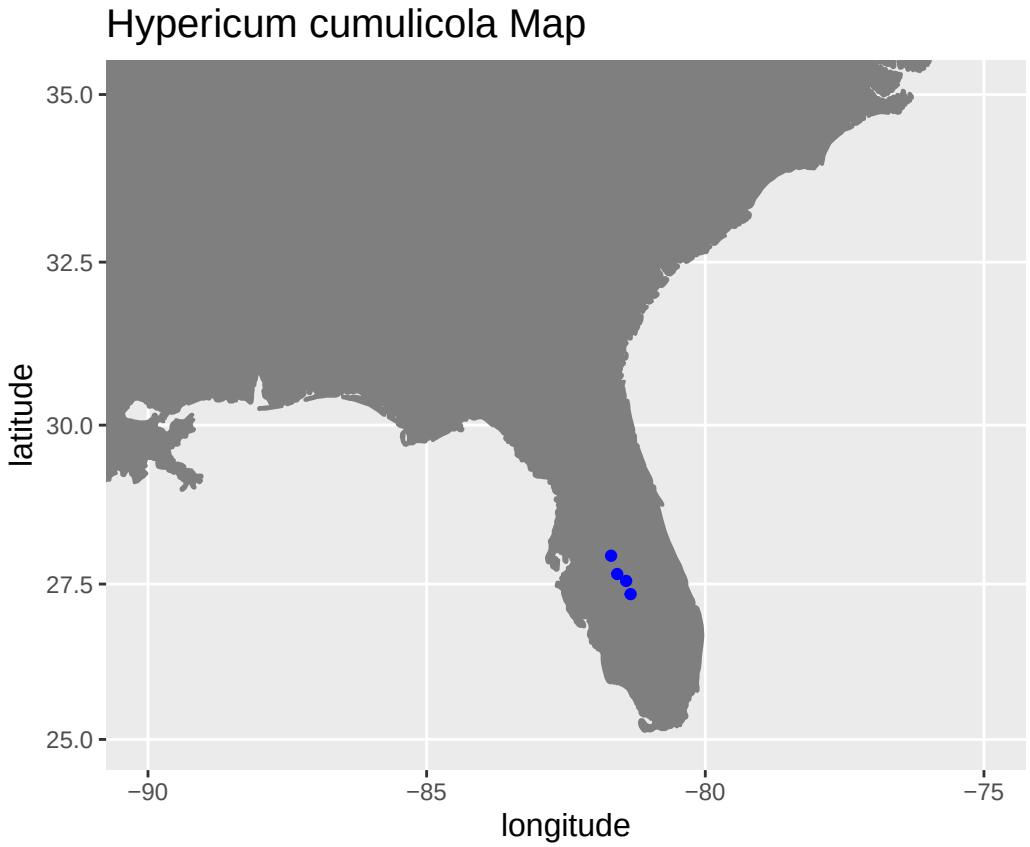
Visualize

Download USA map or World map

```
USA <- borders(database = "usa", colour="gray50", fill="gray50")
#World <- borders(database = "world", colour="gray50", fill="gray50")
```

Plot using ggplot

```
Hypericum_cumulicola_map <- ggplot()+
                USA + # Switch to World if wanting to see the world
                geom_point(Hypericum_cumulicola_df_ready,
                           mapping = aes(x = Longitude, y = Latitude), col = "Blue") +
                coord_map(xlim = c(-90, -75),ylim = c(25, 35))  + # DELETE to see whole US
                theme(axis.title = element_text(size = 12),
                      plot.title = element_text(size = 15),
                      legend.text = element_text(size = 12, face = "italic"),
                      legend.title = element_text(size = 12)) +
                labs(title = "Hypericum cumulicola Map", x = "longitude", y = "latitude")
Hypericum_cumulicola_map
```



Manually inspect your points prior to concluding you can move on to trimming layers.

04_ClimateLayers

Raw layers were downloaded from worldclim We will have multiple sets of **current layers**.

1. **Shared layers**
   - We will use clean occurrence records for all species to set the projection layer.
     - Using all points we will set a convex hull with a buffer and mask/clip bio1 - bio19.
     - We only want to include layers that are not highly correlated. To assess which layers to include, we will look at the **pearson correlation** coefficient among layers.
       * Note: we will not include alt (altitude) in our final dataset because there is not a predicted alt for the future climatic layers.
2. **Species specific layers**
   - We will use clean occurrence records for each species.
     - We will used the cleaned occurrence record for each species and set a convex hull with a buffer and mask/clip the bioclim variables identified as not collinear.

For **future layers** we will only used shared layers - we will not conduct another pearson correlation. We will be using

Load Packages

```
library(rgdal)
```

```
## rgdal: version: 1.4-8, (SVN revision 845)
##  Geospatial Data Abstraction Library extensions to R successfully loaded
##  Loaded GDAL runtime: GDAL 2.4.2, released 2019/06/28
##  Path to GDAL shared files: /Library/Frameworks/R.framework/Versions/3.6/Resources/library/sf/gdal
##  GDAL binary built with GEOS: FALSE
##  Loaded PROJ.4 runtime: Rel. 5.2.0, September 15th, 2018, [PJ_VERSION: 520]
##  Path to PROJ.4 shared files: /Library/Frameworks/R.framework/Versions/3.6/Resources/library/sf/proj
##  Linking to sp version: 1.3-2
```

```
library(raster)
library(gtools)
library(alphahull)
library(rgeos)
```

```
## rgeos version: 0.5-2, (SVN revision 621)
##  GEOS runtime version: 3.7.2-CAPI-1.11.2
##  Linking to sp version: 1.3-1
##  Polygon checking: TRUE
```

```
library(sp)
```

Current projection layers

These are the shared layers for our current projection. These are also the layers we look at to make sure our layers are uncorrelated.

Load Florida shapefile

```
FL <- rgdal::readOGR("data/climatic_layers/FL/FLstate2.shp")
```

Load Rasterlayers

Read more about these layers: https://www.worldclim.org/bioclim

```
files <- list.files(path = "data/climatic_layers/rawlayers/current/",
                    pattern = "*.bil$",
                    full.names = TRUE)

files <- mixedsort(sort(files))
files_raster <- raster::stack(files)
```

For loop

This mask, crops, and writes the Florida extent layers.

```
out <- "data/climatic_layers/projection/current/"
end <- ".asc"
for(i in 1:20){
  ### Subset the raster
  raster <- files_raster[[i]]
  ### Set outputname
  name <- names(raster)
  outpath <- paste(out, name, sep = "")
  outfile <- paste(outpath, end, sep = "")
  ### Mask raster
  raster_d <- mask(raster, FL)
  ### Crop raster
  raster_d <- crop(raster_d, extent(FL))
  ### Write new raster as asc
  writeRaster(raster_d, outfile, format = "ascii")
}
```

Correlation

Based on the newly created layers, we are going to run a correlation to decide which layers to retain. If two layers are highly correlated ($>|0.80|$) we can only retain one of those layers - **we manually decide which layers to keep**.

```
### Read in the clipped files
clippedfiles <- list.files(path = "data/climatic_layers/projection/current/",
                           pattern = "*.asc",
                           full.names = TRUE)
clippedfiles <- mixedsort(sort(clippedfiles))
clippedfiles <- clippedfiles[2:20]
flstack <- raster::stack(clippedfiles)

### Pearson correlation
corr <- layerStats(flstack, 'pearson', na.rm=TRUE)
c <- corr$`pearson correlation coefficient`
c <- abs(c)
write.csv(c, "data/climatic_layers/picklayers_correlation.csv")
```

Training Layers

These are the species specific layers that are trimmed to the convex hull + buffer extent.

Load Cleaned Occurrence Data

```
## Load species list from training folders
### also make path for training layers
species.list <- system("ls data/climatic_layers/training",
                       intern = TRUE)
species.list <- mixedsort(sort(species.list))
species.list.b <- as.character(lapply(species.list,
                                      function(x)
                                        paste("data/climatic_layers/training",
                                              paste(x, "/", sep=""), sep = "/")))

### Load cleaned records
species.files <- list.files(path = "data/cleaned_occurrence/ready/",
                            pattern = "*.csv",
                            full.names = TRUE)
species.files <- mixedsort(sort(species.files))
species.df <- lapply(species.files, read.csv)
```

Load selected layers

```
## Load in layer files
current_layers <- list.files(path = "data/climatic_layers/rawlayers/current/",
                             pattern = "*.asc",
                             full.names = TRUE)
current_layers <- mixedsort(sort(current_layers))
current_layers <- raster::stack(current_layers)
```

Create function to crop, mask, and write

```
training_layers <- function(dataframe, folder){
                  pts <- SpatialPoints(dataframe[,4:3])
                  hull <- gConvexHull(pts)
                  hull_plus_buffer <- gBuffer(hull,width=.2)
                  for (i in 1:8){
                    layer <- current_layers[[i]]
                    name <- names(layer)
                    layerm <- mask(layer, hull_plus_buffer)
                    layermc <- crop(layerm,
                                    extent(hull_plus_buffer))
                    filefull <- paste(folder, name, sep = "")
                    writeRaster(layermc,
                                filefull,
                                format="ascii",
                                overwrite = TRUE)
                  }
}
```

For loop

This creates and writes the training layers for all species.

```
for(i in 1:25){
  folder <- species.list.b[i]
  dataframe <-  species.df[[i]]
  training_layers(dataframe = dataframe, folder = folder)
}
```

Plot training layers

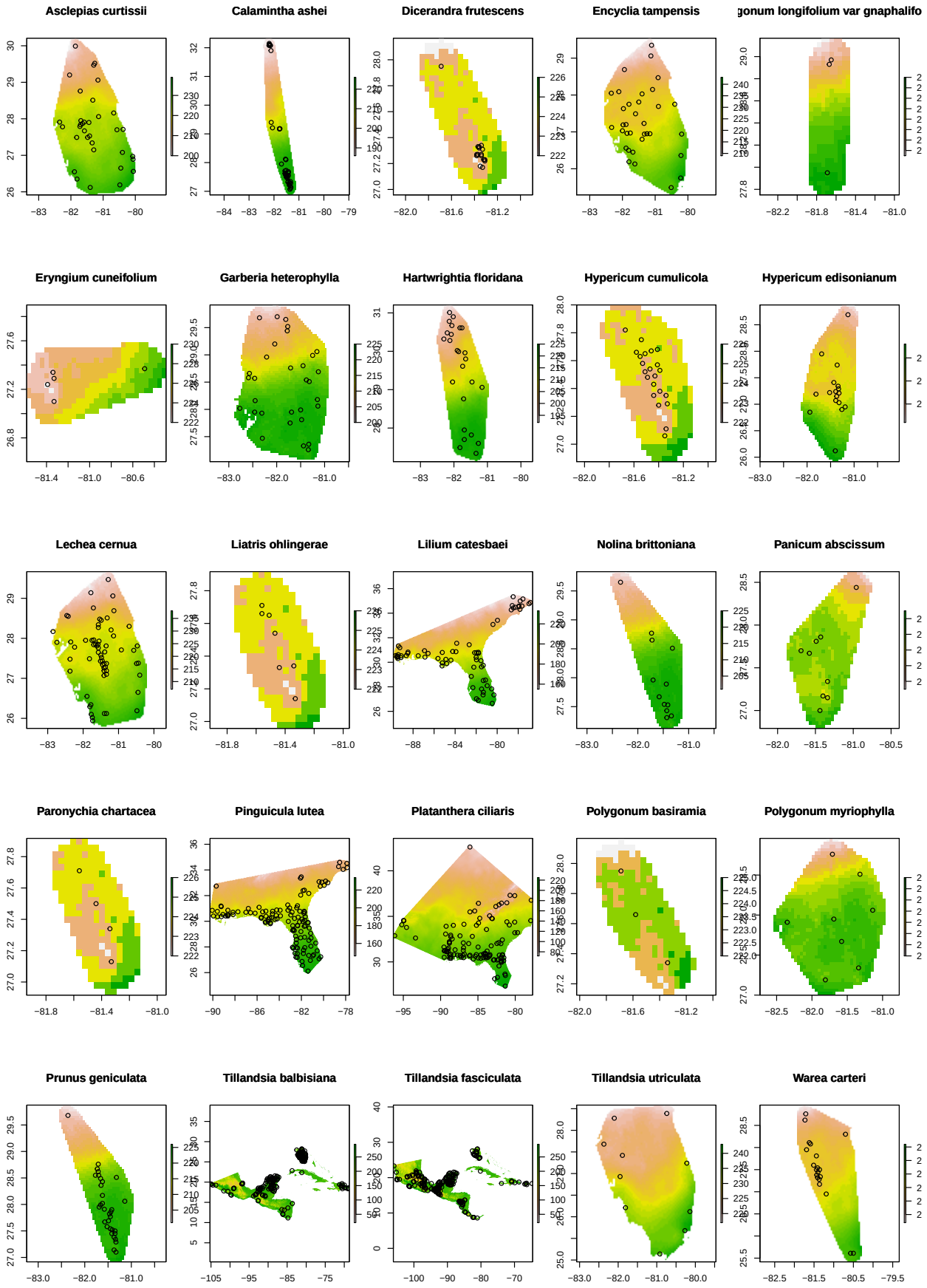
```
#### Make list of training folders
training <- system("ls data/climatic_layers/training",
                       intern = TRUE)
training  <- mixedsort(sort(training))

#### Load in bio1 for all species
front <- "data/climatic_layers/training/"
end <- "/bio1.asc"
files <- c()
for(i in 1:length(training)){
    name <- training[i]
    filed <- paste(front,
                   paste(name, end, sep = ""),
                   sep = "")
    files[i] <- filed
}

#### Read in occurrence records
ocrecords <- list.files(path = "data/cleaned_occurrence/ready/",
                        pattern = "*.csv",
                        full.names = TRUE)
ocrecords_df <- lapply(ocrecords, read.csv)

#### Plot
training <- gsub("_", " ", training)

par(mfrow=c(5,5), mar = c(5, 2, 5, 2))
for(i in 1:length(files)){
    name <- as.character(training[i])
    bio1 <- raster(files[i])
    plot(bio1, main = name)
    title(main = name, font = 2, cex = 0.5)
    points(x = ocrecords_df[[i]]$Longitude,
           y = ocrecords_df[[i]]$Latitude)
}
```


Asclepias curtissii
Calamintha ashei
Dicerandra frutescens
Encyclia tampensis
gonum longifolium var gnaphalifo
Eryngium cuneifolium
Garberia heterophylla
Hartwrightia floridana
Hypericum cumulicola
Hypericum edisonianum
Lechea cernua
Liatris ohlingerae
Lilium catesbaei
Nolina brittoniana
Panicum abscissum
Paronychia chartacea
Pinguicula lutea
Platanthera ciliaris
Polygonum basiramia
Polygonum myriophylla
Prunus geniculata
Tillandsia balbisiana
Tillandsia fasciculata
Tillandsia utriculata
Warea carteri

Future layers

Both sets of raw future layers were obtained on WorldClim. We will mask/crop these layers to the extent of Florida.

#2050 projection

```
future50 <- list.files(path = "data/climatic_layers/rawlayers/CCSM4_rcp26/50/", full.names = TRUE )
future50 <- mixedsort(sort(future50))
f50stack <- raster::stack(future50)

out <- "data/climatic_layers/projection/CCSM4_rcp26_50/"
end <- ".asc"
for(i in 1:8){
  raster <- f50stack[[i]]
  name <- names(raster)
  outpath <- paste(out, name, sep = "")
  outfile <- paste(outpath, end, sep = "")
  raster_d <- mask(raster, FL)
  raster_d <- crop(raster_d, extent(FL))
  writeRaster(raster_d, outfile, format = "ascii")
}
```

2070 projection

```
future70 <- list.files(path = "data/climatic_layers/rawlayers/CCSM4_rcp26/70/", full.names = TRUE )
f70stack <- raster::stack(future70)

out <- "data/climatic_layers/projection/CCSM4_rcp26_70/"
end <- ".asc"
for(i in 1:8){
  raster <- f70stack[[i]]
  name <- names(raster)
  outpath <- paste(out, name, sep = "")
  outfile <- paste(outpath, end, sep = "")
  raster_d <- mask(raster, FL)
  raster_d <- crop(raster_d, extent(FL))
  writeRaster(raster_d, outfile, format = "ascii")
}
```

05_ClimaticNiche

Load Packages

```
library(dplyr)
library(tidyr)
library(raster)
library(tibble)
library(gtools)
library(caret)
library(factoextra)
library(FactoMineR)
library(ggbiplot)
library(agricolae)
source("functions/ClimaticNicheFunctions.R")
```

Load Occurrence Records

```
ocfiles <- list.files(path = "data/cleaned_occurrence/maxent/",
                      pattern = "*Gaynor.csv",
                      full.names = TRUE, recursive = FALSE)
ocdf <- lapply(ocfiles, read.csv)
ocdfc <- bind_rows(ocdf)
ocdfc <- ocdfc[,2:4]
```

Seperate into species

```
unique(ocdfc$name)
```

```
## [1] "Hypericum cumulicola"
## [2] "Eriogonum longifolium var. gnaphalifolium"
## [3] "Paronychia chartacea"
## [4] "Pinguicula lutea"
## [5] "Polygonum basiramia"
## [6] "Polygonum myriophylla"
```

```
Eriogonum_longifolium <- ocdfc %>%
                         filter(name == "Eriogonum longifolium var. gnaphalifolium")

Paronychia_chartacea <- ocdfc %>%
                        filter(name == "Paronychia chartacea")

Pinguicula_lutea <- ocdfc %>%
                    filter(name == "Pinguicula lutea")

Polygonum_basiramia <- ocdfc %>%
                       filter(name == "Polygonum basiramia")

Polygonum_myriophylla <- ocdfc %>%
                         filter(name == "Polygonum myriophylla")
```

Load Layers

Note: For climatic niche investigation you can **ONLY** use current layers.
Why? Your points only occur in the present.

```
currentfiles <- list.files(path = "data/climatic_layers/rawlayers/current/",
                           pattern = "*.asc",
                           full.names = TRUE)
currentfiles <- mixedsort(sort(currentfiles))
current_raster<- raster::stack(currentfiles)
```

Point sample

```
convert_ptExtract <- function(dataframe, name){
                      pt_extacted <- raster::extract(current_raster, dataframe[3:2])
                      value_df <- as.data.frame(pt_extacted)
                      value_df_DONE <- value_df %>%
                                       mutate(species = name)
                      return(value_df_DONE)
}
```

```
ptExtract_Eriogonum_longifolium_df <- convert_ptExtract(Eriogonum_longifolium,
                                        "Eriogonum longifolium var. gnaphalifolium")
ptExtract_Paronychia_chartacea_df <- convert_ptExtract(Paronychia_chartacea,
                                        "Paronychia chartacea")
ptExtract_Pinguicula_lutea_df <- convert_ptExtract(Pinguicula_lutea,
                                        "Pinguicula lutea")
ptExtract_Polygonum_basiramia_df <- convert_ptExtract(Polygonum_basiramia,
                                        "Polygonum basiramia")
ptExtract_Polygonum_myriophylla_df <- convert_ptExtract(Polygonum_myriophylla,
                                        "Polygonum myriophylla")
```

Combined dataframes

```
pointsamples_combined <- rbind(ptExtract_Eriogonum_longifolium_df,
                           ptExtract_Pinguicula_lutea_df,
                           ptExtract_Paronychia_chartacea_df,
                           ptExtract_Polygonum_basiramia_df,
                           ptExtract_Polygonum_myriophylla_df)
pointsamples_combined <- pointsamples_combined %>%
                         drop_na(bio1, bio2, bio3,
                                 bio5, bio9, bio12,
                                 bio14, bio18)
```

```
pointsamples_combined <- pointsamples_combined %>%
                         dplyr::mutate(code =
                         ifelse(species == "Eriogonum longifolium var. gnaphalifolium",
                                           "FA",
                                ifelse(species == "Pinguicula lutea", "FA",
                                       ifelse(species == "Polygonum basiramia", "NF",
                                              ifelse(species == "Paronychia chartacea", "NF",
                                ifelse(species == "Polygonum myriophylla", "NF", NA))))))
```

PCA

Create two dataframes

```
data.bioclim <- pointsamples_combined[, 1:8]
data.species <- pointsamples_combined[, 9]
data.groups <- pointsamples_combined[, 10]
```

run a PCA

```
data.pca <- prcomp(data.bioclim, scale. = TRUE)
```

Understanding the PCA

When you use the command prcomp, your loading variables show up as rotational variables. Thanks to a really great answer on stack overflow you can even convert the rotational variable to show the relative contribution.

```
(loadings <- data.pca$rotation)
```

```
##               PC1         PC2        PC3         PC4         PC5         PC6
## bio1  -0.44846531 -0.29831196 0.11172014 -0.09728073  0.29069669 -0.39819211
## bio2  -0.07067203  0.68091286 0.08756900  0.40276203 -0.37913895 -0.04678725
## bio3  -0.48240669  0.06343923 0.13469023  0.15777314 -0.20047660 -0.59780763
## bio5  -0.27719365  0.53387094 0.08574164 -0.02692321  0.73669398  0.21059439
## bio9  -0.31303530  0.12058085 0.44232677 -0.66420155 -0.38175856  0.30682344
## bio12  0.30781236 -0.06025768 0.70405197  0.05071290  0.19800090 -0.12156012
## bio14  0.44839667  0.08873559 0.38002302  0.02252768  0.03253721 -0.28857102
## bio18 -0.30088648 -0.36370345 0.34437996  0.59871444 -0.05326891  0.49531739
##               PC7          PC8
## bio1   0.09845314 -0.659688843
## bio2  -0.08640675 -0.456153537
## bio3  -0.02585583  0.567443460
## bio5   0.10516361  0.178722121
## bio9   0.10356424 -0.006834760
## bio12 -0.59107843  0.002159763
## bio14  0.74935922  0.016440198
## bio18  0.22242850  0.049792030
```

There are two options to convert the loading to show the relative contribution, they both give the same answer so either can be used.

```
(loadings_relative_A <- t(t(abs(loadings))/rowSums(t(abs(loadings))))*100)
```

```
##             PC1       PC2       PC3       PC4       PC5       PC6       PC7
## bio1  16.930448 13.499424  4.877538  4.813743 12.791529 16.149558  4.966587
## bio2   2.668008 30.813150  3.823134 19.929877 16.683255  1.897560  4.358893
## bio3  18.211802  2.870797  5.880379  7.807090  8.821574 24.245404  1.304329
## bio5  10.464606 24.159105  3.743354  1.332242 32.416753  8.541119  5.305105
## bio9  11.817699  5.456610 19.311340 32.866692 16.798526 12.443900  5.224423
## bio12 11.620523  2.726823 30.737879  2.509427  8.712635  4.930138 29.817663
## bio14 16.927857  4.015526 16.591250  1.114737  1.431735 11.703633 37.802328
## bio18 11.359057 16.458566 15.035126 29.626192  2.343993 20.088687 11.220673
##              PC8
## bio1  34.0531191
## bio2  23.5466324
```

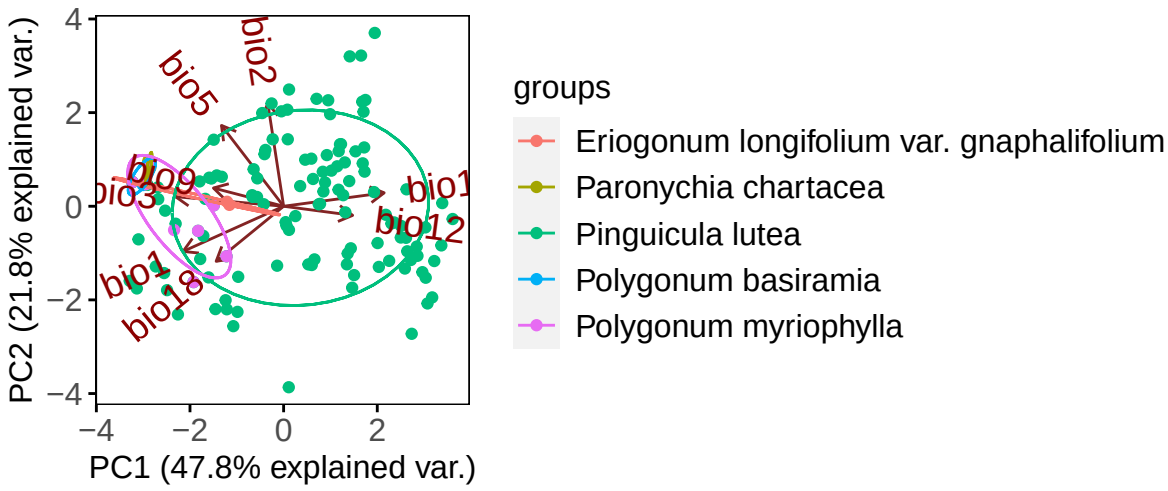
```
## bio3  29.2914151
## bio5   9.2256307
## bio9   0.3528101
## bio12  0.1114869
## bio14  0.8486425
## bio18  2.5702631
```

```
(loadings_relative_B <- sweep(x = abs(loadings), MARGIN = 2,
                              STATS = colSums(abs(loadings)),
                              FUN = "/")*100)
```

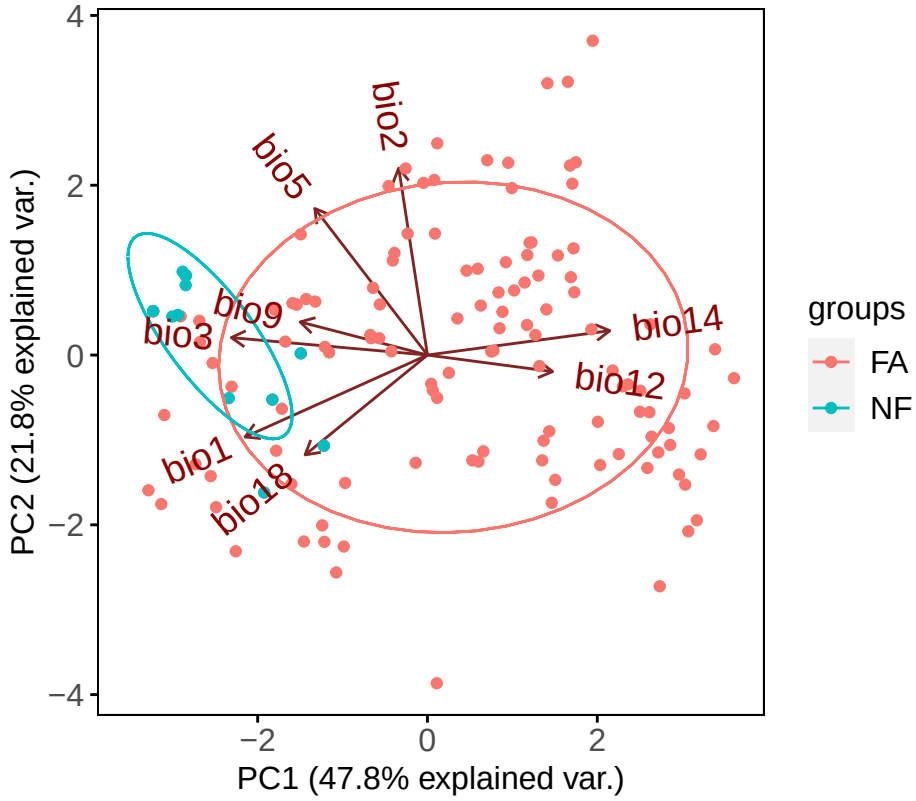
```
##             PC1       PC2       PC3       PC4       PC5       PC6       PC7
## bio1  16.930448 13.499424  4.877538  4.813743 12.791529 16.149558  4.966587
## bio2   2.668008 30.813150  3.823134 19.929877 16.683255  1.897560  4.358893
## bio3  18.211802  2.870797  5.880379  7.807090  8.821574 24.245404  1.304329
## bio5  10.464606 24.159105  3.743354  1.332242 32.416753  8.541119  5.305105
## bio9  11.817699  5.456610 19.311340 32.866692 16.798526 12.443900  5.224423
## bio12 11.620523  2.726823 30.737879  2.509427  8.712635  4.930138 29.817663
## bio14 16.927857  4.015526 16.591250  1.114737  1.431735 11.703633 37.802328
## bio18 11.359057 16.458566 15.035126 29.626192  2.343993 20.088687 11.220673
##              PC8
## bio1  34.0531191
## bio2  23.5466324
## bio3  29.2914151
## bio5   9.2256307
## bio9   0.3528101
## bio12  0.1114869
## bio14  0.8486425
## bio18  2.5702631
```

Plotting the PCA

```
theme <- plotting_pca_theme(theme = TRUE)
g <- ggbiplot(data.pca, obs.scale = 1, var.scale = 1,
              groups = data.species, ellipse = TRUE,
              varname.size = 5)
g <- g +
    theme(legend.position = "right",
          legend.direction = "vertical")
g <- g + theme
g
```




```
theme <- plotting_pca_theme(theme = TRUE)
g <- ggbiplot(data.pca, obs.scale = 1, var.scale = 1,
              groups = data.groups, ellipse = TRUE,
              varname.size = 5)
g <- g +
    theme(legend.position = "right",
          legend.direction = "vertical")
g <- g + theme
g
```



ANOVA

```
bio1_aov <- aov(lm(bio1 ~ species, data = pointsamples_combined))
bio2_aov <- aov(lm(bio2 ~ species, data = pointsamples_combined))
bio3_aov <- aov(lm(bio3 ~ species, data = pointsamples_combined))
bio5_aov <- aov(lm(bio5 ~ species, data = pointsamples_combined))
bio9_aov <- aov(lm(bio9 ~ species, data = pointsamples_combined))
bio12_aov <- aov(lm(bio12 ~ species, data = pointsamples_combined))
bio14_aov <- aov(lm(bio14 ~ species, data = pointsamples_combined))
bio18_aov <- aov(lm(bio18 ~ species, data = pointsamples_combined))

## Make groups
bio1_group <- aov_to_group(bio1_aov)
bio2_group <- aov_to_group(bio2_aov)
bio3_group <- aov_to_group(bio3_aov)
bio5_group <- aov_to_group(bio5_aov)
bio9_group <- aov_to_group(bio9_aov)
bio12_group <- aov_to_group(bio12_aov)
bio14_group <- aov_to_group(bio14_aov)
bio18_group <- aov_to_group(bio18_aov)

### Make plotable
bio1part <- pointsamples_combined %>%
            dplyr::select(species, bio1)
bio1pl <- left_join(bio1part, bio1_group, by = "species")
bio1pl <- dplyr::rename(bio1pl, bio = bio1.x)

bio2part <- pointsamples_combined %>%
            dplyr::select(species, bio2)
bio2pl <- left_join(bio2part, bio2_group, by = "species")
bio2pl <- dplyr::rename(bio2pl, bio = bio2.x)

bio3part <- pointsamples_combined %>%
            dplyr::select(species, bio3)
bio3pl <- left_join(bio3part, bio3_group, by = "species")
bio3pl <- dplyr::rename(bio3pl, bio = bio3.x)

bio5part <- pointsamples_combined %>%
            dplyr::select(species, bio5)
bio5pl <- left_join(bio5part, bio5_group, by = "species")
bio5pl <- dplyr::rename(bio5pl, bio = bio5.x)

bio9part <- pointsamples_combined %>%
            dplyr::select(species, bio9)
bio9pl <- left_join(bio9part, bio9_group, by = "species")
bio9pl <- dplyr::rename(bio9pl, bio = bio9.x)

bio12part <- pointsamples_combined %>%
            dplyr::select(species, bio12)
bio12pl <- left_join(bio12part, bio12_group, by = "species")
bio12pl <- dplyr::rename(bio12pl, bio = bio12.x)

bio14part <- pointsamples_combined %>%
```

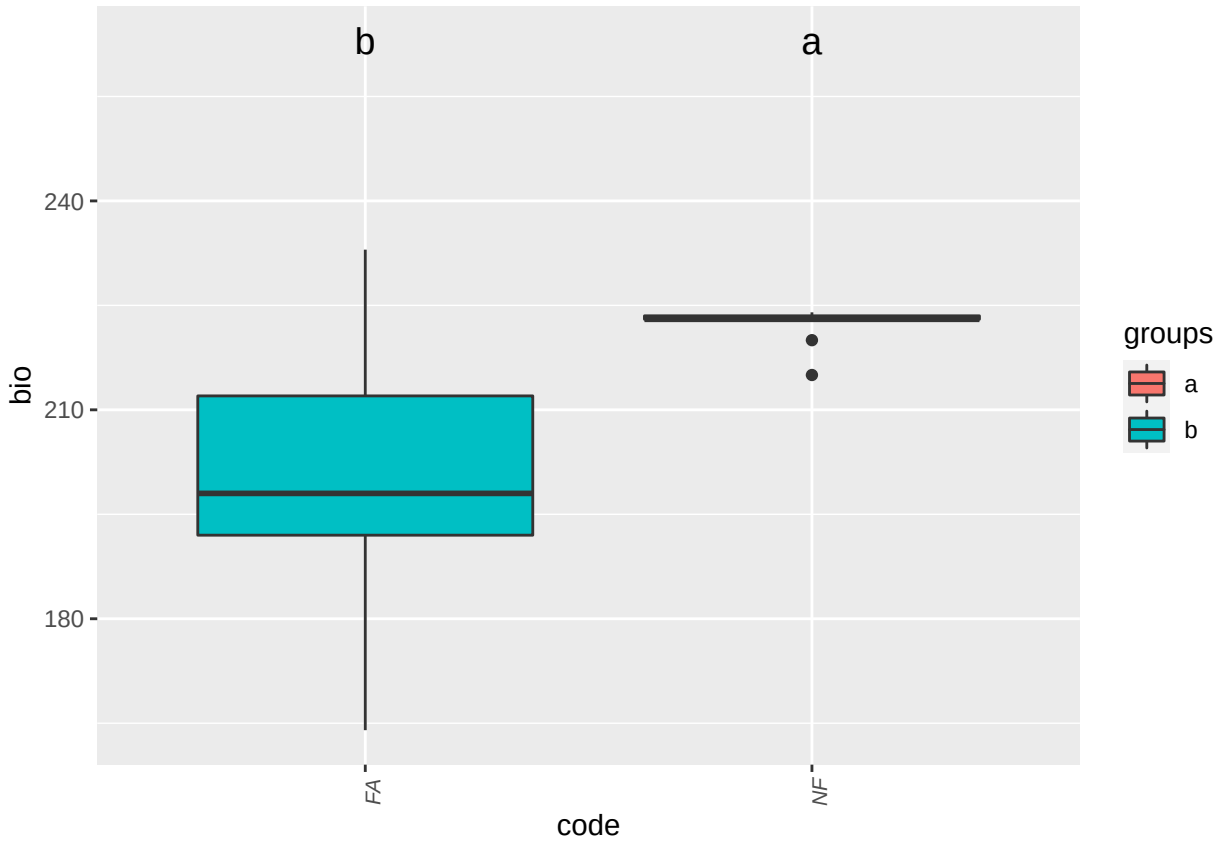
```
            dplyr::select(species, bio14)
bio14pl <- left_join(bio14part, bio14_group, by = "species")
bio14pl <- dplyr::rename(bio14pl, bio = bio14.x)

bio18part <- pointsamples_combined %>%
            dplyr::select(species, bio18)
bio18pl <- left_join(bio18part, bio18_group, by = "species")
bio18pl <- dplyr::rename(bio18pl, bio = bio18.x)

## Plot
bio1plot <-  plotting_aov(bio1pl, bio1_group)
bio2plot <-  plotting_aov(bio2pl, bio2_group)
bio3plot <-  plotting_aov(bio3pl, bio3_group)
bio5plot <-  plotting_aov(bio5pl, bio5_group)
bio9plot <-  plotting_aov(bio9pl, bio9_group)
bio12plot <-  plotting_aov(bio12pl, bio12_group)
bio14plot <-  plotting_aov(bio14pl, bio14_group)
bio18plot <-  plotting_aov(bio18pl, bio18_group)
```

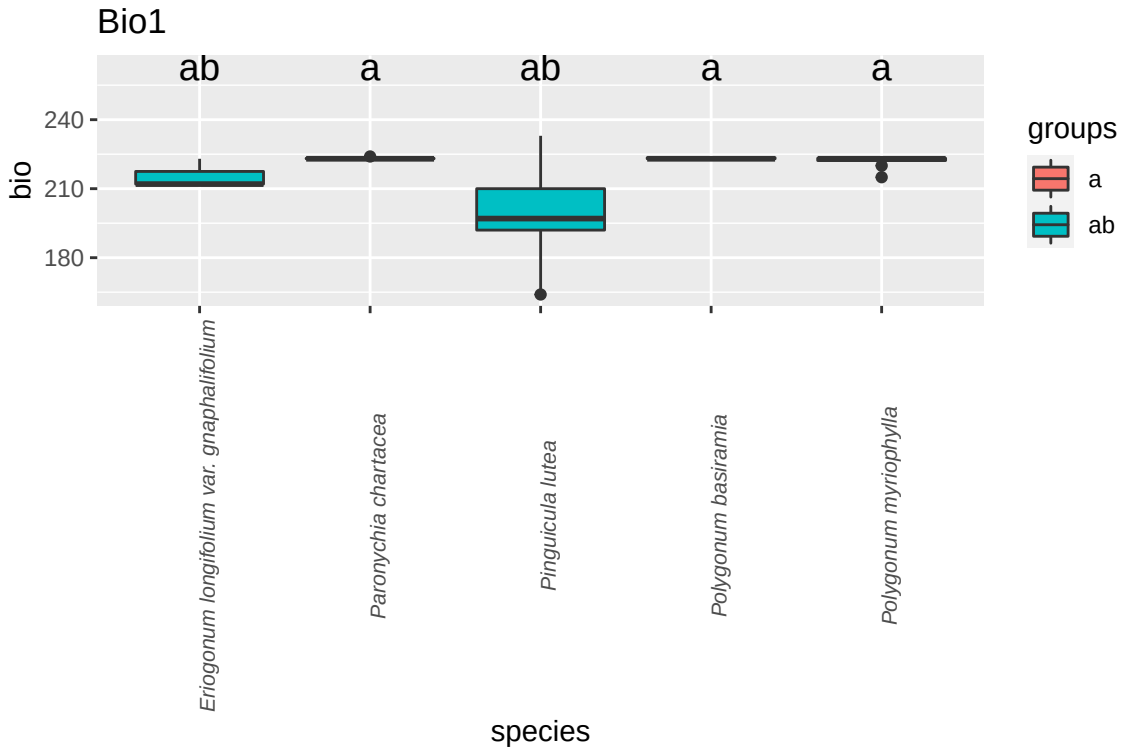
ANOVA with ~code

```
bio1_aov_c <- aov(lm(bio1 ~ code, data = pointsamples_combined))
## Make groups
b_c <- HSD.test(bio1_aov_c, trt = "code", alpha = 0.05)
bio1_group_c <- b_c$groups
bio1_group_c <- rownames_to_column(bio1_group_c, var = "code")
### Make plotable
bio1part_c <- pointsamples_combined %>%
            dplyr::select(code, bio1)
bio1pl_c <- left_join(bio1part_c, bio1_group_c, by = "code")
bio1pl_c <- dplyr::rename(bio1pl_c, bio = bio1.x)
bio1_aov_plot_c <- ggplot(bio1pl_c, aes(x = code, y = bio)) +
                                geom_boxplot(aes(fill = groups)) +
                                geom_text(data = bio1_group_c,
                                          mapping = aes(x = code,
                                                        y = (max(bio1pl_c$bio) + 30),
                                                        label = groups),
                                size = 5, inherit.aes = FALSE) +
                               theme(axis.text.x = element_text(angle = 90,
                                                                size = 8,
                                                                face = 'italic'))
plot(bio1_aov_plot_c)
```

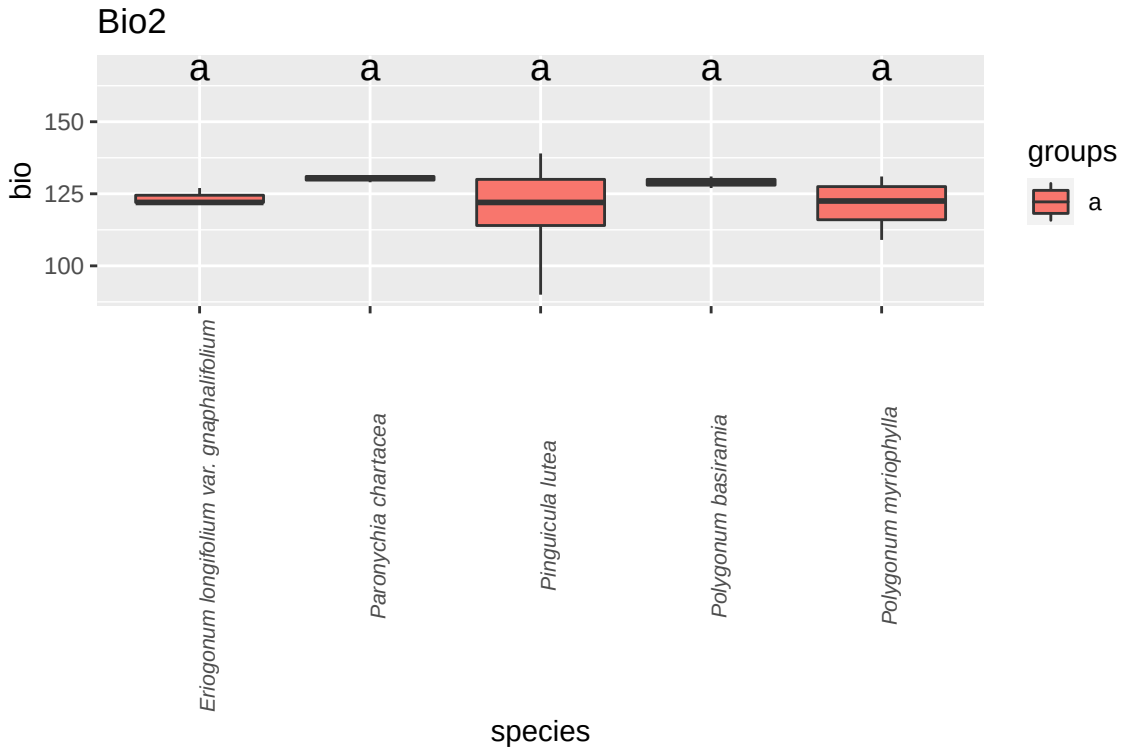


Plots of ANOVA with ~species

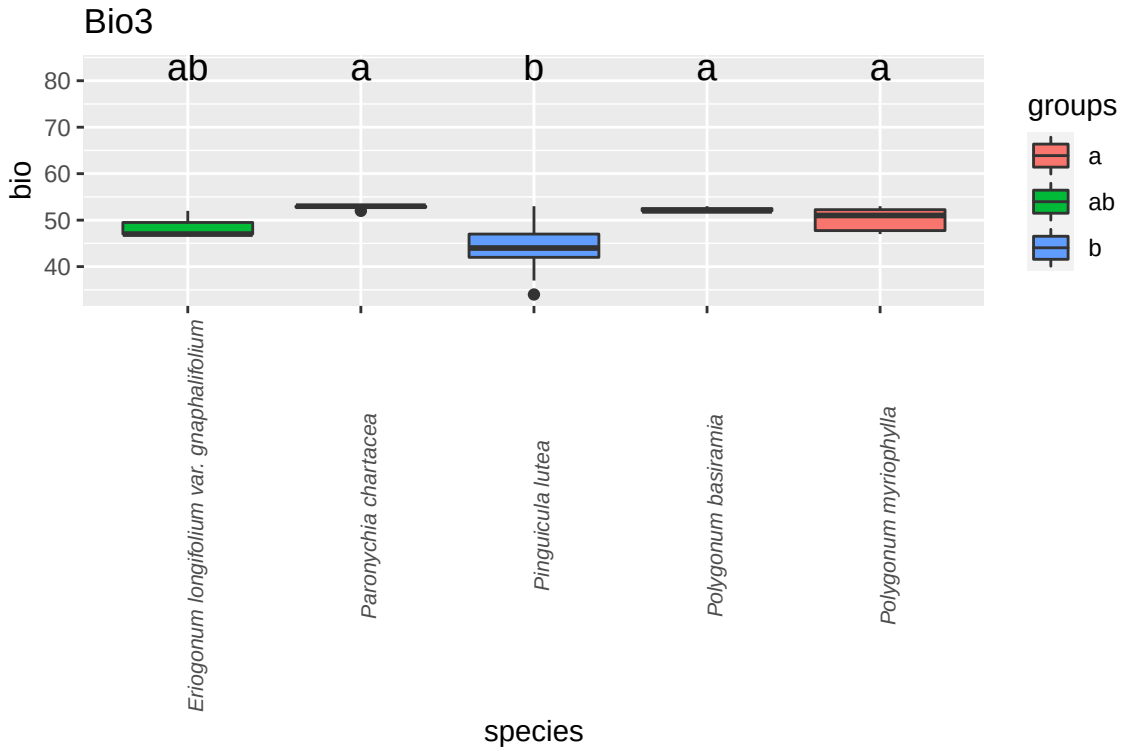
```
bio1plot + ggtitle("Bio1")
```



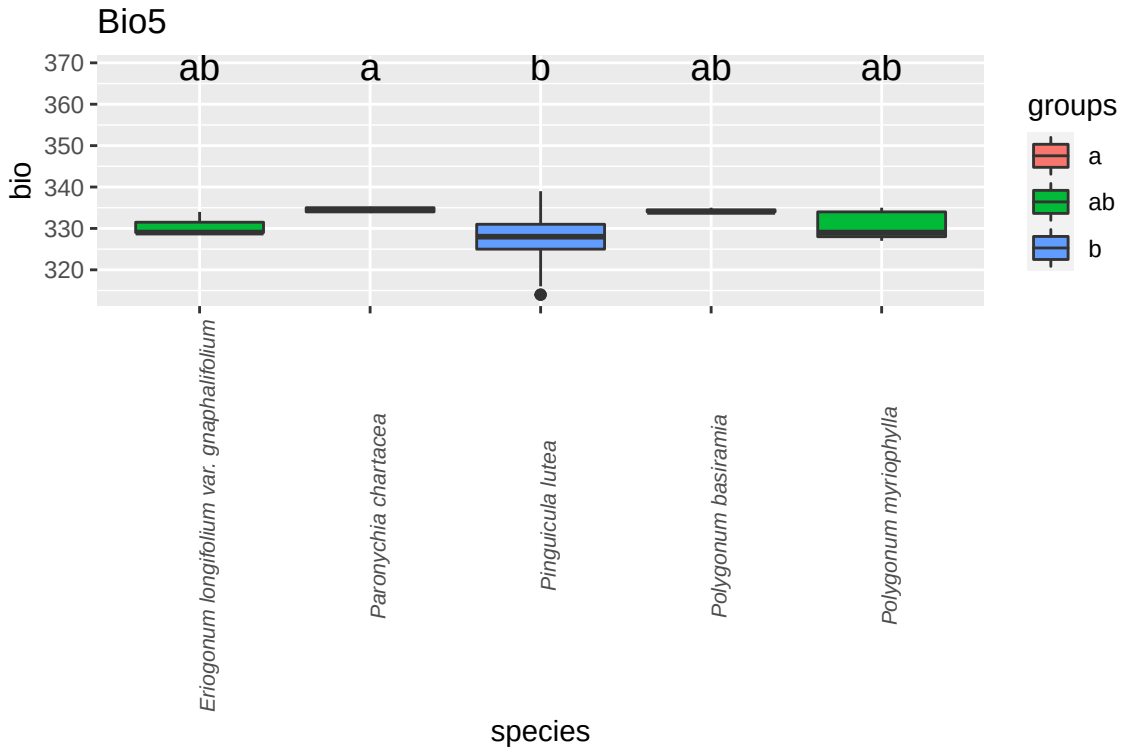

```
bio2plot + ggtitle("Bio2")
```



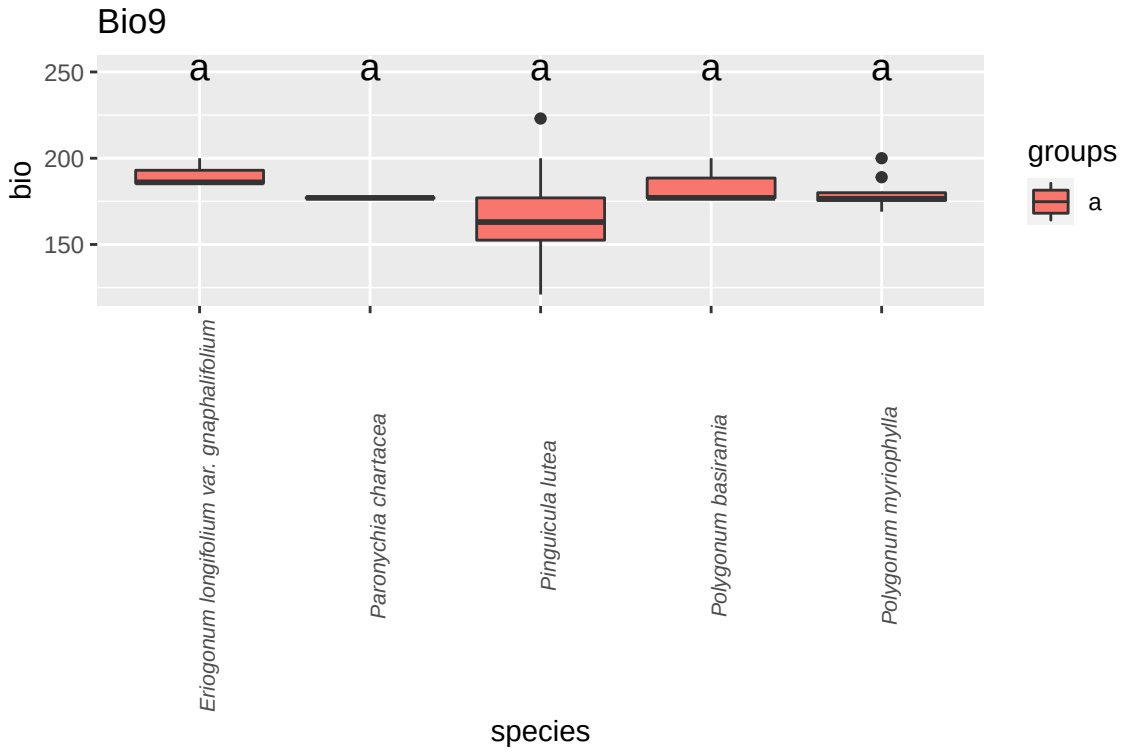

```
bio3plot + ggtitle("Bio3")
```



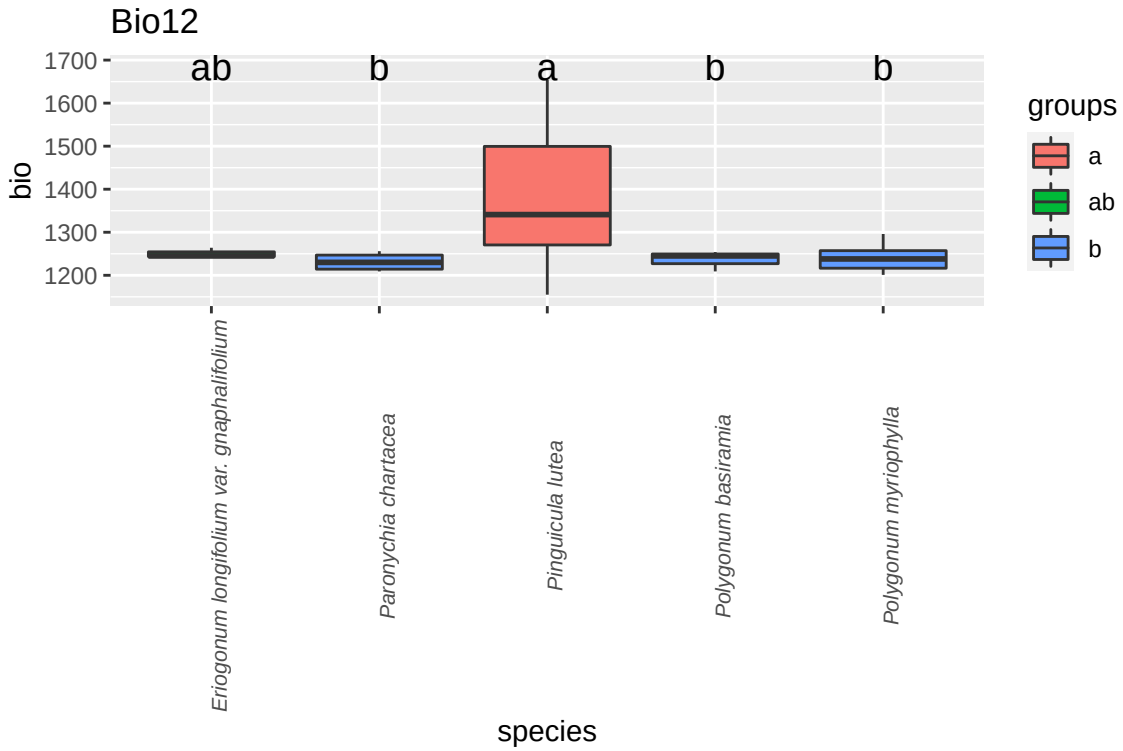

```
bio5plot + ggtitle("Bio5")
```



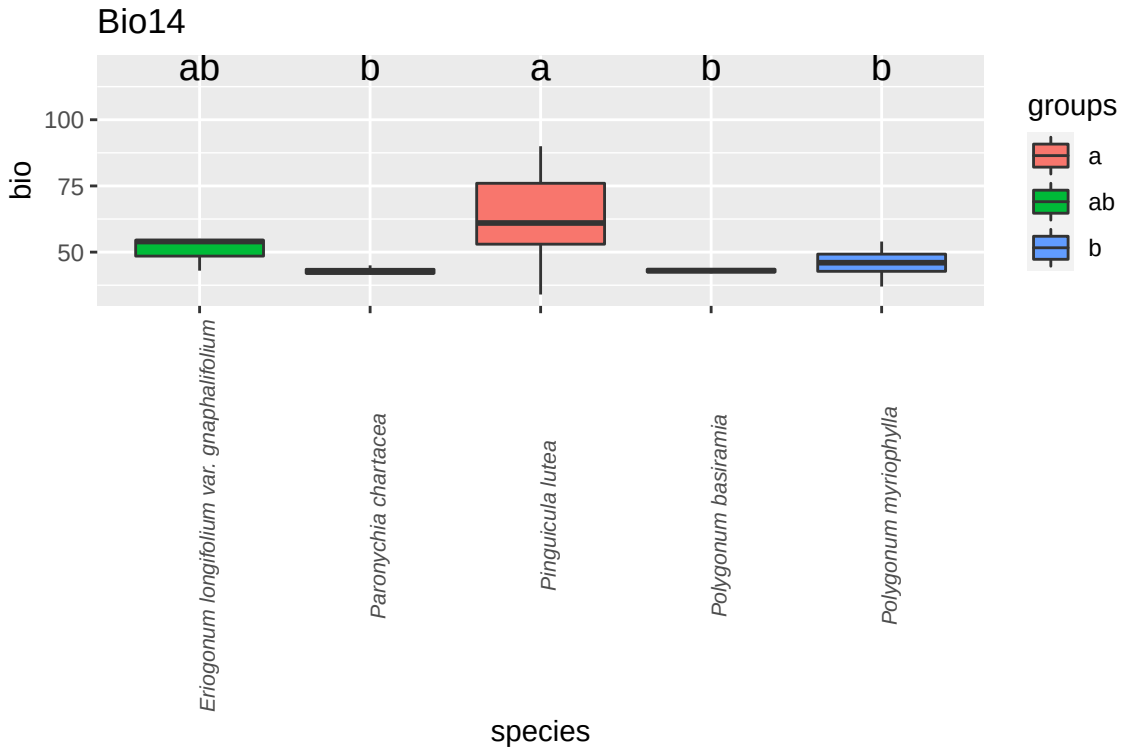

```
bio9plot + ggtitle("Bio9")
```



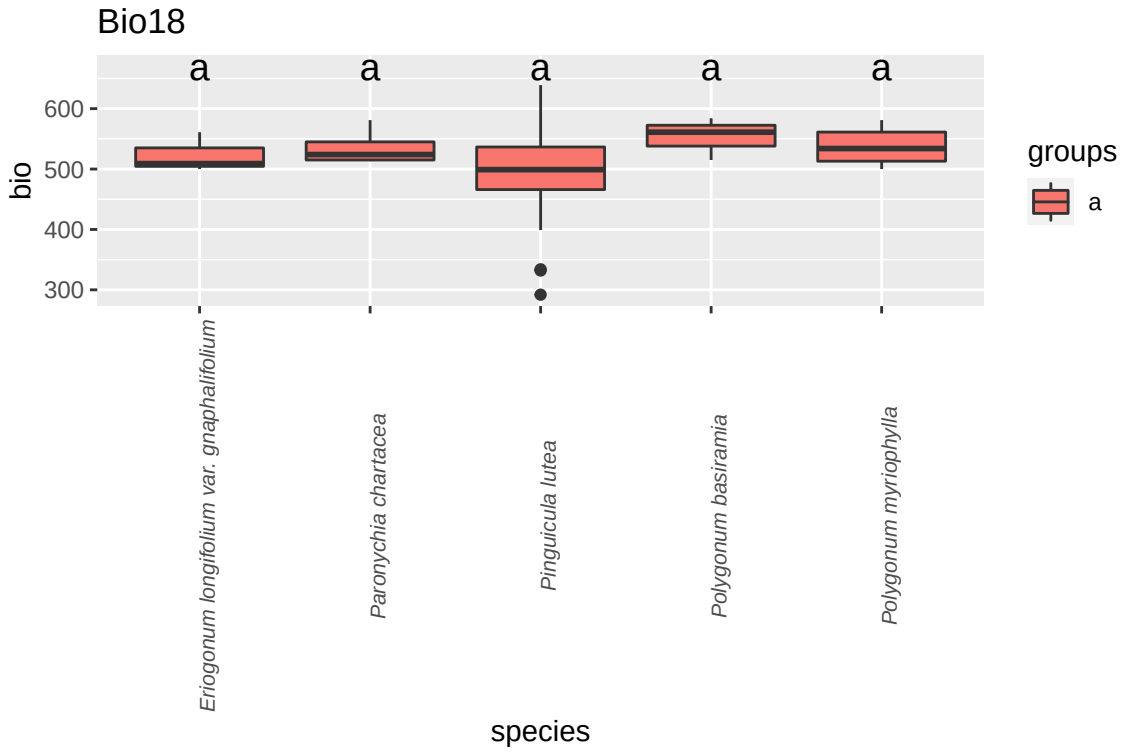

```
bio12plot + ggtitle("Bio12")
```




```
bio14plot + ggtitle("Bio14")
```




```
bio18plot + ggtitle("Bio18")
```



##

06_EcologicalNicheModeling Ecological niche modeling was done in maxent, using a bash script to automate this proccess. For certain species, we are unable to model ecological niche.

Load Packages

```
library(dplyr)
library(gtools)
library(raster)
library(ggplot2)
library(RColorBrewer)
```

Load cleaned records

names	count
Asclepias curtissii	34
Calamintha ashei	47
Dicerandra frutescens	16
Encyclia_tampensis	34
Eriogonum longifolium var. gnaphalifolium	3
Eryngium cuneifolium	5
Garberia heterophylla	31
Hartwrightia floridana	25
Hypericum_cumulicola	22
Hypericum_edisonianum	17
Lechea_cernua	63
Liatris ohlingerae	7
Lilium_catesbaei	66
Nolina brittoniana	12
Panicum abscissum	9
Paronychia chartacea	4
Pinguicula lutea	123
Platanthera ciliaris	116
Polygonum basiramia	3
Polygonum myriophylla	8
Prunus_geniculata	33
Tillandsia balbisiana	188
Tillandsia fasciculata	232
Tillandsia utriculata	10
Warea_carteri	20

Three models

Based on our trimmed layers we generated three models. We can use these models to understand the current niche and how climate change may affect each species.

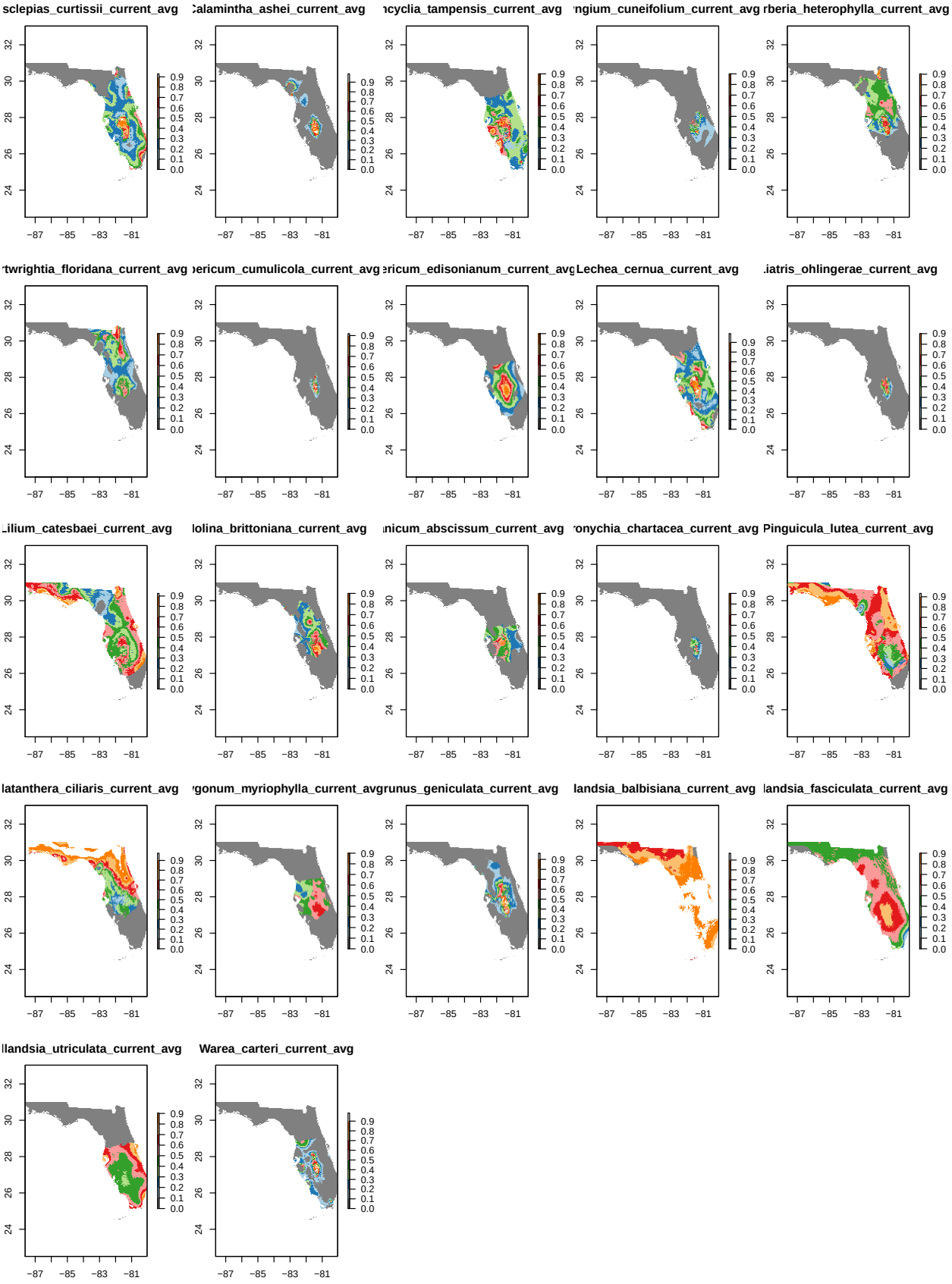
environmental layers	**projection layers**	**model**
training layers	current projections	current
training layers	2050 projections	2050

environmental layers	projection layers	model
training layers	2070 projections	2070

Current

```
current_avg <- list.files(path = "data/models/avg_only/present/",
           pattern = ".asc",
           full.names = TRUE)

l <- raster::stack(current_avg)
color <- c("#808080", brewer.pal(9, "Paired"))
b <- c(0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9)
par(mfrow=c(5, 5), mar=c(3,2,3,4))
for(i in 1:22){
  ras <- l[[i]]
  name <- names(ras)
  plot(ras, col = color, breaks = b)
  title(main = name, font = 2, cex = 0.5)
}
```

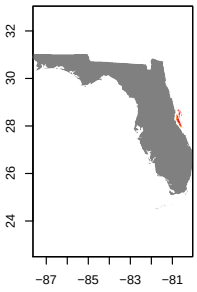

sclepias_curtissii_current_avg
Calamintha_ashei_current_avg
ıcyclia_tampensis_current_avg
ngium_cuneifolium_current_avg
rberia_heterophylla_current_avg
twrightia_floridana_current_avg
ericum_cumulicola_current_avg
ricum_edisonianum_current_avg
Lechea_cernua_current_avg
.iatris_ohlingerae_current_avg
Lilium_catesbaei_current_avg
olina_brittoniana_current_avg
nicum_abscissum_current_avg
onychia_chartacea_current_avg
Pinguicula_lutea_current_avg
latanthera_ciliaris_current_avg
gonum_myriophylla_current_avg
grunus_geniculata_current_avg
landsia_balbisiana_current_avg
landsia_fasciculata_current_avg
llandsia_utriculata_current_avg
Warea_carteri_current_avg

2050

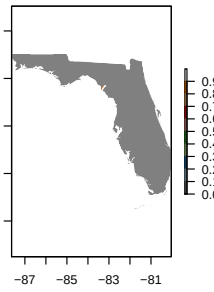
```
f50_avg <- list.files(path = "data/models/avg_only/2050/",
           pattern = ".asc",
           full.names = TRUE)

m <- raster::stack(f50_avg)
color <- c("#808080", brewer.pal(9, "Paired"))
b <- c(0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9)
par(mfrow=c(5, 5), mar=c(3,2,3,4))
for(i in 1:22){
  ras <- m[[i]]
  name <- names(ras)
  plot(ras, col = color, breaks = b)
  title(main = name, font = 2, cex = 0.5)
}
```

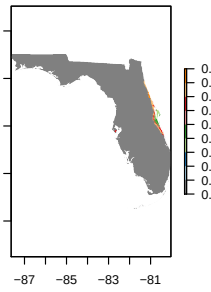
ias_curtissii_CCSM4_rcp26_50_a



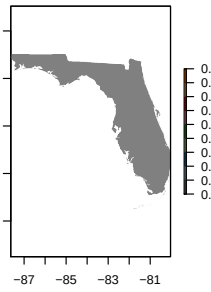
intha_ashei_CCSM4_rcp26_50_a



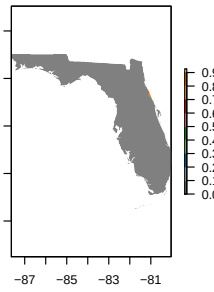
ia_tampensis_CCSM4_rcp26_50_a



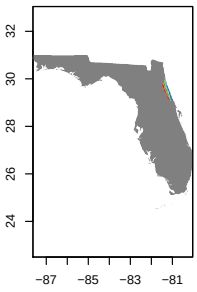
m_cuneifolium_CCSM4_rcp26_50_



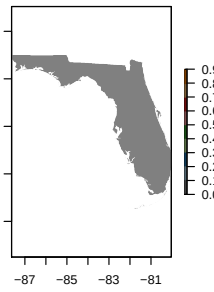
ı_heterophylla_CCSM4_rcp26_50_



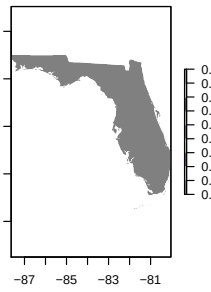
hia_floridana_CCSM4_rcp26_50_



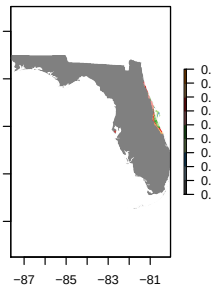
m_cumulicola_CCSM4_rcp26_50_



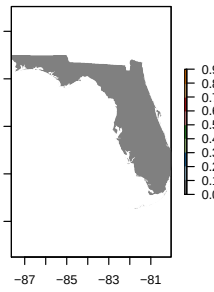
n_edisonianum_CCSM4_rcp26_5



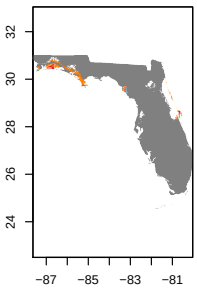
lea_cernua_CCSM4_rcp26_50_av



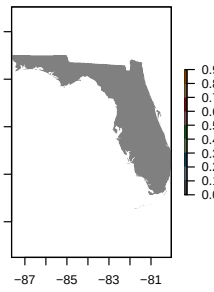
ohlingerae_CCSM4_rcp26_50_a



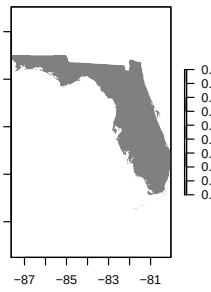
ı_catesbaei_CCSM4_rcp26_50_av



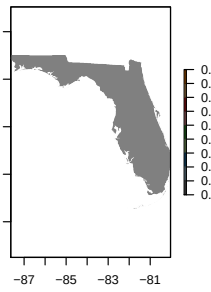
brittoniana_CCSM4_rcp26_50_a



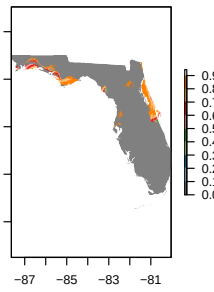
n_abscissum_CCSM4_rcp26_50_



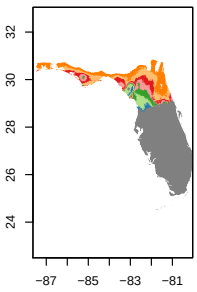
hia_chartacea_CCSM4_rcp26_50_



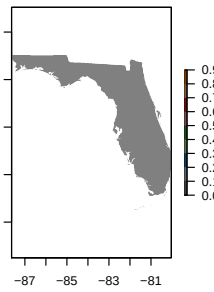
icula_lutea_CCSM4_rcp26_50_av



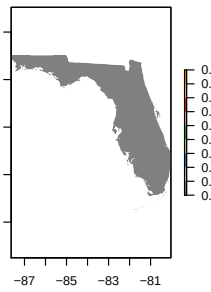
hera_ciliaris_CCSM4_rcp26_50_a



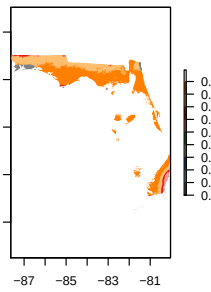
m_myriophylla_CCSM4_rcp26_50_



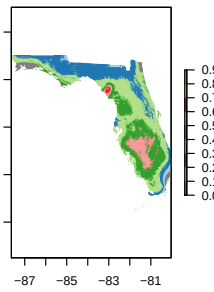
s_geniculata_CCSM4_rcp26_50_a



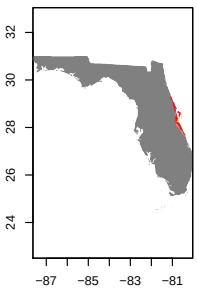
ia_balbisiana_CCSM4_rcp26_50_



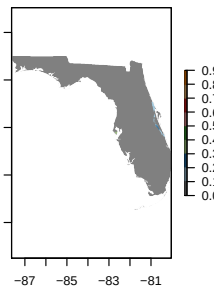
ia_fasciculata_CCSM4_rcp26_50_



sia_utriculata_CCSM4_rcp26_50_



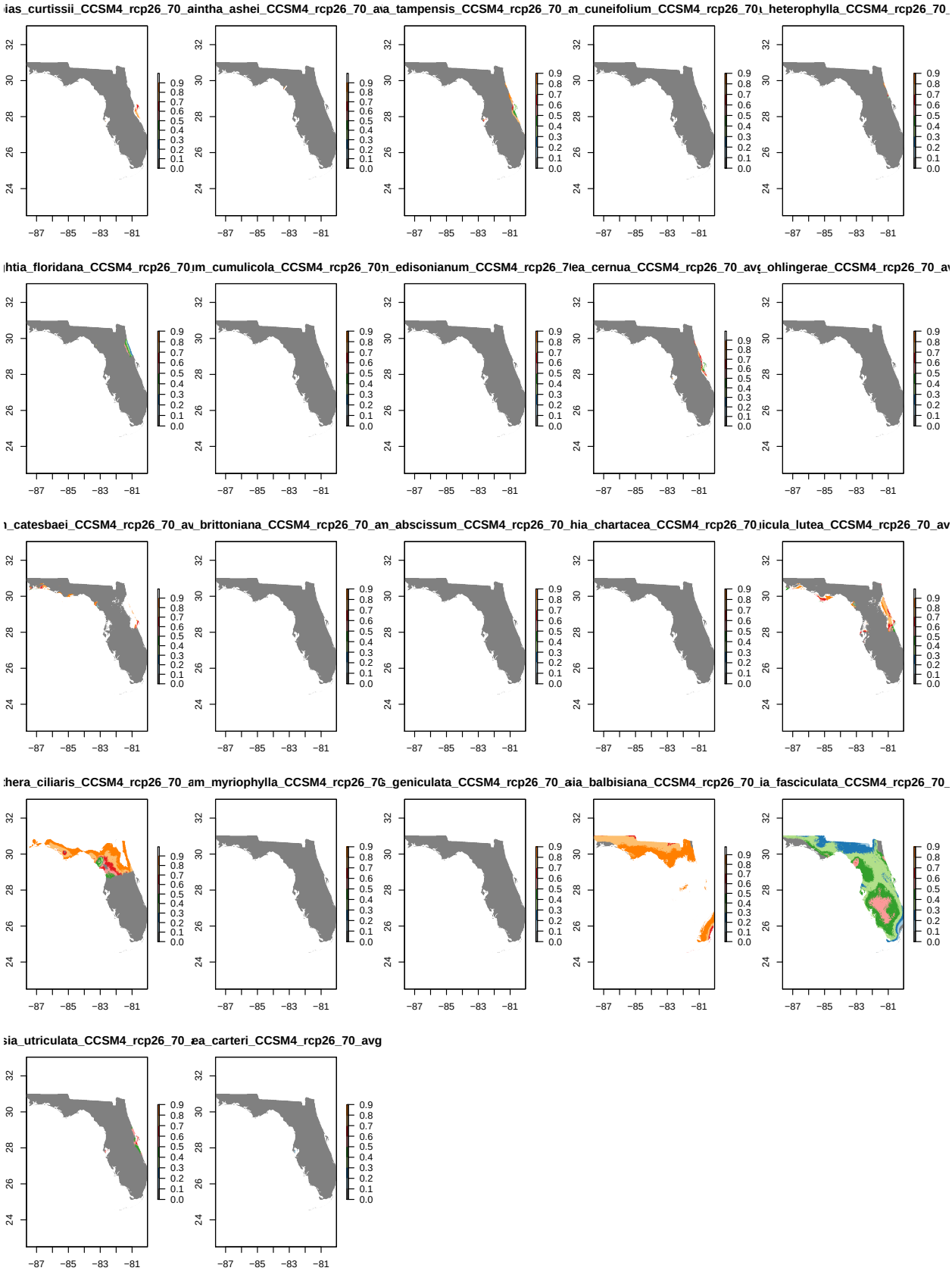
ea_carteri_CCSM4_rcp26_50_avg



2070

```
f70_avg <- list.files(path = "data/models/avg_only/2070/",
           pattern = ".asc",
           full.names = TRUE)

n <- raster::stack(f70_avg)
color <- c("#808080", brewer.pal(9, "Paired"))
b <- c(0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9)
par(mfrow=c(5, 5), mar=c(3,2,3,4))
for(i in 1:22){
  ras <- n[[i]]
  name <- names(ras)
  plot(ras, col = color, breaks = b)
  title(main = name, font = 2, cex = 0.5)
}
```


ias_curtissii_CCSM4_rcp26_70_a
ntha_ashei_CCSM4_rcp26_70_a
a_tampensis_CCSM4_rcp26_70_a
m_cuneifolium_CCSM4_rcp26_70
_heterophylla_CCSM4_rcp26_70_
htia_floridana_CCSM4_rcp26_70
m_cumulicola_CCSM4_rcp26_70
n_edisonianum_CCSM4_rcp26_7
lea_cernua_CCSM4_rcp26_70_av
_ohlingerae_CCSM4_rcp26_70_a
_catesbaei_CCSM4_rcp26_70_av
brittoniana_CCSM4_rcp26_70_an
abscissum_CCSM4_rcp26_70_hia_chartacea_CCSM4_rcp26_70
icula_lutea_CCSM4_rcp26_70_av
hera_ciliaris_CCSM4_rcp26_70_
m_myriophylla_CCSM4_rcp26_70
_geniculata_CCSM4_rcp26_70_a
ia_balbisiana_CCSM4_rcp26_70_
ia_fasciculata_CCSM4_rcp26_70_
sia_utriculata_CCSM4_rcp26_70_
ea_carteri_CCSM4_rcp26_70_avg

07_ENMProcessing

Load packages

```
library(dplyr)
library(raster)
library(ENMTools)
library(ENMeval)
```

Load current models

```
current_avg <- list.files(path = "data/models/avg_only/present/",
           pattern = ".asc",
           full.names = TRUE)

ca <- raster::stack(current_avg)
```

Niche breadth

The raster.breadth command in ENMTools measures the smoothness of suitability scores across a projected landscape. The higher the score, the more of the avalible niche space a species occupies.

We are interested in B2.

```
nb <- c()
for(i in 1:22){
  ras.enm <- ca[[i]]
  nb[[i]] <- ENMTools::raster.breadth(x = ras.enm)
}
nb <- dplyr::bind_rows(nb)
ras_names <- names(ca)
nb <- cbind(ras_names, nb)
```

ras_names	B1	B2
Asclepias_curtissii_current_avg	0.9387776	0.5255712
Calamintha_ashei_current_avg	0.8028150	0.1153488
Encyclia_tampensis_current_avg	0.9256507	0.4790048
Eryngium_cuneifolium_current_avg	0.7472364	0.0881734
Garberia_heterophylla_current_avg	0.8937690	0.3678689
Hartwrightia_floridana_current_avg	0.8989837	0.3735145
Hypericum_cumulicola_current_avg	0.6366444	0.0308430
Hypericum_edisonianum_current_avg	0.8596714	0.2614218
Lechea_cernua_current_avg	0.9242360	0.4622836
Liatris_ohlingerae_current_avg	0.6419262	0.0365825
Lilium_catesbaei_current_avg	0.9813482	0.8045729
Nolina_brittoniana_current_avg	0.8373902	0.2170737
Panicum_abscissum_current_avg	0.8107481	0.1731044
Paronychia_chartacea_current_avg	0.6439688	0.0379000
Pinguicula_lutea_current_avg	0.9880873	0.8770368
Platanthera_ciliaris_current_avg	0.9581522	0.6538569
Polygonum_myriophylla_current_avg	0.8449855	0.2434227
Prunus_geniculata_current_avg	0.8273031	0.1769305
Tillandsia_balbisiana_current_avg	0.9932132	0.9349009
Tillandsia_fasciculata_current_avg	0.9921367	0.9189140
Tillandsia_utriculata_current_avg	0.9127431	0.4484211
Warea_carteri_current_avg	0.8601461	0.2045062

Niche Overlap

Calculating niche overlap, Schoener's D, with ENMEval - Schoener's D ranges from 0 to 1. Zero represents no similarity between projections. One represents completely identical projections.

```
overlap <- ENMeval::calc.niche.overlap(ca)
write.csv(overlap, "data/output/nicheoverlap_040420.csv", row.names = FALSE)
```

Geographic overlap

Read in data

```
current <- raster("data/models/avg_only/present/Asclepias_curtissii_current_avg.asc")
future50 <- raster("data/models/avg_only/2050/Asclepias_curtissii_CCSM4_rcp26_50_avg.asc")
future70 <- raster("data/models/avg_only/2070/Asclepias_curtissii_CCSM4_rcp26_70_avg.asc")
## Read in FL shape file
FL <- rgdal::readOGR("data/climatic_layers/FL/FLstate2.shp")
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "/Users/shellygaynor/Dropbox/Teaching/CURE_FLClimateChange/data/climatic_layers/FL/FLstate2.s
## with 200 features
## It has 13 fields
```

Generate 10000 random points in the FL extent

```
FL_samples <- spsample(FL, n = 10000, type = "random")
plot(FL_samples)
```

Save as dataframe

```
FL_samples_df <- as.data.frame(FL_samples)
#write.csv(FL_samples_df, "data/output/geo_overlap/FL_samples_df.csv", row.names = FALSE)
FL_samples_df <- read.csv("data/output/geo_overlap/FL_samples_df.csv")
```

Point sample

ept_to_df Function

```
ept_to_df <- function(raster){
  pt <- raster::extract(raster, FL_samples_df)
  pt_df <- as.data.frame(pt)
  pt_df_done <- cbind(FL_samples_df, pt_df)
  return(pt_df_done)
}
```

Apply the funtion

```
current_pt <- ept_to_df(current)
future50_pt <- ept_to_df(future50)
future70_pt <- ept_to_df(future70)
```

Rename columns

```
current_pt <- current_pt %>%
              dplyr::rename(current = pt)
future50_pt <- future50_pt %>%
               dplyr::rename(future50 = pt)
future70_pt <- future70_pt %>%
               dplyr::rename(future70 = pt)
```

Join dataframes

```
geo_compare <- left_join(current_pt, future50_pt, by = c("x", "y"))
geo_compare <- left_join(geo_compare, future70_pt, by = c("x", "y"))
```

Convert suitability

Convert suitability of >= 0.25 to 1,convert suitability of < 0.25 to 0

```
geo_compare <- cbind(geo_compare[,1:2], apply(geo_compare[,3:5], 2,
                                              function(x) ifelse(x >= 0.25, 1, 0)))
geo_compare[is.na(geo_compare)] <- 0
```

Just to look at the datasheet

```
write.csv(geo_compare, "data/output/geo_overlap/geo_compare_example.csv",
          row.names = FALSE)
```

Compare

How many points found in the future distribution are also found in the present distribution?

```
geo_compare_current <- geo_compare %>%
                       filter(current == 1)

sum(geo_compare_current$future50)/sum(geo_compare_current$current)
```

```
## [1] 0.03992781
```

```
sum(geo_compare_current$future70)/sum(geo_compare_current$current)
```

```
## [1] 0.04443943
```

Future50

```
geo_compare_future50 <- geo_compare %>%
                       filter(future50 == 1)

sum(geo_compare_future50$current)/sum(geo_compare_future50$future50)
```

```
## [1] 0.9076923
```

```
sum(geo_compare_future50$future70)/sum(geo_compare_future50$future50)
```

```
## [1] 0.9487179
```

Future70

```
geo_compare_future70 <- geo_compare %>%
                        filter(future70 == 1)

sum(geo_compare_future70$current)/sum(geo_compare_future70$future70)
```

```
## [1] 0.9292453
```

```r
sum(geo_compare_future70$future50)/sum(geo_compare_future70$future70)
```

```
## [1] 0.8726415
```