Workshop in biogeographic regionalization

Gabriel Muñoz & Jean-Philippe Lessard

11/20/2019

Table of Contents

Tutorial: Delimiting biogeographic regions using R

Workshop objectives

1) Understand the basis of biogeographical regionalization
2) Retrieve species occurrences programatically
3) Quantify turnover across communities
4) Cluster commnutinies based on dissimilarity values

5) Visualize and export the results

Dependencies

These are the packages you will be using for this workshop.

```
library(mapr)
library(vegan)
library(tibble)
library(mapplots)
library(ggplot2)
library(rgbif)
library(raster)
library(plyr)
library(rgdal)
library(betapart)
library(RColorBrewer)
```

Remember!!

You can install packages using the `install.packages("packageName")`function.

You can get to the help page of any package/function by typing `?` and the package/function name. e.g. `?ggplot2`.

Defining the region of interest (ROI)

This section corresponds to: (1) Defining study area and (1) Defining purpose and objectives from both frameworks (i.e. Morrone / Kreft)

Defining region of interest, commonly abbreviated as **ROI**, is not more than delimiting geographic area in which you (*the researcher*) are interested. For example, you might only be interested in ecological process happening at a any given site, city, province, continent, or planet. As such, and depending of the particular question and interest, the boundaries of the ROI might coincide with political and/or physical boundaries.

Is important to clearly define the ROI because the spatial conclusions you can derive will only apply within the limits of the ROI. Additionally, size and shape of ROI might dramatically change the patterns observed.

These are two very simple common ways to define the ROI:

1) Using standardized geographical units. (e.g. political divisions, eco-regions, etc.)

2) Defining a extent polygon using geographical coordinates (e.g. bounding box)

Using the first alternative, focusing in country-wide ecological patterns is commonly needed. Luckily because of globalization needs, most of the world accepted political divisions and units are already standardized to match unique strings of 3 characters.

Have you observed the city codes of your plane ticket? (e.g. Montreal Airport = YUL) These are called isocodes (because ISO standards). You can query the isocode for any political unit.

Let's take a look for the isocode list at the country level.

This list shows how country names are coded in the GBIF database and its corresponding isocodes

```
head(isocodes) ## remember to load the rgbif package

##   code                 name            gbif_name
## 1   AD              Andorra              ANDORRA
## 2   AE United Arab Emirates UNITED_ARAB_EMIRATES
## 3   AF          Afghanistan          AFGHANISTAN
## 4   AG  Antigua and Barbuda      ANTIGUA_BARBUDA
## 5   AI             Anguilla             ANGUILLA
## 6   AL              Albania              ALBANIA

tail(isocodes)

##     code         name    gbif_name
## 244   YE        Yemen        YEMEN
## 245   YT      Mayotte      MAYOTTE
## 246   ZA South Africa SOUTH_AFRICA
## 247   ZM       Zambia       ZAMBIA
## 248   ZW     Zimbabwe     ZIMBABWE
## 249   NA      Namibia      NAMIBIA
```

Let's now explore how to query the codes for the country (or countries) we are interested.

```
roiNames <- c("United States", "Canada") ## You can modify the vector
here to match any desired countries

roiCodes <- isocodes$code[match(roiNames,isocodes$name)]
names(roiCodes) <- roiNames
roiCodes

## United States        Canada
##          "US"          "CA"
```

NOTE If you are interested to find isocodes for additional entities other than countries look at the `ISOcode` R package.

Downloading species occurrences from gbif

This section corresponds to: (2) Assembling distributional data and (2) Distribution data (i.e. Morrone / Kreft)

The GBIF (Global Biodiversity Information Facility) is a global (and the biggest) repository of species occurrence data. You can explore and download the information from their webpage. However, there is also an API (Application Programmable Interface) freely available. API's allow for programmatic communication between computers and servers. We will access to the GBIF server via the `rgbif` package for R.

Remember to use `?` to explore and get familiarized with all the contents of the package.

For now we will explore the functions only to retrieve the data.

First lets try with a basic search

```
#  define the taxa we are interested
taxa4search <- c("Hymenoptera")
query <- expand.grid(roiCodes, taxa4search, stringsAsFactors = F)
names(query) <- c("country", "taxa")
```

Let's use the `occ_data()` function from **rgbif** along with an `sapply` function to write a function that allow us to search for the occurrences each of the taxa and country combinations.

```
customSearGBIF <- function(query, limit,...){

  sapply(1:length(query$country),
         function(x) rgbif::occ_data(
           scientificName = query$taxa[[x]],
           country = query$country[[x]],
           limit = limit))
  }




testSearch <- customSearGBIF(query, 20)
testSearch[,1]$meta

## $offset
## [1] 0
##
## $limit
## [1] 20
##
## $endOfRecords
## [1] FALSE
##
## $count
## [1] 2841348
```

```
testSearch[,2]$meta

## $offset
## [1] 0
##
## $limit
## [1] 20
##
## $endOfRecords
## [1] FALSE
##
## $count
## [1] 837910
```

That is a lot of information available for Hymenopteras both in the US and Canada

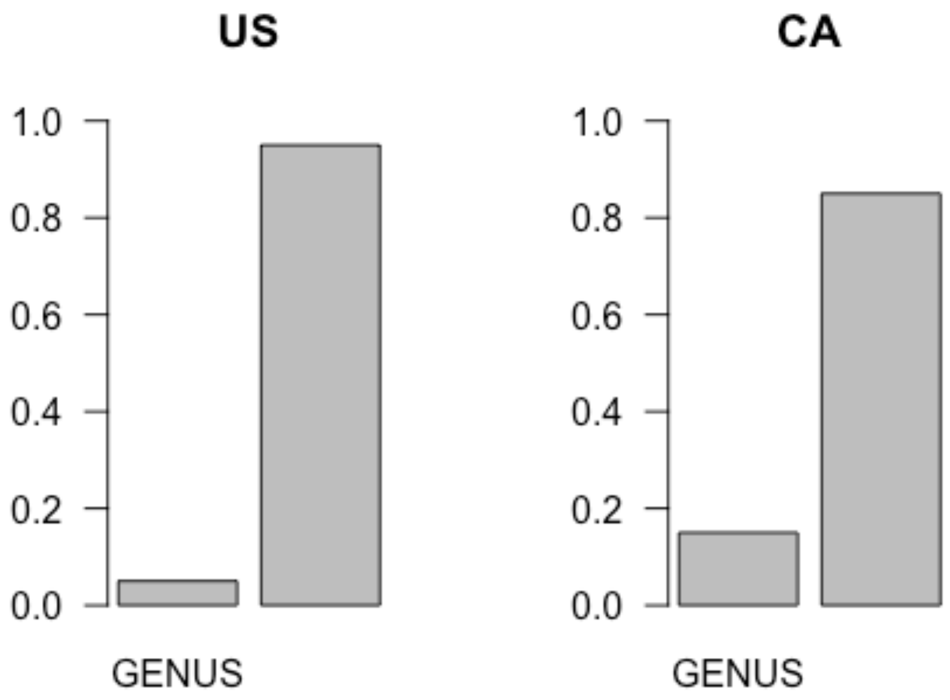
However, we have not downloaded the information yet. We did a test search to obtain the number of records available, that is why we set the limit = 20. (i.e. download only 20 records) The reason that we can not download everything at once is to prevent the API to collapse because of many simultaneous calls from around the world. So, the people at GBIF have set a download limit of 300 for each API call. In other words, we can only download 300 occurrences at each call of the `occ_data()` function.

Now let's see all the variables we can obtain from a gbif search

```
names(testSearch[,1]$data)

##  [1] "key"
##  [2] "scientificName"
##  [3] "decimalLatitude"
##  [4] "decimalLongitude"
##  [5] "issues"
##  [6] "datasetKey"
##  [7] "publishingOrgKey"
##  [8] "installationKey"
##  [9] "publishingCountry"
## [10] "protocol"
## [11] "lastCrawled"
## [12] "lastParsed"
## [13] "crawlId"
## [14] "basisOfRecord"
## [15] "taxonKey"
## [16] "kingdomKey"
## [17] "phylumKey"
## [18] "classKey"
## [19] "orderKey"
## [20] "familyKey"
## [21] "genusKey"
## [22] "speciesKey"
## [23] "acceptedTaxonKey"
## [24] "acceptedScientificName"
## [25] "kingdom"
## [26] "phylum"
## [27] "order"
## [28] "family"
## [29] "genus"
## [30] "species"
## [31] "genericName"
## [32] "specificEpithet"
## [33] "taxonRank"
## [34] "taxonomicStatus"
## [35] "dateIdentified"
## [36] "stateProvince"
## [37] "year"
## [38] "month"
## [39] "day"
## [40] "eventDate"
## [41] "modified"
## [42] "lastInterpreted"
## [43] "references"
## [44] "license"
## [45] "geodeticDatum"
## [46] "class"
## [47] "countryCode"
## [48] "country"
## [49] "rightsHolder"
## [50] "identifier"
## [51] "http://unknown.org/nick"
## [52] "verbatimEventDate"
## [53] "datasetName"
## [54] "collectionCode"
## [55] "verbatimLocality"
## [56] "gbifID"
## [57] "occurrenceID"
## [58] "taxonID"
## [59] "recordedBy"
## [60] "catalogNumber"
## [61] "http://unknown.org/occurrenceDetails"
## [62] "institutionCode"
## [63] "rights"
## [64] "eventTime"
## [65] "identificationID"
## [66] "coordinateUncertaintyInMeters"
## [67] "informationWithheld"
## [68] "occurrenceRemarks"
## [69] "name"
```

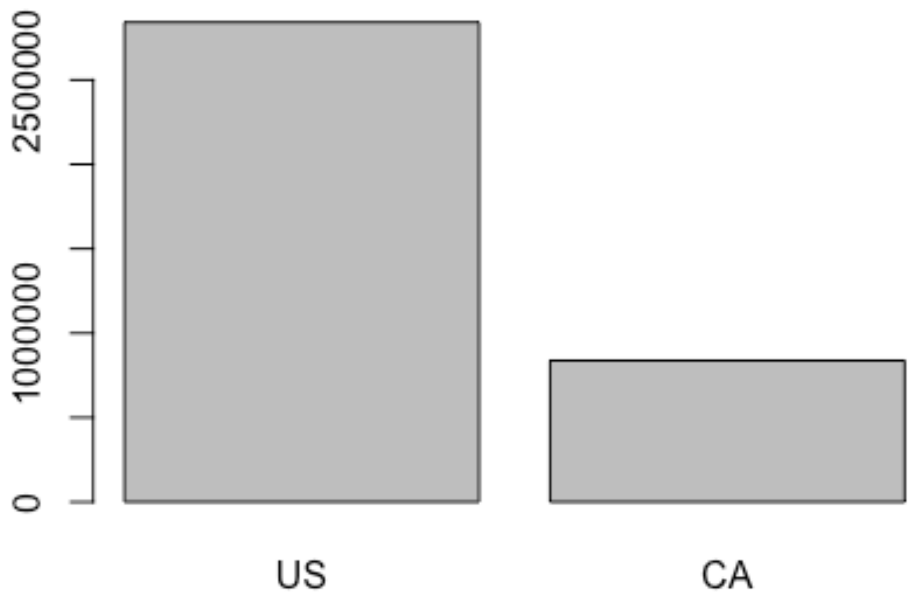
That is a lot of information classes available for a particular occurrence observation. However, we should take in account that the GBIF data is heterogeneous. That is it comes from many sources. The staff at GBIF work hard to standardize all this data, but it that is still a limitation. For example, let's see from our 20 sample records how many of those occurrences correspond to species level information

```
par(mfrow = c(1,2), las = 1)
barplot(prop.table(
  table(testSearch[,1]$data$taxonRank)),
  main = "US",
  ylim = c(0,1)
  )
barplot(prop.table(
  table(testSearch[,2]$data$taxonRank)),
  main = "CA",
  ylim = c(0,1)
  )
```



OK, since we are now familiar with the results we can obtain from each search call, lets expand this to get more than only 20 occurrences. Remember that the API only accepts 300 occurrences per call. *see* `?occ_data` for more details. You should repair in the "start" and "hasCoordinate" parameters.

remember there is different amount of data available for each country



Let's define the starters for each call

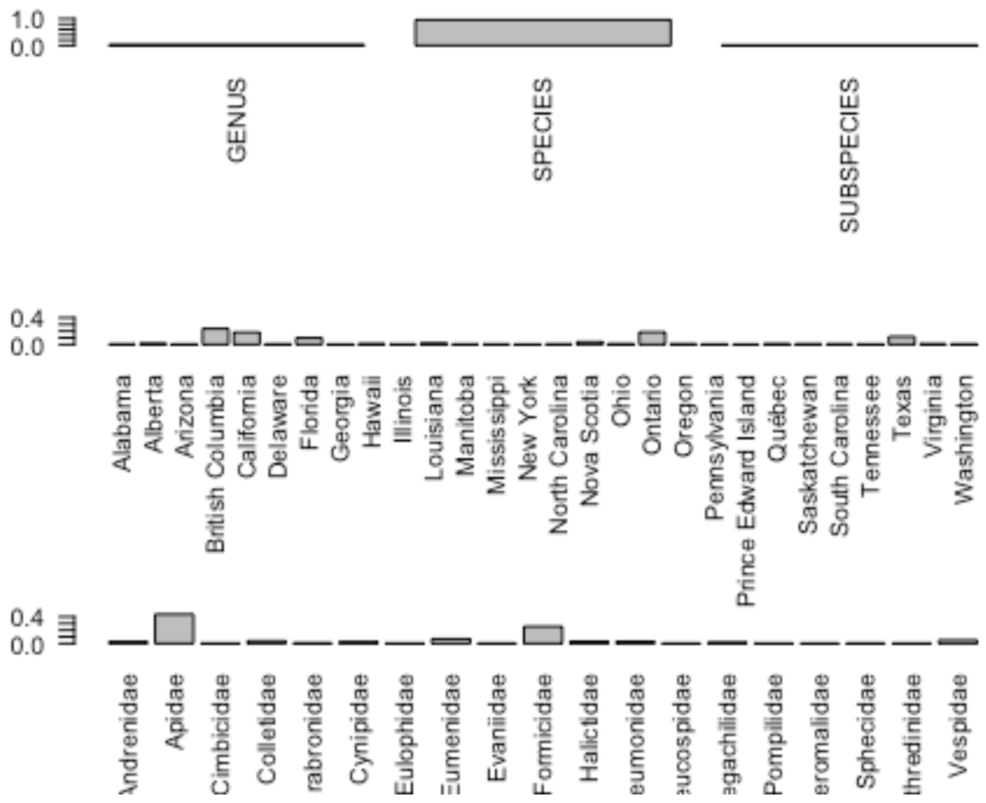
```
testSearch[,2]$meta$count

## [1] 837910

starters <- sample(1:testSearch[,2]$meta$count, 2) ## Modify this line
to change the total "occurrence pages" downloaded
HymCA_US <- c()

for(i in 1:length(starters)){
  print(paste("Starting with query", i, "of 100")) # print the loop
status
  HymCA_US[[i]] <- customSearGBIF(query = query,
                        limit = 300,
                        list("start" = starters[i],
                             "hasCoordinate" = T) # listing the extra
arguments for occ_data function
         )

Sys.sleep(15) # To give a rest of 15 secs between each call
cat("\014") ## clean the console
  }

## [1] "Starting with query 1 of 100"
## [1] "Starting with query 2 of 100"
##

# examine the extructure of the information  US = [,1], Canada = [,2]

HymCA_US[[1]]

##      [,1]    [,2]
## meta List,4  List,4
## data List,72 List,74
```

Let's know do some housekeeping in the data we retrieved

```
## Subset data for US
HymUS <- lapply(1:length(HymCA_US), function(x)
data.frame(HymCA_US[[x]][,1]$data))
HymUS <- dplyr::bind_rows(HymUS)

## Subset data for Canada
HymCA <- lapply(1:length(HymCA_US), function(x)
data.frame(HymCA_US[[x]][,2]$data))
HymCA <- dplyr::bind_rows(HymCA)



## Bind both datasets
HymBoth <- dplyr::bind_rows(list(HymUS, HymCA))
```

Good! Now we have our dataset combined let's get an overview of the information collected A frequency plot the taxonomic resolution of our dataset can help

```
par(mfrow = c(3,1), las = 2)

barplot(prop.table(table(HymBoth$taxonRank)), ylim = c(0,1))
barplot(prop.table(table(HymBoth$stateProvince)), ylim = c(0,0.4), las
= 2)
barplot(prop.table(table(HymBoth$family)), ylim = c(0,0.4), las = 2)
```



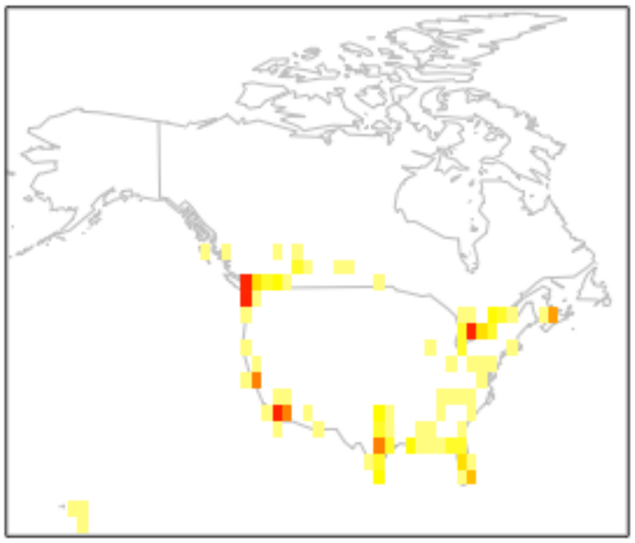
We use the `mapplots` package to make a grid along the range

```
grid <- mapplots::make.grid(HymBoth$decimalLongitude,
                  HymBoth$decimalLatitude,
                  rep(1, length(HymBoth$scientificName)),
                  byx = 2, # degrees
                  byy =  2, # degrees
                  xlim = range(HymBoth$decimalLongitude),
                  ylim = range(HymBoth$decimalLatitude))
```

We use that grid to plot the distribution of available information

```
##
maps::map(col = "grey",
          regions = c("USA", "Canada"),
          xlim = c(-170,-50))
title("Hymenoptera first 60000 occurrences per 2x2 grid")
draw.grid(grid, col = heat.colors(12,1, rev = T))
```

Hymenoptera first 60000 occurrences per 2x2 grid



Depending of the amount of data recollected the last operation might have taken some time. To avoid running this lengthy process each time, we can store the object directly as a file in our working directory. The file extension for R objects is .RDS

```
## Save the object
#saveRDS(HymBoth, "HymCA_US.RDS") # save the object
```

I highly encourage you to explore more the `rgbif` package.

Biogeographic regionalization

Uploading the species occurrence data

For this section we will use previously downloaded datasets. We use a dataset of herb species occurrence data across the United States. It is a dataset of the abundance of herbaceous species across ~15,000 plots across the US, along with related site date (e.g. climate, soil pH, etc.).

You can access the data here

Alternatively, you can ask a copy of the dataset to any of the presenters.

Ok, if you have the dataset you can upload with `read.csv`

```
herb <-
read.csv("Data_gram_forb_abund_and_site_attr_w_no_grad_or_soil_filter_3
1Mar2016_both_N_and_S.csv")
```

Let’s observe the name of each of the columns (variables) included in this dataset

```
names(herb)
```

```
##  [1] "proj_site_orig"          "species"
##  [3] "Category"                "Family"
##  [5] "Federal_Noxious_Status"  "Invasive"
##  [7] "Federal_T_E_Status"      "Habit_simple"
##  [9] "Duration_simple"         "Native_Status_simple"
## [11] "Invasive_introduced"     "cover"
## [13] "proj"                    "site"
## [15] "proj_site"               "proj_orig"
## [17] "totl_spp"                "totl_divN1"
## [19] "totl_divN2"              "annual_spp"
## [21] "annual_divN1"            "annual_divN2"
## [23] "per_spp"                 "per_divN1"
## [25] "per_divN2"               "invasintro_spp"
## [27] "invasintro_divN1"        "invasintro_divN2"
## [29] "intro_spp"               "intro_divN1"
## [31] "intro_divN2"             "native_spp"
## [33] "native_divN1"            "native_divN2"
## [35] "gram_spp"                "gram_divN1"
## [37] "gram_divN2"              "forb_spp"
## [39] "forb_divN1"              "forb_divN2"
## [41] "intgrm_spp"              "intgrm_divN1"
## [43] "intgrm_divN2"            "natfor_spp"
## [45] "natfor_divN1"            "natfor_divN2"
## [47] "area_m2"                 "totl_SD"
## [49] "annual_SD"               "per_SD"
## [51] "intro_SD"                "native_SD"
## [53] "gram_SD"                 "forb_SD"
## [55] "cov_tot"                 "cov_annual"
## [57] "cov_perennial"           "cov_invas_intro"
## [59] "cov_intro"               "cov_native"
## [61] "cov_gram"                "cov_forb"
## [63] "tot_spp_incid"           "field_subplots"
## [65] "inext_subplots"          "raw_spp_coverage"
## [67] "inext_spp_coverage"      "inext_cov90_divN0"
## [69] "inext_cov90_divN1"       "inext_cov90_divN2"
## [71] "CLASSIF_CODE"            "NVC_1"
## [73] "NVC_1_name"              "NVC_2"
## [75] "NVC_2_name"              "NVC_3"
## [77] "NVC_3_name"              "Nat_or_Cult"
## [79] "NVC_HIGHER"              "NVC_ALLIAN"
## [81] "NVC_ASSOCI"              "veg_other"
## [83] "veg_mixed"               "NA_L3CODE"
##  [85] "NA_L3NAME"                     "NA_L2CODE"
##  [87] "NA_L2NAME"                     "NA_L1CODE"
##  [89] "NA_L1NAME"                     "NA_L3KEY"
##  [91] "NA_L2KEY"                      "NA_L1KEY"
##  [93] "NLCD_2001_name"                "forcov1991_name"
##  [95] "NVC_1_name_equiv"              "NVC_2_name_equiv"
##  [97] "NVC_veg_1_2"                   "NVC_veg_1_2_some_from_NLCD"
##  [99] "latitude"                      "longitude"
## [101] "year"                          "elev_m"
## [103] "slope_deg"                     "aspect_deg"
## [105] "N_10dr27wt"                    "N_TDEP0012"
## [107] "N_CMAQ0211tot"                 "S_CMAQ0211tot"
## [109] "Elev_m_NED"                    "precip_mm"
## [111] "temp_C_ave"                    "temp_C_max"
## [113] "temp_C_min"                    "GEOLOGY"
## [115] "HDEN00"                        "AGBUR"
## [117] "hzname"                        "CEC"
## [119] "cec7_r"                        "ecec_r"
## [121] "pH"                            "ph1to1h2o_r"
## [123] "ph01mcacl2_r"                  "pH_source"
## [125] "aws0_5_mm"                     "soc0_5_gm2"
## [127] "rootzone_cm"                   "pot_wetl_per"
## [129] "Ca_ppm"                        "Mg_ppm"
## [131] "K_ppm"                         "Na_ppm"
## [133] "Al_ppm"                        "P_ppm"
## [135] "N_pct"                         "S_10dr27wt"
```

However, for now we will only use the species level name, and its geographic coordinates (i.e. latitude and longitude). Let's subset the dataset

```
herb <- herb[c("species", "latitude","longitude")]
```

Identyfing natural areas

Since our study area is big (almost all of the North American continent) it become obvious that the ROI is not geographically homogeneous. It is a good practice to get familiarized with the abiotic conditions and gradients along the ROI that might influence our results. Also it could help us to formulate adequate spatially-explicit hypothesis.

Let's explore the main abiotic gradients in the EU and Canada.

A great source for global scale geo-physical datasets is the Earth Engine Google data catalog

https://developers.google.com/earth-engine/datasets

Short questions for discussion

- Which abiotic conditions can afect our results?

- Which data can be useful for our purposes?
- What are the limitations of using one/other datasets?
- Do you know about other data sources at global scales?

Quantifying community level dissimilarities

This section corresponds to: (4,5) of Morrone framework and (4,5,6) of Kreft framework

We will now create an species by site matrix at the grid level

Each grid becomes an individual community

The first step is to aggregate the distribution of coordinates into latitude longitude intervals so the resulting grid has a resolution of aprox 1x1 degree.

```
#

Ncut <- round(diff(range(herb$latitude)))/2

herb <- tibble::add_column(herb,
                          "interval" =  paste(cut(herb$latitude, Ncut),
                                              cut(herb$longitude,
Ncut)))


head(herb)

##                    species latitude longitude              interval
## 1 Ambrosia artemisiifolia 46.07556 -83.65944 (45,47] (-88.1,-82.9]
## 2     Hieracium pilosella 46.07556 -83.65944 (45,47] (-88.1,-82.9]
## 3       Oligoneuron album 46.07556 -83.65944 (45,47] (-88.1,-82.9]
## 4  Campanula rotundifolia 46.07556 -83.65944 (45,47] (-88.1,-82.9]
## 5           Poa compressa 46.07556 -83.65944 (45,47] (-88.1,-82.9]
## 6         Solidago juncea 46.07556 -83.65944 (45,47] (-88.1,-82.9]
```

Let’s now make the species by site matrix

```
specTab <-  table(herb$interval,herb$species)
specTab[specTab > 1] <- 1 # turn into presence absence matrix
```

A short reminder about beta-diversity here

We will use the Simpson dissimilarity index (Simpson, 1943, 1960 ;Baselga 2010), because it works better for the biogeographic regionalization purposes.

$$S_i = \frac{a}{a + min(b,c)}$$

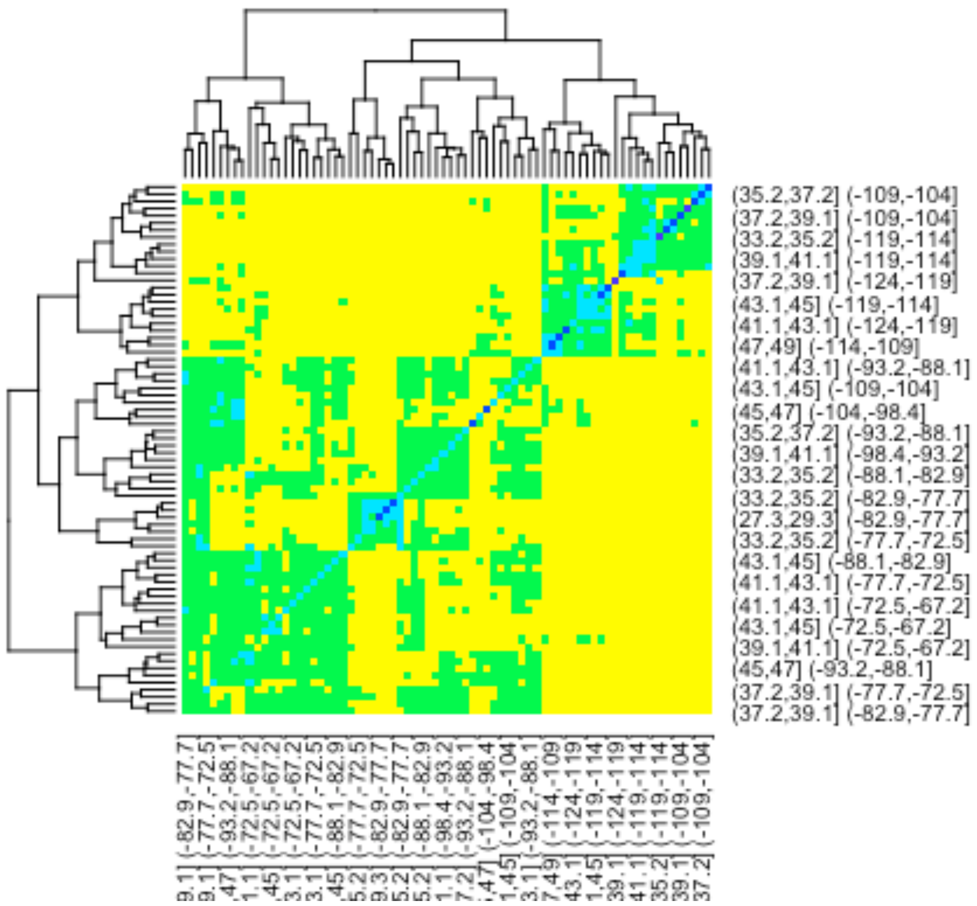
This index can be calculated with the framework of Baselga. 2010 which portions the nestedness-resultant component of dissmilarity of the Sorensen dissimilarity index. In other words, Simpson index is the turnover component of the Sorensen index.

```
# Calcualate "simpson dissimilarity index" (Bsim part of sorensen
dissimilarity)
part <- betapart::beta.pair(specTab, "sorensen")
summary(part$beta.sim) ##  access the portion of the object that
correspond to the Simpson index of dissmilarity

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.5686  0.8261  0.7395  0.9487  1.0000

## Alternatively you can use the "vegan" package to calculate the same
index.

# vegan::betadiver(matrix, "sim")
```

Visualizing turnover (B-diversity)

The `heatmap` function is a great way to visualize the matrix

```
heatmap(as.matrix(part$beta.sim), col = topo.colors(5)) # play with
different color schemes and palette gradients!
```



Short questions

- What inferences can be done from a quick look at this matrix?
- Is there a clear pattern?

Now lets plot the dissimilarity distances in a map

let's use our first grid in the matrix as focal point

```
fp <- as.matrix(part$beta.sim)[,1]
```

Now lets make a new dataframe gathering all the information together

```
map <- data.frame("lat" =
herb$latitude[match(names(fp),herb$interval)])
map$lon <- herb$longitude[match(names(fp),herb$interval)]
map$fp <- fp
```

We need now to define a color pallete that assing colorbins to a distribution of continous data

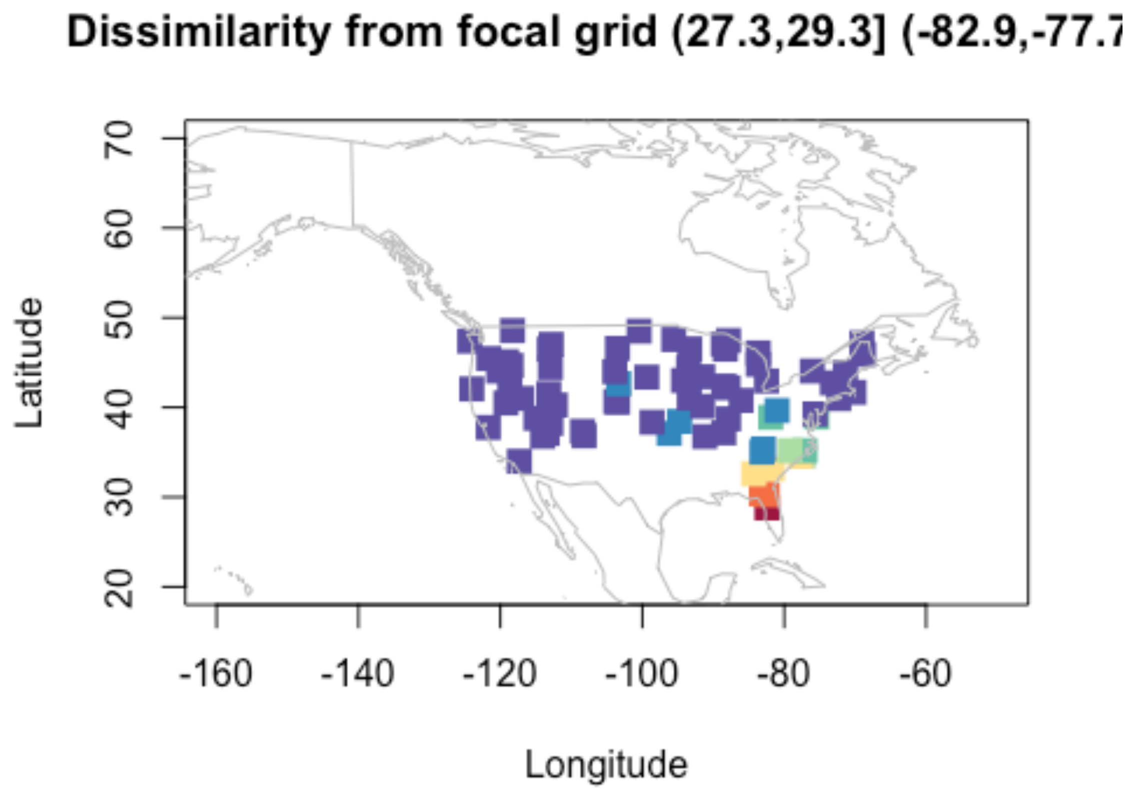
```
f <- function(x,n=10, pal, rev = F){
  if(rev == F){
    rev(RColorBrewer::brewer.pal(n, pal))[cut(x,n)]
  }else{
      (RColorBrewer::brewer.pal(n, pal))[cut(x,n)]
    }
}
```

Using that function, define the color vector for the plot

```
colPlot <- f(map$fp, 10, "Spectral", rev = T)
```

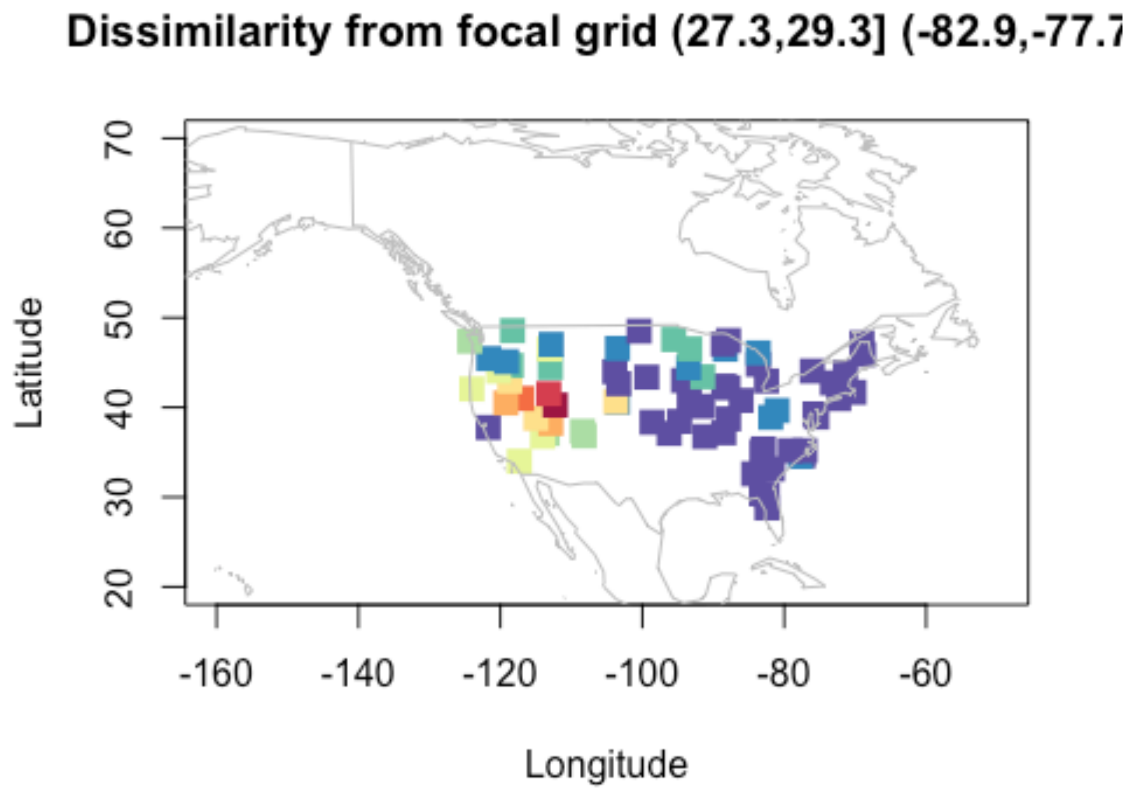
Now let's plot the result

```
# define color vector for the legend
# toLeg <- data.frame(unique(cbind(round(map$fp, 3),colPlot)))
# toLeg <- toLeg[order(toLeg$V1),]

plot(map$lat~ map$lon,
     pch = 15,
     cex = 1.5,
     ylim = c(20,70),
     xlim = c(-160, -50),
     xlab = "Longitude",
     ylab = "Latitude",
     main = paste("Dissimilarity from focal grid",
                  names(fp)[1]),
     col = colPlot)

maps::map(col = "grey", add = T,
          regions = c("USA", "Canada", "MEX","Cuba"))
```



How does it look like from a distinct perspective ?

```r
fp2 <- as.matrix(part$beta.sim)[,30] # Modify the number here to play
around
colPlot2 <- f(fp2, 10, "Spectral", rev = T)

plot(map$lat~ map$lon,
     pch = 15,
     cex =1.5,
     ylim = c(20,70),
     xlim = c(-160, -50),
     xlab = "Longitude",
     ylab = "Latitude",
     main = paste("Dissimilarity from focal grid",
                  names(fp2)[1]),
     col = colPlot2)

maps::map(col = "grey", add = T,
          regions = c("USA", "Canada", "MEX","Cuba"))
```



Ordination

This section corresponds to: (5,6) of Morrone framework and (6) of Kreft framework

Ordination are a set of statistical techniques used principally to reduce the dimensionality of a matrix. In our case we can think about sites and species as dimensions. Using this vector we can construct the most parsimonious organization in 2d (or more) space.

We will use a technique called Non-Metric Multidimensional (NMDS) scaling to find the most parsimonious arragement of our distance-matrix items that preserve the community wide pattern of pairwise distances between species and sites.

We will not to go into details of the procedure, but if you need a reminder of ordination methods or the mathematical basis of NMDS you can find information in the following links:

Ordination:

http://ordination.okstate.edu/overview.htm

https://wiki.qcbs.ca/r_workshop9

NMDS:

http://www.flutterbys.com.au/stats/tut/tut15.1.html

http://cc.oulu.fi/~jarioksa/opetus/metodi/vegantutor.pdf

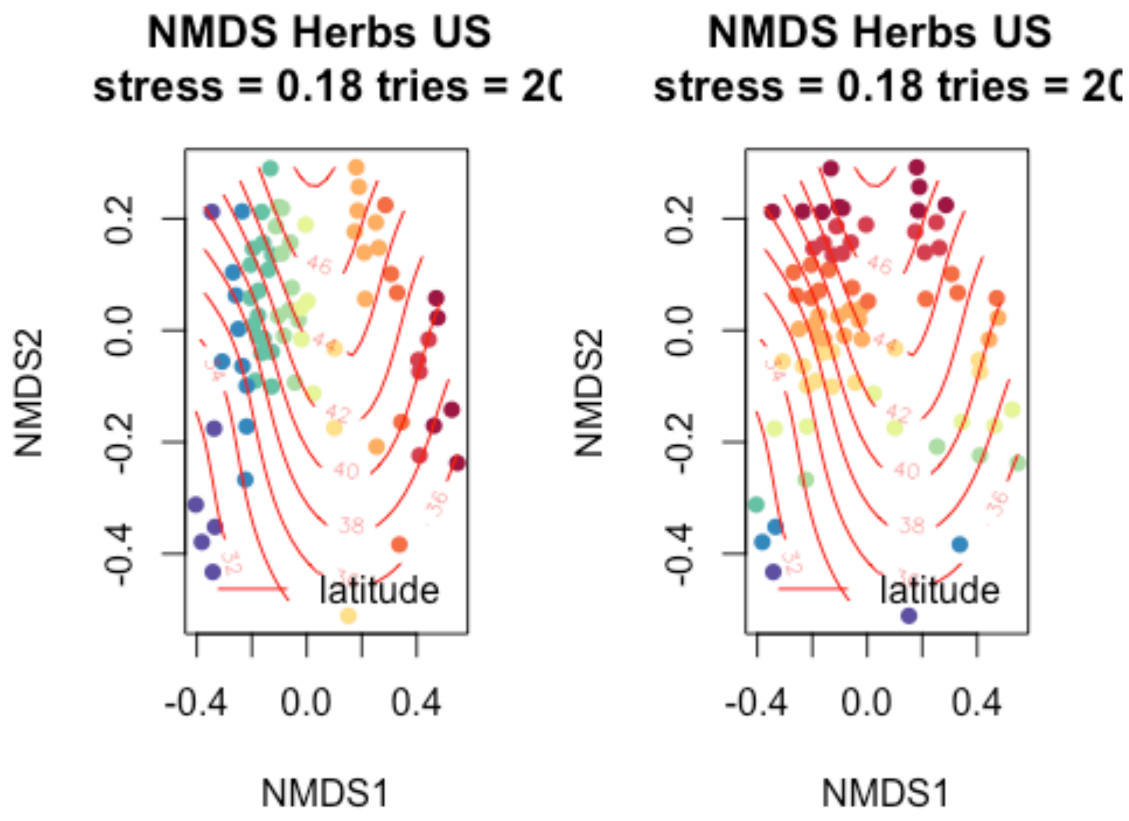
Calculate non-metric multidimensional scaling using the intervals as sites

```
##
metaMDS <- vegan::metaMDS(part$beta.sim) ## remenber we are using the
Simpson disimilarity matrix here as imput

## Run 0 stress 0.1800427
## Run 1 stress 0.1859481
## Run 2 stress 0.1800345
## ... New best solution
## ... Procrustes: rmse 0.002170752  max resid 0.01303691
## Run 3 stress 0.1800436
## ... Procrustes: rmse 0.002113138  max resid 0.01240107
## Run 4 stress 0.1859689
## Run 5 stress 0.1859487
## Run 6 stress 0.1859426
## Run 7 stress 0.1800317
## ... New best solution
## ... Procrustes: rmse 0.0008525444  max resid 0.004944575
## ... Similar to previous best
## Run 8 stress 0.1859995
## Run 9 stress 0.1859404
## Run 10 stress 0.185941
## Run 11 stress 0.1800572
## ... Procrustes: rmse 0.002523548  max resid 0.01290599
## Run 12 stress 0.1800414
## ... Procrustes: rmse 0.002037973  max resid 0.01319688
## Run 13 stress 0.2013313
## Run 14 stress 0.1859409
## Run 15 stress 0.1800522
## ... Procrustes: rmse 0.001996166  max resid 0.01171975
## Run 16 stress 0.1800316
## ... New best solution
## ... Procrustes: rmse 0.0001202317  max resid 0.0003876892
## ... Similar to previous best
## Run 17 stress 0.1800387
## ... Procrustes: rmse 0.001246704  max resid 0.006079061
## ... Similar to previous best
## Run 18 stress 0.1800413
## ... Procrustes: rmse 0.002029987  max resid 0.01308065
## Run 19 stress 0.1800412
## ... Procrustes: rmse 0.002028671  max resid 0.01298999
## Run 20 stress 0.2012241
## *** Solution reached
```

Let’s now plot the results. Since we are interested into projecting this NMDS into geographic space we can use the same function we previously definend above `f` to summarize the variation of each NMDS axis as a graduated color pallete.

We can also fit a surface showing the variation of latitude across our gradient.

```
par(mfrow = c(1,2))
plot(vegan::scores(metaMDS),
     col = f(vegan::scores(metaMDS)[,1],n = 10,pal = "Spectral"),
     pch = 16,
     main = paste("NMDS Herbs US","\n",
                  "stress =", round(metaMDS$stress,4),
                  "tries =", metaMDS$tries))
vegan::ordisurf(metaMDS~map$lat, add = T)

##
## Family: gaussian
## Link function: identity
##
## Formula:
## y ~ s(x1, x2, k = 10, bs = "tp", fx = FALSE)
##
## Estimated degrees of freedom:
## 7.95  total = 8.95
##
## REML score: 159.706

legend("bottomleft","latitude", lty =1, col = "red",bty = "n")

plot(vegan::scores(metaMDS),
     col = f(vegan::scores(metaMDS)[,2],n = 10,pal = "Spectral"),
     pch = 16,
     main = paste("NMDS Herbs US","\n",
                  "stress =", round(metaMDS$stress,4),
                  "tries =", metaMDS$tries))
vegan::ordisurf(metaMDS~map$lat, add = T)

##
## Family: gaussian
## Link function: identity
##
## Formula:
## y ~ s(x1, x2, k = 10, bs = "tp", fx = FALSE)
##
## Estimated degrees of freedom:
## 7.95  total = 8.95
##
## REML score: 159.706

legend("bottomleft","latitude", lty =1, col = "red",bty = "n")
```



Mapping the NMDS

Ok, now lets project the NMDS axis to the grid that we previosly created We have to get the scores of the nmds and add it to the dataframe with the interval info (i.e. grids)

```
scores <- data.frame(cbind(rownames(specTab),
                           vegan::scores(metaMDS, "sites")),
                     stringsAsFactors = F) # Note I'm avoiding the
creation of factor type data
```

Doing some housekeeping in this new dataframe

```
## Lets transform the data to numeric!
str(scores)

## 'data.frame':    74 obs. of  3 variables:
##  $ V1   : chr  "(27.3,29.3] (-82.9,-77.7]" "(29.3,31.2] (-82.9,-
77.7]" "(29.3,31.2] (-88.1,-82.9]" "(31.2,33.2] (-82.9,-77.7]" ...
##  $ NMDS1: chr  "-0.341977535438759" "-0.381817071554515" "-
0.403845344265662" "-0.223767075517883" ...
##  $ NMDS2: chr  "-0.432619987176844" "-0.379027607698212" "-
0.312054765938254" "-0.267294485123869" ...
```

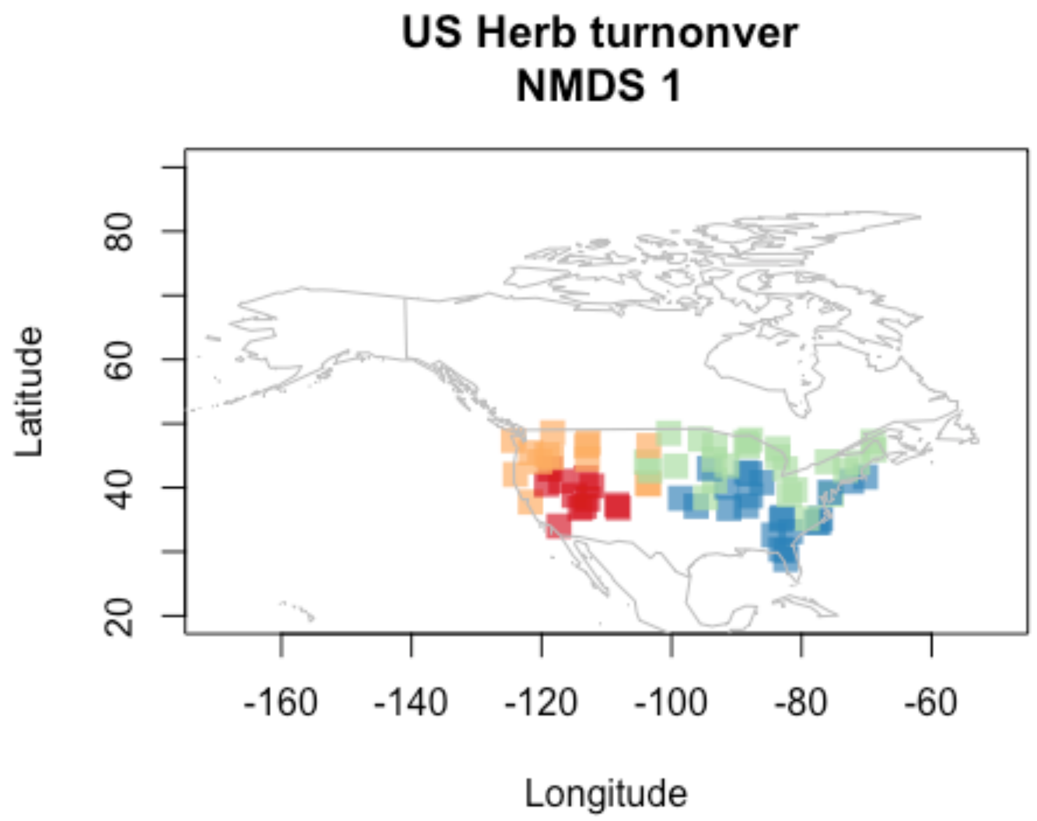
```
names(scores)

## [1] "V1"    "NMDS1" "NMDS2"

scores[,c(2,3)] <- sapply(scores[,c(2,3)], function(x) as.numeric(x))
str(scores) # (y)

## 'data.frame':    74 obs. of  3 variables:
##  $ V1   : chr  "(27.3,29.3] (-82.9,-77.7]" "(29.3,31.2] (-82.9,-
77.7]" "(29.3,31.2] (-88.1,-82.9]" "(31.2,33.2] (-82.9,-77.7]" ...
##  $ NMDS1: num  -0.342 -0.382 -0.404 -0.224 -0.334 ...
##  $ NMDS2: num  -0.433 -0.379 -0.312 -0.267 -0.352 ...

## Add the lat and long of each interval

scores$lat <- herb$latitude[match(scores$V1,herb$interval)]
scores$lon <- herb$longitude[match(scores$V1,herb$interval)]
```

Then we proceed to define the color vectors

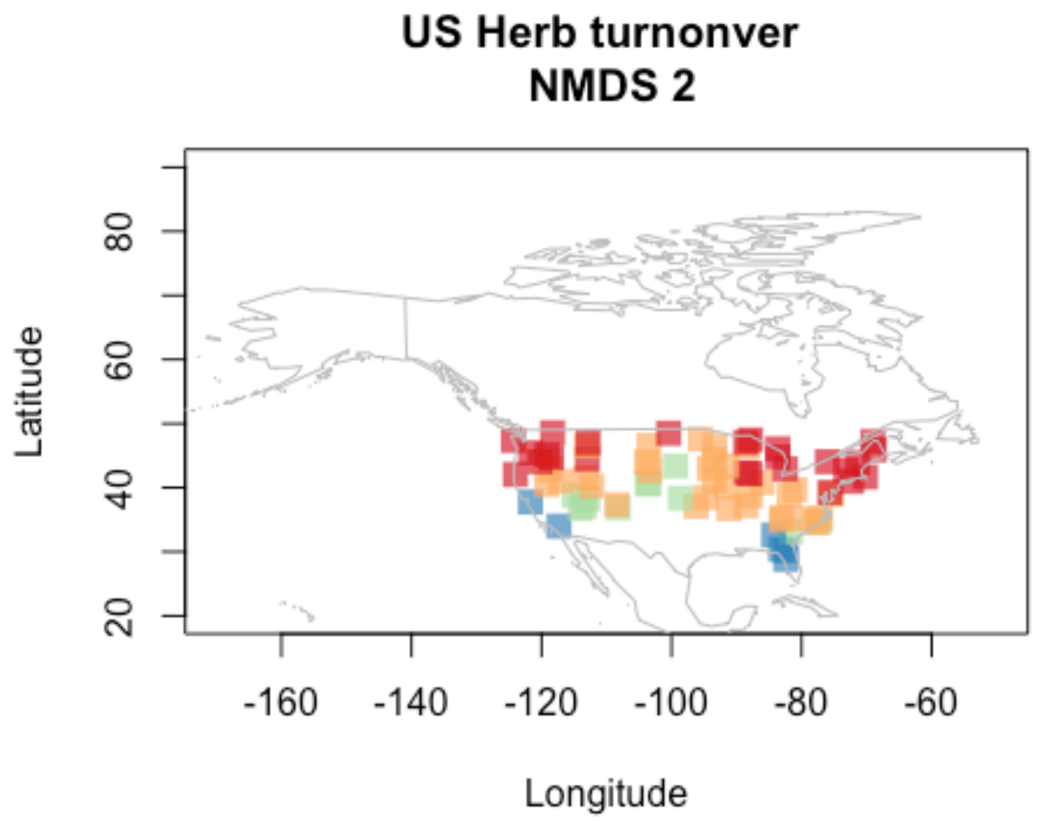
```
# color point vector
col1 <- f(scores$NMDS1, 4, "Spectral", rev = F)
col2 <- f(scores$NMDS2, 4, "Spectral", rev = F)

# color legend vector
# toLeg <- unique(cbind(round(scores$NMDS1, 3),col1))
# toLeg2 <- unique(cbind(round(scores$NMDS2, 3),col2))
```

Let’s plot the map now using base R functions and the `map` R package.

```
## First axis
plot(scores$lat ~scores$lon,
     pch = 15,
     xlab = "Longitude",
     ylab = "Latitude",
     cex =1.5,
     main = "US Herb turnonver \n NMDS 1  ",
     col = scales::alpha(col1,0.7),
     xlim = c(-170,-50),
          ylim  = c(20,90)
     )

# legend("bottomleft",
#         fill =  toLeg[,2] ,
#         legend = toLeg[,1])
 maps::map(col = "grey", add = T,
          regions = c("USA", "Canada", "MEX", "Cuba")
          )
```



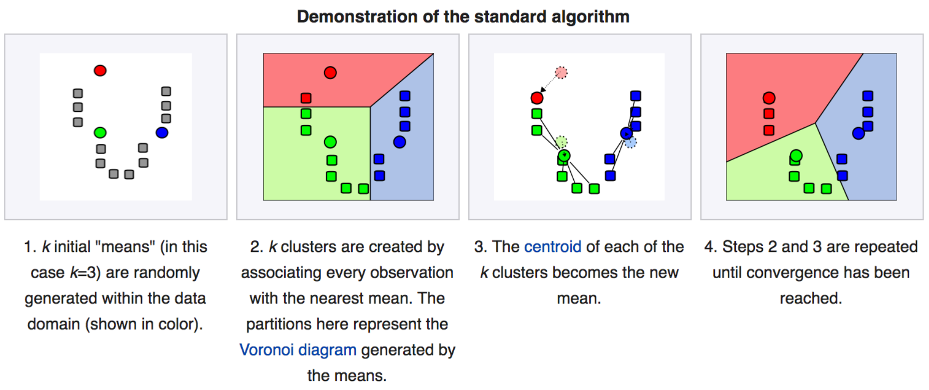
```
## Second axis
plot(scores$lat~ scores$lon,
     pch = 15,
     cex =1.5,
     xlab = "Longitude",
     ylab = "Latitude",
     main = "US Herb turnonver \n NMDS 2",
      xlim = c(-170,-50),
          ylim  = c(20,90),
     col = scales::alpha(f(scores$NMDS2,
             4,
             "Spectral",
             rev = F),0.7)
     )
# legend("bottomleft",
#        fill =  toLeg2[,2] ,
#        legend = toLeg2[,1])
maps::map(col = "grey", add = T,
          regions = c("USA", "Canada", "MEX", "Cuba"))
```



Defining biogeographic regions

k-means is a popular and fast clustering algorithm. The algorithm search to partition the a number of data points into *k* clusters.

This is common methodology for *un-supervised" classfication tasks. That is, we let the data to tell us the best partition into separate classes.


Demonstration of the standard algorithm
1. *k* initial "means" (in this case *k*=3) are randomly generated within the data domain (shown in color).
2. *k* clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.
3. The centroid of each of the *k* clusters becomes the new mean.
4. Steps 2 and 3 are repeated until convergence has been reached.

steps for the K-means algorithm (from wikipedia)

As you can observe, k-mean algorithm uses euclidean distances to create the clusters. However, this distance is almost meaningless in community data.

The **Hellinger** standarization is a workaround to use euclidean metrics with community data.

$$y_{i,j} = \sqrt{\frac{y_{i,j}}{y_i}}$$

Remember that the objective of biogeographic regionalization is to find a way to split our spatial dataset into defined regions.

You can easily spot that a big limitation of the k-mean algorithm, the k-means algorithm is dependant on the initial *k*. So basically, we want to **optimize** the value of *k*. How do we do this?

The answer is to interate over the whole process and find the solution that maximises the variation within each cluster. In other words (or rather statistical jargon), we want that the Sum of squares of the distances between clusters (i.e. the distances of the k-centroids to its mean centroid) approximates the total sum of squares. The ratio between this two will give you the total amount of variation represented in each k-partition.

The function `cascadeKM` from the `vegan` package does exactly this process, or we can write a loop over the clasic `kmeans` base function.

Let's transform our species site matrix

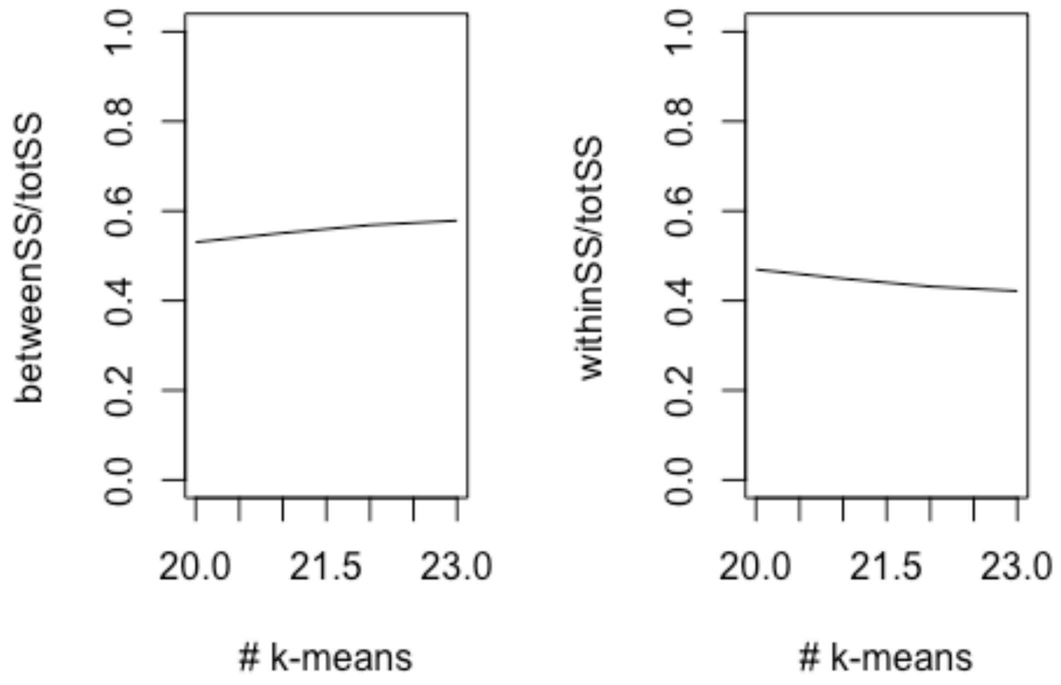
```
transMatHel<- vegan::decostand(specTab, "hell")
```

Now let's first run the `kmeans`function in a loop in order to optimize its solution

```
kmax <-  23# maximum value of k
kmin <- 20


bt<-c()
wt<- c()
for(i in kmin:kmax){
  print(i)
  k <- kmeans(as.matrix.data.frame(transMatHel),i)

  bt[i] <- (k$betweenss/k$totss)
  wt[i] <- k$tot.withinss/k$totss
  cat("\014")
}

## [1] 20
## [1] 21
## [1] 22
## [1] 23
##
```

Let's pull together the results of the loop into a dataframe and plot the results

```
ktry <- data.frame("k" = kmin:kmax,
          "bt" = na.omit(bt),
          "wt" = na.omit(wt))


par(mfrow = c(1,2))

plot(ktry$bt~ktry$k,
     ylab = "betweenSS/totSS",
     xlab = "# k-means",
     type = "l",
     xlim = c(kmin, kmax),
     ylim = c(0,1))



plot(ktry$wt~ktry$k,
     ylab = "withinSS/totSS",
     xlab = "# k-means",
     type = "l",
     xlim = c(kmin, kmax),
     ylim = c(0,1))
```



Alternatively, using vegan.

Remember to use the hellinger transformed species-site matrix

```
##

optimKM <- vegan::cascadeKM(as.matrix.data.frame(transMatHel),20,30)
```

Let’s settle into k=25 for now. You can try other k numbers as well. Or increase the range in the loop.

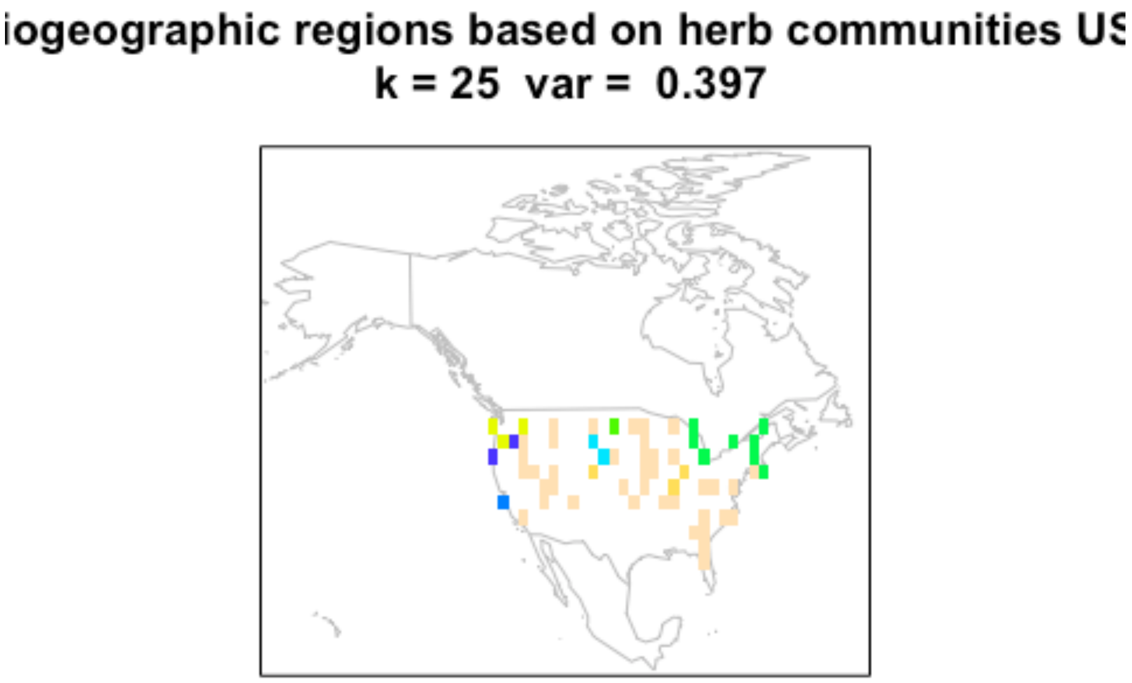
```
# seems the optima
k <- kmeans(as.matrix.data.frame(transMatHel),25)

scores$k <- k$cluster ## add to our previous dataframe
```

Let’s aggregate into a 2x2 grid.

```
gridMDS <- mapplots::make.grid(scores$lon,
                     scores$lat,
                     scores$k,
                     fun = mean,
                     byx = 2,
                     byy = 2,
                     ylim = range(scores$lat),
                     xlim = range(scores$lon))
```

Let’s now plot the map.

```
maps::map(col = "grey",
regions = c("USA","CA", "MEX"),
xlim = c(-170,-50))
title(paste("Biogeographic regions based on herb communities US. \n k =
25", " var = ",
            round(k$tot.withinss/k$totss,3)))
mapplots::draw.grid(gridMDS,
                    breaks = c(0:10) ,
                    col = topo.colors(10))
```



```
 #library(shapefiles)
# mapplots::write.grid(gridMDS, "grid", type = "csv")
```

Hierarchical regionalization

Ok, we have now define some finite geographic regions based on the species occurrences of our dataset. However, the question remains on what are the relationships of such regions with each other?

How we can best summarize the variation into smaller discrete units?

Let’s try to arrange our variation into units in a hierarchical way.

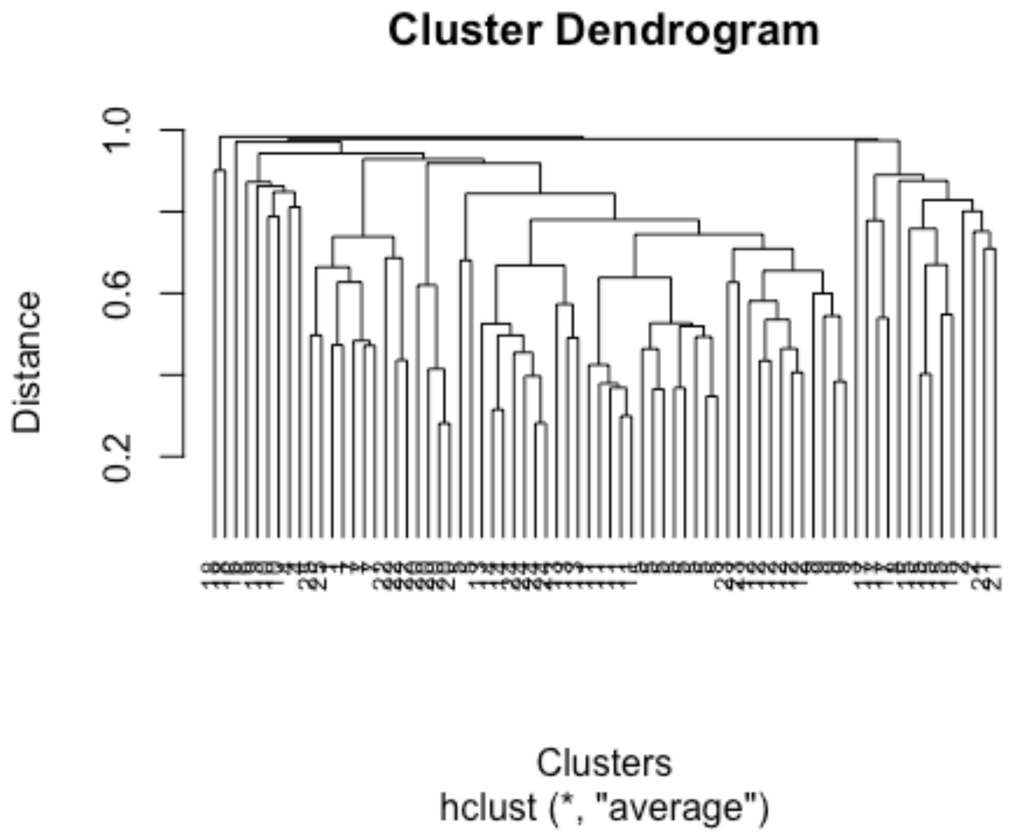
We will use the `hclust`function to implement an hierarchical clustering algorithm, setting the parameter `method = average` tells the function to use the Unweighted Pair Group Method with Arithmatic Mean (UPGMA) as a clustering method.

See the help page for this function for more details `?hclust`.

```
clus <- hclust(vegan::vegdist(transMatHel), method = "average")
```

After running the algorithm, let’s plot the dendrogram and relate the tips of this tree with the cluster affiliation found with the k-means algorithm

```
plot(clus, hang = -1,
     cex = 0.7,
     xlab = "Clusters",
     ylab = "Distance",
     labels = k$cluster)
```



We can use that information to further classify our regions in a hierarchical way. Usually this step involves discussions among specialists. So we will stop there for now.

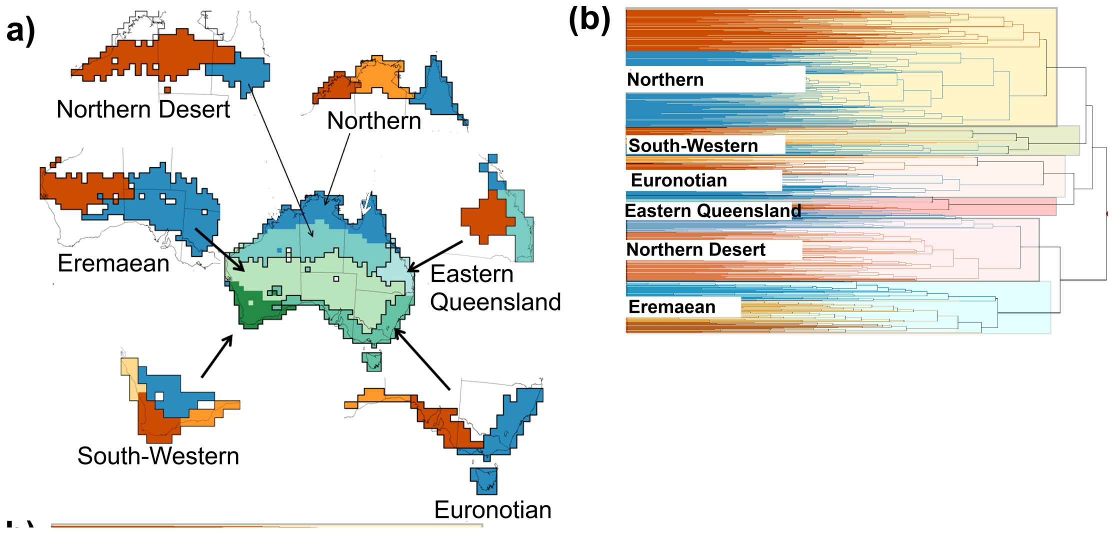
see a shareable output at: http://bit.ly/2QytYgy

Background

What is biogeographic regionalization?

- The characterization of geographical areas in terms of biodiversity.
- *Fundamental abstractions of life organization in the planet* in **response** to past or current ecological forces.

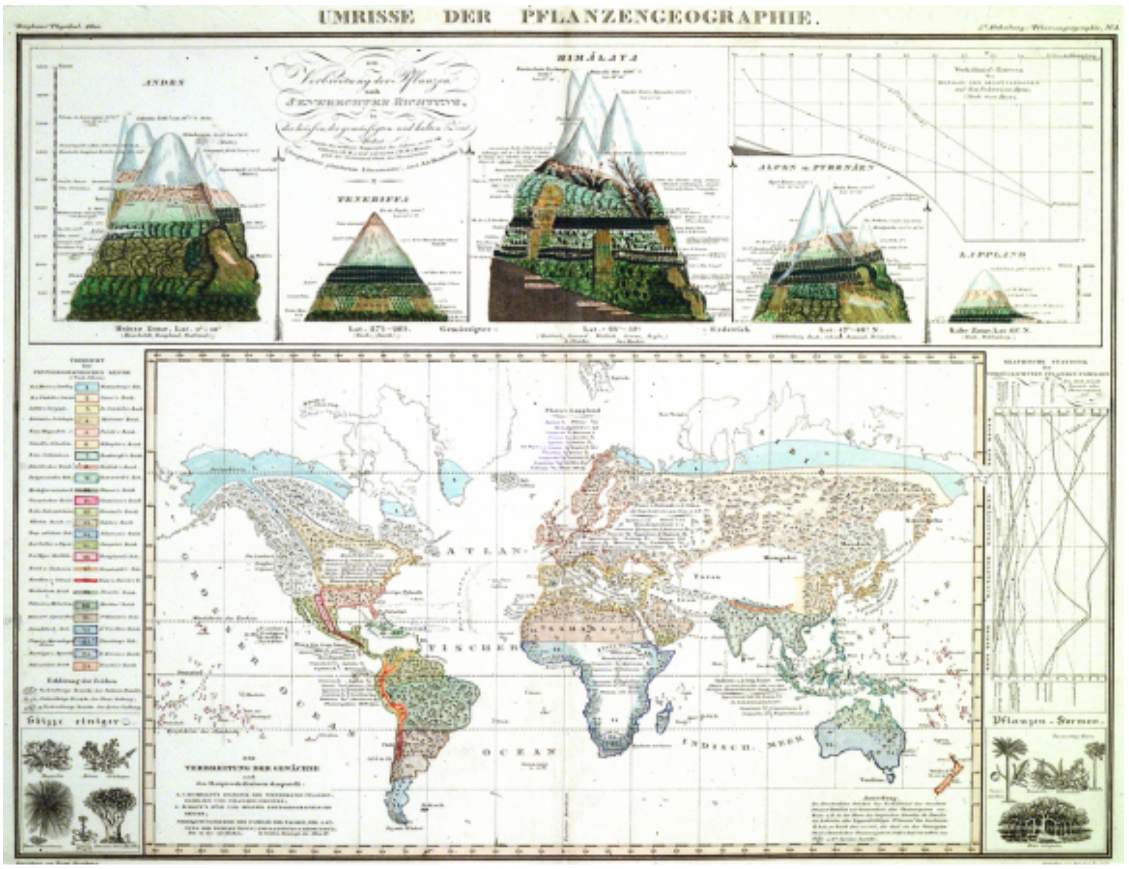
In summary, biogeographic regionalization look to define the patterns of how life is distributed in Earth.



Phytogeographic regions and new subregions proposed for Australia. (Gonzalez-Orosco et al. 2014: 10.1371/journal.pone.0092558)

Historical background

- Biogeograhic regionalization have been a problem in ecological research since the **18th** century.
- Early naturalists started to describe global pattenrs. e.g. vegetation zones, climate-diversity relationships.



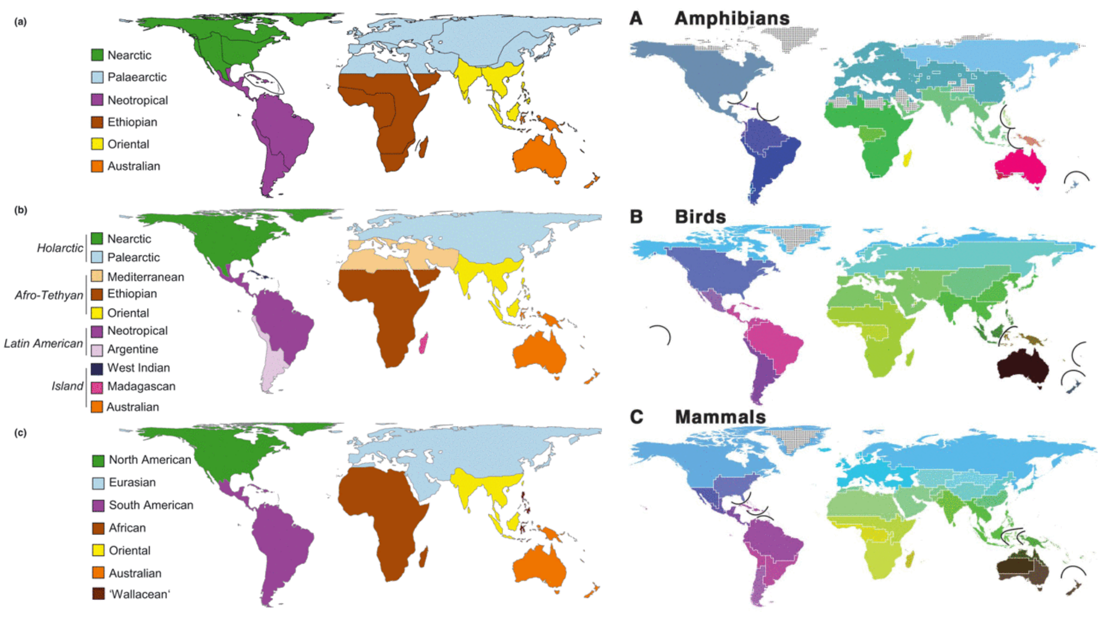
Umrisse des Plfanzengeographie (Outline of the Geography of Plants)H. Berghaus, 1851, Physikalischer Atlas, vol V, plate No. 1.

Biogeographic regionalization have also been critisized for different reasons:

- static vs fluid faunas
- areas of endemism are an artifact of sampling
- comparisons across taxa

Recently, there was a renovated interest in biogeographic regionalization looking to include to phylogenetic relationships.

In addition, new global databases provide basis for newer studies aimed to define global regions



left: Kreft and Jetz 2010 | right: Holt et al. 2012

Relevance

- Separation of global biodiversity in discrete units for analisis / comparison.
- Provide with spatially explicit answers to basic and applied questions in ecology, biodiversity science, conservation, etc.
  - one taxa -> Areas of endemism
  - one clade -> Phyto / Zoo geographic regions
- Evolutionary history and relationships between major geographical regions
- Links with environmental / biotic variables (e.g. isoclines, precipitation)

Classic workflow to delimit biogeographical regions

Hyerachichal framework

Updated steps according to **Morrone et al. 2018**

1.- Defining study area

2.- Assembling distributional data

3.- Identifying natural areas

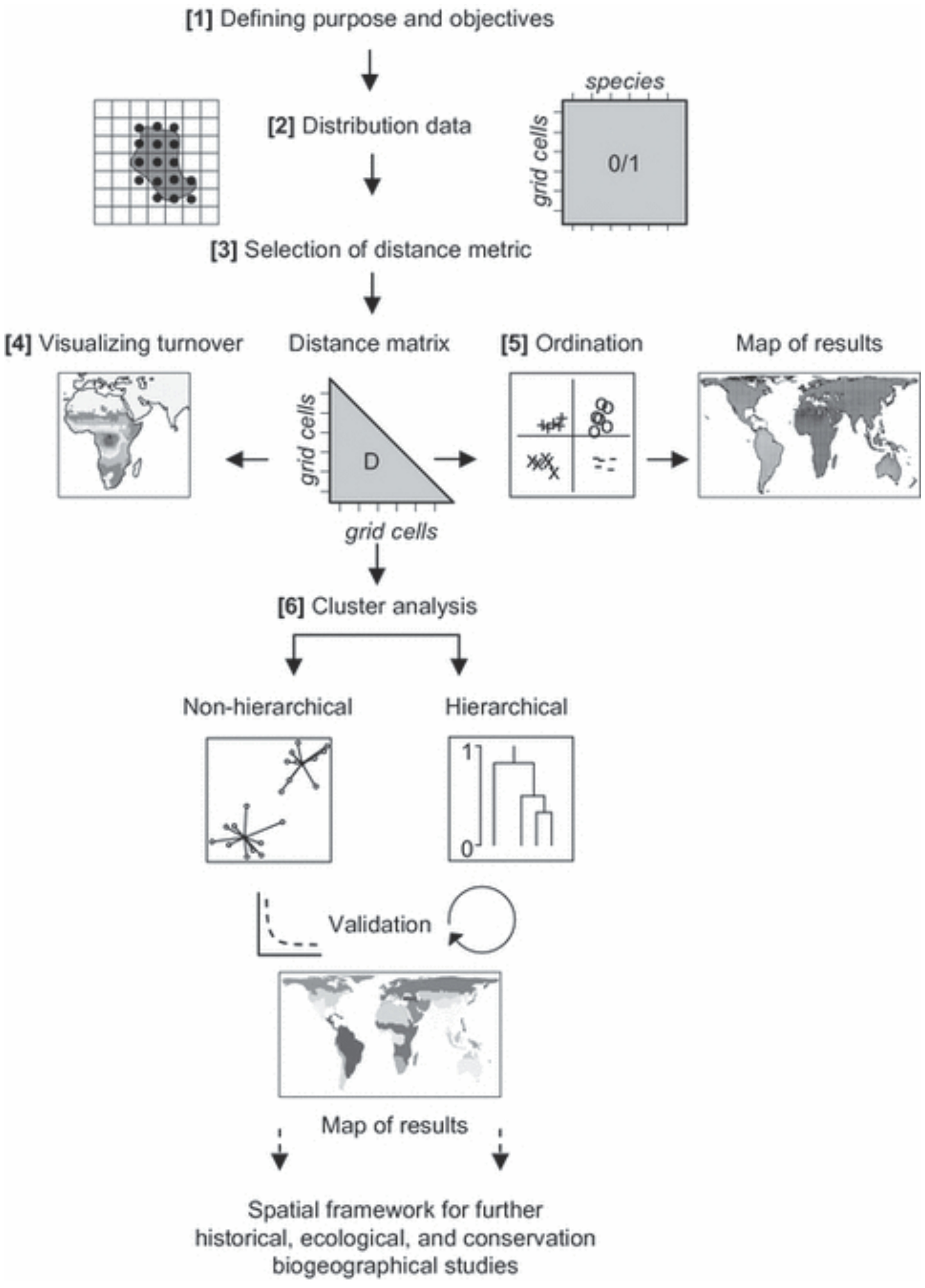
4.- Discovering area relationships

5.- Defining boundaries/transition zones

6.- Regionalizations

7.- Area nomenclature

Kreft and Jetz 2010 Workflow.



Kreft and Jetz 2010. A framework for delineating biogeographical regions

Kreft and Jetz. 2010. A framework for delineating biogeographical regions based on species distributions

Morrone 2018. The spectre of biogeographical regionalization

References and further reading

Kreft and Jetz. 2010. A framework for delineating biogeographical regions based on species distributions

Morrone 2018. The spectre of biogeographical regionalization

Holt et al. 2013. An update of Wallace's zoogeographic regions of the world

Wallace 1876. The geographical distribution of animals: with a study of the relations of living and extinct faunas as elucidating the past changes of the earth's surface