Analysis of species distribution data with R

Robert J. Hijmans

June 4, 2013

1 Introduction

In this vignette I show some techniques that can be used to analyze species distribution data with R . Before going through this document you should at least be somewhat familiar with R and spatial data in R . This document is based on an analysis of the distribution of wild potato species by Hijmans and Spooner (2001). Wild potatoes (Solanaceae; *Solanum* sect. *Petota*) are relatives of the cultivated potato. There are nearly 200 different species that occur in the Americas.

2 Import and prepare data

First download the data (in a zip file), and extract the files from the zip file.

```
> #download.file('http://diva-gis.org/docs/diva_ex1_data.zip', 'wildpot.zip')
> #unzip('wildpot.zip')
```

The extracted file is a tab delimited text file. Normally, you would read such a file with something like:

```
d <- read.table('WILDPOT.txt', header=TRUE)
```

But that does not work in this case (try it) because some lines are incomplete. So we have to resort to some more complicated tricks:

```
> # read all lines
> d <- readLines('WILDPOT.txt')
> # split each line into elements using the tabs
> dd <- strsplit(d, '\t')
> # show that the number of elements varies
> table(sapply(dd, length))

  18   19   20   21   22
 300 1372  170 1511 1647

> # function to complete each line to 22 items
> fun <- function(x) {
```

```
+         r <- rep("", 22)
+         r[1:length(x)] <- x
+         r
+ }
> # apply function to each element of the list
> ddd <- lapply(dd, fun)
> # row bind all elements (into a matrix)
> v <- do.call(rbind, ddd)
> head(v)

     [,1]  [,2]       [,3]       [,4]    [,5]    [,6]
[1,] "ID"  "COLNR"    "DATE"     "LongD" "LongM" "LongS"
[2,] "55"  "OKA 3901" "19710405" "65"    "45"    "0"
[3,] "16"  "OKA 3920" "19710406" "66"    "6"     "0"
[4,] "204" "HOF 1848" "19710305" "65"    "5"     "0"
[5,] "545" "OKA 4015" "19710411" "66"    "15"    "0"
[6,] "549" "OKA 4026" "19710411" "66"    "12"    "0"
     [,7]    [,8]   [,9]   [,10]  [,11]  [,12]
[1,] "LongH" "LatD" "LatM" "LatS" "LatH" "SPECIES"
[2,] "W"     "22"   "8"    "0"    "S"    "S. acaule Bitter"
[3,] "W"     "21"   "53"   "0"    "S"    "S. acaule Bitter"
[4,] "W"     "22"   "16"   "0"    "S"    "S. acaule Bitter"
[5,] "W"     "22"   "32"   "0"    "S"    "S. acaule Bitter"
[6,] "W"     "22"   "30"   "0"    "S"    "S. acaule Bitter"
     [,13]       [,14]     [,15]   [,16]       [,17]
[1,] "SCODE_NEW" "SUB_NEW" "SP_ID" "COUNTRY"   "ADM1"
[2,] "acl"       "ACL"     "1"     "ARGENTINA" "Jujuy"
[3,] "acl"       "ACL"     "1"     "ARGENTINA" "Jujuy"
[4,] "acl"       "ACL"     "1"     "ARGENTINA" "Salta"
[5,] "acl"       "ACL"     "1"     "ARGENTINA" "Jujuy"
[6,] "acl"       "ACL"     "1"     "ARGENTINA" "Jujuy"
     [,18]
[1,] "ADM2"
[2,] "Yavi"
[3,] "Santa Catalina"
[4,] "Santa Victoria"
[5,] "Rinconada"
[6,] "Rinconada"
     [,19]
[1,] "LOCALITY"
[2,] "Tafna."
[3,] "10 km W of Santa Catalina."
[4,] "53 km E of Cajas."
[5,] "\"Near Abra de Fundiciones, 10 km S of Rinconada.\""
[6,] "8 km SW of Fundiciones."
     [,20]   [,21]   [,22]
```

```
[1,] "PLRV1" "PLRV2" "FROST"
[2,] "R"     "R"     "100"
[3,] "S"     "R"     "100"
[4,] "S"     "R"     "100"
[5,] "S"     "R"     "100"
[6,] "S"     "R"     "100"
```

```
> #set the column names and remove them from the data
> colnames(v) <- v[1,]
> v <- v[-1,]
> # coerce into a data.frame and change the type of some variables
> # to numeric (instead of character)
> v <- data.frame(v, stringsAsFactors=FALSE)
```

The coordinate data is in degrees, minutes, seconds (in separate columns, fortunately), so we need to compute longitude and latitude as single numbers.

```
> # first coerce character values to numbers
> for (i in c('LongD', 'LongM', 'LongS', 'LatD', 'LatM', 'LatS')) {
+         v[, i] <- as.numeric(v[,i])
+ }
> v$lon <- -1 * (v$LongD + v$LongM / 60 + v$LongS / 3600)
> v$lat <- v$LatD + v$LatM / 60 + v$LatS / 3600
> # Southern hemisphere gets a negative sign
> v$lat[v$LatH == 'S'] <- -1 * v$lat[v$LatH == 'S']
> head(v)
```

```
   ID     COLNR     DATE LongD LongM LongS LongH LatD LatM
1  55  OKA 3901 19710405    65    45     0     W   22    8
2  16  OKA 3920 19710406    66     6     0     W   21   53
3 204  HOF 1848 19710305    65     5     0     W   22   16
4 545  OKA 4015 19710411    66    15     0     W   22   32
5 549  OKA 4026 19710411    66    12     0     W   22   30
6 551 OKA 4030A 19710411    66    12     0     W   22   28
  LatS LatH          SPECIES SCODE_NEW SUB_NEW SP_ID
1    0    S S. acaule Bitter       acl     ACL     1
2    0    S S. acaule Bitter       acl     ACL     1
3    0    S S. acaule Bitter       acl     ACL     1
4    0    S S. acaule Bitter       acl     ACL     1
5    0    S S. acaule Bitter       acl     ACL     1
6    0    S S. acaule Bitter       acl     ACL     1
    COUNTRY  ADM1           ADM2
1 ARGENTINA Jujuy           Yavi
2 ARGENTINA Jujuy Santa Catalina
3 ARGENTINA Salta Santa Victoria
4 ARGENTINA Jujuy      Rinconada
5 ARGENTINA Jujuy      Rinconada
```

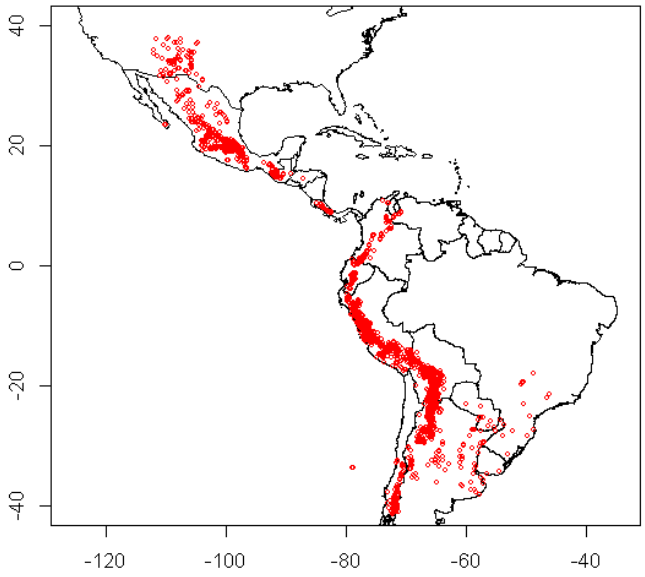
```
6 ARGENTINA Jujuy      Rinconada
                                                LOCALITY PLRV1
1                                                 Tafna.     R
2                           10 km W of Santa Catalina.     S
3                                    53 km E of Cajas.     S
4 "Near Abra de Fundiciones, 10 km S of Rinconada."     S
5                              8 km SW of Fundiciones.     S
6                  "Salveayoc, 5 km SW of Rinconada."     S
  PLRV2 FROST       lon       lat
1     R   100 -65.75000 -22.13333
2     R   100 -66.10000 -21.88333
3     R   100 -65.08333 -22.26667
4     R   100 -66.25000 -22.53333
5     R   100 -66.20000 -22.50000
6     R   100 -66.20000 -22.46667
```

Read a 'shapefile' with most of the countries of the Americas.

```
> library(raster)
> cn <- shapefile('pt_countries.shp')
> proj4string(cn) <- CRS("+proj=lonlat +datum=WGS84")
> class(cn)

[1] "SpatialPolygonsDataFrame"
attr(,"package")
[1] "sp"
```

Make a quick map:

```
> library(raster)
> class(cn)

[1] "SpatialPolygonsDataFrame"
attr(,"package")
[1] "sp"

> plot(cn, xlim=c(-120, -40), ylim=c(-40,40), axes=TRUE)
> points(v$lon, v$lat, cex=.5, col='red')
```



And a SpatialPointsDataFrame with the formula approach:

```
> sp <- v
> coordinates(sp) <- ~lon + lat
```

Alternatively, you can do:

```
> sp <- SpatialPoints(v[, c('lon', 'lat')])
> sp <- SpatialPointsDataFrame(sp, v)
> proj4string(sp) <- CRS("+proj=lonlat +datum=WGS84")
```

3 Summary statistics

We are first going to summarize the data by country. We can use the country variable in the data, or extract that from the countries SpatialPolygonsDataFrame.

```
> table(v$COUNTRY)
```

```
  ARGENTINA     BOLIVIA      BRAZIL       CHILE
       1474         985          17         100
   COLOMBIA  COSTA RICA     ECUADOR   GUATEMALA
        107          24         138          59
   HONDURAS      Mexico      MEXICO      PANAMA
```

```
          1           2         843          13
   PARAGUAY        Peru        PERU UNITED STATES
         19           1        1043         157
    URUGUAY   VENEZUELA
          4          12
```

```
> # note Peru and PERU
> v$COUNTRY <- toupper(v$COUNTRY)
> table(v$COUNTRY)
```

```
   ARGENTINA      BOLIVIA       BRAZIL        CHILE
        1474          985           17          100
    COLOMBIA   COSTA RICA      ECUADOR    GUATEMALA
         107           24          138           59
    HONDURAS       MEXICO       PANAMA     PARAGUAY
           1          845           13           19
        PERU UNITED STATES      URUGUAY    VENEZUELA
        1044          157            4           12
```

```
> # same fix for the SpatialPointsDataFrame
> sp$COUNTRY <- toupper(sp$COUNTRY)
```

Below we determine the country using a spatial query, using the "over" function in rgeos.

```
> library(rgeos)
> ov <- over(sp, cn)
> colnames(ov) <- 'name'
> head(ov)
```

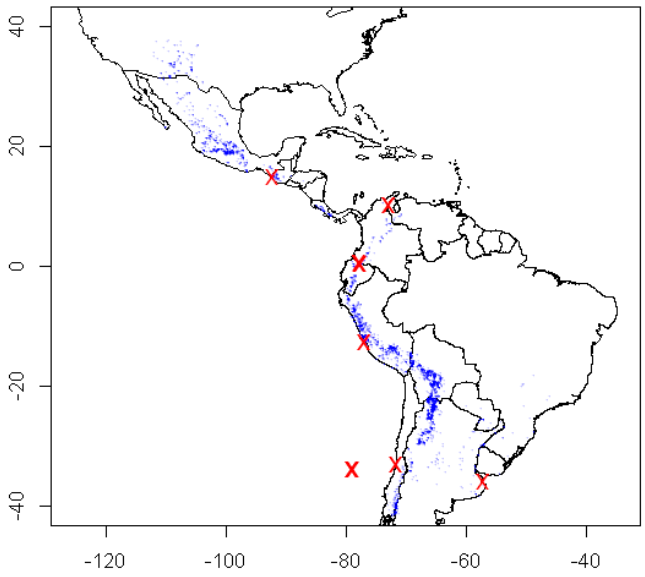
```
       name
1 ARGENTINA
2 ARGENTINA
3 ARGENTINA
4 ARGENTINA
5 ARGENTINA
6 ARGENTINA
```

```
> v <- cbind(v, ov)
> table(v$COUNTRY)
```

```
   ARGENTINA      BOLIVIA       BRAZIL        CHILE
        1474          985           17          100
    COLOMBIA   COSTA RICA      ECUADOR    GUATEMALA
         107           24          138           59
    HONDURAS       MEXICO       PANAMA     PARAGUAY
           1          845           13           19
        PERU UNITED STATES      URUGUAY    VENEZUELA
        1044          157            4           12
```

This table is similar to the previous table, but it is not the same. Let's find the records that are not in the same country according to the original data and the spatial query.

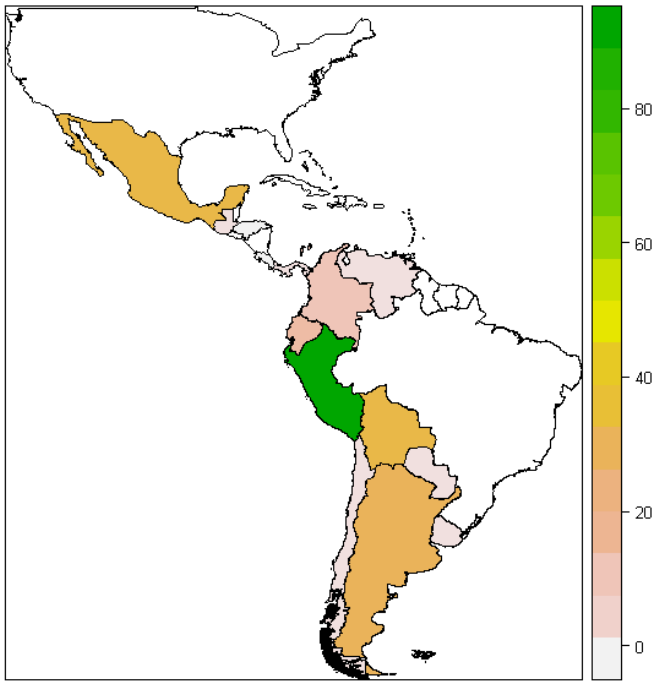
```
> # some fixes first
> # apparantly in the ocean (small island missing from polygon data)
> v$name[is.na(v$name)] <- ''
> # some spelling differenes
> v$name[v$name=="UNITED STATES, THE"] <- "UNITED STATES"
> v$name[v$name=="BRASIL"] <- "BRAZIL"
> i <- which(toupper(v$name) != v$COUNTRY)
> i

 [1]  581  582 1367 1617 1635 1951 1952 1953 1954 2804 2805
[12] 2855 2856 3223 3525

> plot(cn, xlim=c(-120, -40), ylim=c(-40,40), axes=TRUE)
> points(sp, cex=.25, pch='+', col='blue')
> points(sp[i,], col='red', pch='x', cex=1.5)
```



All locations with that fall in a different country than their attribute data suggests are very close to an international border, or in the water. That is reassuring. It can be hard to find out whether the point coordinates are wrong or whether the borders are wrong; but further inspection is warranted.

We can compute the number of species for each country:

```
> spc <- tapply(v$SPECIES, sp$COUNTRY, function(x)length(unique(x)) )
> spc <- data.frame(COUNTRY=names(spc), nspp = spc)
> # prepare for merging with country SpatialPolygonsDataFrame
> cn$order <- 1:nrow(cn)
> # merge
> dat <- merge(cn, spc, by='COUNTRY', all.x=TRUE)
> # assure that valus are sorted correctly
> cn@data <- dat[order(dat$order), ]
> print(spplot(cn, 3, col.regions=rev(terrain.colors(25))))
```



This map shows that Peru is the country with most potato species, followed by Bolivia and Mexico. We can also tabulate the number of occurences of each species by each country:

```
> tb <- table(v[ c('COUNTRY', 'SPECIES')])
> # a big table
> dim(tb)

[1]  16 195

> # we show two columns:
> tb[,2:3]
```

```
               SPECIES
COUNTRY         S. acaule Bitter S. achacachense Cßrdenas
  ARGENTINA                  238                         0
  BOLIVIA                    114                         8
  BRAZIL                       0                         0
  CHILE                        0                         0
  COLOMBIA                     0                         0
  COSTA RICA                   0                         0
  ECUADOR                      0                         0
  GUATEMALA                    0                         0
  HONDURAS                     0                         0
  MEXICO                       0                         0
  PANAMA                       0                         0
  PARAGUAY                     0                         0
  PERU                        52                         0
  UNITED STATES                0                         0
  URUGUAY                      0                         0
  VENEZUELA                    0                         0
```

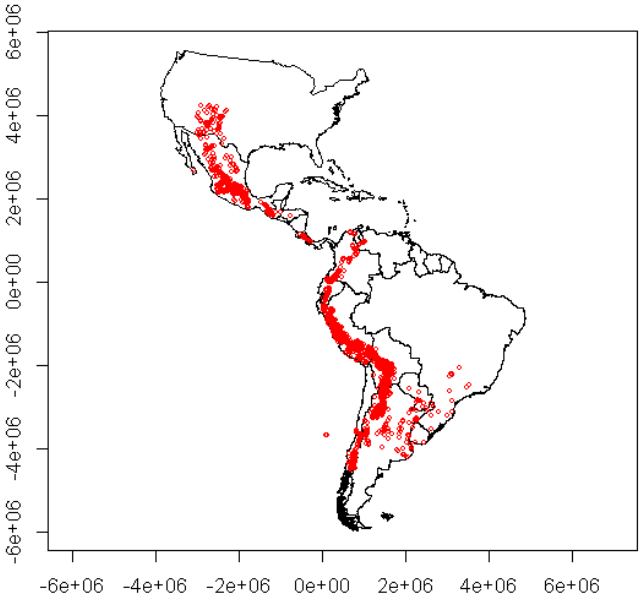
Because the countries have such different sizes and shapes, and because most are very large, it is in most cases better to use grids with cells of equal area, and that is what we will do next.

4 Projecting spatial data

To be able to use a grid with cells of equal area, the data need to be projected to an equal-area coordinate reference system (CRS). If the longitude/latitude date were used, cells of say 1 square degree would get smaller as you move away from the equator: think of the meridians (vertical lines) on the globe getting closer to each other as you go towards the poles.

For small areas, particularly if they only span a few degrees of longitude, UTM can be a good CRS, but it this case we will use a CRS that can be used for a complete hemisphere: Lambert Equal Area Azimuthal. For this CRS, you must choose a map origin for your data. This should be somewhere in the center of the points, to minimize the distance (and hence distortion) from any point to the origin. In this case, the center could be (-80, 0).

```
> library(rgdal)
> # "proj.4" notation of CRS
> projection(cn) <- "+proj=lonlat +datum=WGS84"
> # the CRS we want
> laea <- CRS("+proj=laea  +lat_0=0 +lon_0=-80")
> clb <- spTransform(cn, laea)
> pts <- spTransform(sp, laea)
> plot(clb, axes=TRUE)
> points(pts, col='red', cex=.5)
```



Note that the shape of the countries is now much more similar to their shape on a globe that before projecting. You can also see that the coordinate system has changed by looking at the numbers of the axes. These express the distance from the origin (-80, 0 degrees) in meters.

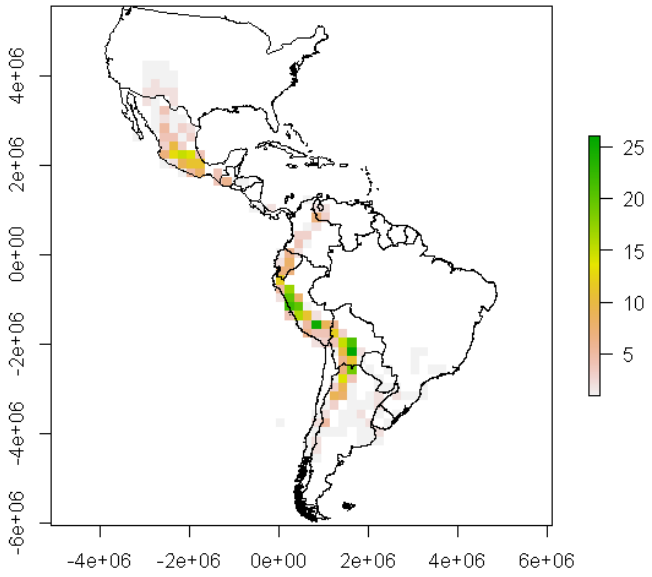
5 Species richness

Let's determine the distribution of species richness using a grid. First we need an empty 'template' raster that has the correct extent and resolution. Here I use 200 by 200 km cells.

```
> r <- raster(clb)
> # 200 km = 200000 m
> res(r) <- 200000
```

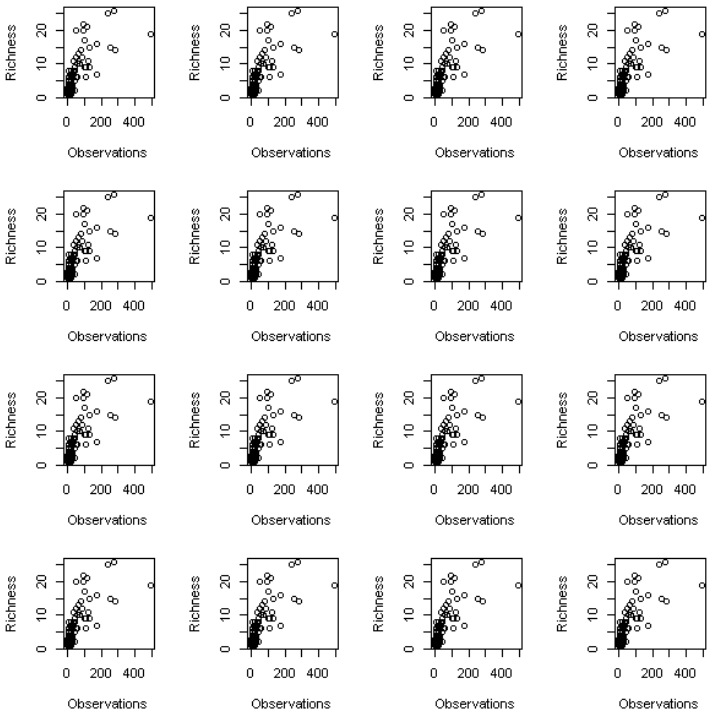
Now compute the number of observations and the number of species richness for each cell

```
> # to work around a bug in the present version
> # turning a character vector into numbers
> field <- as.integer(as.factor(pts$SPECIES))
> rich <- rasterize(pts, r, field, function(x, ...) length(unique(na.omit(x))))
> plot(rich)
> plot(clb, add=TRUE)
```



Now we make a grid of the number of observations and do a cell by cell comparison of the number of species and the number of observations.

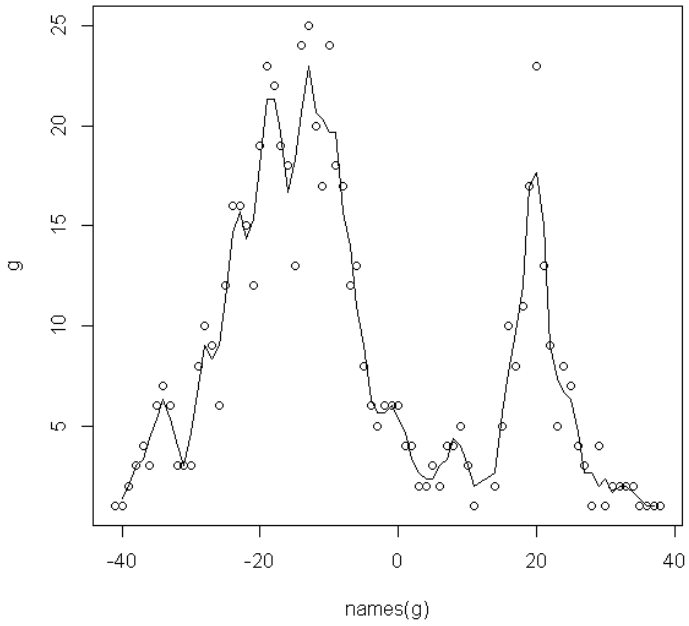
```
> obs <- rasterize(pts, r, fun=function(x, ...)length((na.omit(x))) )
> plot(obs, rich, cex=1, xlab='Observations', ylab='Richness')
```



The problem is of course that this association will always exist. When there are only few species in an area, collectors will not continue to go there to increase the number of (redundant) observations. However, in this case, the relationship is not as strong as it can be, and there is a clear pattern in species richness maps, it is not characterized by sudden random like changes in richness (it looks like there is spatial autocorrelation, which is probably a good thing). Ways to correct for this 'collector-bias' include the use of 'richness estimators'.

There are often gradients of species richness over latitude and altitude. Here is how you can make a plot of the latitudinal gradient in species richness:

```
> d <- v[, c('lat', 'SPECIES')]
> d$lat <- round(d$lat)
> g <- tapply(d$SPECIES, d$lat, function(x) length(unique(na.omit(x))) )
> plot(names(g), g)
> # moving average
> lines(names(g), movingFun(g, 3))
```



The distribution of species richness has two peaks. What would explain the low species richness between -5 and 15 degrees?

6 range sizes

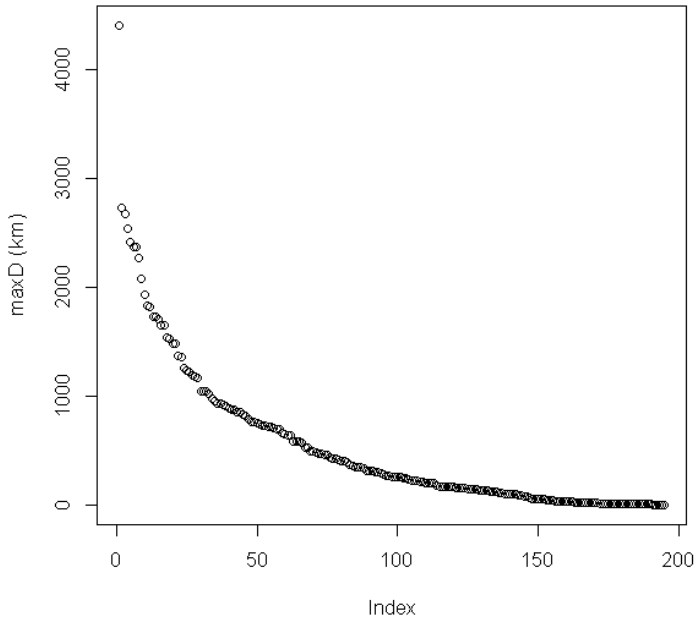
Let's estimate range sizes of the species. Hijmans and Spooner use two ways: (1) maxD, the maximum distance between any pair of points for a species, and CA50 the total area covered by circles of 50 km around each species. Here, I also add the convex hull. I am using the projected coordinates, but it is also possible to use the geosphere package to compute these things from the original longitude/latitude data.

```
> # get the (Lambert AEA) coordinates
> # from the SpatialPointsDataFrame
> xy <- coordinates(pts)
> # list of species
> sp <- unique(pts$SPECIES)
```

Compute maxD for each species

```
> maxD <- vector(length=length(sp))
> for (s in 1:length(sp)) {
+         # get the coordinates for species 's'
```

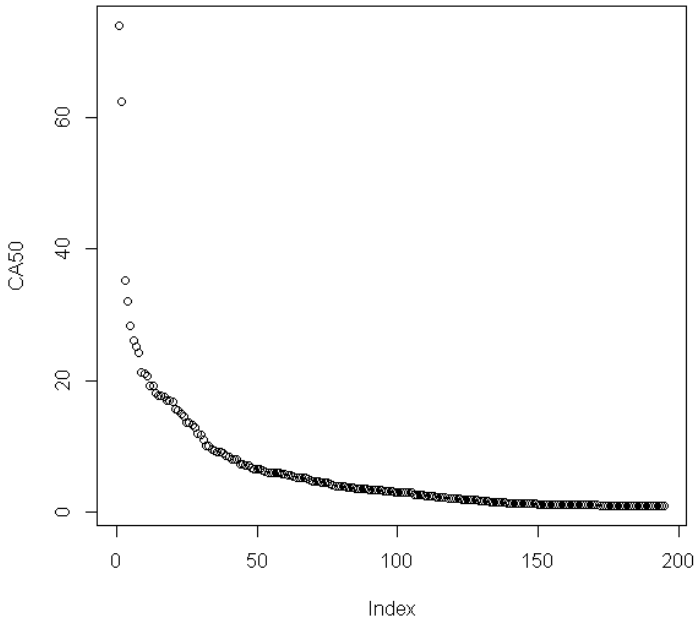
```
+         p <- xy[pts$SPECIES == sp[s], ]
+         # distance matrix
+         d <- as.matrix(dist(p))
+         # ignore the distance of a point to itself
+         diag(d) <- NA
+         # get max value
+         maxD[s] <- max(d, na.rm=T)
+ }
> # typical J shape
> plot(rev(sort(maxD))/1000, ylab='maxD (km)')
```



Compute CA

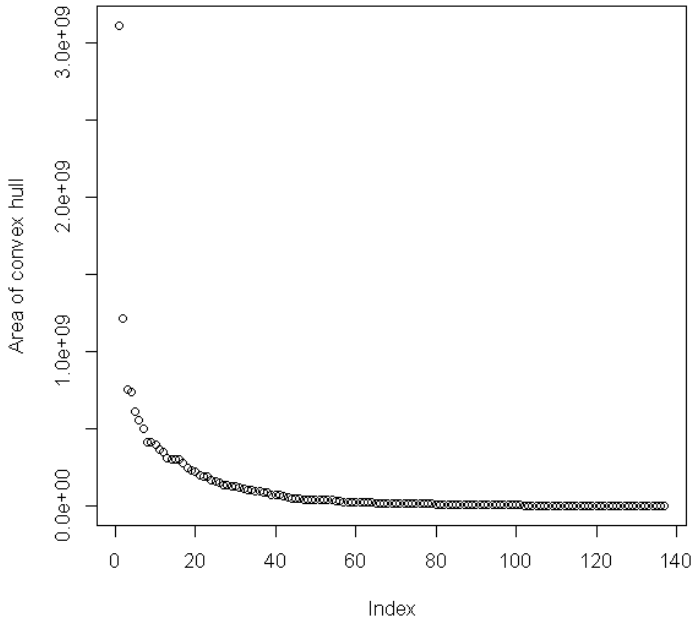
```
> library(dismo)
> library(rgeos)
> CA <- vector(length=length(sp))
> for (s in 1:length(sp)) {
+         p <- xy[pts$SPECIES == sp[s], ,drop=FALSE]
+         # run "circles" model
+         m <- circles(p, d=50000, lonlat=FALSE)
+         # dissolve polygons
+         pol <- gUnionCascaded(m@polygons)
+         CA[s] <- pol@polygons[[1]]@area
```

```
+ }
> # standardize to the size of one circle
> CA <- CA / (pi * 50000^2)
> plot(rev(sort(CA)), ylab='CA50')
```



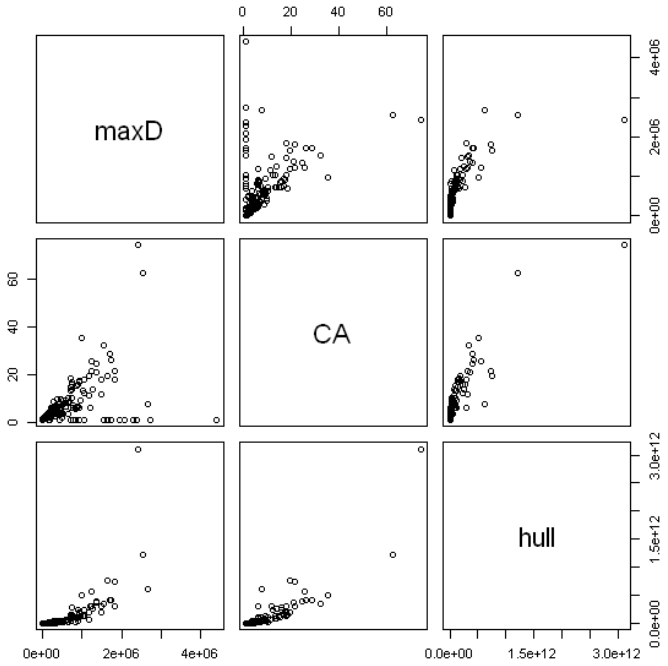
Compute convex hull area

```
> hull <- rep(NA, length(sp))
> for (s in 1:length(sp)) {
+         p <- unique(xy[pts$SPECIES == sp[s], ,drop=FALSE])
+         # need at least three points for hull
+         if (nrow(p) > 3) {
+                 h <- convHull(p, lonlat=FALSE)
+                 pol <- h@polygons
+                 hull[s] <- pol@polygons[[1]]@area
+         }
+ }
> plot(rev(sort(hull))/1000, ylab='Area of convex hull')
```



Compare all three:

```
>   d <- cbind(maxD,CA,hull)
>   pairs(d)
```



7 Mapping traits

There is information about two traits in the data set in field PRLV (tolerance to Potato Leaf Roll Virus) and forst (forst tolerance). Make a map of average frost tolerance and a model that related this trait to climate data of the origin of the plant.

8 References

Hijmans, R.J., and D.M. Spooner, 2001. Geographic distribution of wild potato species. American Journal of Botany 88:2101-2112