Spatial Data Analysis Case Studies

Robert J. Hijmans

May 01, 2021

CONTENTS

CHAPTER

ONE

1. INTRODUCTION

This is a (still very small) collection of case studies of spatial data analyis with *R*.

It is part of these Introduction to Spatial Data Analysis with R resources.

CHAPTER

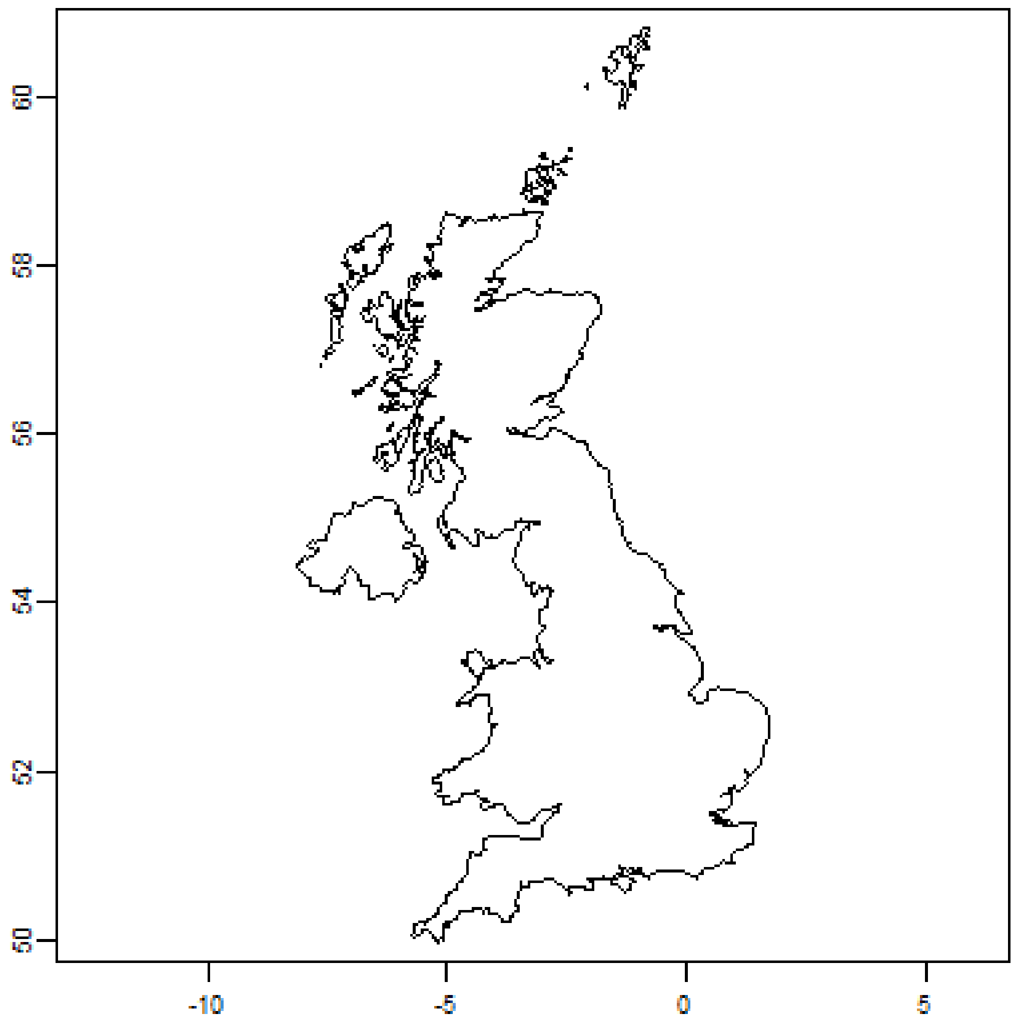
TWO

2. THE LENGTH OF A COASTLINE

How Long Is the Coast of Britain? Statistical Self-Similarity and Fractional Dimension is the title of a famous paper by Benoît Mandelbrot. Mandelbrot uses data from a paper by Lewis Fry Richardson who showed that the length of a coastline changes with scale, or, more precisely, with the length (resolution) of the measuring stick (ruler) used. Mandelbrot discusses the fractal dimension *D* of such lines. *D* is 1 for a straight line, and higher for more wrinkled shapes. For the west coast of Britain, Mandelbrot reports that *D*=1.25. Here I show how to measure the length of a coast line with rulers of different length and how to compute a fractal dimension.

First we get a high spatial resolution (30 m) coastline for the United Kingdom from the GADM database.

```
library(terra)
## terra version 1.2.4
library(geodata)
w <- world(path=".", resolution = 3)
uk <- w[w$GID_0=="GBR", ]
plot(uk)
```



This is a single "multi-polygon" (it has a single geometry) and a longitude/latitude coordinate reference system.

```
as.data.frame(uk)
##   GID_0          NAME_0
## 1   GBR United Kingdom
```

Let's transform this to a planar coordinate system. That is not required, but it will speed up computations. We used the British National Grid coordinate reference system, which is based on the Transverse Mercator (tmerc) projection, with units in meter.

```
prj <- "epsg:27700"
```

With that we can transform the coordinates of `uk` from longitude latitude to the British National Grid.

```
guk <- project(uk, prj)
```

We only want the main island, so want need to separate (disaggregate) the different polygons.

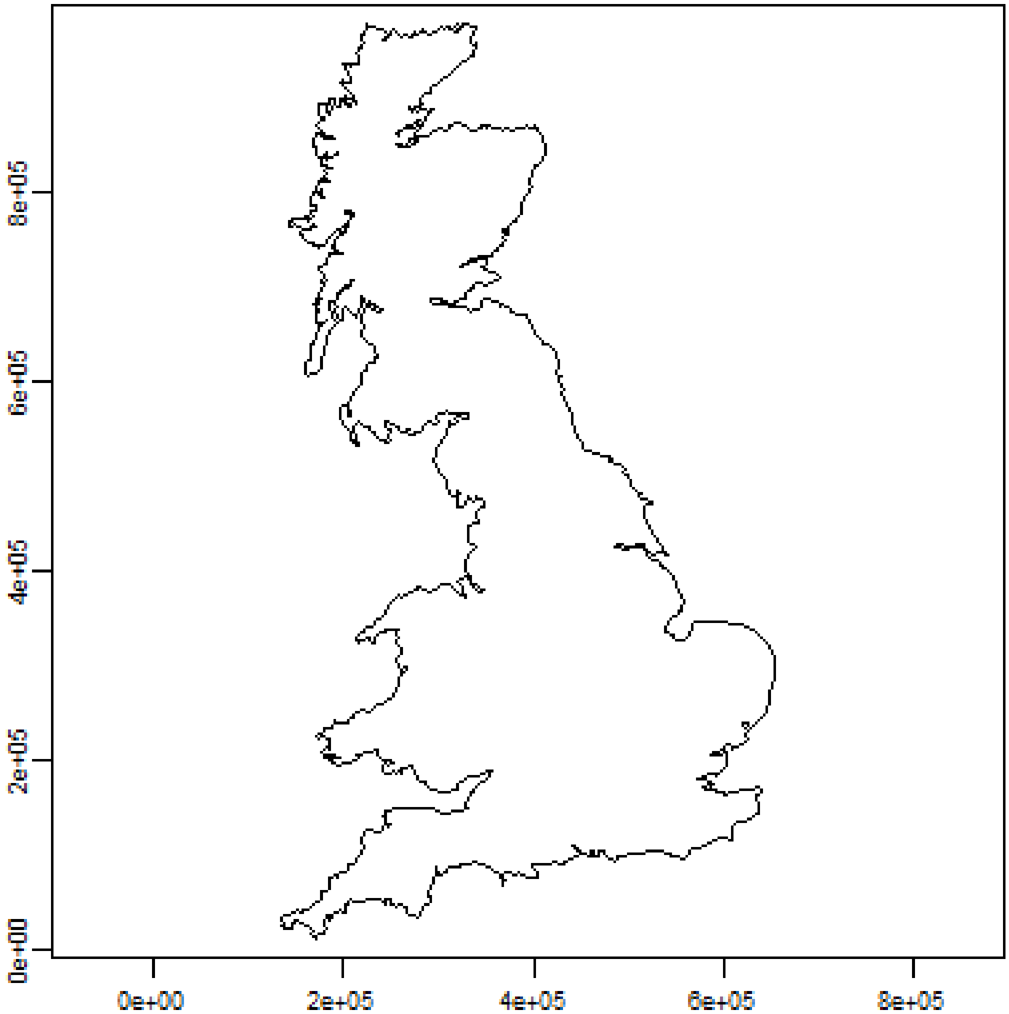
```
duk <- disaggregate(guk)
head(duk)
##    GID_0          NAME_0
## 1    GBR United Kingdom
## 2    GBR United Kingdom
## 3    GBR United Kingdom
## 4    GBR United Kingdom
## 5    GBR United Kingdom
## 6    GBR United Kingdom
```

Now we have 920 features. We want the largest one.

```
a <- area(duk)
i <- which.max(a)
a[i] / 1000000
## [1] 219681.3
b <- duk[i,]
```

Britain has an area of about 220,000 km^2.

```
par(mai=rep(0,4))
plot(b)
```



On to the tricky part. The function to go around the coast with a ruler (yardstick) of a certain length.

```
measure_with_ruler <- function(pols, stick_length, lonlat=FALSE) {
    # some sanity checking
    stopifnot(inherits(pols, "SpatVector"))
    stopifnot(length(pols) == 1)

    # get the coordinates of the polygon
    g <- geom(pols)[, c('x', 'y')]
    nr <- nrow(g)

    # we start at the first point
    pts <- 1
    newpt <- 1
    while(TRUE) {
        # start here
        p <- newpt
```

(continues on next page)

(continued from previous page)

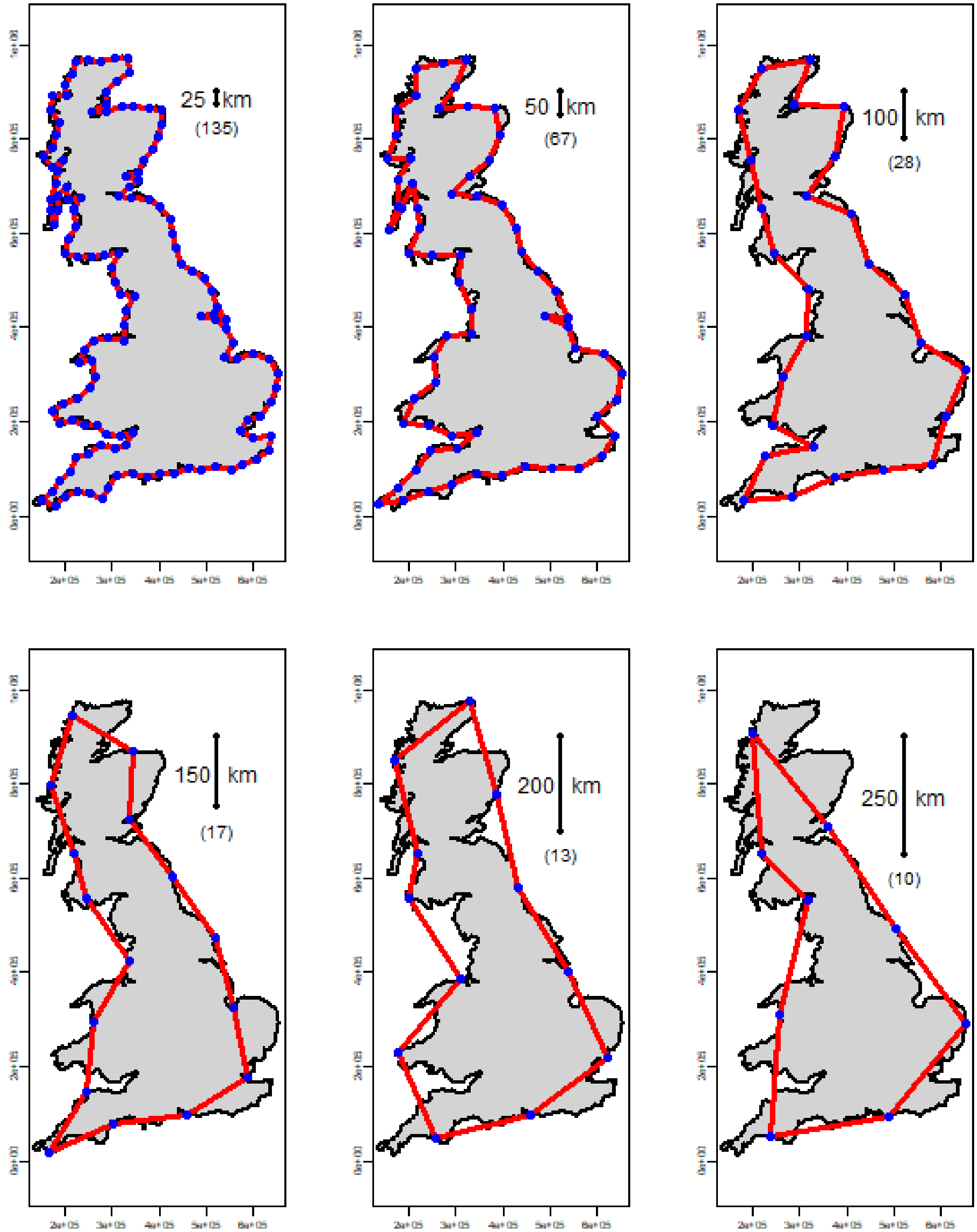
```
        # order the points
        j <- p:(p+nr-1)
        j[j > nr] <- j[j > nr] - nr
        gg <- g[j,]

        # compute distances
        pd <- distance(gg[1,,drop=FALSE], gg, lonlat)
        pd <- as.vector(pd)
        # get the first point that is past the end of the ruler
        # this is precise enough for our high resolution coastline
        i <- which(pd > stick_length)[1]
        if (is.na(i)) {
            stop('Ruler is longer than the maximum distance found')
        }

        # get the record number for new point in the original order
        newpt <- i + p

        # stop if past the last point
        if (newpt >= nr) break

        pts <- c(pts, newpt)
    }
    # add the last (incomplete) stick.
    pts <- c(pts, 1)
    # return the locations
    g[pts, ]
}
```

Now we have the function, life is easy, we just call it a couple of times, using rulers of different lengths (although it takes a while to run).

```
y <- list()
rulers <- c(25,50,100,150,200,250) # km
for (i in 1:length(rulers)) {
    y[[i]] <- measure_with_ruler(b, rulers[i]*1000)
}
```

Object `y` is a list of matrices containing the locations where the ruler touched the coast. We can plot these on top of the map of Britain.

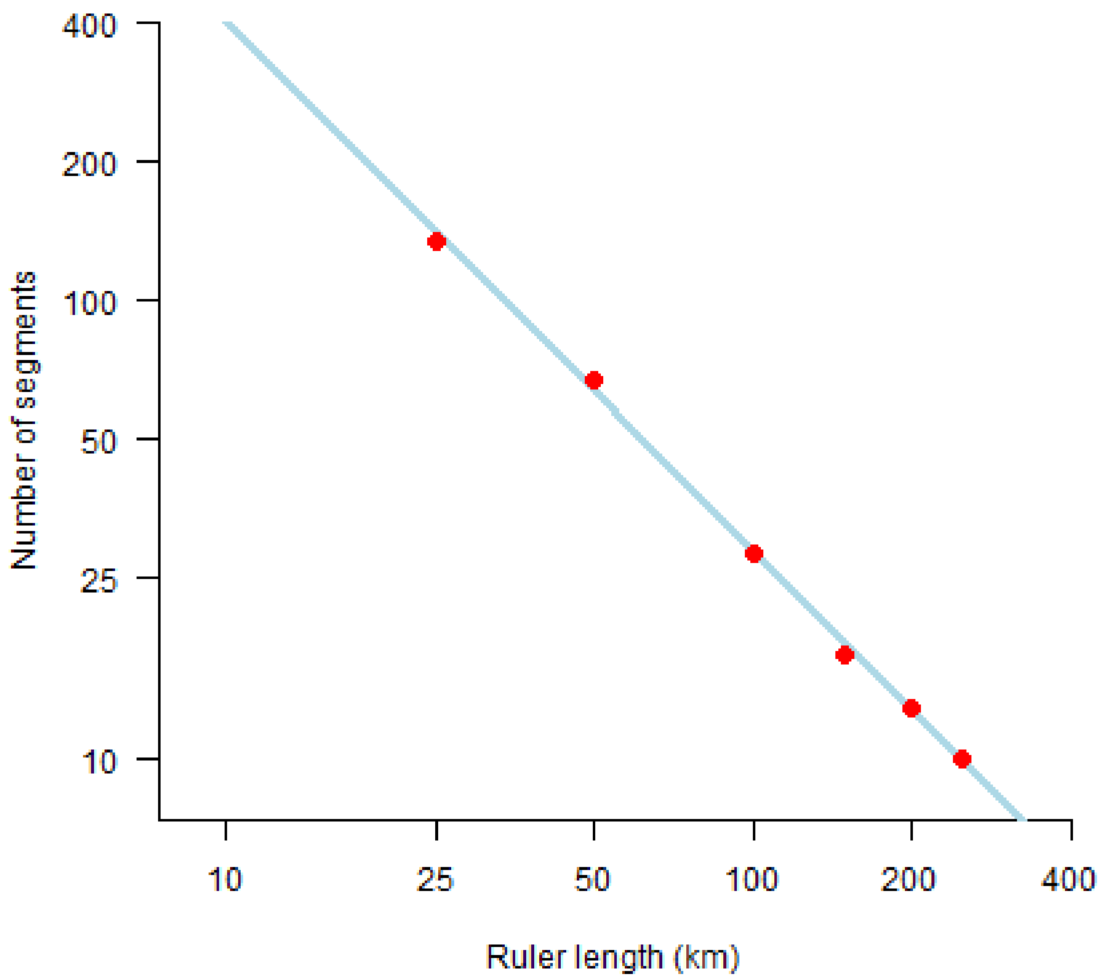
```
par(mfrow=c(2,3), mai=rep(0,4))
for (i in 1:length(y)) {
    plot(b, col='lightgray', lwd=2)
    p <- y[[i]]
    lines(p, col='red', lwd=3)
    points(p, pch=20, col='blue', cex=2)

    bar <- rbind(cbind(525000, 900000), cbind(525000, 900000-rulers[i]*1000))
    lines(bar, lwd=2)
    points(bar, pch=20, cex=1.5)
    text(525000, mean(bar[,2]), paste(rulers[i], ' km'), cex=1.5)
    text(525000, bar[2,2]-50000, paste0('(', nrow(p), ')'), cex=1.25)
}
```



The coastline of Britain, measured with rulers of different lengths. The number of segments is in parenthesis. f

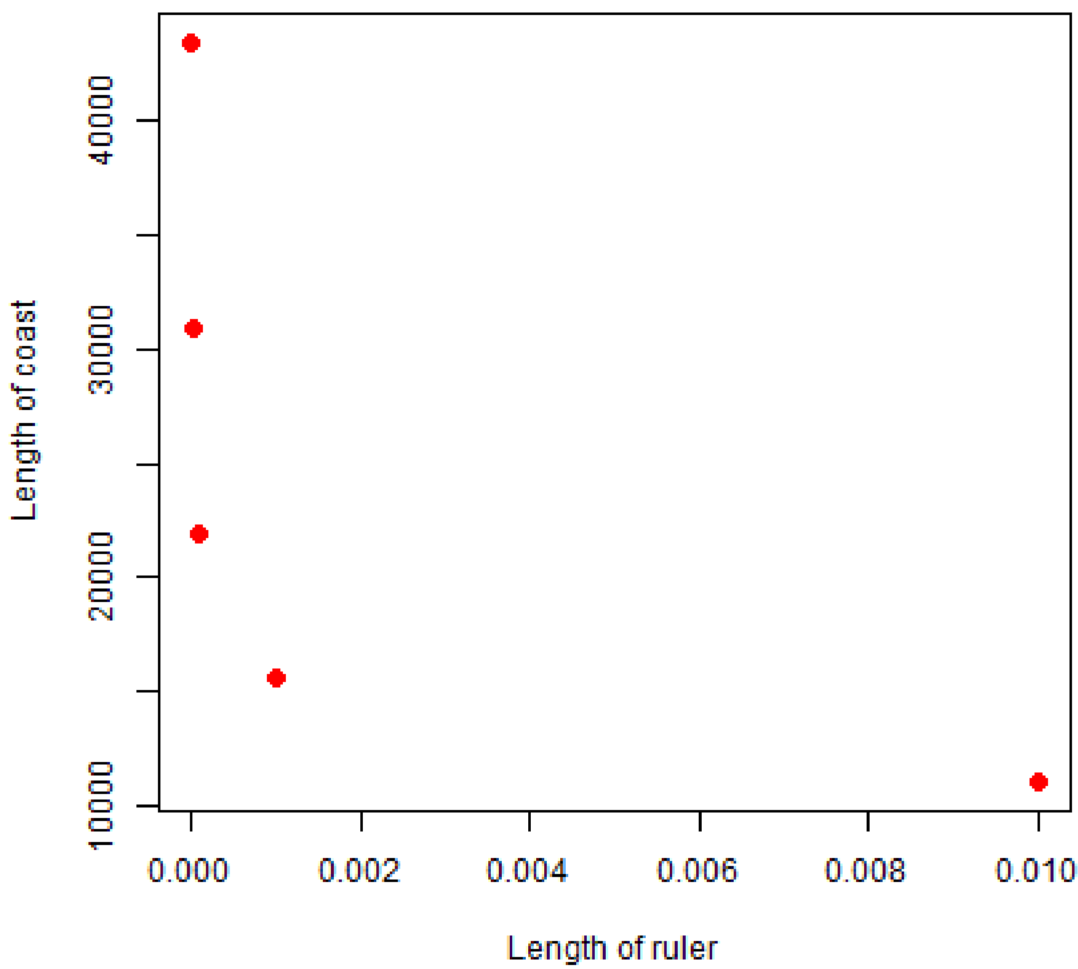
Here is the fractal (log-log) plot. Note how the axes are on the log scale, but that I used the non-transformed values for the labels.

```
# number of times a ruler was used
n <- sapply(y, nrow)

# set up empty plot
plot(log(rulers), log(n), type='n', xlim=c(2,6), ylim=c(2,6), axes=FALSE,
       xaxs="i",yaxs="i", xlab='Ruler length (km)', ylab='Number of segments')

# axes
tics <- c(1,10,25,50,100,200,400)
axis(1, at=log(tics), labels=tics)
axis(2, at=log(tics), labels=tics, las=2)

# linear regression line
m <- lm(log(n)~log(rulers))
abline(m, lwd=3, col='lightblue')

# add observations
points(log(rulers), log(n), pch=20, cex=2, col='red')
```



What does this mean? Let's try some very small rulers, from 1 mm to 10 m.

```
small_rulers <- c(0.000001, 0.00001, 0.0001, 0.001, 0.01)  # km
nprd <- exp(predict(m, data.frame(rulers=small_rulers)))
coast <- nprd * small_rulers
plot(small_rulers, coast, xlab='Length of ruler', ylab='Length of coast', pch=20,␣
↪cex=2, col='red')
```



So as the ruler get smaller, the coastline gets exponentially longer. As the ruler approaches zero, the length of the coastline approaches infinity.

The fractal dimension *D* of the coast of Britain is the (absolute value of the) slope of the regression line.

```
m
##
## Call:
## lm(formula = log(n) ~ log(rulers))
##
```

(continues on next page)

(continued from previous page)

```
## Coefficients:
## (Intercept)  log(rulers)
##       8.632       -1.148
```

Get the slope

```
-1 * m$coefficients[2]
## log(rulers)
##    1.148083
```

Not to far away from Mandelbrot's D = 1.25 for the west coast of Britain.

Further reading.

CHAPTER THREE

3. ANALYSING SPECIES DISTRIBUTION DATA

3.1 Introduction

In this case-study 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 manipulation 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.

3.2 Import and prepare data

The data we will use is available in the rspatial package. First install that from github, using `devtools`.

```
if (!require("rspat")) remotes::install_github('rspatial/rspat')
## Loading required package: rspat
library(rspat)
```

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

```
f <- system.file("WILDPOT.txt", package="rspat")
basename(f)
## [1] "WILDPOT.txt"
d <- read.table(f, header=TRUE)
## Error in read.table(f, header = TRUE): more columns than column names
```

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

```
# read all lines using UTF-8 encoding
d <- readLines(f, encoding="UTF-8")
# 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)
```

(continues on next page)

(continued from previous page)

```
    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]    [,7]    [,8]   [,9]
## [1,] "ID"  "COLNR"    "DATE"     "LongD" "LongM" "LongS" "LongH" "LatD" "LatM"
## [2,] "55"  "OKA 3901" "19710405" "65"    "45"    "0"     "W"     "22"   "8"
## [3,] "16"  "OKA 3920" "19710406" "66"    "6"     "0"     "W"     "21"   "53"
## [4,] "204" "HOF 1848" "19710305" "65"    "5"     "0"     "W"     "22"   "16"
## [5,] "545" "OKA 4015" "19710411" "66"    "15"    "0"     "W"     "22"   "32"
## [6,] "549" "OKA 4026" "19710411" "66"    "12"    "0"     "W"     "22"   "30"
##      [,10]  [,11]  [,12]              [,13]       [,14]     [,15]   [,16]
## [1,] "LatS" "LatH" "SPECIES"          "SCODE_NEW" "SUB_NEW" "SP_ID" "COUNTRY"
## [2,] "0"    "S"    "S. acaule Bitter" "acl"       "ACL"     "1"     "ARGENTINA"
## [3,] "0"    "S"    "S. acaule Bitter" "acl"       "ACL"     "1"     "ARGENTINA"
## [4,] "0"    "S"    "S. acaule Bitter" "acl"       "ACL"     "1"     "ARGENTINA"
## [5,] "0"    "S"    "S. acaule Bitter" "acl"       "ACL"     "1"     "ARGENTINA"
## [6,] "0"    "S"    "S. acaule Bitter" "acl"       "ACL"     "1"     "ARGENTINA"
##      [,17]   [,18]
## [1,] "ADM1"  "ADM2"
## [2,] "Jujuy" "Yavi"
## [3,] "Jujuy" "Santa Catalina"
## [4,] "Salta" "Santa Victoria"
## [5,] "Jujuy" "Rinconada"
## [6,] "Jujuy" "Rinconada"
##      [,19]                                                  [,20]   [,21]
## [1,] "LOCALITY"                                             "PLRV1" "PLRV2"
## [2,] "Tafna."                                               "R"     "R"
## [3,] "10 km W of Santa Catalina."                           "S"     "R"
## [4,] "53 km E of Cajas."                                    "S"     "R"
## [5,] "\"Near Abra de Fundiciones, 10 km S of Rinconada.\"" "S"     "R"
## [6,] "8 km SW of Fundiciones."                              "S"     "R"
##      [,22]
## [1,] "FROST"
## [2,] "100"
## [3,] "100"
## [4,] "100"
## [5,] "100"
## [6,] "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.

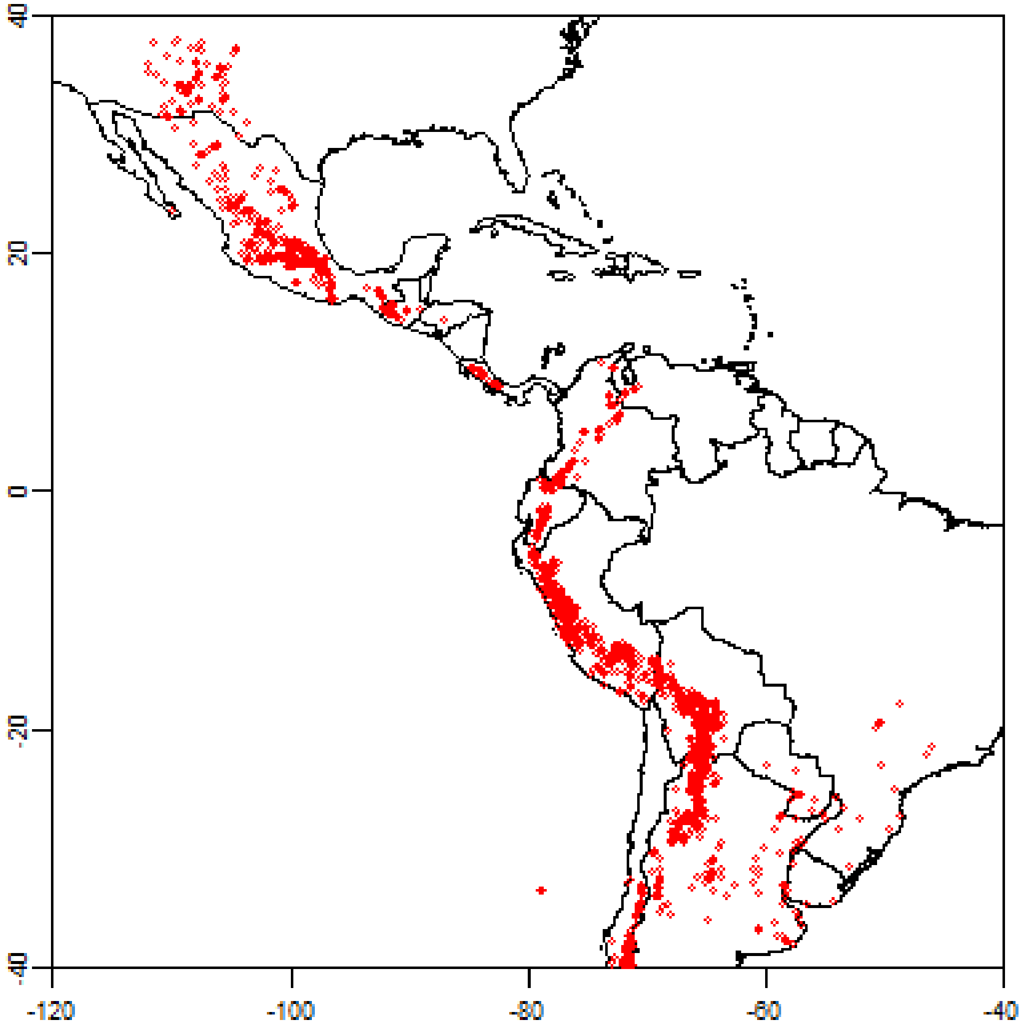
```
# first coerce character values to numbers
for (i in c('LongD', 'LongM', 'LongS', 'LatD', 'LatM', 'LatS')) {
    v[, i] <- as.numeric(v[,i])
}
v$lon <- -1 * (v$LongD + v$LongM / 60 + v$LongS / 3600)
v$lat <- v$LatD + v$LatM / 60 + v$LatS / 3600

# 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 LatS LatH
## 1  55  OKA 3901 19710405    65    45     0     W   22    8    0    S
## 2  16  OKA 3920 19710406    66     6     0     W   21   53    0    S
## 3 204  HOF 1848 19710305    65     5     0     W   22   16    0    S
## 4 545  OKA 4015 19710411    66    15     0     W   22   32    0    S
## 5 549  OKA 4026 19710411    66    12     0     W   22   30    0    S
## 6 551 OKA 4030A 19710411    66    12     0     W   22   28    0    S
##             SPECIES SCODE_NEW SUB_NEW SP_ID   COUNTRY  ADM1           ADM2
## 1 S. acaule Bitter        acl     ACL     1 ARGENTINA Jujuy           Yavi
## 2 S. acaule Bitter        acl     ACL     1 ARGENTINA Jujuy Santa Catalina
## 3 S. acaule Bitter        acl     ACL     1 ARGENTINA Salta Santa Victoria
## 4 S. acaule Bitter        acl     ACL     1 ARGENTINA Jujuy      Rinconada
## 5 S. acaule Bitter        acl     ACL     1 ARGENTINA Jujuy      Rinconada
## 6 S. acaule Bitter        acl     ACL     1 ARGENTINA Jujuy      Rinconada
##                                              LOCALITY PLRV1 PLRV2 FROST       lon
## 1                                              Tafna.     R     R   100 -65.75000
## 2                          10 km W of Santa Catalina.     S     R   100 -66.10000
## 3                                   53 km E of Cajas.     S     R   100 -65.08333
## 4 "Near Abra de Fundiciones, 10 km S of Rinconada."      S     R   100 -66.25000
## 5                             8 km SW of Fundiciones.     S     R   100 -66.20000
## 6                  "Salveayoc, 5 km SW of Rinconada."     S     R   100 -66.20000
##         lat
## 1 -22.13333
## 2 -21.88333
## 3 -22.26667
## 4 -22.53333
## 5 -22.50000
## 6 -22.46667
```

Get a SpatVector with most of the countries of the Americas.

```
cn <- spat_data("pt_countries")
class(cn)
## [1] "SpatVector"
## attr(,"package")
## [1] "terra"
```

Make a quick map

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



And create a SpatVector for the potato data with the formula approach

```
sp <- vect(v, crs="+proj=longlat +datum=WGS84")
```

3.3 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 SpatVector.

```
table(v$COUNTRY)
##
##     ARGENTINA       BOLIVIA        BRAZIL         CHILE      COLOMBIA
##          1474           985            17           100           107
##    COSTA RICA       ECUADOR     GUATEMALA      HONDURAS        Mexico
##            24           138            59             1             2
```

(continues on next page)

(continued from previous page)

```
##        MEXICO         PANAMA       PARAGUAY           Peru           PERU
##           843             13             19              1           1043
## UNITED STATES        URUGUAY      VENEZUELA
##           157              4             12
# note Peru and PERU
v$COUNTRY <- toupper(v$COUNTRY)
table(v$COUNTRY)
##
##     ARGENTINA        BOLIVIA         BRAZIL          CHILE       COLOMBIA
##          1474            985             17            100            107
##    COSTA RICA        ECUADOR      GUATEMALA       HONDURAS         MEXICO
##            24            138             59              1            845
##        PANAMA       PARAGUAY           PERU  UNITED STATES        URUGUAY
##            13             19           1044            157              4
##     VENEZUELA
##            12

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

Below we determine the country using a spatial query, using the `intersect` method.

```
vv <- intersect(sp[, "COUNTRY"], cn)
names(vv)[1] <- "ptCountry"
head(vv)
##   ptCountry   COUNTRY
## 1 ARGENTINA ARGENTINA
## 2 ARGENTINA ARGENTINA
## 3 ARGENTINA ARGENTINA
## 4 ARGENTINA ARGENTINA
## 5 ARGENTINA ARGENTINA
## 6 ARGENTINA ARGENTINA
table(vv$COUNTRY)
##
##          ARGENTINA              BOLIVIA               BRASIL                CHILE
##               1473                  985                   17                   94
##           COLOMBIA           COSTA RICA              ECUADOR            GUATEMALA
##                104                   24                  139                   58
##           HONDURAS               MEXICO               PANAMA             PARAGUAY
##                  1                  846                   13                   19
##               PERU  UNITED STATES, THE              URUGUAY            VENEZUELA
##               1042                  157                    4                   14
```

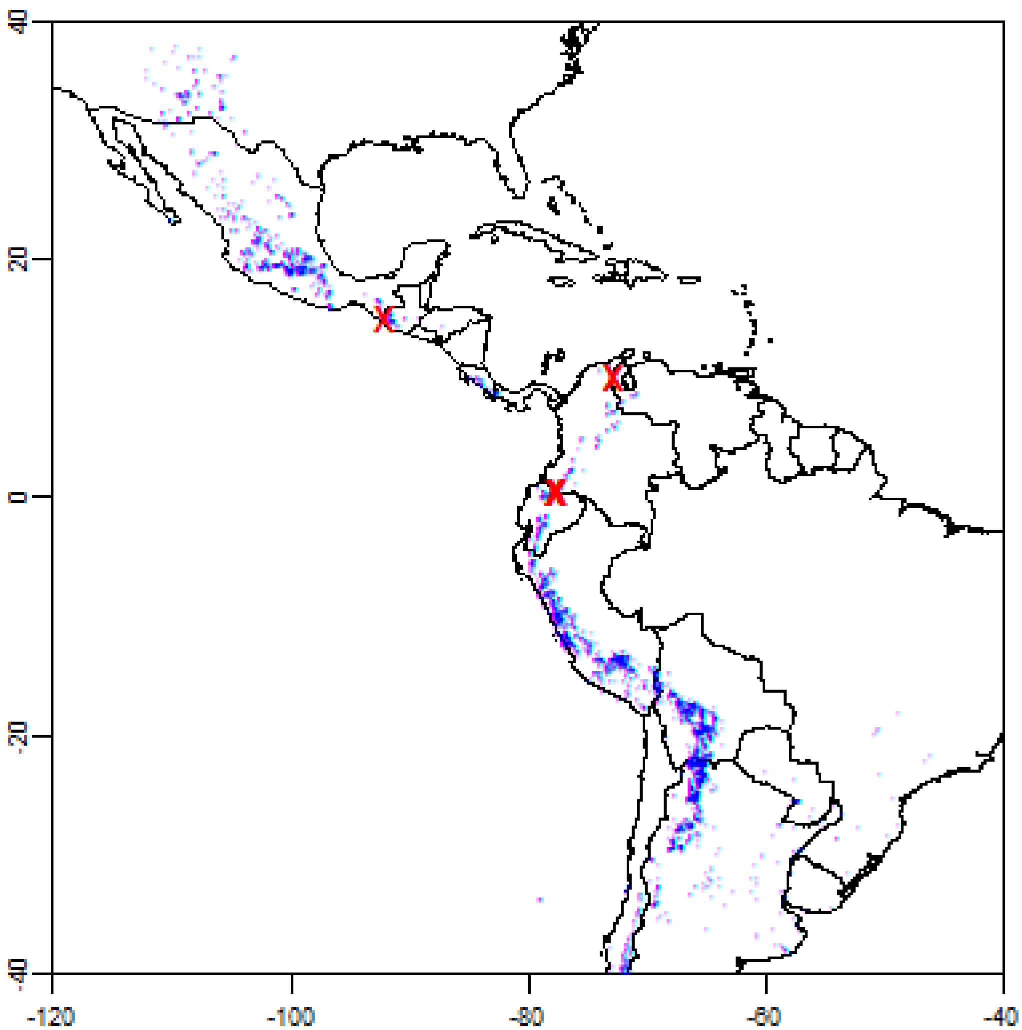
This table is similar to the previous table, but it is not the same. Let's find the records that are not in the same country according to the original data and the spatial query.

```
# some fixes first
# apparently in the ocean (small island missing from polygon data)
vv$COUNTRY[is.na(vv$COUNTRY)] <- ""
# some spelling differenes
vv$COUNTRY[vv$COUNTRY=="UNITED STATES, THE"] <- "UNITED STATES"
vv$COUNTRY[vv$COUNTRY=="BRASIL"] <- "BRAZIL"

i <- which(toupper(vv$ptCountry) != vv$COUNTRY)
i
## [1] 2583 2724 2767 3291 4978 4988
```

(continues on next page)

(continued from previous page)

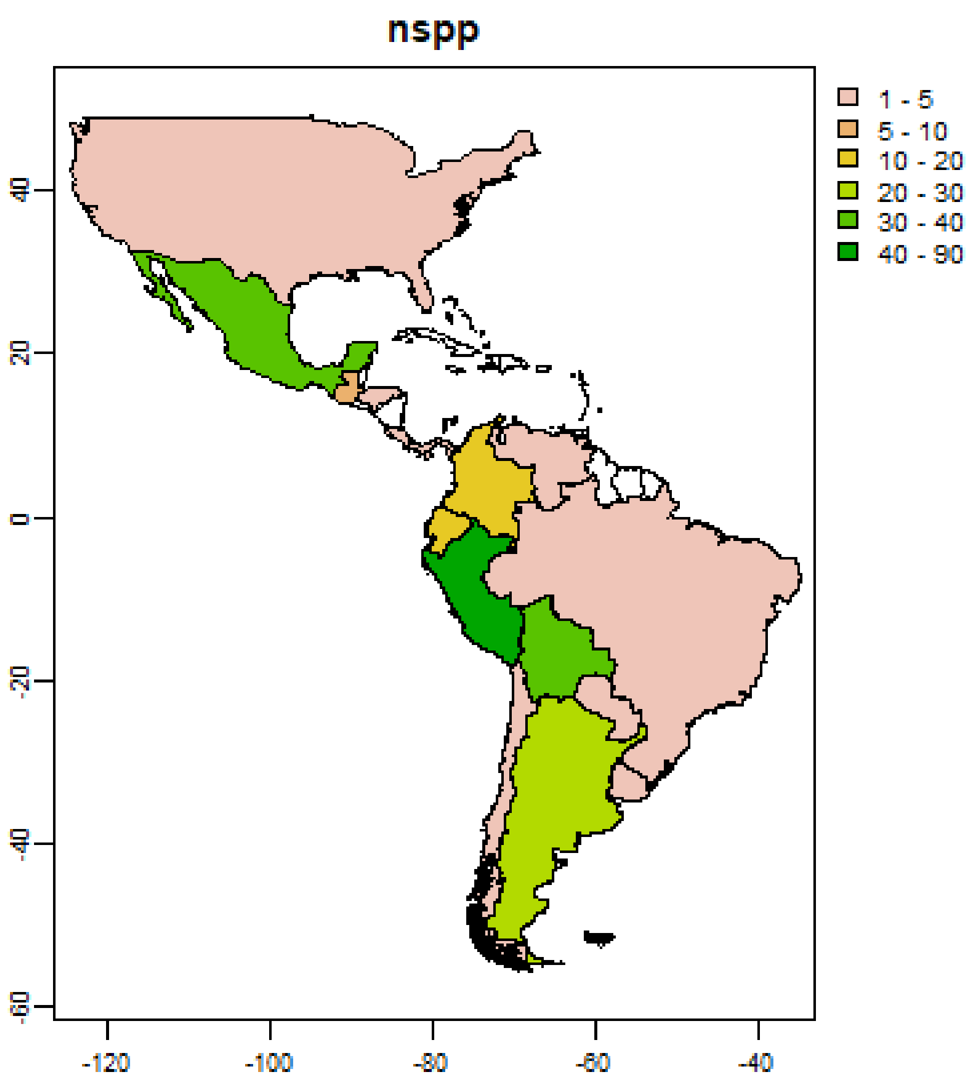
```
as.data.frame(vv[i,])
##   ptCountry   COUNTRY
## 1   ECUADOR  COLOMBIA
## 2  COLOMBIA   ECUADOR
## 3  COLOMBIA   ECUADOR
## 4 GUATEMALA    MEXICO
## 5  COLOMBIA VENEZUELA
## 6  COLOMBIA VENEZUELA
plot(cn, xlim=c(-120, -40), ylim=c(-40,40), axes=TRUE)
points(sp, cex=.25, pch='+', col='blue')

points(vv[i,], col='red', pch='x', cex=1.5)
```



All observations that are in a different country than their attribute data suggests are very close to an international border, or in the water. That suggests that the coordinates of the potato locations are not very precise (or the borders are inexact). Otherwise, this is reassuring (and a-typical). There are often are several inconsistencies, and it can be hard to find out whether the locality coordinates are wrong or whether the borders are wrong; but further inspection is warranted in those cases.

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)

# merge with country SpatVector --- fix the names in the polygons this time
cn$COUNTRY[cn$COUNTRY=="UNITED STATES, THE"] <- "UNITED STATES"
cn$COUNTRY[cn$COUNTRY=="BRASIL"] <- "BRAZIL"

cns <- merge(cn, spc, by="COUNTRY", all.x=TRUE)
plot(cns, "nspp", col=rev(terrain.colors(25)), breaks=c(1,5,10,20,30,40,90))
```



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

```
tb <- table(v[ c('COUNTRY', 'SPECIES')])
```

(continues on next page)

(continued from previous page)

```
# a big table
dim(tb)
## [1]  16 195
# show two columns
tb[,2:3]
##                  SPECIES
## COUNTRY           S. acaule Bitter             S. achacachense C<U+643C><U+3E66>
↪rdenas
##   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
```

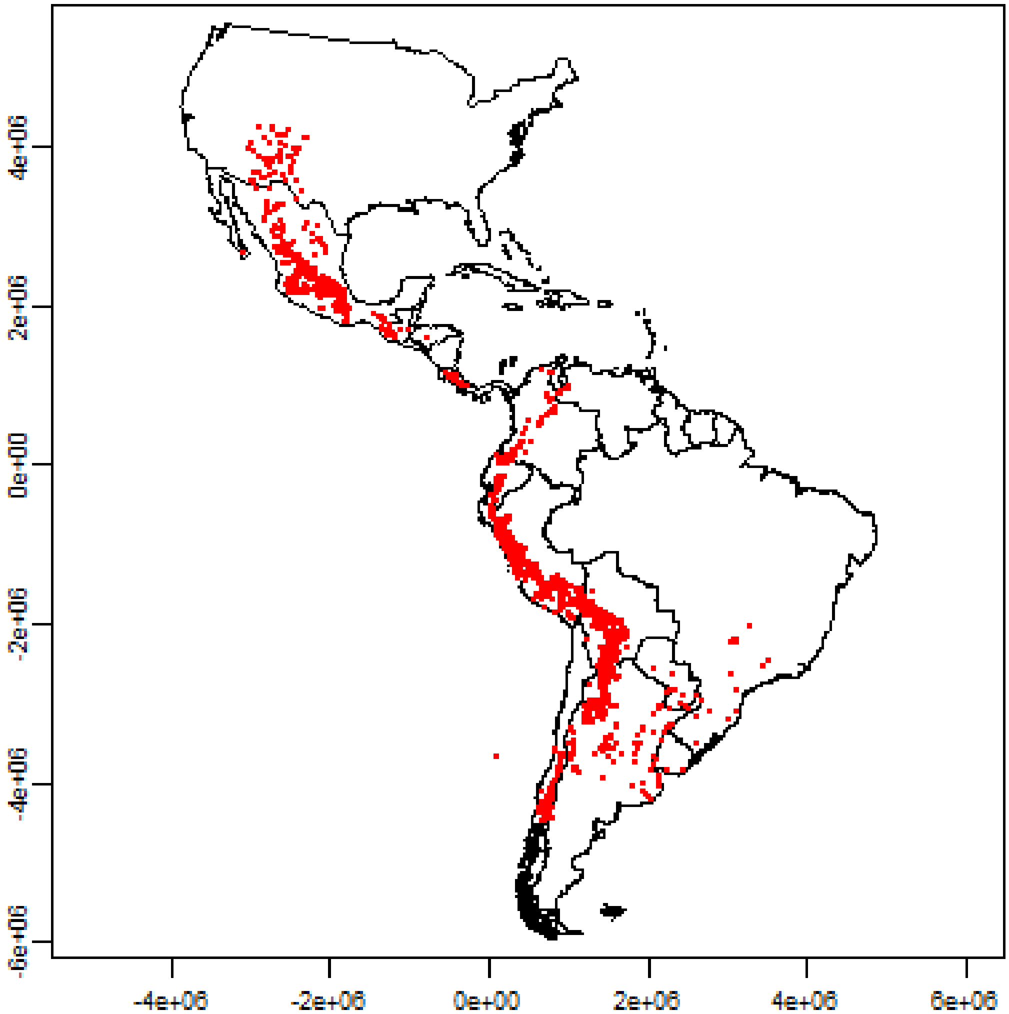
Because the countries have such different sizes and shapes, the comparison is not fair (larger countries will have more species, on average, than smaller countries). Some countries are also very large, hiding spatial variation. The map the number of species, it is in most cases better to use a raster (grid) with cells of equal area, and that is what we will do next.

3.4 Projecting spatial data

To use a raster with equal-area cells, 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, a reasonable location is (-80, 0).

```
# the CRS we want
laea <-"+proj=laea  +lat_0=0 +lon_0=-80"
clb <- project(cn, laea)
pts <- project(sp, laea)
plot(clb)
points(pts, col='red', cex=.5)
```



Note that the shape of the countries is now much more similar to their shape on a globe than before we projected 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) in meters.

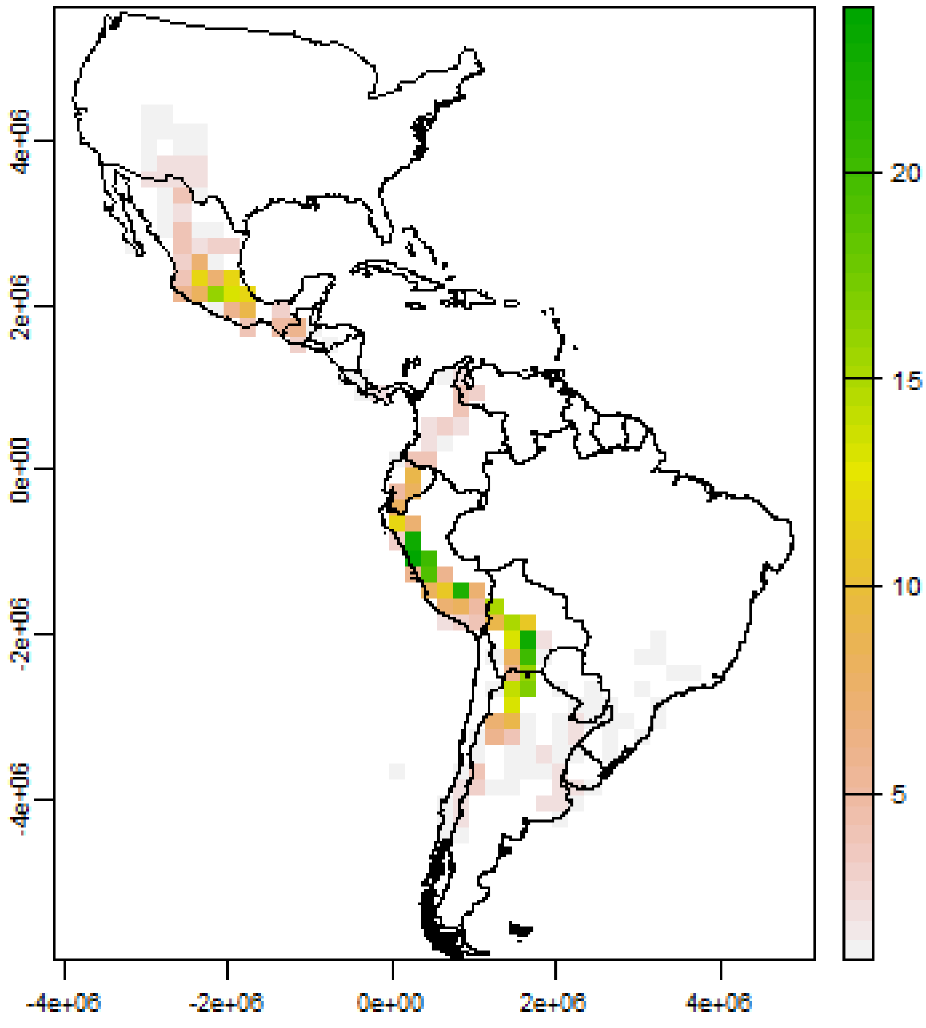
3.5 Species richness

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

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

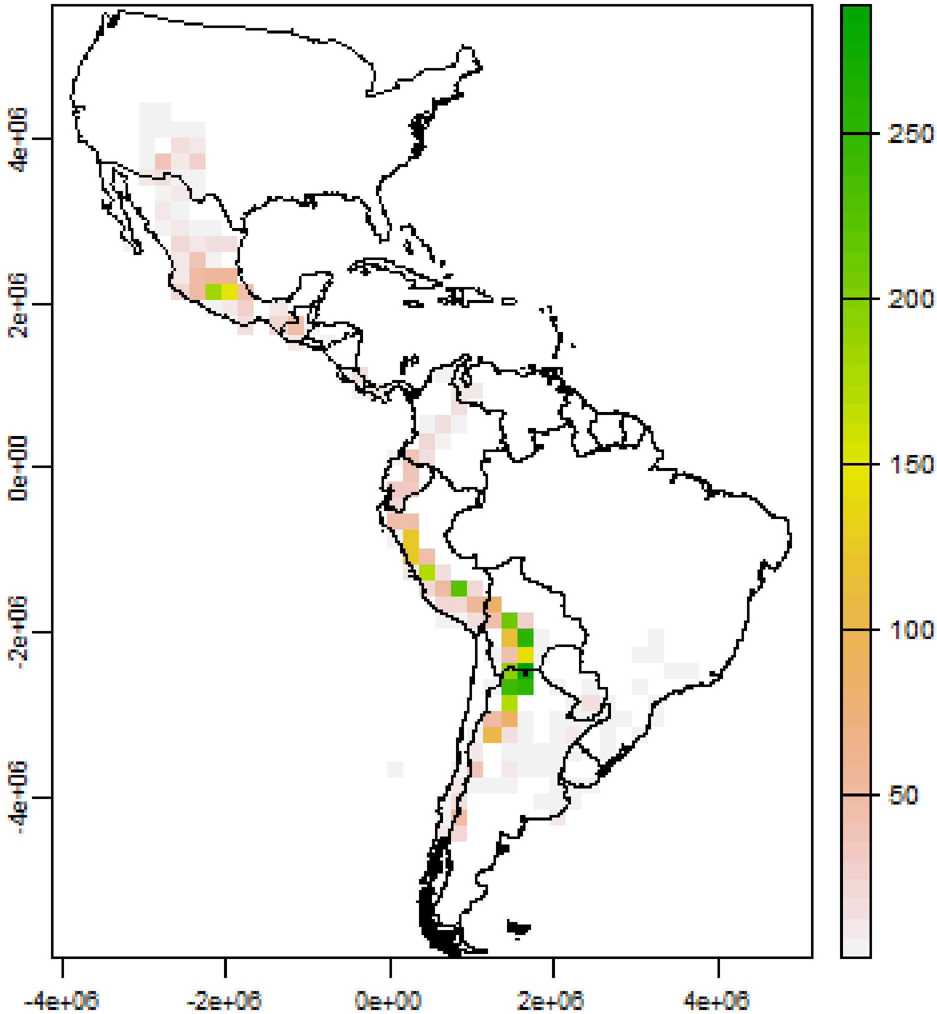
Now compute the number of observations and the number of species richness for each cell.

```
rich <- rasterize(pts, r, "SPECIES", function(x, ...) length(unique(na.omit(x))))
plot(rich)
lines(clb)
```



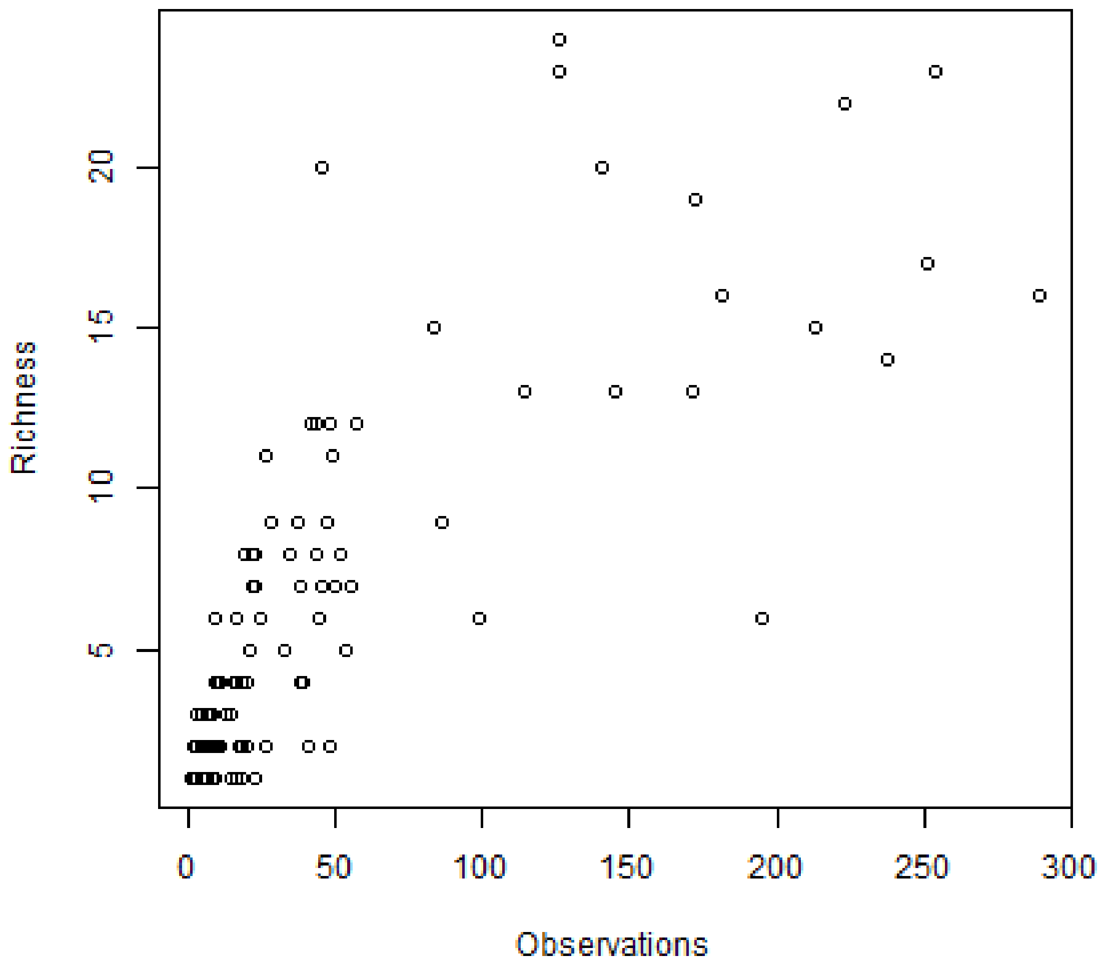
Now we make a raster of the number of observations.

```
obs <- rasterize(pts, r, field="SPECIES", fun=function(x, ...)length((na.omit(x))) )
plot(obs)
lines(clb)
```



A cell by cell comparison of the number of species and the number of observations.

```
plot(obs, rich, cex=1, xlab="Observations", ylab="Richness")
```

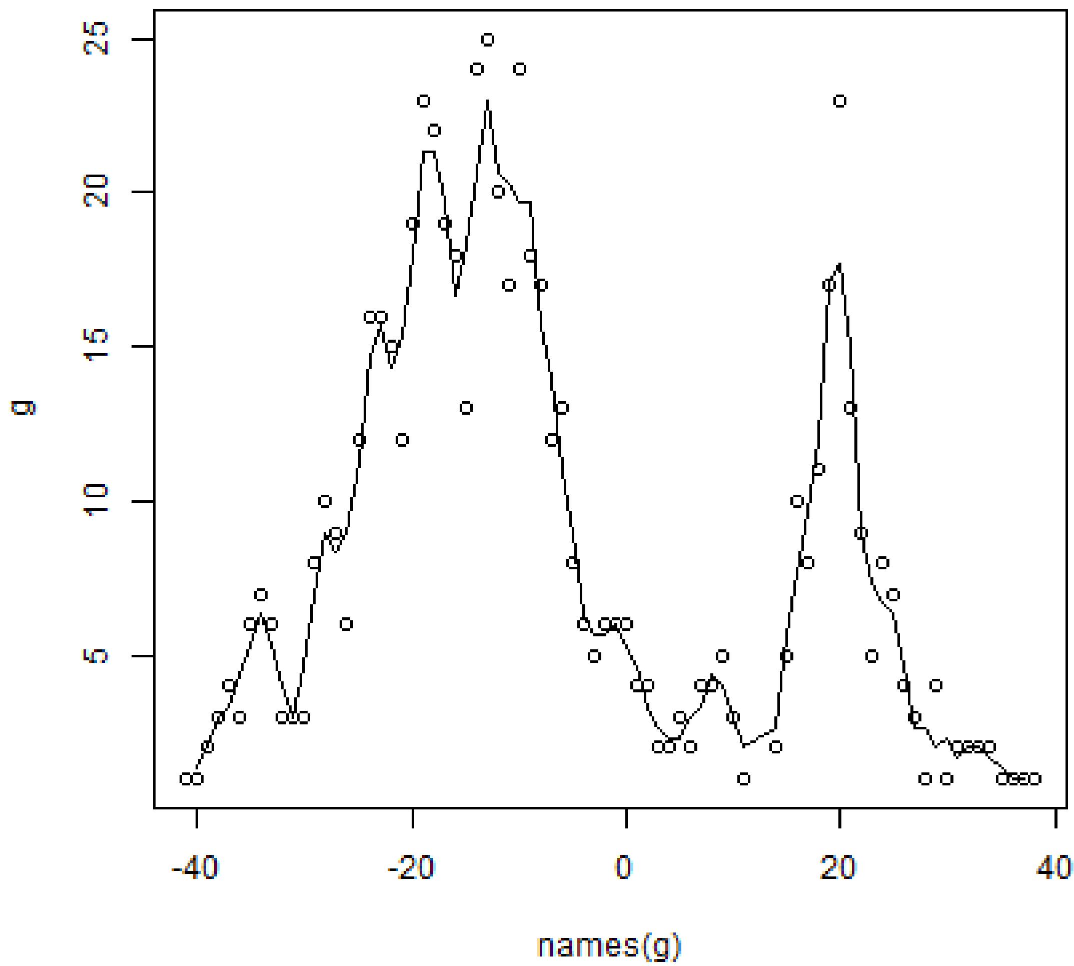


Clearly there is an association between the number of observations and the number of species. It may be that the number of species in some places is inflated just because more research was done there.

The problem is that this association will almost always exist. When there are only few species in an area, researchers 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 a good thing). Ways to correct for this 'collector-bias' include the use of techniques such as 'rarefaction' and '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), raster::movingFun(g, 3))
```



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

3.6 Range size

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 compute these things from the original longitude/latitude data.

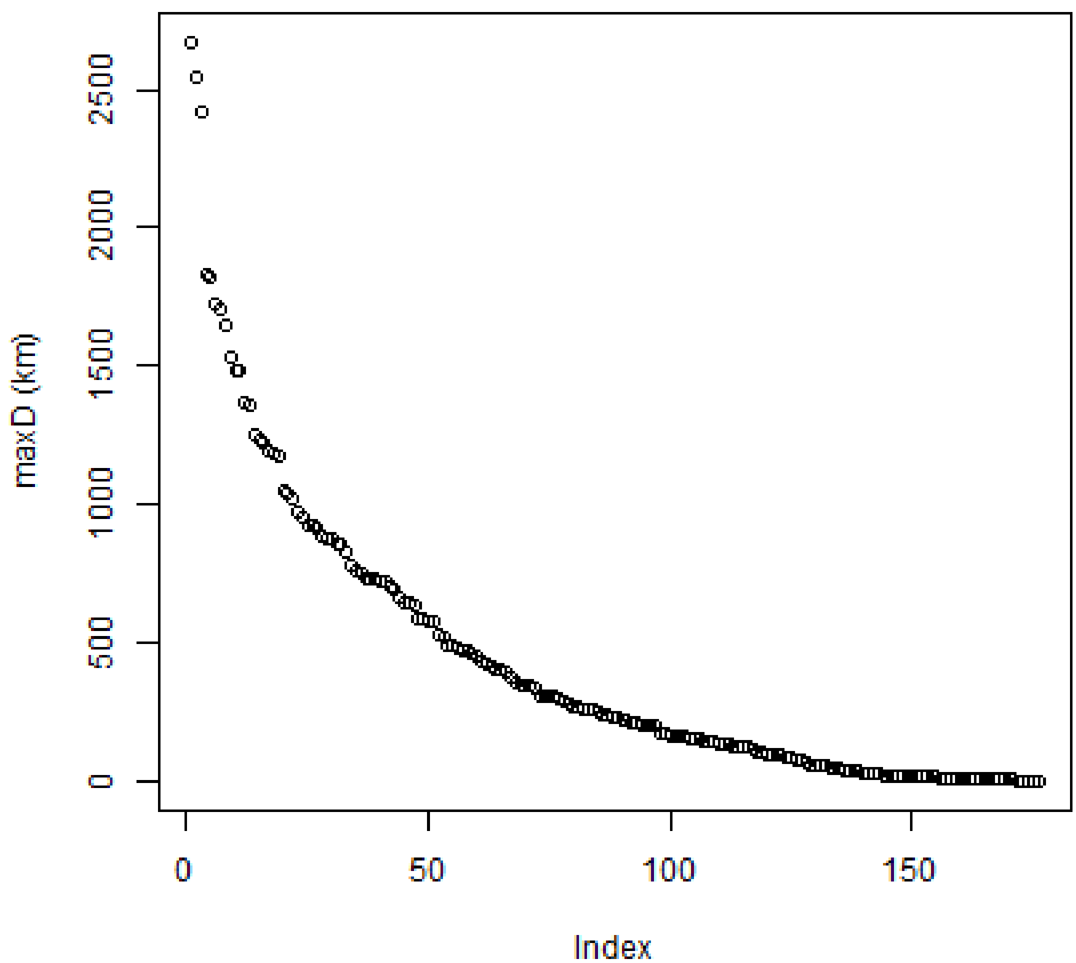
Compute maxD for each species

```
spp <- unique(pts$SPECIES)
maxD <- rep(NA, length(spp))
for (s in 1:length(spp)) {
    # get the coordinates for species 's'
```

(continues on next page)

(continued from previous page)

```
    p <- pts[pts$SPECIES == spp[s], ]
    if (nrow(p) < 2) next
    # distance matrix
    d <- as.matrix(distance(p))
    # ignore the distance of a point to itself
    diag(d) <- NA
    # get max value
    maxD[s] <- max(d, na.rm=TRUE)
}

# Note the typical J shape
plot(rev(sort(maxD))/1000, ylab="maxD (km)")
```



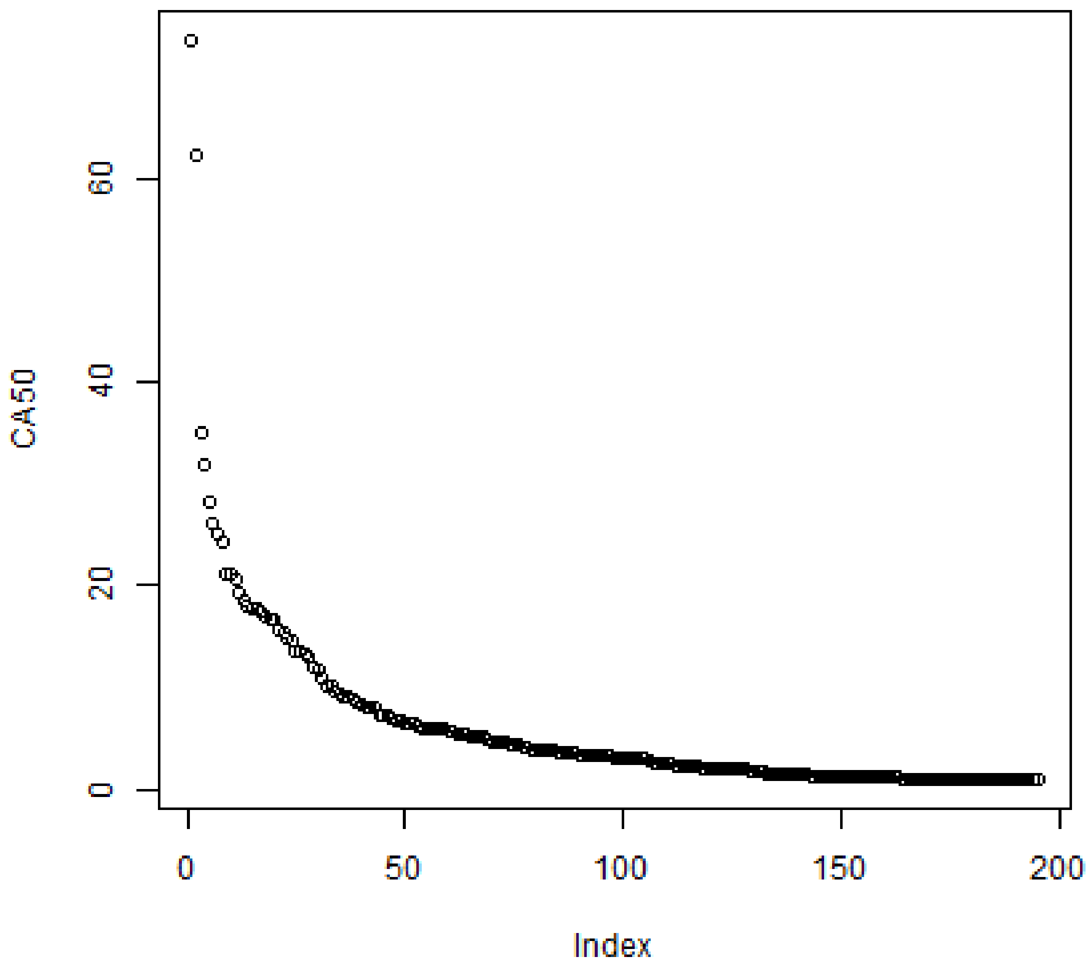
Compute CA

```
CA <- rep(NA, length(spp))
```

(continues on next page)

(continued from previous page)

```
for (s in 1:length(spp)) {
    p <- pts[pts$SPECIES == spp[s], ]
    # run "circles" model
    m <- aggregate(buffer(p, 50000))
    CA[s] <- area(m)
}
# standardize to the size of one circle
CA <- CA / (pi * 50000^2)
plot(rev(sort(CA)), ylab='CA50')
```



Make convex hull range polygons

```
hull <- list()
for (s in 1:length(spp)) {
    p <- unique(pts[pts$SPECIES == spp[s], ])
    # need at least three (unique) points for hull
```

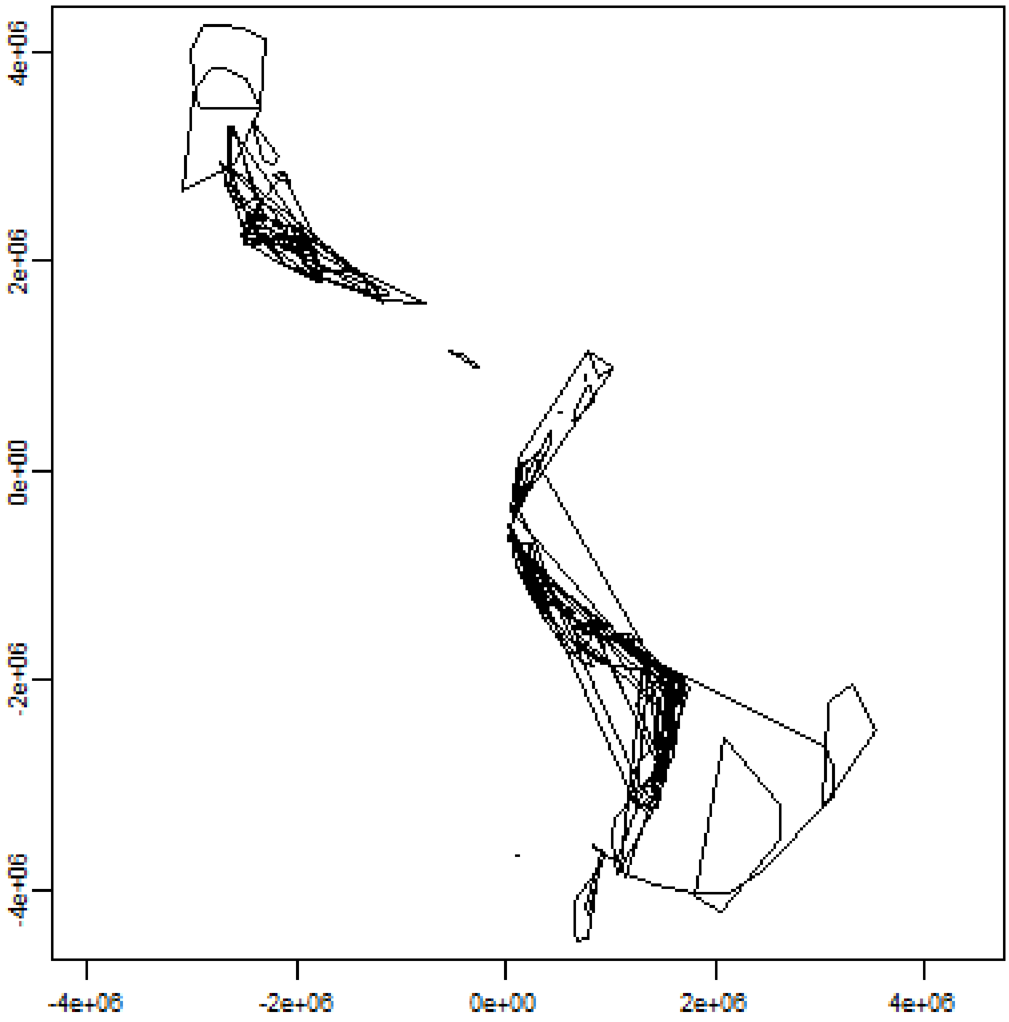
(continues on next page)

(continued from previous page)

```
    if (nrow(p) > 3) {
        h <- convexhull(p)
        if (geomtype(h) == "polygons") {
        hull[[s]] <- h
        }
    }
}
```

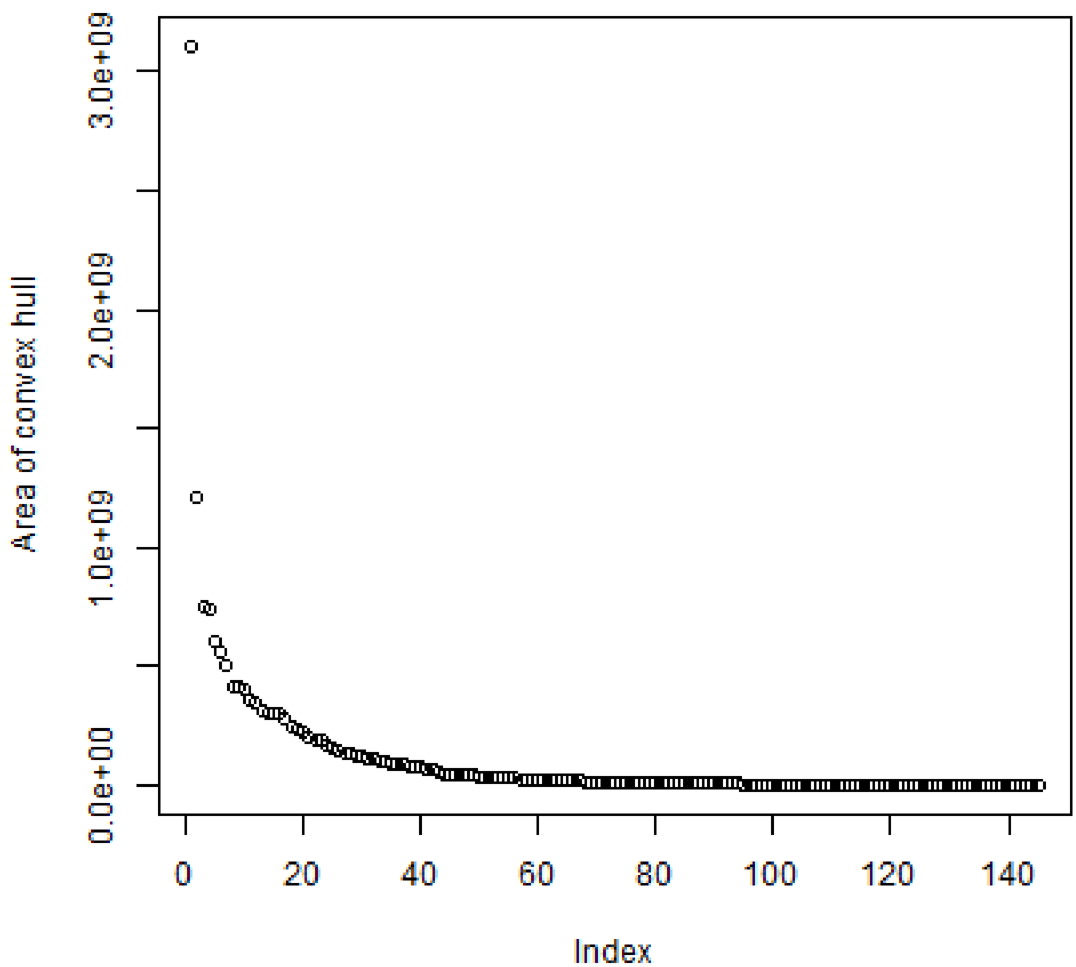
Plot the hulls. First remove the empty hulls (you cannot make a hull if you do not have at least three points).

```
# which elements are NULL
i <- which(!sapply(hull, is.null))
h <- hull[i]
# combine them
hh <- do.call(rbind, h)
plot(hh)
```



Get the area for each hull, taking care of the fact that some are NULL.

```
ahull <- area(hh)
plot(rev(sort(ahull))/1000, ylab="Area of convex hull")
```

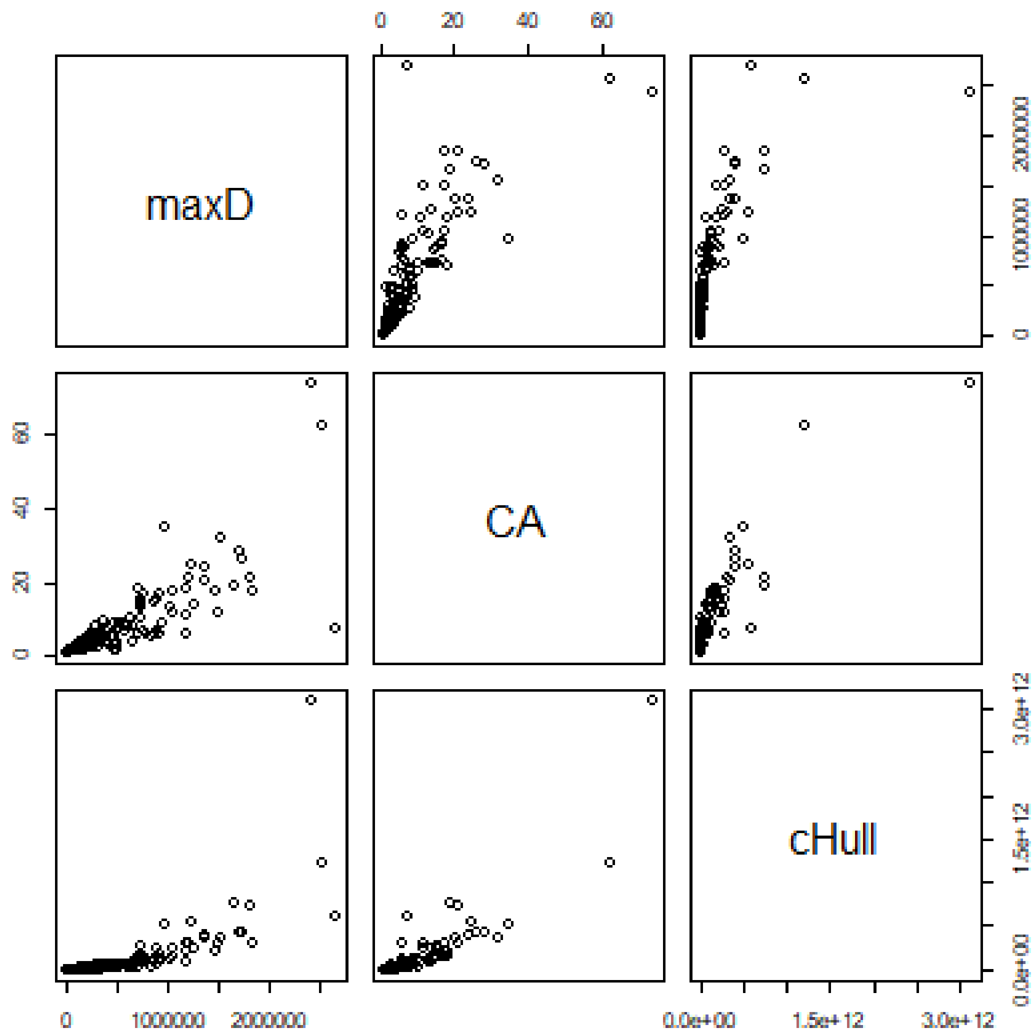


To get a value (even if NA) for all species

```
cHull <- rep(NA, length(spp))
cHull[i] <- ahull
```

Compare all three measures

```
d <- cbind(maxD, CA, cHull)
pairs(d)
```



3.7 Exercises

3.7.1 Exercise 1. Mapping species richness at different resolutions

Make maps of the number of observations and of species richness at 50, 100, 250, and 500 km resolution. Discuss the differences.

3.7.2 Exercise 2. Mapping diversity

Make a map of Shannon Diversity *H* for the potato data, at 200 km resolution.

a) First make a function that computes Shannon Diversity (H) from a vector of species names

`H = -SUM(p * ln(p))`

Where `p` is proportion of each species

To get `p`, you can do

`vv <- as.vector(table(v$SPECIES)) p <- vv / sum(vv)`

b) now use the function

3.7.3 Exercise 3. Mapping traits

There is information about two traits in the data set in field PRLV (tolerance to Potato Leaf Roll Virus) and frost (frost tolerance). Make a map of average frost tolerance.

3.8 References

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