Ejemplo: Matriz de riqueza con loops

Para este ejemplo se necesitarán los paquetes:

"foreign","sp","rgdal","rgeos" y "raster"

```
#setear el directorio de trabajo apropiadamente
setwd("~/backupOct2019/RLadiesTalleres/CursoRladiesRcomolenguaje/ArchivosdbfparaR/")
```

```
library("foreign") #esta librería la necesito para leer los "data base files" (.dbf)
#ejecuto desde R el comando del sistema "ls" que me lista los archivos con extensión .dbf y guardo los
listdbf=list.files(path = ".",pattern=".dbf")
#hago un for desde 1 hasta la cantidad de nombres que tengo en listdbf que será "length(listdbf)"
listdbf
```

```
##   [1] "alou_belz_pl_sj.dbf"  "alou_cara_pl_sj.dbf"  "alou_coib_pl_sj.dbf"
##   [4] "alou_guar_pl_sj.dbf"  "alou_nige_pl_sj.dbf"  "alou_pall_pl_sj.dbf"
##   [7] "alou_pigr_pl_sj.dbf"  "alou_sara_pl_sj.dbf"  "alou_seni_pl_sj.dbf"
##  [10] "alou_ulul_pl_sj.dbf"  "aotu_azar_pl_sj.dbf"  "aotu_infu_pl_sj.dbf"
##  [13] "aotu_lemu_pl_sj.dbf"  "aotu_mico_pl_sj.dbf"  "aotu_nanc_pl_sj.dbf"
##  [16] "aotu_nigr_pl_sj.dbf"  "aotu_triv_pl_sj.dbf"  "aotu_voci_pl_sj.dbf"
##  [19] "atel_belz_pl_sj.dbf"  "atel_cham_pl_sj.dbf"  "atel_fusc_pl_sj.dbf"
##  [22] "atel_geof_pl_sj.dbf"  "atel_hybr_pl_sj.dbf"  "atel_marg_pl_sj.dbf"
##  [25] "atel_pani_pl_sj.dbf"  "brac_arac_pl_sj.dbf"  "brac_hypo_pl_sj.dbf"
##  [28] "caca_calv_pl_sj.dbf"  "caca_mela_pl_sj.dbf"  "call_auri_pl_sj.dbf"
##  [31] "call_bapt_pl_sj.dbf"  "call_barb_pl_sj.dbf"  "call_bern_pl_sj.dbf"
##  [34] "call_brun_pl_sj.dbf"  "call_cali_pl_sj.dbf"  "call_cine_pl_sj.dbf"
##  [37] "call_coim_pl_sj.dbf"  "call_cupr_pl_sj.dbf"  "call_disc_pl_sj.dbf"
##  [40] "call_dona_pl_sj.dbf"  "call_dubi_pl_sj.dbf"  "call_flav_pl_sj.dbf"
##  [43] "call_geof_pl_sj.dbf"  "call_goel_pl_sj.dbf"  "call_hoff_pl_sj.dbf"
##  [46] "call_jacc_pl_sj.dbf"  "call_kuhl_pl_sj.dbf"  "call_luci_pl_sj.dbf"
##  [49] "call_luge_pl_sj.dbf"  "call_mede_pl_sj.dbf"  "call_mela_pl_sj.dbf"
##  [52] "call_mode_pl_sj.dbf"  "call_molo_pl_sj.dbf"  "call_nigr_pl_sj.dbf"
##  [55] "call_oena_pl_sj.dbf"  "call_olal_pl_sj.dbf"  "call_orna_pl_sj.dbf"
##  [58] "call_pall_pl_sj.dbf"  "call_peni_pl_sj.dbf"  "call_pers_pl_sj.dbf"
##  [61] "call_puri_pl_sj.dbf"  "call_regu_pl_sj.dbf"  "call_step_pl_sj.dbf"
##  [64] "call_torq_pl_sj.dbf"  "cebu_albi_pl_sj.dbf"  "cebu_apel_pl_sj.dbf"
##  [67] "cebu_capu_pl_sj.dbf"  "cebu_kaap_pl_sj.dbf"  "cebu_libi_pl_sj.dbf"
##  [70] "cebu_macr_pl_sj.dbf"  "cebu_nigr_pl_sj.dbf"  "cebu_oliv_pl_sj.dbf"
##  [73] "cebu_pygm_pl_sj.dbf"  "cebu_robu_pl_sj.dbf"  "cebu_xant_pl_sj.dbf"
##  [76] "chir_albi_pl_sj.dbf"  "chir_chir_pl_sj.dbf"  "chir_sagu_pl_sj.dbf"
##  [79] "Chir_sata_pl_sj.dbf"  "chir_utah_pl_sj.dbf"  "lago_cana_pl_sj.dbf"
##  [82] "lago_lago_pl_sj.dbf"  "lago_luge_pl_sj.dbf"  "lago_poep_pl_sj.dbf"
##  [85] "leon_cais_pl_sj.dbf"  "leon_chry1_pl_sj.dbf" "leon_chry2_pl_sj.dbf"
##  [88] "Leon_rosa_pl_sj.dbf"  "mico_acar_pl_sj.dbf"  "mico_arge_pl_sj.dbf"
##  [91] "mico_chry_pl_sj.dbf"  "mico_emil_pl_sj.dbf"  "mico_hume_pl_sj.dbf"
##  [94] "mico_humi_pt_sj.dbf"  "mico_inte_pl_sj.dbf"  "mico_leuc_pl_sj.dbf"
##  [97] "mico_mani_pl_sj.dbf"  "mico_marc_pl_sj.dbf"  "mico_maue_pl_sj.dbf"
## [100] "mico_mela_pl_sj.dbf"  "mico_nigr_pl_sj.dbf"  "mico_sate_pl_sj.dbf"
## [103] "oreo_flav_pl_sj.dbf"  "pith_aequ_pl_sj.dbf"  "pith_albi_pl_sj.dbf"
## [106] "pith_irro_pl_sj.dbf"  "pith_mona_pl_sj.dbf"  "pith_pith_pl_sj.dbf"
## [109] "sagu_bico_pl_sj.dbf"  "sagu_fusc_pl_sj.dbf"  "sagu_geof_pl_sj.dbf"
## [112] "sagu_grae_pl_sj.dbf"  "sagu_impe_pl_sj.dbf"  "sagu_inus_pl_sj.dbf"
## [115] "sagu_labi_pl_sj.dbf"  "sagu_leuc_pl_sj.dbf"  "sagu_mart_pl_sj.dbf"
```

```
## [118] "sagu_mida_pl_sj.dbf"  "sagu_myst_pl_sj.dbf"  "Sagu_nige_pl_sj.dbf"
## [121] "sagu_nigr_pl_sj.dbf"  "sagu_oedi_pl_sj.dbf"  "sagu_trip_pl_sj.dbf"
## [124] "saim_boli_pl_sj.dbf"  "saim_oers_pl_sj.dbf"  "saim_sciu_pl_sj.dbf"
## [127] "saim_ustu_pl_sj.dbf"  "saim_vanz_pl_sj.dbf"
```

```
#defino el numero de columnas de salida de la matriz que será el número de archivos mas 2 columnas corr
numcolsalida=length(listdbf)+2
#IMPORTANTE: en este caso cada .dbf corresponde a la matriz de presencia/ausencia de un sola especie. E
explorounamatriz<-read.dbf(listdbf[[1]],as.is=TRUE)
numfilasalida=nrow(explorounamatriz)
mimatriz<-matrix(ncol=numcolsalida,nrow=numfilasalida)
mimatriz<-cbind(explorounamatriz$X,explorounamatriz$Y)

nombrecolumna<-vector()
nombrecolumna<-c("X","Y")
#defino una lista que la llamo list1
list1<-list()

for(i in 1:length(listdbf))
 {
  #guardo un dbf en cada elemento de la lista list1
  list1[[i]]<-read.dbf(listdbf[[i]],as.is=TRUE)
  #reemplazo los valores mayores que 1 en la columna de presencia/ausencia por 1
  list1[[i]]$PRESENCE[which(list1[[i]]$PRESENCE>1)]<-1
  mimatriz<-cbind(mimatriz, list1[[i]]$PRESENCE)
  nombrecolumna<-c(nombrecolumna,unique(list1[[i]]$ENGL_NAME)[2])
 }
# luego, en cada elemento de la lista list1 tendré un archivo .dbf
colnames(mimatriz)<-nombrecolumna
```

Ahora tengo una matriz llamada "mimatriz" cuyas columnas son [x,y,especie1,especie2,especie3.....] y cada elemento es un 1 o un cero según la especie esté o no esté en la coordenada x,y correspondiente. Entonces sumando por filas obtengo la "riqueza" de especies en cada sitio x,y.

La suma por filas puede hacerse con un loop o bien utilizando la función llamada apply que ya viene predefinida en R-base. Esta función apply no es mas que un loop "optimizado".

Veamos la forma de hacerlo con apply:

```
mi_matriz_sin_coord<-mimatriz[,3:ncol(mimatriz)]
system.time( riqueza_apply<-apply(mi_matriz_sin_coord,1,sum) )
```

```
##    user  system elapsed
##   0.011   0.000   0.010
```

Ahora haremos lo mismo con un loop:

```
riqueza_vector<-vector()
system.time(
for(j in 1:nrow(mi_matriz_sin_coord))
{
riqueza_vector[j]<-sum(mi_matriz_sin_coord[j,])
}
)
```

```
##    user  system elapsed
##   0.013   0.000   0.013
```

Los tiempos no difieren mucho (para matrices más grandes el apply debería ser más rápido)

Verifiquemos que ambos vectores sean iguales, una forma es:

```
resta<-riqueza_apply-riqueza_vector
summary(resta) #debería dar todo cero si son iguales los vectores
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0       0       0       0       0       0
```

######Dibujemos todo esto con la ayuda de Paula ;-)

```
#Cargá los  paquetes sp,rgdal, rgeos y raster
library("rgdal")
```

```
## Loading required package: sp
```

```
## rgdal: version: 1.4-7, (SVN revision 845)
##  Geospatial Data Abstraction Library extensions to R successfully loaded
##  Loaded GDAL runtime: GDAL 2.2.3, released 2017/11/20
##  Path to GDAL shared files: /usr/share/gdal/2.2
##  GDAL binary built with GEOS: TRUE
##  Loaded PROJ.4 runtime: Rel. 4.9.3, 15 August 2016, [PJ_VERSION: 493]
##  Path to PROJ.4 shared files: (autodetected)
##  Linking to sp version: 1.3-2
```

```
library("sp")
library("rgeos")
```

```
## rgeos version: 0.5-2, (SVN revision 621)
##  GEOS runtime version: 3.6.2-CAPI-1.10.2
##  Linking to sp version: 1.3-2
##  Polygon checking: TRUE
```

```
library("raster")
#Definir las coordenadas que tendrá el objeto
df=as.data.frame(mimatriz)
x= df$X
y=df$Y
coords = data.frame(x, y)
head(coords)
```

```
##          x        y
## 1 -5049357 -6319778
## 2 -4939357 -6319778
## 3 -4829357 -6319778
## 4 -5269357 -6209778
## 5 -5159357 -6209778
## 6 -5049357 -6209778
```

```
#Convertir las coordenadas a un objeto espacial con la función "coordinates"
coordinates(coords) =  ~x + y

#Ver un sumario de las coordenadas "coords" para ver que se creó un objeto espacial de puntos.
summary(coords)
```

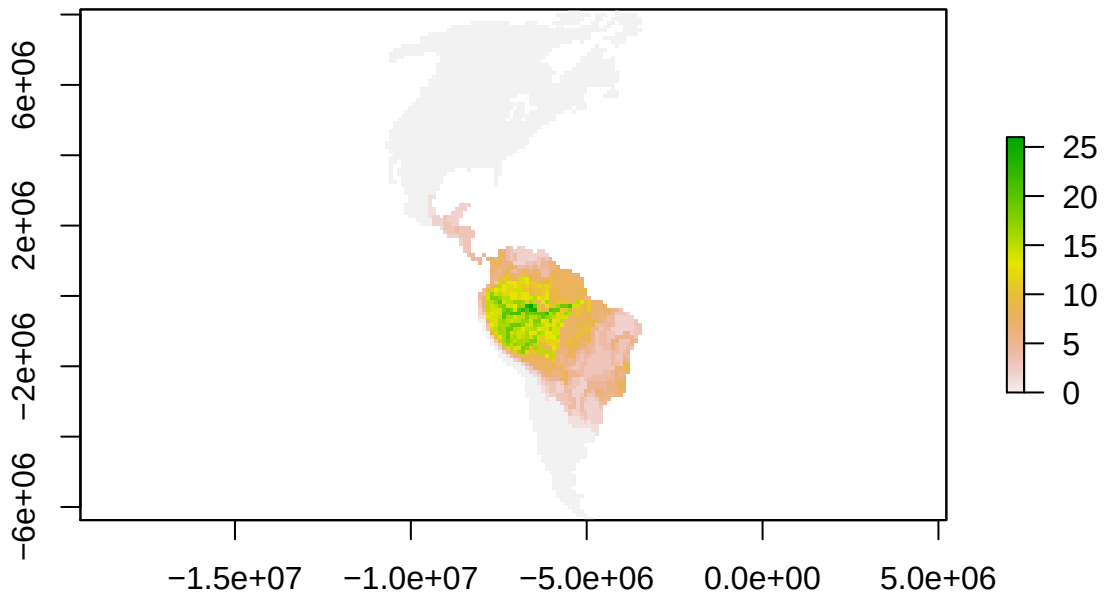
```
## Object of class SpatialPoints
## Coordinates:
##         min      max
```

```
## x -10659357 -3509357
## y  -6319778  8090222
## Is projected: NA
## proj4string : [NA]
## Number of points: 3235
```

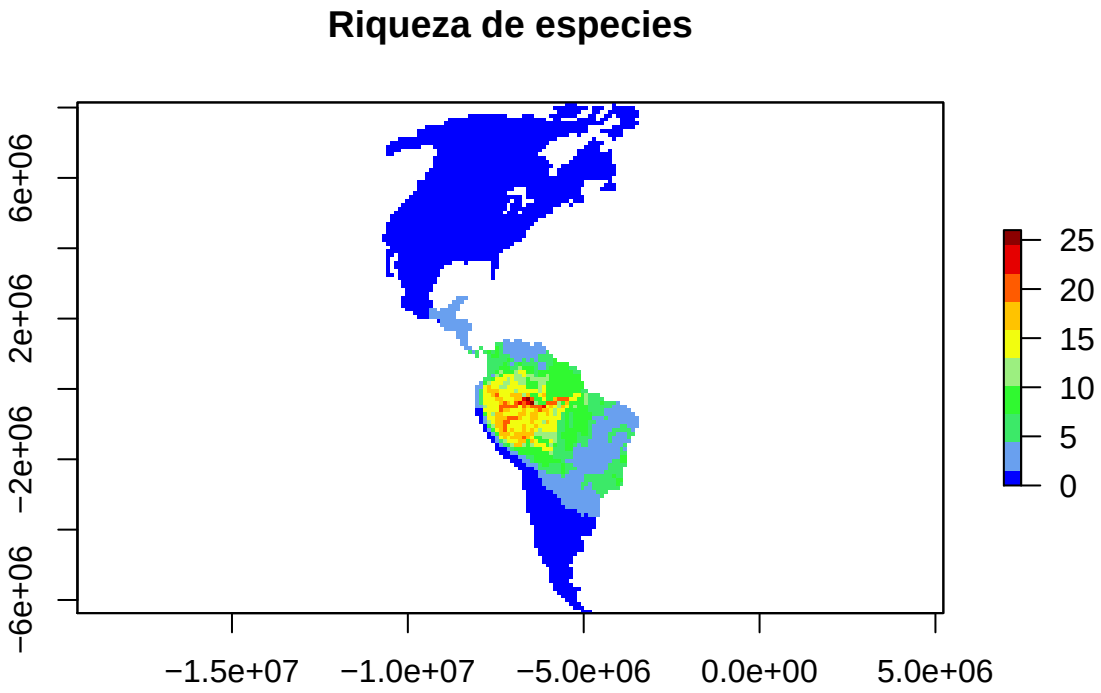
```
#Agregar a "coords" los valores de riqueza
Riqueza_points= SpatialPointsDataFrame(coords, data.frame(riqueza_apply))

#Especificar la proyección del objeto espacial
proj4string(Riqueza_points)=CRS("+proj=moll +lon_0=0 +x_0=0 +y_0=0
+ellps=WGS84 +datum=WGS84 +units=m +no_defs")

#Ver la tabla de atributos del objeto espacial
View(Riqueza_points)

#Chequear que la riqueza está contenida en un objeto espacial de puntos
summary(Riqueza_points)
```

```
## Object of class SpatialPointsDataFrame
## Coordinates:
##         min      max
## x -10659357 -3509357
## y  -6319778  8090222
## Is projected: TRUE
## proj4string :
## [+proj=moll +lon_0=0 +x_0=0 +y_0=0 +ellps=WGS84 +datum=WGS84
## +units=m +no_defs +towgs84=0,0,0]
## Number of points: 3235
## Data attributes:
##  riqueza_apply
##  Min.   : 0.000
##  1st Qu.: 0.000
##  Median : 0.000
##  Mean   : 2.871
##  3rd Qu.: 4.000
##  Max.   :26.000
```

```
#Obtener un raster
Riqueza_raster <- rasterFromXYZ(Riqueza_points, crs="+proj=moll
+lon_0=0 +x_0=0 +y_0=0 +ellps=WGS84 +datum=WGS84 +units=m +no_defs")
plot(Riqueza_raster)
```




```
#Para graficar en la escala de frío a cálido
tempcol <- colorRampPalette(c("blue", "skyblue", "green",
"lightgreen", "yellow", "orange", "red", "darkred"))
plot(Riqueza_raster, main="Riqueza de especies", col=tempcol(10))
```


Riqueza de especies
6e+06
2e+06
−2e+06
−6e+06
−1.5e+07
−1.0e+07
−5.0e+06
0.0e+00
5.0e+06
25
20
15
10
5
0