Species discover rate in sampling

Ivan Gonzalez

Monday, January 04, 2016

The current file is the perform of 'research priorities index' that try to point out the *species discover rate*. That is the number of records needed to get a new species in that location. This index take the species samples as a parameter. In that sense the index output is heavly influenced by community structure. This analysis is performed over a Colombian states shape.

Index perform

We are gonna to simulate the structure of four comumunities and evaluate the index value. Both will have the same number of species and same number of species as follows: * com1: Complex structure; high dominance * com2: Simple structure; high eveness

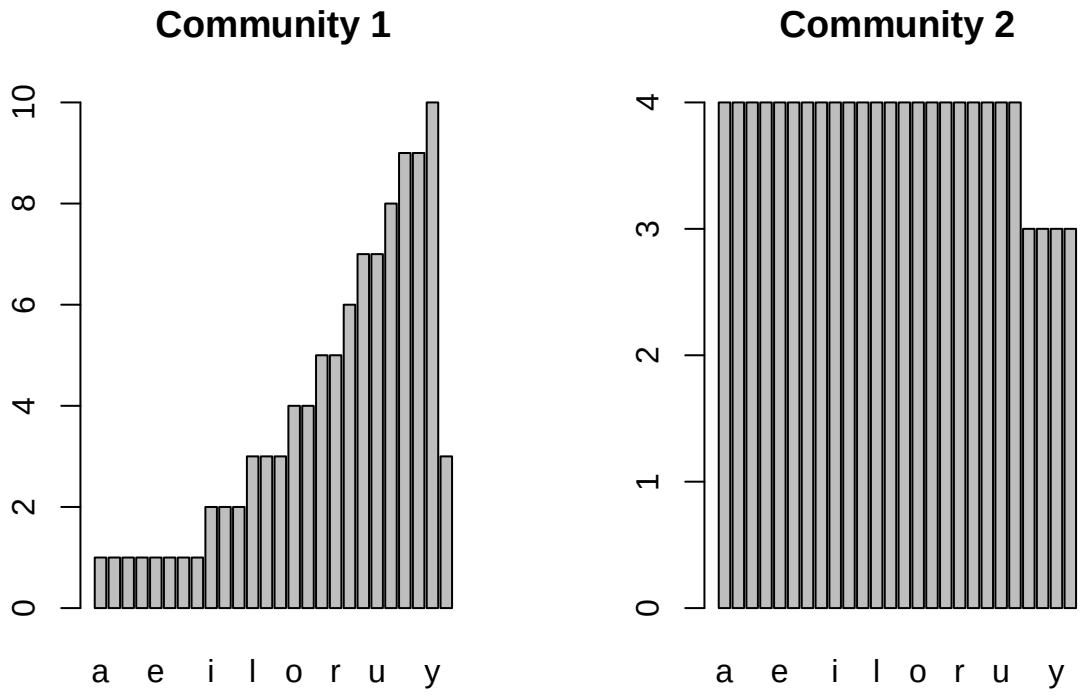
```
nSpecies <- 26
nosim <- 100

# Generate 100 species with 26 species and randomize their positions
com1 <- factor(rep(letters[1:26], times = ceiling((1:26)**2/66))[1:100])
com1 <- com1[sample(x = 1:nosim, size = nosim, replace = FALSE)]

com2 <- factor(rep(letters[1:nSpecies], length.out = 100))
com2 <- com2[sample(x = 1:nosim, size = nosim, replace = FALSE)]

par(mfrow = c(1, 2))

barplot(summary(com1), main = 'Community 1')
barplot(summary(com2), main = 'Community 2')
```



It's possible to say that the first have some dominant species and the second is evener.

```
par(mfrow = c(1, 1))

dup.1 <- !duplicated(com1)
acum.1 <- 1:sum(dup.1)
index.1 <- which(dup.1)

dup.2 <- !duplicated(com2)
acum.2 <- 1:sum(dup.2)
index.2 <- which(dup.2)

index1 <- newSpRate(data = data.frame(species = com1, locality = 'a'), simulations = 20)
```

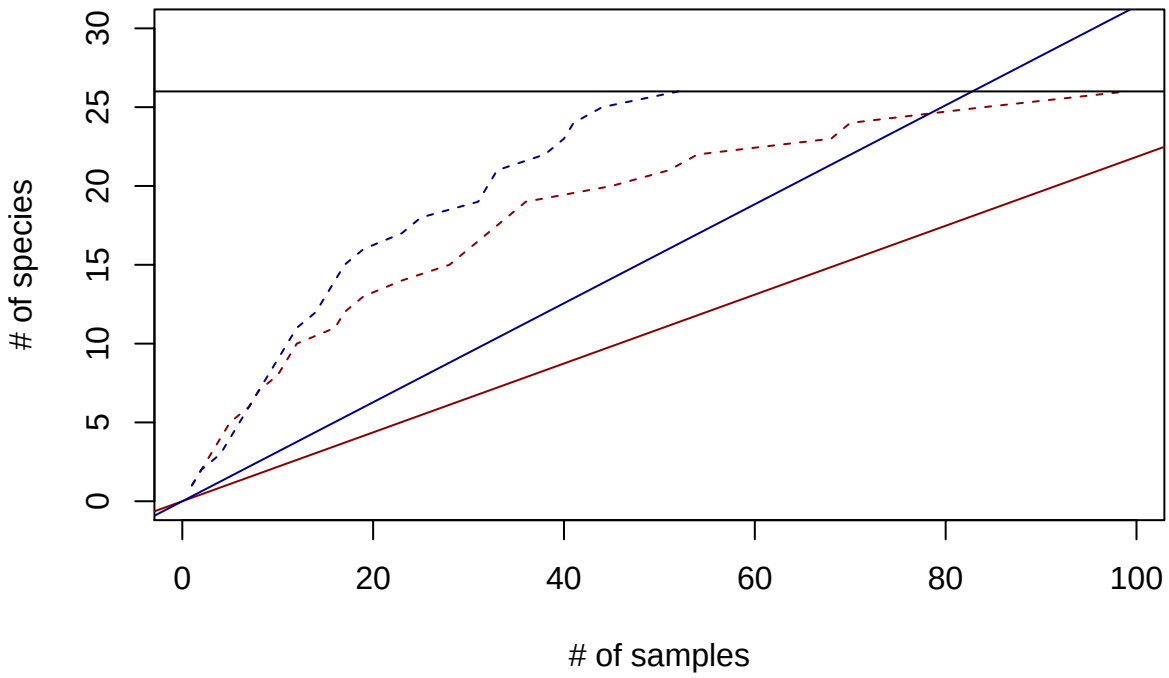
```
## 1 - 1  ||
```

```
index2 <- newSpRate(data = data.frame(species = com2, locality = 'a'), simulations = 20)
```

```
## 1 - 1  ||
```

```
plot(index.1, acum.1, type = 'l', ylim = c(0, 30), lty = 2, col = 'darkred', xlab = '# of samples', ylab
lines(index.2, acum.2, col = 'darkblue', lty = 2)

abline(a = 0, b = index1[1, 2], col = 'darkred')
abline(a = 0, b = index2[1, 2], col = 'darkblue')
abline(h = nSpecies)
```



*Note that both curves reach 26 at the black line

Dashed line for first (red) and second (blue) community represents species acumulation curve. The solid ones are the slope or the required rate for getting one new species in your samples: * Community 1 rate: 0.2184611 * Community 2 rate: 0.3140116 The results allow us to infer that in the evener community is easyer to find a new spaece rather than in the dominant one

Beside both communities have the same number of species and samples the index value differ. This index depend of community structure, species richness but not of sample size

Index usage

The function *newSpRate* require a data.frame as a input with two columns: species and locality. The first one must have a strings with species names and the second one the name of location. This 'locality' column is specially usefull for mahe the analisis over several locations. One unique location value in this column is also valid

```
library(rgdal)
source('function_newSpRate.R')
load('colRecords.RData')
data <- data.frame(species = data$nombre_aceptado, locality = data$sugerencia_depto, stringsAsFactors =
str(data)
```

```
## 'data.frame':    922634 obs. of  2 variables:
##  $ species : chr  "Mimus gilvus" "Mimus gilvus" "Dioscorea alata" "Dioscorea alata" ...
##  $ locality: chr  "San Andrés y Providencia" "San Andrés y Providencia" "San Andrés y Providencia" "S
```

```
# layer <- readOGR('myfolder', 'dinavis_dep')
# data$locality <- iconv(data$locality, from = 'latin1', to ='UTF-8') # fix codification problems
```

The other parameters are: * data (required): two columns data frame as explained above * layer (optional): a vector layer that that will be apendded with results * layerField (optional): the column name of shape table that contains the information of locality * simulations (optional): 10000 by default. The number of repetitions that calculate the index with a randomize * outDir (optional): the folder when the tables will be saved as .csv

Value of index perform by number of simulations

In this code we are gonna to explore how many simulations (perform the analysis with randomized species positions) are required to get an stable index result.

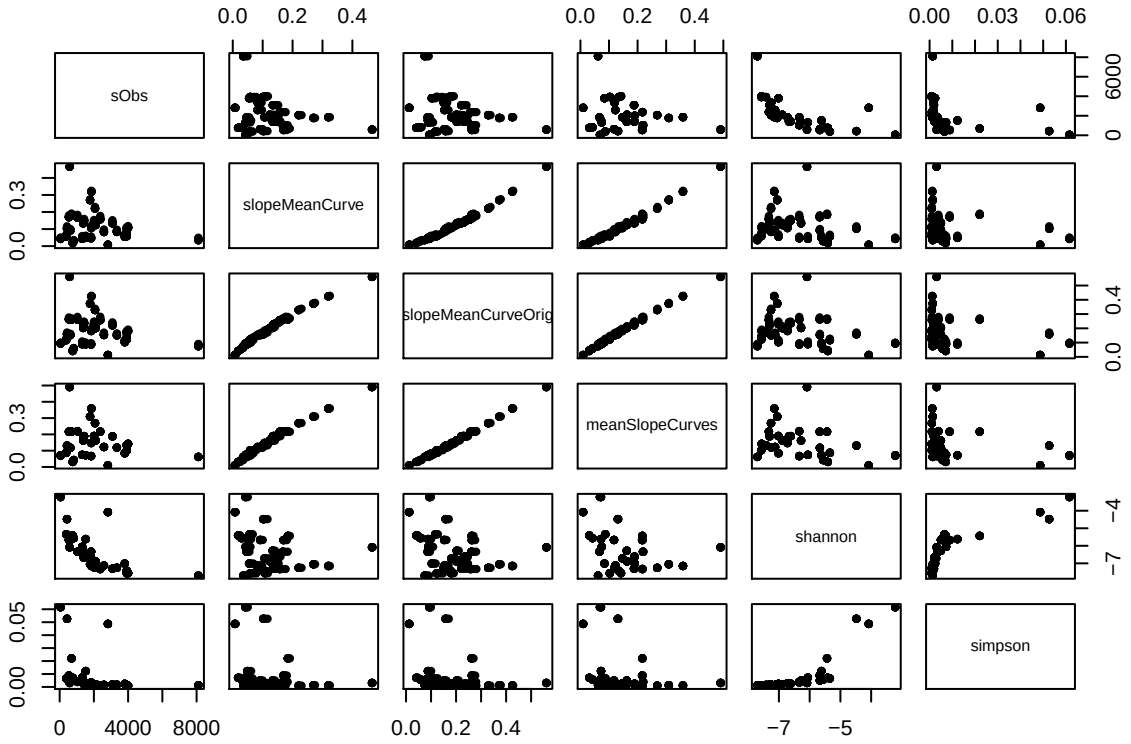
We are not gonna to run this code becouse take some time. Please uncomment the lines and run in your machine.

```
# Set the simulations levels
levelSim <- c(10, 20, 50, 100, 200, 500, 1000, 2000, 5000, 10000)

# spRate <- NULL
# for (s in 1:length(levelSim)) { # }
#   outdir <- paste0('dep', levelSim[s])
#   spRateDepto <- newSpRate(data = data, simulations = levelSim[s], outDir = outdir)
#   spRate <- rbind(spRate, cbind(levelSim = levelSim[s], spRateDepto))
#   cat('\n \n')
# }
# write.csv(spRate, 'spRateBySimulations.csv', row.names = FALSE)

spRate <- read.csv('spRateBySimulations.csv', as.is = TRUE)

pairs(spRate[, c('sObs', 'slopeMeanCurve', 'slopeMeanCurveOrig', 'meanSlopeCurves', 'shannon', 'simpson
```




```
cor(spRate[, c('sObs', 'slopeMeanCurve', 'slopeMeanCurveOrig', 'meanSlopeCurves', 'shannon', 'simpson')]
```

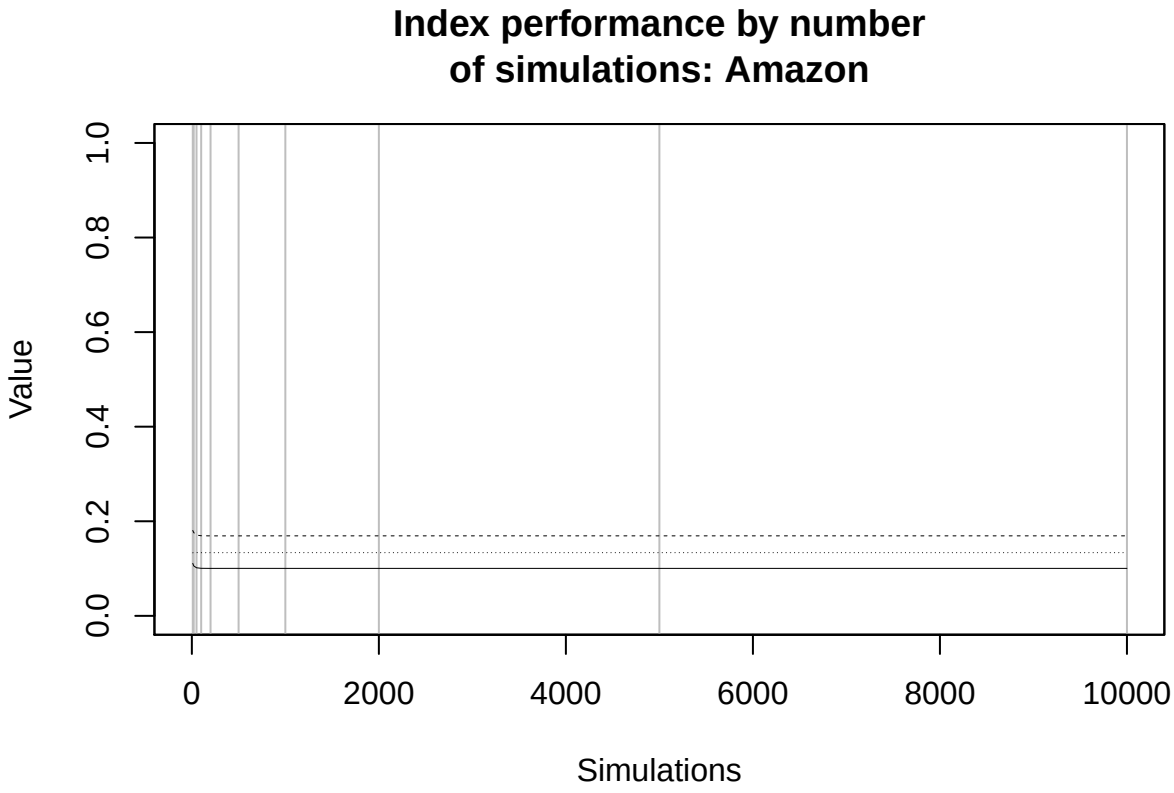
```
##                          sObs slopeMeanCurve slopeMeanCurveOrig
## sObs                1.0000000     -0.2266137         -0.2120954
## slopeMeanCurve     -0.2266137      1.0000000          0.9899258
## slopeMeanCurveOrig -0.2120954      0.9899258          1.0000000
## meanSlopeCurves    -0.2094572      0.9928016          0.9981286
## shannon            -0.6350057     -0.2463206         -0.3127892
## simpson            -0.3160480     -0.2571287         -0.3048354
##                    meanSlopeCurves    shannon    simpson
## sObs                     -0.2094572 -0.6350057 -0.3160480
## slopeMeanCurve            0.9928016 -0.2463206 -0.2571287
## slopeMeanCurveOrig        0.9981286 -0.3127892 -0.3048354
## meanSlopeCurves           1.0000000 -0.3006360 -0.2930405
## shannon                  -0.3006360  1.0000000  0.8487807
## simpson                  -0.2930405  0.8487807  1.0000000
```

The 3 indexes calculated ('slopeMeanCurve', 'slopeMeanCurveOrig' and 'meanSlopeCurves') are highly correlated so we are gonna to use the first, 'slopeMeanCurve', for the ongoing analysis.

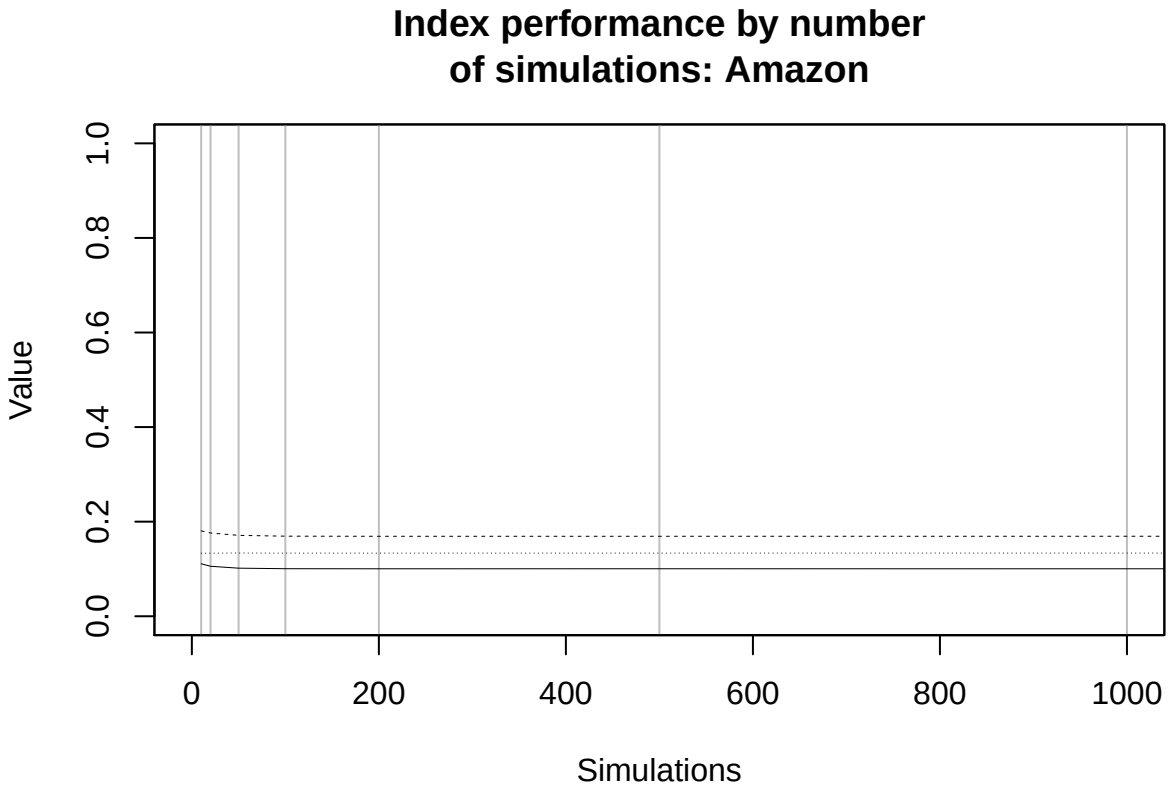
The follow first plot show the values of index trougth simulation levels for the Amazon state using both 1000 and 10000 simulations.

```
levels <- unique(spRate$level)
dept <- subset(spRate, level == levels[32])
```

```
plot(NULL, ylim = c(0, 1), xlim = c(0, 10000), type = 'n', ylab = 'Value', xlab = 'Simulations',
     main = 'Index performance by number\nof simulations: Amazon')
abline(v = dept$levelSim, col = 'grey'); box()
lines(dept$levelSim, dept$slopeMeanCurve, ylim = c(0, 1), lwd = 0.5)
lines(dept$levelSim, dept$slopeMeanCurveOrig, lty = 2, lwd = 0.5)
lines(dept$levelSim, dept$meanSlopeCurves, lty = 3, lwd = 0.5)
```



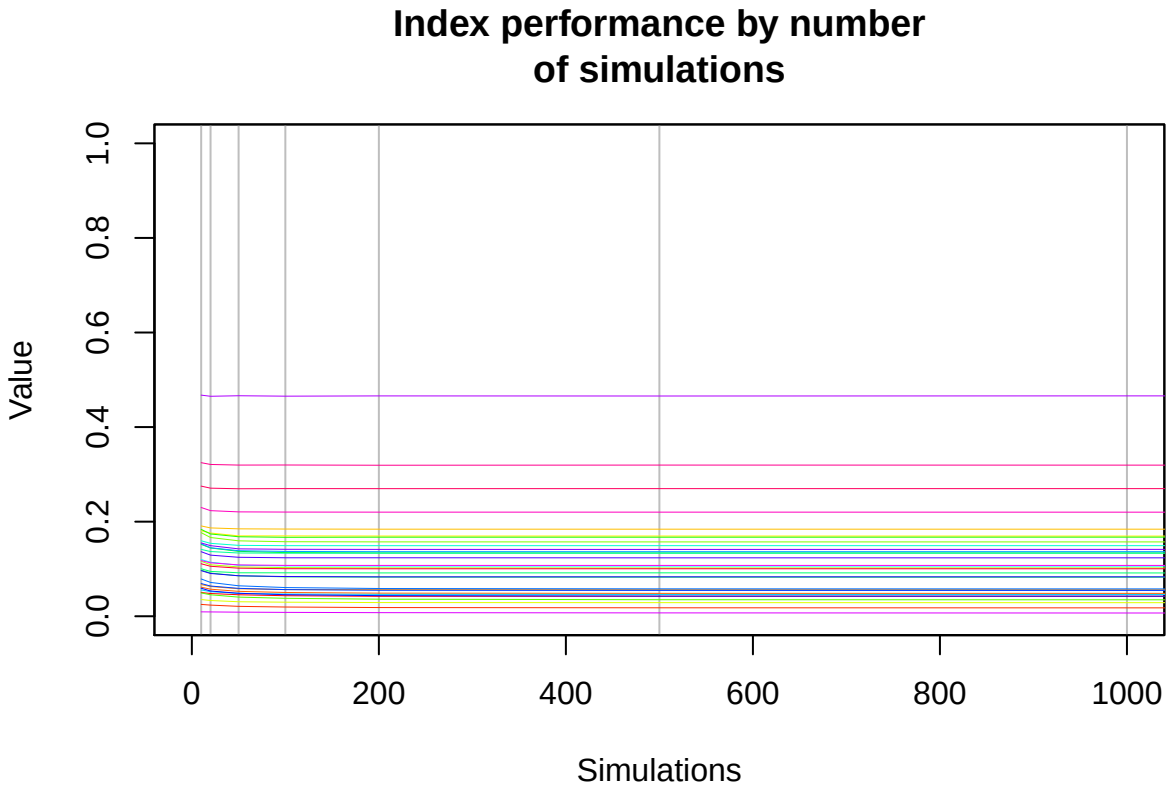

```
plot(NULL, ylim = c(0, 1), xlim = c(0, 1000), type = 'n', ylab = 'Value', xlab = 'Simulations',
     main = 'Index performance by number\nof simulations: Amazon')
abline(v = dept$levelSim, col = 'grey'); box()
lines(dept$levelSim, dept$slopeMeanCurve, ylim = c(0, 1), lwd = 0.5)
lines(dept$levelSim, dept$slopeMeanCurveOrig, lty = 2, lwd = 0.5)
lines(dept$levelSim, dept$meanSlopeCurves, lty = 3, lwd = 0.5)
```



Now the plot for all states

```
plot(NULL, ylim = c(0, 1), xlim = c(0, 1000), type = 'n', ylab = 'Value', xlab = 'Simulations',
     main = 'Index performance by number\nof simulations')
abline(v = dept$levelSim, col = 'grey'); box()

color.l <- rainbow(length(levels))
for (l in 1:length(levels)){
  dept <- subset(spRate, level == levels[l])
  lines(dept$levelSim, dept$slopeMeanCurve, ylim = c(0, 1), lwd = 0.5, col = color.l[l])
}
```



We recomend to use this function with the parameter simulations = 200 due that a higer number of simulations willl no apport new information and will demand computational resources.

Index aplication over Colombian States

We are gonna to use the values of the index performed under the parameter simulations = 200. This code will read all separated values for each state and make a

```
outdir <- 'dep200'
csvPath <- outdir
coefFiles <- list.files(path = csvPath, pattern = 'coef')
cumFiles <- list.files(path = csvPath, pattern = 'Curve')
deptNames <- gsub('coefTable_|.csv', '', coefFiles)

depto <- readOGR('.', 'dep', encoding = 'UTF-8')
```

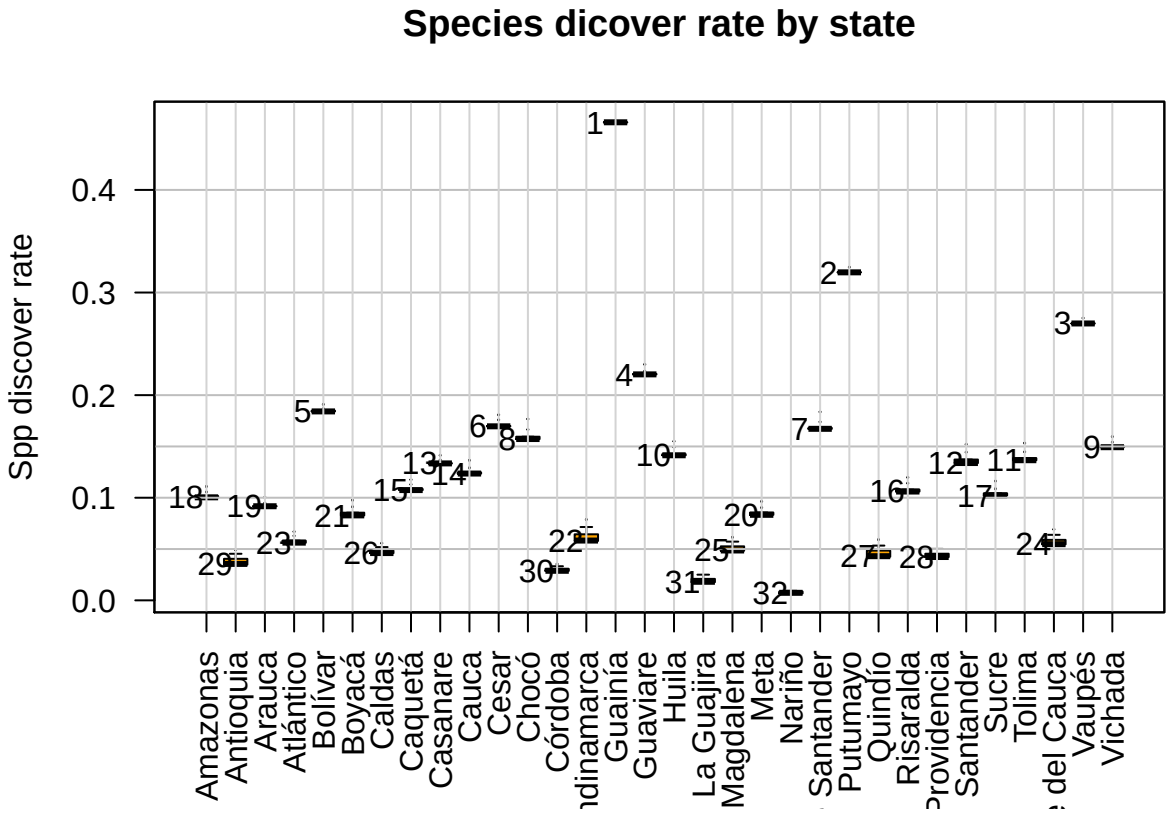
```
## OGR data source with driver: ESRI Shapefile
## Source: ".", layer: "dep"
## with 33 features
## It has 4 fields
```

```
depto@data$coef <- 0

richDept <- NULL
for(d in 1:length(deptNames)){
```

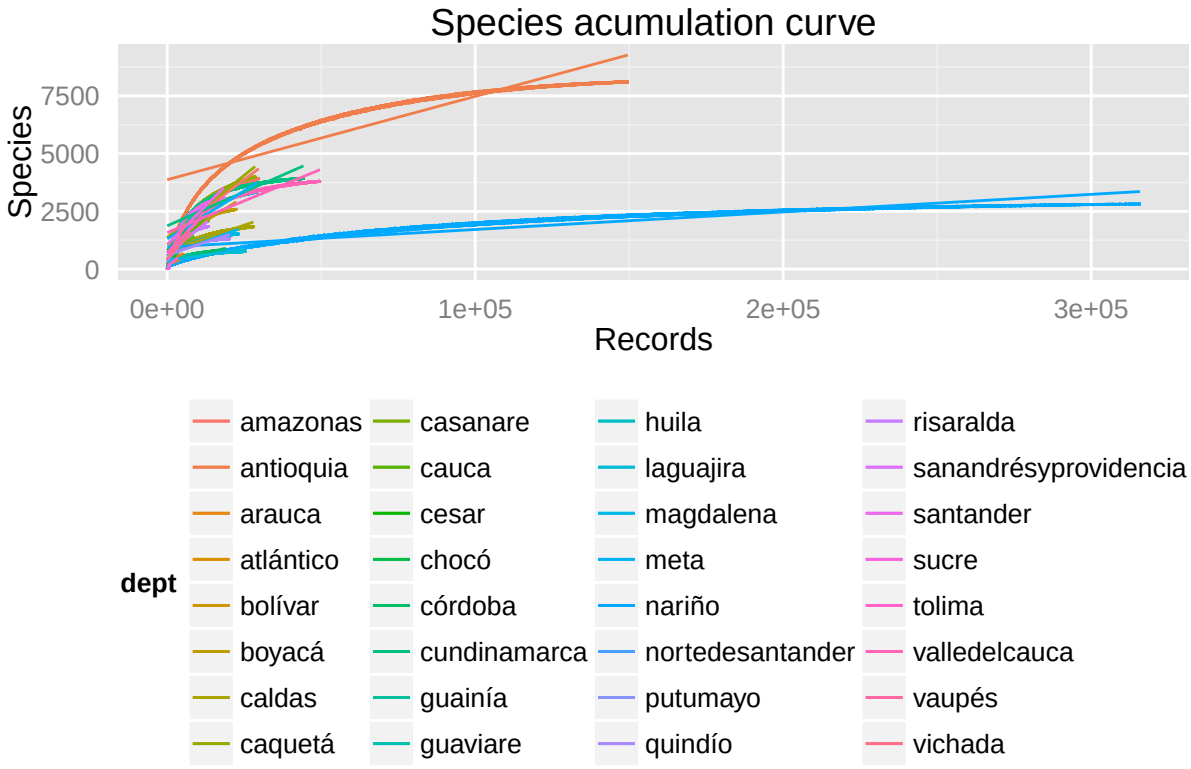
```
  csv.d <- read.csv(paste0(csvPath, '/', coefFiles[d]))
  name.d <- tolower(gsub('coefTable_|.csv| ', '', coefFiles[d]))
  depto$coef[agrep(name.d, tolower(gsub(' ', '', depto$NOM_DPTO)))] <- mean(csv.d$slope)
  curved.d <- read.csv(paste0(csvPath, '/', cumFiles[d]))
  richDept <- rbind(richDept, data.frame(dept = name.d, n = 1:nrow(curved.d), s = curved.d$x))
}

spRate$levelFac <- as.factor(spRate$level)
boxp <- boxplot(formula = slopeMeanCurve ~ levelFac, data = spRate, las = 2, pch = 20, col = 'orange', c
        ylab = "Spp discover rate", main = 'Species dicover rate by state')
abline(h = c(0.05, 0.1, 0.15, 0.2, 0.3, 0.4), col = 'grey')
pos <- order(boxp$stats[3, ])
abline(v = 1:length(pos), col = 'lightgrey')
text(x = pos- 0.7, y = boxp$stats[3, pos], labels = length(pos):1); box()
```



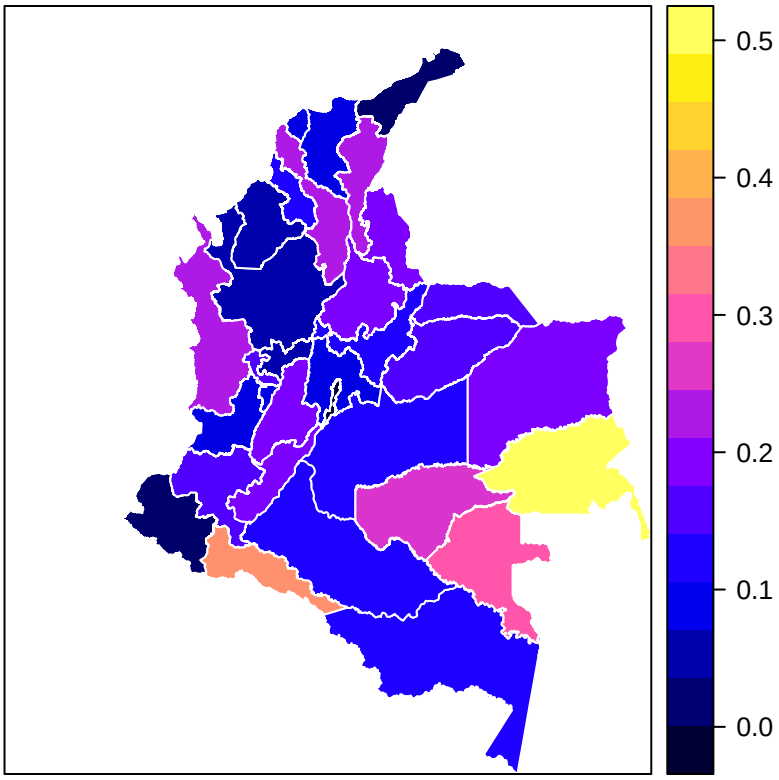
Lets look the acumulation curve and slope line for each state

```
library(ggplot2)
g <- ggplot(data = richDept, aes(x = n, y = s, color = dept))
g + geom_line(data = richDept, aes(x = n, y = s, color = dept), alpha = 1) +
  labs(x = "Records", y = "Species", title = 'Species acumulation curve') +
  theme(axis.ticks.y = element_blank(), axis.ticks.x = element_blank()) +
  geom_smooth(data = richDept, aes(x = n, y = s, color = dept), method = lm, se = FALSE, fullrange = FA
  scale_fill_continuous(guide = guide_legend()) +
theme(legend.position="bottom") +  guides(col = guide_legend(nrow = 8))
```



Save the original shapefile and plot it

```
writeOGR(depto, dsn = "researchPriorities", layer = 'deptos', driver = "ESRI Shapefile", overwrite_laye
spplot(depto, zcol = 'coef', col = 'white')
```


0.5
0.4
0.3
0.2
0.1
0.0