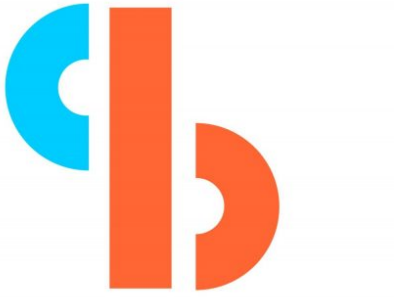
Bioestadística en R - parte práctica clase 1, Módulo 1 : Análisis de la diversidad

Dra. Stephanie Hereira Pacheco

CTCB, UATx

06-03-2023



POSGRADO EN CIENCIAS
BIOLÓGICAS

estación
científica
la malinche.

Análisis de diversidad

Contenido

Parte 1

- `hillR`
- `betapart`
- `vegan`

Parte 2

- `iNext`

Parte 1

hillR

El paquete `hillR` contiene funciones para calcular la diversidad taxonómica, funcional y filogenética en el marco de los números de Hill. Los métodos están basados en la referencia de **Chao, Chiu, & Jost, (2014)** que ya hemos tratado en la parte teórica.

Para instalar este paquete usamos el comando:

```
install.packages("hillR")
# o instala la versión en desarrollo del github
devtools::install_github("daijiang/hillR")
```

También instalaremos el paquete `FD` de dónde usaremos la data ejemplo:

```
install.packages("FD")
```

hillR

Usaremos la data **dummy** de ejemplo que viene en el paquete **FD** para utilizar las funciones de `hillR`.

```
set.seed(123)
library(FD)
dummy_data <- dummy
comunidades<-  dummy_data$abun
funciones <- dummy_data$trait
arbol <- ape::rtree(n = ncol(comunidades), tip.label = paste0("sp", 1:ncol(comunidades)))
```

En este caso, Lo primero es usar la función **'set.seed'** o sembrar semilla, para indicarle a R que nos guarde los cálculos en esta semilla y cada vez que lo corramos den el mismo valor (eso es cuando son muestreos aleatorios como es el caso de la función *rtree* del paquete 'ape'.

hillR

Exploremos los tipos de datos que tenemos:

```
class(comunidades)
```

```
## [1] "matrix" "array"
```

```
class(funciones)
```

```
## [1] "data.frame"
```

```
class(arbol)
```

```
## [1] "phylo"
```

hillR

Definimos nuestras datas a usar:

- comunidades: tabla con counts o recuentos de especies (columnas) por sitio o muestra (filas).
- funciones : tabla con funciones y resultados (numéricos o categóricos) por especie descrita en nuestra tabla de comunidades. Especies como filas y funciones como columnas.
- arbol: objeto tipo "phylo" tipo lista con vértices y nodos describiendo relaciones filogenéticas.

```
head(comunidades)
```

```
##       sp1 sp2 sp3 sp4 sp5 sp6 sp7 sp8
## com1    1   1   0   0   4   2   0   0
## com2    0   0   0   2   1   0   0   5
## com3    2   0   0   0   0   1   0   3
## com4    1   0   7   0   0   0   0   0
## com5    0   0   2   3   3   0   0   0
## com6    0   3   0   0   5   6   1   6
```

```
head(funciones)
```

```
##      num1 num2 fac1 fac2 ord1 ord2 bin1 bin2
## sp1   9.0  4.5    A    X    3    2    0    1
## sp2   8.1  6.0    A    Z <NA>    1    0    1
## sp3    NA  2.3    C    Y    5    3    1    1
## sp4   3.2  5.4    B    Z    1    7    0    0
## sp5   5.8  1.2    C    X    2    6   NA    0
## sp6   3.4  8.5    C    Y    2    1    1    1
```

```
head(arbol)
```

```
## $edge
##       [,1] [,2]
##  [1,]    9   10
##  [2,]   10   11
##  [3,]   11   12
##  [4,]   12    1
##  [5,]   12    2
##  [6,]   11    3
##  [7,]   10   13
##  [8,]   13   14
##  [9,]   14   15
## [10,]   15    4
## [11,]   15    5
## [12,]   14    6
## [13,]   13    7
## [14,]    9    8
##
## $tip.label
## [1] "sp2" "sp6" "sp3" "sp5" "sp4" "sp8" "sp7" "sp
##
## $Nnode
## [1] 7
##
## $edge.length
```

Cálculo de la diversidad taxonómica, funcional y filogenética de cada sitio o muestra (alfa diversidad).

Primero cargamos la librería `hillR` previamente instalada:

```
library(hillR)
```

Luego calculamos la diversidad taxonómica:

```
hill_taxa(comunidades, q = 0)
```

De igual forma la diversidad funcional:

```
hill_func(comunidades, funciones, q = 0)
```

Y por último la diversidad filogenética:

```
hill_phylo(comunidades, arbol, q = 0)
```

Los resultados que nos da cada función son:

1. **hill_taxa()** nos da un vector con el va valor de diversidad alfa para cada sitio o muestra, q = 0 (por defecto) riqueza de especies , q = 1 para obtener shannon (especies frecuentes) y q = 2 el inverso de simpson (especies dominantes).
2. **hill_func()** nos dará una matrix con la información de:

- Q : Q de Rao
- D_q : el numero efectivo de epecies distintas igualmente abundantes y funcionales
- MD_q : diversidad funcional media por especie, la suma efectiva de las distancias por pares entre una especie fija y todas las demás especies
- FD_q : diversidad funcional total, la distancia funcional total efectiva entre especies del conjunto

1. **hill_phylo** nos dará un vector de diversidad filogenética basada en el número de Hill ('PD(T)', longitud total efectiva de la rama) para todos los sitios.

Cálculo de la diversidad taxonómica, funcional y filogenética de a través de diferentes sitios o muestras.

> Este script calcula la diversidad **gamma**, **alfa**, y **beta** a traves de todas las comunidades o muestras asi como su similitud.

Si comm>2 la diversidad gamma es la diversidad juntada (pooled) del ensamble, la alfa es el promedio de la diversidad a través de todos los sitios y la beta es a través de todas las comunidades.

También nos da la medida de homogeneidad de MacArthur, la similtud local (traslape de especies / overlap similar al de Sorensen) y la similitud regional (traslape de especies overlap similar al de Jaccard).

- Para las comunidades:

```
hill_taxa_parti(comunidades, q = 0)
```

```
##   q TD_gamma TD_alpha  TD_beta M_homog local_similarity region_similarity
## 1 0        8      3.6 2.222222    0.45        0.8641975         0.3888889
```

- Para las funciones:

```
hill_func_parti(comunidades, funciones, q = 0)
```

```
##   q raoQ_gamma FD_gamma FD_alpha FD_beta local_similarity
## 1 0  0.4529152 29.66099 14.15941 2.09479        0.9889415
##   region_similarity
## 1         0.4720957
```

- Para las filogenias:

```
hill_phylo_parti(comunidades, arbol, q = 0)
```

```
##   q PD_gamma PD_alpha  PD_beta local_similarity region_similarity
## 1 0 8.292885 5.119871 1.619745        0.9311395          0.574868
```

Cálculo de la diversidad pareada taxonómica, funcional y filogenética

> Calcula la diversidad pareada gamma, alfa y beta para las comunidades así como la similitud

La diversidad pareada hace el cáclulo entre pares de sitios o muestreasw generando ya sea un data.frame o una matriz de distancia que puede ser usada para futuros análisis multivareado. En este ejemplo y por defecto obtendremos el data.frame para la diversidad taxonómica. Y al final haremos un ejemplo de cómo se vería una matriz.

- Para las comunidades:

```
head(hill_taxa_parti_pairwise(comunidades, q = 0, .progress = FALSE), n = 3)
```

```
## # A tibble: 3 × 8
##       q site1 site2 TD_gamma TD_alpha TD_beta local_similarity region_si…¹
##   <dbl> <chr> <chr>    <dbl>    <dbl>   <dbl>            <dbl>       <dbl>
## 1     0 com1  com2         6      3.5    1.71            0.286       0.167
## 2     0 com1  com3         5      3.5    1.43            0.571       0.4
## 3     0 com2  com3         5      3      1.67            0.333       0.2
## # … with abbreviated variable name ¹region_similarity
```

- Para las funciones:

```
head(hill_func_parti_pairwise(comunidades, funciones, q = 0, show_warning = FALSE, .progress =FALS
```

```
## # A tibble: 3 × 8
##       q site1 site2 FD_gamma FD_alpha FD_beta local_similarity region_si…¹
##   <dbl> <chr> <chr>    <dbl>    <dbl>   <dbl>            <dbl>       <dbl>
## 1     0 com1  com2      15.9     10.3    1.54            0.821       0.534
## 2     0 com1  com3      10.7      7.96   1.35            0.883       0.655
## 3     0 com2  com3      11.1      7.03   1.58            0.807       0.511
## # … with abbreviated variable name ¹region_similarity
```

- Para las filogenias

```
head(hill_phylo_parti_pairwise(comunidades, arbol, q = 0, show_warning = FALSE, .progress = FALSE)
```

```
## # A tibble: 6 × 8
##       q site1 site2 PD_gamma PD_alpha PD_beta local_similarity region_si…¹
##   <dbl> <chr> <chr>    <dbl>    <dbl>   <dbl>            <dbl>       <dbl>
## 1     0 com1  com2      6.79     5.06    1.34            0.657       0.489
## 2     0 com1  com3      6.14     4.95    1.24            0.759       0.611
## 3     0 com2  com3      6.75     4.57    1.48            0.523       0.355
## 4     0 com1  com4      6.38     3.99    1.60            0.400       0.250
## 5     0 com2  com4      7.13     3.62    1.97            0.0284      0.0144
## 6     0 com3  com4      5.42     3.51    1.54            0.455       0.295
## # … with abbreviated variable name ¹region_similarity
```

Veamos el ejemplo de cuando queremos obtener las matrices de distancia al orden 0:

```
beta_q0_mat<-hill_taxa_parti_pairwise(
  comunidades, q = 0, .progress = FALSE,
  output = "matrix", pairs = "full")
```

Esta vez en vez de un data.frame nos dará una lista que contendrá 6 elementos, definidos así:

q = el orden al cual fue realizado el cálculo

TD_gamma= gamma pareada

TD_alpha=alfa pareada

Td_beta: dependiendo del orden *q* (*q*=0 Jaccard/Sorensen, *q*=1 Horn y *q*=2 Morisita-Horn).

local_similarity : traslape de especies similar a Sorensen (q=0)

region_similarity: traslape de especies similar a Jaccard (q=0)

Ejemplo aplicado

Tomemos la data 'spiders' del paquete iNEXT que veremos más adelante. Para más información de estos puedes consultar las referencia 🔗📎

```
library(iNEXT)
data("spider")
```

¿Cuál de los dos tratamientos da valores más altos de riqueza?

betapart

- El paquete `betapart` fue desarrollado por el mismo autor que introduce los conceptos de partición de la diversidad beta en los componentes de recambio (turnover) y anidamiento (nestedness).

- Para instalarlo ponemos la siguiente línea de código:

```
install.packages("betapart")
```

Primero llamamos al paquete previamente instalado. Luego definiremos el conjunto de datos de ejemplo, y diremos que las tres primeras columnas pertenecen a un bosque no perturbado y las tres siguientes a un bosque no perturbado.

```r
library(betapart)
comm<- data.frame(comm =1:6,
                          sp1=c(2,2,3,0,0,1),
                          sp2=c(2,2,0,1,1,2),
                          sp3=c(1,0,1,2,3,2),
                          sp4=c(1,0,1,0,2,0),
                          sp5=c(1,2,0,0,0,1),
                          sp6=c(2,2,1,0,0,0),
                          sp7=c(0,0,0,1,0,1),
                          sp8=c(1,0,1,0,1,0), row.names = 1)


groups <- factor(c(rep(1,3), rep(2,3)),
                 labels = c("noperturbado","perturbado"))
head(comm);groups
```

```
##   sp1 sp2 sp3 sp4 sp5 sp6 sp7 sp8
## 1   2   2   1   1   1   2   0   1
## 2   2   2   0   0   2   2   0   0
## 3   3   0   1   1   0   1   0   1
## 4   0   1   2   0   0   0   1   0
## 5   0   1   3   2   0   0   0   1
## 6   1   2   2   0   1   0   1   0
```

```
## [1] noperturbado noperturbado noperturbado perturbado   perturbado
## [6] perturbado
```

betapart

Los índices Jaccard y Sorensen en los que está basado este paquete no consideran la abundancia de especies (*q*=0), así que convertiremos nuestra data a tipo incidencia (presencia/ausencia):

```
presabs<-ifelse(comm>0,1,0)
```

Ahora, calcularemos el índice de Sorensen y sus particiones de recambio y anidamiento. En su lugar, podemos calcular el índice de Jaccard usando el argumento index.family="jaccard".

```
dist_comp<-beta.pair(presabs, index.family="sorensen")
```

Vemos que el resultado es una lista de tres matrices:

```
head(dist_comp)
```

```
## $beta.sim
##           1         2         3         4         5
## 2 0.0000000
## 3 0.0000000 0.5000000
## 4 0.3333333 0.6666667 0.6666667
## 5 0.0000000 0.7500000 0.2500000 0.3333333
## 6 0.2000000 0.2500000 0.6000000 0.0000000 0.5000000
##
## $beta.sne
##            1          2          3          4          5
## 2 0.27272727
## 3 0.16666667 0.05555556
## 4 0.26666667 0.04761905 0.08333333
## 5 0.27272727 0.00000000 0.08333333 0.09523810
## 6 0.13333333 0.08333333 0.00000000 0.25000000 0.05555556
##
## $beta.sor
##           1         2         3         4         5
## 2 0.2727273
## 3 0.1666667 0.5555556
## 4 0.6000000 0.7142857 0.7500000
```

betapart

Para obtener o extraer la partición del índice de recambio de Sorensen por pares entre comunidades, escriba: dist_comp\$beta.sim

Para obtener la partición de anidamiento: dist_comp\$beta.sne

Para obtener toda la diversidad beta: dist_comp\$beta.sor

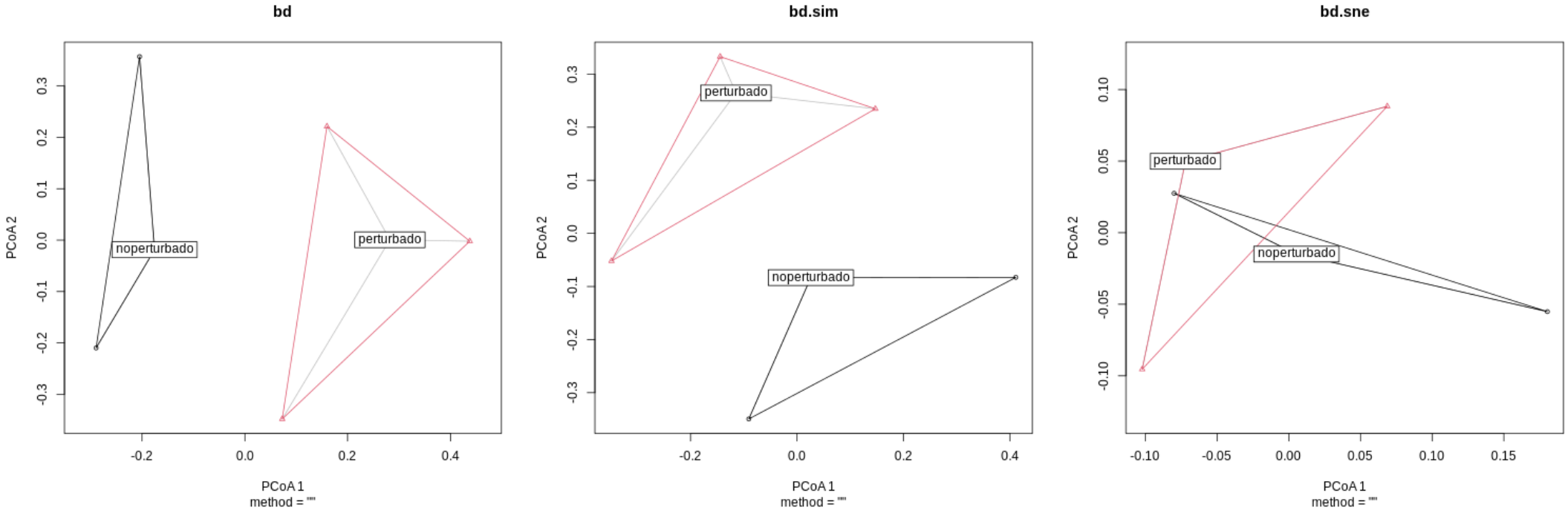
betapart

Ya obtenidas las diversas matrices podemos aplicar cualquier tipo de análisis multivariado deseado.

En el presente ejemplo, si queremos comparar las diversidades beta de las comunidades agregadas por los tratamientos de "no perturbado" y "perturbado", podemos utilizar el análisis "betadisper" del paquete 'vegan':

```
library(vegan)
bd<-betadisper(dist_comp$beta.sor,groups)
bd.sim<-betadisper(dist_comp$beta.sim,groups)
bd.sne<-betadisper(dist_comp$beta.sne,groups)
```

```
plot(bd);plot(bd.sim);plot(bd.sne)
```



El gráfico de dispersión beta indica que hay una diferencia en la composición de especies de los fragmentos de bosque perturbado y no perturbado explicado de mejor manera por el recambio.

betapart

Si queremos obtener estos mismos resultados pero para todos los sitios, entonces usamos la función **beta.multi()**:

```
dist.multi<-beta.multi(presabs,index.family ="sorensen" )
head(dist.multi)
```

```
## $beta.SIM
## [1] 0.4871795
##
## $beta.SNE
## [1] 0.1206637
##
## $beta.SOR
## [1] 0.6078431
```

Ejemplo aplicado

Con la misma data trabajada anteriormente vamos a verificar si dan resultados similares utilizando el índice de Jaccard.

vegan

Esta paquetería es una excelente herramienta para analizar datos ecológicos. En el caso de alfa diversidad se pueden calcular los índices tradicionales (que no están en el marco de los números de Hill).

Aquí realizaremos un ejemplo con una data que trae de ejemplo que se llama BCI (Barro Colorado Island Tree Counts).

Primero llamamos la data:

```
data(BCI, BCI.env)
```

Podemos calcular los índices de diversidad alfa, como Shannon, Simpson, el inverso de Simpson, Fisher, Especies observadas, beta y gamma:

```
shannon <- diversity(BCI)
simpson <- diversity(BCI, "simpson")
inverso_simp <- diversity(BCI, "inv")
fisher <- fisher.alpha(BCI)
especies_num <- specnumber(BCI)
alfa <- with(BCI.env, tapply(specnumber(BCI), Habitat, mean))
gamma <- with(BCI.env, specnumber(BCI, Habitat))
gamma/alfa - 1
```

```
##   OldHigh    OldLow  OldSlope     Swamp     Young
## 0.8425656 1.2883487 0.9981802 0.3617021 0.3000000
```

```
alfa <- with(BCI.env, tapply(diversity(BCI), Habitat, mean)) # promedio
gamma <- with(BCI.env, diversity(BCI, groups=Habitat)) # junta
gamma-alfa
```

```
##   OldHigh    OldLow  OldSlope     Swamp     Young
## 0.2345878 0.4085595 0.3249760 0.1605548 0.1408068
```

Para la beta divesidad podemos calcular nuestras matrices de distancia con la función 'vegdist':

```
library(tidyverse)
distancias<- vegdist(BCI, method = "jaccard")
distancias %>% as.matrix() %>% as.data.frame() %>% dplyr::select(1:4) %>% dplyr::slice(1:4)
```

```
##           1         2         3         4
## 1 0.0000000 0.4260250 0.5186992 0.5382263
## 2 0.4260250 0.0000000 0.4463668 0.4790323
## 3 0.5186992 0.4463668 0.0000000 0.4898911
## 4 0.5382263 0.4790323 0.4898911 0.0000000
```

Parte 2

iNEXT: iNterpolation and EXTrapolation

- Es un paquete disponible en R para rarefacción y extrapolación de la diversidad de especies en el marco de los números de Hill.

- iNEXT se centra en tres medidas de los números de Hill de orden q: riqueza de especies (q=0), diversidad de Shannon (q=1, la exponencial de la entropía de Shannon) y diversidad de Simpson (q=2, la inversa de la concentración de Simpson).

iNEXT: iNterpolation and EXTrapolation

Para cada medida iNEXT utiliza la muestra observada para calcular las estimaciones de diversidad para muestras rarificadas y extrapoladas y los intervalos de confianza del 95 %, además de representar gráficamente los dos tipos siguientes de las curvas de rarefacción y extrapolación (R/E):

> **Curvas de muestreo R/E basadas en el tamaño de la muestra:** iNEXT calcula estimaciones de diversidad para muestras rarificadas y extrapoladas hasta el doble del tamaño de la muestra de referencia o un tamaño especificado por el usuario. Este tipo de curva de muestreo traza las estimaciones de diversidad con respecto al tamaño de la muestra.

Curvas de muestreo R/E basadas en la cobertura: iNEXT calcula estimaciones de diversidad para muestras enrarecidas y extrapoladas con integridad de la muestra (medida por la cobertura de la muestra) hasta el valor de cobertura del doble del tamaño de la muestra de referencia o una cobertura especificada por el usuario. Este tipo de curva de muestreo traza las estimaciones de diversidad con respecto a la cobertura de la muestra.

Además de los dos tipos anteriores de curvas de muestreo, iNEXT también traza una **curva de completitud de la muestra**, que describe cómo varía la estimación de la cobertura de la muestra en función del tamaño de la muestra. La curva de integridad de la muestra se puede considerar como un puente que conecta los dos tipos de curvas mencionados anteriormente.

iNEXT: iNterpolation and EXTrapolation

Para instalar este paquete:

```
## instalando iNEXT del CRAN
install.packages("iNEXT")
```

```
## instalando la versión de desarrollo
install.packages('devtools')
library(devtools)
install_github('AnneChao/iNEXT')
```

```
## cargando el paquete
library(iNEXT)
library(ggplot2)
```

iNEXT: iNterpolation and EXTrapolation

La función principal es:

```
iNEXT(x, q=0, datatype="abundance", se=TRUE, conf=0.95, nboot=50)
```

Donde:

- x : es la data,
- datatype: puede ser "abundance" ó "incidence_raw", ó "incidence_freq",
- se: TRUE o FALSE si se quiere hacer un muestreo tipo 'bootsrap',
- conf: el intervalo de confianza,
- nboot: número de replicaciones de 'bootsrap'

iNEXT(x, q=0, datatype="abundance", se=TRUE, conf=0.95, nboot=50)

Correremos el ejemplo con la data del paquete, vista anteriormente **spider**:

```
data("spider")
str(spider)
```

```
## List of 2
##  $ Girdled: num [1:26] 46 22 17 15 15 9 8 6 6 4 ...
##  $ Logged : num [1:37] 88 22 16 15 13 10 8 8 7 7 ...
```

```
out <- iNEXT(spider, q=c(0, 1, 2), datatype="abundance")
```

```
out
```

Si vemos el output de iNEXT nos da una lista con tres elementos:

> - **\$DataInfo** que nos resume la información de la data
> - **\$iNextEst** los estimados para las muestras rarificadas y extrapoladas (datos para las curvas)
> - **\$AsyEst** muestra la diversidad estimada

- Podemos obtener ahora nuestras curvas por sitio o en este caso por tratamiento a los 3 órdenes:

```
ggiNEXT(out, type=1, facet.var="site")
```

- También podemos obtener ahora nuestras curvas de cobertura:

```
ggiNEXT(out, type=2)
```

- Por último una gráfica de diversidad de especies vs cobertura para los tratamientos:

```
ggiNEXT(out, type=3, facet.var="site")
```

- Y a los diferentes órdenes:

```
ggiNEXT(out, type=3, facet.var="order")
```

- Ahora si se quiere sólo los índices basandonos ya sea en abundancia o incidencia y en tamaño o cobertura, usamos la función **estimateD**, así:

```
estimateD(spider$Logged, datatype="abundance", base="size",  conf=0.95)
```

```
##     m   method order    SC     qD qD.LCL qD.UCL
## 1 252 observed     0 0.945 37.000 31.322 42.678
## 2 252 observed     1 0.945 14.421 11.757 17.085
## 3 252 observed     2 0.945  6.761  5.041  8.482
```

```
estimateD(spider$Girdled, datatype="abundance", base="size",  conf=0.95)
```

```
##     m   method order    SC    qD qD.LCL qD.UCL
## 1 168 observed     0 0.929 26.00 20.338 31.662
## 2 168 observed     1 0.929 12.06  9.927 14.192
## 3 168 observed     2 0.929  7.84  6.110  9.570
```

Ejemplo aplicado

Apliquemos lo visto usando como ejemplo pequeño con datos de incidencia (presencia/ausencia):

```
data(ciliates)
#str(ciliates)
```

Otras funciones

Este paquete también cuenta con otras funciones de interés, tales como:

- ChaoEntropy() : Estimación de la entropía/diversidad de Shannon
- ChaoRichness(): Estimación de la riqueza de especies
- ChaoShannon(): Estimación de la entropía/diversidad de Shannon
- ChaoSimpson(): Estimación del índice de Gini-Simpson o diversidad de Simpson
- ChaoSpecies(): Estimación de la riqueza de especies
- EstSimpson: Estimación del índice de Gini-Simpson o diversidad de Simpson

Un ejemplo

```
ChaoRichness(spider)
```

```
##          Observed Estimator Est_s.e. 95% Lower 95% Upper
## Girdled        26    43.893   14.306    30.511    96.971
## Logged         37    61.403   18.532    43.502   128.583
```

```
ChaoShannon(spider)
```

```
##          Observed Estimator Est_s.e 95% Lower 95% Upper
## Girdled     2.490     2.627   0.110     2.490     2.842
## Logged      2.669     2.793   0.099     2.669     2.988
```

Curvas en vegan

```
library(vegan)
data("BCI")
head(BCI)[1:4,1:4]
```

```
##   Abarema.macradenia Vachellia.melanoceras Acalypha.diversifolia
## 1                  0                      0                     0
## 2                  0                      0                     0
## 3                  0                      0                     0
## 4                  0                      0                     0
##   Acalypha.macrostachya
## 1                     0
## 2                     0
## 3                     0
## 4                     0
```

Curvas en vegan

```
#por sitios
sac <- specaccum(BCI)
plot(sac, ci.type="polygon") #ver vegan para opciones
```

Curvas en vegan

```
#por individuos
sac <- specaccum(BCI, method = "rarefaction")
plot(sac, xvar = "individual", ci.type="polygon")
```