

# Ecología de comunidades en R- clase 1

Ph D.Stephanie Hereira-Pacheco  
CTBC  
UATx

14 - 03 - 2022

El propósito de este módulo del curso es aplicar el cálculo u obtención de algunos conocimientos teóricos en el módulo I. Hay varios paquetes y funciones en R que nos permitirán explorar nuestros datos ecológicos. Inicialmente veremos un poco de los métodos de muestreo y la autocorrelación espacial.

Luego, trataremos otros paquetes, quizás el más conocido y popularmente usado sea **vegan** sin embargo, trataremos de abordar algunos otros más que nos permitirán obtener más resultados y analizar nuestros datos de mejor manera.

# Muestreo estadístico en R<sup>1</sup>

En todo estudio estadístico distinguiremos entre población, (conjunto de sujetos con una o varias características que podemos medir y deseamos estudiar), y muestra (subconjunto de una población).

Supongamos que tenemos definido un transecto para nuestro muestreo o que tenemos pots o macetas en un área de nuestro invernadero o tal vez ya tenemos nuestros datos recolectados pero sólo queremos analizar una parte de estos. Para todos los casos anteriores debemos realizar un muestro estadístico.

Ya en clases anteriores se nos fue detallado los tipos de muestreo en detalle, en esta sesión les mostraré como utilizar R como herramienta de ayuda para lograr este muestreo.

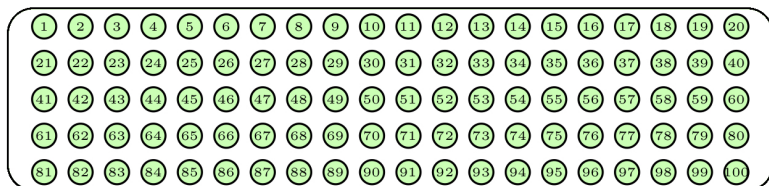
---

<sup>1</sup><https://joanby.github.io/bookdown-estadistica-inferencial/muestreo-estad%C3%ADstico.html>

## Muestreo aleatorio

R nos permite realizar un muestreo simple con la función `sample()`.

Para ilustrar mejor nuestro propósito que es estudiar los tipos de muestreo imaginemos que tenemos una urna conformada por bolas ennumeradas del 1 al 100 y queremos hacer un muestreo de esta población.



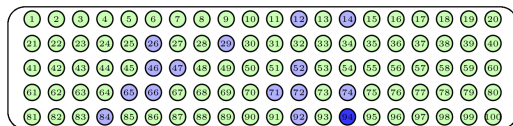
**Figure 1:** Urna de bolas del 1 al 100

Ahora queremos muestrear de manera aleatoria 15 bolas de estas 100.

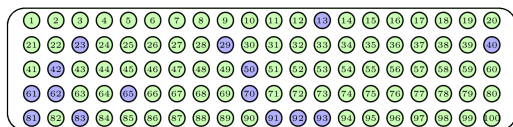
R nos da dos opciones de muestreo aleatorio, uno con reposición (haciendo `replace=TRUE`) y otro sin reposición (el caso contrario).

# Muestreo aleatorio

Para el muestreo con reposición de este ejemplo las bolas violetas son las escogidas para la muestra. La bola azul se ha escogido dos veces al ser el muestreo con reposición y sin reposición no hay repeticiones en nuestras bolas escogidas.



**Figure 2:** muestreo aleatorio con reemplazo



**Figure 3:** muestreo aleatorio sin reemplazo

# Muestreo aleatorio

Ahora vamos a hacerlo en R:

```
#muestreo aleatorio con reposición  
sample(1:100, 15, replace = TRUE)
```

```
## [1] 86 31 64 53 85 10 100 27 74 87 16 93 4 45 71
```

```
#muestreo aleatorio sin reposición  
sample(1:100, 15, replace = FALSE)
```

```
## [1] 83 5 16 22 10 93 62 49 94 26 42 37 13 3 53
```

## Muestreo aleatorio

Vamos a aplicar este tipo de muestreo a una data de ejemplo, ocuparemos la data de iris para hacer un ejemplo de muestreo en un conjunto de datos ya dado:

```
set.seed(123)
nrow(iris)
```

```
## [1] 150
```

```
flores_elegidas<- sample(1:150,5,replace=FALSE)
muestra_iris_flores_elegidas <- iris[flores_elegidas,]
muestra_iris_flores_elegidas
```

##	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
## 14	4.3	3.0	1.1	0.1	setosa
## 50	5.0	3.3	1.4	0.2	setosa
## 118	7.7	3.8	6.7	2.2	virginica
## 43	4.4	3.2	1.3	0.2	setosa
## 150	5.9	3.0	5.1	1.8	virginica

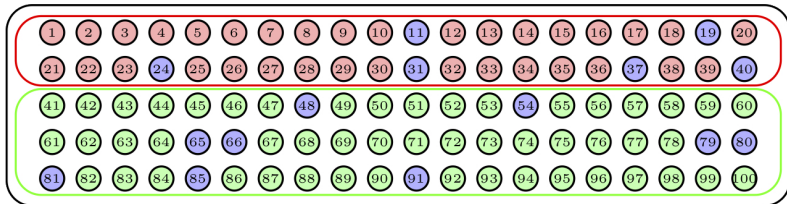
En este caso lo primero que hicimos fue utilizar la función **set.seed()** que se utiliza para que cada vez que realicemos un muestreo aleatorio y lo guardemos como un objeto de R este siempre sea el mismo, porque al ser aleatorio sino 'sembramos o establecemos la semilla' entonces cada vez que lo corramos nos dará diferente. Luego hacemos el muestreo con el número de observaciones de nuestra data usando la sintaxis de subconjunto para usarlo como un filtro.



# Muestreo aleatorio estratificado

Se utiliza cuando la población está clasificada por estratos.

Supongamos que nuestra urna de 100 bolas contiene 40 bolas de un color y 60 de otro color tal como muestra la figura:



**Figure 4:** muestreo aleatorio estratificado

# Muestreo aleatorio estratificado

Hagamoslo en R y vamos a muestrear 4 bolas de cada tipo de color:

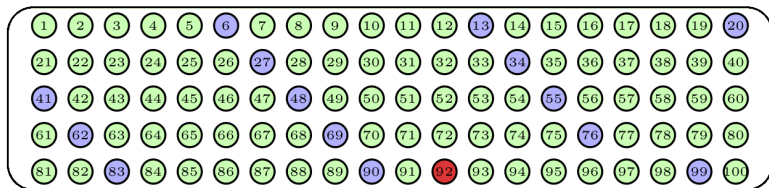
```
set.seed(234)
bolas_rojas <- sample(1:40, 4, replace = FALSE)
bolas_verdes<- sample(41:100, 4, replace = FALSE)

rbind(bolas_rojas, bolas_verdes)
```

```
##           [,1] [,2] [,3] [,4]
## bolas_rojas   33   31   34   38
## bolas_verdes  58   85   96   41
```

## Muestreo sistemático

pongamos que los individuos de una población vienen dados en forma de una lista ordenada. El muestreo sistemático consiste en tomarlos a intervalos constantes escogiendo al azar el primer individuo que elegimos.



**Figure 5:** muestreo sistemático

La figura anterior describe una muestra aleatoria sistemática de 15 bolas de nuestra urna de 100 bolas: hemos empezado a escoger por la bola roja oscura, que ha sido elegida al azar, y a partir de ella hemos tomado 1 de cada 7 bolas, volviendo al principio cuando hemos llegado al final de la lista de bolas.

# Muestreo sistemático

Hagamoslo en R:

```
set.seed(15)
primera_bola <- sample(1:100, 1)
#primera_bolas<- 92
incremento <-7
bolas_elegidas <- seq(from=primera_bola,by=incremento,length.out=5)
```

## Muestreo por conglomerados

Para realizarlo se escoge primero al azar unos subconjuntos en los que la población está dividida, a las que llamamos en este contexto conglomerados (clusters).

Supongamos que las 100 bolas de nuestra urna se agrupan en 20 conglomerados de 5 bolas cada uno según las franjas verticales.

Para obtener una muestra aleatoria por conglomerados de tamaño 15, escogeríamos al azar 3 conglomerados y la muestra estaría formada por sus bolas: los conglomerados escogidos están marcados en azul:

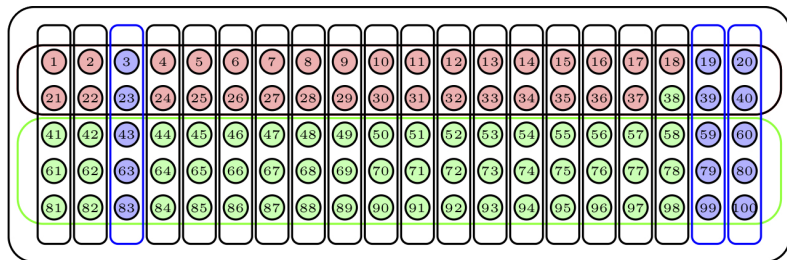


Figure 6: muestreo sistemático

## Autocorrelación espacial - Índice de Morán

Como vimos en clases anteriores hay diversos métodos para evaluar la autocorrelación espacial, tales como el índice de Morán, correlogramas y semivariogramas. En otras palabras medimos qué tan relacionados están los valores de una variable en función de las ubicaciones dónde se midieron.

En esta sesión veremos cómo calcular el índice de Morán. Para esto usaremos una base de datos de ejemplo en el cual varios medidores (32) ubicados en diversos puntos miden la cantidad de ozono y se quiere saber si hay o no autocorrelación espacial entre las ubicaciones. Vamos a cargar los datos y observarlos:

```
ozone <- read.table("https://stats.idre.ucla.edu/stat/r/faq/ozone.csv",  
                    sep="," ,header=T)
```

```
head(ozone)
```

##	Station	Av8top	Lat	Lon
## 1	60	7.225806	34.13583	-117.9236
## 2	69	5.899194	34.17611	-118.3153
## 3	72	4.052885	33.82361	-118.1875
## 4	74	7.181452	34.19944	-118.5347
## 5	75	6.076613	34.06694	-117.7514
## 6	84	3.157258	33.92917	-118.2097

## Autocorrelación espacial - Índice de Morán

Luego escogemos solo las variables geospaciales (coordenadas) y luego obtenemos el inverso de la matriz de distancia euclidiana, por último hacemos 0 las mediciones de la diagonal de la matriz:

```
ozone_coords<- cbind(ozone$Lon, ozone$Lat)
ozone_dist <- as.matrix(dist(cbind(ozone$Lon, ozone$Lat)))

ozone_dist_inv <- 1/ozone_dist
diag(ozone_dist_inv) <- 0

#ozone_dist_inv[1:5, 1:5]
```

## Autocorrelación espacial - Índice de Morán

Luego cargamos la paquetería **ape** que tiene la función **Moran.I()**:

```
library(ape)
Moran.I(ozone$Av8top, ozone_dist_inv)
```

```
## $observed
## [1] 0.2265501
##
## $expected
## [1] -0.03225806
##
## $sd
## [1] 0.03431138
##
## $p.value
## [1] 4.596323e-14
```

Basandonos en estos resultados, podemos rechazar la hipótesis nula de que hay una autocorrelación espacial cero presente en la variable Av8top en  $\alpha = 0.05$ .



Si nos interesa evaluar los otros métodos que son gráficos puedes consultar los paquetes:

- `ncf()` y `pgirmess()`: para correlogramas
- `geoR()`, `gstat()` y `sp()`: para semivariogramas

El paquete hillR contiene funciones para calcular la diversidad taxonómica, funcional y filogenética a través de los números de Hill.

Para instalar este paquete usamos el comando:

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

Usaremos la data de ejemplo para utilizar las funciones:

```
set.seed(123)  
dummy_data <- FD::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 `rand` del paquete '`ape`').

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.

Veamos:

```
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" "sp1"
```

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

```
library(hillR)
```

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

```
##  com1  com2  com3  com4  com5  com6  com7  com8  com9 com10
##      4      3      3      2      3      5      3      4      5      4
```

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

```
##           com1      com2      com3      com4      com5      com6      com7
## Q      0.4016663 0.1922618 0.2780442 0.1146261 0.3816159 0.404177 0.29341
## FDis 0.3481687 0.1670560 0.2375808 0.1146261 0.3211366 0.330233 0.25327
## D_q  4.0974923 3.6518111 3.2454591 3.0237158 3.0655375 5.233241 3.14700
## MD_q 1.6458245 0.7021037 0.9023810 0.3465969 1.1698580 2.115156 0.92337
## FD_q 6.7437533 2.5639502 2.9286406 1.0480104 3.5862436 11.069121 2.90587
##           com8      com9      com10
## Q      0.3343662 0.4156546 0.3844765
## FDis 0.2877931 0.3421687 0.3503927
## D_q  4.3998540 5.2114653 4.1694097
## MD_q 1.4711625 2.1661695 1.6030400
## FD_q 6.4729004 11.2889174 6.6837303
```

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

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

```
##      com1      com2      com3      com4      com5      com6      com7      com8
## 5.430079 4.684280 4.461773 2.551395 5.830078 6.088533 4.763594 6.046474
##      com9      com10
## 6.262164 5.080340
```

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

Los resultados que nos da cada función son:

- ❶ **hill\_taxa()** nos da un vector con el va valor de diversidad alfa para cada sitio o muestra,  $q = 0$  (por defecto) para obtener la riqueza de especies,  $q = 1$  para obtener la entropía de shannon y  $q = 2$  nos dará el inverso de simpson.
- ❷ **hill\_func** nos dará una matrix con la información de:
  - $Q$  :  $Q$  de Rao,
  - $D_q$  : el numero efectivo de especies 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
- ❸ **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.



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

Si queremos calcular a otra q solo ponemos la que queremos, por ejemplo:

```
hill_taxa(comunidades, q = 1)
```

```
##      com1      com2      com3      com4      com5      com6      com7      com8
## 3.363586 2.460233 2.749459 1.457569 2.951152 4.395212 2.906907 3.217222
##      com9      com10
## 4.341153 3.086164
```

```
hill_taxa(comunidades, q = 2)
```

```
##      com1      com2      com3      com4      com5      com6      com7      com8
## 2.909091 2.133333 2.571429 1.280000 2.909091 4.121495 2.813953 2.688889
##      com9      com10
## 3.903226 2.666667
```

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

En este caso será a través de todos los sitios o muestras.

Este script calcula la diversidad gamma, alfa, y beta a través de todas las comunidades o muestras así como su similitud. Si  $comm > 2$  la gamma diversidad es la diversidad juntada (pooled) del ensamble y la alfa es el promedio de la diversidad a través de todos los sitios y la beta es a través de todas las comunidades.

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

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

```
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      0.3888888
```

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

```
##      q raoQ_gamma FD_gamma FD_alpha FD_beta local_similarity region_similarity
## 1 0  0.4529152 29.66099 14.15941 2.09479      0.9889415      0.4720
```

```
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
```

# Calcular la diversidad pareada taxonómica, funcional y filogenética

En este caso será a través de todos los sitios o muestras.

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

```
hill_taxa_parti_pairwise(comunidades, q = 0, show_warning = FALSE, .progres
```

```
## # A tibble: 45 x 8
```

```
##       q site1 site2 TD_gamma TD_alpha TD_beta local_similarity region_s
##   <dbl> <chr> <chr>   <dbl>   <dbl>   <dbl>         <dbl>
## 1     0 com1  com2     6     3.5    1.71         0.286
## 2     0 com1  com3     5     3.5    1.43         0.571
## 3     0 com2  com3     5     3     1.67         0.333
## 4     0 com1  com4     5     3     1.67         0.333
## 5     0 com2  com4     5     2.5    2            0
## 6     0 com3  com4     4     2.5    1.6           0.4
## 7     0 com1  com5     6     3.5    1.71         0.286
## 8     0 com2  com5     4     3     1.33         0.667
## 9     0 com3  com5     6     3     2            0
## 10    0 com4  com5     4     2.5    1.6           0.4
```

```
## # ... with 35 more rows
```

## Calcular la diversidad pareada taxonómica, funcional y filogenética

```
hill_func_parti_pairwise(comunidades, funciones, q = 0, show_warning = FALSE)
```

```
## # A tibble: 45 x 8
```

```
##       q site1 site2 FD_gamma FD_alpha FD_beta local_similarity region_s
```

```
##    <dbl> <chr> <chr>    <dbl>    <dbl>    <dbl>          <dbl>
```

```
##  1      0 com1  com2     15.9     10.3     1.54          0.821
```

```
##  2      0 com1  com3     10.7      7.96     1.35          0.883
```

```
##  3      0 com2  com3     11.1      7.03     1.58          0.807
```

```
##  4      0 com1  com4     11.6      7.90     1.47          0.843
```

```
##  5      0 com2  com4     11.7      6.85     1.70          0.765
```

```
##  6      0 com3  com4      6.60      4.45     1.48          0.839
```

```
##  7      0 com1  com5     17.3     11.3     1.54          0.821
```

```
##  8      0 com2  com5      7.86      5.92     1.33          0.891
```

```
##  9      0 com3  com5     16.2      9.72     1.66          0.780
```

```
## 10      0 com4  com5      8.00      5.32     1.50          0.832
```

```
## # ... with 35 more rows
```

## Calcular la diversidad pareada taxonómica, funcional y filogenética

```
hill_phylo_parti_pairwise(comunidades, arbol, q = 0, show_warning = FALSE,

## # A tibble: 45 x 8
##       q site1 site2 PD_gamma PD_alpha PD_beta local_similarity region_s
##   <dbl> <chr> <chr>   <dbl>   <dbl>   <dbl>         <dbl>
## 1     0 com1  com2     6.79     5.06     1.34         0.657
## 2     0 com1  com3     6.14     4.95     1.24         0.759
## 3     0 com2  com3     6.75     4.57     1.48         0.523
## 4     0 com1  com4     6.38     3.99     1.60         0.400
## 5     0 com2  com4     7.13     3.62     1.97         0.0284
## 6     0 com3  com4     5.42     3.51     1.54         0.455
## 7     0 com1  com5     7.04     5.63     1.25         0.750
## 8     0 com2  com5     6.54     5.26     1.24         0.756
## 9     0 com3  com5     7.71     5.15     1.50         0.502
## 10    0 com4  com5     6.42     4.19     1.53         0.467
## # ... with 35 more rows
```

## 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. También está disponible una versión en línea de iNEXT Online para usuarios sin experiencia en R.

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).

Para cada medida de diversidad, iNEXT utiliza la muestra observada de datos de abundancia o incidencia para calcular las estimaciones de diversidad para muestras enrarecidas y extrapoladas y los intervalos de confianza del 95 % (predeterminados) asociados

## iNEXT: (iNterpolation and EXTrapolation)

Además representa 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 enrarecidas y extrapoladas hasta el doble del tamaño de la muestra de referencia (por defecto) 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. El tamaño de la muestra se refiere al número de individuos en una muestra para datos de abundancia, mientras que se refiere al número de unidades de muestreo para datos de incidencia.
- 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 (por defecto) 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')
```

# iNEXT: (iNterpolation and EXTrapolation)

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

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 'bootstrap',
- `conf`: el intervalo de confianza,
- `nboot`: número de replicaciones de 'bootstrap'

## iNEXT: (iNterpolation and EXTrapolation)

Correremos el ejemplo con la data del paquete, denominada **bird**:

```
data(bird)
str(bird)
```

```
## 'data.frame':    41 obs. of  2 variables:
## $ North.site: int  0 0 41 0 3 1 5 4 4 11 ...
## $ South.site: int  3 18 31 2 1 2 5 1 6 32 ...
```

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

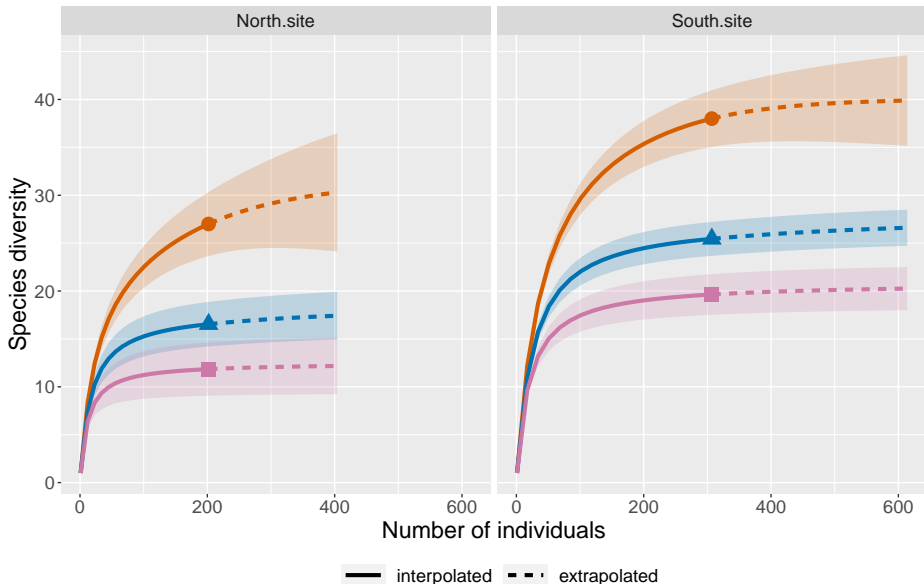
## iNEXT: (iNterpolation and EXTrapolation)

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

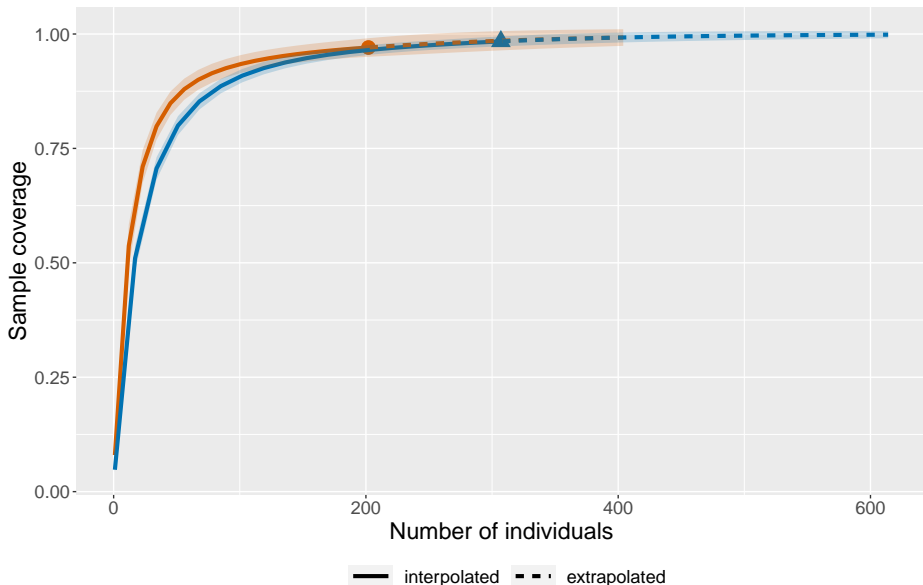
# iNEXT: (iNterpolation and EXTrapolation)

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



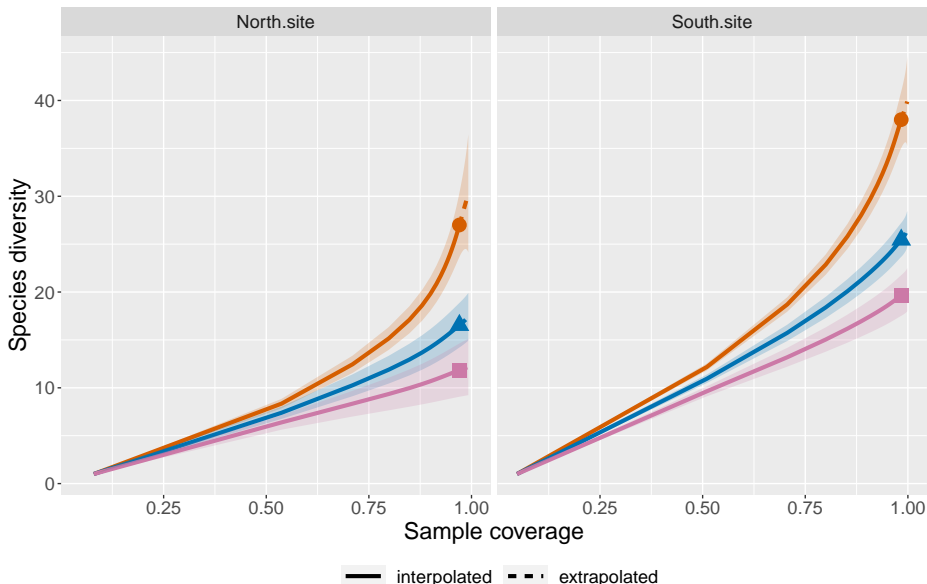
# iNEXT: (iNterpolation and EXTrapolation)

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



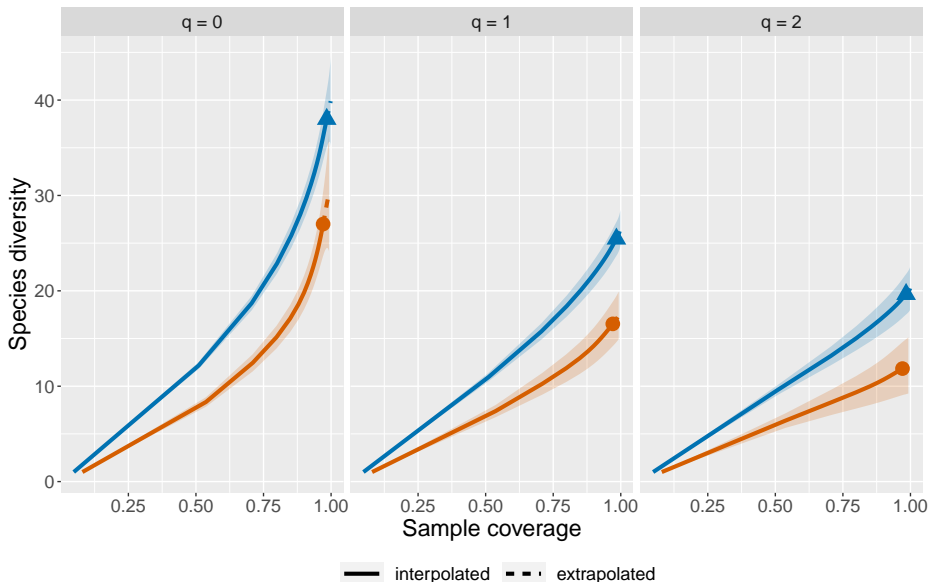
# iNEXT: (iNterpolation and EXTrapolation)

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



# iNEXT: (iNterpolation and EXTrapolation)

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





## iNEXT: (iNterpolation and EXTrapolation)

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

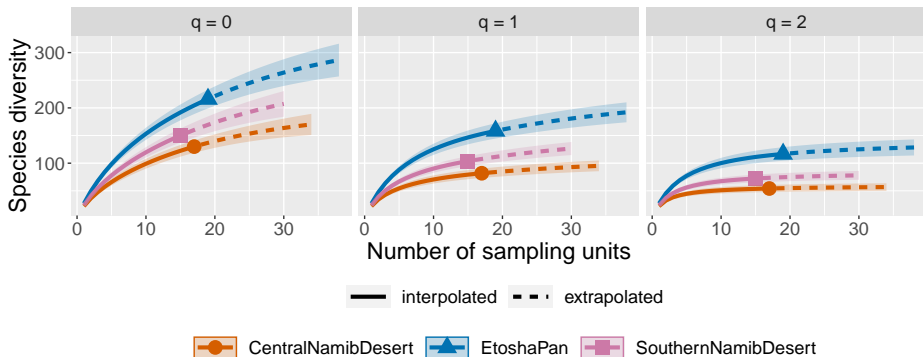
```
estimateD(bird, datatype="abundance", base="coverage", conf=0.95)
```

```
##           site    m      method order          SC          qD      qD.LCL      qD.UC
## 1 North.site 404 extrapolated      0 0.9922391 30.30001 18.516722 42.0833
## 2 North.site 404 extrapolated      1 0.9922391 17.42200 14.670012 20.1739
## 3 North.site 404 extrapolated      2 0.9922391 12.17643  9.616829 14.7360
## 4 South.site 401 extrapolated      0 0.9922464 39.08021 28.448060 49.7123
## 5 South.site 401 extrapolated      1 0.9922464 25.93440 23.007071 28.8617
## 6 South.site 401 extrapolated      2 0.9922464 19.92379 17.077747 22.7698
##           goalSC
## 1 0.9922391
## 2 0.9922391
## 3 0.9922391
## 4 0.9922391
## 5 0.9922391
## 6 0.9922391
```

# iNEXT: (iNterpolation and EXTrapolation)

Vamos a ver un ejemplo pequeño con datos de incidencia (presencia/ausencia):

```
data(ciliates)
#str(ciliates)
out2 <- iNEXT(ciliates, q=c(0,1,2), datatype="incidence_raw")
ggiNEXT(out2, facet.var="order", type=1)
```



## iNEXT: (iNterpolation and EXTrapolation)

Este paquete tambien cuenta con otras funciones de interes, 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

## hilldiv()

hilldivs un paquete de R que proporciona un conjunto de funciones para asistir en el análisis de la diversidad basados en números de Hill, usando tablas de especies o de OTU/ASV y árboles filogenéticos como entradas. El paquete incluye funciones para la medición de la (filo)diversidad, el trazado del perfil de la (filo)diversidad, la comparación de la (filo)diversidad entre muestras y grupos, la partición de la (filo)diversidad y la medición de la (di)similitud. Todos estos basados en números de Hill basados en la abundancia y en la incidencia. Para encontrar más información sobre el marco de los números de Hill aplicados a diversidad lee el siguiente artículo: [artículo\\_Hill](#).

Para instalar este paquete:

```
#versión de CRAN  
install.packages("hilldiv")  
  
#versión en desarrollo  
install.packages("devtools")  
library(devtools)  
install_github("anttonalberdi/hilldiv")
```

Cargamos la librería:

```
library(hilldiv)
```

La función principal es **hilldiv()** y la data de ejemplo es 'bat.diet' que es una otutable data con diferentes individuos de diferentes especies de murciélagos.

```
data(bat.diet.otutable)
data(bat.diet.tree)
data(bat.diet.hierarchy)
```

```
bat.diet.otutable[1:3, 1:4]
```

```
##      Msc1 Msc2 Msc3 Msc4
## OTU1    0    0    0    0
## OTU2    0    0    0    0
## OTU3    0    0    0    0
```

```
class(bat.diet.tree)
```

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

```
head(bat.diet.hierarchy)
```

```
##      Sample      Species
## 1   Msc1 Miniopterus schreibersii
## 2   Msc2 Miniopterus schreibersii
## 3   Msc3 Miniopterus schreibersii
## 4   Msc4 Miniopterus schreibersii
## 5   Msc5 Miniopterus schreibersii
## 6   Mmy1           Myotis myotis
```

## hilldiv()

Usemos la función principal para calcular la diversidad filogenética y taxonómica:

*#Basado en abundancia*

```
hill_div(bat.diet.otutable,0)
```

```
## Msc1 Msc2 Msc3 Msc4 Msc5 Mmy1 Mmy2 Mmy3 Mmy4 Mmy5 Rme1 Rme2 Rme3 Rme4 Rm
##      3    11     9     1    11     5    13     9    12     2     2     1    10     4
## Mda2 Mda3 Mda4 Mda5 Mca1 Mca2 Mca3 Mca4 Mca5 Reu1 Reu2 Reu3 Reu4 Reu5 Me
##      7    10    22     2    26    19    13    10     2    10    13     9    12     4
## Mem3 Mem4 Mem5 Rhi1 Rhi2 Rhi3 Rhi4 Rhi5
##     13    20    10    24    13    17    20    15
```

```
hill_div(bat.diet.otutable, 1, bat.diet.tree)
```

```
##      Msc1      Msc2      Msc3      Msc4      Msc5      Mmy1      Mmy2      Mmy3
## 1.152278 1.199440 1.071783 1.000000 1.120999 1.202859 1.011519 1.237840
##      Mmy4      Mmy5      Rme1      Rme2      Rme3      Rme4      Rme5      Mda1
## 1.223297 1.001831 1.000436 1.000000 1.017877 1.006787 1.058370 1.003925
##      Mda2      Mda3      Mda4      Mda5      Mca1      Mca2      Mca3      Mca4
## 1.265173 1.669337 1.475675 1.077681 1.250052 1.401592 1.030458 1.174893
##      Mca5      Reu1      Reu2      Reu3      Reu4      Reu5      Mem1      Mem2
## 1.002285 1.098038 1.048106 1.111219 1.224831 1.050408 1.220784 1.013132
```

*#Basado en incidencia*

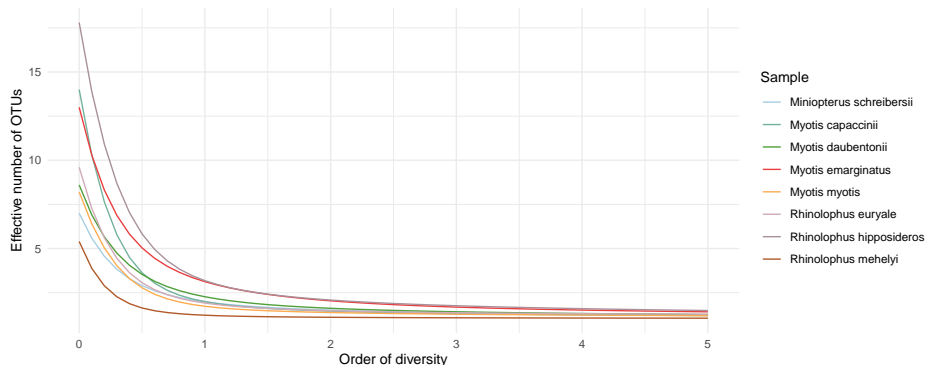
```
hill_div(to.incidence(bat.diet.otutable, bat.diet.hierarchy), 2)
```

##	Miniopterus schreibersii	Myotis myotis	Rhinolophus mehel
##	29.87805	39.09302	15.510
##	Myotis daubentonii	Myotis capaccinii	Rhinolophus eurya
##	39.34043	58.33333	44.307
##	Myotis emarginatus	Rhinolophus hipposideros	
##	52.16049	45.26286	



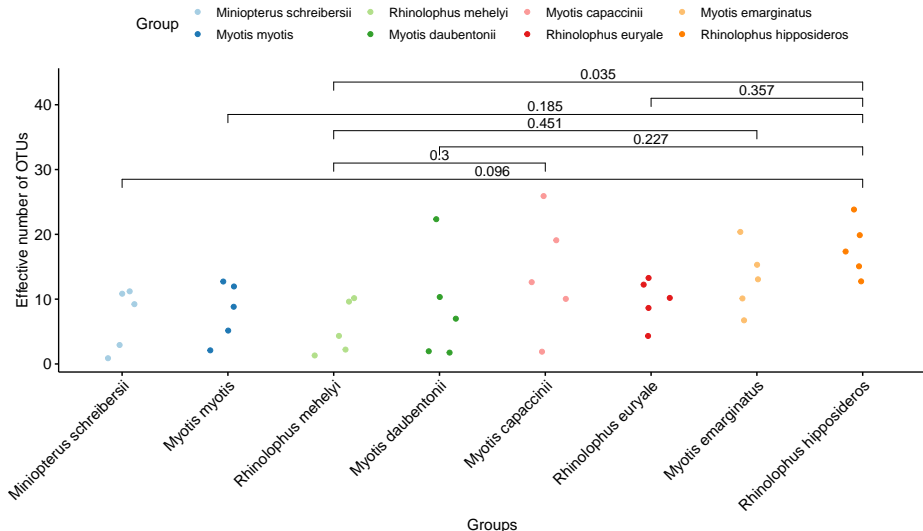
También podemos graficar esto haciendo un perfil de diversidad, así:

```
profile.multiplegroups <- div_profile(bat.diet.otutable, hierarchy=bat.diet.
div_profile_plot(profile.multiplegroups)
```



Además, tiene una función para comparaciones pareadas que evalúa automáticamente si los datos cumplen las propiedades de las estadísticas paramétricas y realiza la prueba adecuada en consecuencia: T de Student, ANOVA, Wilcoxon o Kruskal-Wallis. Si el argumento `post hoc` se establece como `TRUE`, las comparaciones de grupos múltiples se complementan con pruebas post hoc por pares, ya sea la prueba de Tukey (paramétrica) o la prueba de Dunn con corrección de Benjamini-Hochberg (no paramétrica).

```
pareada<-div_test(bat.diet.otutable,qvalue=0,hierarchy=bat.diet.hierarchy,p
div_test_plot(pareada,chart="jitter",posthoc=TRUE,threshold=0.5)
```



Por otro lado, tiene una función que calcula la cobertura estimada en el marco de los números de Hill, en otras palabras y en este caso, evaluar si la profundidad de secuenciación de cada muestra es suficiente para recuperar toda la diversidad de una muestra.

```
head(depth_cov(bat.diet.otutable,qvalue=1))
```

##		Depth	Observed	Estimated	Coverage
##	Msc1	15200	1.51	1.51	99.99
##	Msc2	28911	3.12	3.12	99.98
##	Msc3	29585	2.01	2.01	99.99
##	Msc4	15942	1.00	1.00	100.00
##	Msc5	12523	3.24	3.24	99.96
##	Mmy1	41634	2.39	2.39	100.00

Tiene otras funciones que pueden ser de interés tales como:

- `alpha_div()`: Cálculo de diversidad alfa (sistema)
- `gamma_div()`: Cálculo de diversidad gamma
- `beta_dis()`: Cálculo de (des)similitud basado en diversidades beta
- `UqN()`: Cálculo de traslape de tipo Jaccard a partir de diversidades beta
- `CqN()`: Superposición/Traslape tipo Sørensen de diversidades beta
- `SqN()`: Complemento del recambio de tipo Jaccard de diversidades beta
- `VqN()`: Complemento del recambio de tipo Sørensen de diversidades beta

## Visualizar nuestros datos de diversidad









```
library(tidyverse)
library(ggpubr)
q0<-hill_div(bat.diet.otutable,0) %>% as.data.frame() %>% mutate(
  qs="q0") %>% rownames_to_column(var = "Sample")
q1<-hill_div(bat.diet.otutable,1)%>% as.data.frame() %>% mutate(
  qs="q1") %>% rownames_to_column(var = "Sample")
q2<-hill_div(bat.diet.otutable,2)%>% as.data.frame() %>% mutate(
  qs="q2") %>% rownames_to_column(var = "Sample")

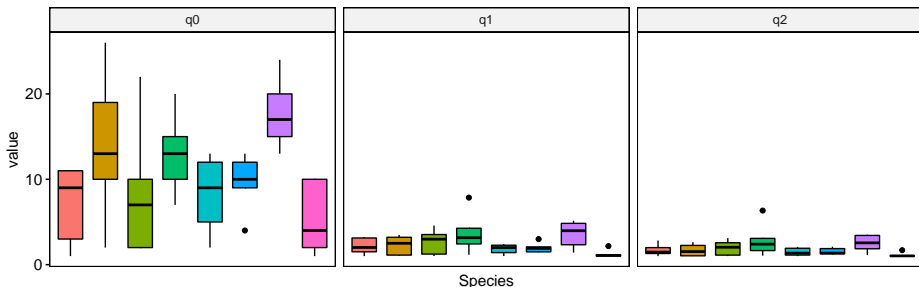
diversidad_data<- rbind(q0,q1,q2) %>% full_join(bat.diet.hierarchy)
colnames(diversidad_data)[2]<- "value"
```

# Visualizar nuestros datos de diversidad

```
ggboxplot(data = diversidad_data, x = "Species", y = "value",  
          fill = "Species", facet.by = "qs")+theme(  
          axis.text.x = element_blank(), axis.ticks.x = element_blank())
```

Species









 <i>Miniopterus schreibersii</i>	 <i>Myotis daubentonii</i>	 <i>Myotis myotis</i>	 <i>Rhinolophus hipposideros</i>
 <i>Myotis capaccinii</i>	 <i>Myotis emarginatus</i>	 <i>Rhinolophus euryale</i>	 <i>Rhinolophus mehelyi</i>



# Visualizar nuestros datos de diversidad

```
ggbarplot(data = diversidad_data, x = "Species", y = "value",  
  fill = "Species", facet.by = "qs", add = "mean_sd")+theme(  
  axis.text.x = element_blank(), axis.ticks.x = element_blank())
```

Species

 <i>Miniopterus schreibersii</i>	 <i>Myotis daubentonii</i>	 <i>Myotis myotis</i>	 <i>Rhinolophus hipposideros</i>
 <i>Myotis capaccinii</i>	 <i>Myotis emarginatus</i>	 <i>Rhinolophus euryale</i>	 <i>Rhinolophus mehelyi</i>

