Ejemplo sobre los pasos que sigue el script scripts/centrality_measures_keystone_species_tomate.R

2022-10-21

#Introducción

Este documento RMarkdown representa un ejemplo sencillo para entender el uso del script ../scripts/centrality_measures_keys así como una versión más breve del RMarkdown ../docs/centrality_measures_keystone_species_tomate.R. Los datos con los que se corre este ejemplo son una versión de "juguete" (una pequeña componente conexa) de los datos presentes en el archivo data/table.from_maiz.txt, y de su red correspondiente data/networks/maiz_species_raw_network.csv.

#Carga de paquetes

Aquí se presentan los paquetes que serán necesarios para nuestro análisis. "vegan" se usará para calcular disimilaridades de Bray-Curtis (βs-diversidad) entre las muestras. "igraph" será utilizada para calcular la componente conexa principal de la red de correlación y las medidas de centralidad de los nodos. "apcluster" será usada para descartar muestras "ruido". "plyr" se usará para simplificar la aplicación de funciones a listas.

```
if (!require(vegan)) install.packages('vegan')
```

```
## Loading required package: vegan
```

```
## Loading required package: permute
```

```
## Loading required package: lattice
```

```
## This is vegan 2.6-4
```

```
library(vegan)
if (!require(igraph)) install.packages('igraph')
```

```
## Loading required package: igraph
```

```
##
## Attaching package: 'igraph'
```

```
## The following object is masked from 'package:vegan':
##
##     diversity
```

```
## The following object is masked from 'package:permute':
##
##     permute
```

```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union
```

```
library(igraph)
if (!require(apcluster)) install.packages('apcluster')
```

```
## Loading required package: apcluster

##
## Attaching package: 'apcluster'

## The following object is masked from 'package:igraph':
##
##     similarity

## The following object is masked from 'package:stats':
##
##     heatmap
```

```
library(apcluster)
if (!require(plyr)) install.packages('plyr')
```

```
## Loading required package: plyr
```

```
library(plyr)
```

#Carga y análisis de las muestras

Como primer paso se carga la tabla de muestras como un dataframe de R. Los "nombres" de los OTUs, presentes en la primera columna, se refieren al ID de NCBI a nivel especie del respectivo OTU. Esta columna se usa para nombrar las filas del dataframe. Las demás columnas se refieren a las instancias (reads asignados) de los OTUs por muestra. Los datos son normalizados por columna.

```
setwd(".")
data <- read.table("./table_maiz_toy.txt" , row.names = 1, header = FALSE , sep = "," )
for (i in 1:dim(data)[2]){
  data[,i] <- data[,i]/sum(data[,i])
}
head(data)
```

```
##                V2        V3        V4        V5        V6        V7
## 2487344 0.2629921 0.1004391 0.2653563 0.3037543 0.2375566 0.2658824
## 529     0.2803150 0.1512193 0.2628993 0.2593857 0.1832579 0.3011765
## 419475  0.1543307 0.3153789 0.1621622 0.1262799 0.2828054 0.1411765
## 571256  0.1905512 0.2734747 0.1744472 0.1706485 0.1606335 0.1411765
## 239106  0.1118110 0.1594880 0.1351351 0.1399317 0.1357466 0.1505882
```

##Agrupación de muestras y descarte de “outliers”

A continuación se agrupan las muestras según los metadatos presentes en el archivo “data/fastp_metadat.csv”. En este caso, cada uno de los vectores se corresponde a una “etapa fenológica” del cultivo del tomate. Los elementos de las vectores son los nombres de las columnas del dataframe “data” correspondientes a cada grupo.

```
produccion <- c("V2", "V3","V4" )
formacion_espiga <- c("V5","V6","V7")
```

El análisis aquí planteado busca maximizar el parecido entre las muestras de un mismo grupo. Por esa razón se excluirán las muestras que al estar muy separadas de las demás, y por ende de su propio grupo, probablemente generen ruido. Como se mencionó anteriormente la “distancia” usada en este script es la disimilaridad de Bray-Curtis. Las muestras que se considerarán “outliers” son las que queden en un clúster unitario tras una clusterización por propagación de afinidad.

```
#Se crea la "matriz" de disimilaridades de bray-curtis
bc_dist <- vegdist(t(data), method = "bray")

#Dado que la propagación de afinidad requiere de similaridades, a 1 restamos la distancia bray-curtis p
s <- 1 - bc_dist
s <- as.matrix(s)
#La función apcluster hace la clusterización
clustering <- apcluster(s)
#El objeto de clustering que nos interesa es el llamado clusters
clusters <- clustering@clusters
#Aquí preguntamos qué clústeres tienen tamaño al menos uno (no outliers) y nos quedamos con el vector d
filtro_0 <- lapply(clusters, length)
clusters_no_outliers <- clusters[which(filtro_0 > 1)]
no_outliers <- unlist(clusters_no_outliers)
no_outliers <- names(no_outliers)
```

En la siguiente celda finalmente se reducen las muestras (en la variable “data”) y los grupos fenológicos a las no “outliers”.

```
data <- data[,no_outliers]

grupos <- list()
grupos[[1]] <- intersect(produccion,no_outliers)
grupos[[2]] <- intersect(formacion_espiga ,no_outliers)

#Grupos necesarios para el análisis auc
len_list <- llply(grupos , length)
len_list <- which(len_list > 1)
grupos <- grupos[len_list]
```

##Filtración de OTUs según su aparición en las muestras

Se filtrarán los OTUs que aparezcan en una sola muestra, dado que estos OTUs tendrán un grado artificialmente alto en la red de correlación sin que esto refleje una “centralidad ecológica”. Dado que los nodos en el archivo “data/networks/tomate_species_raw_network.csv” están ya numerados desde 0, una columna de nodos se agrega a data para no perder la biyección entre otus y su nodo en la red.

Filtraciones de este tipo se pueden generalizar a filtrar otus que aparezcan, por ejemplo, al menos 10 veces en solamente menos de 10% de las muestras. En futuros análisis estos umbrales podrían ajustarse.

```
data$nodos <- 0:(dim(data)[1]-1)

filt <- c()
for (i in 1:dim(data)[1]) {

    v_i <- as.vector(data[i,1:(dim(data)[2]-1)])
    #el siguiente 1 es filtro
      if (length(v_i [ v_i > 0 ]) > 1 ) {
        filt <- c(filt, i)
        }
     }

data <- data[filt,]
```

#Carga y análisis de la red

El siguiente paso es cargar la red de coocurrencia correspondiente a nuestras muestras. En este caso se trata de una red de correlación de Spearman, que está en el archivo "data/networks/tomate_species_raw_network.csv". Se trata de una red ponderada no dirigida cuyos pesos se encuentran entre -1 y 1. Cada una de las filas de este archivo se refiere a una arista de la red: las primeras dos entradas son los nodos correspondientes y la tercera es su peso. Este archivo se cargará como dataframe para el análisis previo a su conversión a red binaria no dirigida como objeto igraph.

```
red <- read.csv("./red_maiz_toy.txt")
red <- red[,1:3]
```

##Conversión a red binaria y limpieza previa al análisis igraph

En la siguiente celda se filtran las aristas de la red con el siguiente criterio: los nodos deben estar en data$nodos, es decir, deben haber sobrevivido a la filtración de OTUs, y su correlación debe ser positiva. En la red final estas aristas serán consideradas de peso 1.

```
edges <- c()
for (i in 1:dim(red)[1]) {


  if (is.element(red[i,1], data$nodos) && is.element(red[i,2], data$nodos) && red[i,3] > 0  ){
    edges <- c(edges , i)
  }
}

red <- red[edges, 1:2]
```

En las siguientes dos celdas las etiquetas, hasta ahora numéricas, se transforman en los dataframes red y data a "strings" del tipo "v_#" para simplificar el trabajo con la librería igraph.

```
for (i in 1:dim(red)[1]){
  for (j in 1:dim(red)[2]){
    red[i,j] <- paste("v_",as.character(red[i,j]))
  }
}
```

```
for (i in 1:dim(data)[1]){
  data[i,"nodos"] <- paste("v_" , as.character(data[i,"nodos"]))
}
```

##Conversión de la red a objeto igraph y obtención de componente conexa principal

Desde el dataframe, que esencialmente ya está convertido en una "lista" de aristas, obtenemos un objeto de igraph.

```
net_work <- graph_from_edgelist(as.matrix(red) , directed = FALSE )
```

En la siguiente celda se obtienen las componentes conexas de la red, con la función "components", se ubica la mayor de ellas y se obtiene finalmente el vector de los nodos que pertenecen a ella. Finalmente convertimos a la red en su mayor componente conexa. Este nuevo filtrado se llevó a cabo porque el aparentemente el cálculo de medidas de centralidad lo requeriría, y porque esta componente conexa resultó ser considerablemente más grande que el resto de ellas.

```
compo_conexas <- components(net_work)
size_compo_conexas <- compo_conexas$csize
princ <- which(size_compo_conexas == 5)
pertenencia <- compo_conexas$membership
compo_princ <- which(pertenencia == princ )
compo_princ <- names(compo_princ)

net_work <- induced_subgraph(net_work, compo_princ ,"auto")
```

En la siguiente celda reajustamos nuestros datos a la componente principal, que es nuestra nueva red.

```
filtro_componente <- c()
for (i in 1:dim(data)[1]){
  if(is.element(data[i,"nodos"],compo_princ)){
    filtro_componente <- c(filtro_componente, i)
  }
}


data <- data[filtro_componente,]
```

#Cálculo de medidas de centralidad

En esta búsqueda de potenciales especies clave se utilizaron tres medidas de centralidad de red para cada nodo v:

- Grado, que es el número de aristas en las que participa v,
- Centralidad de cercanía, que es el inverso de la suma de las distancias de v a los demás, y
- Centralidad de intermediación, que es la proporción de caminos mínimos entre dos nodos que pasan por v, sobre el total de estos.

Cada una de las siguientes tres celdas cálcula las respectivas centralidades de nuestros OTUs y agrega la correspondiente columna al dataframe "data". La última de ellas, correspondiente al cálculo de centralidad de intermediación, tardó entre tres y cuatro horas en correr.

```
degrees <- c()
for (i in 1:dim(data)[1]) {
  d_i <- degree(net_work, data[i,"nodos"])
  degrees <- c(degrees, d_i)
}
data$degrees <- degrees
```

```
closeness_cent <- c()
for (i in 1:dim(data)[1]) {
  c_i <- closeness(net_work, data[i,"nodos"])
  closeness_cent <- c(closeness_cent, c_i)
}
data$closeness <- closeness_cent
```

```
betweenness_cent <- c()
for (i in 1:dim(data)[1]) {
  b_i <- betweenness(net_work, data[i,"nodos"])
  betweenness_cent <- c(betweenness_cent, b_i)
}
data$betweenness <- betweenness_cent
```

En la siguiente celda se crean tres copias de data donde las filas (OTUs) se ordenan de forma decreciente según las medidas de centralidad. Dadas la importancia que estos datos pueden tener para otros análisis y la particular tardanza de su cómputo, estas tres copias se guardan en archivos .csv.

```
data_deg <- data[order(data$degrees, decreasing = TRUE),]
data_close <- data[order(data$closeness , decreasing = TRUE),]
data_between <- data[order(data$betweenness, decreasing = TRUE),]

data_deg
```

```
##                V2        V4        V5        V6        V7 nodos degrees
## 529     0.2803150 0.2628993 0.2593857 0.1832579 0.3011765  v_ 1       3
## 571256  0.1905512 0.1744472 0.1706485 0.1606335 0.1411765  v_ 3       3
## 239106  0.1118110 0.1351351 0.1399317 0.1357466 0.1505882  v_ 4       3
## 2487344 0.2629921 0.2653563 0.3037543 0.2375566 0.2658824  v_ 0       2
## 419475  0.1543307 0.1621622 0.1262799 0.2828054 0.1411765  v_ 2       1
##         closeness betweenness
## 529     0.2000000           1
## 571256  0.2000000           3
## 239106  0.2000000           1
## 2487344 0.1428571           0
## 419475  0.1250000           0
```

```
data_close
```

```
##                V2        V4        V5        V6        V7 nodos degrees
## 529     0.2803150 0.2628993 0.2593857 0.1832579 0.3011765  v_ 1       3
## 571256  0.1905512 0.1744472 0.1706485 0.1606335 0.1411765  v_ 3       3
## 239106  0.1118110 0.1351351 0.1399317 0.1357466 0.1505882  v_ 4       3
## 2487344 0.2629921 0.2653563 0.3037543 0.2375566 0.2658824  v_ 0       2
```

```
## 419475  0.1543307 0.1621622 0.1262799 0.2828054 0.1411765  v_ 2       1
##         closeness betweenness
## 529     0.2000000           1
## 571256  0.2000000           3
## 239106  0.2000000           1
## 2487344 0.1428571           0
## 419475  0.1250000           0
```

```
data_between
```

```
##                V2        V4        V5        V6        V7 nodos degrees
## 571256  0.1905512 0.1744472 0.1706485 0.1606335 0.1411765  v_ 3       3
## 529     0.2803150 0.2628993 0.2593857 0.1832579 0.3011765  v_ 1       3
## 239106  0.1118110 0.1351351 0.1399317 0.1357466 0.1505882  v_ 4       3
## 2487344 0.2629921 0.2653563 0.3037543 0.2375566 0.2658824  v_ 0       2
## 419475  0.1543307 0.1621622 0.1262799 0.2828054 0.1411765  v_ 2       1
##         closeness betweenness
## 571256  0.2000000           3
## 529     0.2000000           1
## 239106  0.2000000           1
## 2487344 0.1428571           0
## 419475  0.1250000           0
```

```
#write.csv(data_deg,"name_bydegree.csv", row.names = TRUE)
#write.csv(data_close,"name_bycloseness.csv", row.names = TRUE)
#write.csv(data_between,"name_bybetweenness.csv", row.names = TRUE)
```

#Primer reporte de OTUs clave

A partir de las tablas ordenadas por medidas de centralidad es posible conseguir un primer reporte de posibles candidatos a OTU clave. Serían los de simultáneamente mayor grado, mayor centralidad de cercanía y menor centralidad de intermediación. En esta versión del script “mayor” (resp. “menor”) significa significa mayor (resp. “ “) a la mediana.

```
hdeg <- which(data$degrees >= quantile(data$degrees , probs = seq(0, 1, 0.5))[2])
hclose <- which(data$closeness >= quantile(data$closeness , probs = seq(0, 1, 0.5))[2])
lbetween <- which(data$betweenness <= quantile(data$betweenness , probs = seq(0, 1, 0.5))[2])

results_1 <- intersect(hdeg,hclose)
results_1 <- intersect(results_1 , lbetween)
data_report_1 <- data[results_1,]
print(data_report_1)
```

```
##             V2        V4        V5        V6        V7 nodos degrees closeness
## 529    0.280315 0.2628993 0.2593857 0.1832579 0.3011765  v_ 1       3       0.2
## 239106 0.111811 0.1351351 0.1399317 0.1357466 0.1505882  v_ 4       3       0.2
##        betweenness
## 529              1
## 239106           1
```

#Funciones para la búsqueda de candidatos a especie clave

La función calcula el área bajo la curva de las medidas de centralidad. Sus argumentos son un dataframe, y el nombre de alguna medida de centralidad, que debería ser el nombre de una columna del dataframe. Su valor es simplemente la suma los valores de dicha columna.

```
auc <- function(df, centrality){
  sm_p <- sum(df[,centrality],na.rm = TRUE)
return(sm_p)}
```

El objetivo de la siguiente función es encontrar el "intervalo" máximo de OTUs cuyos valores por medida de centralidad sumen menos que cierta cota. Sus argumentos son un dataframe, que en nuestro caso estará ordenado por una medida de centralidad, el nombre de dicha medida de centralidad, y la cota deseada. Su valor es la longitud del intervalo buscado.

```
auc_percent <- function(df, centrality, bound){
  #df es nuestro dataframe de muestras y medidas de centralidad, centrality un string con el nombre de

  i <- 1
  sum_par <- 0
  #c <- c()
  while(i <= dim(df)[1] && sum_par < bound) {
    #c <- c(c,i)
    g_i = df[i, centrality]
    sum_par = sum_par + g_i
    i = i+1
  }
  return(i-1)
}
```

En la siguiente celda se incluye una función para calcular el estadístico pseudo-F que se utiliza en los análisis de permanova. Su objetivo es comparar la similaridad de nuestros datos (en este caso muestras) dentro de unos grupos dados (en este caso etapas fenológicos) con respecto a la similaridad entre todas las muestras. Su valor será mayor si las disimilaridades entre cualquier par de muestras es en promedio mayores a las disimilaridades entre las muestras provenientes del mismo grupo. En este caso la disimilaridad entre las muestras es la de Bray-Curtis. Los argumentos de esta función son un dataframe de muestras y una lista de los grupos. Su valor es el estadístico.

```
pseudo_F <- function(df , groups){
  #df es dataframe de muestras, groups, una lista de vectores de etiquetas por grupo , por grupos
  N <- dim(df)[2] #número de muestras
  a <- length(groups) #número de grupos

  df_groups <- list() #se crea la lista que incluirá los subdataframes por grupo

  for (i in 1:length(groups)){
    df_i <- df[,groups[[i]]]
    df_groups[[i]] <- df_i
  }

  n_s <- llply( .data = df_groups , .fun = ncol )
  n_s <- unlist(n_s)
  n <- mean(n_s) #número promedio de muestras por grupo

  dist <- vegdist(t(df))
  dist <- as.vector(dist) #distancias bray_curtis entre todo par de muestras


  df_groups <- llply(.data = df_groups , .fun = t)
```

```
  in_dist <- llply(.data = df_groups , .fun = vegdist )
  in_dist <- llply(.data = in_dist , .fun = as.vector )
  in_dist <- unlist(in_dist) #distancias bray-curtis intra-grupo

  #calculo del estadistico pseudo-f

  ss_t <- sum(dist^2)/N

  ss_w <- sum(in_dist^2)/n

  ss_a <- ss_t - ss_w

  f_stat <- (ss_a/(a-1))/(ss_w/(N-a))

  return(f_stat)
}
```

#Intervalos de n*33.33% de área bajo la curva y varianza correspondiente

En las siguientes celdas de código se calculan finalmente los intervalos de OTUs correspondientes a n*33.33% (con n entre 1 y 3) del área bajo la curva por medida de centralidad. En cada caso se guardan en un vector las longitudes de dichos intervalos.

```
area_deg <- auc(data_deg , "degrees")
auc5_percent_deg <- c()
for (x in 1:3){
  auc5_percent_deg = c(auc5_percent_deg , auc_percent(data_deg, "degrees" ,(area_deg/3)*x))

}
```

```
area_close <- auc(data_close , "closeness")
auc5_percent_close <- c()
for (x in 1:3){
  auc5_percent_close = c(auc5_percent_close , auc_percent(data_close, "closeness" ,(area_close/3)*x))

}
```

```
area_between <- auc(data_between , "betweenness")
auc5_percent_between <- c()
for (x in 1:3){
  auc5_percent_between = c(auc5_percent_between , auc_percent(data_between, "betweenness" ,(area_betwee

}
```

En las próximas celdas finalmente se computa el estadístico pseudo-F con cada una de las medidas y cada uno de los dataframes restringidos a los intervalos. Se calcula también la varianza acumulada de dicho estadístico conforme los intervalos crecen. Se guarda finalmente un archivo .csv con las longitudes de los intervalos, y los estadísticos pseudo-F con las varianzas correspondientes. Los OTUs del intervalo con un máximo local del estadístico que tenga baja varianza serán considerados OTUs clave.

```
f_stat_deg <- c()
var_deg <- c()
for (i in auc5_percent_deg){
```

```
  df_i <- data_deg[1:i ,1:5]
  f_i <- pseudo_F(df_i , grupos)
  f_stat_deg <- c(f_stat_deg , f_i)

  var_i <- var(f_stat_deg)
  var_deg <- c(var_deg, var_i)
}

analisis_auc_deg <- data.frame(auc5_percent_deg, f_stat_deg , var_deg)
analisis_auc_deg
```

```
##   auc5_percent_deg  f_stat_deg    var_deg
## 1                2  0.06771791         NA
## 2                3  0.47692407 0.08372484
## 3                5 -0.12596812 0.09474049
```

```
#write.csv(analisis_auc_deg, "./redes_correlacion_coocurrencia/results/tomate_analisis_auc_bydegrees.cs
```

```
f_stat_close <- c()
var_close <- c()
for (i in auc5_percent_close){
  df_i <- data_close[1:i ,1:5]
  f_i <- pseudo_F(df_i , grupos)
  f_stat_close <- c(f_stat_close , f_i)

  var_i <- var(f_stat_close)
  var_close <- c(var_close, var_i)
}


analisis_auc_close <- data.frame(auc5_percent_close, f_stat_close , var_close)
analisis_auc_close
```

```
##   auc5_percent_close f_stat_close  var_close
## 1                  2   0.06771791         NA
## 2                  3   0.47692407 0.08372484
## 3                  5  -0.12596812 0.09474049
```

```
#write.csv(analisis_auc_close, "./redes_correlacion_coocurrencia/results/tomate_analisis_auc_byclosenes
```

```
f_stat_between <- c()
var_between <- c()
for (i in auc5_percent_between){
  df_i <- data_between[1:i ,1:5]
  f_i <- pseudo_F(df_i , grupos)
  f_stat_between <- c(f_stat_between , f_i)

  var_i <- var(f_stat_between)
  var_between <- c(var_between ,var_i)
}
```

```
analisis_auc_between <- data.frame(auc5_percent_between, f_stat_between , var_between)
analisis_auc_between
```

```
##   auc5_percent_between f_stat_between var_between
## 1                    1     2.76778926          NA
## 2                    2     0.06771791    3.645193
## 3                    3     0.47692407    2.117650
```

```
#write.csv(analisis_auc_between, "./redes_correlacion_coocurrencia/results/tomate_analisis_auc_bybetwee
```

Las últimas celdas nos proveen de un segundo reporte de OTUs clave consistente de los OTUs que por medida de centralidad maximicen el valor del esatdístico pseudo-F y minimicen la varianza acumulada. En esta versión del script esto se logró maximizando la resta de el valor del estadístico menos la varianza. En las siguientes tres celdas se muestra este proceso para cada medida de centralidad. En la variable n_auc_. se guarda la cantidad de OTUs de centralidad mayor que serán considerados para el segundo reporte.

```
analisis_auc_deg <- analisis_auc_deg[2:dim(analisis_auc_deg)[1],]
n_auc_deg <- which((analisis_auc_deg[,2] - analisis_auc_deg[,3]) == max(analisis_auc_deg[,2] - analisis
n_auc_deg <- n_auc_deg + 1
n_auc_deg <- auc5_percent_deg[n_auc_deg]
```

```
analisis_auc_close <- analisis_auc_close[2:dim(analisis_auc_close)[1],]
n_auc_close <- which((analisis_auc_close[,2] - analisis_auc_close[,3]) == max(analisis_auc_close[,2] - a
n_auc_close <- n_auc_close + 1
n_auc_close <- auc5_percent_close[n_auc_close]
```

```
analisis_auc_between <- analisis_auc_between[2:dim(analisis_auc_between)[1],]
n_auc_between <- which((analisis_auc_between[,2] - analisis_auc_between[,3]) == max(analisis_auc_betwee
n_auc_between <- n_auc_between + 1
n_auc_between <- auc5_percent_between[n_auc_between]
```

En esta última celda se guardan los nombres de los OTUs elegidos en las celdas elegidas, se unen y se usan para filtrar los OTUs clave en el dataframe data.

```
results_2_deg <- row.names(data_deg[1:n_auc_deg,])
results_2_close <- row.names(data_close[1:n_auc_close,])
results_2_between <- row.names(data_between[1:n_auc_between,])

results_2 <- union(results_2_deg , results_2_close)
results_2 <- union(results_2 , results_2_between)

data_report_2 <- data[results_2,]
print(data_report_2)
```

```
##               V2        V4        V5        V6        V7 nodos degrees
## 529    0.2803150 0.2628993 0.2593857 0.1832579 0.3011765  v_ 1       3
## 571256 0.1905512 0.1744472 0.1706485 0.1606335 0.1411765  v_ 3       3
## 239106 0.1118110 0.1351351 0.1399317 0.1357466 0.1505882  v_ 4       3
##        closeness betweenness
## 529          0.2           1
## 571256       0.2           3
## 239106       0.2           1
```

```
#write.csv(data_report_2 , paste0("./results/",report_2) , row.names = TRUE)
```