

# **CODA para estudios del microbioma Seminario del proyecto METACIRCLE**

Grupo BIOCOM

6/8/09

# Table of contents

<b>Introducción</b>	<b>3</b>
Objetivos del seminario . . . . .	3
<b>1 Inconvenientes de trabajar con conteos absolutos y conteos normalizados</b>	<b>4</b>
1.1 Inconvenientes de compararar muestras con abundancias relativas . . . . .	5
1.2 Solución . . . . .	9
<b>2 Principios básicos del CODA</b>	<b>11</b>
2.1 Metodología . . . . .	13
<b>3 Reemplazos composicionales del tool kit tradicional</b>	<b>14</b>
<b>4 CODA en acción : Ejemplo 1 (A,E,I)</b>	<b>16</b>
4.1 Carga de librerías . . . . .	16
4.2 Carga y limpieza de los datos . . . . .	16
<b>5 Tratamiento de los ceros</b>	<b>18</b>
5.1 Géneros que aparecen únicamente en un tipo de muestra . . . . .	23
5.2 Imputación de ceros con la previa de Jeffreys . . . . .	25
5.3 Imputación de ceros con Geometric Bayesian multiplicative . . . . .	25
5.4 Imputación de ceros con un método Iterativo . . . . .	26
5.5 ¿Qué método para imputar los ceros es mejor? . . . . .	27
5.6 Comparando matrices de distancias de Aitchison: valor pequeño indica mejor. .	27
5.7 Si se quieren filtrar las muestras outliers . . . . .	28
5.8 Sin filtrar variables . . . . .	30
5.9 Clustering jerárquico . . . . .	31
5.10 ALDEx . . . . .	32

# Introducción

- Los conjuntos de datos recogidos mediante secuenciación de alto rendimiento (HTS) pueden ser analizados a partir de conteos absolutos o normalizados, si el propósito es hacer estadística descriptiva (no siempre coinciden).
- Si el propósito es hacer inferencia estadística, se debe usar CODA (Compositional Data Analysis).

## Objetivos del seminario

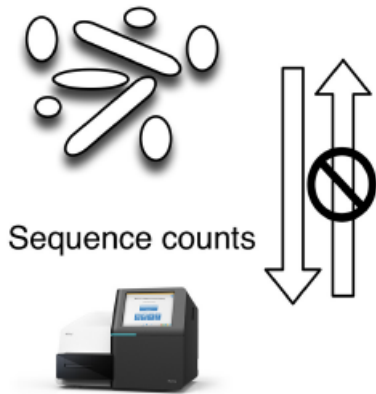
- i.- Discutiremos los **orígenes** de la composicionalidad en los datos del microbioma.*
- ii.- Mostraremos los principales **problemas** de utilizar los métodos estadísticos tradicionales para tratar los datos de secuenciación.*
- iii.- Indicaremos los reemplazos composicionales de las técnicas del **tool kit** de análisis de datos de microbioma.*
- iv.- **CODA en acción**: trabajaremos con un ejemplo de datos de la microbiota humana, veremos el tipo de resultados que podemos obtener.*

# 1 Inconvenientes de trabajar con conteos absolutos y conteos normalizados

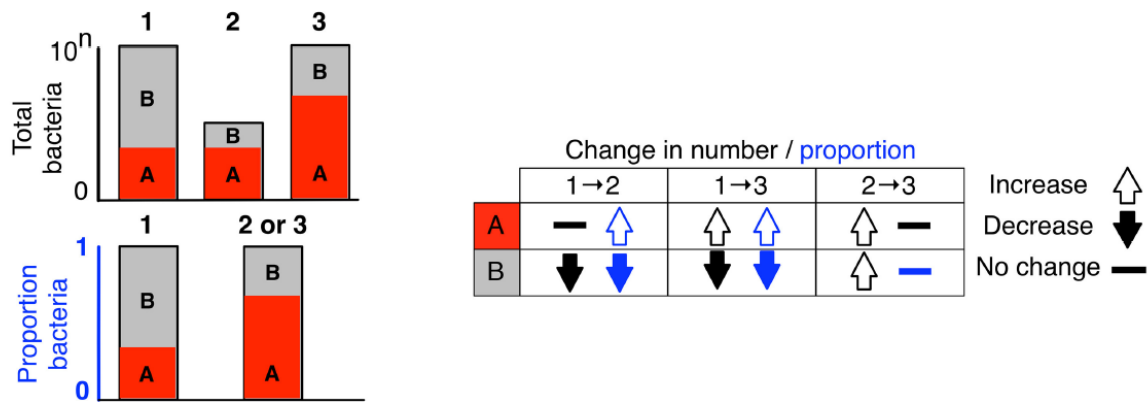
En los datos de secuenciación de alto rendimiento (HTS ):

- El número de lecturas **no** representa la abundancia total o absoluta en el ecosistema.
- Una muestra HTS tiene un tamaño fijo que viene fijado por la capacidad del aparato utilizado.

Bacterial population



- No podemos comparar muestras con conteos absolutos.
- Podemos ver la diferencia entre conteos y proporciones comparando los datos de tres muestras en el gráfico que se presenta a continuación.



- Los diagramas de barras muestran la diferencia entre el conteo de moléculas y la proporción de moléculas para dos características, A y B en tres muestras.
- Las muestras 2 y 3 tienen las mismas abundancias proporcionales, aunque tengan **conteos absolutos diferentes**.
- En la tabla podemos observar que la relación entre la abundancia absoluta y la abundancia relativa cambia de manera significativa .

## 1.1 Inconvenientes de comparar muestras con abundancias relativas

- Si normalizamos, de manera de que las sumas totales de las filas no importa, entonces las matrices de distancias usuales entre muestras, cambian.

### Ejemplo

```
set.seed(5)
library(ecodist)
x= round( rnorm (3 ,100 ,10) )
y= round( rnorm (3 ,100 ,10) )
z= round( rnorm (3 ,1000 ,100) )
X= cbind(x,y,z)
knitr::kable(X) #Conteos absolutos
```

x	y	z
92	101	953
114	117	936

x	y	z
87	94	971

```
knitr::kable(as.matrix(bcdist(X),ncol=2)) # Matriz de distancia de Bray-Curtis conteos abs
```

	1	2	3
0.0000000	0.0237786	0.0130548	
0.0237786	0.0000000	0.0366537	
0.0130548	0.0366537	0.0000000	

```
X1=apply(X, 1, function (x) x/sum(x))
knitr::kable(X1) #Conteos normalizados
```

	x	y	z
x	0.0802792	0.0976864	0.0755208
y	0.0881326	0.1002571	0.0815972
z	0.8315881	0.8020566	0.8428819

```
knitr::kable(as.matrix(bcdist(X1),ncol=2)) # Matriz de distancia de Bray-Curtis conteos no
```

	x	y	z
x	0.0000000	0.0315212	0.8142965
y	0.0315212	0.0000000	0.8033966
z	0.8142965	0.8033966	0.0000000

- Algunas soluciones:
  - Subsampling, pero se pierde precisión.
  - Usar solo proporciones, pero se añaden correlaciones espurias
- Correlación espuria (Pearson 1896).

Dos o más OTUs estarán correlacionados simplemente porque los datos han sido transformados a una suma constante.

$$x = \begin{bmatrix} 790 & 488 & 1174 & 1037 \\ 737 & 470 & 1052 & 1064 \\ 589 & 386 & 1112 & 772 \\ 634 & 344 & 741 & 870 \end{bmatrix}$$



$$\text{cor}(x) = \begin{bmatrix} 1 & 0.89 & 0.43 & 0.94 \\ & 1 & 0.76 & 0.83 \\ & & 1 & 0.28 \\ & & & 1 \end{bmatrix}$$

$$\pi_x = \begin{bmatrix} 0.226 & 0.139 & 0.336 & 0.297 \\ 0.221 & 0.141 & 0.316 & 0.320 \\ 0.206 & 0.135 & 0.388 & 0.270 \\ 0.244 & 0.132 & 0.286 & 0.336 \end{bmatrix}$$



$$\text{cor}(\pi_x) = \begin{bmatrix} 1 & -0.28 & -0.93 & 0.88 \\ & 1 & 0.03 & -0.04 \\ & & 1 & -0.98 \\ & & & 1 \end{bmatrix}$$

- Incoherencia subcomposicional

$$x = \begin{bmatrix} 790 & 488 & 1174 & 1037 \\ 737 & 470 & 1052 & 1064 \\ 589 & 386 & 1112 & 772 \\ 634 & 344 & 741 & 870 \end{bmatrix}, \quad \text{cor}(\pi_x) = \begin{bmatrix} 1 & -0.28 & -0.93 & 0.88 \\ & 1 & 0.03 & -0.04 \\ & & 1 & -0.98 \\ & & & 1 \end{bmatrix}$$

$$y = \begin{bmatrix} 790 & 488 & 1174 \\ 737 & 470 & 1052 \\ 589 & 386 & 1112 \\ 634 & 344 & 741 \end{bmatrix}, \quad \text{cor}(\pi_y) = \begin{bmatrix} 1 & 0.64 & -0.98 \\ & 1 & -0.76 \\ & & 1 \end{bmatrix}$$





- Si hemos normalizado, los conteos de los microorganismos no son variables independientes ya que están ligados por el valor de la suma.

## ARTICLE

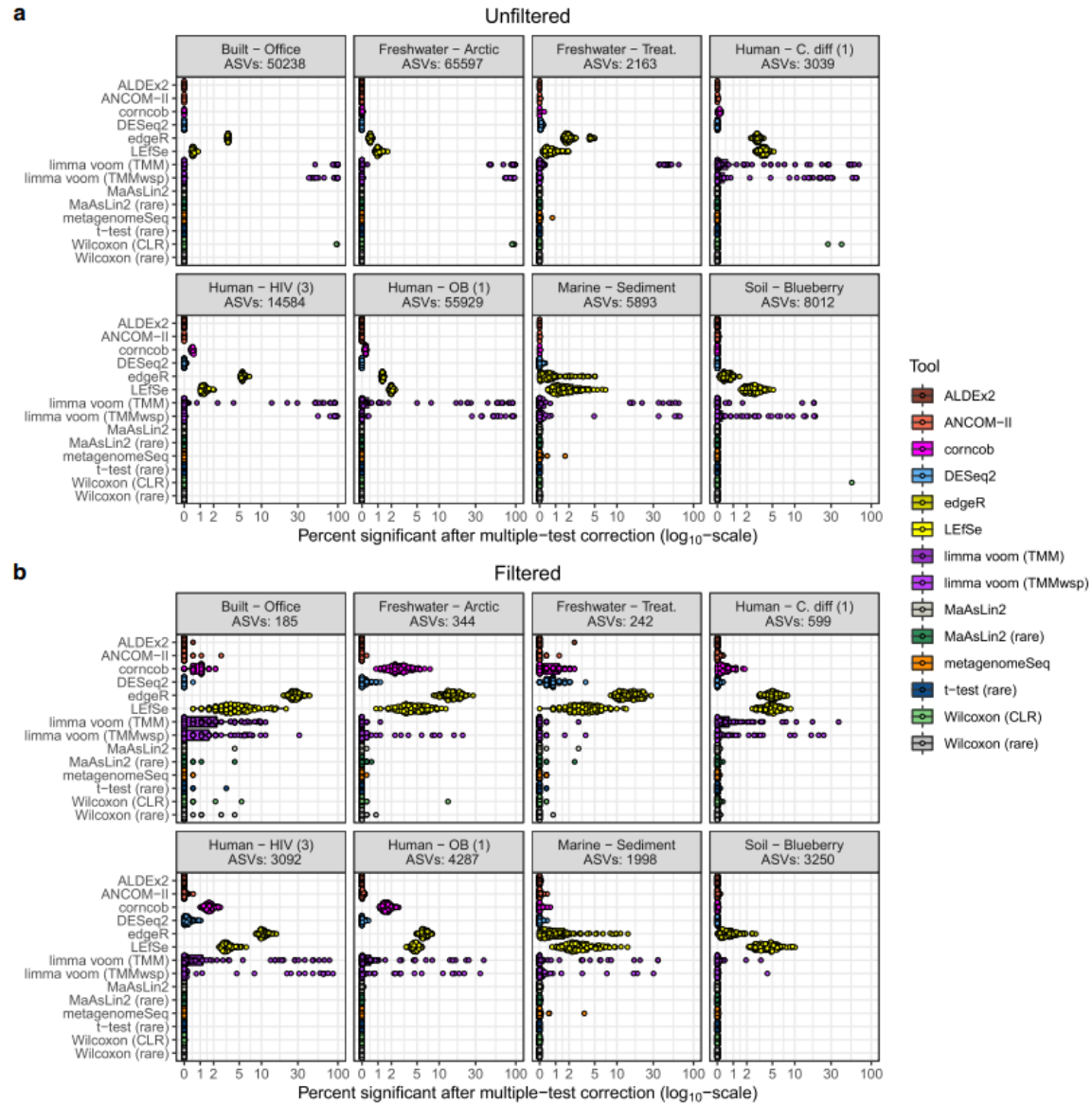
<https://doi.org/10.1038/s41467-022-28034-z>

OPEN

# Microbiome differential abundance methods produce different results across 38 datasets

Jacob T. Nearing <sup>1,7✉</sup>, Gavin M. Douglas<sup>1,7</sup>, Molly G. Hayes <sup>2</sup>, Jocelyn MacDonald<sup>3</sup>, Dhwani K. Desai<sup>4</sup>, Nicole Allward<sup>5</sup>, Casey M. A. Jones<sup>6</sup>, Robyn J. Wright<sup>6</sup>, Akhilesh S. Dhanani <sup>4</sup>, André M. Comeau <sup>4</sup> & Morgan G. I. Langille<sup>4,6</sup>





## 1.2 Solución

Utilizamos otra forma de “normalizar” que preserva la composición de cada muestra y nos permite compararla:

**CODA= Compositional Data Analysis**



# Microbiome Datasets Are Compositional: And This Is Not Optional

Gregory B. Gloor<sup>1\*</sup>, Jean M. Macklaim<sup>1</sup>, Vera Pawlowsky-Glahn<sup>2</sup> and Juan J. Egozcue<sup>3</sup>

<sup>1</sup> Department of Biochemistry, University of Western Ontario, London, ON, Canada, <sup>2</sup> Departments of Computer Science, Applied Mathematics, and Statistics, Universitat de Girona, Girona, Spain, <sup>3</sup> Department of Applied Mathematics, Universitat Politècnica de Catalunya, Barcelona, Spain

## 2 Principios básicos del CODA

- Cada muestra se considera como un vector de datos composicionales  $(x_1, \dots, x_k)$  con cada  $x_i \geq 0$ .
- Consideramos que dos vectores son equivalentes cuando son proporcionales.
- El análisis de los datos se basa en las ratios  $x_i/x_j$ .
- Las ratios se transforman mediante logaritmos para traducir comparación relativa en comparación absoluta (y para acercar las ratios a una normal).

Estas transformaciones producen los mismos resultados si los conteos son absolutos o proporciones.

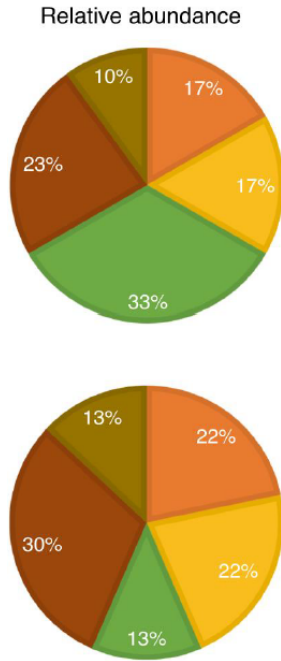
### *Example: log-ratio scale invariance*

$$(100, 200, 400, 300), \quad \log(X_3/X_2) = \log(400/200) = \log(2)$$

$$(2000, 4000, 8000, 6000), \quad \log(X_3/X_2) = \log(8000/4000) = \log(2)$$

$$(0.1, 0.2, 0.4, 0.3), \quad \log(X_3/X_2) = \log(0.4/0.2) = \log(2)$$

*Exploring all possible log-ratios we can infer which are the real differences*



$$\log(0.17/0.17) = \log(0.22/0.22) = 0$$

$$\log(0.23/0.10) = \log(0.30/0.13) = \log(2.3)$$

$$\log(0.23/0.17) = \log(0.30/0.22) = \log(1.36)$$

$$\log(0.23/\mathbf{0.33}) \neq \log(0.30/\mathbf{0.13})$$

$$\log(0.17/\mathbf{0.33}) \neq \log(0.22/\mathbf{0.13})$$

$$\log(\mathbf{0.33}/0.10) \neq \log(\mathbf{0.13}/0.13)$$

Los cocientes logarítmicos son números reales, una gran ventaja para la aplicación de los métodos estadísticos estándar desarrollados para variables aleatorias continuas.

A menudo se utiliza la transformación log-ratio centrada (clr) introducida Aitchison (1986).

Dado un vector  $\mathbf{x}$  de conteos de OTUs,  $\mathbf{x} = (x_1, \dots, x_D)$ , su transformación **clr** es:

$$\mathbf{x}_{clr} = (\log(x_1/g(\mathbf{x})), \dots, \log(x_D/g(\mathbf{x}))),$$

donde  $g(\mathbf{x})$  es la media geométrica de  $\mathbf{x}$ , es decir,  $g(\mathbf{x}) = \sqrt[D]{x_1 \cdot x_2 \cdots x_D}$ .

#### Tratamiento de los ceros:

- Si alguna variable es cero, no se pueden calcular las ratios.
- Imputación de ceros: **zCompositions** de R, como una distribución de probabilidad utilizando **ALDEx2** disponible en Bioconductor.

## 2.1 Metodología

Ante una tabla  $X$  de datos composicionales

- ① Eliminamos filas y columnas constantes 0, y aplicamos cualquier otro filtrado que encontremos conveniente
- ② Sustituimos (**imputo**) los NA y los ceros
- ③ Transformamos logarítmicamente los datos
- ④ Analizamos con técnicas de CODA
  - ACP y biplots adecuados
  - Clustering con distancias adecuadas
  - ANOVA  $\rightarrow$  ALDEx
  - Correlaciones adecuadas

### 3 Reemplazos composicionales del tool kit tradicional

Operation	Standard approach	Compositional approach
Normalization	Rarefaction 'DESeq'	Centered Log Ratio (CLR) Isometric Log Ratio (ILR) Additive Log Ratio (ALR)
Distance	Bray-Curtis UniFrac Jenson-Shanon	Aitchison
Correlation	Pearson Spearman	SparCC SpiecEasi
Differential abundance	metagenomSeq LEfSe DESeq	ALDEx2 ANCOM

#### Distancia de Aitchison

- La distancia de Aitchison no es más que la distancia euclidiana entre muestras después de transformación clr.

### Aitchison distance

$$\downarrow$$
$$d(x_i, x_j) = \left( \sum_{k=1}^D \left( \log\left(\frac{x_{ik}}{g(\mathbf{x}_i)}\right) - \log\left(\frac{x_{jk}}{g(\mathbf{x}_j)}\right) \right)^2 \right)^{\frac{1}{2}} \quad \text{Aitchison, 1986}$$

***Transform data ( $x \leftrightarrow y$ ):***

### Euclidean distance

$$\downarrow$$
$$d(y_i, y_j) = \left( \sum_{k=1}^N (y_{ik} - y_{jk})^2 \right)^{\frac{1}{2}}$$

### Correlaciones de datos composicionales

Existen varias técnicas para analizar la correlación de los datos del microbioma que suelen ser matrices “sparse”. Uno de ellos es **r-sparc**.

## 4 CODA en acción : Ejemplo 1 (A,E,I)

### 4.1 Carga de librerías

```
set.seed(1)
library(data.table)
library(compositions)
library(zCompositions)
library(ALDEx2)
library(kableExtra)
library(ggplot2)
library(easyCODA)
library(RColorBrewer)
library(robCompositions)
library(dendextend)
library(coda4microbiome)
library(propr)
library(ppclust)
library(factoextra)
library(cluster)
library(fclust)
library(nnet)
library(corrplot)
source("funcionsCODACesc.R")
```

### 4.2 Carga y limpieza de los datos

```
DF.0=read.table("count_table_otus.tsv",header=TRUE,sep="\t")
rownames(DF.0)=DF.0[,1]
DF.0=DF.0[,-1]
# Eliminando los _
rownames(DF.0)=gsub("_",".", rownames(DF.0))
```



```
colnames(DF.0)=gsub("_",".", colnames(DF.0))

# Las filas deben ser muestras y las columnas taxa
DF.0=t(DF.0)
dim(DF.0)
```

```
[1] 218 280
```

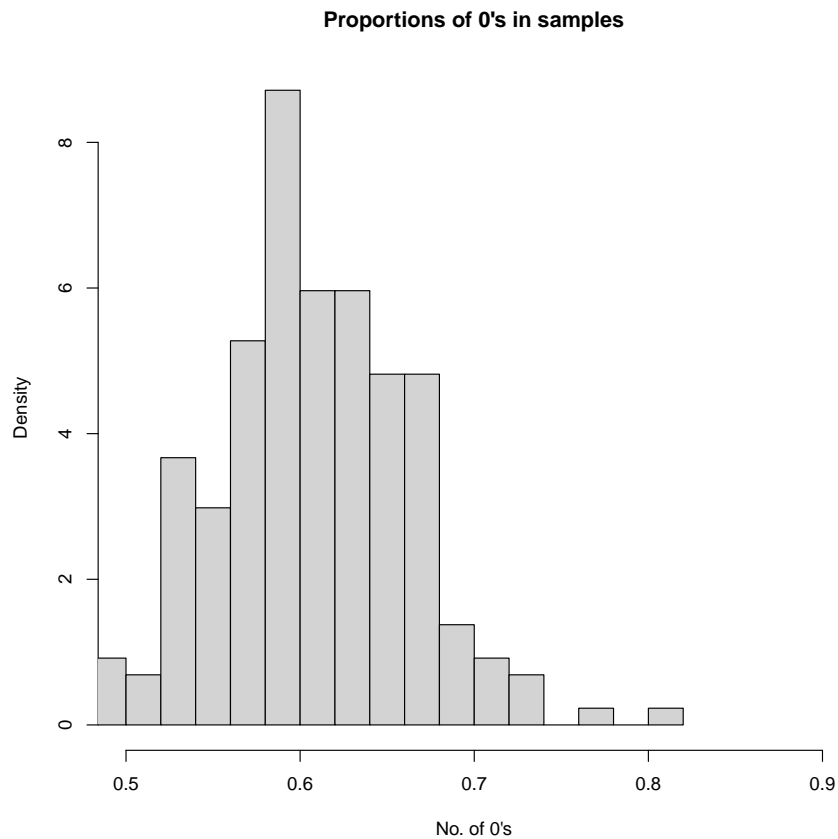
```
# [1] 218 280
# Eliminamos las filas y columnas con 0
DF.0=DF.0[apply(DF.0, 1, sum)>0,apply(DF.0, 2, sum)>0]
mostres=rownames(DF.0)
bitxos=colnames(DF.0)
colnames(DF.0)=1:dim(DF.0)[2]
Grups=as.factor(substr(mostres,1,1))
colors=c("green","blue","brown")[Grups]
```

## 5 Tratamiento de los ceros

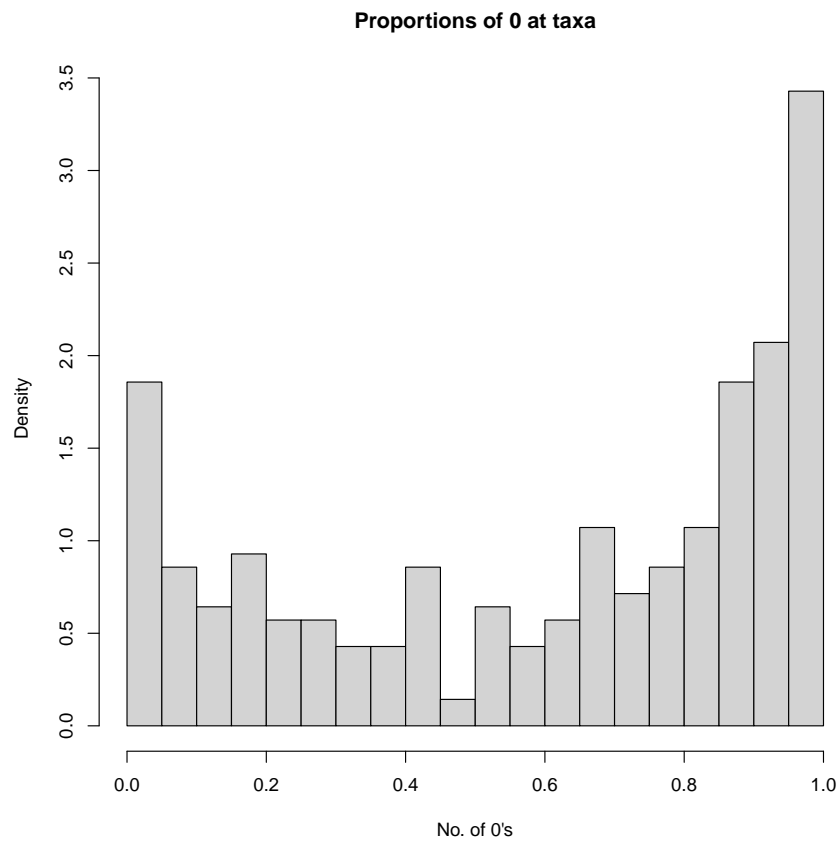
```
#Proporciones de ceros por muestras
Zeros.row=apply(DF.0,MARGIN=1,FUN=function(x){length(x[x==0])/length(x)})

#Proporciones de ceros por taxa
Zeros.col=apply(DF.0,MARGIN=2,FUN=function(x){length(x[x==0])/length(x)})

hist(Zeros.row,breaks=20,freq=FALSE,xlim=c(0.5,0.9),xlab="No. of 0's", main="Proportions of 0's in samples")
```



```
hist(Zeros.col,breaks=20,freq=FALSE,xlab="No. of 0's", main="Proportions of 0 at taxa")
```



```
# zPatterns de la librería zCompositions  
zPatterns(DF.0,label=0,suppress.print=TRUE,main="Global")
```

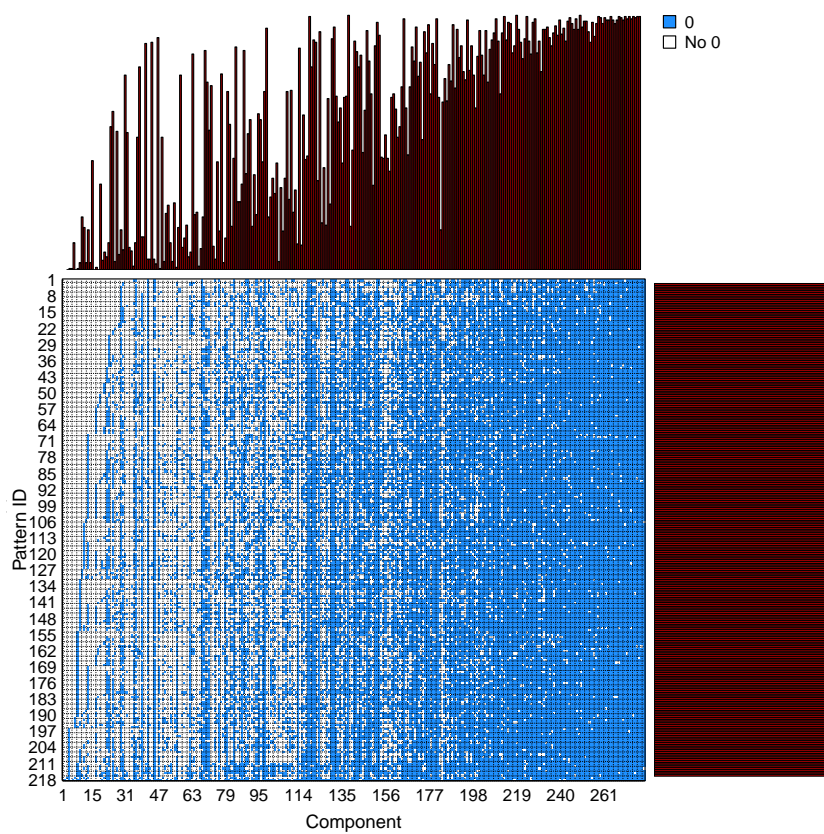


Figure 5.1: Zero Patterns in global sample

```
zPatterns(DF.0[Grups=="A",],label=0,suppress.print=TRUE,main="Global")
```

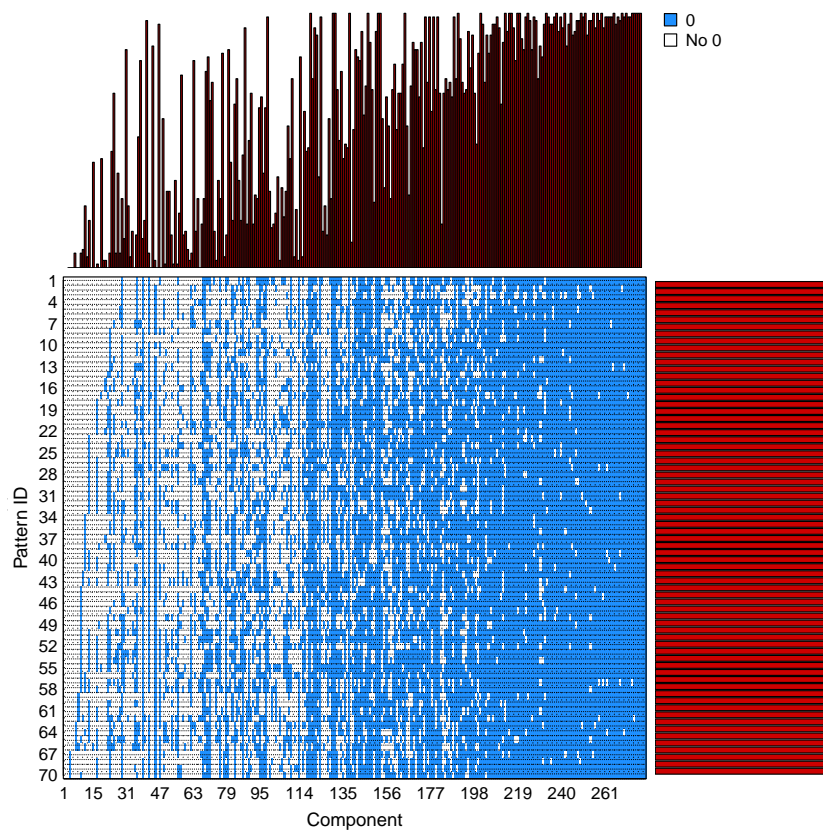


Figure 5.2: Zero Patterns in Adults samples

```
zPatterns(DF.0[Grups=="E",],label=0,suppress.print=TRUE,main="Global")
```

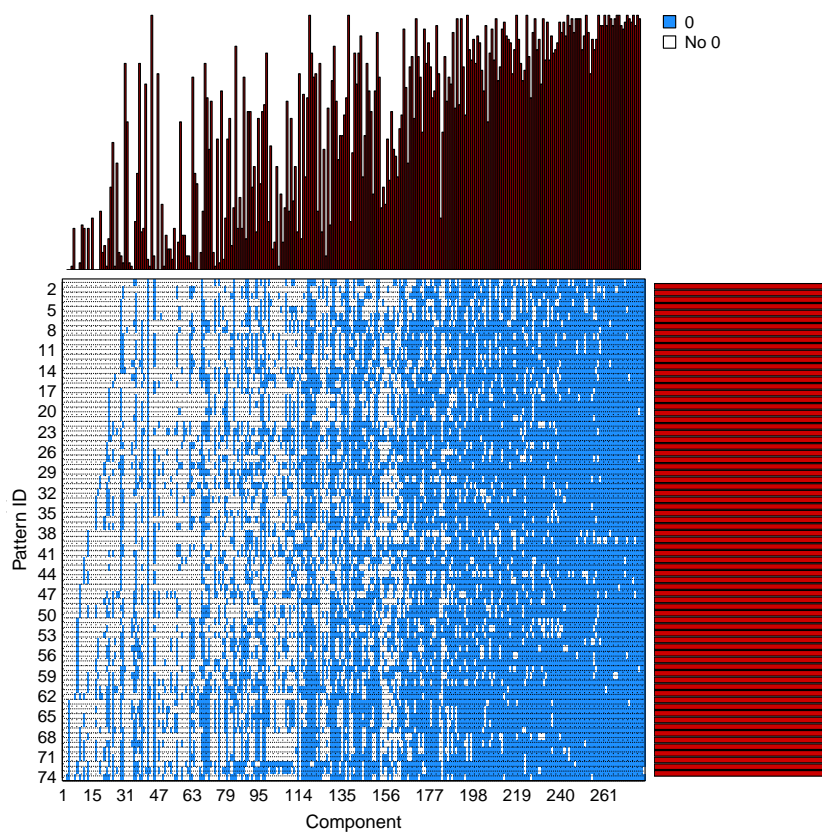


Figure 5.3: Zero Patterns in Ederly samples

```
zPatterns(DF.0[Grups=="I",],label=0,suppress.print=TRUE,main="Global")
```

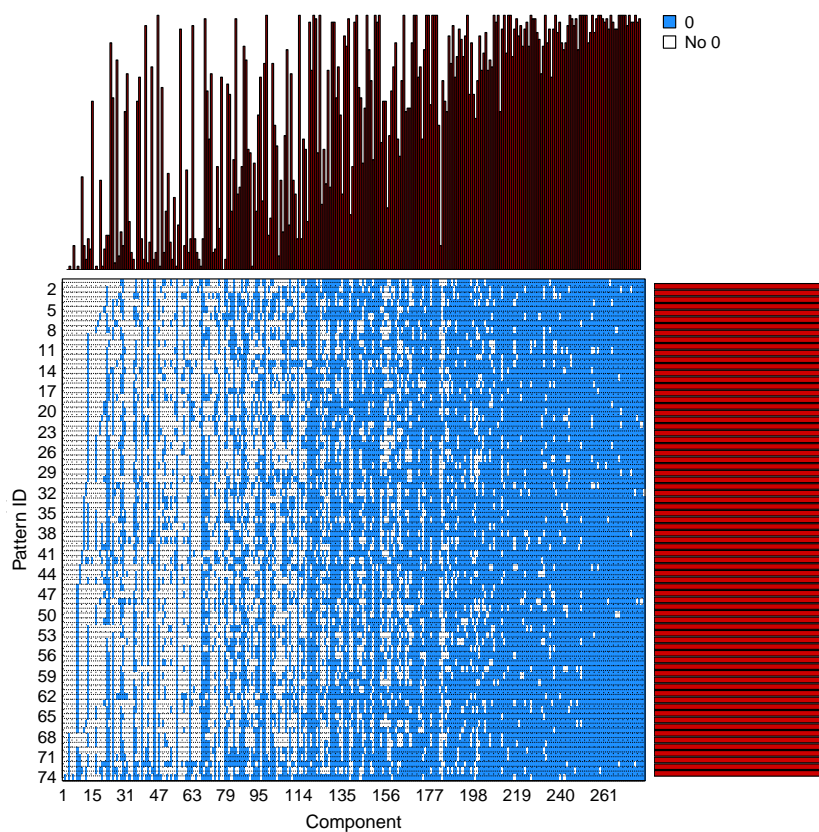


Figure 5.4: Zero Patterns in Infants samples

## 5.1 Géneros que aparecen únicamente en un tipo de muestra

- Solo en Adultos

```
bitxos.nomesA=bitxos[which(apply(DF.0[Grups!="A",], 2, sum)==0)]
length(bitxos.nomesA)
```

[1] 4

```
bitxos.nomesA%>%
  kbl() %>%
  kable_styling()
```

x
OTU.196
OTU.254
OTU.260
OTU.270

x
OTU.147
OTU.153
OTU.165
OTU.214
OTU.216
OTU.217
OTU.234
OTU.239
OTU.246
OTU.251
OTU.253
OTU.257
OTU.261
OTU.271
OTU.273
OTU.278

- Solo en los Ancianos (Elderly)

```
bitxos.nomesE=bitxos[which(apply(DF.0[Grups!="E",], 2, sum)==0)]
length(bitxos.nomesE)
```

[1] 16

```
bitxos.nomesE%>%
  kbl() %>%
  kable_styling()
```

- Solo en los Infantes

```
bitxos.nomésI=bitxos[which(apply(DF.0[Grups!="I",], 2, sum)==0)]
length(bitxos.nomésI)
```



x
OTU.119
OTU.138
OTU.220
OTU.225
OTU.277
OTU.279

[1] 6

```
bitxos.nomésI%>%
  kbl() %>%
  kable_styling()
```

- Solo en uno

```
Nomes.a.un=which(apply(DF.0[Grups!="A",], 2, sum)==0 | apply(DF.0[Grups!="E",], 2, sum)==0)
```

## 5.2 Imputación de ceros con la previa de Jeffreys

```
#cmultRepl de la librería zCompositions Bayesian-Multiplicative replacement of count zeros
# previa de Jeffreys 1/2, todos los valores tienen la misma probabilidad de ocurrir.

DF.J=cmultRepl(DF.0, method="user", t=matrix(1/dim(DF.0)[2],nrow=dim(DF.0)[1],ncol=dim(DF.0)[2],
output="p-counts",suppress.print=TRUE)
```

## 5.3 Imputación de ceros con Geometric Bayesian multiplicative

```
# Este es el método por defecto de cmultRepl
# Hay que quitar columnas con solo una entrada diferente a 0
Unics=which(apply(DF.0, 2, function(x){length(which(x>0))})==1)
bitxos.unics=bitxos[Unics]
length(bitxos.unics)
```

[1] 3

OTU.138
OTU.220
OTU.251

```
bitxos.unics%>%
  kbl(col.names =NULL) %>%
  kable_styling()
```

Damos un vistazo a lo que nos perderemos si las quitamos: Sus frecuencias relativas dentro de sus muestras únicas

```
Què.ens.perdem=rep(0,length(Unics))
for (i in 1:length(Unics)){
  y=attr(Unics,"names")[i]
  x=which(DF.0[,y]>0)
  Què.ens.perdem[i]=DF.0[x,y]/sum(DF.0[x,])
}
round(Què.ens.perdem,6)
```

```
[1] 0.024063 0.001049 0.000288
```

La matriz con los ceros imputados ...

```
DF.0U=DF.0[,-Unics]
DF.GBM=cmultRepl(DF.0U,method="GBM",output="p-counts",suppress.print=TRUE)
bitxos.nounics=bitxos[-Unics]
```

## 5.4 Imputación de ceros con un método Iterativo

```
# imprZilr de la librería robCompositions
#
# Tarda mucho en compilar
# DF.0n=as.data.frame(DF.0)
# DF.0n=as.data.frame(apply(DF.0n,MARGIN=2,as.numeric))
# DF.It=imprZilr(DF.0n, eps=0.05, method = "pls", dl=rep(1, dim(DF.0)[2]),maxit = 10,verbo
# saveRDS(DF.It, file="DFItnou.RData")
```

```
DF.It=readRDS("DFItnou.RData")$x
```

## 5.5 ¿Qué método para imputar los ceros es mejor?

(Lubbe-Filznoser-Templ Chemolab 2021)

Comparando matrices de correlaciones de Kynclova-Hron-Filzmoser: un valor pequeño indica que el método es mejor.

La función `corCoDa` del paquete `robCompositions` que las calcula no aguanta matrices grandes (al menos en el ordenador de Cesc), por lo tanto lo ha hecho por muestreo.

```
# Sustitución de los ceros por algo muy pequeño
# multRepl de la librería zCompositions
DF.0.alt=multRepl(DF.0,d1=rep(1, ncol(DF.0)),frac=10^(-12),label=0)
DF.0.alt.U=DF.0.alt[, -Unics]

# X Y con las mismas dimensiones
# m < 30 o da NaN
f=function(X,Y,m){
  x=sample(dim(X)[2],m)
  (1/m)^2*sum((corCoDa(X[,x])-corCoDa(Y[,x]))^2)
}
mean(replicate(200,f(DF.0.alt,DF.J,25)))
mean(replicate(200,f(DF.0.alt.U,DF.GBM,25)))
mean(replicate(200,f(DF.0.alt,DF.It,25)))
# [1] 0.01688433 0.01664001
# [1] 0.009403911 0.009399878
# [1] 0.06253841 0.05720421
```

## 5.6 Comparando matrices de distancias de Aitchison: valor pequeño indica mejor.

```
(1/dim(DF.0)[1])^2*sum((aDist(DF.0.alt)-aDist(DF.J))^2)
```

```
[1] 20279.85
```

```
(1/dim(DF.0U)[1])^2*sum((aDist(DF.0.alt.U)-aDist(DF.GBM))^2)
```

```
[1] 19972.44
```

```
 #(1/dim(DF.0)[1])^2*sum((aDist(DF.0.alt)-aDist(DF.It))^2)
```

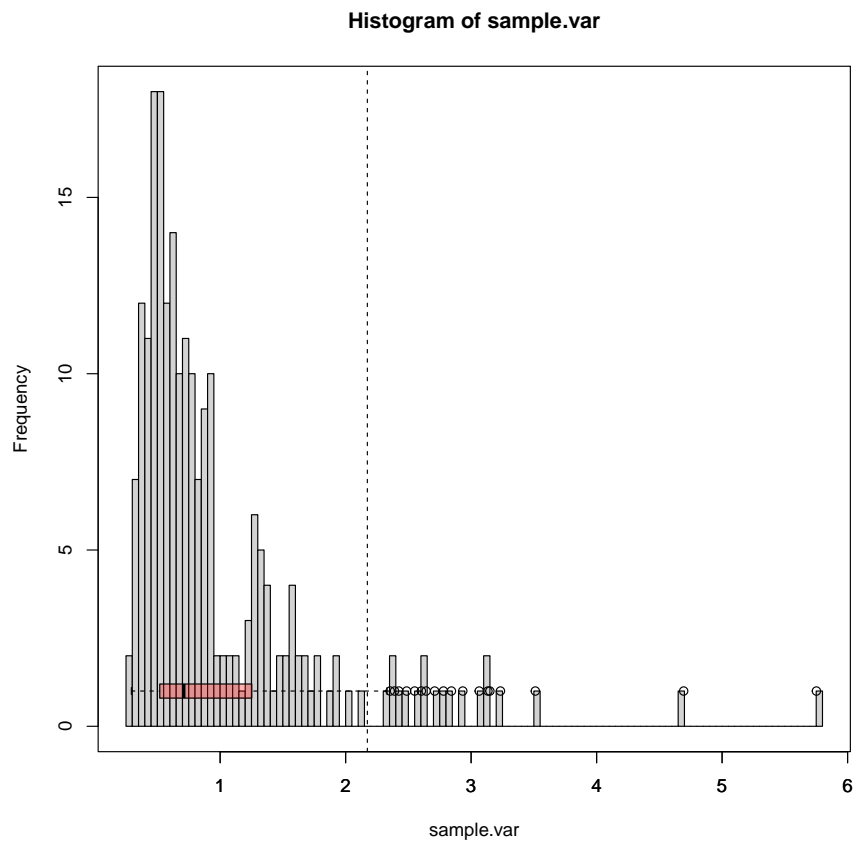
```
Imputa0=c("J","GBM","It")
Imputa0="GBM"
if (Imputa0=="J"){
  DF=DF.J
}
if (Imputa0=="GBM"){
  DF.0=DF.0U
  DF=DF.GBM
  bitxos=bitxos.nounics
}
if (Imputa0=="It"){
  DF=DF.It
}
```

## 5.7 Si se quieren filtrar las muestras outliers

Quitamos muestras que contribuyen a tener mucha varianza

Utilizó codaSeq.outlier de funcionesCODACesc.R adaptada de EasyCODA, que le da error

```
DF.CLR=acomp(DF)
CSOut=codaSeq.outlier(DF.CLR, plot.me=TRUE)
```



```
outliers=CSOut$bad
bones=CSOut$good
```

Las muestras outliers son

```
mostres[outliers]
```

```
[1] "A04T4" "A05T1" "A07T2" "A08T4" "A08T6" "E01T1" "E01T2" "E06T4" "E09T2"
[10] "E10T3" "I01T2" "I01T4" "I02T7" "I05T2" "I07T4" "I07T6" "I07T7" "I09T2"
[19] "I10T6"
```

Si quitamos estas muestras, tenemos que volver a controlar que no nos quede ninguna columna de 0s

```

DF.OB=DF.0[bones,]
conserv=which(apply(DF.OB, 2, sum)>0)
DF.OB=DF.OB[,conserv]

# si se emplea DF.It, igual conviene re-calcularlo porque depende de las muestras

# Hay que quitar bichos que hayan quedado a 0 en todo
conserv=which(apply(DF.OB, 2, sum)>0)
DF.OB=DF.OB[,conserv]
DF.OBn=as.data.frame(DF.OB)
DF.OBn=as.data.frame(apply(DF.OBn,MARGIN=2,as.numeric))
DFB.It=impRZilr(DF.OBn, eps=0.05, method = "pls", dl=rep(1, dim(DF.OB)[2]),maxit = 10,verb
saveRDS(DFB.It, file="DFItBnou.RData")

```

Con la función QuinesMostres indicamos si cogemos solo las muestras buenas o todas

```

#QuinesMostres=c("totes","bones")
QuinesMostres="bones"
if (QuinesMostres=="bones"){
  DF.0=DF.OB
  DF=DF[bones,conserv]
  Grups=Grups[bones]
  colors=colors[bones]
  mostres=mostres[bones]
  bitxos=bitxos[conserv]
}
DF.CLR=acomp(DF)
DF.prop=t(apply(DF, 1, function(x){x/sum(x)}))

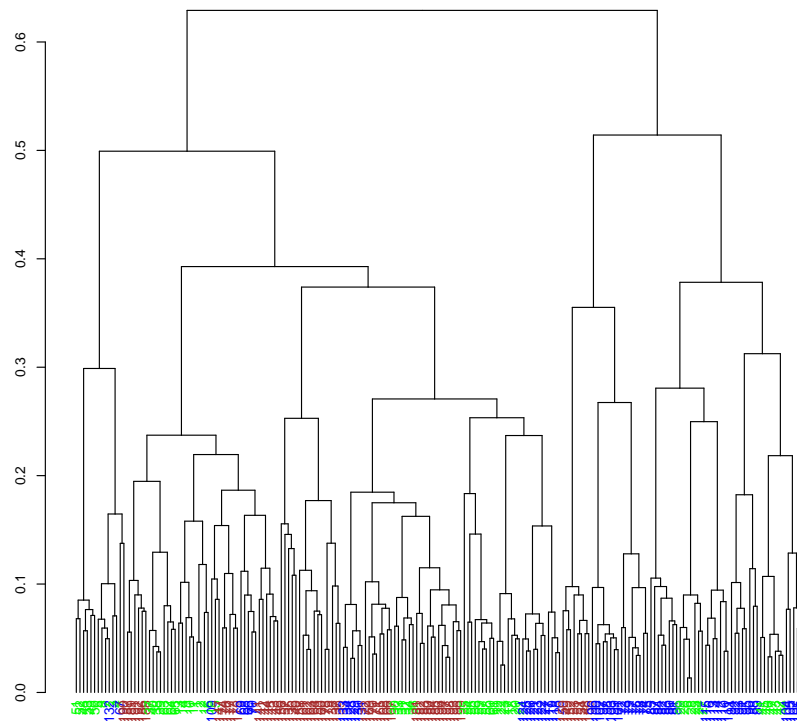
```

## 5.8 Sin filtrar variables

No hace biplot porque con tanta variable no se ve nada

## 5.9 Clustering jerárquico

```
# Clustering jerárquico con distancias euclidianas con pesos
# Función WARD de EasyCODA, necesita que la matriz de clr's se calcule con la misma librería
DF.CLR.W=CLR(DF)
hc=WARD(DF.CLR.W,weight=TRUE)
dend=as.dendrogram(hc)
labels_colors(dend)=colors[hc$order]
par(cex=0.75)
plot(dend, main = "")
```



```
par(cex=1)
# No dibujamos barplot de composiciones porque el gráfico no es informativo
```

	1	2	3
A	51	14	0
E	25	28	16
I	57	0	8

```

clust1=data.frame(Orig=Grups[hc$order],
                  clust=cutree(dend, k = 3)[order.dendrogram(dend)])
table(clust1)%>%
  kbl() %>%
  kable_styling()

```

## 5.10 ALDEx

ALDEx2 estima la variación por edad (A, E, I) dentro de cada muestra utilizando el método de Monte Carlo, las muestras se extraen de la distribución Dirichlet (la beta en el caso multivariado). El muestreo a partir de esta distribución devuelve la distribución de probabilidad posterior de los datos observados bajo un modelo de muestreo repetido. Todas las salidas de ALDEx2 son salidas de las distribuciones posteriores, ya sean valores esperados o intervalos de confianza.

```

DF0.Aldex=rbind(DF.0[Grups=="A",],DF.0[Grups=="E",], DF.0[Grups=="I",])
DF0.t=data.frame(t(DF0.Aldex))
conds=c(rep("A", dim(DF.0[Grups=="A",])[1]),rep("E", dim(DF.0[Grups=="E",])[1]), rep("I",
#
x.clr.kw=aldex.clr(DF0.t[,1:5], conds=conds[1:5], mc.samples=10, verbose=FALSE)
mc.instances <- numMCInstances(x.clr.kw)
mc.all <- getMonteCarloInstances(x.clr.kw)

```

```

DF0.Aldex=rbind(DF.0[Grups=="A",],DF.0[Grups=="E",], DF.0[Grups=="I",])
DF0.t=data.frame(t(DF0.Aldex))
conds=c(rep("A", dim(DF.0[Grups=="A",])[1]),rep("E", dim(DF.0[Grups=="E",])[1]), rep("I",
#
x.clr.kw=aldex.clr(DF0.t, conds=conds, mc.samples=1000, verbose=FALSE)
x.kw=aldex.kw(x.clr.kw, verbose=FALSE)
# valores esperados del test Kruskal-Wallis y un glm sobre los datos
mm=model.matrix(~conds,data.frame(conds))
x.clr.glm=aldex.clr(DF0.t, conds=mm, mc.samples=1000, verbose=FALSE)
x.glm=aldex.glm(x.clr.glm, mm)

```



bitxos	kw.ep	kw.eBH	glm.ep	glm.eBH	Intercept::Est	Intercept::SE	Intercept::t.val
OTU.00 1	0.0801547	0.1570136	0.0303225	0.0797589	11.554637	0.1854587	62.30798
OTU.00 2	0.0206440	0.0579031	0.8077564	0.8585892	10.371632	0.3250578	31.95125
OTU.00 3	0.0000000	0.0000001	0.0000000	0.0000001	11.002260	0.2578001	42.77776
OTU.00 4	0.0005343	0.0031584	0.0136801	0.0424850	8.736565	0.5371141	16.30443
OTU.00 5	0.0017423	0.0078298	0.0006925	0.0041580	12.369761	0.1919890	64.43325
OTU.00 6	0.0008415	0.0044561	0.0350900	0.0835593	9.693963	0.2458796	39.53140

```
#
x.tot=cbind(bitxos,x.kw,x.glm)
saveRDS(x.tot, file="xtotTotal.RData")

x.tot=readRDS("xtotTotal.RData")
x.tot$bitxos=paste0("OTU.",c(paste("00",1:9),paste("0",10:99),100:276))

x.tot=readRDS("xtotTotal.RData")

head(x.tot)%>%
  kbl() %>%
  kable_styling()
```

Aquí podemos basarnos en kw.eBH, glm.eBH, model.condsE,  $\Pr(>|t|)$ .BH, model.condsI  $\Pr(>|t|)$ .BH

```
signif.p1=which(x.tot$kw.eBH < 0.01)
signif.g1=which(x.tot$glm.eBH < 0.01)
signif.p1=intersect(signif.p1,signif.g1)

length(signif.p1)
```

[1] 40

```
x.tot.sign=x.tot[signif.p1,c(3,5,19,20)]
names(x.tot.sign)=c("p-val KW corregit", "p-val glm corregit", "p-val E vs A corregit", "p-
rownames(x.tot.sign)=bitxos[signif.p1]

x.tot.sign%>%
  kbl() %>%
```

```
kable_styling()
```

```
prova=unlist(sapply(rownames(x.tot.sign),FUN=function(x) which(x==x.tot$bitxos)))  
names(prova)=NULL
```

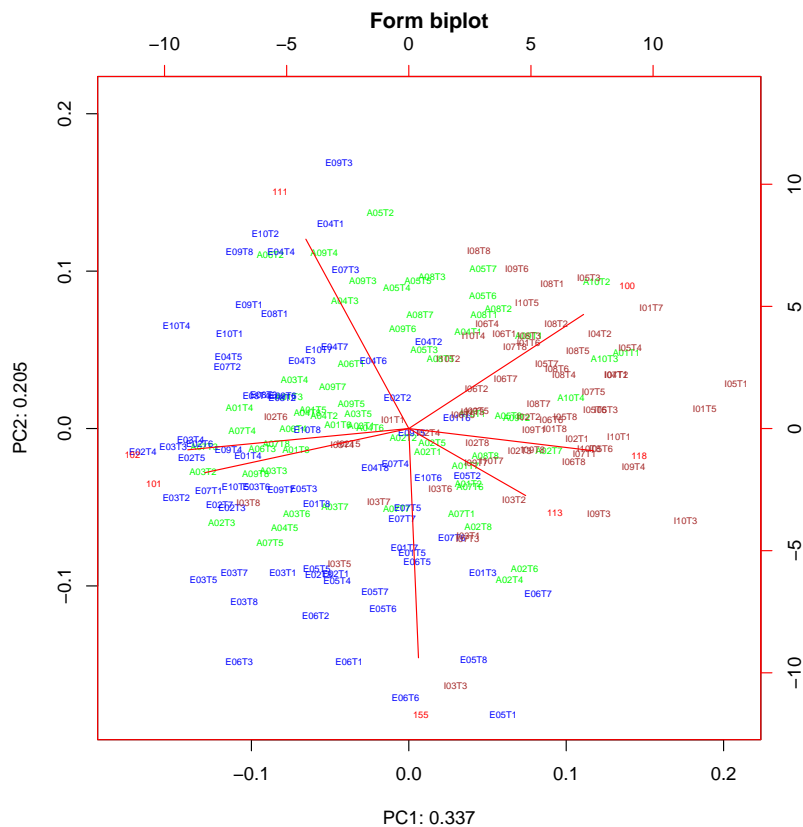
Si nos restringimos a estos bichos:

```
DF.s=DF[,prova]  
DF.CLR=acomp(DF.s)
```

```
pcx=princomp(DF.CLR,cor=TRUE)
```

```
coloredBiplot(pcx, cex=0.5, col="red", arrow.len=0, scale=1,var.axes=TRUE,  
  xlab=paste("PC1", round(pcx$sdev[1]^2 / sum(pcx$sdev^2),3), sep=": "),  
  ylab=paste("PC2", round(pcx$sdev[2]^2 / sum(pcx$sdev^2),3), sep=": "),  
  xlab.col=colors, main="Form biplot")
```

	p-val KW corregit	p-val glm corregit	p-val E vs A corregit	p-val I vs A corregit
OTU.003	0.0000001	0.0000001	0.0972514	0.8239804
OTU.005	0.0078298	0.0041580	0.2511886	1.0000000
OTU.008	0.0000070	0.0000016	1.0000000	0.0000085
OTU.011	0.0010125	0.0059162	1.0000000	0.0787355
OTU.013	0.0000433	0.0000009	0.2828991	0.5834326
OTU.016	0.0038849	0.0006962	0.0327502	1.0000000
OTU.020	0.0097656	0.0033890	1.0000000	0.0918489
OTU.022	0.0000326	0.0000097	1.0000000	0.0358061
OTU.025	0.0002163	0.0001919	1.0000000	0.0091624
OTU.030	0.0000002	0.0000000	0.0002360	0.0000002
OTU.034	0.0000000	0.0000000	1.0000000	0.0000011
OTU.038	0.0001000	0.0006898	1.0000000	0.0438187
OTU.044	0.0000015	0.0000002	1.0000000	0.0000073
OTU.046	0.0011419	0.0006191	0.5639068	0.0063995
OTU.047	0.0001250	0.0001823	0.2585800	0.9991464
OTU.049	0.0053303	0.0072328	0.1026868	0.9789205
OTU.050	0.0000264	0.0002277	0.3148470	1.0000000
OTU.052	0.0001237	0.0075423	0.9952454	1.0000000
OTU.054	0.0013229	0.0097693	1.0000000	0.7449599
OTU.057	0.0000489	0.0000655	1.0000000	0.1825672
OTU.059	0.0000000	0.0000000	1.0000000	0.0000004
OTU.060	0.0002872	0.0042465	0.6414395	1.0000000
OTU.063	0.0004161	0.0014406	0.4197127	1.0000000
OTU.064	0.0000174	0.0001876	1.0000000	0.0845730
OTU.066	0.0000025	0.0000036	0.0061348	0.0000254
OTU.067	0.0003092	0.0026454	0.7734519	0.0414771
OTU.075	0.0091920	0.0052941	0.1359825	1.0000000
OTU.077	0.0000837	0.0003245	0.9095772	0.9990152
OTU.078	0.0001277	0.0005558	0.7500225	1.0000000
OTU.086	0.0005582	0.0022955	0.1265078	1.0000000
OTU.088	0.0000000	0.0000000	1.0000000	0.0000000
OTU.091	0.0000012	0.0000026	0.6333928	0.4631056
OTU.096	0.0007613	0.0024963	0.7773785	0.9987909
OTU.100	0.0087198	0.0049639	0.4523354	1.0000000
OTU.101	0.0000000	0.0000000	1.0000000	0.0000138
OTU.102	0.0000000	0.0000053	0.9922000	0.0525332
OTU.111	0.0004223	0.0016563	0.3737696	0.0432119
OTU.113	0.0021340	0.0040426	0.1122496	1.0000000
OTU.118	0.0000004	0.0000050	0.8946424	0.0804560
OTU.154	0.0010999	0.0060971	0.1554282	0.9981464



```
DF.CLR.W.s=CLR(DF.s)
hc2=WARD(DF.CLR.W.s,weight=TRUE)
dend.2=as.dendrogram(hc2)
labels_colors(dend.2)=colors[hc2$order]
DF0r=DF.s[hc2$order,]
#Reordemanos las muestras para dibujar los barplot en el mismo orden
DF0r.CLR=acom(DF0r)
d.names=colnames(DF.s)[order(apply(DF.s, 2, sum), decreasing=T) ]
nb.cols=dim(DF.s)[2]
colors.OTU=colorRampPalette(brewer.pal(length(d.names),"Spectral"))(nb.cols)
#Dibujo
layout(matrix(c(1,3,2,3),2,2, byrow=T), widths=c(6,2), height=c(4,4))
par(mar=c(2,1,1,1)+0.1,cex=0.75)
plot(dend.2, main = "")
barplot(DF0r.CLR, legend.text=F, col=colors.OTU, axisnames=F, border=NA, xpd=T,)
```

```

par(mar=c(0,1,1,1)+0.1,cex=1)
plot(1,2, pch = 1, lty = 1, ylim=c(-20,20), type = "n", axes = FALSE, ann = FALSE)
legend(x="center", legend=d.names, col=colors.OTU, lwd=5, cex=.6, border=NULL)

```

