Francielli Vilela Peres

Bióloga e doutoranda no programa de Microbiologia do Instituto de Ciências Biomédicas - ICB-USP. O foco da minha pesquisa consiste na descrição da diversidade e estrutura da comunidade microbiana em áreas de pockmarks e diápiros de sal na margem continental brasileira.

Currículo Lattes:http://lattes.cnpq.br/3313110333561429

Meus Exercícios

Linque para a página com os meus exercícios resolvidos Execícios

Proposta A: Core Microbiome

Contextualização

Em estudos de microbiologia, comumente utilizamos o termo Unidade Taxonômica Operacional (OTU) para nos referirmos individualmente a um microrganismo, a um grupo taxonômico ou a um grupo cujas características evolutivas não foram bem estabelecidas, porém, possuem um conjunto de características compartilhadas. Este termo é propositalmente vago, cabendo ao cientista especificar em sua pesquisa ao que se refere. Tipicamente, o agrupamento de OTUs é definido por um limiar de 97% de similaridade entre as sequências do gene ribossomal 16S. Através da análise de OTUs, podemos determinar o core microbiome, que consiste no conjunto estável de membros compartilhados entre diferentes grupos de amostras, ambientes ou tratamentos. A identificação desses grupos centrais é importante para o esclarecimento da ecologia dos consórcios microbianos em diferentes ambientes. Essa análise é baseada em um conjunto de dados onde presença e ausência de OTUs são computadas. O resultado desta análise, usualmente, é apresentado em um Venn Diagram, sendo as áreas sobrepostas do gráfico as OTUs compartilhadas entre os diferentes grupos analisados.

Planejamento da função

Entrada: O arquivo de entrada consiste de um data frame (otu.table), com o ID de cada OTU e seus valores de abundância por amostra e/ou tratamento. Vantagem: Este input consiste em uma tabela bem conhecida e muito utilizada pelos pesquisadores de Ecologia Microbiana.

- O arquivo de entrada não precisará de alterações. A função se encarregará de transforma-la em um arquivo binário.
- Valores iguais a zero, serão considerados ausência e valores iguais a 1 serão considerados presença. Só será contabilizada a OTU que estiver presente em 100% das amostras comparadas. Exemplo: O core microbiome entre o Controle e o Tratamento1 será 4 (OTU1;OTU2;OTU3;OTU5). Já o core microbiome total será 2 (OTU1 e OTU2).

- Parece fácil, não é mesmo!? Agora pense em um data frame com dezenas de tratamentos e milhares de OTUs

Saída

- Data frame com a quantidade de OTUs compartilhadas entre as categorias comparadas.
- Venn Diagram representando os valores de OTUs compartilhadas entre as categorias comparadas (core microbiome).
- Exemplo de saída:

Referências SHADE, A.; HANDELSMAN, J. Beyond the Venn diagram: the hunt for a core microbiome. Environmental Microbiology, v.14, n. 1, p. 4-12, 2012.

ZAURA, E.; KEIJSER, B. J. F.; HUSE, S. M.; CRIELAARD, W. Defining the healthy “core microbiome” of oral microbial communities. BMC Microbiology, p. 1-12, 2009.

NAM-PHUONG NGUYEN, N.; WARNOW, T.; MIHAI POP, M.; WHITE, B. A perspective on 16S rRNA operational taxonomic unit clustering using sequence similarity. Biofilms and Microbiomes, 2016.

Comentários Felipe Ernesto

Oi Francielli!

Gostei bastante da sua proposta! Parece bem útil pra área e acho que está dentro do esperado em relação à complexidade do código. Gostei bastante da contextualização também, as imagens me ajudaram bastante. A única coisa que faltou foi o pseudo-código.

Só tenho um comentário a fazer. Suponho que a ideia é receber entradas com um número arbitrário de tratamentos. Isso é bem legal, mas pode dar algum trabalho. Isso pq o número de combinações de tratamentos cresce bem rápido e vc vai ter que generalizar. Por exemplo, com 4 tratamentos, temos as seguintes combinações: os 4 tratamentos sozinho; depois as 6 combinações dois a dois; depois as 4 combinações três a três; e depois todos os tratamentos juntos; totalizando 15 combinações. Não sei direito como escrever fórmulas aqui, mas vou escrever em código o número de combinações de tratamentos, usando a função choose pra representar o coeficiente binomial:

choose(4,1) + choose(4,2) + choose(4,3) + choose(4,4)

De forma geral, pra N tratamentos:

choose(N,1) + choose(N,2) + choose(N,3) + … + choose(N,N)

Note que essa soma além de ser o número de combinações tb é o número de regiões do diagrama de venn.

Enfim, vc vai precisar fazer um monte de combinações e a princípio generalizar isso seria mais chatinho. Porém tem uma função salvadora do R, não sei ao certo se vc já tinha pensado em usá-la. De resto vc vai ter que fazer uns loops encaixados pra conseguir varrer todas essas combinações. Pode dar algum trabalho, mas nada impossível.

Acho que meu comentário foi mais pq sem o pseudo-código eu não tenho como saber se vc tinha

pensado nisso e quis avisar aonde vai pegar.

Qualquer coisa posta aqui que eu respondo.

Abs

(PS: eu tinha feito uma confusão com os termos, mas já corrigi, não sei se vc chegou a ver rs)

Resposta Francielli

Oi Felipe!

Muito obrigada pelos seus comentários. Foi justamente sobre essa generalização que fiquei na dúvida e acabei não postando o pseudo-código por receio de deixar-lo confuso. Ainda preciso descobrir essa "função salvadora", para não ir pelo caminho mais longo rs.

Então, a princípio seguirei com a proposta A. Obrigada novamente :)

Abs

Proposta B: Organização de tarefas

Contextualização

Manter a organização e as tarefas em dia no ambiente de trabalho é fundamental para o bom funcionamento e desenvolvimento do grupo. É certo que em locais maiores que comportam um grande número de pessoas, manter a organização nem sempre é fácil. Pensando nisso, trago minha segunda proposta de função. Como exemplo, trago um laboratório composto de alunos de graduação, pós-graduação e técnicos que são responsáveis por diferentes tarefas que precisam ser realizadas diariamente, semanalmente e quinzenalmente. As tarefas são distribuídas aleatoriamente, porém, seguindo um critério de seleção: alunos de graduação são responsáveis por determinadas tarefas (peso1), alunos de pós-graduação por outras (peso2) e determinadas tarefas só devem ser realizadas por técnicos (peso3). A ideia desta função surgiu através da necessidade de organização no laboratório, cujas tarefas são redistribuídas mensalmente. O exemplo contextualizado é um laboratório, porém, o arquivo de entrada pode ser adaptado para qualquer ambiente que possua pessoas e tarefas que precisem ser distribuídas aleatoriamente.

Planejamento da Função

Entrada: O arquivo de entrada será um data frame com o nome dos componentes do grupo e as tarefas a serem sorteadas, separadas por período (tarefas diárias, semanais e quinzenais).

Saída: O arquivo de saída será um data frame com o nome dos alunos/técnicos e suas respectivas tarefas. Exemplo de arquivo de saída:

Comentários Felipe Ernesto

Oi Francielli!

Achei a ideia legal e com potencial. Porém faltou vc explicar melhor como seriam divididas as tarefas. O pseudo-código teria ajudado também. Da forma como vc apresentou não consigo saber se a ideia é dividir tarefas de forma a todo mundo trabalhar mais ou menos o mesmo tanto nem se pesoas de peso 2 ou 3 poderiam fazer trabalhos de peso 1 caso houvessem muitos trabalhos e poucos alunos de peso 1. Também não consigo saber como vc faria essas coisas.

Enfim, parece uma ideia legal, mas precisa ser mais detalhada pra eu poder comentar. De toda forma acho que vc não terá problemas com o plano A.

Qualquer coisa só postar aqui!

Abs

Resposta Francielli

Oi Felipe!

Obrigada pelos comentários. A intenção seria que os alunos de peso1 ficassem apenas com tarefas de peso1, e alunos de peso2 apenas com tarefas de peso2 e técnicos (peso3) ficassem apenas com tarefas de peso3, independente da quantidade de tarefas, seria essa a divisão.

A pricípio, seguirei com a proposta A então :)

Abs

Código da Função

```
##########################################################################
############
# Função coremicrobiome() -Análise do Core Microbime para estudos de
Ecologia Microbiana#
##########################################################################
############
coremicrobiome = function(otu_table, numbercateg=NULL, namecateg=NULL)
#O arquivo otu_table poderá conter, ou não, informações taxônomicas.
#O usuário deverá indicar o número de categorias
(p.ex.:numbercateg=4)(limitado a quatro) e o identificador único #das
categorias análisadas (p.ex.:namecateg=c("Cnt", "Lag", "Oce", "Esg").
{
  check=rep(NA, ncol(otu_table))#Crie repetições de NAs de acordo com o
número de colunas do seu data.frame.
  for(i in 1:ncol(otu_table))#Guarde o número de colunas dentro de i.
  {
    check[i]=is.numeric(otu_table[,i])#Verifica se os valores das linhas do
```

```
arquivo otu_table é númerico. Numérico: TRUE. Não numérico: FALSE.
  }
  if (sum(check)!=ncol(otu_table))#Se a soma dos elementos de check não for
igual (!=) ao número de colunas do seu
                                  #data.frame, significa que seu arquivo
possui uma coluna com informações
                                  #taxonomicas, para proseguirmos, usaremos
um data.frame sem esta coluna.
  {
    otu=otu_table[,1:ncol(otu_table)-1]#O objeto otu guardará as informações
do seu data.frame, exceto a última
                                       #coluna.
  } else                               #Caso a condição if não for
verdadeira, ou seja, caso seu data.frame não contenha informações
taxonomicas, seguiremos com o objeto "otu".
  {
    otu=otu_table                      #Objeto "otu" contendo as informações
do seu data.frame sem informações
                                       #taxonomicas.
  }
  check2=rep(NA, ncol(otu))#Crie repetições de NAs de acordo com o número de
colunas do seu data.frame.
  for(i in 1:ncol(otu))#Guarde o número de colunas dentro de i.
  {
    check2[i]=is.numeric(otu[,1])#Verifica se os valores das linhas do
objeto otu é númerico. Numérico: TRUE. Não
                                  #numérico: FALSE.
  }
  if (sum(check2)==ncol(otu))#Se a soma (sum) dos elementos de check2 for
igual (==) ao número de colunas do
                             #data.frame.
  {
    otu2=otu                 #otu2 permanecerá igual ao objeto otu.
  } else                     #Caso a condição if não for verdadeira o
comando irá parar (stop) e enviará a mensagem
                             #informando ao usuário que o arquivo deve
seguir o formato classico do Qiime para uma
                             #Otu Table.
  {
    stop("Input data must be in Qiime Classic Otu Table format (with or
without taxonomy annotations) ===>
https://www.drive5.com/usearch/manual/qiime_classic.html")
  }
  if("plyr" %in% rownames(installed.packages()) == FALSE)#Se o pacote "plyr"
não estiver instalado...
                                                         #plyr é um pacote
do R que permite a fragmentação e a
                                                         #junção de
diferentes porções dos seus dados.
  {
    stop("The 'plyr' package is required. Please, install it before
```

```
continuing")#Se o pacote "plyr" não estiver
#instalado, o comando irá parar
#(stop) e enviará uma mensagem
#pedindo para o usuário instalar o
#pacote para continuar.
  } else
  {
    library("plyr")#Se o pacote já estiver instalado, ele será "chamado"
pelo comando library("plyr").
  }
  if("VennDiagram" %in% rownames(installed.packages()) == FALSE)#Se o pacote
"VennDiagram" não estiver instalado...
                                                                #O pacote
"VennDiagram" é utilizado para criação de
                                                                #diagramas
do tipo Venn.
  {
    stop("The 'VennDiagram' package is required. Please, install it before
continuing")#Se o pacote "VennDiagram"
#não estive instalado, o
#comando irá parar (stop) e
#enviará uma mensagem
#pedindo para o usuário
#instalar o pacote para
#continuar.
  } else
  {
    suppressPackageStartupMessages(library("VennDiagram"))#Se o pacote já
estiver instalado, ele será "chamado"
                                                          #pelo comando
library("VennDiagram), além de suprir as
                                                          #mensagens de
iniciação.
  }
  bin=as.data.frame((otu2>0)*1)#Com o comando otu2>0 convertemos o objeto
otu2 para um objeto da classe logical,
                               #onde valores diferentes de zero se tornam
TRUE e todos os iguais a zero se tornam
                               #FALSE. Em vetores da classe logical, TRUE
corresponde a 1 e FALSE corresponde a 0
                               #em operações matemáticas. Sendo assim, se
multiplicarmos todo o objeto lógico por 1
                               #teremos um data.frame binário.
                               #as.data.frame força o arquivo a ser da
classe data.frame.
  if(is.null(numbercateg))#Pegará todas as colunas do arquivo de entrada e
avaliará o core microbiome de todas
                          #elas.
  {
    cor.01=subset(bin, apply(bin, 1, sum)==ncol(bin))#apply retorna um
```

```
vetor, matriz ou uma lista obtidos aplicando
                                                        #uma função aos dados.
Neste caso, bin é o data.frame, 1
                                                        #indica que a operação
ocorrerá em linhas e a função será
                                                        #soma.
                                                        #cor.01: a soma das
linhas deve ser igual ao número de
                                                        #colunas.
    cor=list("Quantitative"=nrow(cor.01),
"Qualitative"=rownames(cor.01))#list: criação de uma lista.
#Fará uma lista com as informações de
#número de linhas (nrow) e o nome de cada
#linha (rownames) (OTU_ID) do objeto
#cor.01 que atenderam a exigência
#anterior.
    draw.single.venn(nrow(cor.01), category=deparse(substitute(x)),
lty="blank", fill="green", alpha=0.5, cex=2.5, cat.cex=2.5)#draw.single.venn
cria um diagrama de Venn com um único conjunto.
            #lty: Define o padrão de traço da circunferência do diagrama.
            #fill: Define a cor de preenchimento.
            #alpha: Transparência do círculo.
            #cex: Define o tamanho a área.
            #cat.cex: Define o tamanho do nome da categoria.
  } else if(numbercateg==1)#Se o número de categoria for igual a 1.
  {
    stopifnot(length(namecateg)==numbercateg)#Verifica os argumentos da
função. No caso se o tamanho dos
                                            #identificadores únicos são
iguais ao número de categoria.
    cor.01=subset(bin, select=grepl(namecateg, names(bin)))#O objeto cor.01,
baseado no objeto bin, guardará as
                                                          #informações da
sua categoria (no caso 1).
    cor.02=subset(cor.01, apply(cor.01, 1, sum)==ncol(cor.01))#cor.02: a
soma das linhas do objeto cor.01 deve ser
                                                             #igual ao
número de colunas do mesmo objeto.
    cor=list("Quantitative"=nrow(cor.02),
"Qualitative"=rownames(cor.02))#Fará uma lista com as informações de
#número de linhas (nrow) e o nome de cada
#linha (rownames) (OTU_ID) do objeto
#cor.02. Serão as OTUs compartilhadas
#entre todas as suas amostras da
#categoria (core microbiome).
    draw.single.venn(nrow(cor.02), category=namecateg, lty="blank",
fill="green", alpha=0.5, cex=2, cat.cex=2)#draw.single.venn cria um diagrama
de Venn com um único conjunto.
          #A quantidade de OTUs pertencentes ao seu core microbiome.
  } else if(numbercateg==2)#Se o número de categorias for igual a 2.
  {
```

```
    stopifnot(length(namecateg)==numbercateg)#Verifica os argumentos da
função. No caso, o tamanho dos
                                             #identificadores únicos devem
ser iguais ao número de categorias.
    cor.01a=subset(bin, select=grepl(namecateg[1], names(bin)))#grepl
utiliza expressões regulares para combinar
                                                                #padrões,
neste caso, combinará apenas os padrões
                                                                #encontrados
no identificador único 1 do objeto bin.
                                                                #O subset
retornará a condição exigida.
                                                                #O objeto
cor.01a, baseado no objeto bin, guardará
                                                                #apenas as
informações do primeiro identificador
                                                                #único.
    cor.02a=subset(cor.01a, apply(cor.01a, 1, sum)==ncol(cor.01a))#A soma
das linhas do objeto cor.01a deve ser
                                                                   #igual ao
número de colunas do mesmo objeto.
                                                                   #O objeto
cor.02a guardará a informação do core
#microbiome da primeira categoria.
    cor.01b=subset(bin, select=grepl(namecateg[2], names(bin)))#O objeto
cor.01b, baseado no objeto bin, guardará
                                                                #apenas as
informações do segundo identificador
                                                                #único.
    cor.02b=subset(cor.01b, apply(cor.01b, 1, sum)==ncol(cor.01b))#A soma
das linhas do objeto cor.01b deve ser
                                                                   #igual ao
número de colunas do mesmo objeto.
                                                                   #O objeto
cor.02b guardará a informação do core
#microbiome da segunda categoria.
    cor.03=Reduce(intersect, list(rownames(cor.02a), rownames(cor.02b)))#O
objeto cor.03 será o valor de
#sobreposição dos objetos cor.02a e
#cor.02b, criando uma lista com o nome das
#linhas que são comuns entre os dois
#objetos (OTU_ID).
    cor.04=c(nrow(cor.02a), nrow(cor.02b), length(cor.03))#Concatena o
número de linhas dos objetos cor.02a,
                                                        #cor.02b e o
número de observações do cor.03.
    names(cor.04)=c(namecateg[1], namecateg[2], paste0(namecateg[1],
namecateg[2]))#names faz referência a um
#objeto pelo nome, no caso,
#cor.04.
```

```
#paste tem por objetivo
#concatenar uma sequências de
#caracteres.
    cor.05=list(rownames(cor.02a), rownames(cor.02b), cor.03)#Criação de uma
lista.
                                                                #rownames: nome
de cada linha.
                                                                #cor.05
guardará em uma lista o nome das linhas
                                                                #(OTU_ID) dos
objetos cor.02a e cor.02b
                                                                #e o valor de
observação do cor.03 (sobreposição de
                                                                #cor.02a e
cor.02b).
    names(cor.05)=names(cor.04)#Obter ou definir os nomes dos objeto.
    cor=list("Quantitative"=cor.04, "Qualitative"=cor.05)#A informação é
guardada na forma de lista.
    draw.pairwise.venn(nrow(cor.02a), nrow(cor.02b), length(cor.03),
category=c(namecateg[1], namecateg[2]), lty="blank", fill=c("green",
"deepskyblue"), alpha=0.5, cex=2, cat.cex=2)
    #draw.pairwise.vem cria um diagrama de Venn com dois conjuntos, no caso,
com o número de linhas do objeto
    cor.02a e cor.02b e o valor de sobreposição entre esses dois objetos.
   } else if(numbercateg==3)#Se o número de categorias for igual a 3.
  {
    stopifnot(length(namecateg)==numbercateg)#Verifica os argumentos da
função. No caso, o tamanho dos
                                            #identificadores únicos devem
ser iguais ao número de tratamento.
    cor.01a=subset(bin, select=grepl(namecateg[1], names(bin)))#O objeto
cor.01a, baseado no objeto bin, guardará
                                                                  #apenas as
informações do primeiro identificador
                                                                  #único.
    cor.02a=subset(cor.01a, apply(cor.01a, 1, sum)==ncol(cor.01a))#A soma
das linhas do objeto cor.01a deve ser
                                                                     #igual ao
número de colunas do mesmo objeto.
                                                                     #O objeto
cor.02a guardará a informação do core
#microbiome da primeira categoria.
    cor.01b=subset(bin, select=grepl(namecateg[2], names(bin)))#O objeto
cor.01b, baseado no objeto bin, guardará
                                                                  #apenas as
informações do primeiro identificador
                                                                  #único.
    cor.02b=subset(cor.01b, apply(cor.01b, 1, sum)==ncol(cor.01b))#A soma
das linhas do objeto cor.01b deve ser
                                                                     #igual ao
número de colunas do mesmo objeto.
```

```
                                                                    #O objeto
cor.02b guardará a informação do core
#microbiome da segunda categoria.
    cor.01c=subset(bin, select=grepl(namecateg[3], names(bin)))#O objeto
cor.01c, baseado no objeto bin, guardará
                                                                  #apenas as
informações do terceiro identificador
                                                                  #único.
    cor.02c=subset(cor.01c, apply(cor.01c, 1, sum)==ncol(cor.01c))#A soma
das linhas do objeto cor.01c deve ser
                                                                     #igual ao
número de colunas do mesmo objeto.
                                                                    #O objeto
cor.02c guardará a informação do core
#microbiome da terceira categoria.
    cor.03=Reduce(intersect, list(rownames(cor.02a), rownames(cor.02b),
rownames(cor.02c)))
    #O ojbeto cor.03 será o valor de sobreposição dos objetos cor.02a,
cor.02b e
    #cor.02c, criando uma lista com o nome das linhas que são comuns entre
os
    #três objetos (OTU_ID).
    sobrep02ab=Reduce(intersect, list(rownames(cor.02a),
rownames(cor.02b)))#Lista com o nome das linhas comuns aos
#ojbetos cor.02a e cor.02b.
    sobrep02bc=Reduce(intersect, list(rownames(cor.02b),
rownames(cor.02c)))#Lista com o nome das linhas comuns aos
#objetos cor.02b e cor.02c.
    sobrep02ca=Reduce(intersect, list(rownames(cor.02c),
rownames(cor.02a)))#Lista com o nome das linhas comuns aos
#objetos cor.02c e cor.02a.
    cor.04=c(nrow(cor.02a), nrow(cor.02b), nrow(cor.02c),
length(cor.03))#Concatena o número de linhas dos objetos
#cor.02a, cor.02b, cor.02c e o número de
#observações do cor.03.
    names(cor.04)=c(namecateg[1], namecateg[2], namecateg[3],
paste0(namecateg[1], namecateg[2], namecateg[3]))#Concatena a sequências de
caracteres.
    cor.05=list(rownames(cor.02a), rownames(cor.02b), rownames(cor.02c),
cor.03)#cor.05 guardará em uma lista o
#nome das linhas (OTU_ID) dos
#objetos cor.02a cor.02b, cor.02c
#e o valor de observação do cor.03
#(sobreposição de cor.02a, cor.02b
#e cor.02c).
    names(cor.05)=names(cor.04)#Obter ou definir os nomes dos objeto.
    cor=list("Quantitative"=cor.04, "Qualitative"=cor.05)#A informação é
guardada na forma de lista.
    draw.triple.venn(nrow(cor.02a), nrow(cor.02b), nrow(cor.02c),
length(sobrep02ab),  length(sobrep02bc), length(sobrep02ca),length(cor.03),
```

```
category=c(namecateg[1], namecateg[2], namecateg[3]), lty="blank",
fill=c("blue", "red", "green"), alpha=0.5, cex=2, cat.cex=2)#draw.triple.vem
cria um diagrama de Venn com três
                                                                #conjuntos, no
caso, com o número de linhas dos objetos
                                                                #cor.02a,
cor.02b e cor.02c e os valores de
                                                                #sobreposição
entre esses objetos.
  } else if(numbercateg==4)#Se o número de tratamento for igual a 4.
  {
    stopifnot(length(namecateg)==numbercateg)#Verifica os argumentos da
função. No caso, o tamanho dos
                                                #identificadores únicos devem
ser iguais ao número de categoria.
    cor.01a=subset(bin, select=grepl(namecateg[1], names(bin)))#O objeto
cor.01a, baseado no objeto bin, guardará
                                                                   #apenas as
informações do primeiro identificador
                                                                   #único.
    cor.02a=subset(cor.01a, apply(cor.01a, 1, sum)==ncol(cor.01a))#A soma
das linhas do objeto cor.01a deve ser
                                                                      #igual ao
número de colunas do mesmo objeto.
                                                                      #O objeto
cor.02a guardará a informação do core
#microbiome da primeira categoria.
    cor.01b=subset(bin, select=grepl(namecateg[2], names(bin)))#O objeto
cor.01b, baseado no objeto bin, guardará
                                                                   #apenas as
informações do segundo identificador
                                                                   #único.
    cor.02b=subset(cor.01b, apply(cor.01b, 1, sum)==ncol(cor.01b))#A soma
das linhas do objeto cor.01b deve ser
                                                                      #igual ao
número de colunas do mesmo objeto.
                                                                      #O objeto
cor.02b guardará a informação do core
#microbiome da segunda categoria.
    cor.01c=subset(bin, select=grepl(namecateg[3], names(bin)))#O objeto
cor.01c, baseado no objeto bin, guardará
                                                                   #apenas as
informações do terceiro identificador
                                                                   #único.
    cor.02c=subset(cor.01c, apply(cor.01c, 1, sum)==ncol(cor.01c))#A soma
das linhas do objeto cor.01c deve ser
                                                                      #igual ao
número de colunas do mesmo objeto.
                                                                      #O objeto
cor.02c guardará a informação do core
#microbiome da terceira categoria.
```

```
    cor.01d=subset(bin, select=grepl(namecateg[4], names(bin)))#O objeto
cor.01d, baseado no objeto bin, guardará
                                                                #apenas as
informações do quarto identificador
                                                                #único.
    cor.02d=subset(cor.01d, apply(cor.01d, 1, sum)==ncol(cor.01d))#A soma
das linhas do objeto cor.01d deve ser
                                                                   #igual ao
número de colunas do mesmo objeto.
                                                                   #O objeto
cor.02d guardará a informação do core
#microbiome da quarta categoria.
    cor.03=Reduce(intersect, list(rownames(cor.02a), rownames(cor.02b),
rownames(cor.02c), rownames(cor.02d)))
    #Objeto cor.03 será o valor de sobreposição dos objetos cor.02a,
cor.02b, cor.02c e cor.02d, criando uma lista
    #com o nome das linhas que são comuns entre os quatro objetos (OTU_ID).
    sobrep02ab=Reduce(intersect, list(rownames(cor.02a), rownames(cor.02b)))
    #Lista com o nome das linhas comuns aos objetos cor.02a, cor.02b.
    sobrep02bc=Reduce(intersect, list(rownames(cor.02b), rownames(cor.02c)))
    #Lista com o nome das linhas comuns aos objetos cor.02b e cor.02c.
    sobrep02cd=Reduce(intersect, list(rownames(cor.02c), rownames(cor.02d)))
    #Lista com o nome das linhas comuns aos objetos cor.02c e cor.02d.
    sobrep02da=Reduce(intersect, list(rownames(cor.02d), rownames(cor.02a)))
    #Lista com o nome das linhas comuns aos objetos cor.02d e cor.02a.
    sobrep02ac=Reduce(intersect, list(rownames(cor.02a), rownames(cor.02c)))
    #Lista com o nome das linhas comuns aos objetos cor.02a e cor.02c.
    sobrep02bd=Reduce(intersect, list(rownames(cor.02b), rownames(cor.02d)))
    #Lista com o nome das linhas comuns aos objetos cor.02b e cor.02d.
    sobrep02abc=Reduce(intersect, list(rownames(cor.02a), rownames(cor.02b),
rownames(cor.02c)))
    #Lista com o nome das linhas comuns aos objetos cor.02a, cor.02b e
cor.02c.
    sobrep02bcd=Reduce(intersect, list(rownames(cor.02b), rownames(cor.02c),
rownames(cor.02d)))
    #Lista com o nome das linhas comuns aos objetos cor.02b, cor.02c e
cor.02d.
    sobrep02abd=Reduce(intersect, list(rownames(cor.02a), rownames(cor.02b),
rownames(cor.02d)))
    #Lista com o nome das linhas comuns aos objetos cor.02a, cor.02b e
cor.02d.
    sobrep02acd=Reduce(intersect, list(rownames(cor.02a), rownames(cor.02c),
rownames(cor.02d)))
    #Lista com o nome das linhas comuns aos objetos cor.02a, cor.02c e
cor.02d.
    cor.04=c(nrow(cor.02a), nrow(cor.02b), nrow(cor.02c), nrow(cor.02d),
length(cor.03))
    #Concatena o número de linhas dos objetos cor.02a, cor.02b, cor.02c,
cor.02d e
    #o número de observações do cor.03.
```

```
    names(cor.04)=c(namecateg[1], namecateg[2], namecateg[3], namecateg[4],
paste0(namecateg[1], namecateg[2], namecateg[3], namecateg[4]))#Concatena a
sequências de caracteres.
    cor.05=list(rownames(cor.02a), rownames(cor.02b), rownames(cor.02c),
rownames(cor.02d), cor.03)
    #cor.05 guardará em uma lista o nome das linhas (OTU_ID) dos objetos
cor.02a cor.02c, cor.02d e o
    #valor de observação do cor.03.
    names(cor.05)=names(cor.04)#Obter ou definir os nomes dos objeto.
    cor=list("Quantitative"=cor.04, "Qualitative"=cor.05)#A informação é
guardada na forma de lista.
    draw.quad.venn(nrow(cor.02a), nrow(cor.02b), nrow(cor.02c),
nrow(cor.02d), length(sobrep02ab), length(sobrep02bc), length(sobrep02cd),
length(sobrep02da), length(sobrep02ac), length(sobrep02bd),
length(sobrep02abc), length(sobrep02bcd), length(sobrep02abd),
length(sobrep02acd), length(cor.03), category=c(namecateg[1], namecateg[2],
namecateg[3], namecateg[4]), lty="blank", fill=c("blue", "red", "green",
"purple"), alpha=0.5, cex=2, cat.cex=2)
    #draw.quad.vem cria um diagrama de Venn com quatro conjuntos, no caso,
com o número de linhas dos objetos
    #cor.02a, cor.02b, cor.02c e cor.02d e os valores de sobreposição entre
esses objetos.
  }
return(cor)#Retorna a lista com as informações de core microbiome das
categorias análisadas.
  }
```

Help da Função

```
coremicrobiome               package:unknown               R Documentation

Análise do Core Microbime para estudos de Ecologia Microbiana

Description
Através da análise de presença e ausência de OTUs, podemos determinar o core
microbiome, que consiste no conjunto estável de membros compartilhados entre
diferentes grupos de amostras, ambientes ou tratamentos.

Usage
coremicrobiome <- function(otu_table, numbercateg, namecateg)

Arguments
otu_table: Dataframe (otu_table). O arquivo otu_table deve seguir o formato
clássido do Qiime, podendo conter ou não informações taxonômicas.
Ver: https://www.drive5.com/usearch/manual/qiime_classic.html

numbercateg: Número de categorias. Para esta função, limitaremos o número de
categorias a quatro. A partir de cinco categorias, o Venn Diagram se torna
```

```
pouco intuitivo. Sugiro a utilização de scatterplot para cinco ou mais
categorias.

namecateg: Nome das categorias. Importante: Cada categoria deve conter um
conjunto de letras possíveis de serem utilizados como identificadores
únicos.

Details
Através desta função podemos determinar o core microbiome de diferentes
grupos de amostras, ambientes ou tratamentos análisando a presença e
ausência de OTUs
nos diferentes grupos avaliados. Serão considerados pertencentes ao core
microbiome apenas as OTUs presentes em 100% das amostras das categorias
comparadas.

Value
Esta função retornará uma lista com o número de OTUs (e suas respectivas
identificações) que foram consideradas core microbiome para as categorias
análisadas.
Venn Diagram com os valores de core microbiome de cada categorias e os
valores de core microbiome entre categorias.

Author
Francielli Vilela Peres
francielliperes@usp.br

References
*SHADE, A.; HANDELSMAN, J. Beyond the Venn diagram: the hunt for a core
microbiome. Environmental Microbiology, v. 14, n. 1, p. 4-12, 2012.
*ZAURA, E.; KEIJSER, B. J. F.; HUSE, S. M.; CRIELAARD, W. Defining the
healthy “core microbiome” of oral microbial communities. BMC Microbiology,
p. 1-12, 2009.
*NAM-PHUONG NGUYEN, N.; WARNOW, T.; MIHAI POP, M.; WHITE, B. A perspective
on 16S rRNA operational taxonomic unit clustering using sequence similarity.
Biofilms and Microbiomes, 2016.



Examples

## Dataframe de Input

otu_table <- read.csv("otu_table.csv", header=TRUE, row.names=1, sep=";",
dec=".")
otu_table

     Cnt1    Cnt2    Cnt3    Lag1    Lag2    Lag3    Mar1    Mar2    Mar3
Esg1    Esg2    Esg3
OTU1     30.0    21.0    13.0    12.0    300.0    15.0    0.0    3.0    7.0
5.0    0.0    1.0
OTU10    32.0    120.0    41.0    10.0    482.0    42.0    219.0    86.0
```

```
507.0    0.0    3.0    1.0
OTU100   6.0    2.0    0.0    1.0    2.0    0.0    1.0    3.0    0.0
122.0    382.0    793.0
OTU1000  1.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
0.0    2.0
OTU1001  11.0    2.0    0.0    1.0    0.0    0.0    0.0    0.0    0.0    0.0
0.0    0.0
OTU1002  0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    9.0
0.0    0.0
OTU1003  0.0    2.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
8.0    0.0
OTU1004  0.0    24.0    0.0    0.0    2.0    0.0    0.0    0.0    0.0    0.0
0.0    0.0
OTU1005  7.0    0.0    9.0    0.0    2.0    0.0    0.0    0.0    0.0    0.0
0.0    0.0
OTU1006  11.0    23.0    19.0    56.0    14.0    76.0    0.0    7.0    7.0
0.0    0.0    0.0
OTU1007  9.0    0.0    3.0    0.0    7.0    0.0    7.0    0.0    2.0    7.0
2.0    8.0
OTU1008  0.0    18.0    18.0    0.0    0.0    0.0    0.0    0.0    0.0
0.0    0.0    0.0
OTU1009  0.0    0.0    0.0    0.0    0.0    13.0    0.0    0.0    0.0    0.0
0.0    0.0
OTU1010  4.0    17.0    24.0    20.0    43.0    82.0    0.0    0.0    28.0
0.0    0.0    0.0
OTU1011  0.0    0.0    0.0    0.0    0.0    10.0    0.0    0.0    26.0
0.0    0.0    0.0
OTU1012  7.0    0.0    3.0    3.0    0.0    2.0    0.0    0.0    0.0    18.0
5.0    7.0
OTU1013  0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
0.0    0.0
OTU1014  0.0    0.0    0.0    28.0    2.0    10.0    0.0    0.0    0.0
0.0    0.0    0.0
OTU1015  0.0    7.0    8.0    2.0    0.0    0.0    0.0    0.0    0.0    2.0
53.0    0.0
OTU1016  0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    3.0
0.0    2.0
OTU1017  0.0    32.0    25.0    0.0    0.0    0.0    0.0    0.0    0.0
0.0    0.0    0.0


## Output

Esta função retornará uma lista com o número de OTUs (e suas respectivas
identificações) que foram consideradas core microbiome para as categorias
análisadas.
Venn Diagram com os valores de core microbiome de cada categorias e os
valores de core microbiome entre categorias.
```

Arquivos da Função:
Função:funcao.txt

Help:help_funcao.txt
Input:otu_table.csv
Output1:
Output2: