```
diver <- function (pi, alfa=c("shannon", "simpson"), beta.gam=TRUE, perm) {
## Criando a funcao "diver" com os argumentos "pi", "alfa", "beta.gam" e
"perm"

    ## Definindo argumentos da funcao

    pi <- as.matrix(pi) ## Definindo a classe do argumento "pi"
    pi[pi==0] <- NA ## Atribuindo NA a cada numero 0 achado no objeto "pi"
(ausencia de especie)
    alfa <- match.arg(alfa) ## Indicando que o usuario desta funcao poderia
usar só as letras iniciais dos indices usados no argumento "alfa"

    ## Calculando a diversidade gama

    gam <- nrow(pi) ## Selecionando o numero de linhas do objeto "pi", o
qual é equivalente ao numero de especies totais

    ## Calculando a diversidade alfa em termos do exponencial do indice
"Shannon Entropy"

    if (alfa=="shannon") ## Criando condicao
    {

        shannon <- function(x) { exp((-sum((x/sum(x,
na.rm=TRUE))*(log(x/sum(x, na.rm=TRUE))), na.rm=TRUE))) } ## Criando a
funcao que calculará o exponencial do indice "shannon entropy"

        s <- apply(pi,2,shannon) ## Aplicando o indice as colunas da matriz
"pi" e guardando o resultado no objeto "s"

        ## Fazendo uma reamostragem com reposicao para calcular a icerteza
dos calculos do indice por colunas (BOOSTRAP)

        res <- matrix(NA, nrow= perm, ncol= ncol(pi)) ## Criando uma matriz
de NA's com o mesmo numero de colunas da matriz "pi" e com o numero de
linhas equivalente ao numero de permutacoes que o usuario escolha fazer.
Essa matriz é guardada no objeto "res"

        for (i in 1:perm) ## Criando um ciclo com repeticoes igual ao numero
de permutacoes que o usuario escolha fazer
        {

            ram <- apply(pi,2,sample,replace=T) ## Aleatorizando com
reposicao as especies da matriz "pi" por cada coluna e guardando essas
aleatorizacoes no objeto "ram"
            cal <- apply(ram,2,shannon) ## Aplicando o indice toda vez que
as especies do objeto "ram" sao aleatorizadas e guardando o resultado no
objeto "cal"
            res[i,] <- cal ## Preenchendo as linhas da matriz "res" com os
resultados calculados no objeto "cal"
```

```
        }

        colnames(res) <- colnames(pi) ## Renomeando as colunas do objeto
"res" de forma que os nomes sejam iguais aos usados na matriz "pi"
        IC <- apply(res, 2, quantile, p=c(0.025, 0.95)) ## Calculando os
quantis para os intervalos de confianca ao 95%
        ic.s <- s + IC[2,] ## Somando o intervalo superior com o valor do
indice calculado no objeto "s"
        ic.i <- s - IC[1,] ## Somando o intervalo inferior com o valor do
indice calculado no objeto "s"
        res ## Confirmando os valores
        s ## Confirmando os valores
    }

    ## Calculando a diversidade alfa em termos do inverso do indice "Simpson
Concentration"


    if (alfa=="simpson") ## Criando condicao
    {

        simpson <- function(y) { 1/(sum((y/sum(y, na.rm=TRUE))^2,
na.rm=TRUE)) } ## Criando uma funcao que calculará o inverso do indice
"simpson concentration"

        s2 <- apply(pi, 2, simpson) ## Aplicando o indice por cada coluna da
matriz "pi" e guardando o resultado no objeto "s2"

        ## Fazendo uma reamostragem com reposicao para calcular a icerteza
dos calculos do indice por colunas (BOOSTRAP)

        res2 <- matrix(NA, nrow= perm, ncol= ncol(pi)) ## Criando uma matriz
de NA's com o mesmo numero de colunas da matriz "pi" e com o numero de
linhas equivalente ao numero de permutacoes que o usuario escolha fazer.
Essa matriz é guardada no objeto "res2"

        for (i in 1:perm) ## Criando um ciclo com repeticoes igual ao numero
de permutacoes que o usuario escolha fazer
        {

            ram2 <- apply(pi, 2, sample, replace=TRUE) ## Aleatorizando com
reposicao as especies da matriz "pi" por cada coluna e guardando essas
aleatorizacoes no objeto "ram2"
            cal2 <- apply(ram2, 2, simpson) ## Aplicando o indice toda vez
que as especies  do objeto "ram2" sao aleatorizadas e guardando o resultado
no objeto "cal2"
            res2[i,] <- cal2  ## Preenchendo as linhas da matriz "res2" com
os resultados calculados no objeto "cal2"

        }
```

```
        colnames(res2) <- colnames(pi) ## Renomeando as colunas do objeto
"res" de forma que os nomes sejam iguais aos usados na matriz "pi"
        IC2 <- apply(res2, 2, quantile, p=c(0.025, 0.95))  ## Calculando os
quantis para os intervalos de confianca ao 95%
        ic.s2 <- s2 + IC2[2,] ## Somando o intervalo superior com o valor do
indice calculado no objeto "s2"
        ic.i2 <- s2 - IC2[1,] ## Somando o intervalo inferior com o valor do
indice calculado no objeto "s2"
        res2 ## Confirmando os valores
        s2 ## Confirmando os valores

    }

    ## Calculando a diversidade beta

    if (alfa=="shannon" & beta.gam==TRUE) ## Criando condicao
    {

            dist <- matrix(NA, nrow=dim(pi)[2], ncol=dim(pi)[2]) ## Criando
uma matriz de distancia preenchida com NA's, com numero de linhas e colunas
igual a dimensao 2 da matriz "pi", ou seja, igual ao numero de colunas da
matriz "pi". A matriz é guardada no objeto "dist"
            dimnames(dist) <- list(colnames(pi),colnames(pi)) ## Renomeando
as linhas e colunas da matriz "dist" de forma que sejam iguais aos nomes da
matriz "pi"
        for (i in 1:dim(pi)[2]-1) ## Criando um ciclo com o contador i que
percorra desde a primera até a penultima coluna da matriz "pi"
        {


            for (j in (i+1):dim(pi)[2]) ## Criando um ciclo com o contador j
que percorra desde a coluna i+1 até a ultima coluna da matriz "pi"

            {

                a <- length(na.omit(pi[,i])) ## Calculando o comprimento das
colunas percorridas pelo contador i, omitindo os NA's da matriz "pi"
(equivalente ao numero de especie das colunas percorridas pelo contador i).
Os resultados sao guardados no objeto "a"

                b <- length(na.omit(pi[,j])) ## Calculando o comprimento das
colunas percorridas pelo contador j, omitindo os NA's da matriz "pi"
(equivalente ao numero de especie das colunas percorridas pelo contador j).
Os resultados sao guardados no objeto "b"

                J <- (sum(as.numeric(as.numeric(!is.na(pi[,i])) +
as.numeric(!is.na(pi[,j]))>1))) ## Calculando as especies compartilhadas das
colunas i e j

                dist[i,j] <- round((a + b - 2*J)/(a + b),2) ## Aplicando a
formula da diversidade beta e guardando o resultado nas colunas i e j da
```

```
matriz "dist". (só o triangulo superior da matriz "dist" é preenchido com os
valores da diversidade beta)

            }

        }

            diag(dist) <- 1 ## Atribuindo 1 a diagonal da matriz "dist"
            x <- 1:ncol(pi) ## Preparando o eixo x do grafico
            par(las=1, bty="l", family="serif", cex.lab=1.2, mar=c(7,5,2,2))
## Definindo os parametros do grafico
            plot(s, type="b", pch=19, col="orange",
ylim=range(c(ic.s,ic.i)), xlab="Sample sites", ylab=" Exp. Shannon index")
## Graficando os valores do indice por cada coluna da matriz "pi"
            arrows(x, ic.s, x, ic.i, code= 3, angle=90, length=0.05) ##
Graficando os intervalos de confianca
            return(list(shan=s, confidence.inter=IC, beta=dist, gam=gam)) ##
Retornando os valores do exponencial do indice shannon entropy, os
intervalos de confianca dos calculos, a diversidade beta, a diversidade gama
e o grafico, caso esta condicao seja verdadeira

    }

    ## A seguinte condicao faz o mesmo que a anterior, só que ela retorna o
inverso do indice de Simpson Concentration ao inves do exponencial Shannon
Entropy no caso a condicao seja verdadeira. Nao tem comentarios, pois seriam
os mesmo comentarios da condicao anterior


    if (alfa=="simpson" & beta.gam==TRUE)
    {
        dist2 <- matrix(NA, nrow=dim(pi)[2], ncol=dim(pi)[2])
        dimnames(dist2) <- list(colnames(pi),colnames(pi))

        for (i in 1:dim(pi)[2]-1)
        {

            for (j in (i+1):dim(pi)[2])
            {
                a <- length(na.omit(pi[,i]))
                b <- length(na.omit(pi[,j]))
                J <- (sum(as.numeric(as.numeric(!is.na(pi[,i])) +
as.numeric(!is.na(pi[,j]))>1)))
                dist2[i,j] <- round((a + b - 2*J)/(a + b),2)
            }

        }


        diag(dist2) <- 1
```

```
        ic.s2 <- s2 + IC2[2,]
        ic.i2 <- s2 - IC2[1,]
        x2 <- 1:ncol(pi)
        par(las=1, bty="l", family="serif", cex.lab=1.2, mar=c(7,5,2,2))
        plot(s2, type="b", pch=19, col="orange", ylim=range(c(ic.s2,ic.i2)),
xlab="Sample sites", ylab=" Inv. Simpson index")
        arrows(x2, ic.s2, x2, ic.i2, code= 3, angle=90, length=0.05)
        return(list(simp=s2, confidence.inter=IC2, beta=dist2, gam=gam))

    }

    ## Tornando alguns argumentos flexiveis, neste caso o argumento
"beta.gam", de forma que o usuario possa decidir se fazer os calculos da
diversidade beta e gama ou nao

    if (alfa=="shannon" & beta.gam==FALSE) ## Criando condicao
    {
        x <- 1:ncol(pi) ## Preparando o eixo x do grafico
        par(las=1, bty="l", family="serif", cex.lab=1.2, mar=c(7,5,2,2)) ##
Preparando parametros do grafico
        plot(s, type="b", pch=19, col="orange", ylim=range(c(ic.s,ic.i)),
xlab="Sample sites", ylab=" Exp. Shannon index") ## Graficando os valores do
indice Shannon Entropy por colunas/amostras da matriz "pi"
        arrows(x, ic.s, x, ic.i, code= 3, angle=90, length=0.05) ##
Graficando os intervalos de confianca dos calculos
        return(list(shan=s, confidence.inter=IC)) ## Retorna o calculo do
exponencial do indice Shannon Entropy, os intervalos de confianca e o
grafico.
    }

    if (alfa=="simpson" & beta.gam==FALSE) ## Criando condicao
    {
        x2 <- 1:ncol(pi) ## Preparando o eixo x
        par(las=1, bty="l", family="serif", cex.lab=1.2, mar=c(7,5,2,2)) ##
Preparando os parametros do grafico
        plot(s2, type="b", pch=19, col="orange", ylim=range(c(ic.s2,ic.i2)),
xlab="Sample sites", ylab=" Inv. Simpson index") ## Graficando os valores do
indice Simpson Concentration por colunas-amostras da matriz "pi"
        arrows(x2, ic.s2, x2, ic.i2, code= 3, angle=90, length=0.05) ##
Graficando os intervalos de confianca
        return(list(simp=s2, confidence.inter=IC2)) ## Retorna o calculo do
inverso do indice Simpson Concentration, os intervalos de confianca dos
calculos e o grafico

    }


}
```