Proposta 1: Índice de diversidade (shannon e simpson) por reamostragem

```
diversity.random<- function(x, indice = c("sh", "si"), nsim=100)
{
  x<- data.frame(x) #transforma o objeto de entrada em um data frame
  indice<- match.arg(indice) # indexa os niveis do argumento "indice" que
serão empregados para os "ifs"
  nsim<- nsim #numero de simulações utilizados para a aleatorização dos
dados
  x[x== " "] <- NA # converte os valores faltantes em NA's
  x[is.na(x)]<-0 #converte NA's em zero
  result<- rep(NA, nrow(x)) # cria um vetor com a mesma quantidade de linhas
do data frame de entrada
  if(indice == "sh" ) # se argumento "indice" for igual a "sh", retorna o
valores do indice de shannon
    for(j in 1:length(x[,1])) # permite calculo através das linhas
    {
      ind.sh<- -sum((x[j,]/(sum(x[j,])))* ((log(x[j,]/(sum(x[j,])))),na.rm
= TRUE) # divide o valor da linha "[j,]" pela sua soma, em seguida irá
multiplica pelo logaritimo natural, somando tudo ao final.
      result[j]<- ind.sh  #carrega os valores calculados no objeto "results"
    }
  if(indice =="si") # se argumento "indice" for igual a "si", retorna os
valores do indice de simpson
    for(j in 1:length(x[,1])) # permite calculo através das linhas
    {
      ind.si<-  sum((x[j,]/(sum(x[j,])))^2, na.rm=TRUE) # divide o valor da
linha "[j,]" pela sua soma, em seguida irá multiplica pelo logaritimo
natural, somando tudo ao final.
      result[j]<- 1- ind.si #carrega os valores calculados no objeto
"results"
    }
  s.result<- list(rep(NA, nrow(x)),nsim) # irá retornar um lista com uma
quantide de matrizes igual ao numero de simulações informados no "nsim"
  s.result[[1]]<-result #guarda o indice observado no primeiro objeto da
lista "s.result"
  if(indice == "sh" ) # se indice igual a "sh" o looping irá calcular os
indices de shannon e guardar na lista a partir do segundo objeto desta.
    for (i in 2: nsim) #contador de 2 ao comprimento do numero de
simulações.
    {
      sx<- x[sample(nrow(x)),] # aletorizaçao das linhas do data frame
      sx2<- sample(sx) # aletorizaçao das colunas do data frame
      colnames(sx2)=colnames(x) #renomeia as colunas do data frame
aleatorizado com a sequencia original contida no data frame de entrada
      rownames(sx2)=rownames(x) #renomeia as linhas do data frame
aleatorizado com a sequencia original contida no data frame de entrada
      total<- apply(sx2, 1, sum,na.rm=TRUE) #soma o total por linha
      li<- sx2/total #divide cada valor da linha pelo seu total
      log.li<- li*log(li) #valores das linhas multiplicados por seu
logaritimo natural
```

```
      ind.sh<- abs(apply(log.li,1, sum, na.rm=TRUE)) #soma os valores das
linhas multiplicadas por seu logaritimo natural e, na sequencia converte os
valores negativos.
      s.result[[i]]<- ind.sh #guarda as simulações dentro da lista
"s.result"
    }
  if(indice =="si") # se indice igual a "si" o looping irá calcular os
indices de shannon e guardar na lista a partir do segundo objeto desta.
    for(i in 2: nsim) #contador de 2 ao comprimento do numero de simulações.
    {
      sx<- x[sample(nrow(x)),] # aletorizaçao das linhas do data frame
      sx2<- sample(sx) # aletorizaçao das colunas do data frame
      colnames(sx2)=colnames(x) #renomeia as colunas do data frame
aleatorizado com a sequencia original contida no data frame de entrada
      rownames(sx2)=rownames(x) #renomeia as linhas do data frame
aleatorizado com a sequencia original contida no data frame de entrada
      total<- apply(sx, 1, sum,na.rm=TRUE) #soma o total por linha
      li<- sx/total  #divide cada valor da linha pelo seu total
      ind.sim<- abs(1- apply((li^2), 1, sum, na.rm=TRUE)) #calcula o indice
de simpson (1- linhas dividas por sua soma e elevadas ao quadrado), na
sequencia converte os valores negativos
      s.result[[i]]<- ind.sim #guarda as simulações dentro da lista
"s.result"
      }
  if(indice=="sh") # se indice igual a "sh"
  {
    sh.simul<- sapply(s.result, "[", 1: nrow(x)) #transpõe as linhas das
listas em colunas
    sh.logic<- apply(sh.simul>=result, 1, sum)/nsim # faz o teste logico
observando a chance de se obter aleatoriamente um valor maior ou igual ao do
indice observado para cada parcela.
    p.valor<-sh.logic #guarda os valores dentro do objeto "pvalor"
  }
  if(indice=="si") # se indice for igual a "si"
  {
    sim.simul<- sapply(s.result, "[", 1: nrow(x)) #transpõe as linhas das
listas em colunas
    sim.logic<- apply(sim.simul>=result, 1, sum)/nsim # faz o teste logico
observando a chance de se obter aleatoriamente um valor maior ou igual ao do
indice observado para cada parcela.
    p.valor<-sim.logic #guarda os valores dentro do objeto "pvalor"
  }
  {
    matriz.indice<- result #guarda os indices observados dentro da matriz
"matriz.indice"
    matriz.indice<- as.data.frame(matriz.indice) #transforma o objeto
"matriz.indice" em um data frame
    rownames(matriz.indice)<- rownames(x) #atribui a linhas do onjeto
"matriz" os nomes da linhas data frame de entrada
  }
```

```
  {
    matriz.p<- matrix(p.valor) #guarda os valores de p obtidos por meio das
simulações no objeto "matriz.p"
    colnames(matriz.p) <- "valor-p" #Renomenia o nome da coluna do objeto
    rownames(matriz.p)<- rownames(x) #renomei as linhas com os nomes do data
frame de entrada
  }
  if(indice=="sh") # se indice for igual a "sh"
  {
    m.result=apply(sh.simul,1,mean) #calcula a média dos valores observados
    interv.conf=apply(sh.simul,1,quantile,probs=c(0.025,0.975)) # calcula o
IC para os valores obtidos por meio valores simulados
  }
  if(indice=="si") # se indice for igual a "si"
  {
    m.result=apply(sim.simul,1,mean) #calcula a média dos valores observados
    interv.conf=apply(sim.simul,1,quantile,probs=c(0.025,0.975)) # calcula o
IC para os valores obtidos por meio valores simulados
  }
  names(m.result)=rownames(x) #atribui ao objeto "m.result" o nome das linha
do data frame de entrada
  colnames(interv.conf)=rownames(x) #atribui ao objeto "interv.conf" o nome
da linhas do data frame de entrada
  output<-list(s.result, matriz.indice, matriz.p, m.result, interv.conf)
#retorna os valores das simulações, os valores dos indices observados, o
valor de p, a média das simulações e o intervalo de confiança dos valores
simulados.
  names(output)<- c("Simulações", "Indice de diversidade observado", "valor-
p", "Media das simulações", "IC - simulações") #renomei os objetos
retornados a partir da lista gerado no "return"
  return(output) # retorná uma lista contendo as simulações, uma matriz com
valores guardados em "result", e uma segunda matriz contendo o valor de p,
calculado a partir das simulações
 }
```