Esta função chama-se “autocor” e testa a autocorrelação espacial entre sítios, a partir da distância geográfica e a dissimilaridade na composição e abundância de espécies entre sítios.

```
autocor = function(x, y, dec = TRUE, binario = TRUE, perm = 1000) # função
{
  NAx=sum(is.na(x)) # cria um objeto com as somas de NAs em x
  if (NAx > 0) # se houver algum NA, a função para
    {
    stop ("Alguns sítios não possuem coordenadas geográficas") # texto
retornado quando a função é interrompida
    }
  NAy=sum(is.na(y)) # cria um objeto com a soma de NAs em y
  y[which(is.na(y))]=0 # substitui todo valor NA em y por 0
  if(length(x[,1])!=length(y[,1])) # verifica se os dois data.frames possuem
o mesmo número de sítios
    {
    stop ("Data.frames com número de sítios distintos") # texto retornado
quando a função é interrompida
    }
  if(dec==FALSE) # Neste caso o usuário apresentou uma planilha com
coordenadas em graus, minutos e segundos
    {
      conv = data.frame(x[,1], x[,2]+x[,3]/60+x[,4]/3600, x[,5],
x[,6]+x[,7]/60+x[,8]/3600, x[,9]) # conversão dos dados para graus decimais
      colnames(conv) = c("sitio","lat", "h1", "lon", "h2") # conversão dos
nomes das colunas
      for (i in 1:length(conv[,1]))  # início da ciclagem para verificação
de qual hemisfério o dado pertence
        {
        if (conv[i,3] == "S") # Hemisfério sul?
         {
         conv[i,2] = conv[i,2]*(-1) # Se sim, os dados ficam negativos
         }
        if (conv[i,5] == "O") # Hemisfério ocidental?
         {
         conv[i,4] = conv[i,4]*(-1) # se sim, os dados ficam negativos
         }
      }
    }
  if(dec==TRUE) # Como os dados estão em graus decimais, este script somente
altera o nome do velor e nome das colunas somente para o script ler os dados
   {
    conv = data.frame(x[,1], x[,2], x[,3], x[,4], x[,5])
    colnames(conv) = c("sitio","lat", "h1", "lon", "h2")
   }
  conv[, c(2,4)] = conv[, c(2,4)]*pi/180 # Tranformando os dados em
radianaos
  ### Calcular as distâncias geográficas entre os sítios ######
  dist=matrix(0, ncol=length(conv$sitio), nrow=length(conv$sitio),
dimnames=list((conv[,1]),(conv[,1]))) # criando uma matriz para inserir os
```

```
dados gerados pelo contador abaixo
  r = 6371  # raio da terra em km
  for(i in 1:length(conv$sitio)-1) # Ciclagem para calcular as distâncias
geográficas entre cada par de sítio
    {
      for (j in (i+1):length(conv$sitio)) # ciclo para cálculo entre pares
de sítios
      {
      diflat=(conv$lat[j]-conv$lat[i]) # diferenças entre latitudes
      diflon=(conv$lon[j]-conv$lon[i]) # diferenças entre longitudes
      dist[i,j] =
2*r*asin(sqrt((sin(diflat/2)^2)+cos(conv$lat[i])*cos(conv$lat[j])*(sin(diflo
n/2)^2))) # Aplicação da fórmula de Haversine e inserção dos dados criados
na matriz dist
      dist[j,i] =
2*r*asin(sqrt((sin(diflat/2)^2)+cos(conv$lat[i])*cos(conv$lat[j])*(sin(diflo
n/2)^2))) # Repetição para inserir todos os dados na matriz
      }
    }
  dist.b=dist[lower.tri(dist)] # cria um vetor com os dados de distância
geográfica
  ############## Obtendo os dados de similaridade entre sítios ##########
  library("vegan") # abrir o pacote Vegan
  s= y[, 2:length(y)]# convertendo o data.frame "y" no formato que o vegdist
trabalha
  if(binario == TRUE) # Para dados qualitativos - presença/ausência, índice
de jaccard
    {
    indice = vegdist(s, indice = "jaccard", binary = TRUE)
    }
  if(binario == FALSE) # para dados quantitativos - abundância das espécies,
índice Bray-Curtis
    {
    indice = vegdist(s, indice = "jaccard", binary = FALSE)
    }
  indice = as.matrix(indice) # transformando o resultado das similaridades
em uma matrix
  dimnames(indice)=list(y[,1], y[,1]) # atribuindo o nome dos sítios à
matriz
  indice.b=indice[lower.tri(indice)] # cria um vetor com os dados de
similaridade
  ######### Cálculo da correlação de Pearson #############
  num<-rep(NA,length(dist.b))# Cria um vetor que irá ser o numerador da
formula do coeficiente de correlação de Pearson
  den.1<-rep(NA,length(dist.b))# Cria um vetor que irá ser o primeiro termo
do denominador da formula
  den.2<-rep(NA,length(indice.b))# Cria um vetor que irá ser o segundo termo
do denominador da formula.
  for(a in 1:length(dist.b))#ciclagem que irá calcular o coeficiente de
correlação de Pearson
```

```
    {
    num[a]=(dist.b[a]-mean(dist.b))*(indice.b[a]-mean(indice.b))# Calculo do
numerador
    den.1[a]=(dist.b[a]-mean(dist.b))^2# Calculo do primeiro termo do
denominador
    den.2[a]=(indice.b[a]-mean(indice.b))^2# Calculo do segundo termo do
denominador
    }
  pearson.obs=sum(num)/(sqrt(sum(den.1))*sqrt(sum(den.2)))# Calculo do
coeficiente de correlação de Pearson
  pearson.est=rep(NA, perm)# Cria vetor para serem inseridas as correlações
estimadas
  for (f in 1:perm) # Ciclagem para realizar o número de permutações
escolhidas pelo usuário, para cálculo das correlações de Pearson
    {
    indice.aleatorio<-sample(indice.b)# Aleatorizando o valor das
similaridades entre sítios
      for(h in 1:length(indice.b))# Cálculos das correlações aleatórias
      {
      num[h]=(dist.b[h]-mean(dist.b))*(indice.aleatorio[h]-
mean(indice.aleatorio)) # Calculo do numerador
      den.1[h]=(dist.b[h]-mean(dist.b))^2 # Cálculo do primeiro termo do
denominador
      den.2[h]=(indice.aleatorio[h]-mean(indice.aleatorio))^2 # Cálculo do
segundo termo do denominador
      }
    pearson.est[f]<-sum(num)/(sqrt(sum(den.1))*sqrt(sum(den.2))) #
Adicionando os valores estimados de pearson no vetor criado.
    }
  quartz() # abre uma nova janela gráfica, isso para o mac, para windows =
X11()
  par (mfrow = c(1,2)) # dá o comando de montar dois gráficos em uma linha,
nesta janela gráfica
  plot (dist.b, indice.b, xlab = "Dist. geográfica", cex.axis=0.8, ylab =
"Dissimilaridade", bty = "l", pch=16) # plota os valores das distâncias
geográficas e das similaridade
  abline(lm(indice.b~dist.b),col="red") # plota a linha com o valor da
correlação de pearson observada
  title(paste("Pearson obs. =", round(pearson.obs, digits = 3))) # plota o
valor de Pearson observado, se significativo, fica na coloração vermelha
  hist(pearson.est, axes=T, main="", xlab="Valores estimados de Pearson",
ylab="Frequência", xlim = c(-1,1), cex.axis=0.8) # monta um histograma com a
frequencia das correlações observadas
  abline(v=pearson.obs,col="red") # plota a linha com o valor da correlação
de pearson observada
  abline(v=mean(pearson.est), col = "blue") # plota a média dos valores de
pearson estimados em azul
  abline(v=mean(pearson.est)-pearson.obs+mean(pearson.est), col = "red") #
plota a linha do valor correspondente de pearson observada pra o teste
bicaudal
  p.cont.menor=length(which(pearson.est<pearson.obs)) # calcula proporção de
```

```
valores abaixo do Pearson observado
  p.cont.maior=length(which(pearson.est>((mean(pearson.est)-
pearson.obs)+mean(pearson.est))))# calcula proporção de valores acima do
Pearson observado
  p.cont = (p.cont.menor+p.cont.maior)/length(pearson.est) # Cálculo do "P"
  title(paste("P = ", round(p.cont, digits = 3)), col = "red") # Quando
pearson não significativo
  planilhas=list ()# Cria uma lista chamada "planilhas"
  planilhas$NAs.substituídos.y=NAy# Coloca na lista o total de NAs
substituídos por 0
  planilhas$Dist.geográfica=dist# Coloca na lista a matriz de distâncias
geográficas entre sítios
  planilhas$Dissimilaridade=indice# Coloca na lista a matriz de
dissimilaridade entre sítios
  return(planilhas)# Retorna o total de Nas e as matrizes de distância
geográfica e de dissimilaridades
}
```

Help da função

```
autocor                package:unknown                R Documentation

Autocorrelação espacial - autocor

Descrição:

Esta função calcula um valor de correlação de Pearson (estatística r de
Mantel) para dois conjuntos de dados de diferentes sítios: (1) distâncias
geográficas euclidianas entre diferentes sítios; (2) dissimilaridades na
composição ou abundância de espécies nos diferentes sítios.
Esta função também realiza permutações (teste de Mantel) com os dados de
dissimilaridade aleatorizados para avaliar a significância estatística do
valor observado de Pearson.

Usage:

autocor = function(x, y, dec = TRUE, binario = TRUE, perm = 1000)

Argumentos:

x - É um data.frame, contendo nas colunas as coordenadas geográficas de cada
sítio de amostragem.

y - É um data.frame, contendo nas colunas, a presença/ausência ou abundância
de cada espécie por sítio de amostragem.

dec - É um argumento que o usuário poderá indicar se os dados em “x” estão
em graus decimais (TRUE), ou em graus, minutos e segundos (FALSE).
```

binario - É um argumento que o usuário indica se seus dados correspondem a presença/ausência (TRUE) de espécies ou se corresponde a dados de abundância (FALSE). Este argumento ainda permite que dados de abundância sejam transformados em presença/ausência (TRUE). O índice de dissimilaridade utilizado nesta função é o de Jaccard.

perm - É um argumento onde o usuário indica o número de permutações a serem realizadas no teste de Mantel.

Detalhes:

Esta função não aceita valores NAs no data.frame x, e somente aceita dois formatos de coordenadas geográficas: (1) graus decimais e (2) graus, minutos e segundos. Os hemisférios também devem ser discriminado em uma coluna a parte (ver exemplos abaixo).
Caso os dados estejam no formato 2, os valores de graus, minutos e segundos devem estar segregados pelo separador, ou seja estes valores devem ser apresentados em coluna separadas (ver exemplos abaixo).
Esta função transforma automaticamente os dados NAs em x em 0.
Nesta função, o índice de Jaccard é calculado a partir da função vegdist, contida no pacote vegan, desta forma tal pacote deve ter sido previamente instalado em seu computador. Tal pacote pode ser baixado pelo link: http://CRAN.R-project.org/package=vegan.
Esta função não permite a utilização de outros índices de dissimilaridade contidos na função vegdist. Mas caso o usuário deseje utilizar outro índice ele pode acrescentar o script para o cálculo do referido índice.

Valor:

Esta função retorna o total de valores NAs substituídos por 0 do data.frame y e duas matrizes, (1) distâncias geográficas e (2) dissimilaridade entre os diferentes sítios. Esta função retorna uma gráfico de dispersão entre as distâncias geográficas e similaridades entre sítios, apresentando a inclinação da reta (linha vermelha) e o valor calculado da correlação de Pearson entre as duas matrizes.
A função ainda retorna um histograma mostrando os valores estimados de Pearson, bem como duas linhas vermelhas correspondentes aos valores de Pearson observados (distribuição bi-caudal), assim como o valor da média de Pearson estimado (azul), além de exibir, neste histograma, o nível de significância do valor de Pearson observado em relação ao valores de Pearson estimado.

Cuidado:

Esta função não aceita valores NAs no data.frame x.
O número de sítios em x e y devem ser iguais.
A função reconhece somente dois formatos de coordenadas geográficas (graus decimais e graus, minutos e segundos)
É necessário instalar o pacote Vegan em seu computador previamente a executar a função (http://CRAN.R-project.org/package=vegan).

```
Note:

Para maiores informações sobre a forma do cálculo do índice de
dissimilaridade, ver também o help da função vegdist do pacote Vegan.

Autor(s):

Ricardo Sampaio (rcosampaio@gmail.com, ricardo.sampaio@icmbio.gov.br)

Referencias:

http://pt.wikipedia.org/wiki/Coeficiente_de_correla%C3%A7%C3%A3o_de_Pearson
http://en.wikipedia.org/wiki/Mantel_test
Jari Oksanen, F. Guillaume Blanchet, Roeland Kindt, Pierre Legendre, Peter
R. Minchin, R. B. O'Hara, Gavin L.  Simpson, Peter Solymos, M. Henry H.
Stevens and Helene Wagner (2015). vegan: Community Ecology Package. R
package version 2.2-1. http://CRAN.R-project.org/package=vegan
Legendre, P. (1993). Spatial Autocorrelation : Trouble or New Paradigm?
Ecology, 74(6), 1659–1673.

Exemplo 1:
Faça o download dos arquivos dist1.txt e diss1.text
dist1=read.table(“dist1.txt", sep = "\t", header = T)
diss1=read.table(“diss1.txt", sep = "\t", header = T)
autocor(dist1, diss1, dec = FALSE, binario = TRUE, perm = 1000)

Exemplo 2:
Faça o download dos arquivos dist2.txt e diss2.text
dist2=read.table(“dist2.txt", sep = "\t", header = T)
diss2=read.table(“diss2.txt", sep = "\t", header = T)
autocor(dist2, diss2, dec = TRUE, binario = FALSE, perm = 1000)
```

dist1.txt diss1.txt dist2.txt diss2.txt