André Mouro D'Angioli

Biólogo formado pela Unicamp. Atualmente Mestrado em Ecologia pela UNICAMP

Curriculum lattes http://lattes.cnpq.br/9966926210642260

email: andremourodangioli@gmail.com

exec

Help/função documentacao.rtf

Proposta A proposta_a.docx

Proposta B Elaborar uma função que realize um teste ANOVA entre as variáveis e um teste de confiança de Tukey, retornando ao usuário os valores de cada uma das operações e também um box plot que descreva (com letras 'a,b,c...') os resultados do teste Tukey, indicando se há ou não diferenças entre as amostras. Os dados devem ser colocados como data.frames, e pode ser inserida mais de uma variável preditora, sendo que o usuário deve informar se há ou não interação entre as variáveis.

> Comentários:
>
> Amei a proposta A. Em outras palavras, vai dar trabalho!
>
> Pra não se perder, tente dividir a tarefa em várias funções auxiliares. Uma função que calcula a função L, outra que gera modelos nulos, etc; a sua função final deve chamar essas auxiliares.
>
> A proposta B é um exercício interessante, desde que vc não use os resultados prontos dos pacotes que já existem ;) —- *André Chalom*

```
BesL<- function(data,focal, area, t, nsim, type, ylim, xlim)
{
  df=data
  library(reshape2) ####pacote necessário para ordenação das espécies por
ordem decrescente de abundância
  library(dplyr)    ####pacote necessário para ordenação das espécies por
ordem decrescente de abundância

  B <- df[x!=is.na(df$x),] %.% group_by(sp) %.% summarize(Nrows=n())
####identificar os nomes das espécies do usuário, e a quantidade de
indivíduos por espécie (Nrows). como o número de linhas deve ser igual para
```

```
todas as espécies (preenchimento com NAs para isto) todos os NAs serão
excluídos para a comparação, retornando assim a abundância real das
espécies.
  B <- B[order(B$Nrows,B$sp, decreasing=TRUE),]      ####ordenação das
espécies por abundância
  B<- na.exclude(B)
  P <- df[sp == focal,]  ####criação de um data.frame separado com as
coordenadas da espécie focal(determinado pelo usuário).

  K = rep(NA, each=length(B$sp)) ####criação de um vetor no qual será
colocados os valores de K calculados para cada espécie em relação à focal.

  L= rep(NA, each= length(K)) ####criação de um vetor no qual será colocados
os valores de K calculados para cada espécie em relação à focal.

  MonteCarlox = matrix(NA, nrow=length(df$x), ncol=nsim)  ####vetor no qual
será colocado os valores da reamostragem das coordenadas x do data.frame
principal.
  MonteCarloy = matrix(NA, nrow=length(df$x), ncol=nsim)  ####vetor no qual
será colocado os valores da reamostragem das coordenadas y do data.frame
principal.
  for(i in 1:nsim)  ####'i' reamostragens das coordenadas x e y. O número de
reamostragens será determinada pelo usuário no argumento 'nsim'.
    {
    MonteCarlox[,i]= sample(df$x)
    MonteCarloy[,i]= sample(df$y)
    }
 Mcx<-data.frame(MonteCarlox, df$sp)  ####data.frame criado para as
reamostragens da coordenada x, atribuindo cada valor a uma das espécies.
 Mcy<-data.frame(MonteCarloy, df$sp)  ####data.frame criado para as
reamostragens da coordenada y, atribuindo cada valor a uma das espécies.



##########################################################
##############IntraEspecífica#############################
##########################################################
if(type=='intra')  ####caso o usuário escolha uma análise de agregação
intraespecífica serão feitos os calculos seguintes.
{
 uP<- ((sqrt(((P$x-rep(P$x, each=length(P$x)))^2)+((P$y-rep(P$y,
each=length(P$y)))^2)))<=t)*1  ####uP corresponde aos valores de I. Neste
caso, é calculada a distância euclidiana entre todos os pontos dos
indivíduos focais. As distâncias que forem menor que t serão multiplicadas
por 1
####e utilizadas posteriormente para o calculo de K.
 uPt<- ((sqrt(((P$x-rep(P$x, each=length(P$x)))^2)+((P$y-rep(P$y,
each=length(P$y)))^2)))<=t)*1*(sqrt(((P$x-rep(P$x,
each=length(P$x)))^2)+((P$y-rep(P$y, each=length(P$y)))^2)))####uPt
corresponde a distância entre os indivíduos que seja menor que t. Este vetor
```

```
possui o valor real e não 1
####será utilizado posteriormente para a correção das bordas.

 df1<- df  ####criação de um novo data.frame idêntico ao original para que
seja modificados sem alterações dos dados originais.

 df1$x<-(((df$x+t > xlim)|(df$x-t<0))*df$x)  ####análise condicional dos
valores de x para correção das bordas. serão considerados apenas os pontos
cuja soma com o raio t desejado ultrapasse o limite x do plano cartesiano.
 df1$y<-(((df$y+t>ylim)|(df$y-t<0))*df$y)    ####análise condicional dos
valores de y para correção das bordas. serão considerados apenas os pontos
cuja soma com o raio t desejado ultrapasse o limite y do plano cartesiano.

 ex<- xlim-df1$x  ####determinação da distância entre a borda do plano
cartesiano e a coordenada x dos pontos.
 ey<- ylim-df1$y  ####determinação da distância entre a borda do plano
cartesiano e a coordenada y dos pontos.

 ex[ex==(xlim-0)]<-NA  ####como na análise condicional os pontos que não
estiverem de acordo com a condição foram considerados como 0, as distâncias
que corresponderem a xlim-0 serão desconsideradas.
 ey[ey==(ylim-0)]<-NA  ####como na análise condicional os pontos que não
estiverem de acordo com a condição foram considerados como 0, as distâncias
que corresponderem a ylim-0 serão desconsideradas.

 de<-data.frame(df$x,df$y,df$sp,ex,ey)  ####data.frame com os dados das
distâncias das coordenadas x e y dos pontos que obedecem a condição a cima,
agrupando a identidade das espécies de cada ponto.

 PP<-de[de$df.sp==focal,]  ####seleção dos dados dos pontos da espécie focal
escolhida pelo usuário.
 Ex<-rep(PP$ex, each=length(P$x))  ####repetição dos valores de ex da
espécie focal, para que desta forma haja correspondência entre os valores de
uPt e ex para cada ponto.
 Ey<-rep(PP$ey, each=length(P$y))  ####repetição dos valores de ex da
espécie focal, para que desta forma haja correspondência entre os valores de
uPt e ex para cada ponto.



 uPt[uPt==0] <-NA  ####para que não de erro durante o calculo de w, todos os
uPt que forem igual a 0 (ou seja, aqueles que não cumpriram a condição
inicial) serão transformados a NA.

 wx<- Mod(1-((cos(Ex/uPt)^-1)/pi))  ####calculo do corretor de borda para x.
fórmula extraída de (>>>>>>>>>>>>>>>)
 wy<- Mod(1-((cos(Ey/uPt)^-1)/pi))  ####calculo do corretor de borda para y.
fórmula extraída de (>>>>>>>>>>>>>>>)

 wx[is.na(wx)]<-1  ####todos os valores NA serão convertidos para 1 para que
não alterem o cálculo de K. Os valores de NA são aqueles que não cumprem a
condição estabelecida a cima.
```

```
 wy[is.na(wy)]<-1

 uP[is.na(uP)]<-0  ####como os dados inseridos podem possuir NA, todos estes
valores serão convertidos a 0 para que não influenciem no cálculo do K.

 length(na.exclude(PP$df.x))

 K<- ((length(na.exclude(PP$df.x))^-2)*area)*sum(wx^-1*wy^-1*uP) ####cálculo
de K intraespecífico. Multiplica-se o (total de pontos^-2/area) pela soma da
multiplicação dos valores dos corretores de borda e pelo valor de I(uP)

 L<- (sqrt(K/pi))-t  ####cálculo de L.

 ########MonteCarloIntraEspecífico##############

 McxP<-Mcx[Mcx$df.sp==focal,]  ####selecionados os valores de x das
reamostragens feitas no início apenas da espécie focal selecionada pelo
usuário.
 McyP<-Mcy[Mcy$df.sp==focal,]  ####selecionados os valores de y das
reamostragens feitas no início apenas da espécie focal selecionada pelo
usuário.

 uPMc<- matrix(NA,nrow=length(uP), ncol=nsim)
 uPMct<-uPMc

  for(i in 1:nsim) ####neste ponto serão refeitas o cálculo de uP e uPt
(descrito a cima) i vezes. i corresponde ao número de simulações escolhidas
pelo usuário.
  {
  uPMc[,i]<- ((sqrt(((McxP[,i]-rep(McxP[,i],
each=length(PP$df.x)))^2)+((McyP[,i]-rep(McyP[,i],
each=length(PP$df.y)))^2)))<=t)*1
  uPMct[,i]<- ((sqrt(((McxP[,i]-rep(McxP[,i],
each=length(PP$df.x)))^2)+((McyP[,i]-rep(McyP[,i],
each=length(PP$df.y)))^2)))<=t)*1*((sqrt(((McxP[,i]-rep(McxP[,i],
each=length(PP$df.x)))^2)+((McyP[,i]-rep(McyP[,i],
each=length(PP$df.y)))^2))))
  }

 McxP1<- matrix(nrow=length(McxP[,1]), ncol=nsim)

  for(i in 1:nsim)  ####neste ponto serão refeitas as análises condicionais
para x (descrito em df1$x), i vezes, ou seja, serão feitas análises
condicionais para todas as reamostragens feitas.
  {
  McxP1[,i]<-((McxP[,i]+t>xlim)|(McxP[,i]-t<0))*McxP[,i]
  }
 McyP1<- matrix(nrow=length(McyP[,1]), ncol=nsim)

  for(i in 1:nsim) ####semelhante ao que foi feito para McxP1, mas para os
```

```
valores de y.
  {
  McyP1[,i]<-((McyP[,i]+t>ylim)|(McyP[,i]-t<0))*McyP[,i]
  }

 Mcex<- matrix(nrow= length(McxP[,1]), ncol=nsim)
 Mcey<- matrix(nrow= length(McxP[,1]), ncol=nsim)

  for(i in 1:nsim)  ####calculado os valores de ex e ey(descritos a cima)
para cada uma das simulações feitas.
  {
  Mcex[,i]<- xlim-McxP1[,i]
  Mcey[,i]<- ylim-McxP1[,i]
  }

 Mcex[Mcex==(xlim-0)]<-NA  ####como na análise condicional os pontos que não
estiverem de acordo com a condição foram considerados como 0, as distâncias
que corresponderem a xlim-0 serão desconsideradas.

 Mcey[Mcey==(ylim-0)]<-NA

 Mcex1<- matrix(nrow= length(uP), ncol=nsim)
 Mcey1<- matrix(nrow= length(uP), ncol=nsim)

  for(i in 1:nsim)  ####repetição dos valores de ex e ey da espécie focal,
para que desta forma haja correspondência entre os valores de uPt e ex e ey
para cada ponto. Será feito isto para cada uma das simulações.
  {
  Mcex1[,i]<-rep(Mcex[,i], each=length(P$x))
  Mcey1[,i]<-rep(Mcey[,i], each=length(P$x))
  }

 Mcwx<- matrix(nrow= length(uP), ncol=nsim)
 Mcwy<- matrix(nrow= length(uP), ncol=nsim)

  uPMct[uPMct==0] <-NA  ####para que não de erro durante o calculo de w,
todos os uPt que forem igual a 0 (ou seja, aqueles que não cumpriram a
condição inicial) serão transformados a NA.

  for(i in 1:nsim)  ####calculo dos valores de wx e wy (corretores de borda)
para cada uma das simulações.
  {
  Mcwx[,i]<- Mod(1-((cos(Mcex1[,i]/uPMct[,i])^-1)/pi))
  Mcwy[,i]<- Mod(1-((cos(Mcey[,i]/uPMct[,i])^-1)/pi))
  }

 Mcwx[is.na(Mcwx)]<-1 ####todos os valores NA serão convertidos para 1 para
que não alterem o cálculo de K. Os valores de NA são aqueles que não cumprem
a condição estabelecida a cima.
 Mcwy[is.na(Mcwy)]<-1
```

```
 uPMc[is.na(uPMc)]<-0 ####semelhante ao que foi feito a cima, os valores de
NA serão convertidos a 0 para que não influenciem no cálculo de K.

 McK<- rep(NA, each=nsim)

 for(i in 1:nsim)  ####serão calculados os valores de K (conforme descrito a
cima) para cada uma das simulações.
  {
  McK[i]<-
((length(na.exclude(PP$df.x))^-2)*area)*sum(Mcwx[,i]^-1*Mcwy[,i]^-1*uPMc[,i]
)
  }

 McL<- rep(NA, each=nsim)

 for(i in 1:nsim)  ####serão calculados os valores de L para cada um dos
valores de K das simulações (agrupados em McK).
 {
 McL[i]<- (sqrt(McK[i]/pi))-t
 }

 p.unima=(sum(L>=McL))/(length(McL))  ####será calculado o valor de p para
os dados do usuário. este valor é dado pelo número de simulações que sejam
menor ou igual ao valor de L observado sobre
                                      ####o total de número de simulações.
Caso p seja <=0.01 pode-se dizer que os indivíduos estão agrupados. Caso
p>=0.99 os indivíduos estão distribuídos
                                      ####uniformemente no espaço. Caso p
seja diferente dos dois casos anteriores os indivíduos estão distribuídos
aleatoriamente no espaço.

 Focal<- focal  ####os vetores serão renomeados para que tenham nomes mais
claros durante a apresentação do data.frame(sumario).
 Specie<- P$sp[1]
 L.value<- L
 P.value<- p.unima

 sumario<- data.frame(Focal, Specie, L.value, P.value)  ####os dados serão
agrupados em um data.frame com o nome da espécie focal, o nome da espécie
que foi comaparada (no caso a focal), o valor de L observado e o valor de p
calculado.

}

#########################################################
#############InterEspecífico#############################
#########################################################
if (type == 'inter')  ####caso o usuário tenha escolhido a opção 'inter'
será rodado o seguinte script.
{
```

```
uP<- matrix(NA,nrow=length(P$x-rep(df[sp == B$sp[1],]$x,
each=length(P$x))),ncol=length(na.exclude(B$sp))) ####criação de uma matriz
para armazenar os dados(valores de I para interações com cada espécie)
obtidos pela função for
uPt<- uP ####criação de uma matriz para armazenar as distâncias entre dois
pontos que distem um do outro <=t.

for(i in 1:(length(na.exclude(B$sp))))
{
uP[,i]<-((sqrt(((P$x-rep(df[sp == B$sp[i],]$x, each=length(P$x)))^2)+
####uP corresponde ao valores de I para cada espécie. I possui valor 1 caso
os pontos distem <=t e 0 caso distem mais que t. Cada coluna desta matriz
corresponde aos valores de I da relação entre a espécie focal e a espécie i
((P$y-rep(df[sp == B$sp[i],]$y, each=length(P$y)))^2)))<=t)*1
uPt[,i]<-((sqrt(((P$x-rep(df[sp == B$sp[i],]$x, each=length(P$x)))^2)+
####uPt corresponde às distâncias entre dois pontos cuja distância seja
menor que t. cada coluna desta matriz corresponde às distâncias entre os
pontos da espécie focal(escolhida pelo usuário) e os pontos das demais
espécies
((P$y-rep(df[sp == B$sp[i],]$y, each=length(P$y)))^2)))<=t)*
(sqrt(((P$x-rep(df[sp == B$sp[i],]$x, each=length(P$x)))^2)+
((P$y-rep(df[sp == B$sp[i],]$y, each=length(P$y)))^2)))
}

df1<-df  ####criação de um novo data.frame para que possa ser modificado sem
alterar os dados do data.frame original.

df1$x<-(((df$x+t > xlim)|(df$x-t<0))*df$x) ####análise condicional dos
valores de x para correção das bordas. serão considerados apenas os pontos
cuja soma com o raio t desejado ultrapasse o limite x do plano cartesiano.
df1$y<-(((df$y+t>ylim)|(df$y-t<0))*df$y)

ex<- xlim-df1$x  ####determinação da distância entre a borda do plano
cartesiano e a coordenada x dos pontos.
ey<- ylim-df1$y  ####determinação da distância entre a borda do plano
cartesiano e a coordenada y dos pontos.

ex[ex==(xlim-0)]<-NA  ####como na análise condicional os pontos que não
estiverem de acordo com a condição foram considerados como 0, as distâncias
que corresponderem a xlim-0 serão desconsideradas.
ey[ey==(ylim-0)]<-NA  ####idem ao de cima, porém para os pontos y.

de<-data.frame(df$x,df$y,df$sp,ex,ey)  ####data.frame com os dados das
distâncias das coordenadas x e y dos pontos que obedecem a condição a cima,
agrupando a identidade das espécies de cada ponto.

PP<-de[de$df.sp==focal,] ####seleção dos dados dos pontos da espécie focal
escolhida pelo usuário. como os pontos que importam para esta medida são
apenas os pontos da espécie focal, é selecionado apenas os dados referentes
à espécie focal
Ex<-rep(PP$ex, each=length(P$x))  ####repetição dos valores de ex da espécie
```

```
focal, para que desta forma haja correspondência entre os valores de uPt e
ex e ey para cada ponto. Será feito isto para cada uma das simulações.
Ey<-rep(PP$ey, each=length(P$y))  ####repetição dos valores de ey da espécie
focal, para que desta forma haja correspondência entre os valores de uPt e
ex e ey para cada ponto. Será feito isto para cada uma das simulações.

wx<- matrix(0,ncol=length(na.exclude(B$sp)),nrow=length(rep(df[sp ==
B$sp[1],]$x, each=length(df[sp == B$sp[1],]$x))))  ####criação de matriz
para armazenamento dos valores do corretor. Número de linhas desta matriz
corresponde ao total de valores de distâncias (uPt) entre a espécie focal e
as demais espécies. Como todas as espécies devem possuir o mesmo número de
linhas (preenchimento com NA, caso não tenham sido
####amostrados o mesmo número de indivíduos, pode-se escolher qualquer
espécie para definir o número de linhas.
wy<- wx

 uPt[uPt==0] <-NA  ####para que não de erro durante o calculo de w, todos os
uPt que forem igual a 0 (ou seja, aqueles que não cumpriram a condição
inicial) serão transformados a NA.

for(i in 1:length(na.exclude(B$sp)))
{
wx[,i]<- Mod(1-((cos(Ex)/uPt[,i]^-1)/pi))  ####corretor para cada espécie
para o eixo x.
wy[,i]<- Mod(1-((cos(Ey/uPt[,i])^-1)/pi))  ####idem ao de cima para o eixo
y.
}

wx[is.na(wx)]<-1  ####todos os valores NA serão convertidos para 1 para que
não alterem o cálculo de K. Os valores de NA são aqueles que não cumprem a
condição estabelecida a cima.
wy[is.na(wy)]<-1

uP[is.na(uP)]<-0  ####como os dados inseridos podem possuir NA, todos estes
valores serão convertidos a 0 para que não influenciem no cálculo do K.

for(i in 1:length(B$sp))
{
K[i]=area/((length(na.exclude(P$x))*(length(na.exclude(df[sp==B$sp[i],]$x)))
))*  ####calculo dos valores de K, fórmula extraída de (>>>>>>>>>). Cada
número corresponde ao valor de K entre a interação da espécie focal com a
espécie i.
sum(uP[,i]/(wx[,i]*wy[,i]))
}

for(i in 1: length(K))
    {
    L[i] = (sqrt(K[i]/pi))-t  ####calculo do valor de L.
    }
```

```
###############MonteCarloInterEspecífico###################

 McxP<-Mcx[Mcx$df.sp==focal,]  ####selecionados os valores de x da
reamostragem da espécie focal.
 McyP<-Mcy[Mcy$df.sp==focal,]  ####idem ao de cima mas para os valores de y.

uPMc <- array(dim=c(length(uP[,1]),nsim,length(B$sp)))  ####criação de um
array para o armazenamento dos valores de uP de cada amostragem. Os valores
de uP serão colocados nas linhas, os valores de uP de cada reamostragem será
colocado nas colunas, que serão organizados por espécies (terceira dimensão
do array).
uPMct<- uPMc  ####idem ao de cima mas para os valores de uPt de cada
reamostragem para todas as espécies.

for(i in 1:nsim)
for(n in 1:length(B$sp))
{
uPMc[,i,n]<- ((sqrt(((McxP[,i]-rep(Mcx[Mcx$df.sp==B$sp[n],][,i],
each=length(PP$df.x)))^2)+  ####calculo do uP de cada espécie (colocados na
terceira dimensão do array), para o número de simulações escolhidas pelo
usuário (colunas do array). serão armazenados com o nome uPMc(referente ao
método de montecarlo).
((McyP[,i]-rep(Mcx[Mcx$df.sp==B$sp[n],][,i],
each=length(PP$df.x)))^2)))<=t)*1
uPMct[,i,n]<-((sqrt(((McxP[,i]-rep(Mcx[Mcx$df.sp==B$sp[n],][,i],
each=length(PP$df.x)))^2)+  ####idem ao de cima mas para os valores de uPt.
((McyP[,i]-rep(Mcx[Mcx$df.sp==B$sp[n],][,i],
each=length(PP$df.x)))^2)))<=t)*
(sqrt(((McxP[,i]-rep(Mcx[Mcx$df.sp==B$sp[n],][,i],
each=length(PP$df.x)))^2)+
((McyP[,i]-rep(Mcx[Mcx$df.sp==B$sp[n],][,i], each=length(PP$df.x)))^2)))
}

McxP1<- matrix(nrow=length(McxP[,1]), ncol=nsim)  ####neste ponto será
testada a condicionalidade para cada valor de x reamostrado para a espécie
focal. a condicionalidade é de que a distância entre o ponto + t seja maior
que xlim ou que a distância do ponto menos t seja menor que 0.

  for(i in 1:nsim)
  {
  McxP1[,i]<-((McxP[,i]+t>xlim)|(McxP[,i]-t<0))*McxP[,i]
  }
 McyP1<- matrix(nrow=length(McyP[,1]), ncol=nsim) ####idem ao que foi feito
acima mas para os valores de y da espécie focal.

  for(i in 1:nsim)
  {
  McyP1[,i]<-((McyP[,i]+t>ylim)|(McyP[,i]-t<0))*McyP[,i]
  }

 Mcex<- matrix(nrow= length(McxP[,1]), ncol=nsim)  ####neste ponto será
```

```
calculado o valor de e da função corretora. e corresponde a distância do
ponto a ser analisado até o limite (borda) do plano cartesiano.
                                    ####este valor será calculado apenas para
os pontos que cumpriram a condição acima. aqui será calculado para x.
 Mcey<- matrix(nrow= length(McxP[,1]), ncol=nsim)  ####idem ao de cima mas
calculado para os valores de y.

  for(i in 1:nsim)
  {
  Mcex[,i]<- xlim-McxP1[,i]
  Mcey[,i]<- ylim-McxP1[,i]
  }

 Mcex[Mcex==(xlim-0)]<-NA  ####como na análise condicional os pontos que não
estiverem de acordo com a condição foram considerados como 0, as distâncias
que corresponderem a xlim-0 serão desconsideradas.
 Mcey[Mcey==(ylim-0)]<-NA  ####idem ao de cima mas para os valores de y.

 Mcex1<- matrix(nrow= length(uP[,1]), ncol=nsim)  ####repetição dos valores
de ex e ey da espécie focal, para que desta forma haja correspondência entre
os valores de uPt e ex e ey para cada ponto. Será feito isto para cada uma
das simulações.
 Mcey1<- matrix(nrow= length(uP[,1]), ncol=nsim)

  for(i in 1:nsim)
  {
  Mcex1[,i]<-rep(Mcex[,i], each=length(P$x))
  Mcey1[,i]<-rep(Mcey[,i], each=length(P$x))
  }

 Mcwx<- array(dim=c(length(uPMc[,1,1]),nsim,length(B$sp)))  ####será
calculado os valores do corretor (w) para cada uma das espécies (n) para
cada uma das simulações (i). para os valores de x.
 Mcwy<- array(dim=c(length(uPMc[,1,1]),nsim,length(B$sp)))  ####será
calculado os valores do corretor (w) para cada uma das espécies (n) para
cada uma das simulações (i). para os valores de y.

 uPMct[uPMct==0] <-NA  ####para que não de erro durante o calculo de w,
todos os uPt que forem igual a 0 (ou seja, aqueles que não cumpriram a
condição inicial) serão transformados a NA.

for(n in 1:length(B$sp))
for(i in 1:nsim)
{
Mcwx[,i,n]<- Mod(1-((cos(Mcex1[,i]/uPMct[,i,n])^-1)/pi))
Mcwy[,i,n]<- Mod(1-((cos(Mcey[,i]/uPMct[,i,n])^-1)/pi))
}

Mcwx[is.na(Mcwx)]<-1  ####todos os valores NA serão convertidos para 1 para
que não alterem o cálculo de K. Os valores de NA são aqueles que não cumprem
```

```
a condição estabelecida a cima.
Mcwy[is.na(Mcwy)]<-1

uPMc[is.na(uPMc)]<-0  ####como os dados inseridos podem possuir NA, todos
estes valores serão convertidos a 0 para que não influenciem no cálculo do
K.

McK<- matrix(NA, nrow=nsim, ncol=length(B$sp)) ####serão calculados os
valores de K conforme descrito para a seção acima (InterEspecífico). os
valores serão calculados para cada espécie(n) e para cada simulação(i).

for(n in 1:length(B$sp))
for(i in 1:nsim)
{
McK[i,n]<-
area/((length(na.exclude(P$x))*(length(na.exclude(df[sp==B$sp[n],]$x)))))*
sum(uPMc[,i,n]/(Mcwx[,i,n]*Mcwy[,i,n]))
}

McL<-matrix(NA, nrow=nsim, ncol=length(B$sp)) ####serão calculados os
valores de L para cada espécie (n) para cada simulação (i).

for(n in 1:length(B$sp))
for(i in 1: nsim)
    {
    McL[i,n] = (sqrt(McK[i,n]/pi))-t
    }

p.unima<- rep(NA, each=length(na.exclude(B$sp)))
for(n in 1:length(B$sp))  ####será calculado o valor de p para cada uma das
espécies (n) conforme descrito na seção anterior.
{
p.unima[n]=(sum(L[n]>=McL[,n]))/(length(McL[,n]))
}

Specie<-B$sp  ####os vetores serão renomeados para melhor apresentação dos
dados.
L.value<-L
P.value<-p.unima
Focal<-focal

sumario<-data.frame(Focal, Specie, L.value, P.value)
sumario[sumario$Specie==focal,]<-NA  ####como o calculo para interespecífico
é diferente do calculo para intraespecífico, aqueles dados que
corresponderem à mesma espécie (focal e comparada) serão excluídos.
sumario<-na.exclude(sumario)
}
return(sumario)
}
```

```
BesL{base}                                              R Documentation
```

```
L-funtion

Description:
Estimates de value of the linearized version of the Ripley's K-funtion
(Ripley, 1977), as proporsed by Bessag (1977), for a group of points in a
cartesian plane. Also, will test the significance of the L-value (if the
points pattern is aggregated, random or regular), by Monte Carlo
simulations.
Usage:

     BesL(data,focal, area, t, nsim, type='intra', ylim, xlim)

Arguments:
data        the name of the data.frame to be analysed.
focal       the name of the focal specie which will be compared with the
others.
area        the total area of the cartesian plane.
t       the radius, which will be centered in the points of the focal
specie; points         within this radius will be counted.
nsim        the number of Monte Carlo simulations. it is important for the
determination of the p-value.
type        the type of analysis to be made. it can be 'intra', for
intraspecific analysis or         'inter' for interspecific analysis.
ylim        the y limit of the cartesian plane.
xlim        the x limit of the cartesian plane.

Details:
The data to be analysed must be organized as a data.frame. It must have a
column named 'x' with the x-coordinates of each point; it must have a column
named 'y' with the y-coordinate of each point, and must have a column named
'sp' with the name of the specie corresponding to the x y coordinates.
It's important that all species have the same number of xy coordinates. For
this, species with less observed xy coordinates must have it's x and y
values filled with NA.
Also, the user must have installed the packages 'reshape2' and 'dplyr'.
Ripley's K-function is based on the variance (second-order analysis) of all
point-to-point distances in a two dimensional space, and gives a description
of the spatial structure of the points. The intraspecific analysis is
defined by the expected number of individuals of the same specie as the
focal within a radius t, centred in an arbitrary individual. While the
interspecific analysis is defined by the expected number of individuals of a
different specie of the focal within a radius t, centred in an arbitrary
individual.
For the intraspecific K, will be used the following formula (Haase, 1995):
K(t)= n^-2*area*sum(wij^-1*It(uij))
Where 'n' is the number of individuals in the analysed plot, 'area' is the
total area of the plot, 'uij' is the distance between points i and j, and
'It' have a binary value (0,1), it's value is 1 if 'uij' <= t and it's value
is 0 if 'uij' > t; 'wij' is a correcting factor for edge effects, it has the
value of 1 if the circle centred at i and passing through the point j (with
```

radius of t) is completly inside the plot area, if part of the circle falls outside the plot area the correctos used is, as proposed by Getis & Franklin (1987):
wij= 1-cos^-1(ei/uij)/pi
Where ei is the distance between the point e and the plot boundary.
For the interspecific K, will be used the following formula (Wiegand & Moloney, 2004):
K12(t)= (area/(n1*n2))*sum(It(uij)/wij)
Where 'n1' is the number of individuals of the focal specie in the analysed plot, and 'n2' is the number of the individuals of other specie in the analysed plot. The 'wij' used is the same as the one proposed by Getis & Franklin (1987) explained above.
To determine the L-value, the formula that will be used is and adaptation of the one proposed by Bessag (1977), extracted from Haase(1995):
L(t)=sqrt(K(t)/pi)-t
If L-value is positive, and above the upper limit of the confidence envelope, the clumped ditribution can be assumed, if L-value is negative the patter can be discribed as dispersed or regular. Else the distribution is alleatory.
For the determination of the confidence levels, it will be realized Monte Carlo simulations, sampling de x and y values, for the creation of alleatory distributions, and comparing the L-values with the L-values simulated, extracting a p-value. If p-value is higher than 0.97 the distribution of the points can be considered aggregated, if p-value is lower than 0.03 the distribution of the points can be considered dispersed or regular.

Value:
A data.frame is returned, with a column 'Focal' with the name of the focal specie chosed by the user; a column 'Specie' with the name of the specie compared with the focal; a column 'L-value' with the L-value of the observed pattern and a column 'P-value' with the value of p obtained with the comparison with the L-value of the Monte Carlo simulations.

Warning:
For the correct estimation, none x and y value must fall up the xlim or the ylim. Also, two individuals must not fall on the same point (same x and y values). Also, t must obey the following condition: t < (area/2)1/2 (Dixon, 2006).
Author(s):
André Mouro D'Angioli, e-mail andremourodangioli@gmail.com.

References:
Besag, J. (1977). Contribution to the discussion of Dr. Ripley’s paper. Journal of the Royal Statistical Society, Series B 39: 193-195.
Dixon, P. M. (2006). Ripley's K function. Encyclopedia of environmetrics.
Getis, A., & Franklin, J. (1987). Second-order neighborhood analysis of mapped point patterns. Ecology, 68(3), 473-477.
Haase, P. (1995). Spatial pattern analysis in ecology based on Ripley's K-function: Introduction and methods of edge correction. Journal of Vegetation Science, 6(4), 575-582.
Ripley, B. D. (1977). Modelling spatial patterns. Journal of the Royal

```
Statistical Society. Series B (Methodological), 172-212.
Wiegand, T., & A Moloney, K. (2004). Rings, circles, and null-models for
point pattern analysis in ecology. Oikos, 104(2), 209-229.

Examples:
####creating a perfect uniform distribution####
x<- rep(1:10, each=10)
y<- rep(1:10, times=10)
sp<-'a'
plot(x~y)  ####visualizing the distributions
data=data.frame(x,y,sp)

BesL(data, 'a', 100, 1, 1000, 'intra', ylim=10.1, xlim=10.1)

####creating a perfect uniform distribution with NAs####
x1<- c(rep(1:10, each=10),NA,NA)
y1<- c(rep(1:10, times=10),NA,NA)
sp<-'a'
plot(x1~y1)
data1=data.frame(x1,y1,sp)
BesL(data1, 'a', 100, 1, 1000, 'intra', ylim=10.1, xlim=10.1) ####notice
that the L.values are the same of data and data1

########################################

x<- c(runif(10, 1,4),runif(10, 6,9))
y<- c(runif(10,1,4), runif(10, 6,9))
sp<-rep(c('a','b'),each=10)

data= data.frame(x,y,sp)

plot(data[data$sp=='a',]$y~data[data$sp=='a',]$x, xlim=c(0,10.1),
ylim=c(0,10.1))
par(new=T)
plot(data[data$sp=='b',]$y~data[data$sp=='b',]$x, col='red',xlim=c(0,10.1),
ylim=c(0,10.1) )

BesL(data, 'a', 100, 1, 1000, 'intra', ylim=10, xlim=10) ####notice that the
specie 'a' is clumped
BesL(data, 'a', 100, 1, 1000, 'inter', ylim=10, xlim=10) ####notice that the
specie 'a' is dispersed in relation to the specie 'b'

#####################################
#####Random Distribution##############

x<- runif(50,1,10)
y<- runif(50,1,10)
sp<- rep(c('a','b','c','d','e'), each=10)
distribution<-data.frame(x,y,sp)
```

```
BesL(distribution, 'a', 100, 1, 100, 'intra', ylim=10.1, xlim=10.1)
BesL(distribution, 'a', 100, 1, 100, 'inter', ylim=10.1, xlim=10.1)
```