———————-Trabalho final: ———————-

→ Propostas iniciais (20.03.2015):

Plano A - Função “predação na matriz”

Link: funcao_predacao_na_matriz.docx

Plano B - Função “maior distância das bordas”:

Link: funcao_maior_distancia_das_bordas.docx

```
Comentários dessa postagem:
```

Camila, poste as propostas diretamente aqui no wiki. O plano A está um pouco confuso. Poste o que você pensou para cada passo da sua função aqui no wiki. O plano B é uma ideia interessante para o trabalho final. Mas é preciso definir a entrada no help da sua função. — Juliana Vendrami 2015/03/24 14:52

Comentários Vitor Rios

Reescreva suas propostas diretamente no wiki. A princípio a 2 me parece mais clara, mas ambas precisam ser trabalhadas. Detalhe melhor a entrada de cada função e a saída, e coloque passo a passo o que a função vai fazer

→ Postagem aceitando os comentários acima (30.03.2015):

Plano A - Função “predação na matriz” A função irá ler um dataframe com uma coluna contendo a variável qualitativa e com linhas referentes a níveis de uma variável quantitativa, e irá gerar um dataframe com um único valor em porcentagem da variável qualitativa em função dos níveis da variável quantitativa. Feito isso, a função irá gerar um gráfico do tipo “plot” da variável qualitativa em função da variável quantitativa. Cada gráfico gerado está associado a uma circunstância, por isso, a função irá mostrar lado a lado o resultado gráfico da variável qualitativa em função da variável quantitativa de cada circunstância, ou seja, a função irá gerar mais de um gráfico. Explicando a função: A função irá tratar valores binários (sim e não) de uma variável qualitativa (por exemplo, predação) e irá transformar esses valores em um único valor de porcentagem (no caso, predação). Cada valor único de porcentagem está relacionado a um nível do fator da variável quantitativa (por exemplo, distância da borda da mata). Feito isso, a função irá gerar um gráfico da variável qualitativa em função da variável quantitativa. Cada gráfico gerado está associado a um contexto (por exemplo, cobertura florestal na paisagem), por isso, a função irá gerar 4 gráficos, um para cada contexto, no caso quantidade de cobertura florestal na paisagem.

Plano B - Função “maior distância das bordas”: A função irá ler os dados de uma matriz de “0” e “1” e irá reconhecer como “unidade” as combinações entre colunas e linhas que possuam mais de um “1” agregados. Reconhecidas as “unidades”, a função irá ler as colunas e linhas em busca da combinação de uma linha e uma coluna que irá gerar maior quantidade de “1” agregados. Encontrada a combinação de uma linha e uma coluna, a função dará a posição de interseção entre a linha e a coluna. Explicando a função: A matriz a ser inserida é uma matriz de uma paisagem. Os valores de “1” e “0” são células de habitat e não habitat respectivamente. A agregação de “1” que chamamos de “unidade” representa um fragmento de habitat. O ponto de interseção entre uma coluna e uma linha com maior número de “1” agregados seria o ponto mais distante da borda do fragmento.

→ Postagem “escolhendo o desafio e explorando a função” (30.03.2015):

Proposta escolhida para o desafio: Plano B.

Separando a função passo-a-passo (no “mundo das ideias” pois o código ainda não está pronto):

Eu irei usar a função landscapeGEN() (construída pelo aluno Bruno Travassos no ano 2014) que gera a partir de uma matriz binária, um dataframe com informações de identidade de fragmentos de uma paisagem (matriz binária). Eu irei usar esse dataframe para fazer as minhas análises de distância da borda.

Passo 1: A função irá identificar para cada célula do fragmento (ou seja, células de habitat do fragmento X), se algum dos seus vizinhos é de não habitat. Caso ele seja de não habitat, essa célula vizinha será definida como uma célula de borda do fragmento em análise. A função irá fazer isso para todas as células de cada fragmento. O output disso é um conjunto de fragmentos e suas respectivas células de borda.

Passo 2: Definidos os fragmentos e suas respectivas bordas, a função irá calcular o somatório das distâncias de uma célula do fragmento (célula focal) em relação a todas as células de borda deste fragmento. Esse valor referente ao somatório das distâncias será calculado para todas as células do fragmento, célula por célula. Ou seja, o output desse passo 2 são valores de distância de todas as bordas de um fragmento para cada célula deste fragmento. Cada célula do fragmento terá seu valor, a célula com o maior valor de distância será a célula do fragmento mais distante da borda.

Passo 3: Definido qual a célula com maior valor de distância da borda por fragmento, e definido qual esse valor, a função irá calcular o valor médio dessa distância da célula mais distante em relação as células do fragmento. Ou seja, o output desse passo 3 é um valor médio de distância da borda para cada fragmento.

Passo 4: A função irá retornar para o usuário um gráfico que identifica os fragmentos com suas respectivas células de maior distância da borda. Além disso, a função irá retornar na forma de lista os valores médios de distância da célula mais distante em relação borda do fragmento.

→ Função (Proposta Plano B)

```
distEdge=function(DATA=NA, NEI8=F, TAMX=30, TAMY=30) # definidor da função,
note que alguns os parâmetros são necessários para chamar a funçãoo interna
landscapeGen()
{
  calcDist=function(i,j)# função de para calcular distâncias entre dois
pontos em um plano cartesiano
  {
    dX=dataframe.cell$cellX[i]-dataframe.cell$cellX[j] # distancia em x.
cateto1
    dY=dataframe.cell$cellY[i]-dataframe.cell$cellY[j] # distancia em y.
cateto2
    dist=sqrt(dX*dX+dY*dY) # calculo de pitágoras
    return(dist) #retorna a distancia entre os dois pontos
  }
source("Landscape function.R")  # chama a função landscapeGen
dataframe.cell=landscapeGen(FRAG=0.01, METRICS=F,NEI8 = NEI8, TAMX=TAMX,
```

```
TAMY=TAMY) #roda a função landscapeGen para gerar a paisagem e/ou
identificar os fragmentos (a função vai definir o objeto dataframe.cells com
a informação de todas as células dessa paisagem), os parâmetros estão
definidos assim para que o usuário da minha função possa definir
propriedades dessa função

mDists=NA #cria objeto mDist
isCentral=F #cria objeto isCentral
borda=NA #cria objeto borda
dataframe.cell=cbind(dataframe.cell,mDists) #define mDist como uma coluna em
dataframe.cells
dataframe.cell=cbind(dataframe.cell,isCentral) #define isCentral como uma
coluna em dataframe.cells
dataframe.cell=cbind(dataframe.cell,borda) #define borda como uma coluna em
dataframe.cells
dataframe.cell$NFRAG=as.numeric(dataframe.cell$NFRAG) #redefine a coluna
NFRAG (identidade de fragmento das células) de dataframe.cells como numérico

for (p in 1:max(dataframe.cell$NFRAG,na.rm=T)) #rotina para selecionar os
fragmentos. note que o valor máximo da coluna NFRAG é justamente o número de
fragmentos na paisagem
{
  Frag_focal= dataframe.cell[which(dataframe.cell[,5]==p),] #define um novo
dataframe somente com as células do fragmento focal
Edge_focal=dataframe.cell[which(dataframe.cell[,5]==paste("Ed",p,sep="")),]
# cria um dataframe somente com as células de matriz que estão na borda do
fragmento total (por enquanto esse dataframe está vazio)
  for (c in 1:length(Frag_focal$X1.AREA)) # rotina para selecionar as
células focais dentro do fragmento e definir quem são as bordas conectadas
nessa célula
  {
   id=Frag_focal$X1.AREA[c] # define o identificador da célula c no
dataframe.cells como id
if(NEI8==F){vizinho=dataframe.cell[which(calcDist(id,dataframe.cell$X1.AREA[
])<=1),] } # se regra de vizinhança for 4, cria um dataframe "vizinho" com a
célula atual e seus 4 vizinhos mais próximos
else{vizinho=dataframe.cell[which(calcDist(id,dataframe.cell$X1.AREA[])<2),]
} # caso contrário, cria dataframe "vizinho" com célula atual e os 8
vizinhos mais próximos
   vizinho$borda[which(vizinho$isVoid==T)]=paste("Ed",p,sep="") # aqueles
vizinhos que forem de matriz receberão a informação de que eles são células
de borda desse fragmento.
   dataframe.cell[vizinho$X1.AREA,]=vizinho  # copia as informações
modificadas em no dataframe "vizinho" para o dataframe.cell
  }
Edge_focal=dataframe.cell[which(dataframe.cell$borda==paste("Ed",p,sep="")),
] #coleta as informações de todas as células que são de borda do fragmento
focal

  for (c in 1:length (Frag_focal$X1.AREA)) #rotina que calcula quem é a
célula central do fragmento focal
```

```
  {
    id=Frag_focal$X1.AREA[c] ## define o identificador da célula c no
dataframe.cells como id
    Frag_focal$mDists[c] =
min(calcDist(id,dataframe.cell$X1.AREA[which(dataframe.cell$isVoid==T)]))  #
calcula a distância da célula focal para todas as células de matriz e guarda
a menor distância
  }
  Frag_focal$isCentral[which(Frag_focal$mDists==max(Frag_focal$mDists))]=T #
define como célula central aquela que tem a maior menor distância da matriz
  dataframe.cell[Frag_focal$X1.AREA,]=Frag_focal # transporta as informações
do dataframe "Frag_focal" para o dataframe.cells

}

edges=dataframe.cell[grep("Ed",dataframe.cell$borda),] #cria um dataframe
com todas as bordas da paisagem independentemente do fragmento


for(i in 1: length(edges$X1.AREA)) # rotina para desenhar a linha em todas
as células de matriz que margeiam na paisagem
{
  id=edges$X1.AREA
rect(col="pink",border=1,xright=dataframe.cell$cellX[id],xleft=dataframe.cel
l$cellX[id]-1,ytop=dataframe.cell$cellY[id],ybottom=dataframe.cell$cellY[id]
-1)

}

for(i in 1: length(dataframe.cell$X1.AREA)) # rotina para pintar de azul as
células centrais do fragmento. note que mais de uma célula pode ser a
central do fragmento desde que a métrica (maior menor distancia) seja
repetida entra as células.
{
if(dataframe.cell$isCentral[i]==T){rect(col="blue",border=NA,xright=datafram
e.cell$cellX[i],xleft=dataframe.cell$cellX[i]-1,ytop=dataframe.cell$cellY[i]
,ybottom=dataframe.cell$cellY[i]-1)}
}

}
```

Help da Função distEdge

```
distEdge                package:nenhum                R Documentation

Identifica qual/quais as células centrais de cada fragmento em uma paisagem.


Description:
Identifica qual a célula central de cada fragmento de uma paisagem (matriz
```

```
binária) através de um conjunto de cálculos que relacionam a posição de cada
célula do fragmento em relação às bordas deste.

Usage:

distEdge (DATA, NEI8, TAMX, TAMY)


Arguments:
DATA        Define se haverá entrada de uma matriz binária ou se essa será
gerada pela função “Landscape function.R” (quando a função estiver em NA).

NEI8        Define qual será o critério de vizinhos a ser usado, caso seja
inserido TRUE a regra de vizinhos será as 8 direções cardinais, caso FALSE,
a regra de vizinhos será as 4 direções cardinais.

TAMX        Número inteiro para definir o número de células do eixo X da
matriz binária quando DATA=NA.
TAMY              Número inteiro para definir o número de células do eixo X
da matriz binária quando DATA=NA.
Details:
A ou as células centrais de cada fragmento na matriz binária são
identificadas através de um algoritmo que relaciona a posição de cada célula
com as distâncias desta para cada célula de borda do fragmento. A regra de
vizinhos é importante para a definição de quais são as células de borda do
fragmento.

Author:
Camila Celestino Hohlenwerger
cch.camila@gmail.com

Examples:
distancia<-distEdge(DATA=NA, NEIG8=F, TAMX=40, TAMY=40)
```

Arquivos: Função: distedge.r Help: distedge_help.txt Função “Landscape function.R” pertencente a Bruno Travassos de Britto: landscape_function.r