Função do Pietro: matrixmap

Algumas mudanças foram feitas em relação ao “Plano A aprimorado”. Primeiramente, o argumento “w” e o argumento “shape” foram substituídos por “wx” e “wy”, deixando o código muito mais enxuto sem perder plasticidade ao determinar o formato da escala. Ademais, o argumento “explicit” foi adicionado para melhorar a visualização nos mapas plotados, caso o usuário julgue adequado.

Arquivos

Help da função matrixmap

Código da função matrixmap

Help da função

```
matrixmap                package:unknown                R Documentation

Matrizes e mapas de densidade para dois fatores de organismos

Description:

    A partir de dados de ocorrência (coordenadas métricas) de dois tipos
de organismos (e.g., duas espécies, machos e fêmeas, dois morfotipos), cria
matrizes de densidade (ou razão entre densidades) e mapas coloridos
semelhantes a heatmaps.

Usage:
    matrixmap(data,wx,wy,output=3,plot=TRUE,explicit=FALSE)

Arguments:
data        Data frame de ocorrência dos indivíduos, contendo três colunas
        (coordenada x, coordenada y e fator do indivíduo).

wx      Valor numérico (maior que 2), determina quantos segmentos do
        eixo x serão gerados.
wy      Valor numérico (maior que 2), que determina quantos segmentos
        do eixo y serão gerados.
output      1, 2 ou 3. Determina qual o resultado dado, se =1 mostra apenas
        a densidade para indivíduos de primeiro fator, se =2 mostra
        apenas a densidade para indivíduos de segundo fator. Se =3
        (padrão) mostra a razão das densidades entre indivíduos de
        primeiro fator por indivíduos de segundo fator.

plot        Valor lógico. Se =FALSE, retorna apenas a matriz de densidade;
        se =TRUE (padrão), produz o mapa de densidade com cores
```

```
        correspondentes (e com barra de cores).
explicit    Valor lógico. Se =FALSE (padrão), não produz nenhum efeito;
        se =TRUE, coloca os valores de densidade diretamente no mapa.

Details:

    A distribuição e a densidade de organismos no espaço é vista muitas
vezes como homogênea. Entretanto, é sempre possível que indivíduos de
diferentes
sexos ou morfotipos ocupem diferentes locais no ambiente. Essa função tem
como
objetivo permitir uma análise gráfica exploratória em diferentes escalas
(ajustando wx e wy) da distribuição e densidade de dois tipos de indivíduos
observados. A matriz gerada dá a possibilidade analisar mais profundamente
a heterogeneidade em uma dada escala ou entre escalas diferentes.

Values:

    A função pode retornar uma matriz (plot=F) ou um mapa (plot=T). Tanto
a matriz quanto o mapa representam os mesmos dados: número de indivíduos de
fator 1 por célula (output=1), número de indivíduos de fator 2 por célula
(output=2)
ou razão fator 1:fator 2 (output=3) por célula.



Warnings:

    O data frame de entrada deve ter apenas dois fatores (e.g., 1 e 2, M e
F),
caso contrário, a função irá parar e mostrar mensagem de erro. Ademais, o
data
frame não pode conter NAs; caso tenha, a função avisará o usuário para que a
correção
seja feita.

    Caso wx ou wy seja um número decimal, irá ser arredondado para baixo
(e.g.,2.9=2). Caso wx ou wy seja muito grande, a função poderá levar mais
tempo
para rodar.

Notes:

    A matriz resultante da razão entre densidade de indivíduos de fator 1
pela densidade de indivíduos de fator 2 pode ter valores Inf, NaN e zero
(além
dos demais, que são divisões "normais"). Inf é resultante da divisão de
qualquer valor acima de zero por zero, isto é, representa que naquela célula
só há indivíduos de fator 1. Zero (0) é resultante da divisão de zero por
qualquer valor maior que zero, isto é, representa que naquela célula só há
```

```
indivíduos de fator 2. NaN é resultante da divisão de zero por zero, isto é,
representa que naquela célula não há indivíduos nem de fator 1 nem de fator
2.

    Os valores para x e y devem ser coordenadas métricas. É importante
ficar claro que a ordem dos fatores é estabelecida alfabeticamente, isto é,
se os fatores (tipos de indivíduos) no dataframe forem "M" e "F", "F" será o
"fator 1" e "M" será o "fator 2".

    Para os mapas, no caso de output=1 ou =2, o gradiente de cores é feito a
 fim de representar quantos indivíduos há em cada célula do respectivo
fator. No caso
 de output=3, o gradiente de cores é feito a fim de representar a razão de
indivíduos
 de fator 1 por indivíduos de fator 2 em cada célula.

    Se explicit=T, quando output=1 ou =2, os números plotados representam
quantos
indivíduos há em cada célula. Quando output=3, os números plotados mostram
quantos
indivíduos de fator 1 e quantos indivíduos de fator 2 há em cada célula
separado por
":", já que o mapa representa justamente o valor da razão entre os dois em
cada célula.


Autor(s):

Pietro Pollo (pietro_pollo@hotmail.com).

Examples:

#data frame de exemplo:
exe=data.frame(x=c(rnorm(150,100,15),rnorm(250,100,20)),y=c(rnorm(150,100,20
), rnorm(250,100,22)),fatores=c(rep("M",150),rep("F",250)))
#note que apesar de indivíduos "M" estarem antes dos indivíduos "F" no
dataframe, "F" é o fator 1 e "M" o fator 2.

matrixmap(exe,15,15,output=1,plot=F)
#matriz de densidade referente aos indivíduos de fator 1 somente, quando x e
y tem 15 segmentos cada.

matrixmap(exe,15,15,output=1)
#mapa de densidade referente aos indivíduos de fator 1 somente (quando x e y
tem 15 segmentos cada).

matrixmap(exe,15,15,output=1,explicit=T)
#mapa de densidade referente aos indivíduos de fator 1 somente, com valores
explícitos no plot (quando x e y tem 15 segmentos cada).

matrixmap(exe,15,15,output=2,plot=F) #matriz de densidade referente aos
```

```
indivíduos de fator 2 somente, quando x e y tem 15 segmentos cada.

matrixmap(exe,15,15,output=2) #mapa de densidade referente aos indivíduos de
fator 2 somente (quando x e y tem 15 segmentos cada).

matrixmap(exe,15,15,output=2,explicit=T) #mapa de densidade referente aos
indivíduos de fator 2 somente, com valores explícitos no plot (quando x e y
tem 15 segmentos cada).

matrixmap(exe,15,15,plot=F) #matriz resultante da razão entre quantia de
indivíduos de fator 1 por fator 2 em cada célula, quando x e y tem 15
segmentos cada.

matrixmap(exe,15,15) #mapa que mostra a razão entre densidades (quando x e y
tem 15 segmentos cada).

matrixmap(exe,15,15,explicit=T) #mapa que mostra a razão entre densidades
com os valores explícitos plotados (quando x e y tem 15 segmentos cada).
```

Código da função

```
matrixmap=function(data,wx,wy,output=3,plot=TRUE,explicit=FALSE) #Nomeia a
função e respectivos argumentos.
{
  library(plotrix) #Carrega o pacote plotrix (um dos pacotes básicos do R)
que será necessário para usar algumas funções de cores mais a frente (e.g.,
color.scale).
   ### VERIFICANDO ERROS ESPECÍFICOS PARA O DATA FRAME DE ENTRADA ###
  data[,3]=as.factor(data[,3]) #Transforma a terceira coluna em fator (caso
ainda não seja dessa classe).
  if (length(levels(data[,3]))!=2) #Caso a terceira coluna do data frame
tenha mais ou menos que dois fatores...
  {
    stop("Terceira coluna do data frame deve ter somente dois fatores") #...
a função para e mostra essa mensagem de erro.
  }
  if(length(data[is.na(data)])!=0) #Caso o data frame contenha algum NA...
  {
    stop("O data frame contém um ou mais NAs, corrija-o") #... a função para
e mostra essa mensagem de erro.
  }
  ### VERIFICANDO ERROS ESPECÍFICOS PARA OS ARGUMENTOS ###
  if (wx<2) #Se o argumento wx for menor que 2...
  {
    stop("wx precisa ser um número maior que 2") #... a função para e mostra
essa mensagem de erro.
  }
  if (wy<2) #Se o argumento wy for menor que 2...
```

```
  {
    stop("wy precisa ser um número maior que 2") #... a função para e mostra
essa mensagem de erro.
  }
  if(plot!=TRUE & plot!=FALSE) #Caso o argumento plot não seja TRUE nem
FALSE...
  {
    stop("O argumento plot deve ser TRUE ou FALSE") #... a função para e
mostra essa mensagem de erro.
  }
  if(explicit!=TRUE & explicit!=FALSE) #Caso o argumento explicit não seja
TRUE nem FALSE...
  {
    stop("O argumento explicit deve ser TRUE ou FALSE") #... a função para e
mostra essa mensagem de erro.
  }
  ### CRIANDO SUBSETS DE ACORDO COM FATOR ####
  mat1=as.matrix(subset(data,data[,3]==levels(data[,3])[1])[,-3],ncol=2)
#Pega o subset apenas com os valores do primeiro fator e gera uma matriz com
apenas x e y (exclui a terceira coluna).
  mat2=as.matrix(subset(data,data[,3]==levels(data[,3])[2])[,-3],ncol=2)
#Pega o subset apenas com os valores do segundo fator e gera uma matriz com
apenas x e y (exclui a terceira coluna).
  ### VERIFICANDO ERROS ESPECÍFICOS PARA OS ARGUMENTOS ###
  if (wx<2) #Se o argumento wx for menor que 2...
  {
    stop("wx precisa ser um número maior que 2") #... a função para e mostra
essa mensagem de erro.
  }
  if (wy<2) #Se o argumento wy for menor que 2...
  {
    stop("wy precisa ser um número maior que 2") #... a função para e mostra
essa mensagem de erro.
  }
  ### CRIANDO SUBSETS DE ACORDO COM FATOR ####
  mat1=as.matrix(subset(data,data[,3]==levels(data[,3])[1])[,-3],ncol=2)
#Pega o subset apenas com os valores do primeiro fator e gera uma matriz com
apenas x e y (exclui a terceira coluna).
  mat2=as.matrix(subset(data,data[,3]==levels(data[,3])[2])[,-3],ncol=2)
#Pega o subset apenas com os valores do segundo fator e gera uma matriz com
apenas x e y (exclui a terceira coluna).
  ### DELIMITANDO OS SEGMENTOS E FAZENDO AS CÉLULAS/QUADRANTES ###
  seqx=seq(min(data[,1]),max(data[,1]),len=(wx+1)) #Cria vetor com wx
intervalos de x.
  seqy=seq(min(data[,2]),max(data[,2]),len=(wy+1)) #Cria vetor com wy
intervalos de y.
  matfin1=matrix(NA,ncol=wx,nrow=wy) #Cria uma matriz nula, que será
"completada" (valores substituídos) a fim de torná-la a matriz final dos
dados de fator 1 (densidade dos indivíduos em cada célula).
  matfin2=matrix(NA,ncol=wx,nrow=wy) #Cria uma matriz nula, que será
"completada" (valores substituídos) a fim de torná-la a matriz final dos
```

```
dados de fator 2 (densidade dos indivíduos em cada célula).
  for (i in 1:wx) #Divide os dados de cada fator nos segmentos de x.
  {
    if (i==1) #Para o primeiro segmento somente, pega inclusive os pontos no
limite inferior do segmento (valores mínimos). Necessário para não repetir
valores na matriz (notar a diferença ">=" ao contrário de ">").
    {
      matint11=matrix(mat1[which(mat1[,1]>=seqx[i] &
mat1[,1]<=seqx[i+1]),],ncol=2) #Matriz com o primeiro segmento de x do fator
1.
      matint12=matrix(mat2[which(mat2[,1]>=seqx[i] &
mat2[,1]<=seqx[i+1]),],ncol=2) #Matriz com o primeiro segmento de x do fator
2.
    }
    else #Para os outros segmentos, não inclui os valores mínimos do
segmento (">" ao contrário de ">=")
    {
      matint11=matrix(mat1[which(mat1[,1]>seqx[i] &
mat1[,1]<=seqx[i+1]),],ncol=2) #Matriz com os outros segmentos de x do fator
1.
      matint12=matrix(mat2[which(mat2[,1]>seqx[i] &
mat2[,1]<=seqx[i+1]),],ncol=2) #Matriz com os outros segmentos de x do fator
2.
    }
    for(k in 1:wy) #A partir de cada matriz de segmento de x acima, divide
nos segmentos de y.
    {
      if(k==1) #Para o primeiro segmento somente, pega inclusive os pontos
no limite inferior do segmento (valores mínimos). Necessário para não
repetir valores na matriz (notar a diferença ">=" ao contrário de ">").
      {
        matint21=matrix(matint11[which(matint11[,2]>=seqy[k] &
matint11[,2]<=seqy[k+1]),],ncol=2) #Matriz com o primeiro segmento de y
(para cada segmento de x) do fator 1.
        matint22=matrix(matint12[which(matint12[,2]>=seqy[k] &
matint12[,2]<=seqy[k+1]),],ncol=2) #Matriz com o primeiro segmento de y
(para cada segmento de x) do fator 2.
      }
      else #Para os outros segmentos, não inclui os valores mínimos do
segmento (">" ao contrário de ">=")
      {
        matint21=matrix(matint11[which(matint11[,2]>seqy[k] &
matint11[,2]<=seqy[k+1]),],ncol=2) #Matriz com os outros segmentos de y
(para cada segmento de x) do fator 1.
        matint22=matrix(matint12[which(matint12[,2]>seqy[k] &
matint12[,2]<=seqy[k+1]),],ncol=2) #Matriz com os outros segmentos de y
(para cada segmento de x) do fator 2.
      }
      matfin1[k,i]=length(matint21[,1]) #A partir das matrizes
intermediárias geradas acima, conta o número de indivíduos em cada segmento
```

```
de y em cada segmento de x, formando a matriz de densidade final dos dados
de fator 1.
      matfin2[k,i]=length(matint22[,1]) #A partir das matrizes
intermediárias geradas acima, conta o número de indivíduos em cada segmento
de y em cada segmento de x, formando a matriz de densidade final dos dados
de fator 2.
    }
  }
  matfin3=matfin1/matfin2 #Matriz resultante da razão entre as matrizes de
densidade final de fator 1 pela matriz de densidade final de fator 2. Dito
isso, valores finitos maiores que zero representam uma real razão, porém, há
possibilidade de haverem outros três tipos de resultado: 0, Inf e NaN.
Quando o valor é 0, significa que a divisão foi 0/?, Inf foi ?/0 e NaN 0/0.
  if (plot==FALSE) #Se o usuário deseja somente as matrizes "cruas" geradas
aqui em cima, dá as seguintes possibilidades:
  {
    if (output==1) #Somente resultado referente ao fator 1.
    {
      return(matfin1) #Mostra a respectiva matriz.
    }
    if (output==2) #Somente resultado referente ao fator 2.
    {
      return(matfin2) #Mostra a respectiva matriz.
    }
    if (output==3) #Resultado da razão entre fator 1 por fator 2.
    {
      return(matfin3) #Mostra a respectiva matriz.
    }
  }
  ### PLOTS ###
  if (plot==TRUE) #Se o usuário optar por plotar os resultados:
  {
    if (output==1) #Referente ao fator 1 somente:
    {
      layout(matrix(c(1,1,1,1,2,2,2,2), ncol=2),widths=c(8.8, 1.2),
heights=c(1,1)) #Organiza o espaço (layout) para os plots a seguir. Está
assim tão complexo para ficar exatamente nas posições do output=3, ou seja,
para padronizar.
      par(mar=c(5,4,2,0.5)) #Modifica as margens para o próximo plot.
plot(NA,type="n",xlim=c(min(data[,1]),max(data[,1])),ylim=c(min(data[,2]),ma
x(data[,2])),xlab="x",ylab="y") #Plota o mapa "em branco".
      for (d in 1:wx) #Para cada segmento de x,
      {
        for (e in 1:wy) #E para cada segmento de y (de cada segmento de x),
        {
rect(xleft=seqx[d],xright=seqx[d+1],ybottom=seqy[e+1],ytop=seqy[e],col=apply
(matrix(color.scale(c(matfin1,0),cs1=c(1,0.7),cs2=c(1,0),cs3=c(1,0))[-
length(color.scale(c(matfin1,0)))],ncol=dim(matfin1)[2]),2,rev)[e,d],border=
T) #Gera um retângulo com uma cor equivalente ao respectivo valor da célula.
Cor vem de um gradiente de cores criado para a matriz de fator 1 (tons
vermelhos).
```

```
        }
      }
      if (explicit==TRUE) #Se o usuário escolhe que quer os valores
diretamente plotados no mapa:
      {
        for (n in 1:wx) #Para cada segmento de x,
        {
          for (o in 1:wy) #E para cada segmento de y (de cada segmento de
x),
          {
text(x=mean(c(seqx[n],seqx[n+1])),y=mean(c(seqy[o+1],seqy[o])),labels=apply(
matfin1,2,rev)[o,n]) #Gera um texto contendo o número de indivíduos de fator
1 em cada célula.
          }
        }
      }
      par(mar=c(5,1.2,2,3.6)) #Modifica as margens para o próximo plot.
image(x=1,y=(0:max(matfin1)),z=matrix(0:max(matfin1),nrow=1),col=color.scale
(0:max(matfin1),cs1=c(1,0.7),cs2=c(1,0),cs3=c(1,0)),axes=FALSE,xlab="",ylab=
"") #Cria a escala de cor, ao lado do plot principal.
      mtext("number of factor 1 individuals",side=2,cex=0.7,line=0.15) #Gera
um texto explicativo para a barra de cores.
      axis(4,las=2) #Eixo para mostrar os valores da respectiva escala de
cor.
      layout(matrix(1,ncol=1)) #"Reseta" o layout gráfico.
      par(mar=c(5,4,4,2)) #Muda as margens para os valores default.
    }
    if (output==2) #Referente ao fator 2 somente:
    {
      layout(matrix(c(1,1,1,1,2,2,2,2), ncol=2),widths=c(8.8, 1.2),
heights=c(1,1)) #Organiza o espaço (layout) para os plots a seguir. Está
assim tão complexo para ficar exatamente nas posições do output=3, ou seja,
para padronizar.
      par(mar=c(5,4,2,0.5)) #Modifica as margens para o próximo plot.
plot(NA,type="n",xlim=c(min(data[,1]),max(data[,1])),ylim=c(min(data[,2]),ma
x(data[,2])),xlab="x",ylab="y") #Plota o mapa "em branco".
      for (a in 1:wx) #Para cada segmento de x,
      {
        for (b in 1:wy) #E para cada segmento de y (de cada segmento de x),
        {
rect(xleft=seqx[a],xright=seqx[a+1],ybottom=seqy[b+1],ytop=seqy[b],col=apply
(matrix(color.scale(c(matfin2,0),cs1=c(1,0),cs2=c(1,0),cs3=c(1,0.7))[-
length(color.scale(c(matfin2,0)))],ncol=dim(matfin2)[2]),2,rev)[b,a],border=
T) #Gera um retângulo com uma cor equivalente ao respectivo valor da célula.
Cor vem de um gradiente de cores criado para a matriz de fator 2 (tons
azuis).
        }
      }
      if (explicit==TRUE) #Se o usuário escolhe que quer os valores
diretamente plotados no mapa:
```

```
      {
        for (t in 1:wx) #Para cada segmento de x,
        {
          for (u in 1:wy) #E para cada segmento de y (de cada segmento de
x),
          {
text(x=mean(c(seqx[t],seqx[t+1])),y=mean(c(seqy[u+1],seqy[u])),labels=apply(
matfin2,2,rev)[u,t]) #Gera um texto contendo o número de indivíduos de fator
2 em cada célula.
          }
        }
      }
      par(mar=c(5,1.2,2,3.6)) #Modifica as margens para o próximo plot.
image(x=1,y=(0:max(matfin2)),z=matrix(0:max(matfin2),nrow=1),col=color.scale
(0:max(matfin2),cs1=c(1,0),cs2=c(1,0),cs3=c(1,0.7)),axes=FALSE,xlab="",ylab=
"") #Cria a escala de cor, ao lado do plot principal.
      mtext("number of factor 2 individuals",side=2,cex=0.7,line=0.15) #Gera
um texto explicativo para a barra de cores.
      axis(4,las=2) #Eixo para mostrar os valores da respectiva escala de
cor.
      layout(matrix(1,ncol=1)) #"Reseta" o layout gráfico.
      par(mar=c(5,4,4,2)) #Muda as margens para os valores default.
    }
    if (output==3) #Referente à razão entre fator 1 e fator 2 (resultado
principal):
    {
      matfincol=as.vector(matfin3) #Transforma a matriz final 3 em vetor
para facilitar as posteriores transformações no objeto.
      if(length(matfincol[matfincol!=0 & matfincol!=Inf &
!is.nan(matfincol)])==0) #Caso haja somente zeros, Inf e NaNs na matriz, o
gráfico não será gerado e portanto...
      {
        stop("Não é possível executar a função gráfica, pois não há células
em que indivíduos dos dois fatores estejam presentes") #... para a função e
mostra essa mensagem de erro.
      }
      matfincol[which(matfincol==0)]=9999 #Transforma todos os zeros em
9999, já que zero não faz parte da proporção.
      matfincol[which(matfincol!=9999 & matfincol!=Inf & !is.nan(matfincol)
& matfincol<1)]=2-(1/(matfincol[which(matfincol!=9999 & matfincol!=Inf &
!is.nan(matfincol) & matfincol<1)])) #Transforma todos os valores menores de
1 (na prática todos entre 0 e 1) nos seus inversos e soma 2. A que se deve
essa transformação? Como 1 a 2 deve ter o mesmo número de cores que 1 a 0.5
(1:2), fazer o inverso faz com que as frações (0<x<1) sejam transformados em
números negativos equivalentes. A adição de 2 é justamente para deixar a
sequencia contínua, pois assim o valor designado a 0.5 (1:2) acaba sendo 0,
o de 0.333 (1:3) é -1, o de 0.25 (1:4) é -2, ou seja, deixa a escala de
cores linear.
      maximo=max(matfincol[which(matfincol!=9999 & matfincol!=Inf &
!is.nan(matfincol))]) #Objeto com o valor máximo do resultado (excluindo-se
Infs, 9999 (antigo zero) e NaNs).
```

```
      minimo=min(matfincol[which(matfincol!=9999 & matfincol!=Inf &
!is.nan(matfincol))]) #Objeto com o valor mínimo do resultado (excluindo-se
Infs, 9999 (antigo zero) e NaNs).
      cores=rainbow(((maximo-minimo)+1)*100,start=0,end=0.9) #Cria vetor de
cores que serão usadas no gráfico final da razão entre fatores (matfin3). A
multiplicação por 100 é para deixar o gradiente de cores mais suave.
      matfincol2=as.character(matfincol) #Transforma o vetor com os
resultados em caracter, pois cada valor será substituído por cores, que são
caracteres.
      for(r in 1:length(matfincol[matfincol!=9999 & matfincol!=Inf &
!is.nan(matfincol)])) #r será o número de valores que são razões (aqueles
que não são Infs, 9999 (antigo zero) e NaNs).
      {
        matfincol2[matfincol!=9999 & matfincol!=Inf &
!is.nan(matfincol)][r]=cores[((matfincol[matfincol!=9999 & matfincol!=Inf &
!is.nan(matfincol)][r]-minimo)+1)*100] #Substitui os valores (razões) pelas
rescpectivas cores de um gradiente (excluindo-se Infs, 9999 (antigo zero) e
NaNs).
      }
      matfincol2[which(matfincol2=="Inf")]="grey48" #Inf na verdade é quando
a célula só tem indivíduos do fator 1, portanto ganha uma cor única, no caso
cinza escuro.
      matfincol2[which(matfincol2=="9999")]="grey78" #9999 (provindo do 0
original) na verdade é quando a célula só tem indivíduos do fator 2,
portanto ganha uma cor única, no caso cinza claro.
      matfincol2[which(matfincol2=="NaN")]="white" #NaN (provindo de 0/0) é
quando a célula não contém qualquer indivíduo, portanto ganha uma cor única,
no caso branco.
      matfincol3=matrix(matfincol2,ncol=dim(matfin3)[2]) #Transforma o vetor
gerado até aqui em matriz novamente para a confecção dos retângulos.
      layout(matrix(c(1,1,1,1,2,3,4,5), ncol=2),widths=c(8.8, 1.2),
heights=c(5.7,1,1,2.3)) #Organiza o espaço (layout) para os plots a seguir.
      par(mar=c(5,4,2,0.5)) #Modifica as margens para o próximo plot.
plot(NA,type="n",xlim=c(min(data[,1]),max(data[,1])),ylim=c(min(data[,2]),ma
x(data[,2])),xlab="x",ylab="y") #Plota o mapa "em branco".
      for (l in 1:wx) #Para cada segmento de x,
      {
        for (m in 1:wy) #E para cada segmento de y (de cada segmento de x),
        {
rect(xleft=seqx[l],xright=seqx[l+1],ybottom=seqy[m+1],ytop=seqy[m],col=apply
(matfincol3,2,rev)[m,l],border=T) #Gera um retângulo com uma cor equivalente
ao respectivo valor da célula. Cor vem do objeto cores.
        }
      }
      if (explicit==TRUE) #Se o usuário escolhe que quer os valores
diretamente plotados no mapa:
      {
        mattext=rep(NA,length(as.vector(matfin1))) #Cria um vetor nulo com o
mesmo número de valores de matfin1 (poderia ser matfin2 ou matfin3, já que
todas as matrizes possuem o mesmo comprimento).
```

```
        for (v in 1:length(as.vector(matfin1))) #Para cada célula no vetor,
        {
          mattext[v]=paste(matfin1[v],matfin2[v],sep=":") #Coloca os
caractres que mostram a divisão que foi feita para chegar a matfin3 (matfin1
: matfin2).
        }
        for (p in 1:wx) #Para cada segmento de x,
        {
          for (q in 1:wy) #E para cada segmento de y (de cada segmento de
x),
          {
text(x=mean(c(seqx[p],seqx[p+1])),y=mean(c(seqy[q+1],seqy[q])),labels=apply(
matrix(mattext,ncol=dim(matfin3)[2]),2,rev)[q,p]) #Gera um texto contendo o
número de indivíduos de fator 1 : número de indivíduos de fator 2 em cada
célula.
          }
        }
      }
      par(mar=c(0,1.2,2,3.6)) #Modifica as margens para o próximo plot.
image(x=1,y=(100:length(cores)),z=matrix(100:length(cores),nrow=1),col=cores
[-(1:99)],axes=FALSE,xlab="",ylab="") #Gera a escala de cores. Os primeiros
99 valores do gradiente na verdade não são usados devido ao método da
multiplicação por 100 no momento de confeccionar os retângulos coloridos.
      mtext("ratio (factor 1:factor 2)",side=2,cex=0.7,line=0.15) #Gera um
texto explicativo para a barra de cores.
      axis(4, at=((((-3:5)-
minimo)+1)*100),labels=paste(c(rep(1,5),2:5),c(5:2,rep(1,5)),sep=":"),las=2)
#Eixo da escala de cores com valores de 5:1 a 1:5, range de valores mais
comumente encontrados.
      box(which="plot") #Coloca borda (linha preta) ao redor da escala de
cores.
      par(mar=c(0,1.2,1,3.6)) #Modifica as margens para o próximo plot.
      image(x=1,y=1,z=matrix(1,1,1),col="grey48",axes=FALSE,xlab="",ylab="")
#Gera um quadrado de cor cinza escuro.
      mtext("only\nfactor 1", side=4, line=0.4,cex=0.6,las=2) #Faz o texto
ao lado do quadrado de cor (legenda).
      box(which="plot") #Coloca borda (linha preta) ao redor do quadrado de
cor.
      image(x=1,y=1,z=matrix(1,1,1),col="grey78",axes=FALSE,xlab="",ylab="")
#Gera um quadrado de cor cinza claro.
      mtext("only\nfactor 2", side=4, line=0.4,cex=0.6,las=2) #Faz o texto
ao lado do quadrado de cor (legenda).
      box(which="plot") #Coloca borda (linha preta) ao redor do quadrado de
cor.
      par(mar=c(5,1.2,1,3.6)) #Modifica as margens para o próximo plot.
      image(x=1,y=1,z=matrix(1,1,1),col="white",axes=FALSE,xlab="",ylab="")
#Gera um quadrado de cor branca.
      box(which="plot") #Coloca borda (linha preta) ao redor do quadrado de
cor.
      mtext("empty", side=4, line=0.4,cex=0.6,las=2) #Faz o texto ao lado do
quadrado de cor (legenda).
```

```
      layout(matrix(1,ncol=1)) #"Reseta" o layout.
      par(mar=c(5,4,4,2)) #Coloca as margens em valores default.
    }
  }
}
```