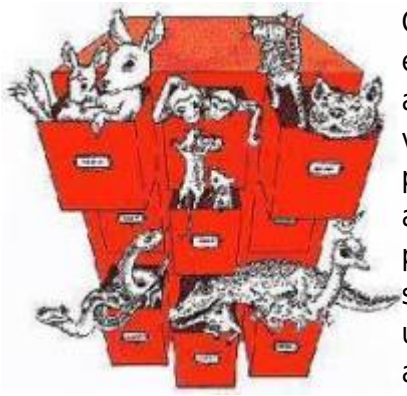
-

Análise de classificação - Roteiro em R



Os métodos de classificação agrupam objetos conforme a similaridade entre eles. Se tomamos amostras de comunidades em que registramos a presença ou abundância de cada espécie, isso pode ser útil para verificarmos se as amostras formam conjuntos discretos. Para tanto, precisamos primeiro calcular um índice que nos diga o quanto cada amostra é similar às outras em termos de composição de espécies. A primeira parte dessa atividade explica como calcular medidas de similaridade e distância entre parcelas amostrais. Em seguida usamos um método de agrupamento dessas parcelas para finalmente apresentar os dados graficamente em um dendrograma.

Similaridade & Distância

Existe uma infinidade de índices utilizados em ecologia para medir a similaridade ou dissimilaridade (distância) entre objetos. O primeiro passo para classificar comunidades é usar esses índices para expressar a diferença entre pares de amostras das comunidades. Em ecologia, essas amostras, em geral, podem ter dois tipos de dados: i) quais espécies estão presentes (dados de presença/ausência, ou binários), ou; ii) qual a abundância (que pode ser medida de diversas formas) de cada espécie presente (dados quantitativos).

A seguir apresentamos alguns dos índices mais simples para cálculo de similaridade para dados de presença/ausência e de abundância.

Similaridade por presença e ausência

Jaccard

O índice de Jaccard indica a proporção de espécies compartilhadas entre duas amostras em relação ao total de espécies. Uma forma de calculá-lo é:

$$J= \frac{S_{com}}{s_1 + s_2 - S_{com}}$$

O que é o mesmo que:

$$J= \frac{S_{com}}{S}$$

onde:

- S_{com} é o número de espécies em comum nas duas amostras
- $ s_1$ e s_2 é o número total de espécies em cada uma das amostras
- S é o total de espécies no conjunto de amostras

Vamos supor que foi feita uma amostra de duas parcelas de vegetação. O número de indivíduos das espécies de plantas encontradas em cada uma dessas parcelas fictícias foi:

	plot_1	**plot_2**
sp_1	2	8
sp_2	13	6
sp_3	0	5
sp_4	1	15
sp_5	20	0

Comparando a presença/ausência de espécies, você acha que as parcelas são muito parecidas ou muito diferentes? E comparando a abundância das espécies?

Vamos usar esses valores para o cálculo do índice de Jaccard no R. Para isso, copie os comandos abaixo e cole-os na linha de comando do R (obs.: as frases que iniciam com o símbolo "##" são apenas textos explicativos):

```
## Duas parcelas fictícias
(plot1=c(2,13,0,1,20))
(plot2=c(8,6,5,15,0))
## Criando o objeto //parcela// com as duas parcelas:
(parcelas=data.frame(plot1,plot2))
## Calculando o número de espécies em cada parcela
(nsp_1=sum(plot1>0))
(nsp_2=sum(plot2>0))
##Calculando o total de espécies em comum
(nsp_com=sum(plot1>0 & plot2>0))
## Indice de Jaccard
jacc= nsp_com/(nsp_1 + nsp_2 - nsp_com)
jacc
```

Qual a proporção de similaridade observada entre as duas parcelas fictícias? Anote essa informação.

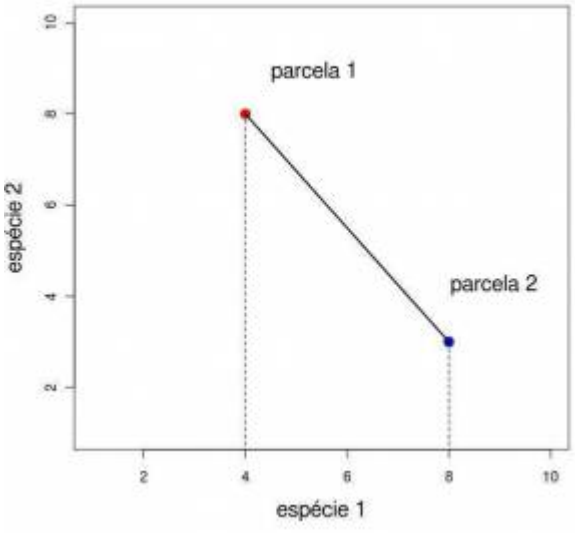
Similaridade por abundância

Quando queremos comparar amostras não só pelas presenças, mas também por suas abundâncias precisamos usar um índice quantitativo. Uma das maneiras de pensar nesses índices é pela medida de distância, ou seja pelo quanto uma parcela se diferencia da outra, medida pela diferença entre as abundâncias das espécies. A seguir dois dos índices quantitativos mais usados em ecologia de comunidades.

Distância Euclidiana

O inverso ou complementar da similaridade são as medidas de distância. No exemplo anterior poderíamos dizer que a similaridade é de 60% entre as parcelas ou que a dissimilaridade (ou distância) entre elas é de 40%!

Uma das medidas de distância muito comum é a distância euclidiana, baseada na medida em um sistema de coordenadas cartesianas. Podemos usá-la para expressar a distância entre duas amostras de vegetação ao registramos as abundâncias de cada espécie. Se registramos apenas duas espécies, suas abundâncias podem ser representadas em um plano cartesiano, em que no eixo X temos a abundância de uma das espécies e no eixo Y a abundância da outra espécie. Os pontos no gráfico representam duas amostras/parcelas (Obs.: não são as nossas parcelas fictícias criadas acima).



Nesse caso a distância euclidiana pode ser descrita como a distância entre duas parcelas medidas por

unidades de indivíduos de duas espécies. Para calcular a distância utilizamos o teorema de Pitágoras e calculamos a hipotenusa, como representado na figura. Usamos a equação abaixo:

$$d_E=\sqrt{(x_1-x_2)^2 + (y_1-y_2)^2}$$

Onde x_1, x_2 são as abundâncias de uma espécie nas parcelas 1 e 2, e y_1, y_2 são as abundâncias da outra espécie nas mesmas parcelas.

Para um maior número de espécies fica difícil fazer uma representação gráfica, mas a lógica é a mesma e a equação é generalizada assim:

$$d_E\ = \ \sqrt{\sum_{i=1}^S(n_{i1}-n_{i2})^2}$$

Onde n_{i1} e n_{i2} são as abundâncias da i-ésima espécie na primeira e segunda parcela, e S é o total de espécies, como já definido.

Vamos calcular então a distância euclidiana entre as nossas duas parcelas fictícias do início dessa atividade:

```
eucl=sqrt(sum((plot1-plot2)^2))
eucl
```

Nesse caso, a distância entre as parcelas é de 26,6 indivíduos. E o que isso representa? A distância entre essas duas parcelas é pequena ou grande? Difícil responder isso com apenas um valor, não é?

Apesar da distância euclidiana ser muito fácil de ser entendida e muito útil para análises de classificação, ela não varia em um intervalo de 0 a 1 e, então, o valor obtido não é comparável aos valores obtidos por outras medidas de similaridade que variam entre 0 e 1. Então, para podermos comparar a medida de similaridade obtida para dados de presença/ausência (pelo índice de Jaccard), com uma medida de similaridade obtida para dados de abundância vamos usar uma outra medida chamada de “Bray-Curtis” em homenagem aos seus autores.

Bray-Curtis

O índice de Bray-Curtis pode ser expresso como uma proporção de similaridade ou dissimilaridade (distância) na abundância das espécies. Em qualquer um dos casos seus valores vão de um máximo de um ao mínimo de zero. Essa padronização no intervalo entre um e zero facilita a interpretação e comparação.

Similaridade

A similaridade de Bray-Curtis é:

$$\frac{2\sum_{i=1}^S \min(n_{i1},n_{i2})}{N}$$

Onde N é a soma de indivíduos de todas as espécies e parcelas, e $\min(n_{i1},n_{i2})$ é a menor

das duas abundâncias da espécie i, entre as duas parcelas. Como já definido, n_{i1} e n_{i2} são as abundâncias da i-ésima espécie na primeira e segunda parcela, S é o total de espécies.

Distância

A distância de Bray Curtis é:

$$\frac{\sum_{i=1}^S|n_{i1}-n_{i2}|}{N}$$

Onde $|n_{i1}-n_{i2}|$ é o valor absoluto da diferença das abundâncias da espécie i nas duas parcelas.

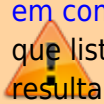
Cálculo

Vamos calcular então a similaridade de Bray-Curtis entre as duas parcelas:

```
bc.sim=2*sum(apply(parcelas, 1, min))/(sum (parcelas))
bc.sim
```

Qual valor você observou? Ele é parecido com o valor do índice de Jaccard? Como você explicaria isso?

Matriz de Similaridade



Para seguir em frente é necessário que você tenha na área de trabalho do R as amostras das comunidades virtuais montadas, como descrito no roteiro padrões de gradientes em comunidades. Certifique-se disso com o comando `ls()`, que lista os objetos da área de trabalho do R. Na lista resultante devem estar os objetos `amost.cont` e `amost.disc`:

```
> ls()
 [1] "amost"    "amost.cont"  "amost.disc"
```

Agora vamos aplicar o índice de Bray-Curtis para o conjunto de parcelas tomadas ao longo de um gradiente simulado. Para tanto temos que fazer uma função que calcule a similaridade entre cada par de parcelas. Ei-la:

```
sim<-function(dados, indice="bc")
    {
    nplot=dim(dados)[2]
    similar=matrix(1,ncol=nplot,nrow=nplot)
    rownames(similar)<-paste("plot", c(1:nplot))
    colnames(similar)<-paste("plot", c(1:nplot))
```

```
        for(i in 1:(nplot-1))
        {
        m=i+1
        for(m in m:nplot)
        {
        if(indice=="jacc")
            {
            dados[dados>0]=1
            co.oc=sum(dados[,i]>0 & dados[,m]>0)
            total.sp=sum(dados[,i])+sum(dados[,m])-co.oc
            similar[i,m]=co.oc/total.sp
            similar[m,i]=co.oc/total.sp
            }
        if(indice=="bc")
            {
            bc.sim=2*sum(apply(dados[,c(i,m)], 1, min))/(sum
(dados[,c(i,m)]))
            similar[i,m]=bc.sim
            similar[m,i]=bc.sim
            }
        }
        }
    return(round(similar,3))
    }
```

Agora vamos aplicar essa função para a nossa comunidade virtual contínua:

```
sim.cont1=sim(amost.cont, indice="bc")
```

Verifique a matriz resultante

```
sim.cont1
```

Qual o elemento da linha 1 coluna 4 desta matriz?

```
sim.cont1[1,4]
```

E da linha 4 e coluna 1?

```
sim.cont1[4,1]
```

Verifique o mesmo para linha 5 e 7 e coluna 5 e 7

```
sim.cont1[5,7]
sim.cont1[7,5]
```

O que significam os valores **1,00** na matriz resultante?
Por que os valores na posição [linha1,coluna4] são iguais aos da posição [linha4,coluna1]?
Consegue identificar padrões nos dados mostrados na matriz?

Faça o mesmo para amostras da comunidade discreta (objeto `amost.disc`), mas guarde o resultado em um objeto de nome `sim.disc1`. Chegamos nas matrizes de similaridades, o primeiro passo da nossa classificação.

Agrupamento

Há vários métodos e algoritmos[1)] para agrupar os objetos em uma análise de classificação. Primeiro precisamos decidir se queremos iniciar juntando os elementos (análise aglomerativa) ou se queremos tomar todo o conjunto de objetos e ir separando em grupos (análise divisiva). No nosso caso, vamos iniciar com as parcelas como unidades e vamos montando grupos sucessivamente a partir daqueles já formados. Isso se chama análise de classificação (ou agrupamento) hierárquica aglomerativa. Apesar do nome feio, não há nada de complicado na lógica da análise. Vamos acompanhar passo a passo.

Qual o máximo de similaridade nesta matriz?

```
max(sim.cont1, na.rm=TRUE)
```

Vamos perguntar ao R quais valores da matriz são iguais a este máximo

```
sim.cont1==max(sim.cont1, na.rm=TRUE)
```

Na matriz produzida agora, o “TRUE” indica as posições em que os valores máximos são observados. Notem que todos os “TRUE” estão na diagonal principal da matriz produzida. Essas similaridades não nos interessam, pois na nossa matriz esses valores correspondem à similaridade de uma parcela com ela mesma. Não há porque agrupar algo com ele mesmo. Além disso, os valores se repetem acima e abaixo da diagonal [2)] Portanto, vamos retirar da nossa matriz de similaridade a diagonal, assim como as redundâncias (o triângulo superior da matriz).

```
nome.par=rownames(sim.cont1)
upper.tri(sim.cont1, diag=TRUE)
sim.cont1[upper.tri(sim.cont1, diag=TRUE)]=NA
sim.cont1
```

A função “upper.tri” seleciona todos os valores que estão acima da diagonal principal na matriz. E, em resumo, o que estamos fazendo aqui é criando uma nova matriz (mas vamos manter o mesmo nome “sim.cont1”) em que os valores da diagonal principal e os valores no triângulo acima da diagonal principal serão desconsiderados nas futuras buscas [3)].

Primeira Ligação

Agora que desconsideramos a diagonal, vamos procurar o maior valor de similaridade

```
max1=max(sim.cont1,na.rm=TRUE)
max1
```

e então perguntamos ao R qual o par de parcelas que apresenta essa similaridade:

```
maior1=which(sim.cont1==max(sim.cont1,na.rm=TRUE), arr.ind=TRUE)
maior1
```

Vamos tomar apenas o primeiro valor maior. Para garantir isso, caso haja outros, vamos fazer assim.

```
par1=nome.par[maior1[1,]]
par1
cat("\n\t 1a. ligação ", paste(par1[1],par1[2], sep=" x "), "; ligação = ",
max1, "\n" )
```

Esse é nosso primeiro grupo: Agora temos que tomar uma outra decisão: como esse grupo formado agora irá se ligar às outras parcelas? Existem diferentes “Métodos de Ligação”. Podemos decidir que ele irá se ligar às outras parcelas pelo valor da máxima similaridade de seus componentes (ligação máxima), pela mínima, pela média, pelo centroide do grupo, etc. No nosso caso, vamos usar o método de ligação pela a média do grupo (sigla em inglês UPGMA).

No UPGMA, para definir a próxima parcela que será ligada usamos a média aritmética da similaridade entre a parcela que se quer incluir em um grupo a cada parcela já existente nesse grupo. A parcela é então atribuída ao grupo com o qual ela tem maior similaridade média com todas as parcelas (Valentin 2012).

Para fazer isso, criamos uma nova matriz de distâncias igual à anterior, com uma diferença: as duas parcelas agrupadas foram substituidas pelo grupo.

Essa nova matriz tem então as similaridades entre todas as parcelas que não foram agrupadas, e entre todas e o grupo criado. Como estamos usando o método UPGMA, a similaridade entre o grupo e as demais parcelas é a média das similaridades das parcelas do grupo com as demais parcelas.

O código abaixo faz isso passo a passo. Não se preocupe com os comandos de cada passo, mas tente entender o resultado final, que é exibido após o último comando:

```
## Preambulo: um rotulo para o grupo na matriz
g1.n1=strsplit(nome.par[maior1[1,2]]," ")[[1]][2]
g1.n2=strsplit(nome.par[maior1[1,1]]," ")[[1]][2]
g1.nome=paste("g", paste(g1.n1,g1.n2, sep=","))
g1.nome # este objeto apenas tem um novo rotulo para a nova matriz
## Nova matriz
```

```
mat.g1=sim.cont1[-maior1[1,],-maior1[1,]]
g1a=apply(sim.cont1[maior1[1,],-maior1[1,]],2,mean)
g1a[is.na(g1a)]=0
g1b=apply(sim.cont1[-maior1[1,],maior1[1,]],1,mean)
g1b[is.na(g1b)]=0
gr1=rbind(mat.g1,g1a+g1b)
grupo1=cbind(gr1,NA)
rownames(grupo1)<-c(nome.par[-maior1[1,]],g1.nome)
colnames(grupo1)[dim(grupo1)[2]]<-g1.nome
grupo1 # nova matriz com o primeiro grupo formado
```

Segunda Ligação

Agora repetimos os mesmos passos. Primeiro procuramos o par de elementos mais próximos. Note que esses elementos podem ser:

- Duas outras parcelas diferentes do par já agrupado. Nesse caso, um novo grupo é criado.
- Uma parcela e o grupo já existente. Nesse caso a parcela será agregada ao grupo já existente.

Execute o código abaixo para descobrir o que ocorre no seu conjunto de dados

```
  nome.par2=rownames(grupo1)
  max2=max(grupo1,na.rm=TRUE)
  max2
  maior2=which(grupo1==max(grupo1,na.rm=TRUE), arr.ind=TRUE)
  maior2
  g2.n1=strsplit(rownames(grupo1)[maior2[1,2]]," ")[[1]][2]
  g2.n2=strsplit(rownames(grupo1)[maior2[1,1]]," ")[[1]][2]
  g2.nome=paste(paste("g",g2.n1,sep="_"),g2.n2, sep=",")
  g2.nome
  cat("\n\n\t 2a. ligação ",
paste(nome.par2[maior2[1,2]],nome.par2[maior2[1,1]], sep=" x "), "; ligação
= ", max2, "\n" )
```

Em criamos a nova matriz de similaridade:

- Se um novo grupo foi criado, repetimos a primeira ligação: as duas parcelas que o compõem serão retiradas da matriz de similaridade, e substuída pelo grupo. As similaridades são calculadas entre cada parcela e o novo grupo. Para isso, calcula-se a similaridade de cada parcela do grupo a cada parcela que não é do grupo. Em seguida calcula-se a média dessas similaridades, que então é usada como similaridade do grupo às parcelas.
- Se uma parcela é adicionada ao grupo já existente, a similaridade do grupo a cada parcela é recalculada. Isso é feito calculando-se a média da similaridade das parcelas do grupo (que agora são três) a cada parcela. Em seguida calcula-se a média dessas similaridades, que então é usada como similaridade do grupo às parcelas.

O código abaixo faz tudo isso. A última linha mostra a matriz resultante

```
  mat.g2=grupo1[-maior2[1,],-maior2[1,]]
  g2a=apply(grupo1[maior2[1,],-maior2[1,]],2,mean)
```

```
  g2a[is.na(g2a)]=0
  g2b=apply(grupo1[-maior2[1,],maior2[1,]],1,mean)
  g2b[is.na(g2b)]=0
  gr2=rbind(mat.g2,g2a+g2b)
  grupo2=cbind(gr2,NA)
  rownames(grupo2)<-c(nome.par2[-maior2[1,]],g2.nome)
  colnames(grupo2)[dim(grupo2)[2]]<-g2.nome
  grupo2
```

Compare a matriz de similaridade após a segunda ligação com a obtida com a primeira ligação, digitando:

```
grupo1
```

Terceira Ligação

Agora mais uma vez buscamos a maior ligação entre todas na nova matriz.

```
  nome.par3=rownames(grupo2)
  max3=max(grupo2,na.rm=TRUE)
  max3
  maior3=which(grupo2==max(grupo2,na.rm=TRUE), arr.ind=TRUE)
  maior3
  g3.n1=strsplit(rownames(grupo2)[maior3[1,2]]," ")[[1]][2]
  g3.n2=strsplit(rownames(grupo2)[maior3[1,1]]," ")[[1]][2]
  g3.nome=paste(paste("g",g3.n1,sep="_"),g3.n2, sep=",")
  g3.nome
  cat("\n\n\t 3a. ligação ",
paste(nome.par3[maior3[1,2]],nome.par3[maior3[1,1]], sep=" x "), "; ligação
= ", max3, "\n" )
```

E formamos um novo grupo.

```
  mat.g3=grupo2[-maior3[1,],-maior3[1,]]
  g3a=apply(grupo2[maior3[1,],-maior3[1,]],2,mean)
  g3a[is.na(g3a)]=0
  g3b=apply(grupo2[-maior3[1,],maior3[1,]],1,mean)
  g3b[is.na(g3b)]=0
  gr3=rbind(mat.g3,g3a+g3b)
  grupo3=cbind(gr3,NA)
  rownames(grupo3)<-c(nome.par3[-maior3[1,]],g3.nome)
  colnames(grupo3)[dim(grupo3)[2]]<-g3.nome
  grupo3
```

Se continuarmos a fazer a mesma operação teremos um único grupo no final, no qual existe a última ligação entre os grupos formados anteriormente. Engenhoso e didático, mas bastante tedioso!

Por sorte temos uma função no pacote básico do R que dá conta do recado para nós. Agora que entendemos como é feito, podemos usar a função. Caso tenha ainda alguma dúvida sobre o processo

de classificação hierárquica aglomerativa, refaça os passos anteriores ou peça socorro! Caso tenha entendido já pode perguntar numa rodinha na Vila Madalena: - “Vocês conhecem o algoritmo hierárquico aglomerativo?”. Seu sucesso reprodutivo vai ficar em baixa, mas vale a pena para ver a cara dos(as) coleguinhas...

Todas as ligações de uma vez!

Vamos agora rodar a análise de classificação completa usando a nossa função de similaridade e a função básica do R *hclust()*. O argumento `method` permite usar outros métodos de ligação. Para saber mais cada método veja a seção *para saber mais*. Experimente a função com o método de ligação média:

```
clas.cont1=hclust(as.dist(1-sim.cont1), method="average")
```

Uma boa representação gráfica da análise de agrupamentos é o *dendrograma*. Crie este gráfico para a análise que você acabou de fazer com os comandos:

```
dend.cont1=as.dendrogram(clas.cont1, hang=-1)
plot(dend.cont1)
```

Um dendrograma representa uma classificação hierárquica como a que fizemos. Ou seja, representa graficamente uma sequência de ligações de elementos a outros elementos. O resultado é uma hierarquia porque são grupos dentro de grupos dentro de grupos... Para interpretar um dendrograma, tome uma das parcelas, que são os ramos terminais do dendrograma. Siga até encontrar a primeira ligação. Este é o primeiro elemento que foi ligado à parcela. Esse “primeiro vizinho” pode ser outra parcela ou um grupo de parcelas. Na escala ao lado do dendrograma veja o valor no ponto da ligação. Essa é a distância (dissimilaridade) entre os dois elementos. Você pode fazer isso com quaisquer elementos do dendrograma.

Agora você pode criar e comparar os dendrogramas obtidos com análises de agrupamento com os quatro diferentes métodos de ligação:

```
par(mfrow=c(2,2))
clas.cont1a=hclust(as.dist(1-sim.cont1), method="single")
plot(as.dendrogram(clas.cont1a, hang=-1), ylab="Bray-Curtis", main="Ligação
simples")
clas.cont1b=hclust(as.dist(1-sim.cont1), method="complete")
plot(as.dendrogram(clas.cont1b, hang=-1), ylab="Bray-Curtis", main="Ligação
completa")
clas.cont1c=hclust(as.dist(1-sim.cont1), method="average")
plot(as.dendrogram(clas.cont1c, hang=-1), ylab="Bray-Curtis", main="Ligação
```

```
média")
clas.cont1d=hclust(as.dist(1-sim.cont1), method="centroid")
plot(as.dendrogram(clas.cont1d, hang=-1), ylab="Bray-Curtis", main="Ligação
centroide", ylim=c(0,0.7))
```

Entendendo o que foi feito com código acima:

1. Para fazer vários gráficos em uma mesma janela foi alterado o parâmetro *mfrow* da função ***par***

```
par(mfrow=c(2,2))
```

O primeiro valor indica o número de linhas e o segundo o número de colunas em que sua janela gráfica será dividida. Para retornar ao padrão inicial de apenas um gráfico por janela basta digitar

```
par(mfrow=c(1,1))
```

ou fechar a janela do gráfico atual. Quando você chamar outro gráfico uma nova janela no padrão inicial vai se abrir

2. Para usar diferentes métodos de ligação foi alterado o argumento `method` da função `hclus`. Veja exemplos nos códigos da seção anterior.

Agora vamos fazer o dendrograma da análise de agrupamento para nossa amostra das comunidades discretas, usando o método de ligação média (UPMGA). Para isso:

1. Calcule a matriz de similaridade Bray-Curtis e guarde em um objeto (`sim.disc` no caso):

```
 sim.disc=sim(amost.disc, indice="bc")
```

2. Rode a análise de agrupamento e guarde em outro objeto do R (`clas.disc` no caso):

```
 clas.disc=hclust(as.dist(1-sim.disc), method="average")
```

3. Faça o dendrograma, aplicando a função `plot` ao objeto criado com a análise:

```
 plot(as.dendrogram(clas.disc, hang=-1), ylab="Bray-Curtis",main="Ligação
média")
```

E, por fim, você pode criar e comparar os dendrogramas obtidos com análises de agrupamento com os quatro diferentes métodos de ligação também para as comunidades discretas:

```
par(mfrow=c(2,2))
clas.disc1a=hclust(as.dist(1-sim.disc), method="single")
plot(as.dendrogram(clas.disc1a, hang=-1), ylab="Bray-Curtis", main="Ligação
simples")
clas.disc1b=hclust(as.dist(1-sim.disc), method="complete")
```

```
plot(as.dendrogram(clas.disc1b, hang=-1), ylab="Bray-Curtis", main="Ligação
completa")
clas.disc1c=hclust(as.dist(1-sim.disc), method="average")
plot(as.dendrogram(clas.disc1c, hang=-1), ylab="Bray-Curtis", main="Ligação
média")
clas.disc1d=hclust(as.dist(1-sim.disc), method="centroid")
plot(as.dendrogram(clas.disc1d, hang=-1), ylab="Bray-Curtis", main="Ligação
centroide", ylim=c(0,0.7))
```

Agora é só você...

1. Explique as diferenças entre os resultados das análises de agrupamentos com diferentes métodos de ligação aplicados à mesma amostra, como as que fizemos na seção anterior.
2. Compare os dendrogramas das comunidades contínuas e discretas usando um mesmo método de ligação e interprete as diferenças entre as duas comunidades em relação ao gradiente amostrado.

- Para responder a primeira pergunta você precisa entender os algoritmos de ligação. Veja a o ótimo resumo do curso de análises multivariadas de Alan Fielding.

Para saber mais

- Análise de agrupamento na Wikipedia
- Método de centroide na Wikipedia
- Clustering and Classification methods for Biologists: site do excelente curso de Alan Fielding.
- Manly, B. 2008. Métodos Estatísticos Multivariados: Uma Introdução. 3 Ed. Artmed, Porto Alegre. (*Uma das melhores introduções a técnicas multivariadas para biólogos*).
- Valentin, J. 2012. Ecologia Numérica: Uma Introdução à Análise Multivariada de Dados Ecológicos. Interciência, Rio de Janeiro. (*Outro ótimo texto introdutório*)
- Legendre, P., & Legendre, L. 2012. Numerical ecology. Elsevier, Amsterdan. (*o livro de refrência de ecologia numérica. Completo e didático, mas é uma leitura mais avançada*).

1)

http://pt.wikipedia.org/wiki/Algoritmo

2)

o valor na célula [1,4] é o mesmo da célula [4,1], lembra-se?

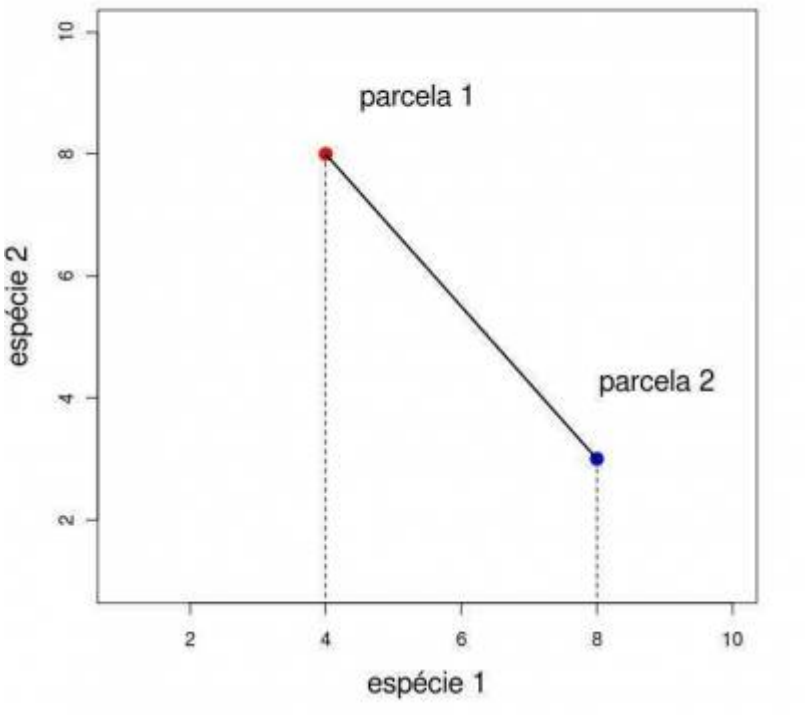
3)

fazemos isso colocando valores NA nessas posições da matriz. NA é o código do R para dado faltante (*Not available*)

-

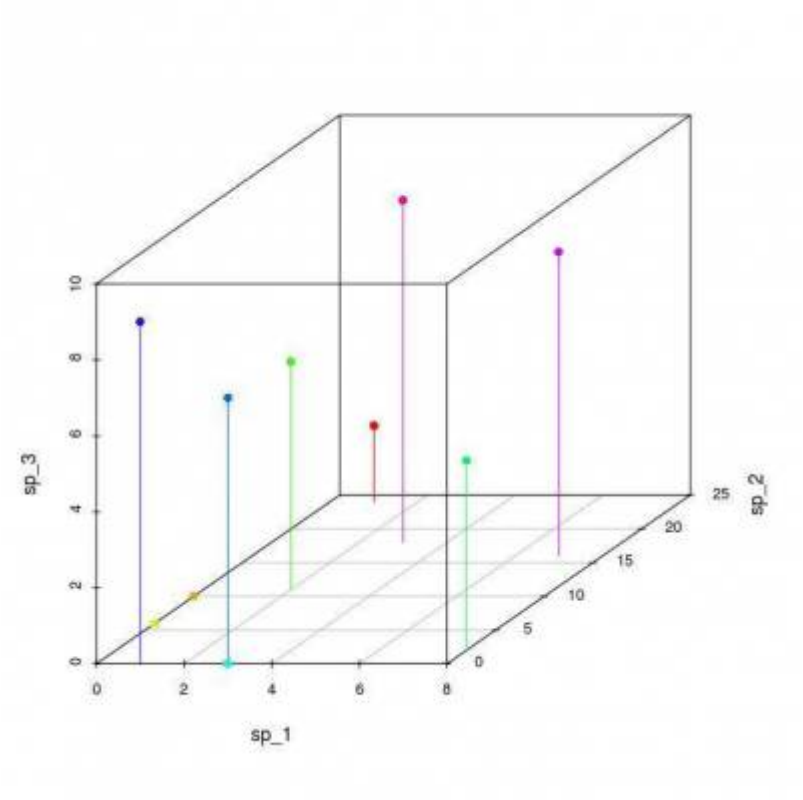
Introdução à ordenação multivariada

Ordenação é um método de redescrição dos dados multivariados de forma a apresentá-los em poucas dimensões, geralmente 2 ou 3, com a menor perda possível de informação. Veja o exemplo da representação gráfica de duas parcelas em um espaço dimensional de duas espécies.



Quando temos apenas duas dimensões de atributos (no caso duas espécies) a representação gráfica dos objetos, no nosso caso as parcelas, é direta.

Conforme vamos acrescentando informações de novos atributos (espécies) aos nossos objetos (parcelas), a representação gráfica torna-se mais difícil. Com três dimensões ainda conseguimos representar nossas parcelas em um gráfico.



Os muitos métodos de ordenação como Análise de Componentes Principais (PCA), Análise de Coordenadas Principais (PCO), Análise de Correspondência (CA), têm como objetivo a redução das dimensões descritivas para visualizar melhor as relações entre os objetos, quando são descritos por muitas medidas.

Para seguir em frente é necessário que você tenha na área de trabalho do R as amostras das comunidades virtuais montadas, como descrito no roteiro padrões de gradientes em comunidades. Certifique-se disso com o comando `ls()`, que lista os objetos da área de trabalho do R. Na lista resultante devem estar os objetos `amost.cont` e `amost.disc`:

```
> ls()
 [1] "amost"     "amost.cont"  "amost.disc"
```

Ordenação Polar

Foi um dos primeiros métodos de ordenação e devido a sua simplicidade foi amplamente usado por ecólogos. Apesar de suas limitações e de ter sido preterido por outros métodos mais sofisticados, a Ordenação Polar (OP) pode produzir resultados com interpretação ecológica relevante. Além disso, entender seu algorítmo simples é uma ótima maneira de entender a lógica geral da ordenação. O objetivo da OP é representar as parcelas em um sistema de coordenadas de forma que a distância entre as parcelas represente suas similaridades e que revelem o gradiente ambiental subjacente. Aqui vamos usar o algoritmo de OP desenvolvido por Bray e Curtis (1957), especificamente para a análise de dados de comunidades de plantas. Para iniciar a análise, primeiro calculamos a dissimilaridade de Bray-Curtis como nossa medida de distância[4)].

Abaixo uma função da função que calcula a dissimilaridade (distância) de Bary-Curtis entre todos os pares de parcelas. Para usar esta função, copie o código abaixo e cole-a na linha de comando do R.

```
dis.bc<-function(dados){
    nplot=dim(dados)[2]
    similar=matrix(NA,ncol=nplot,nrow=nplot)
    rownames(similar)<-paste("plot", c(1:nplot))
    colnames(similar)<-paste("plot", c(1:nplot))
    for(i in 1:(nplot-1)){
        m=i+1
        for(m in m:nplot){
            bc.dist=sum(
                abs(dados[,i]-dados[,m]))/(sum (dados[,c(i,m)])
                                           )
            similar[m,i]=similar[i,m]=bc.dist
            diag(similar)<-0
        }
    }
    return(round(similar,3))
}
```

Algoritmo da Ordenação Polar

- **1.** Use a função acima para produzir a matriz de dissimilaridade para a nossa amostra de comunidades contínuas:

```
dis1.cont=dis.bc(amost.cont)
```

- **2.** Calcule, para cada uma das parcelas, a soma das distâncias dela em relação às outras parcelas:

```
somadist1.cont=apply(dis1.cont, 1, sum, na.rm=TRUE)
somadist1.cont
```

- **3.** Assinalar a parcela com a maior soma de distâncias [5)] e guardar o seu valor para indicar o início do primeiro eixo (x). Essa será nossa primeira referência no espaço de descrição das

nossa parcelas (**ax**).

```
max(somadist1.cont)
nomes.parc = names(somadist1.cont)
parc.ax = nomes.parc[somadist1.cont==max(somadist1.cont)][1]
parc.ax
```

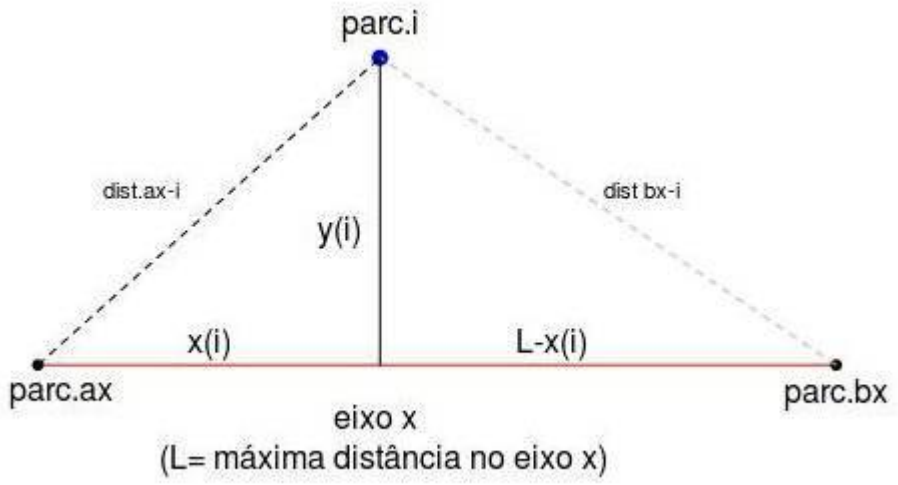
- **4.** Encontrar agora a parcela que tem a maior distância em relação à primeira referência. Essa será nossa segunda referência no nosso espaço de descrição (**bx**)

```
dist.ax=dis1.cont[,parc.ax]
dist.ax
max.ax=max(dist.ax)
max.ax
parc.bx=nomes.parc[dist.ax==max.ax]
parc.bx
```

- 4a. Algumas vezes várias parcelas têm valores iguais de distância em relação à parcela que definimos como **ax**. Nesse caso, ficaremos com aquela que tiver maior soma de distâncias em relação às outras parcelas. Para verificar se este critério de desempate é necessário fazemos assim:

```
somamax.bx=max(somadist1.cont[parc.bx])
parc.bx=parc.bx[somadist1.cont[parc.bx]==somamax.bx][1]
parc.bx
```

- **5.** Agora que temos duas referências do primeiro eixo (*parc.ax* e *parc.bx*), podemos calcular a posição de todas as outras parcelas nesse eixo x. Isso funciona como a triangulação que é usada em telemetria para encontrar um animal que está usando um emissor de rádio. Colocando receptores em dois pontos conhecidos e com a informação da distância do emissor (o animal com um rádio colar por exemplo) aos dois receptores, é possível calcular as coordenadas em que o animal está. Veja a representação gráfica do que estamos dizendo:



Para o cálculo utilizamos a equação de Beal (1965) que nada mais é do que resolver a equação da hipotenusa dos dois triângulos acima. Após diversos passos, chegamos finalmente à equação de Beal:

$$ x_i \ = \ \frac{ L^2 \ + \ dist_{ax_i}^2 \ - \ dist_{bx_i}^2 }{2L} $$

onde “L” é a distância entre *parc.ax* e *parc.bx* (distância máxima no eixo x), “$dist_{ax\_i}$” é a distância da parcela “i” em relação à *parc.ax*, e “$dist_{bx\_i}$” é a distância da parcela “i” em relação à *parc.bx*, conforme pode ser visualizado na figura acima.

- 5a. Agora que entendemos a equação de Beal, antes de aplicá-la, precisamos recuperar os valores de distância das outras parcelas em relação às nossas referências para o primeiro eixo: *parc.ax* e *parc.bx*. Para a *parc.ax* já organizamos em um passo anterior (passo 4) e está no objeto *dist.ax*. Você pode visualizar esses dados executando a primeira linha do código abaixo. Depois, você pode calcular e inspecionar os valores de distância das outras parcelas em relação à *parc.bx*, com os dois comandos seguintes:

```
  dist.ax
  dist.bx=dis1.cont[,parc.bx]
  dist.bx
```

- 5b. Agora aplique a equação de Beal a todas as parcelas para encontrar a posição delas no eixo x:

```
xi = (max.ax^2 + dist.ax^2 - dist.bx^2)/(2*max.ax)
xi
```

- **6.** Agora que temos os valores do eixo x de todas as parcelas podemos calcular o valor do segundo eixo dado pelo teorema de pitágoras:

$$ y_i = \sqrt{dist_{ax_i}^2 - x_i^2} $$

```
yi=sqrt((dist.ax)^2-xi^2)
yi
```

- 6a. Agora fazemos um ajuste do valor da *parc.bx* no eixo *yi*, conforme o código abaixo:

```
yi[parc.bx]=max.ax
yi
```

- **7.** Temos agora as coordenadas de cada parcal em dois eixos ortogonais. Vamos organizá-las em um único objeto e fazer um gráfico:

```
op1.cont=data.frame(xi,yi)
op1.cont
plot(op1.cont, pch=19, col=rainbow(length(xi)), xlab="Eixo 1", ylab="Eixo
2")
text(op1.cont+0.01, labels=rownames(op1.cont))
```

Função Ordenação Polar

Como somos muito legais, montamos uma função que faz todas essas operações para você não ter que fazer tudo novamente passo a passo. Veja abaixo:

```
ordena.polar=function(dist)
{
```

```
somadist1.cont=apply(dist, 1, sum, na.rm=TRUE) + apply(dist,2,sum,
na.rm=TRUE)
nomes.parc=names(somadist1.cont)
parc.ax=nomes.parc[somadist1.cont==max(somadist1.cont)][1]
dist.ax=dist[,parc.ax]
max.ax=max(dist.ax)
parc.bx=nomes.parc[dist.ax==max.ax]
    if(length(parc.bx)>1)
    {
      somamax.bx=max(somadist1.cont[parc.bx])
      parc.bx=nomes.parc[somadist1.cont==somamax.bx][1]
      parc.bx
    }
dist.bx=dist[,parc.bx]
xi= (max.ax^2 + dist.ax^2 - dist.bx^2)/(2*max.ax)
yi=sqrt((dist.ax)^2-xi^2)
yi[parc.bx]=max(dist.ax)
op.xy=data.frame(xi,yi)
opx=jitter(op.xy[,1],10)
opy=jitter(op.xy[,2],10)
plot(opx, opy, pch=19, col=rainbow(length(xi)), xlim=c(-0.1, 1.1),
ylim=c(-0.1,1.1), main="Ordenação Polar", sub="Distância Bray-Curtis")
text(opx-0.02,opy-0.02 , labels=paste("p",1:dim(dist)[1], sep=""), cex=0.7)
return(op.xy)
}
```

Vamos aplicar a função para os dados da comunidade virtual contínua e ver se nossos cálculos estão corretos.

```
ordena.polar(dis1.cont)
```

Agora vamos aplicá-la para as comunidades virtuais discretas também.

Primeiro, você precisa usar a função `disc.bc` para produzir a matriz de dissimilaridade para a nossa amostra de comunidades discretas:

```
dis1.disc=dis.bc(amost.disc)
```

Depois, basta aplicar a função `ordena.polar` na matriz de similaridade das comunidades discretas

```
ordena.polar(dis1.disc)
```

Agora é por sua conta!

Interprete os padrões observados tendo em vista as características que foram utilizadas para construir ambas comunidades.

Interpretação de Ordenações

Algumas dicas de interpretação de resultados de ordenações, não só da Ordenação Polar, baseadas no material produzido por Michael Palmer em seu, visualmente não muito atraente, mas ótimo site sobre ordenação:

1. A direção dos eixos é arbitrária e não deve afetar a interpretação;
2. A escala numérica do eixos não é muito útil para a interpretação (existem algumas exceções como DCA, em que a escala é em unidades de beta diversidade);
3. Na OP, assim como muitas outras técnicas de ordenação, a ordem dos eixos é importante. Assim, o eixo 1 é mais importante para a interpretação dos gradientes ambientais que o segundo;
4. A experiência prévia do sistema estudado e conhecimento da literatura pertinente são as ferramentas mais poderosas para a interpretação dos gradientes subjacentes ao padrão revelado pela ordenação;
5. A interpretação de outros eixos além dos dois primeiros (quando a análise produz) é possível e a decisão de onde parar é uma questão arbitrária e dependente da quantidade e qualidade dos dados. Em algumas técnicas há estatísticas que ajudam a tomar essa decisão;
6. É desejável que os eixos não sejam correlacionados, o que é garantido por algumas técnicas. Dessa forma é possível interpretar os eixos como gradientes diferentes.

Depois de passar pelas etapas de construir comunidades virtuais, amostrá-las e aplicar métodos de classificação e ordenação, acreditamos que vocês compreendam os princípios básicos dos métodos analíticos usados para a descrição de comunidades em ecologia. Como dito no início desses roteiros, existem muitos métodos diferentes para conhecer e muitas coisas ainda para resolver em relação a esses métodos. Um vasto mundo interessantíssimo!

Para saber mais

- Ordination Methods for Ecologists, Mike Palmer. (*um excelente panorama das técnicas de ordenação e seu uso em ecologia*).
- * Clustering and Classification methods for Biologists: *site do excelente curso de* Alan Fielding.
- Manly, B. 2008. Métodos Estatísticos Multivariados: Uma Introdução. 3 Ed. Artmed, Porto Alegre. (*Uma das melhores introduções a técnicas multivariadas para biólogos*).
- Prado, P.I. *et al.* 2002. Ordenação multivariada na ecologia e seu uso em ciências ambientais. Ambiente & Sociedade, (10), 69-83.
- Valentin, J. 2012. Ecologia Numérica: Uma Introdução à Análise Multivariada de Dados Ecológicos. Interciência, Rio de Janeiro. (*Outro ótimo texto introdutório*)
- Legendre, P., & Legendre, L. 2012. Numerical ecology. Elsevier, Amsterdan. (*o livro de refrência de ecologia numérica. Completo e didático, mas é uma leitura mais avançada*).

4)

para saber mais sobre medidas de similaridade e o índice de Bray-Curtis veja a a primeira parte de nosso roteiro sobre análise de agrupamento

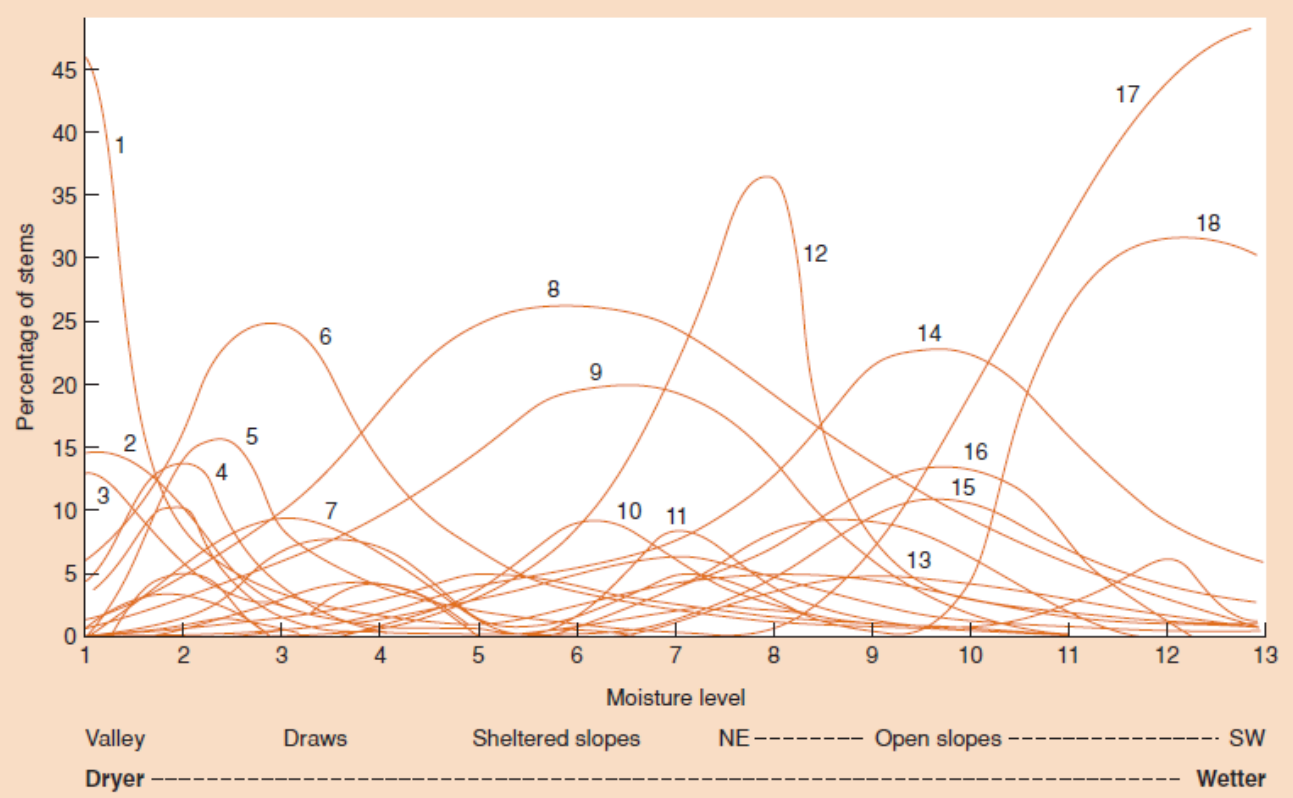
5)

ou seja, aquela mais diferente de todas as outras

-
- Versão longa

Padrões de gradientes em comunidades - simulação em R

Vamos construir uma comunidade virtual de plantas. Para isso vamos nos basear na distribuição de indivíduos em um gradiente ambiental. Partindo da premissa que as espécies tem uma distribuição normal de abundâncias ao longo do gradiente, podemos simular algo parecido com dados empíricos.



- **Figura 16.6** do livro de Ecologia de Begon et al. (2006), mostrando a distribuição de espécies vegetais em um gradiente de umidade nas Montanhas Great Smoky, Tennessee. Dados retirados do trabalho clássico de Robert Whittaker. Cada curva representa a porcentagem de caules de uma espécie em relação ao total de caules em um local. Note como a distribuição normal pode ser uma boa aproximação das abundâncias das espécies ao longo de um gradiente.

A distribuição normal, ou Gausiana, tem dois parâmetros que correspondem à média e ao desvio-padrão. A partir desses parâmetros podemos construir curvas teóricas da proporção dos indivíduos de cada espécie ao longo do gradiente.

Gráfico do Gradiente

No R podemos facilmente criar uma comunidade com espécies que estejam distribuídas ao longo do gradiente. Copie e cole o função abaixo na linha de comando do R:

```
graf.com=function(medias, desvios, minimo, maximo, leg=TRUE)
{
    dnorm.trunc=function(x, minimo=-Inf, maximo=Inf, media=0, desvio=1)
    {
    res=numeric(length(x))
    x.prov=dnorm(x,mean=media, sd=desvio)
    ampl.norm=pnorm(maximo,mean=media, sd=desvio)-pnorm(minimo,mean=media,
sd=desvio)
    x.prov/ampl.norm
    }
  nsp=length(medias)
  cor=rainbow(nsp)
  n.min=which.min(desvios)
  curve(dnorm.trunc(x, medias[n.min], desvios[n.min], maximo=maximo,
minimo=minimo),from=minimo, to=maximo, ylab="densidade da população",
xlab="valor do gradiente", main="Distribuição no gradiente", col=cor[n.min])
seqsp=1:nsp
seqsp=seqsp[-n.min]
    for (i in seqsp)
    {
    curve(dnorm.trunc(x, medias[i], desvios[i], maximo=maximo,
minimo=minimo),from=minimo, to=maximo,add=TRUE, col=cor[i])
    }
    if(leg==TRUE)
    {
     n.medias=medias + (maximo-minimo) * 0.05
    text(n.medias[n.min], dnorm.trunc(medias[n.min], medias[n.min],
desvios[n.min],maximo=maximo,minimo=minimo),
labels=paste("sp",n.min,sep="_"), col=cor[n.min], cex=.7)
    text(n.medias[-n.min], dnorm.trunc(medias[-n.min], medias[-n.min],
desvios[-n.min],maximo=maximo,minimo=minimo),
labels=(paste("sp",seqsp,sep="_")), col=cor[-n.min], cex=.7)
    }
}
```

Para criar o gráfico precisamos alimentar a função com as seguintes informações:

1. os valores ótimos para cada espécie [6)]
2. as variâncias das espécies [7)]
3. o valor mínimo do gradiente
4. o valor máximo do gradiente

Como usar a função??

Veja o exemplo abaixo:

```
graf.com(medias=c(2,3,4,5,6,7,8), desvios=c(1,1,1,1,1,1,1), minimo=0,
maximo=10)
```

Não parece uma comunidade muito realística. No R é muito fácil criar sequências de números aleatórios a partir de alguma função conhecida. Siga passo a passo os seguintes comandos no R e veja o que eles estão gerando:

```
s1=seq(from=1.5, to=19.5, by=0.25)
s1
med=sample(s1, size=10)
med
desv <-runif(10,0.5,2.5)
desv
```

Tudo bem, eu explico!

- A primeira linha cria um objeto chamado ***s1*** que é uma sequência de valores de 1,5 até 19,5 em intervalos de 0,25 [8)].
- A segunda linha só pede para mostrar o que o objeto ***s1*** contém.
- 3a. linha: faz uma amostra de 10 valores dentre os que estão em ***s1*** e guarda no objeto ***med***
- 4a. linha: mostra os valores do objeto ***med***
- 5a. linha: sorteia 10 valores, de um intervalo que vai de 0,5 até 2,5 e guarda esses valores no objeto ***desv***
- 6a. linha: mostra os valores do objeto ***desv***

Agora que criamos esses objetos, só precisamos executar a função que lê esses valores:

```
graf.com(medias=med, desvios=desv, minimo=0, maximo=20)
```

Dê uma olhada, como quem não quer nada, no gráfico do seu coleguinha ao lado. Por que o dele é mais bonito? Os gráficos não deveriam ser iguais já que ambos seguiram o mesmo roteiro, com o mesmo código? Rode os comandos abaixo, repetindo a segunda linha mais 3 vezes.

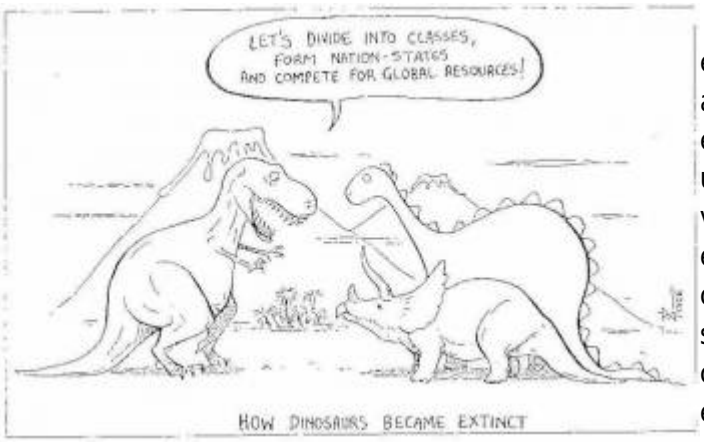
```
par(mfrow=c(2,2))
graf.com(medias=sample(2:19, size=10),desvios=sample(seq(from=0.5, to=2.5,
by=0.1),10), minimo=1, maximo=20)
```

1. A primeira linha do comando acima faz com que o dispositivo gráfico (uma janela gráfica) seja dividido em quatro partes, duas linhas e duas colunas. Para desligar, feche a janela gráfica que o padrão de um único gráfico por janela retorna.
2. A segunda linha é a função que já conhecem para criar o gráfico da comunidade no gradiente. Para repetir essa linha de comando no R é só apertar a tecla ↑ que o R mostra a última linha de comando executado, se apertar novamente, a penúltima... Portanto, não precisa digitar ou copiar o comando do gráfico várias

vezes, apenas aperte ⇑ e depois *Enter*

Você usou 4 vezes o mesmo comando para gerar comunidades virtuais. O que você observou?

Comunidade Discreta e Contínua



Lembram da história, lá nos primórdios da ecologia, sobre dois pesquisadores com visões antagônicas de como as comunidades eram estruturadas? Um deles, Frederic Clements, tinha uma visão muito organizada das comunidades vegetais, com uma forte dependência mútua entre as espécies no sistema. Para Clements as comunidades vegetais funcionavam como um superorganismo que nasce, cresce e morre. Em contraste, o botânico Henry Gleason, na mesma época entendia as comunidades como um resultado da interação das espécies com o ambiente, combinada com acontecimentos ao acaso. Legal, vamos brincar com essas ideias contrastantes em nosso gradiente ambiental virtual! Como reproduzir essas visões? Na primeira (mundo Clementsiano), as comunidades são discretas, ou seja, são compreendidas por um conjunto de espécies características que sempre ocorrem juntas. Já no mundo Gleisoniano as espécies têm limites de tolerância diferentes às condições ambientais e ocorrem independentemente das outras, ou seja, não há limites claros entre comunidades.

Primeiro vamos montar uma comunidade contínua com 40 espécies em um gradiente de 0 a 150:

1. sorteamos 40 valores de uma sequência de valores inteiros de 10 a 150, para representar o ótimo da espécie no gradiente
2. sorteamos 40 valores de uma sequência que começa em 4 e vai até 10, a cada 0.5, para definir os desvios de cada espécie
3. fazemos o gráfico da nossa comunidade virtual usando nossa função *graf.com*.

```
com.cont=sample(10:150, 40)
desv=sample(seq(from=4, to=10, by=0.5),40, replace=TRUE)
graf.com(medias=com.cont, desvios=desv, minimo=0, maximo=150)
```

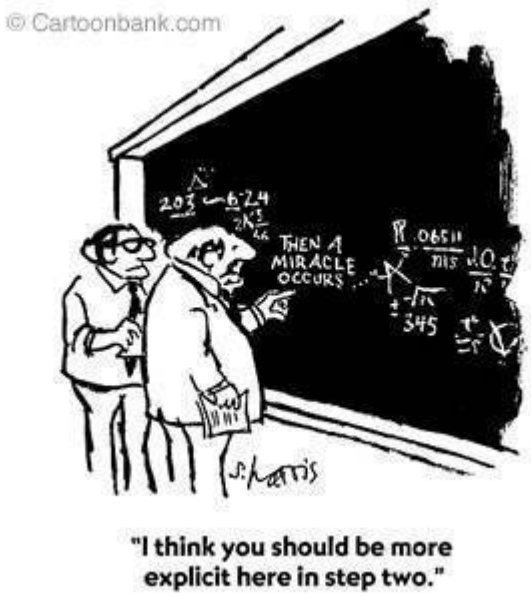
Gerar comunidades discretas ao longo do gradiente é um pouco mais complicado. Nossas espécies devem formar grupos, ou associações, ao longo do gradiente. Vamos formar quatro grupos centrados nos valores: 30, 60, 90 e 120 do nosso gradiente. Para cada grupo sorteamos 10 espécies com algum desvio em torno do valor central. Para isso, vamos seguir esses passos:

1. sorteamos 10 valores de uma distribuição normal com média 30 e desvio 5, para representar nossas primeiras 10 espécies
2. repetimos o passo um para as médias de 60, 90 e 120
3. usamos os mesmos valores de desvios da comunidade contínua (objeto ***desv*** criado acima) para as espécies
4. construímos o gráfico das espécies no gradiente com esses valores, usando a função *graf.com*.

```
com1=rnorm(10, mean=30, sd=5)
com2=rnorm(10, mean=60, sd=5)
com3=rnorm(10, mean=90, sd=5)
com4=rnorm(10, mean=120, sd=5)
com.disc=c(com1,com2,com3,com4)
graf.com(medias=com.disc, desvios=desv, minimo=0, maximo=150)
```

Vocês sentiram uma sensação de realização celestial? Agora temos poderes divinos... podemos criar nosso próprio mundo! Nada pode nos deter!!

Amostrando a Comunidade Virtual



Vamos retornar à nossa condição humana e seguir com um processo de amostragem da nossa comunidade. Vamos imaginar que esse gradiente existe e que as comunidades são exatamente essas que criamos, mas que não temos nenhuma informação prévia dos sistemas. Como somos humanos e queremos entender como as comunidades se estruturam, podemos nos fazer algumas perguntas, por exemplo:

1. A comunidade responde ao gradiente ambiental?
2. Se sim, essa resposta se dá por uma substituição gradual das espécies ao longo do gradiente ou por formação de subgrupos discretos de espécies em cada região do gradiente?

Para responder a essas perguntas, temos que ir ao sistema e conhecê-lo. Entretanto, não temos muito dinheiro nem tempo para estudar todas as espécies e todos os indivíduos que ocorrem ao longo de todo o gradiente, por isso tomamos a decisão de fazer uma amostragem. Como trata-se de um gradiente, podemos tomar a decisão de fazer essa amostra de forma sistemática em detrimento de uma completamente aleatória [9)]. Para isso vamos usar duas funções de amostragem abaixo *amostra.com* e *prob.ssp*. Copie e cole os códigos abaixo no R:

```
amostra.com=function(medias, desvios, amostra, n.ind=100, minimo=0,
maximo=150)
{
pnorm.trunc=function(x,minimo=-Inf, maximo=Inf, media=0, desvio=1)
{
denom <- pnorm(maximo, mean=media, sd=desvio) - pnorm(minimo, mean=media,
sd=desvio)
qtmp <- pnorm(x, mean=media, sd=desvio) - pnorm(minimo, mean=media,
sd=desvio)
qtmp/denom
}
nsp=length(medias)
namostra=length(amostra)
resulta=prob.resulta=matrix(0, nrow=nsp, ncol=namostra)
sp.name=paste("sp", 1:nsp, sep="_")
rownames(resulta)<-sp.name
colnames(resulta)=paste("plot", 1:namostra, sep="_")
  for(k in 1:namostra)
  {
    for(i in 1:nsp)
    {
    prob.resulta[i,k]= pnorm.trunc(amostra[k]+1,minimo=minimo,
maximo=maximo,media=medias[i], desvio=desvios[i])-
pnorm.trunc(amostra[k],minimo=minimo, maximo=maximo,media=medias[i],
desvio=desvios[i] )
    }
  s1=sample(sp.name, size=n.ind, prob=prob.resulta[,k], replace=TRUE)
  conta.s1=table(s1)
  pos.sp=match(names(conta.s1),sp.name)
  resulta[,k][pos.sp]<-conta.s1
  }
return(resulta)
}
```

Agora, vamos testar para ver se funciona. Lembre-se que já criamos os objetos que representam as medias e os desvios relacionados a cada uma das nossas comunidades. Para completar, temos que criar valores que representarão nossa amostra de cada comunidade. Entenda os passos:

1. criar uma sequencia de 14 valores iniciando no 10 e terminando no 140 a cada 10. Esses valores representam os pontos do gradiente em que nossa amostra será realizada.
2. amostrar a comunidade discreta a partir dos valores que usamos para criar os gráficos anteriores
3. fazer o mesmo para a comunidade contínua

O código é o seguinte:

```
amost=seq(10,140, by=10)
amost
amost.disc<-amostra.com(medias=com.disc, desvios=desv, amostra=amost)
head(amost.disc)
amost.cont<-amostra.com(medias=com.cont, desvios=desv, amostra=amost)
head(amost.cont)
```

Comparando a amostra com as comunidades criadas virtualmente

Até agora criamos nossas comunidades virtuais e fizemos uma amostra dela ao longo de um gradiente ambiental. Será que a amostra (composta pelos 14 pontos de amostragem) é fiel ao padrão expresso pela espécie no gradiente?[10)] Vamos colocar os valores lado a lado em um gráfico para investigar.

```
par(mfrow=c(2,2))
graf.com(medias=com.disc, desvios=desv, minimo=0, maximo=140)
matplot(amost,t(amost.disc), type="l", lty=2,
col=rainbow(dim(amost.disc)[1]), main="Amostra",xlab='valor do
gradiente',ylab='indivíduos por parcela' )
graf.com(medias=com.cont, desvios=desv, minimo=0, maximo=140)
matplot(amost,t(amost.cont), type="l", lty=2,
col=rainbow(dim(amost.cont)[1]), main="Amostra",xlab='valor do
gradiente',ylab='indivíduos por parcela' )
par(mfrow=c(1,1))
```

Agora temos uma amostra da comunidade discreta *amost.disc*, na qual as espécies ocorrem juntas em determinados pontos do gradiente e uma amostra da comunidade contínua *amost.cont*, na qual

as espécies apresentam seus ótimos aleatoriamente dispostos no gradiente. Podemos então usar métodos analíticos de descrição de comunidades usuais em ecologia para verificar se com essas amostras conseguimos identificar os padrões das comunidades virtuais que criamos. Podemos até entender melhor as limitações e utilidades dos métodos, definindo a sensibilidade dos métodos para situações que esperamos encontrar no campo. Para isso vamos fazer as duas outras atividades propostas para esse módulo.

Um exercício interessante para você fazer em um outro momento (aquela hora que você está sem fazer nada em casa e fica procurando algo interessante na internet, sabe?) é montar comunidades discretas e contínuas com seus próprios parâmetros (número de espécies, tamanho do gradiente, amostra etc...). Tente rever como foram criadas as comunidade contínua e discreta do nosso roteiro e refaça os passos para criar as suas com novos parâmetros.

Para saber mais

Ainda no mundo dos mortais

Depois de “brincar de Deus” e criar seu gradiente ambiental, vamos continuar na pele do ecólogo que vai analisar esse gradiente, tentando descobrir se há algum padrão.

Conheça as análises que ecólogos usam para descobrir padrões a partir das variações de abundâncias das espécies. Os roteiros abaixo aplicam essas análises nos dados que você criou neste exercício:

- Classificação por agrupamento, ou análise de *cluster*.
- Análise de ordenação

Decifre o código

Caso queira entender melhor as funções em R, veja a versão deste roteiro com comentários do autor:

- código comentado da comunidade virtual

R, comunidades, análise padrão

6) média da distribuição normal

7) quanto maior o valor, mais espalhados os indivíduos da espécie estarão no gradiente

8) a função se chama *seq*

9) isso é contestável, mas defensável

10) ou seja, será que nossa função funfa??

From:
http://ecovirtual.ib.usp.br/ -

Permanent link:
http://ecovirtual.ib.usp.br/doku.php?id=ecovirt:roteiro:comuni:comuni_virt1

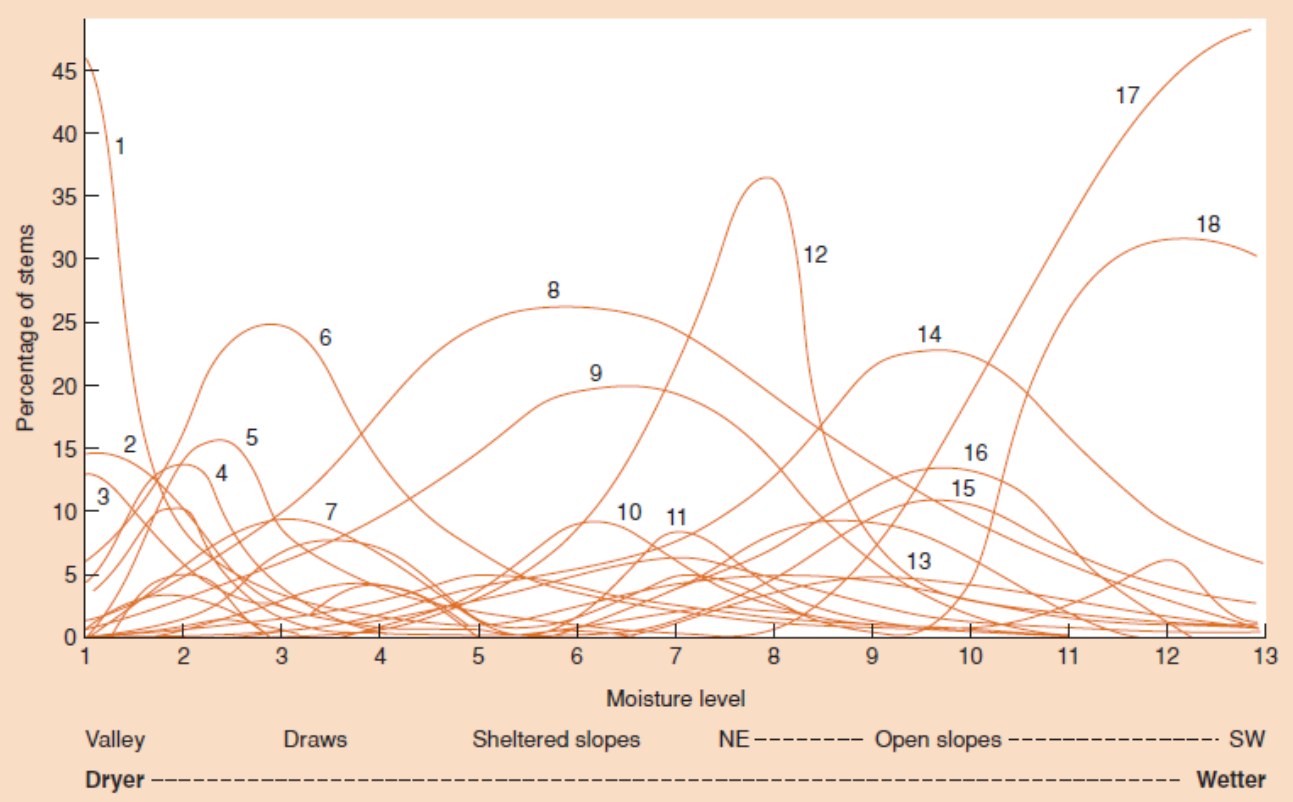
Last update: **2016/05/10 07:19**

-
- Versão longa

Padrões de gradientes em comunidades - simulação em R (código comentado)

Versão longa com construção dos códigos passo a passo
Caso queira o roteiro mais simples e direto vá para versão sem comentários ao código deste roteiro.

Vamos construir uma comunidade de plantas virtual. Para isso vamos nos basear na distribuição de indivíduos em um gradiente ambiental. Partindo da premissa que as espécies tem uma distribuição normal de abundâncias ao longo do gradiente, podemos simular algo parecido com dados empíricos.



- **Figura 16.6** do livro de Ecologia de Begon et al., mostrando a distribuição de espécies vegetais em um gradiente de umidade nas Montanhas Great Smoky, Tennessee. Dados retirados do trabalho clássico de R. Whittaker. Cada curva representa a porcentagem de caules de uma espécie em relação ao total de caules em um local. Note como a distribuição normal pode ser uma boa aproximação das abundâncias das espécies ao longo de um gradiente.

A distribuição normal, ou Gausiana, tem dois parâmetros, que correspondem à média e ao desvio-padrão. A partir desses parâmetros podemos construir curvas teóricas da proporção dos indivíduos de cada espécie ao longo do gradiente. Vamos ver como essas curvas são construídas no R. A função *curve()* desenha gráficos a partir de uma função matemática. No nosso caso a função é *dnorm()*, que também já está no R. Vamos ver como isso funciona!

Veja como fazer um gráfico de uma normal com média 0 e desvio 1[11]:

```
curve(dnorm(x, mean=0,sd=1), from=-5, to= 5)
curve(dnorm(x, mean=0,sd=1), from=-5, to= 5, ylim=c(0,1)) ## ylim padroniza
a escala y
```

Vamos agora adicionar outra curva a esse primeiro gráfico, mas agora com outros desvios:

```
curve(dnorm(x, mean=0,sd=0.5), from=-5, to= 5, add=TRUE, col="red")
curve(dnorm(x, mean=0,sd=2.5), from=-5, to= 5, add=TRUE, col="blue")
curve(dnorm(x, mean=-2,sd=0.7), from=-5, to= 5, add=TRUE, col="green")
curve(dnorm(x, mean=2,sd=1.5), from=-5, to= 5, add=TRUE, col="orange")
```

Veja como essas curvas são similares às do gráfico acima de distribuição de espécies de plantas em um gradiente de umidade nas Montanhas Great Smoky.

Vamos agora criar nosso gradientes e a distribuição das espécies nele. Para começar, nossa comunidade terá 10 espécies distribuídas aleatoriamente em um gradiente 1 a 20. Logo, o ótimo [12] para cada uma das espécies poderá ser qualquer valor entre 1,5 e 19,5 (vamos eliminar o extremo do gradiente para facilitar apenas). Esse valor nada mais é do que a média da sua distribuição de abundância. Para evitar sobreposição integral de nicho e criar um limite onde a coexistência de espécies não é possível, vamos sortear as médias de uma sequência de valores discretos ao longo do gradiente, sem reposição. No nosso exemplo adotamos um sequência de valores espaçados em 0,25. Já usamos as funções para sortear valores de um vetor usando *sample* e para produzir sequências numéricas *seq*. Caso queira saber mais sobre as funções no R peça socorro (help!)[13].

```
s1=seq(from=1.5, to=19.5, by=0.25)
s1
medias=sample(s1, size=10)
medias
```

Legal! Precisamos agora decidir os desvios... vamos sortear agora valores aleatórios de uma distribuição uniforme entre 0,5 até 2,5. Por quê? Porque podemos, esse é o nosso mundo e nele somos o criador!

```
desvios <-runif(10,0.5,2.5)
desvios
```

Agora temos médias e desvios das nossas 10 espécies. Ótimo! Vamos construir nossa comunidade no gradiente. Vamos fazê-lo para a primeira espécie:

```
medias[1]
desvios[1]
curve(dnorm(x, medias[1], desvios[1]),from=1, to=20)
```

Vamos agora só ajustar a escala y, para que todas as curvas sejam exibidas:

```
curve(dnorm(x, medias[1], desvios[1]),from=1, to=20,ylim=c(0,1))
```

Agora adicionando a espécie 2:

```
curve(dnorm(x, medias[2], desvios[2]),from=1, to=20,add=TRUE, col=2)
```

Adicione as demais com o mesmo código acima, mudando o valor 2 para 3 e assim por diante até incluir as 10 espécies.

```
curve(dnorm(x, medias[3], desvios[3]),from=1, to=20,add=TRUE, col=3)
...
...
```

Parece tudo ótimo... mas há um problema! Veja como as espécies que apresentam médias próximas ao limite do gradiente tem sua distribuição truncada. Isso faz com que a probabilidade total (área sob a curva) seja menor que um. A distribuição normal teórica vai de menos infinito a mais infinito. Quando restringimos o gradiente para valores finitos a distribuição precisa ser ajustada, truncando-a no intervalo desejado. Vamos usar a função abaixo para fazer esse ajuste. Copie e cole todo o código abaixo no R.

```
dnorm.trunc=function(x, minimo=-Inf, maximo=Inf, media=0, desvio=1)
{
res=numeric(length(x))
x.prov=dnorm(x,mean=media, sd=desvio)
ampl.norm=pnorm(maximo,mean=media, sd=desvio)-pnorm(minimo,mean=media,
sd=desvio)
x.prov/ampl.norm
}
```

Agora faça novamente o gráfico da nossa comunidade no gradiente, com o truncamento. Para isso é só usar a nova função dnorm.trunc ao invez do *dnorm*:

```
  curve(dnorm.trunc(x,  minimo=1, maximo=20, media= medias[1],desvio=
desvios[1]),from=1, to=20,ylim=c(0,1))
  curve(dnorm.trunc(x,  minimo=1, maximo=20, media= medias[2],desvio=
desvios[2]),from=1, to=20,ylim=c(0,1), col=2, add=TRUE)
 curve(dnorm.trunc(x,  minimo=1, maximo=20, media= medias[3],desvio=
desvios[3]),from=1, to=20,ylim=c(0,1), col=3, add=TRUE)
  ...
   curve(dnorm.trunc(x,  minimo=1, maximo=20, media= medias[10],desvio=
desvios[10]),from=1, to=20,ylim=c(0,1), col=3, add=TRUE)
```

Isso foi legal, mas para simular uma comunidade maior seria tedioso ficar copiando linhas e mudando o valor do indexador dos objetos *medias* e *desvios*. Veja o quadro abaixo para ver como automatizar essa tarefa.

Fazendo ciclos de tarefas no R

Para tarefas tediosas e repetitivas podemos usar o R para nos ajudar. No caso do gráfico acima,

podemos automatizar o código para que ele repita a tarefa mudando apenas os valore que queremos a cada ciclo. Para isso usamos o *for()* dessa forma:

```
curve(dnorm.trunc(x, minimo=1, maximo=20, media= medias[1],
desvio=desvios[1]),from=1, to=20,ylim=c(0,1))
for (i in 2:10)
 {
 curve(dnorm.trunc(x, minimo=1, maximo=20,
media=medias[i],desvio=desvios[i]),from=1, to=20,add=TRUE, col=i)
 }
```

Vamos melhorar ainda mais, colocando legendas e título no gráfico:

```
curve(dnorm.trunc(x, medias[1], desvios[1], maximo=20, minimo=1),from=1,
to=20,ylim=c(0,1), ylab="densidade da população", xlab="valor do
gradiente", main="Distribuição de populações ao longo de gradiente")
for (i in 2:10)
 {
 curve(dnorm.trunc(x, medias[i], desvios[i], maximo=20, minimo=1),from=1,
to=20,add=TRUE, col=i)
 }
text(medias+1, dnorm.trunc(medias, medias, desvios,maximo=20,minimo=1),
labels=(paste("sp",1:10,sep="_")), col=1:10)
```

Para automatizar completamente o gráfico podemos colocar esses comandos concatenados em uma função. Incluindo também a função dnorm.trunc e um parâmetro para colocar ou não a legenda das espécies. Assim sempre que desejarmos podemos montar o gráfico apenas usando a mesma função. Note que ao colar uma vez a função no R, não há necessidade de colar novamente, precisa apenas chamá-la. Veja a função final abaixo:

```
graf.com=function(medias, desvios, minimo, maximo, leg=TRUE)
{
    dnorm.trunc=function(x, minimo=-Inf, maximo=Inf, media=0, desvio=1)
    {
    res=numeric(length(x))
    x.prov=dnorm(x,mean=media, sd=desvio)
    ampl.norm=pnorm(maximo,mean=media, sd=desvio)-
pnorm(minimo,mean=media, sd=desvio)
    x.prov/ampl.norm
    }
  nsp=length(medias)
  cor=rainbow(nsp)
  n.min=which.min(desvios)
  curve(dnorm.trunc(x, medias[n.min], desvios[n.min], maximo=maximo,
minimo=minimo),from=minimo, to=maximo, ylab="densidade da população",
xlab="valor do gradiente", main="Distribuição no gradiente", col=cor[1])
seqsp=1:nsp
seqsp=seqsp[-n.min]
    for (i in seqsp)
    {
    curve(dnorm.trunc(x, medias[i], desvios[i], maximo=maximo,
minimo=minimo),from=minimo, to=maximo,add=TRUE, col=cor[i])
    }
    if(leg==TRUE)
    {
    text(medias+1, dnorm.trunc(medias, medias,
```

```
desvios,maximo=maximo,minimo=minimo),
labels=(paste("sp",(1:(nsp)),sep="_")), col=cor, cex=.7)
    }
}
```

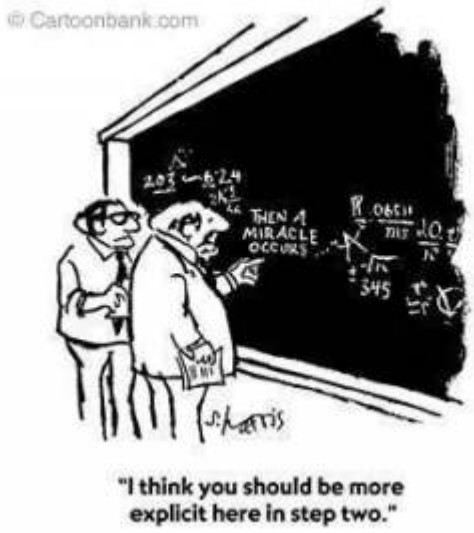
Agora teste o gráfico:

```
graf.com(medias=medias, desvios=desvios, minimo=1, maximo=20)
```

Dê uma olhada, como quem não quer nada, no gráfico do seu coleguinha ao lado. Por que o dele é mais bonito? Os gráficos não deveriam ser iguais já que ambos seguiram o mesmo roteiro com o mesmo código? Rode os comandos abaixo,depois repita o comando dos gráficos para preencher as quatro janelas do dispositivo.

```
par(mfrow=c(2,2))
graf.com(medias=sample(2:19, size=10),desvios=sample(seq(from=0.5, to=2.5,
by=0.1),10), minimo=1, maximo=20)
```

Amostrando a Comunidade Virtual



Agora vamos avançar seguindo um processo de amostragem da nossa comunidade. Vamos imaginar que esse gradiente existe e que a comunidade é exatamente essa que construímos. Sem nenhuma informação prévia do sistema, há duas perguntas sobre estrutura da comunidade:

1. A comunidade responde ao gradiente ambiental?
2. Se sim, esta resposta se dá por uma substituição gradual das espécies ao longo do gradiente ou por formação de sugrupos discretos de espécies em cada região do gradiente?

Para isso podemos fazer uma amostra em diferentes pontos do suposto[14)] gradiente e comparar as comunidades nessas amostras. Vamos primeiro fazer uma amostra de parcelas quadradas de lado 1 ao longo de nosso gradiente. Note que nosso gradiente começa em 1 e termina em 20, então podemos colocar até 19 parcelas sem sobreposição. Mas vamos colocar apenas dez, deixando um espaço de comprimento um entre elas. Como trata-se de um gradiente, podemos tomar a decisão de fazer esta amostra sistemática em detrimento de uma completamente aleatória [15)]. Primeiro

definimos as parcelas criando valores entre 1 e 19, com intervalo de 2:

```
amostra=seq(from=1, to=19, by=2)
amostra
```

A proporção dos indivíduos de cada espécie que estão na parcela é dada pela sua curva normal, definida acima[16)]. Por exemplo, a proporção dos indivíduos da espécie 1 esperada para a parcela 1 é:

```
p1sp1= pnorm(amostra[1]+1,mean=medias[1], sd=desvios[1] )-
pnorm(amostra[1],mean=medias[1], sd=desvios[1] )
p2sp1=pnorm(amostra[2]+1,mean=medias[1], sd=desvios[1] )-
pnorm(amostra[2],mean=medias[1], sd=desvios[1] )
#...
p10sp1=pnorm(amostra[10]+1,mean=medias[1], sd=desvios[1] )-
pnorm(amostra[10], mean=medias[1], sd=desvios[1] )
```

Da mesma forma, para a espécie 2 as proporções são:

```
p1sp2= pnorm(amostra[1]+1,mean=medias[2], sd=desvios[2])-
pnorm(amostra[1],mean=medias[2], sd=desvios[2] )
p2sp2= pnorm(amostra[2]+1,mean=medias[2], sd=desvios[2] )-
pnorm(amostra[2],mean=medias[2], sd=desvios[2] )
#...
p10sp2=pnorm(amostra[10]+1,mean=medias[2], sd=desvios[2] )-
pnorm(amostra[10],mean=medias[2], sd=desvios[2] )
```

Aqui também temos o problema do truncamento da distribuição e para resolve-los fazemos o ajuste com a função abaixo:

```
pnorm.trunc=function(x,minimo=-Inf, maximo=Inf, media=0, desvio=1)
{
denom <- pnorm(maximo, mean=media, sd=desvio) - pnorm(minimo, mean=media,
sd=desvio)
qtmp <- pnorm(x, mean=media, sd=desvio) - pnorm(minimo, mean=media,
sd=desvio)
qtmp/denom
}
```

Seria tedioso refazer tudo para todas as espécies e mesmo para um conjunto maior de espécies e amostra, por isso vamos fazer uma função e automatizar toda a tarefa.

```
prob.ssp=function(medias, desvios, amostra, minimo=1, maximo=20)
{
nsp=length(medias)
namostra=length(amostra)
resulta=matrix(NA, nrow=nsp, ncol=namostra)
rownames(resulta)=paste("sp", 1:nsp, sep="_")
colnames(resulta)=paste("plot", 1:namostra, sep="_")
  for(k in 1:namostra)
  {
```

```
    for(i in 1:nsp)
    {
    resulta[i,k]= pnorm.trunc(amostra[k]+1,minimo=minimo, maximo=maximo,
media=medias[i], desvio=desvios[i])- pnorm.trunc(amostra[k],minimo=minimo,
maximo=maximo, media=medias[i], desvio=desvios[i] )
    }
  }
return(resulta)
}
```

Vamos testá-la:

```
amostra.prob01=prob.ssp(medias,desvios, amostra)
amostra.prob01
```

Parece que funciona! Mas como combinar as proporções de indivíduos da espécie que devem ocorrer na parcela, para calcular o número de indivíduos de cada espécie? Se presupormos que todas as populações tem tamanhos iguais, os valores obtidos com a curva normal são proporcionais ao número de indivíduos de cada espécie na amostra[17)].

Agora, vamos transformar isso em valores de número de indivíduos de cada espécie na amostra. Para tanto precisamos nos valer de outra premissa: cada amostra comporta o mesmo número de indivíduos. Essa premissa é razoável para um gradiente que não envolve mudanças muito drásticas, mas pode ser problemático se estamos pensando em um gradiente que vai de um campo limpo a uma floresta fechada. Por enquanto ficamos com ele para não complicar a vida virtual de nossa comunidade. Para a primeira parcela se tivessemos 25 indivíduos na amostra, vamos sortear as espécies:

```
  sp.name=rownames(amostra.prob01)
  sp.name
  s1=sample(sp.name, size=25, prob=amostra.prob01[,1], replace=TRUE)
  s1
  conta.s1=table(s1)
  conta.s1
```

Estamos quase lá! Vamos agora montar nossa amostra com a função abaixo que apenas junta tudo que exercitamos acima:

```
amostra.com=function(medias, desvios, amostra, n.ind=25,minimo=1,
maximo=20)
{
nsp=length(medias)
namostra=length(amostra)
resulta=prob.resulta=matrix(0, nrow=nsp, ncol=namostra)
sp.name=paste("sp", 1:nsp, sep="_")
rownames(resulta)<-sp.name
colnames(resulta)=paste("plot", 1:namostra, sep="_")
  for(k in 1:namostra)
  {
    for(i in 1:nsp)
    {
```

```
    prob.resulta[i,k]= pnorm.trunc(amostra[k]+1,minimo=minimo,
maximo=maximo,media=medias[i], desvio=desvios[i])-
pnorm.trunc(amostra[k],minimo=minimo, maximo=maximo,media=medias[i],
desvio=desvios[i] )
    }
  s1=sample(sp.name, size=n.ind, prob=prob.resulta[,k], replace=TRUE)
  conta.s1=table(s1)
  pos.sp=match(names(conta.s1),sp.name)
  resulta[,k][pos.sp]<-conta.s1
  }
return(resulta)
}
```

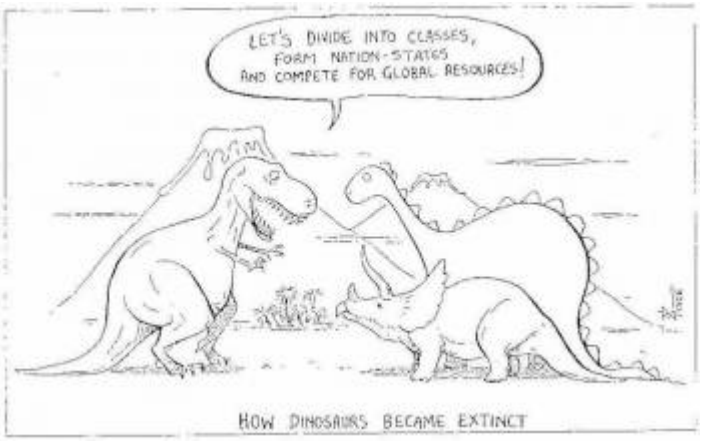
Vamos testar agora para ver se funciona. Lembre-se que já criamos os objetos medias, desvios e amostra, que são também argumentos da nossa função. Não confunda um com outro, apesar de mesmo nome, um é argumento que só existe no interior da função e outro é objeto que está na sua área de trabalho[18)].

```
com1.cont=amostra.com(medias=medias, desvios=desvios, amostra=amostra)
com1.cont
```

Vamos ver quantos indivíduos cada espécie tem em cada unidade amostral e também se cada U.A tem 25 indivíduos que é o padrão da função.

```
apply(com1.cont,1,sum)
apply(com1.cont,2,sum)
```

Comunidade Discreta



Agora sabendo que está funcionando, vamos montar uma comunidades discretas ao longo do gradiente. Para isso nossas espécies devem formar grupos, ou associações, ao longo do gradiente. Vamos imaginar 15 espécies em três grupos de cinco.

- 1. Fazemos um amostra de cinco valores de um ponto do gradiente com uma certa variação: essas serão as primeiras 5 médias das espécies
- 2. Repetimos isso duas vezes mais para outros pontos do gradiente e juntamos os valores para compor nosso objeto *med1*
- 3. Sorteamos valores de desvio para cada espécie e guardamos no nosso objeto *desv1*

- 4. Construimos o gráfico do gradiente
- 5. Criamos o objeto *amost1* designando em que pontos do gradiente vamos amostrar a comunidade
- 4. Rodamos a função *amostra.com* a função com os objetos formados

```
med1a=rnorm(5, mean=5, sd=1.5)
med1b=rnorm(5, mean=10, sd=1.5)
med1c=rnorm(5, mean=15, sd=1.5)
med1=c(med1a,med1b,med1c)
med1
med1<1.5
med1[med1<1.5]<-1.5
med1[med1>19.5]<-19.5
desv1=sample(seq(from=0.5, to=2.5, by=0.1),15)
graf.com(medias=med1, desvios=desv1, minimo=1, maximo=20)
amost1=seq(1.5,19.5,by=1.5)
com1.disc=amostra.com(medias=med1, desvios=desv1, amostra=amost1,
n.ind=25,minimo=1, maximo=20)
apply(com1.disc,1, sum)
apply(com1.disc,2, sum)
```

Comparando a Amostra com a População

Até agora criamos uma comunidade e fizemos uma amostra dela ao longo de um gradiente ambiental. Será que a amostra é fiel ao padrão expresso pela espécie no gradiente?[19)] Vamos colocar os valores lada a lado em um gráfico para investigar.

```
par(mfrow=c(2,2))
graf.com(medias=medias, desvios=desvios, minimo=1, maximo=20)
matplot(amostra,t(com1.cont), type="l", lty=2,
col=rainbow(dim(com1.cont)[1]), main="Amostra",xlab='valor do
gradiente',ylab='indivíduos por parcela' )
graf.com(medias=med1, desvios=desv1, minimo=1, maximo=20)
matplot(amost1,t(com1.disc), type="l", lty=2,
col=rainbow(dim(com1.disc)[1]), main="Amostra",xlab='valor do
gradiente',ylab='indivíduos por parcela' )
```

Agora temos uma comunidade discreta **com1.disc** e uma onde as espécies apresentam seus ótimos aleatoriamente dispostas no gradiente **com1.cont**. Agora podemos verificar se e como os nossos métodos de descrição da comunidade respondem às duas perguntas colocadas no começo do roteiro. Podemos até entender melhor suas limitações e utilidades, definindo a sensibilidade dos métodos para situações que esperamos encontrar no campo. Agora vocês tem uma ferramente poderosa para entender a descrição da comunidade, passando por descritores como índices de diversidade, até técnicas mais complexas como métodos de classificação e ordenação. Vamos fazer isso, mas antes monte sua comunidade discreta e contínua no gradiente com seus próprios parâmetros (número de espécies, tamanho do gradiente, amostra etc...). Tente rever como foram criadas as comunidade continua e discreta do nosso roteiro e refaça os passos para criar a sua com novos parâmetros.

11)

note que temos duas funções uma dentro da outra

12)

onde a espécie apresenta a maior proporção de seus indivíduos no gradiente

13)

para acessar o help do R use a função help!: *help(sample)* ou simplesmente digite *?rep*

14)

lembre-se que quem está amostrando não tem certeza do gradiente, mas quer testar se ele existe

15)

isso é contestável, mas defensável

16)

mais precisamente, é dada pela área acumulada sob a curva normal na extensão coberta pela parcela

17)

para ajustar essas probabilidades para populações de tamanhos diferentes é só multiplicar o amostra.prob01 por um vetor com os valores do tamanho da população de cada espécie na mesma ordem

18)

parece complicado?!É só a lógica da linguagem!

19)

será que nossa função funfa??

Comunidades

Estrutura de Comunidades

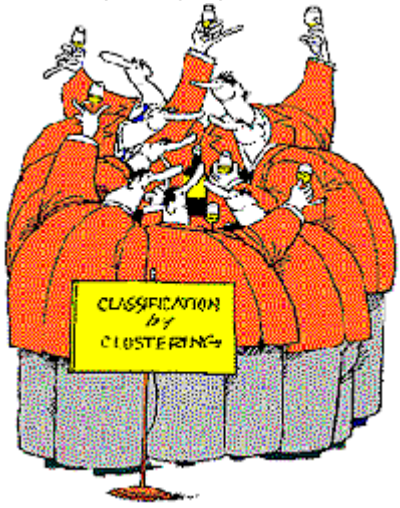
Nos três exercícios desta seção iremos montar uma comunidade hipotética ao longo de um gradiente ambiental para depois utilizar as ferramentas de classificação e ordenação de comunidades e ver como elas se comportam. A ideia é contrastarmos duas visões concorrentes da ecologia de comunidades : (1) as comunidades tem limites definidos e (2) há uma transição gradual na substituição de espécies ao longo de gradientes ambientais.

Comunidade Virtual

Vamos construir comunidade vegetais virtuais baseadas nas ideias de “superorganismo” de Frederic Clements (comunidades com limites bem definidos) e no conceito individualístico de Henry Gleason (comunidades sem limites claramente definidos).

- Roteiro Comunidade Virtual

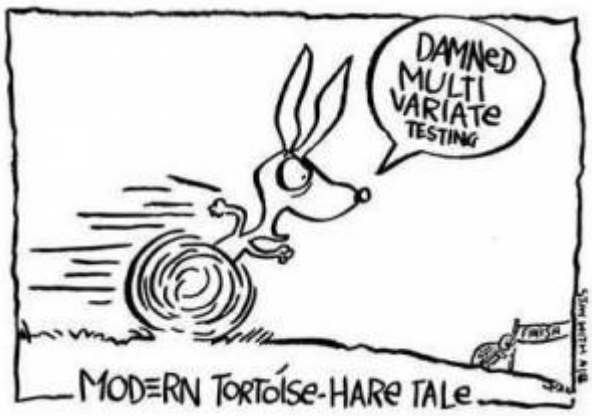
Classificação por Agrupamento



Os métodos de classificação agrupam objetos conforme a similaridade de seus atributos. Servem para identificarmos conjuntos de objetos mais similares que, no nosso caso, são parcelas de plantas. Entretanto, o método pode ser usado também para outros fins nobres, como classificar whisky escocês single malt. Esse trabalho foi publicado na revista Applied Statistics (1994) 43, No. 1, pp. 237-257, uma das mais conceituadas revistas na área de Matemática e Estatística, tendo como autor um dos mais renomados ecólogos da atualidade, Pierre Legendre, da Universidade de Montreal no Canadá. Ele e seu filho publicaram um dos livros que é referência para quem está interessado em métodos quantitativos em ecologia: “Numerical Ecology”. Este exercício é dedicado a ele.

- Roteiro de Classificação por agrupamento

Ordenação



Ordenação é um método de redescrição de dados multivariados de forma a apresentá-los em poucas dimensões, geralmente 2 ou 3, com perda mínima de informação. Vamos usar um dos primeiros métodos de ordenação descritos - a ordenação polar. Esse método foi desenvolvido por John Thomas Curtis e serviu como forma de revelar gradientes em seus estudos com comunidades de plantas em Wisconsin, considerados como evidências contra a teoria de comunidades como superorganismo de Clements.

- Roteiro Ordenação

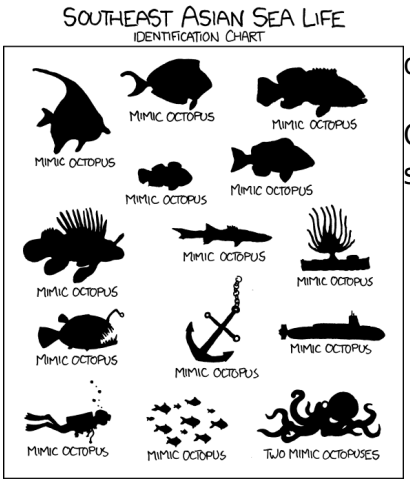
Partição da Variação

- Partição univariada: nesse primeiro roteiro utilizamos uma única espécie para ilustrar como a partição da variância pode nos informar sobre os processos relacionados à abundância da espécie.
- Partição multivariada: roteiro que generaliza para múltiplas espécies o método apresentado no roteiro anterior.

Distúrbio e Sucessão

Distúrbios ou perturbações naturais foram considerados por muito tempo um fator exógeno e raro nos sistema naturais, mas hoje estão plenamente incorporados como um importante fator na organização das comunidades. Além disso, o estudo das perturbações adquiriu maior importância com o aumento de sua intensidade e frequência pela ação do homem. Após perturbações, as comunidades vegetais tendem a retornar a um estado similar ao inicial, o que chamamos de sucessão ecológica. Vamos abordar aqui as relações entre perturbação e sucessão, focando a coexistência e demanda conflitante colonização/competição em modelos simples de dinâmica de comunidade.

Diversidade e Estabilidade



Comunidades com mais espécies são mais estáveis? Na década de 1970 Robert May mostrou que o contrário pode acontecer.

Conheça o método usado por May para avaliar a estabilidade de sistemas dinâmicos e reproduza seus resultados.

- Roteiro Diversidade e Estabilidade

Distúrbio e Coexistência

- Roteiro Distúrbio e Coexistência

Distúrbio e Tradeoff: Demandas Conflitantes

O conceito de demandas conflitantes (*tradeoff* no original em inglês) é muito importante para a ecologia e evolução. No contexto ecológico está relacionado, por exemplo, a restrições energética que impedem um indivíduo de investir em várias estratégias ótimas simultaneamente, como crescer e reproduzir. No contexto evolutivo está relacionado à seleção de estratégias ecológicas em detrimento de outras também eficiente, por exemplo o tamanho de frutos e a quantidade produzida por evento reprodutivo. Aqui apresentamos o conceito associado a diferentes regimes de distúrbios.

- Roteiro Demandas Conflitantes

Tipos de Sucessão Ecológica

- Roteiro Tipos de Sucessão Ecológica

Nicho de Regeneração

- Nicho de Regeneração

Dinâmicas Neutras

Modelos neutros em ecologia partem da premissa de que todas as espécie são competitivamente equivalentes, ao contrário dos modelos baseados em nicho. Vamos examinar duas das teorias neutras mais importantes em ecologia, ambas publicados na influente série de monografias MPB, da Universidade de Princenton.

Biogeografia de ilhas

Além da teoria neutra de evolução molecular de Motoo Kimura de 1968, Hubbell baseou-se fortemente na teoria de biogeografia de ilhas de MacArthur e Wilson. A teoria de biogeografia de ilhas foi criada para explicar um padrão muito recorrente na natureza: a relação entre o número de espécies de um local e sua área. A teoria mais aceita até então, baseada na idéia de que cada espécie possui um nicho, propunha que ilhas maiores tinham maior riqueza de espécies porque continham maior diversidade de habitats. Já a teoria de MacArthur e Wilson propõe que o número de espécies de uma ilha é determinado apenas pela taxa de extinção das espécies já presentes na ilha e pela taxa de imigração de espécies vindas do continente.

- Roteiro Biogeoografia de Ilhas

Teoria Neutra da Biodiversidade

Stephen Hubbell partiu da Teoria de Biogeografia de Ilhas e de sua vasta experiência com dinâmica de florestas tropicais para propor um processo simples de nascimentos e mortes que explicaria a grande diversidade nos trópicos. Conheça mais sobre a polêmica “Teoria Neutra Unificada da Biodiversidade e Biogeografia”, onde as espécies são equivalentes.

- Roteiro Teoria Neutra da Biodiversidade

Índice

Roteiros

- Colonização Interna
- Efeito Resgate

Duas Espécies

- Coexistência em Metapopulações
- Destruição de Habitat

Comunidades

- Introdução

Estrutura

- Comunidade Virtual
- Classificação por agrupamento
- Ordenação
- Partição univariada
- Partição multivariada

Dinâmica e Distúrbio

- Diversidade e Estabilidade
- Distúrbio e Coexistência
- Demandas Conflitantes
- Sucessão Ecológica
- Nicho de Regeneração

Dinâmicas Neutras

- Biogeoografia de Ilhas
- Teoria Neutra da Biodiversidade

Matemática e Estatística

- Introdução

Cálculo Integral e Diferencial

- Taxas de crescimento, derivadas e função exponencial
- Antiderivadas e integral definida
- Introdução a equações diferenciais
- Integração numérica de equações diferenciais
- Análise de estabilidade

Processos Estocásticos

- Caminhada aleatória em uma dimensão
- Dinâmica de soma zero

Links Externos

Visitantes