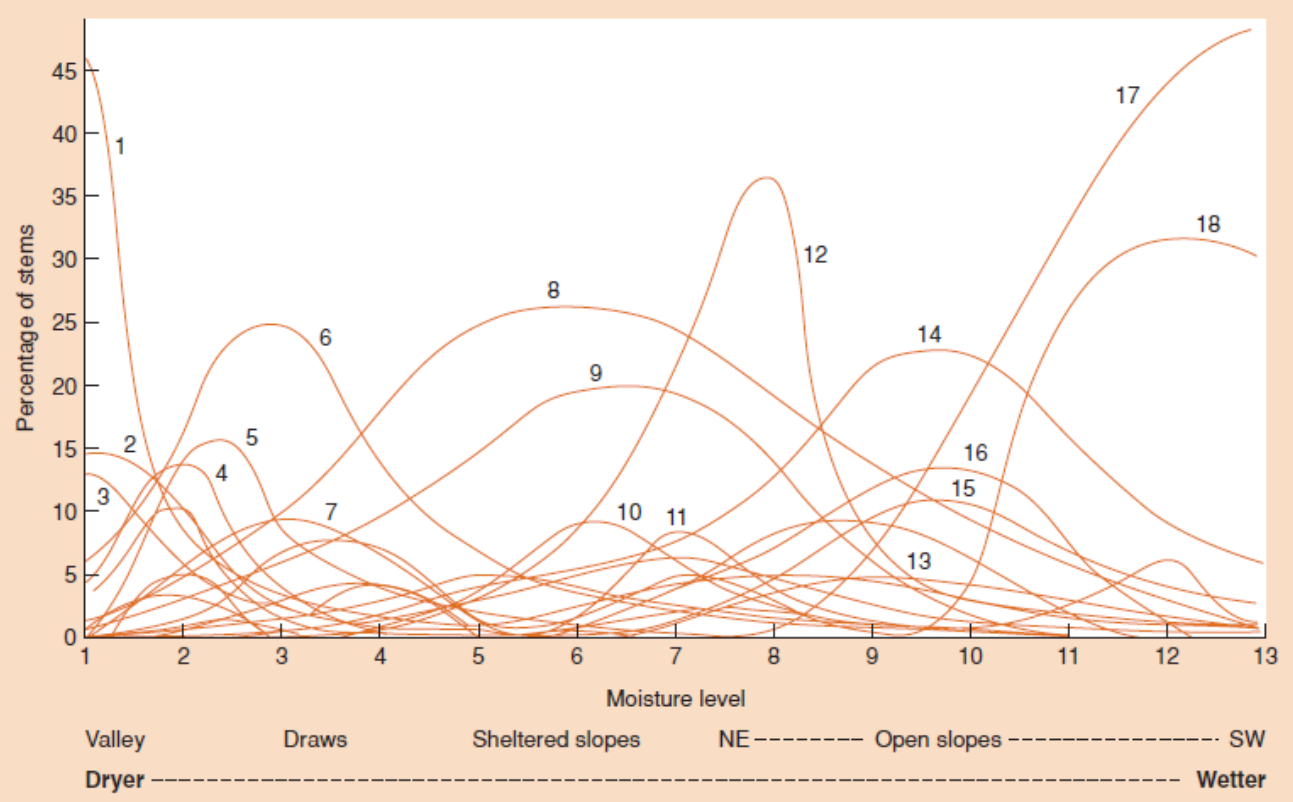
-
- Versão longa

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[10]:

```
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 [11] 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!)[12].

```
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)
    }
}
```

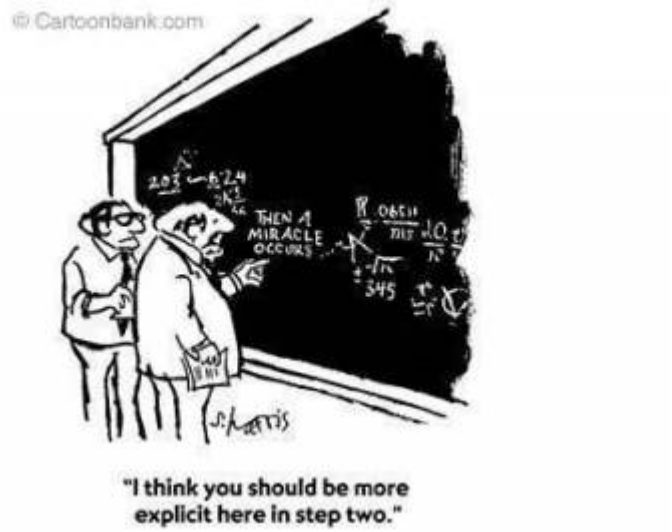
Agora teste o gráfico:

```
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[13)] 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 [14)]. 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[15)]. 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[16)].

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[17)].

```
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?[18)] 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.

10)

note que temos duas funções uma dentro da outra

11)

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

12)

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

13)

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

14)

isso é contestável, mas defensável

15)

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

16)

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

17)

parece complicado?!É só a lógica da linguagem!

18)

será que nossa função funfa??