Hebert

Mestrando em Biodiversidade Vegetal e Meio Ambiente, Núcleo de Pesquisa em Ecologia, Instituto de Botânica, São Paulo.

Área do Projeto: Dinâmica de comunidade vegetal em remanescente florestal.

Motivações: Entendimento das respostas das comunidades vegetais às mudanças ambientais, antrópicas ou não; medidas conservacionistas para a manutenção e restauração de ecossistemas florestais.

exec

Propostas de funções

Proposta A

Dada a importância dos estudos permanentes em comunidades vegetais e a rotina de mensuração, transformação e organização de dados dendrométricos de organismos vegetais, como perímetro, diâmetro e área basal de árvores, propõe-se a construção de uma função que a partir de dados de perímetro de árvores ramificadas, retorne um data.frame com as medidas do diâmetro equivalente e área basal. A função calculará também o diâmetro e área basal de árvores com apenas um valor de perímetro (árvores não ramificadas na altura do ponto de medida).

Proposta B

Criar uma função semelhante a da proposta A, com a tarefa adicional de realizar as transformações reversas, por exemplo, de área basal ou diâmetro para perímetro e vice versa, de acordo com o argumento utilizado. Além disso, a partir de um conjunto de dados temporais, realizar o cálculo da taxa do crescimento relativo de árvores e identificar árvores com “crescimento negativo”, o que pode servir de alerta para o pesquisador da existência de erros de mensuração ou da interferência de fatores naturais como a inclinação da árvore, a perda de casca e/ou morte do exemplar, por exemplo.

Fórmula para o cálculo da taxa de crescimento relativo de árvores em um período de tempo (t0: tempo inicial; t: tempo final), a partir de dados de diâmetro a 1,3 m (DAP) (Welden et al. 1991):

```
TCR = [(DAPt/DAPt0)^(1/Δt) – 1] x 100
```

A função também poderia realizar uma análise comparativa das taxas de crescimento relativo entre classes pré-estabelecidas, de tamanho ou grupo ecológico, por exemplo; com a escolha de um teste paramétrico ou não paramétrico, de acordo com a distribuição dos dados e o número de classes comparadas.

Comentários - Diogo

Plano A e B

Sua primeira proposta é interessante mais ainda um pouco básica. Note que isso é quase idêntico a um exercício dado na disciplina. Mas ela pode ser trivialmente estendida. Será que tem alguma analise que vc teria de fazer com esses dados que poderia também ser automatizada pela sua função? Nesse sentido a proposta B é muito mais interessante e completamente plausível. A inclusão do teste seria bem legal e útil pra vc no futuro..

Resposta - Hebert

Investirei na proposta B! Obrigado pelos comentários!

Trabalho Final

Página de Ajuda

```
tcr.analises                package:"Trabalhos finais da disciplina BIE5782
- USP"                R Documentation


Conversões de medidas dendrométricas, cálculo da taxa de crescimento
relativo, análise exploratória de dados e aplicação de teste estatístico


Description:

    Realiza conversão entre as unidades dendrométricas de PAP, DAP e Área
Basal. Além disso, a partir de um conjunto de dados temporais
    (de um tempo inicial e final), faz o cálculo das taxas de crescimento
relativo de árvores e identifica indivíduos com “crescimento negativo”, o
que pode
    servir de alerta para o pesquisador da existência de erros de mensuração
ou da interferência de fatores naturais como a inclinação da árvore, a perda
de casca e/ou
    morte do exemplar, por exemplo. A função cria gráficos para análise
exploratória de dados e aplica, se desejável, um teste estatístico
comparativo das taxas de crescimento relativo entre classes.

Usage:

     tcr.analises(x, medida, n.max.ramos, delta.t, teste=FALSE, n.classes)
```

```
Arguments:

x       um data frame (ver detalhes)
medida      "PAP"(perímetro a 1.3 m do solo), "DAP"(diâmetro a 1.3 m do
solo) ou "AB" (área basal do ramo)
n.max.ramos     número máximo de ramificações observado no conjunto de
unidades amostrais
delta.t     período de tempo entre a obtenção das medidas dendrométricas
iniciais e finais apresentadas
teste       TRUE para a realização de teste estatístico comparativo entre as
taxas de crescimento relativo positivas e FALSE (default) para a não
realização.
n.classes   número de classes a serem consideradas pelo teste estatístico. A
função compara duas ou três classes.


Details:

     As colunas 1, 2 e 3 do data frame devem estar reservadas. Recomenda-se
que sejam destinadas a informações básicas como parcela, nome da espécie e
número do indivíduo.
     O nome da coluna referente ao número dos indivíduos deve ser igual a
"N". As medidas, sejam de PAP, DAP ou AB, devem aparecer a partir da quarta
coluna, sequencialmente, de acordo
     com o número máximo de ramos estabelecido, primeiramente as medidas do
tempo 1 (inicial) e secundariamente as medidas do tempo 2 (final).
     Ex:
                ^ Parcela ^ Espécie ^ N ^ PAPinicial1 ^ PAPinicial2 ^...^
PAPinicialn ^ PAPfinal1 ^ PAPfinal2 ^ ... ^ PAPfinaln ^ Classe ^
     Não deve existir nenhum campo de PAP, DAP ou AB sem valor (NA). Para os
indivíduos que não possuem determinada ramificação, deve-se preencher o
respectivo
     campo com zero (0). Caso o data frame possua apenas os dados para os
cálculos da função, o código > x[is.na(x)]=0 pode ser utilizado antes de
aplicar a função.
     Recomenda-se que o usuário verifique atentamente os campos da planilha
que podem ser preenchidos por zero.
     Deve ser informado o número de classes a serem comparadas no argumento
classe e existir uma coluna do data frame com o nome "Classe" com a
numeração da classe dos indivíduos.

     Após o estudo dos gráficos apresentados deve-se responder a pergunta
que aparece no R-console. Caso a resposta seja "n" é calculado o teste t,
para a comparação de duas classes;
     e o teste de ANOVA, para a comparação de três classes. Caso a resposta
seja diferente é calculado o teste não-paramétrico Wilcoxon rank sum test
     (Mann-Whitney), para a comparação de duas classes; e o teste de
Kruskal-Wallis para a comparação de três classes.

Value:
```

```
     A função retorna um data frame com as colunas originais  dos dados de
entrada e com as colunas referentes às conversões das medidas de PAP, DAP e
AB do tempo inicial e final; ao diâmetro equivalente
inicial e final; à área basal total inicial e final; e à Taxa de Crescimento
Relativo "TCR" dos indivíduos.

  PAP1,..., PAPn, PAPf1,..., PAPfn
  DAP1,..., DAPn, DAPf1,..., DAPfn
  AB1,..., ABn, ABf1,..., ABfn
  AB.total.t0
  AB.final.tf
  DAP.eq.t0
  DAP.eq.tf
  TCR

  Se existirem indivíduos com TCR negativa, um alerta com o número dos
indivíduos é apresentado no R-console.
  Caso o argumento teste seja TRUE é impresso no R-console uma janela
gráfica com o histograma, o boxplot, o qqnorm e a qqline das classes de
individuos com TCR>0 (indivíduos que cresceram no período de tempo
considerado).
  Após a resposta do usuário a pergunta realizada é impresso no R-console o
resultado do teste estatístico.

Warning:

     Os vetores com as medidas devem ser numéricos para a efetivação dos
cálculos.
     A função não salva o data frame criado.


Note:

     É necessária a utilização de uma função do pacote "plotrix". Caso o
usuário não possua o pacote o mesmo pode ser instalado pelo comando:
     > install.packages("plotrix").

     Para a comparação de duas classes de interesse numere previamente a
coluna "Classe" dos indivíduos com o número 1 ou 2.
     Para a comparação de três classes numere previamente a coluna "Classe"
com o número 1, 2 ou 3, de acordo com o grupo estabelecido ao qual pertence
o indivíduo.


Author:

     Hebert Kondrat
     hebberkon@hotmail.com
```

```
References:

     WELDEN, C. W., HEWETT, S. W., HUBBELL, S.P. & FOSTER, R.B. 1991.
Sapling survival, growth, and recruitment: relationship to canopy height in
a
     neotropical forest. Ecology 72(1): 35-50.


Examples:

     ## 1. Aplicação da função tcr.analises ao conjunto de dados "teste1"
        ##Baixe o arquivo "teste1.csv"
    #Duas classes
        x=read.table("teste1.csv", sep=";", header=TRUE, as.is=T)
        y=tcr.analises(x, medida="PAP", n.max.ramos=5, delta.t=6,
teste=TRUE, n.classes=2)
    y # data frame criado

    #Três classes
     y=tcr.analises(x, medida="PAP", n.max.ramos=5, delta.t=6, teste=TRUE,
n.classes=3)
     y

    ##2. Criação de um data frame qualquer
    set.seed(3)
    AB1=sample(rep(c(6,8,10,15,30,50,100), times=c(15,15,10,10,5,3,2)))
    set.seed(3)
AB2=sample(rep(c(5.8,6.2,6.5,7,7.7,8.2,8.5,9,9.8,10.1,11,12.5,0,16,17,18,31,
33,34,54,60,70,150,180),times=c(2,3,5,3,4,3,5,5,2,2,4,2,2,2,4,2,1,2,2,1,1,1,
1,1)))
    x=data.frame("Parcela"="","Espécie"=paste("sp", 1:60,
sep=""),"N"=1:60,"AB1"=AB1,"AB2"=AB2,"Classe"=rep(1:3,each=20))
         #Aplicação da função aos dados gerados (medida=AB e n.max.ramos=1).
    tcr.analises(x, medida="AB", n.max.ramos=1, delta.t=6, teste=TRUE,
n.classes=3)
```

Código da Função

```
tcr.analises<-function(x, medida, n.max.ramos, delta.t, teste=FALSE,
n.classes)
{
    stopifnot(medida=="PAP"|medida=="AB"|medida=="DAP")#para de executar a
função caso o argumento medida seja diferente de "PAP", "AB" ou "DAP"
    P1=c((length(x[])+1)):(length(x[]) + n.max.ramos) #conserva um vetor com
valores fixos referente a posição de colunas, para uso posterior
    if(medida=="DAP")#determina a condição caso o argumento "medida" seja
igual a "DAP".
    {
    colnames(x)[4:(3+n.max.ramos)]=paste("DAP", 1:n.max.ramos, sep="")
```

```
#nomeia as colunas com os valores de DAP inicial
    colnames(x)[(4+n.max.ramos) :(3+2*n.max.ramos)]=paste("DAPf",
1:n.max.ramos, sep="") #nomeia as colunas com os valores de DAP final
    x[c((length(x[])+1)): (length(x[]) +
2*n.max.ramos)]=x[4:(3+2*n.max.ramos)]*pi # transforma os valores de DAP em
PAP
    colnames(x)[P1]=paste("PAP", 1:length(P1),sep="")    #nomeia as colunas
com os valores de PAP inicial
    colnames(x)[(max(P1)+1):(max(P1)+n.max.ramos)]= paste("PAPf",
1:length(P1),sep="")#nomeia as colunas com os valores de PAP final
    P2=c((length(x[])+1)): (length(x[]) + n.max.ramos)#conserva um vetor com
valores fixos referente a posição de colunas de interesse(colunas de AB)
    x[c((length(x[])+1)): (length(x[]) +
2*n.max.ramos)]=(x[4:(3+2*n.max.ramos)]^2*pi)/4 # transforma os valores de
DAP em AB
    colnames(x)[P2]=paste("AB", 1:length(P1),sep="") # nomeia as colunas com
os valores de AB inicial
    colnames(x)[(max(P2)+1):(max(P2)+n.max.ramos)]= paste("ABf",
1:length(P1),sep="") # nomeia as colunas com os valores de AB final
    }

    if(medida=="AB")#determina a condição caso o argumento "medida" seja
igual a "AB"
    {
    colnames(x)[4:(3+n.max.ramos)]=paste("AB", 1:n.max.ramos, sep="")
#nomeia as colunas com os valores de área basal inicial
    colnames(x)[(4+n.max.ramos) :(3+2*n.max.ramos)]=paste("ABf",
1:n.max.ramos, sep="") # nomeia as colunas com os valores de área basal
final
    x[c((length(x[])+1)): (length(x[]) +
2*n.max.ramos)]=sqrt((x[4:(3+2*n.max.ramos)]*4)/pi) # tranforma os valores
de AB em DAP
    colnames(x)[P1]=paste("DAP", 1:length(P1),sep="") #nomeia as colunas com
os valores de DAP inicial
    colnames(x)[(max(P1)+1):(max(P1)+n.max.ramos)]= paste("DAPf",
1:length(P1),sep="")#nomeia as colunas com os valores de DAP final
    P3=c((length(x[])+1)): (length(x[]) + n.max.ramos)#conserva um vetor com
valores fixos referente a posição de colunas de interesse(colunas de PAP)
    x[c((length(x[])+1)): (length(x[]) +
2*n.max.ramos)]=sqrt((x[4:(3+2*n.max.ramos)]*4)/pi)*pi  # transforma AB em
PAP
    colnames(x)[P3]=paste("PAP", 1:length(P1),sep="") #nomeia as colunas com
os valores de PAP inicial
    colnames(x)[(max(P3)+1):(max(P3)+n.max.ramos)]= paste("PAPf",
1:length(P1),sep="") #nomeia as colunas com os valores de PAP final
    P2=4:(3+n.max.ramos)#determina que o objeto P2 seja igual às colunas de
AB inicial, para uso posterior no cálculo da AB total
    }

    if(medida=="PAP") #determina a condição caso o argumento "medida" seja
```

```
igual a "PAP"
    {
    colnames(x)[4:(3+n.max.ramos)]=paste("PAP", 1:n.max.ramos,
sep="")#nomeia as colunas com os valores de PAP inicial
    colnames(x)[(4+n.max.ramos) :(3+2*n.max.ramos)]=paste("PAPf",
1:n.max.ramos, sep="") #nomeia as colunas com os valores de PAP final
    x[c((length(x[])+1)): (length(x[]) + 2*n.max.ramos)]=x[4:(3+
2*n.max.ramos)]/pi #tranforma os valores de PAP em DAP
    colnames(x)[P1]=paste("DAP", 1:length(P1),sep="") #nomeia as colunas com
os valores de DAP inicial
    colnames(x)[(max(P1)+1):(max(P1)+n.max.ramos)]= paste("DAPf",
1:length(P1),sep="") #nomeia as colunas com os valores de DAP final
    P2=c((length(x[])+1)): (length(x[]) + n.max.ramos)#conserva um vetor com
valores fixos referente a posição de colunas de interesse(colunas de AB)
    x[c((length(x[])+1)): (length(x[]) + 2*n.max.ramos)]= ((x[4:(3+
2*n.max.ramos)])^2)/(4*pi) #tranforma os valores de PAP em AB
    colnames(x)[P2]=paste("AB", 1:length(P1),sep="") # nomeia as colunas com
os valores de AB
    colnames(x)[(max(P2)+1):(max(P2)+n.max.ramos)]= paste("ABf",
1:length(P1),sep="") # nomeia as colunas com os valores de AB final
    }
    x$AB.total.t0=apply(x[P2], MARGIN=1, sum) #cria uma coluna com a soma
das áreas basais iniciais de um indivíduo
    x$AB.total.tf=apply(x[(max(P2)+1):(max(P2)+ n.max.ramos)], MARGIN=1,
sum)#cria um coluna com a soma das áreas basais finais de um indivíduo
    x$DAP.eq.t0=sqrt((x$AB.total.t0*4)/pi) #cria uma coluna com o diâmetro
equivalente inicial de um indivíduo
    x$DAP.eq.tf=sqrt((x$AB.total.tf*4)/pi) #cria uma coluna com o diâmetro
equivalente final de um indivíduo
    x$TCR=((x$DAP.eq.tf/x$DAP.eq.t0)^(1/delta.t)- 1)*100 #cria uma coluna
com as taxas de crescimento relativo dos indivíduos

    if(sum(x$TCR<0)>0)#determina a condição caso exista alguma Taxa de
Crescimento Relativo (TCR) menor que zero
    {
    individuos=x[x$TCR<0,]$N # cria um vetor chamado individuos com o número
dos indivíduos que apresentaram TCR menor que zero
    cat("\nOs seguintes indivíduos apresentaram crescimento negativo:",
paste(individuos), "\nFatores como inclinação de árvore, perda de casca e/ou
morte de exemplar, por exemplo, podem estar presentes.
        Verifique atentamente se não existem erros de mensuração.\n\n")#
cria uma mensagem de alerta com os números dos indivíduos com TCR menor que
zero
    }
    if(teste==TRUE) #determina a condição caso o argumento teste seja igual
a TRUE
    {
    stopifnot(n.classes==2|n.classes==3)#para de executar a função caso o
argumento n.classes seja diferente dos números 2 ou 3
    library(plotrix) #carrega a função plotrix
```

```
    if(n.classes==2) #determina a condição caso o argumento n.classes seja
igual a 2
    {
    dados1=x[x$TCR>0&x$Classe==1,]$TCR #cria um objeto com os valores de TCR
da classe 1 maiores que zero
    dados2=x[x$TCR>0&x$Classe==2,]$TCR #cria um objeto com os valores de TCR
da classe 2 maiores que zero
    graficos <- function(dados1,dados2) # criação da função graficos
    {
    x11(width=12, height=10) #determina o tamanho da janela de gráficos
    par(mfrow = c(2,2)) # determina o número de linhas e colunas da janela
de gráficos, nas quais serão desenhados os gráficos
    vetores<-list(dados1,dados2)#cria uma lista
    multhist(vetores, col=c("white", "gray"), xlab="Valores",
ylab="Frequência", main="Histograma das amostras") #aplica a função do
pacote plotrix para a criação de um histograma personalizado
    legend("topright", c("Amostra1", "Amostra2"), fill = c("white", "gray"))
# cria uma legenda para o histograma
    boxplot(vetores, col=c("white", "gray"), xlab="Amostras",
ylab="Dispersão", main="Boxplot das amostras") #cria um boxplot da lista
vetores
    qqnorm(dados1, main="qqplot da Amostra1", xlab="Quantis Teóricos",
ylab="Quantis da amostra") # cria o qqnorm do objeto dados1
    qqline(dados1) # cria a qqline do objeto dados1
    qqnorm(dados2, main="qqplot da Amostra2", xlab="Quantis Teóricos",
ylab="Quantis da amostra") #cria o qqnorm do objeto dados2
    qqline(dados2) # cria a qqline do objeto dados2
    cat("\t\t Sumário da Amostra1 \n") # apresenta no R-Console o texto
criado, que para esse caso serve como título do sumário da amostra 1
    print(summary(dados1))# imprime o sumário da amostra considerada no R-
Console
    cat("\n\t\t Sumário da Amostra2 \n")#apresenta no R-Console o texto
criado, que para esse caso serve como título do sumário da amostra 2
    print(summary(dados2))# imprime o sumário da amostra considerada no R-
Console
    }
    graficos(dados1, dados2)#aplica a função graficos
    Pergunta <- function() # criação da função Pergunta
    {
     ANSWER <- readline("\n Deseja realizar um teste não-paramétrico?")
#Estabelece a pergunta
     if (substr(ANSWER, 1, 1) == "n") #determina a condição caso a resposta
do usuário seja "n"
    t=t.test(dados1,dados2) #cria o objeto t, relacionado a aplicação do
teste t aos dados
    else # determina a condição caso a resposta não seja "n"
    t=wilcox.test(dados1,dados2) #cria o objeto t, relacionado a aplicação
do Wilcoxon rank sum test aos dados.
    print(t) #imprime o objeto t no R-Console.
    }
```

```
    }

    if(n.classes==3) #determina a condição caso o argumento n.classes seja
igual a 3
    {
    dados1=x[x$TCR>0&x$Classe==1,]$TCR #cria um objeto com os valores de TCR
da classe 1 maiores que zero
    dados2=x[x$TCR>0&x$Classe==2,]$TCR #cria um objeto com os valores de TCR
da classe 2 maiores que zero
    dados3=x[x$TCR>0&x$Classe==3,]$TCR #cria um objeto com os valores de TCR
da classe 3 maiores que zero
    graficos2 <- function(dados1,dados2,dados3) #criação da função graficos2
    {
    x11(width=14, height=10)#determina o tamanho da janela de gráficos
    par(mfrow = c(2,3)) #determina o número de linhas e colunas da janela de
gráficos, nas quais serão desenhados os gráficos
    vetores<-list(dados1,dados2,dados3) #cria uma lista
    multhist(vetores, col=c("white", "gray", "gray50"), xlab="Valores",
ylab="Frequência", main="Histograma das amostras") #aplica a função do
pacote plotrix para a criação de um histograma personalizado
    legend("topright", c("Amostra1", "Amostra2", "Amostra3"), fill =
c("white", "gray", "gray50")) # cria uma legenda para o histograma
    boxplot(vetores, col=c("white", "gray", "gray50"), xlab="Amostras",
ylab="Dispersão", main="Boxplot das amostras") #cria um boxplot da lista
vetores
    qqnorm(dados1, main="qqplot da Amostra1", xlab="Quantis Teóricos",
ylab="Quantis da amostra") # cria o qqnorm do objeto dados1
    qqline(dados1) # cria a qqline do objeto dados1
    qqnorm(dados2, main="qqplot da Amostra2", xlab="Quantis Teóricos",
ylab="Quantis da amostra")  # cria o qqnorm do objeto dados2
    qqline(dados2) # cria a qqline do objeto dados2
    qqnorm(dados3, main="qqplot da Amostra3", xlab="Quantis Teóricos",
ylab="Quantis da amostra") # cria o qqnorm do objeto dados3
    qqline(dados3) # cria a qqline do objeto dados3
    cat("\t\t Sumário da Amostra1 \n") # apresenta no R-Console o texto
criado, que para esse caso serve como título do sumário da amostra 1
    print(summary(dados1))# imprime o sumário da amostra considerada no R-
Console
    cat("\n\t\t Sumário da Amostra2 \n") # apresenta no R-Console o texto
criado, que para esse caso serve como título do sumário da amostra 2
    print(summary(dados2))# imprime o sumário da amostra considerada no R-
Console
    cat("\n\t\t Sumário da Amostra3 \n")# apresenta no R-Console o texto
criado, que para esse caso serve como título do sumário da amostra 3
    print(summary(dados3))    # imprime o sumário da amostra considerada no
R-Console
    }
    graficos2(dados1, dados2, dados3) #aplica a função graficos2
    Pergunta <- function() #criação da função pergunta
    {
    var.resp=c(dados1,dados2,dados3) #concatena os valores dos dados
```

```
considerados em um objeto
classe=factor(rep(c("dados1","dados2","dados3"),times=c(length(dados1),lengt
h(dados2),length(dados3)))) #cria um fator com a repetição de caracteres
respectivos aos dados considerados
    res.anova=aov(var.resp~classe) #cria um objeto relacionado a aplicação
do teste de ANOVA aos dados
     ANSWER <- readline("\n Deseja realizar um teste não-paramétrico?")
#Estabelece a pergunta
     if (substr(ANSWER, 1, 1) == "n") #determina a condição caso a resposta
seja "n"
    t=summary(res.anova) #cria um objeto com os resultados do teste de ANOVA
    else # determina a condição caso a resposta não seja "n"
    t=kruskal.test(list(dados1,dados2,dados3)) #cria um objeto relacionado a
aplicação do teste de Kruskal-Wallis aos dados
    print(t) # imprime no R-console o objeto t criado
    }
    }
    Pergunta() #aplica a função Pergunta
    }
return(x) #retorna o data frame x criado
}
```

Arquivos

teste1.csv Script da função Help da função