André Luiz Giles

Doutorando em ecologia pela UNICAMP-IB. Possuí interesse em Ecologia funcional de plantas e desenvolve projetos relacionados com efeitos da seca no solo em ecossistemas tropicas. Possui enfoque nos efeitos da seca como determinantes de um novo estágio na vegetação e a relação hídrica e de carbono em plantas.

exec

PROPOSTA A Descrição da função:

```
“str.community”
```

A função irá calcular parâmetros para análises exploratórias de dados de comunidades vegetais. A função calculará por cada unidade amostral (parcela, ponto quadrante, área) ou pelo total de amostragem os seguintes parâmetros: área basal, densidade, altura máxima, altura média, altura min, número total de espécies, nº de famílias, Riqueza rarefeita para o menor número de indivíduos, diversidade, equabilidade, diâmetro max, diâmetro min, diâmetro médio. A função também calculará os parâmetros por espécies: Frequência absoluta e relativa, dominância absoluta e relativa, densidade absoluta e relativa, diâmetro médio (max e min), altura média (max e min), índice de valor de importância.

Entrada dos dados: A entrada dos dados deverá ser um data frame, contendo as espécies nas linhas e valores de altura, diâmetro e área de amostragem nas colunas. Os argumentos da função serão: (x)= data.frame, method= (ponto quadrante ou parcela), param= (“total” se calculado como uma única coluna ou “sep” se calculado por unidades distintas descriminados nas colunas do data frame). A função irá incorporar os argumentos da função “hist.” para realizar o gráfico de distribuição diamétrica e de altura, e da função “plot” para riqueza em espécies.

Saída da função: A função irá retornar um data frame, contendo os índices e parâmetros citado acima e dois gráficos representado riqueza de espécies e distribuição diâmétrica.

> Olá André,
>
> Interessante sua função, só fiquei com algumas dúvidas. As linhas do df de entrada contém as espécies vegetais, ou um indivíduo? Não faz sentido pra mim que as linhas sejam espécies, sendo que uma linha é uma observação. No caso mais simples, você teria um indivíduo por espécie vegetal. Esclarece isso melhor. Outra coisa, você vai calcular vetores com novas variáveis e diversos parâmetros que não terão o mesmo comprimento, pelo que a saída não pode ser UM data frame. Talvez você pode adicionar as novas variáveis ao df inicial e criar outro com os parâmetros descritivos calculados, e retornar tudo numa lista. Eu aconselho você ir em frente com essa proposta seguindo minhas sugestoes se concordar. Att, — *Gustavo Agudelo 2017/06/02 19:13*

PROPOSTA B Descrição da função:

```
“nsc.test”
```

A função irá calcular a quantidade de carboidratos não estruturais a partir de data.frames numéricos com valores de massa, absorbância de amostras e absorbância de referência. Esse cálculo é feito baseado na comparação de curvas de absorbância com amostras de referência (apenas com enzimas e agua) e as amostras de interesse.

Eixo x = OD (optical density = absorbância) Eixo y = [Glucose] mg / mL Os valores de [glicose] mg / mL para as amostras usando a equação de sua curva padrão Y = m * x + b [Glucose] mg / mL = m (OD) + b (b=(intercept))Após isso é necessária a conversão de unidades para % de peso seco:

Por fim as concentrações de açúcares NSC

Glucose / Frutose = Quantidade de glicose da placa de “açúcares livres” Sacarose = quantidade de glicose da placa “açúcares livres + sacarose”, menos a quantidade de glicose da placa de “açúcares livres” Amido = quantidade de glicose da placa “NSC total”, menos a quantidade de glicose da placa “açúcares livres + sacarose” NSC = Quantidade de glicose da placa “NSC total”

Entrada dos dados: A entrada dos dados deverá ser três data frame ou três matrizes. A primeira contendo os valores de absorbância de cada amostra e respectivas enzimas, o segundo contendo a massa (pesagem previa) das amostras. Os argumentos da função serão: (x)= data.frame ou matrix da amostra, (y)= data.frame ou matrix da amostra de referência, (z)= data.frame ou matrix com massa das amostras, enzima= (GHK-PGI, Invertase, Amyloglucosidas).

Saída da função: A função ira retornar um data frame, contendo os valores de Amido, NSC total, sacarose, glucose e frutose.

Consulte aqui o documento referente a função da proposta B

> André,
>
> Pelo que entendi, você pretende calcular os valores de Amido, NSC total, sacarose, glucose e frutose a partir dos valores preditos por um modelo linear que incluiria as absorbâncias das amostras e dos controles, aplicando depois uma conversão usando a variável massa. É isso? Se for, você pode colocar todas esas variáveis em um só df de entrada e não precisaria de três. Depois de ajustar o modelo, você pode extrair os valores preditos pelo modelo e adicionar colunas ao df inicial que contenham os valores corrigidos pela massa. Seria legal que a função retornasse um diagnóstico gráfico desse modelo para o usuário avaliar se precisa transformar as variáveis (seria viável colocar um

argumento de transformação?), e assim (des)transformar de novo ao se calcular os valores reais das variáveis de interesse. — *Gustavo Agudelo 2017/06/02 19:24*

TRABALHO FINAL

Código da Função

Função:

```
“str.community”
```

```
str.community=function(dados, method, estimativa="CAP", dim.plot)
{
  if(class(dados) != "data.frame")
  {
    stop( "o objeto não é um dataframe")
  }
  else
    if(method=="parcela")
    {
      ###### Garantindo que o R irá ler corretamente os dados####
      dados[,1]= as.factor(dados[,1])
      dados[,5]= as.numeric(dados[,5])
      dados[,6]=as.numeric(dados[,6])
      ##########################
      n.individuos.plot= table(dados[ ,1]) ## numero de individuos por
unidade amostral
      nplot=levels(dados[,1]) ## captando a variação das parcelas
      nplot=length(nplot) ##
      nplot.total= dim.plot*length(nplot) ## numero total da área amostral
parcelas
      density.tot= (length(dados[,1])/nplot.total) * (10000) ## densidade
total
      alt.max= max(dados[,6]) # altura máxima
      alt.min= min(dados [,6])# altura mínima
      alt.mean= mean(dados[,6]) # media da altura
      n.sp=length(table(dados[,4]))  # numero de espécies total
      n.ind.total= sum(length(dados[,3])) # numero total de individuos
      n.ind.sp= table(dados[ ,4]) ## numero de individuos por espécie
      n.sp.fam= table(dados[,3]) ## numero de espécies por família
      n.fam.total= length(n.sp.fam) ## numero total de família
      ##diversidade e equabilidade
      ## div shannon total
      prop.i= n.ind.sp/n.ind.total
      h.linha.total= -sum(prop.i*log(prop.i))
```

```
      ## div de simpson
      simpson.d.total= sum((prop.i)^2)
      ### Equabilidade de pielou
      pielou.j= h.linha.total/log(n.sp)
      ### Riqueza rarefeita
      ### diametro e área basal
      if(estimativa=="CAP") # se estimativa for circuferência a altura do
peito
      {
        dados[ ,5]= dados[ ,5]/ pi # calculando DAP apartir do CAP
        AB= (pi*(2*(dados[ ,5])^2)) # Calculo da área basal total
        AB.media= (pi*(2*(dados[ ,5])^2))/length(dados[ ,5]) #Calculo da
área basal me´dia
        diam.mean= mean(dados[,5]) # Calculo do diâmetro médio
        diam.min= min(dados[,5])# Calculo do diamentro mínimo
        diam.max= max(dados[,5])# Calculo do diâmetro máximo
        diam.plot.max= aggregate(dados[ ,5], list(dados[ ,1]), max) #
diametro maximo por plot
        diam.plot.min= aggregate(dados[ ,5], list(dados[ ,1]), min)
#diametro minimo por plot
        diam.plot.mean= aggregate(dados[ ,5], list(dados[ ,1]), mean) #
dimatro medio por plot
        ##Fazendo os plots
        x11() ## Abrindo novo dispositivo gráfico
        plot.dist.diam= hist(dados[,5], breaks= 10, freq=T ,col="gray",
ylab="Frequência", xlab= "Diâmetro", main=" Distribuição diâmetrica", las=1,
cex.axis=1.5, cex.lab=1.5) ## plotando distribuição diamétrica dos
indivíduos
      }
      if(estimativa== "DAP") # se a estimativa for diâmetro a altura do
peito
      {
        AB= (pi*(2*(dados[ ,5])^2))# Calculo da área basal total
        AB.media= (pi*(2*(dados[ ,5])^2))/length(dados[ ,5])#Calculo da área
basal me´dia
        diam.mean= mean(dados[,5])# Calculo do diâmetro médio
        diam.min= min(dados[,5])# Calculo do diamentro mínimo
        diam.max= max(dados[,5])# Calculo do diâmetro máximo
        diam.plot.max= aggregate(dados[ ,5], list(dados[ ,1]), max) #
diametro maximo por plot
        diam.plot.min= aggregate(dados[ ,5], list(dados[ ,1]), min)
#diametro minimo por plot
        diam.plot.mean= aggregate(dados[ ,5], list(dados[ ,1]),
mean)#diametro mediopor plot
        ### Fazendo o plot da distribuição diâmetrica
        x11()  # novo dispositivo gráfico
        plot.dist.diam= hist(dados[,5], breaks= 10, freq=T ,col="gray",
ylab="Frequência", xlab= "Diâmetro", main=" Distribuição diâmetrica", las=1,
cex.axis=1.5, cex.lab=1.5) ## Fazendo o plot da distribuição diâmetrica dos
individuos
```

```
      }
      ########################################
      ########Calculando por plots#########
      ####################################
      ### nº de indivíduos por parcela
      n.individuos.plot= table(dados[ ,1])
      ## ## densidade por plot
      density.plot= n.individuos.plot/dim.plot
      ### Por plots
      tabela.especie.parcela= table(dados[ ,4], dados[ ,1]) ## selecionando
numero de individuos por espécie em cada parcela
      dimensao=dim(tabela.especie.parcela)## calculando dimensão para montar
matrix
      matrix.especie.parcela= matrix(tabela.especie.parcela,
ncol=dimensao[2], nrow = dimensao[1]) ## motando matrix
      matrix.especie.parcela.S.rar= matrix.especie.parcela # guardando a
matrix em um novo objeto para o calculod e riqueza estimada psoteriormente
      matrix.especie.parcela[(matrix.especie.parcela)==0] = NA ##
transformando valores em NA
      ## Nº de familias por plot
      n.sp.fam.plot= table(dados[,3], dados[,1]) ## numero de espécies por
família em cada plot
      #######
      n.especie.parcela= matrix(tabela.especie.parcela, ncol=dimensao[2],
nrow = dimensao[1]) ## numero espécie por parcela
      n.especie.parcela[(matrix.especie.parcela)>1] = 1 ##transformando em 0
e 1 para somar
      n.total.sp.plot= apply(n.especie.parcela,2, sum) ## usando apply para
calcular n total de sp por plot
      names(n.total.sp.plot)= c(levels(dados[ ,1])) # renomeando o vetor
      n.total.sp.plot
      ################Criando função para diversidade de Shannon e Simpson
##########################
      shannon= function(y)
      {
        if (sum(is.na(y)>0))
        {dados.div=(na.omit(y))}
        else
        {dados.div=y}
        prop.i= dados.div/sum(dados.div)
        h.linha= -sum(prop.i*log(prop.i))
      }
      simpson= function(x)
      {
        if (sum(is.na(x)>0))
        {dados=(na.omit(x))}
        else
        {dados=x}
        prop.i= dados/sum(dados)
        simp= sum((prop.i)^2)
      }
```

```
      ##################################################################
      ########Calculando diverisdade, equabilidade por plot###########
      ################################################################
      div.plot.shannon= apply(matrix.especie.parcela,2,shannon) ##
Calculando Shannon por plot
      names(div.plot.shannon)= c(levels(dados[,1]))
      div.plot.simpson= apply(matrix.especie.parcela,2,simpson) ##
Calculando Simpson por plot
      names(div.plot.simpson)= c(levels(dados[,1]))
      ###Calculando equabilidade por plot
      pielou.j.plot= div.plot.shannon/ log(n.total.sp.plot)
      ########################################################
      ##### Calculando densidade e altura e diametro por plot######
      ########################################################
      ### densidade  por plot
      n.individuos.plot= table(dados[ ,1])
      density.plot= (n.individuos.plot/dim.plot) * 10000 # individuos por
hectate
      density.plot=as.vector(density.plot)
      ## altura por plot
      alt.plot.max= aggregate(dados[ ,6], list(dados[ ,1]), max) # alt max
      alt.plot.min= aggregate(dados[ ,6], list(dados[ ,1]), min) # alt min
      alt.plot.mean= aggregate(dados[ ,6], list(dados[ ,1]), mean) # alt
media
##########################################################################
##########################################################################
#####################################################################
##########################################################################
###
      ##calculo da riqueza rarefeita ////
      ###############################################################
      matrix.especie.parcela.2=matrix.especie.parcela.S.rar ## renomeando a
matrix
      matrix.especie.parcela.2[(matrix.especie.parcela.2)>1] = 1 ##
colocando matriz de prensença e ausendia
      matrix.especie.parcela.rarefacao=
apply(matrix.especie.parcela.2,1,sum) ## aplicando somatoria para detectar
espécies que ocorrem apenas em uma amostra, e que ocorrem duas (singletons e
doubletons)
n.sp.ocurr.1sample=length(matrix.especie.parcela.rarefacao[(matrix.especie.p
arcela.rarefacao==1)])##detectando espécies que ocorrem apenas em uma
amostra
n.sp.ocurr.2sample=length(matrix.especie.parcela.rarefacao[(matrix.especie.p
arcela.rarefacao==2)])##detectando espécies que em duas amostras
      n.sample= length(n.total.sp.plot) # tamanho da amostra
#########################################################################
      ######### Riqueza estimada pela rarefação Colwel, et al
2012##############
#########################################################################
      S.estimate= n.sp +  (((n.sample-1)/(n.sample)) *
```

```
((n.sp.ocurr.1sample*(n.sp.ocurr.1sample-1))/(2*(n.sp.ocurr.2sample+1))))
##Riqueza estimada baseada em amostra que ocorrem apenas 1 individuo =
      ######################################################################
      ############### Riqueza estimada por plot CHAO 1###################
      ######################################################################
      matrix.especie.parcela.S.rar.1= matrix.especie.parcela.S.rar ##
atribuindo novo nome para selecionar apenas especies que possuem 1 individuo
por parcela
      matrix.especie.parcela.S.rar.2= matrix.especie.parcela.S.rar ##
atribuindo novo nome para selecionar apenas especies que possuem 2
individuos por parcela
      n.individuos.plot= as.numeric(n.individuos.plot) ## transformando o
vetor de numero de individuos por plot em um vetor numerico
      matrix.especie.parcela.S.rar.1[(matrix.especie.parcela.S.rar.1)>1]=0
## transformando em 0 e 1 para calculo de especie que ocorrem com apenas um
individuo
matrix.especie.parcela.S.rar.2[(matrix.especie.parcela.S.rar.2)!=2]=0##
excluindo tudo que é diferente de 2
      matrix.especie.parcela.S.rar.2[(matrix.especie.parcela.S.rar.2)==2]=1
## transformando em 0 e 1 para calculo das espécies que ocorrem com 2
individuos
      f1=apply(matrix.especie.parcela.S.rar.1,2,sum) #utilizando apply para
quantificar espécies que possuem apenas 1 indivíduo por amostra
      f2=apply(matrix.especie.parcela.S.rar.2,2,sum) # renomeando para a
fórmula
      ### calculando Riqueza rafeita pelo Chao 1, por parcelas ( mesma base
para construção do gráfico de riqueza)
      s.estimate.plot= n.sp
+(((n.ind.total-1)/(n.ind.total))*(f1*(f1-1))/(2*(f2+1))) ## Riqueza de
espécie estimada por plot
##########################################################################
##############################
      ## Calculo dFrequência absoluta e relativa, dominância absoluta e
relativa, densidade absoluta e relativa, diâmetro médio (max e min), altura
média (max e min), índice de valor de importância, por espécie
##########################################################################
##############################
      tabela.par.diam.sp= aggregate(dados[,5], list(dados[,4]), mean) ##
calculo do diâmetro médio
      tabela.par.alt.sp=aggregate(dados[,6], list(dados[,4]), mean) ##
calcula da altura média
      ## Densidade absoluta das espécies DeAbi = ni x 1ha/A
      n.ind.sp=data.frame(n.ind.sp)
      dens.abs.sp=n.ind.sp
      dens.abs.sp[,3]= (n.ind.sp[,2]/nplot.total) * 10000 ## densidade
absoluta por hectare
      dens.abs.sp[ ,4]= (n.ind.sp[,2]/n.ind.total)*100 ## densidade relativa
em %
      matrix.sp.param=matrix.especie.parcela.S.rar ## renomeando a matrix
      matrix.sp.param[(matrix.sp.param)>1] = 1 ## criando ua matrix de
presença e ausência
```

```
      dimnames(matrix.sp.param)= list(levels(dados[,4]),levels(dados[,1])) #
renomenado a matrix
      #Frequencia
      dens.abs.sp[ ,5]= (apply(matrix.sp.param,1,sum)/ length(nplot)) * 100
# Em porcentagem afrequencia absoluta
      dens.abs.sp[,6]=(dens.abs.sp[,5]/(sum(dens.abs.sp[,5]))) * 100 #
frequência relativa
      ## dominancia
      dados[,7]= (((dados[ ,5])^2)*pi)/4 ## criando uma nova coluna no date
frame original para área basal
      ############################ Parenteses para calcuo da área basal por
plot ( aproveitando a nova coluna)
      AB.plot= aggregate(dados[,7], list(dados[,1]), sum)###calculando área
Basal por plot
      AB.plot.media=  aggregate(dados[,7], list(dados[,1]), mean) ## Area
basal média por plot
      ######################################################################
      #######
      ##
      dominancia.abs= aggregate(dados[,7], list(dados[,4]), sum)## usando
aggregante para juntar dados de area basal por espécie dominancia absoluta
      dens.abs.sp[,7]= dominancia.abs[,2] ## dominancia absoluta
      dens.abs.sp[,8]=(dens.abs.sp[,7]/sum(dens.abs.sp[,7])) *100 ###
dominancia relativa
      ##IVI
      dens.abs.sp[,9]= dens.abs.sp[,4] + dens.abs.sp[,6] + dens.abs.sp[,8]
######################################################################
######################
      ##### Motando a tabela dos parâmetros por espécie#################
######################################################################
##################
      tabela.param.sp= dens.abs.sp
      tabela.param.sp[,10]=  tabela.par.diam.sp[ ,2]
      tabela.param.sp[,11]=  tabela.par.alt.sp[ ,2]
      names(tabela.param.sp)= c("Espécie","Nº de indivíduos",
"Densidade","Densidade relativa"," Frequência","Frequência relativa",
"Dominância", "Dominância relativa", "IVI", "Diâmetro médio", "Altura
média")
      #################
      tabela.param.sp
      ##############
      #######
######################################################################
######################
      ####Tabela parametros totais###
      tabela.parametro.total= data.frame (
        nplot,
        density.tot,
        n.sp,
        n.ind.total,
```

```
        n.fam.total,
        alt.max,
        alt.min,
        alt.mean,
        AB,
        AB.media,
        diam.max,
        diam.min,
        diam.mean,
        h.linha.total,
        simpson.d.total,
        pielou.j,
        S.estimate)
      tabela.parametro.total= tabela.parametro.total[1,]
      names(tabela.parametro.total)= c( "número de parcelas","Densidade",
"Nº de espécies", "Nº de indivíduos", "N de famílias", "Altura
máxima","Altura mínina","Altura média", "Área basal","Área basal
média","Diâmetro máximo", "Diâmetro mínimo","Diâmetro médio", "Shannon",
"Simpson", "Pielou","Riqueza estimada")
##########################################################################
##############################
      ## Fazendo data frame com parametros por parcela
      tabela.parametro.plot= data.frame(n.individuos.plot, n.total.sp.plot,
density.plot,div.plot.shannon,div.plot.simpson,pielou.j.plot,s.estimate.plot
, alt.plot.max[,2],alt.plot.min[,2],alt.plot.mean[,2],    diam.plot.max[,2],
diam.plot.min [,2], diam.plot.mean[,2],         AB.plot[,2],
AB.plot.media[,2]) ## Juntando os dados
      names(tabela.parametro.plot)= c("Nº de individuos", "Nº de
espécies","Densidade", "Shannon", "Simpson", "Pielou","Riqueza estimada",
"Altura máxima","Altura mínima", "Altura média","Diâmetro máximo","Diâmetro
mínimo", "Diâmetro médio", "Área basal", "Área basal média") ## Renomeando
as colunas
      ##########################
      n.sp.fam=data.frame(n.sp.fam) ## criando um data frame com numero de
espécies por família
      names(n.sp.fam)= c("Família","Nº de espécie") ####
      ########## Retornando lista final
      list.final=
list(tabela.parametro.total,tabela.param.sp,n.sp.fam,tabela.parametro.plot)
      names(list.final)  = c("Tabela total", "Tabela de espécies","Nº de
espécies por família","Tabela por Parcela")
      ##############parametro.plot
      ########################################################################
      ###retornando o plot de altura
      plot.dist= hist(dados[,6], breaks= 10, freq=T ,col="gray",
ylab="Frequência", xlab= "Altura", main=" Distribuição de altura", las=1,
cex.axis=1.5, cex.lab=1.5)
      ### retornando o plot de riqueza estimada pela rarefação
      tabela.especie.parcela.plot= table(dados[ ,1], dados[ ,4]) ##
selecionando numero de individuos por espécie em cada parcela
      dimensao=dim(tabela.especie.parcela.plot)## calculando dimensão para
```

```
montar matrix
      matrix.especie.parcela.plot= matrix(tabela.especie.parcela.plot,
ncol=dimensao[2], nrow = dimensao[1])
      ## Instalando pacote necessário para calculo de Riqueza acmulada
      library(vegan) ## Carregando o pacote
      matrix.plot=specaccum(matrix.especie.parcela.plot,
"rarefaction",gamma="chao2",permutations = 1000) ## Calculando riqueza
rarefeita pelo chao 2 e permutando 1000 vezes
      x11()
      plot(matrix.plot, ci.type="poly", col="black", lwd=2, ci.lty=0,
ci.col="gray", las=1, ylab="Número de espécies", xlab="Número de parcelas",
cex.lab=1.5, cex.axis=1.5, bty="l")
      ## Plotando curva de acumulação de espécies
      return(list.final)}
  if (method=="quadrante") ## Controle de fluxo para estimativa de quadrante
que o dim.plot é diferente de 0.
    if(dim.plot != 0)
    {
      stop( "Valor errado no argumento dim.plot. Colocar dim.plot= 0")
    }
  else
  {
    ###### Garantindo que o R irá ler corretamente os dados####
    dados[,1]= as.factor(dados[,1])
    dados[,5]= as.numeric(dados[,5])
    dados[,6]=as.numeric(dados[,6])
    ##########################
    n.individuos= length(dados[,1])
    n.individuos.plot= table(dados[ ,1]) ## numero de individuos por unidade
amostral
    nplot=levels(dados[,1]) ## captando a variação dos quadratntes
    nplot=length(nplot) ##  numero de quadrantes
    if(estimativa=="CAP") # se estimativa for circuferência a altura do
peito
    {
      dados[ ,5]= dados[ ,5]/ pi # calculando DAP apartir do CAP
      AB= (pi*(2*(dados[ ,5])^2)) # Calculo da área basal total
      AB.media= (pi*(2*(dados[ ,5])^2))/length(dados[ ,5]) #Calculo da área
basal me´dia
      diam.mean= mean(dados[,5]) # Calculo do diâmetro médio
      diam.min= min(dados[,5])# Calculo do diamentro mínimo
      diam.max= max(dados[,5])# Calculo do diâmetro máximo
      diam.plot.max= aggregate(dados[ ,5], list(dados[ ,1]), max) # diametro
maximo por plot
      diam.plot.min= aggregate(dados[ ,5], list(dados[ ,1]), min) #diametro
minimo por plot
      diam.plot.mean= aggregate(dados[ ,5], list(dados[ ,1]), mean)
      x11()
      plot.dist.diam= hist(dados[,5], breaks= 10, freq=T ,col="gray",
ylab="Frequência", xlab= "Diâmetro", main=" Distribuição diâmetrica", las=1,
```

```
cex.axis=1.5, cex.lab=1.5)
    }
    if(estimativa== "DAP")# se a estimativa for diâmetro a altura do peito
    {
      AB= (pi*(2*(dados[ ,5])^2))# Calculo da área basal total
      AB.media= (pi*(2*(dados[ ,5])^2))/length(dados[ ,5])#Calculo da área
basal me´dia
      diam.mean= mean(dados[,5])# Calculo do diâmetro médio
      diam.min= min(dados[,5])# Calculo do diamentro mínimo
      diam.max= max(dados[,5])# Calculo do diâmetro máximo
      diam.plot.max= aggregate(dados[ ,5], list(dados[ ,1]), max) # diametro
maximo por plot
      diam.plot.min= aggregate(dados[ ,5], list(dados[ ,1]), min) #diametro
minimo por plot
      diam.plot.mean= aggregate(dados[ ,5], list(dados[ ,1]), mean)
      x11()
      plot.dist.diam= hist(dados[,5], breaks= 10, freq=T ,col="gray",
ylab="Frequência", xlab= "Diâmetro", main=" Distribuição diâmetrica", las=1,
cex.axis=1.5, cex.lab=1.5) ## Fazendo o plot da distribuição diâmetrica
    }
    ########################
    alt.max= max(dados[,6]) # altura máxima
    alt.min= min(dados [,6])# altura mínima
    alt.mean= mean(dados[,6]) # media da altura
    n.sp=length(table(dados[,4]))  # numero de espécies total
    n.ind.total= sum(length(dados[,3])) # numero total de individuos
    n.ind.sp= table(dados[ ,4]) ## numero de individuos por espécie
    n.sp.fam= table(dados[,3]) ## numero de espécies por família
    n.fam.total= length(n.sp.fam) ## numero total de família
    ##diversidade e equabilidade
    ## div shannon total
    prop.i= n.ind.sp/n.ind.total
    h.linha.total= -sum(prop.i*log(prop.i))
    ## div de simpson
    simpson.d.total= sum((prop.i)^2)
    ### Equabilidade de pielou
    pielou.j= h.linha.total/log(n.sp)
    ###Calculando densidade total
    dist.corr= (dados[,5]/200)+(mean(dados[,2])) ## Calculando a distancia
corrigida do ponto
    A=nplot*(dist.corr/n.individuos) ## calculando a área media de cada
ponto quadrante
    B= sum(dados[,2]/n.individuos) ## calculando a distancia média
    DTA= sum((A)/(B^2)) ## densidade total
    density.tot= DTA* (10000)  # densidade toal = individuos por hectare
    ########################################
    ########Calculando por ponto quadrante#########
    ####################################
    ### nº de indivíduos por parcela
    n.individuos.plot= table(dados[ ,1])
    ### Por plots
```

```
    tabela.especie.parcela= table(dados[ ,4], dados[ ,1]) ## selecionando
numero de individuos por espécie em cada parcela
    dimensao=dim(tabela.especie.parcela)## calculando dimensão para montar
matrix
    matrix.especie.parcela= matrix(tabela.especie.parcela, ncol=dimensao[2],
nrow = dimensao[1]) ## motando matrix
    matrix.especie.parcela.S.rar= matrix.especie.parcela # guardando a
matrix em um novo objeto para o calculod e riqueza estimada psoteriormente
    matrix.especie.parcela[(matrix.especie.parcela)==0] = NA ##
transformando valores em NA
    ## Nº de familias por plot
    n.sp.fam.plot= table(dados[,3], dados[,1]) ## numero de espécies por
família em cada plot
    #######
    n.especie.parcela= matrix(tabela.especie.parcela, ncol=dimensao[2], nrow
= dimensao[1]) ## numero espécie por parcela
    n.especie.parcela[(matrix.especie.parcela)>1] = 1 ##transformando em 0 e
1 para somar
    n.total.sp.plot= apply(n.especie.parcela,2, sum) ## usando apply para
calcular n total de sp por plot
    names(n.total.sp.plot)= c(levels(dados[ ,1])) # renomeando o vetor
    ################Criando função para diversidade de Shannon e Simpson
##########################
    shannon= function(y)
    {
      if (sum(is.na(y)>0))
      {dados.div=(na.omit(y))}
      else
      {dados.div=y}
      prop.i= dados.div/sum(dados.div)
      h.linha= -sum(prop.i*log(prop.i))
    }
    simpson= function(x)
    {
      if (sum(is.na(x)>0))
      {dados=(na.omit(x))}
      else
      {dados=x}
      prop.i= dados/sum(dados)
      simp= sum((prop.i)^2)
    }
    ######################################################################
    ########Calculando diverisdade, equabilidade por plot###########
    ############################################################
    div.plot.shannon= apply(matrix.especie.parcela,2,shannon) ## Calculando
Shannon por plot
    names(div.plot.shannon)= c(levels(dados[,1]))
    div.plot.simpson= apply(matrix.especie.parcela,2,simpson) ## Calculando
Simpson por plot
    names(div.plot.simpson)= c(levels(dados[,1]))
```

```
    ###Calculando equabilidade por plot
    pielou.j.plot= div.plot.shannon/ log(n.total.sp.plot)
    ########################################################
    ##### Calculando densidade e altura e diametro por plot######
    ########################################################
    # dimatro medio por plot
    ###Correção da distancia pelo DAP para calculo de densidade
    #### Calculando a densidade por ponto
    ####
    dados[,7]= (dados[,5]/200) ## nova coluna con distância corrigida
    dados[,7]= dados[,5]+(mean(dados[,2])) ## nova coluna com a distância
corrigida
    density.plot.P=tapply(dados[,7], list(dados[,1]), sum) ## calculando
distancia corrigida por plot
    A.plot= density.plot.P/n.individuos ## calculando valor de A
    dados[,8]= dados[,2]/n.individuos ## coluna considerando a distância do
ponto central divida pelo numero de individuos
    B.plot= tapply(dados[,8], list(dados[,1]), sum) ###
    DTA.plot= ((A.plot)/(B.plot^2)) ###
    density.plot= DTA.plot* (10000)  # densidade toal = individuos por
hectare
    ## altura por plot
    alt.plot.max= aggregate(dados[ ,6], list(dados[ ,1]), max) # alt max
    alt.plot.min= aggregate(dados[ ,6], list(dados[ ,1]), min) # alt min
    alt.plot.mean= aggregate(dados[ ,6], list(dados[ ,1]), mean) # alt media
########################################################################
########################################################################
########################################################################
########################################################################
###
    ##calculo da riqueza rarefeita ////
    ################################################################
    matrix.especie.parcela.2=matrix.especie.parcela.S.rar ## renomeando a
matrix
    matrix.especie.parcela.2[(matrix.especie.parcela.2)>1] = 1 ## colocando
matriz de prensença e ausendia
    matrix.especie.parcela.rarefacao= apply(matrix.especie.parcela.2,1,sum)
## aplicando somatoria para detectar espécies que ocorrem apenas em uma
amostra, e que ocorrem duas (singletons e doubletons)
n.sp.ocurr.1sample=length(matrix.especie.parcela.rarefacao[(matrix.especie.p
arcela.rarefacao==1)])##detectando espécies que ocorrem apenas em uma
amostra
n.sp.ocurr.2sample=length(matrix.especie.parcela.rarefacao[(matrix.especie.p
arcela.rarefacao==2)])##detectando espécies que em duas amostras
    n.sample= length(n.total.sp.plot) # tamanho da amostra
########################################################################
    ######### Riqueza estimada pela rarefação Colwel, et al
2012##############
########################################################################
    S.estimate= n.sp +  (((n.sample-1)/(n.sample)) *
((n.sp.ocurr.1sample*(n.sp.ocurr.1sample-1))/(2*(n.sp.ocurr.2sample+1))))
```

```
##Riqueza estimada baseada em amostra que ocorrem apenas 1 individuo =
    ######################################################################
    ############### Riqueza estimada por plot CHAO 1###################
    ######################################################################
    matrix.especie.parcela.S.rar.1= matrix.especie.parcela.S.rar ##
atribuindo novo nome para selecionar apenas especies que possuem 1 individuo
por parcela
    matrix.especie.parcela.S.rar.2= matrix.especie.parcela.S.rar ##
atribuindo novo nome para selecionar apenas especies que possuem 2
individuos por parcela
    n.individuos.plot= as.numeric(n.individuos.plot) ## transformando o
vetor de numero de individuos por plot em um vetor numerico
    matrix.especie.parcela.S.rar.1[(matrix.especie.parcela.S.rar.1)>1]=0 ##
transformando em 0 e 1 para calculo de especie que ocorrem com apenas um
individuo
    matrix.especie.parcela.S.rar.2[(matrix.especie.parcela.S.rar.2)!=2]=0##
excluindo tudo que é diferente de 2
    matrix.especie.parcela.S.rar.2[(matrix.especie.parcela.S.rar.2)==2]=1 ##
transformando em 0 e 1 para calculo das espécies que ocorrem com 2
individuos
    f1=apply(matrix.especie.parcela.S.rar.1,2,sum) #utilizando apply para
quantificar espécies que possuem apenas 1 indivíduo por amostra
    f2=apply(matrix.especie.parcela.S.rar.2,2,sum) # renomeando para a
fórmula
    ### calculando Riqueza rafeita pelo Chao 1, por parcelas ( mesma base
para construção do gráfico de riqueza)
    s.estimate.plot= n.sp
+(((n.ind.total-1)/(n.ind.total))*(f1*(f1-1))/(2*(f2+1))) ## Riqueza de
espécie estimada por plot
######################################################################
##############################
    ## Calculo dFrequência absoluta e relativa, dominância absoluta e
relativa, densidade absoluta e relativa, diâmetro médio (max e min), altura
média (max e min), índice de valor de importância, por espécie
######################################################################
##############################
    tabela.par.diam.sp= aggregate(dados[,5], list(dados[,4]), mean) ##
calculo do diâmetro médio
    names(tabela.par.diam.sp)= c("espécie","diâmetro médio")
    tabela.par.alt.sp=aggregate(dados[,6], list(dados[,4]), mean) ## calcula
da altura média
    names(tabela.par.alt.sp)= c("espécie","altura média")
    #### Densidade absoluta das espécies metodo ponto quadrante
    n.ind.sp=data.frame(n.ind.sp)
    dens.abs.sp=n.ind.sp
    dens.abs.sp[,3]= density.tot*(n.ind.sp[,2]/n.individuos) ## densidade
absoluta por hectare
    dens.abs.sp[ ,4]= (dens.abs.sp[,3]/sum(dens.abs.sp[,3]))*100 ##
densidade relativa em % densidade absoluta de cada espécie dividio pela
somatario da densidade total
```

```
    matrix.sp.param=matrix.especie.parcela.S.rar ## renomeando a matrix
    matrix.sp.param[(matrix.sp.param)>1] = 1 ## criando ua matrix de
presença e ausência
    dimnames(matrix.sp.param)= list(levels(dados[,4]),levels(dados[,1])) #
renomenado a matrix
    #Frequencia
    dens.abs.sp[ ,5]= (apply(matrix.sp.param,1,sum)/ length(nplot)) * 100 #
Em porcentagem afrequencia absoluta
    dens.abs.sp[,6]=(dens.abs.sp[,5]/(sum(dens.abs.sp[,5]))) * 100 #
frequência relativa
    ## dominancia
    dados[,9]= (((dados[ ,5])^2)*pi)/4 ## criando uma nova coluna do date
frame original para área basal
    ############################## Parenteses para calcuo da área basal por
plot ( aproveitando a nova coluna)
    AB.plot= aggregate(dados[,9], list(dados[,1]), sum)###calculando área
Basal por plot
    AB.plot.media=  aggregate(dados[,9], list(dados[,1]), mean) ## Area
basal média por plot
    ######################################################################
    #######
    ##
    dominancia.abs= aggregate(dados[,9], list(dados[,4]), sum)## usando
aggregante para juntar dados de area basal por espécie
    dens.abs.sp[,7]= dominancia.abs[,2] ## dominancia absoluta por especie
    dens.abs.sp[,8]=(dens.abs.sp[,7]/sum(dens.abs.sp[,7])) *100 ###
dominancia relativa por especie
    ##IVI
    dens.abs.sp[,9]= dens.abs.sp[,4] + dens.abs.sp[,6] + dens.abs.sp[,8]
########################################################################
######################
    ##### Motando a tabela dos parâmetros por espécie#################
########################################################################
##################
    tabela.param.sp= dens.abs.sp
    tabela.param.sp[,10]=  tabela.par.diam.sp[ ,2]
    tabela.param.sp[,11]=  tabela.par.alt.sp[ ,2]
    names(tabela.param.sp)= c("Espécie","Nº de indivíduos",
"Densidade","Densidade relativa"," Frequência","Frequência relativa",
"Dominância", "Dominância relativa", "IVI", "Diâmetro médio", "Altura
média")
    #################
    ##############
    #######
########################################################################
######################
    ####Tabela parametros totais###
    tabela.parametro.total= data.frame (
      nplot,
      density.tot,
      n.sp,
```

```
      n.ind.total,
      n.fam.total,
      alt.max,
      alt.min,
      alt.mean,
      AB,
      AB.media,
      diam.max,
      diam.min,
      diam.mean,
      h.linha.total,
      simpson.d.total,
      pielou.j,
      S.estimate)
    tabela.parametro.total= tabela.parametro.total[1,]
    names(tabela.parametro.total)= c( "número de parcelas","Densidade", "Nº
de espécies", "Nº de indivíduos", "N de famílias", "Altura máxima","Altura
mínina","Altura média", "Área basal","Área basal média","Diâmetro máximo",
"Diâmetro mínimo","Diâmetro médio", "Shannon", "Simpson", "Pielou","Riqueza
estimada")
##########################################################################
##############################
    ## Fazendo data frame com parametros por parcela
    tabela.parametro.plot= data.frame(n.individuos.plot, n.total.sp.plot,
density.plot,div.plot.shannon,div.plot.simpson,pielou.j.plot,s.estimate.plot
, alt.plot.max[,2],alt.plot.min[,2],alt.plot.mean[,2],    diam.plot.max[,2],
diam.plot.min [,2], diam.plot.mean[,2],         AB.plot[,2],
AB.plot.media[,2]) ## Juntando os dados
    names(tabela.parametro.plot)= c("Nº de individuos", "Nº de
espécies","Densidade", "Shannon", "Simpson", "Pielou","Riqueza estimada",
"Altura máxima","Altura mínima", "Altura média","Diâmetro máximo","Diâmetro
mínimo", "Diâmetro médio", "Área basal", "Área basal média") ## Renomeando
as colunas
    ##########################
    n.sp.fam=data.frame(n.sp.fam) ## criando um data frame com numero de
espécies por família
    names(n.sp.fam)= c("Família","Nº de espécie") ####
    ##########
    list.final=
list(tabela.parametro.total,tabela.param.sp,n.sp.fam,tabela.parametro.plot)
    names(list.final)  = c("Tabela total", "Tabela de espécies","Nº de
espécies por família","Tabela por Parcela")
    ##############parametro.plot
    ##########################################################################
    ###retornando o plot de altura
    plot.dist= hist(dados[,6], breaks= 10, freq=T ,col="gray",
ylab="Frequência", xlab= "Altura", main=" Distribuição de altura", las=1,
cex.axis=1.5, cex.lab=1.5)
    ### retornando o plot de riqueza estimada pela rarefação
    tabela.especie.parcela.plot= table(dados[ ,1], dados[ ,4]) ##
```

```
selecionando numero de individuos por espécie em cada parcela
    dimensao=dim(tabela.especie.parcela.plot)## calculando dimensão para
montar matrix
    matrix.especie.parcela.plot= matrix(tabela.especie.parcela.plot,
ncol=dimensao[2], nrow = dimensao[1])
    ## Instalando pacote necessário para calculo de Riqueza acmulada
    library(vegan) ## Carregando o pacote
    matrix.plot=specaccum(matrix.especie.parcela.plot,
"rarefaction",gamma="chao2",permutations = 1000) ## Calculando riqueza
rarefeita pelo chao 2 e permutando 1000 vezes
    x11()
    plot(matrix.plot, ci.type="poly", col="black", lwd=2, ci.lty=0,
ci.col="gray", las=1, ylab="Número de espécies", xlab="Número de
quadrantes", cex.lab=1.5, cex.axis=1.5, bty="l")
    ## Plotando curva de acumulação de espécies
    return(list.final)
  }
}  ###Selecionar o ultimo parênteses para finalizar a função!
```

Help

```
str.community                package: unknown              R Documentation


Estrutura da comunidade vegetal lenhosa
Description:

A função para estimar parâmetros para análises exploratórias de dados
de comunidades vegetais a partir de um data.frame de levantamento da
vegetação utilizando método de parcelas ou quadrantes.

Usage:

str.community (dados, method, estimativa="CAP", dim.plot)

Arguments:

dados    data.frame composto por seis colunas. A primeira coluna
         (dados[,1]) contendo o fator unidade amostral
         (“parcela” ou “quadrante”), a segunda coluna (dados[,2])
         contendo o número de indivíduos para o
         “method= “parcela”” e contendo a distância do indivíduo
         até o ponto central do quadrante para o “method=    “quadrante’.
         A terceira e a quarta coluna (dados[,3] & (dados[,4]) contendo
         família botânica e espécie, respectivamente. A quinta coluna
         (dados[,5]) contendo diâmetro a altura do peito (DAP)
         ou circunferência a altura do peito (CAP) e na última
         coluna (dados[,6]) contendo os valores de altura.
```

```
method   Método de levantamento de dados da vegetação.
         Para método de parcelas, method= “parcela”,
         para método de ponto quadrante
         method=”ponto quadrante”.

estimativa   Estimativa utilizada para cálculo de parâmetros
             como dominância e área basal. Esse argumento deve estar de
acordo
             com os dados da quinta coluna (dados[,5]).  Argumento
estimativa= “DAP”,
             para dados de circunferência a altura do peito (DAP) ou
estimativa= “CAP”
             para circunferência a altura do peito (CAP). Para dados de
circunferência
             e diâmetro a altura do solo vale a mesma notação, apenas
discriminando
             se a medida foi perímetro (CAP) ou diâmetro (DAP).

dim.plot      Tamanho da unidade amostral quando método é
             de parcelas method= “parcela”.
             Quando método utilizado forponto quadrante,
             method= “quadrante”, entrar com dim.plot= 0 .

Detalhes:

A função calcula por cada unidade amostral (parcela, ponto quadrante, área)
e pelo total de amostragem os seguintes parâmetros: área basal, densidade,
altura mínima, altura máxima, altura média, diâmetro máximo, diâmetro
mínimo, diâmetro média, número de indivíduos, número total de espécies,
número de famílias, número de espécies pro família, riqueza rarefeita para o
menor número de indivíduos (CHAO 1 para área total e CHAO 2 para plots) que
retorna o número absoluto estimado de espécie, diversidade de Shannon e
Simpson e  equabilidade de Pielou.
A função também calcula os parâmetros por espécies: número de indiíduos,
frequência absoluta e relativa, dominância absoluta e relativa, densidade
absoluta e relativa, diâmetro, altura média, índice de valor de importância.
A função também incorpora função “hist” e “plot” do pacote básico e função
“specaccum” do pacote “vegan”. É necessário ter pacote “vegan” instalado
previamente (install.packages(“vegan”)).
A função possuí opções de entrar com dados levantados pelo método de
quadrante e método de parcelas. Também permite entra com dados de perímetro
do tronco (CAP), pois ela calcula e retorna valores de diâmetro. Para que a
função realize os cálculos de densidade é necessário entrar com as dimensões
da parcela no argumento dim.plot. As dimensões das parcelas devem estar
multiplicando os valores em metros de cada lado da parcela (ex,
dim.plot=10*10). Se o método utilizado for o método de quadrante, utilizar
dim.plot=0. Isso devido a cada ponto quadrante possuir valores diferentes de
dimensões para os cálculos de densidade. A densidade de cada ponto quadrante
leva em consideração a distância dos indivíduos até o ponto central e a área
ocupada por cada indivíduo.
```

```
Value:
 A função retorna um objeto list contendo quatro data.frame: “Tabela total”,
“Tabela de espécies”, “Nº de espécies por família”, “Tabela por unidade
amostral”.  Tabelas total contém estimativas para toda a comunidade. Tabela
de espécies contém os parâmetros para todas as espécies amostradas na
comunidade. Nº de espécies por família, contém número de espécies por
família em toda comunidade amostrada. Tabela por unidade amostral contém as
estimativas por unidades amostrais contidas no data.frame inicial (dados).
A função também retorna dois parâmetros gráficos, primeiro referente a
distribuição diâmétrica e segundo referente a curva de acumulação de espécie
pelo método de rarefação utilizando 1000 permutações.

Warning:
A função aceita como dados apenas objetos da classe "data.frame".
Se method= “quadrante” a função só aceita o argumento dim.plot= 0.

Note:
.....
Author(s):
André Luiz Giles
Giles, A.L. (andregiles74@yahoo.com.br)

References:

Coleman, B.D. 1981. On random placement and species-area relations.
Mathematical Biosciences 54, 191-215.
Coleman, B.D., Mares, M.A., Willig, M.R. & Hsieh, Y.-H. 1982. Randomness,
area, and species richness. Ecology 63, 1121-1133.
Colwell, R. K., A. Chao, N. J. Gotelli, S.-Y. Lin, C. X. Mao, R. L. Chazdon,
and J. T. Longino. 2012. Models and estimators linking individual-based and
sample-based rarefaction, extrapolation, and comparison of assemblages.
Journal of Plant Ecology 5:3-21.
Moro, M.F. & Martins F.R. 2011. Métodos de levantamento do componente
arbóreo-arbustivo. In Fitossociologia no Brasil: métodos e estudos de casos
(J.M. Felfili, P.V. Eisenlohr, M.M.R.F Melo, L.A. Andrade & Neto J.A.A.M.,
eds.). Editora UFV, Viçosa, p.174-212.

See Also:
……..

Examples:
### Criando vetores para exemplo
ponto.quadrante= rep(1:10, each= 4)
distancia.do.ponto= seq(from=5,to=200, by=5)
familia= c ("Rubiaceae,"   ,     "Lauraceae" ,      "Chrysobalanaceae"
,"Polygonaceae"  ,   "Fabaceae",
"Flacourtiaceae" ,  "Flacourtiaceae", "Rubiaceae" ,       "Lauraceae"  ,
"Euphorbiaceae",
 "Rubiaceae"   ,     "Euphorbiaceae" ,   "Rubiaceae" ,       "Sapindaceae"
,   "Arecaceae"  ,
"Sapindaceae"   ,   "Sapindaceae"   ,   "Rubiaceae" ,       "Rubiaceae"    ,
```

```
"Elaeocarpaceae" ,
"Lauraceae"    ,    "Phyllanthaceae"  , "Rubiaceae"  ,      "Annonaceae"   ,
"Myrtaceae",
"Euphorbiaceae" ,   "Phyllanthaceae",   "Arecaceae" ,       "Myrtaceae"
,  "Sapindaceae",
"Phyllanthaceae",   "Phyllanthaceae"  , "Elaeocarpaceae",   "Phyllanthaceae"
, "Arecaceae",
"Elaeocarpaceae" ,  "Arecaceae", "Fabaceae", "Rubiaceae", "Sapindaceae")

especie= c ("Psychotria nuda" ,          "Ocotea silvestris"   ,
"Licania octandra"  ,
          "Coccoloba sp."      ,        "Sclerolobium denudatum"  ,
"Casearia obliqua"  ,
           "Casearia obliqua"    ,       "Psychotria mapoureoides",
"Ocotea dispersa" ,
           "Pera glabrata" ,             "Psychotria nuda"  ,          "Pera
glabrata" ,
           "Psychotria nuda"   ,         "Cupania oblongifolia",
"Euterpe edulis" ,
           "Cupania oblongifolia" ,      "Cupania oblongifolia" ,
"Psychotria mapoureoides",
           "Psychotria nuda",            "Sloanea guianensis" ,
"Aniba viridis",
           "Hyeronima alchorneoides" ,   "Psychotria nuda" ,
"Xylopia langsdorffiana",
           "Myrcia pubipetala"  ,        "Pera glabrata"   ,
"Hyeronima alchorneoides" ,
          "Astrocaryum aculeatissimum", "Eugenia catharinensis" ,
"Cupania oblongifolia",
          "Hyeronima alchorneoides" ,   "Hyeronima alchorneoides",
"Sloanea guianensis",
           "Hyeronima alchorneoides"   , "Euterpe edulis"   ,
"Sloanea guianensis" ,
          "Attalea dubia"   ,           "Sclerolobium denudatum"  ,
"Psychotria mapoureoides" ,
          "Cupania oblongifolia")

CAP= c(18 , 41 , 73  ,38 ,145 , 19 , 18 , 56 , 38,  85,  19,  21 , 22,  22 ,
29,  35,  25,  85,  24,  28,  22,  68,  15,  16,  71,  33,  61  ,25 , 18 ,
12,  51 , 58,  59,  40 , 20 , 35 , 72, 112,  40 , 12)
altura= c(3, 12, 15,  8, 20,  4,  6, 12,  7, 12,  4,  6,  4,  6, 6,  8 , 5
,12 , 4,  7 , 7 ,20,  3 , 4, 20 ,11, 15 , 3 , 4 , 7 ,12, 14 ,10 , 8 , 3, 9,
7 ,17 , 3 ,10)

##Exemplo de quadrante
dados.exemplo.quadrante= data.frame (ponto.quadrante,distancia.do.ponto,
familia,especie,CAP,altura) ## Concatenando os vetores
## Rodando a função para quadrante
str.community(dados = dados.exemplo.quadrante, estimativa = "CAP",
method="quadrante", dim.plot=0)
```

```
### Exemplo de parcelas
n.individuos= seq(from=1, to=40, by=1)
parcela= rep(1:4, each= 10)
dados.exemplo.parcela = data.frame (parcela,n.individuos,
familia,especie,CAP,altura)  #Concatenando os vetores
## roando  a função para dados de parcela
str.community(dados = dados.exemplo.parcela, estimativa = "CAP",
method="parcela", dim.plot=10*10)
###############
```

Código da função str_community