Função dens.flux()

Arquivo da função: dens.flux.r

```
dens.flux<- function (df, df2, graphic= TRUE)   #estabelecer os argumentos
da função "dens.flux"

{
  if (class (df) != "data.frame")               #começar a verificar
parâmetros: se a classe de "df" for diferente de dataframe
  {
    stop ("df precisa ser da classe data frame")#parar a função e emitir
mensagem de erro
  }
  if (class (df [,1]) != "factor")              #se a classe do vetor que
compõe a primeira coluna de "df" não for fator
  {
    df [,1]<- as.factor (df [,1])               #transformar para fator
  }
  if (class (df [,2]) != "character")           #se a classe do vetor que
compõe a segunda coluna de "df" não for caracter
  {
    df [,2]<- as.character (df [,2])            #transformar para caracter
  }
  if (class (df [,3]) != "numeric")             #se a classe do vetor que
compõe a terceira coluna de "df" não for numérico
  {
    df [,3] <- as.numeric (df [,3])             #transformar para numerico
  }
  if (class (df2 [,1]) != "factor")             #se a classe do vetor que
compõe a primeira coluna de "df2" não for fator
  {
    df2 [,1]<- as.factor (df2 [,1])             #transformar para fator
  }
  if (class (df2 [,2]) != "integer")            #se a classe do vetor que
compõe a segunda coluna de "df2" não for integer
  {
    df2[,2] <- as.integer (df2[,2])             #transformar para integer
  }
  if (class (df2 [,3]) != "integer")            #se a classe do vetor que
compõe a terceira coluna de "df2" não for integer
  {
    df2[,3] <- as.integer (df2[,3])             #transformar para integer
  }
  rotacoes<- (df2[,2]) >= (df2[,3])             #criar vetor lógico
"rotacoes" para verificar se os valores da segunda coluna de "df2" são
menores ou iguais aos valores da terceira coluna
  rota.verif<- any (rotacoes)                   #criar objeto "rota.verif"
```

```
para verificar se existe algum TRUE no vetor "rotacoes"
  if (rota.verif == "TRUE")                    #se "rota.verif" for TRUE
  {
    stop (" Os valores da coluna 2 não podem ser maiores ou iguais aos
valores da coluna 3 em df")  #parar a função e emitir mensagem de erro
  }
  if (class(df2[,4]) != "numeric")             #se a classe do vetor que
compõe a quarta coluna de "df2" não for numérico
  {
      df2[,4]<- as.numeric (df2[,4])           #transformar para numérico
  }
  area.zero<- (df2[,3] <= 0)                   #criar vetor lógico
"area.zero" para verificar se os valores da terceira coluna de "df2" são
menores ou iguais a zero
  a.zero<- any (area.zero)                     #criar objeto "a.zero" para
verificar se existe algum TRUE no vetor "area.zero"
  if (a.zero == "TRUE")                        #se houver TRUE em "a.zero"
  {
    stop (" A área da rede não pode ser menor ou igual a 0")  #parar a
função e emitir mensagem de erro
  }
  fator.zero<- (df2[,4] <=0)                   #criar vetor lógico "fator
zero" para verificar se os valores da quarta coluna de "df2" são menores ou
iguais a zero
  f.zero<- any (fator.zero)                    #criar objeto "f.zero" para
verificar se existe algum TRUE no vetor "fator.zero"
  if (f.zero == "TRUE")                        #se "f.zero" for TRUE
  {
    stop (" O fator de calibração do equipamento não pode ser menor ou igual
a 0")   #parar a função e emitir mensagem de erro
  }
  n.df<- is.na(df)                             #criar vetor lógico "n.df"
para verificar se existem NAs em "df"
  na.df<- any (n.df)                           #criar objeto "na.df" para
verificar se há algum TRUE em "n.df"
  if (na.df == "TRUE")                         #se "na.df" for TRUE
  {
    df<-na.omit (df)                           #omitir linhas que contêm
NAs
    warning ( "Atenção, as linhas de df que contêm NA foram ocultadas")
#emitir mensagem de aviso
  }
  n.df2<- is.na(df2)                           #criar vetor lógico "n.df2"
para verificar se existem NAs em "df2"
  na.df2<- any (n.df2)                         #criar objeto "na.df2" para
verificar se há algum TRUE em "n.df2"
  if (na.df2 == "TRUE")                        #se "na.df2" for TRUE
  {
    df2<-na.omit (df2)                         #omitir as linhas que contêm
NAs
```

```
    warning ( "Atenção, as linhas de df2 que contêm NA foram ocultadas")
#emitir mensagem de aviso
  }
  niveis1<- levels (df [,1])                    #verificar os níveis da
primeira coluna de "df1"
  niveis2<- levels (df2 [,1])                   #verificar os níveis da
primeira coluna de "df2"
  niveis.dif<- (niveis1 != niveis2)             #criar o vetor lógico
"niveis.dif" para verificar se os níveis da primeira coluna de "df" e "df2"
são os mesmos
  niveis.real<- any (niveis.dif)                #criar objeto "niveis.real"
para verificar se existe algum TRUE no vetor "niveis.dif"
  if (niveis.real == "TRUE")                    #se "niveis.real" for TRUE
  {
    stop ("Os data frames precisam ter o mesmo número de amostras")
#parar a função e emitir mensagem de erro
  }
  tam<- length (niveis1)                        #criar objeto "tam" para
guardar o comprimento de "niveis1"
  for ( i in 1: tam)                            #criar um ciclo que vai de i
até o comprimento de "níveis1"
  {
    for.df<- df [df [,1] == niveis1[i],]        #criar o data frame "for.df"
que vai conter somente os valores referentes a amostra i
    double.vector <- duplicated (for.df [,2])   #criar o vetor lógico
"double.vector" que vai verificar se existem valores duplicados na segunda
coluna de "for.df"
    double<- any (double.vector)                #criar objeto "double" para
verificar se há algum TRUE em "double.vector"
    if (double == "TRUE")                       #se "double" for TRUE
    {
      stop ( paste ( "Espécie repetida na amostra", for.df [1,1])) #parar a
função e emitir mensagem de erro com referência ao fator em que há uma
duplicação
    }
  }
  if (names(df[1]) != names(df2[1]))            #se os nomes da primeira
coluna de "df" e "df2" não forem iguais
  {
    names (df)[1] <- "Amostras"                 #mudar nome da primeira
coluna de df para "Amostras"
    names (df2) [1] <- "Amostras"               #mudar nome da primeira
coluna de df2 para "Amostras"
    warning ("Atenção, o nome da primeira coluna de df e df2 foi alterado
para 'Amostras'") #emitir mensagem de aviso
  }
  df3<- merge (df,df2)                          #juntar os data frames "df"
e "df2" em um novo data frame "df3"
  dens<- df3[,3]/((df3[,5]-df3[,4])*df3[,6]*df3[,7])
#aplicar a equação indexando as colunas de "df3"
  df$dens<- dens                                #inserir o novo vetor "dens"
```

```
em "df"
  if (graphic)                                    #se o usuário optar por
gerar o gráfico
  {
    med.dens<- tapply (dens, df[,1], mean)      #aplicar a função média nos
valores de "dens" para cada fator da primeira coluna de "df"
    par(mar=c(4,5,4,4))                         #definir padrão de margem do
gráfico
    barplot (med.dens, col= "blue", xlab= "Amostra", ylab= "Densidade
(org/m³)",main= "Densidade média das amostras",  las= 1, tcl=0.3)   #plotar
gráfico e definir legendas dos eixos, cor das barras, título, posição do
eixo y e comprimento das linhas dos eixos
  }
  return (df)                                     #retornar o dataframe
inicial "df" que agora possui uma nova coluna de valores
  }                                               #fim da função
```