# Jeronymo Dalapicolla

Biólogo e Capixaba
Doutorando em Ecologia Aplicada (ESALQ/CENA)
Escola Superior de Agricultura "Luiz de Queiroz"
Universidade de São Paulo

### Tema de Estudo

Trabalho com Taxonomia, Evolução e Biogeografia de pequenos mamíferos, especialmente roedores e marsupiais. Na tese trabalho com variação morfológica e genética de roedores amazônicos, os ratos-de-espinho do gênero *Proechimys*.

### Contatos

Currículo Lattes
jdalapicolla@usp.br
jdalapicolla@gmail.com

---

## Exercícios

Exercícios - Jeronymo
Aula 2 - Funções Matemáticas
Aula 3 - Leitura e Manipulação de Dados
Aula 4 - Análise Exploratória
Aula 5 - Criação e Edição de Gráficos
Aula 6 - Testes de Significância
Aula 7a - Regressão Linear Simples
Aula 7b - Regressão Linear Múltipla
Aula 8 - Reamostragem e Simulação
Aula 9 - Construção de Funções Simples

---

## Trabalho Final

### Proposta A

### Contextualização

Análises Morfométricas são importantes para diversas áreas como a Engenharia Florestal e a Agronomia. Nas Ciências Biológicas são análises amplamente usadas na Taxonomia, Sistemática, Ecologia, Genética Quantitativa, entre outras. Normalmente os dados são coletados com paquímetro digital ou digitalizador e, mesmo reduzido no digitalizador, há sempre uma possibilidade de erro na mensuração, seja por erros de digitação, na manipulação de colunas ou linhas das planilhas, ou por inconsistências dos equipamentos. Existem várias formas de reduzir os erros, uma delas é fazer uma análise exploratória dos dados, antes de qualquer análise estatística. Diversas análises exploratórias são factíveis de realização, e diversos trabalhos se dedicam a esse tema. Por exemplo Zuur *et al.* 2010 comentou e explicou oito tipos de análises exploratórias que podem ser feitas com dados ecológicos. Essa proposta visa a criação de uma função que faça cinco dessas oito abordagens de Zuur *et al.* que são aplicáveis a dados morfométricos. Mais detalhes no linque abaixo:

TESTE DE PREMISSAS

Comentários Bruno

Oi Jeronymo, Sua proposta A parece ser interessante. Talvez você possa incluir uma coluna de localidade por exemplo para poder fazer essas análises exploratórias entre as diferentes localidades.

Na proposta B, acho que ao invés de não aceitar Nas, seria interessante pensar em como lidar com os NAs. Além disso, testar a normalidade dos dados que serão utilizados.

Bruno

Resposta Jeronymo

Olá Bruno!

Vou seguir com a proposta A. Quando eu sugeri o argumento *group* também imaginei ele funcionando com uma coluna de localidades. Mas acho que vai ficar bem complicado, mas vou tentar!

Muito Obrigado
Jeronymo

## Proposta B

### Contextualização

Outro grupo de análises comumente usado na Morfometria são as análises multivariadas. As três análises multivariadas mais utilizada na Zoologia para fins taxonômicos são: Análise de Componentes Principais, Análise de Função Discriminante e MANOVA. A maioria das funções para essas análises,

contém apenas o script das análises, sem a representação gráfica que algo essencial para melhor visualização dos dados. Muitas vezes é necessário criar outras funções para a criação dos gráficos em pacotes mais complicados, com muitos argumentos. Essa proposta visa uma função mais simples para as 03 análises multivariadas e seus respectivos gráficos. Mais detalhes da proposta no linque abaixo:

ANÁLISES MULTIVARIADAS

---

Olá Jeronymo, uma questão importante das suas duas propostas é que elas são basicamente um script de análise de dados e não necessariamente uma função. Acho que você tomou o cuidado de tornar os passos escolha do usuário e implementados de modo geral dentro da função. Mas ainda assim, eu me preocuparia com alguns pontos. Não entendi o porquê de precisar gerar os arquivos .csv. Me parece apenas para visualizar os dados, é isso? (pra que tirar do R as informações se podemos lidar com ela no R?). Acho mais interessante que a função retorne 5 listas, cada uma contendo os outputs das 5 explorações que você propõe. Você também pode deixar o usuário escolher quais gráficos quer ver (isto vale para a segunda proposta também. Uma dúvida em relação à segunda proposta é como você vai calcular pca, adf e manova? O ideal é que mesmo que estas funções já existam no R, você implemente à sua maneira. Apenas usar as funções já implementadas no R me parece algo muito simples. Sugiro que você siga com a proposta que toque mais seu coração e te motive mais a encarar desafios em termos de programação. — *Sara Mortara*

Olá Sara! Agora fiquei desanimado :(
Na sexta passada eu só vi o comentário do Bruno e não o seu. Então comecei a trabalhar na primeira função. Eu realmente não entendo o que se quer como função. Isso me deixa bastante frustado (o fato de eu não conseguir entender). Eu teria que usar usar funções primárias? Trabalhar do zero? Reinventar a roda? Porque tudo que eu imaginei já tinha função pronta. E eu não preciso usar um script com outras funções para fazer coisas novas?

Eu montei nessa semana a parte da função da proposta A, a que se refere aos outliers. Vou te mandar o script aqui e o meu banco de dados aqui. Eu tava todo animado conseguindo fazer loop, modificando nomes, criando pastas, agora já não sei se estou no caminho certo. Com relação aos resultados finais, eu prefiro que a saída seja em um .csv para

a visualização mesmo, ainda não tenho prática de manipulação de listas, posso usar um array ou invés de matriz para guardar os resultados em listas se preferir. Para os gráficos dotchart e bivariáveis estou tentando sem sucesso colocar a label/nome da linha dos pontos, mas só para os outliers indicados pelo boxplot para não ficar muito poluído a imagem. Queria ver as posições só dos outliers nos gráficos.

Pensei em fazer funções para gráficos das análises multivariadas como o pca, construir os gráficos do zero, montando cada parâmetro por vez, mas seria muito simples, por isso da segunda proposta. Com relação a ela, eu usaria com funções já existentes para fazer o PCA e as outras. Não sei como calcular um PCA ou MANOVA na mão para fazer à minha maneira. Sem contar a chance de eu errar na matemática. Enfim, agora estou muito perdido, não sei se continuo, se faço outra coisa, alguma outra função seria melhor? Alguma sugestão? O tempo tá acabando... tenho até dia 15 para terminar, né?

---

### Proposta A - REFORMULADA

#### Contextualização

Em vários campos da Ciência o uso de variáveis quantitativas é importante. Essas variáveis podem ser obtidas de várias formas e com os mais diversos equipamentos de mensuração, dependendo do objetivo do trabalho e de qual análise é necessário fazer. Entretanto há algo em comum a todos os trabalhos que usam dados quantitativos, os *outliers*. Há sempre uma possibilidade de erro na mensuração, seja por erros de digitação, na manipulação de colunas ou linhas das planilhas, ou por inconsistências dos equipamentos. Existem várias formas de identificar os *outliers*, e essa proposta visa a criação de uma função que abarque várias dessas opções. O usuário ficaria responsável por escolher qual das abordagens usar (ou todas) para a identificação dos *outliers* e como seria o *output*. Mais detalhes no linque abaixo:

OUTLIERS

# Página de Ajuda

```
id.outliers{unknown}                    package:unknown                    R
```

```
Documentation


                        IDENTIFICAÇÃO DE OUTLIERS EM DADOS NÚMERICOS


     DESCRIPTION:
     Função para a identificação de possíveis outliers em um data frame com
dados númericos (quantitativos). São gerados até três objetos de saídas:
tabelas em formato .csv contento os outliers, gráficos em formato .jpeg para
visualização dos outliers e um objeto tipo "list" contendo os outliers para
manipulação dentro do ambiente R.
     USAGE:
     id.outliers(x=data, quant=6:30, group=0, id="box", NUMBER=10,
visual="boxplot", res="LOW", csv=FALSE)
     ARGUMENTS:

     "x": é o data frame, a tabela com todas as colunas e linhas;
     "quant": um vetor contendo as posições das colunas que representem as
variáveis quantitativas/numéricas de "x";
     "group": indica apenas uma coluna do data frame que contém a informação
dos subgrupos de interesses (pode ser espécies, áreas, experimentos, idades,
sexo, localidades etc.). Com esse argumento indicado, a função transformará
a tal coluna em um objeto da classe "factor" e os testes serão realizados
para todos os níveis desse fator separadamente. Se não for definido, o teste
será feito considerando todas as observações do data frame como pertencentes
ao mesmo grupo;
     "id": indica o algoritmo utilizado para a identificação dos outliers.
Existem quatro opções, o default é "box":
        id=“box”: utilizará a função boxplot() para a identificação dos
outliers;
        id=“z”: utilizará o teste modified Z-Score (Iglewicz & Hoaglin,
1993);
        id=“ESD”: utilizará o teste generalized ESD test (Rosner 1983) para
a identificação do número de outliers e não quais são eles. Para esse teste
os dados devem estar distribuídos próximo a curva normal;
        id=“ALL”: ulitizará todos os três testes acimas;
     "NUMBER": Argumento necessário só se o método "ESD" ou "ALL" forem
selecionados. Indica o número máximo de outliers que as variáveis podem ter.
O default é 10, significa que o teste analisará a possibilidade de "quant"
ter entre 1 e 10 outliers. Não há limite máximo, mas é recomendado testar no
máximo a metade do número amostral.
     "visual": determina a o tipo de gráfico para a visualização dos
outliers, o default é "boxplot":
        visual=“boxplot”: utilizará a função boxplot() para a construção do
gráfico;
        visual=“pontos”: utilizará um gráfico de pontos com a função
dotchart();
        visual=“biplot”: um gráfico de dispersão com duas variáveis será
construído. Uma das variáveis do gráfico será aquela com os outliers
identificados e a outra variável será retira da lista de variáveis
```

```
quantitativas informada pelo argumento "quant";
        visual="ALL": criará todos os três gráficos acima;
     "res": determina a resolução dos gráficos. Terá três opções, o default
é LOW:
        res="LOW": qualidade alta, 150 dpi;
        res="MED": qualidade média, 300 dpi;
        res="HIGH" qualidade alta, 600 dpi;
     "csv": indica se a função criará um output com os outliers no formato
.csv para a utilização em outros programas. O default é FALSE, a função
retornará apenas uma lista com os outliers.
     DETAILS:
     Para a realização das análises, cada variável quantitativa deve
apresentar cinco ou mais observações. Se esse número não for alcançado a
função retorna uma mensagem no console informando que não foi possível
realizar tal teste, para tal variável, pelo número de amostras ser
insuficiente;
     Para a realização do generalized ESD test (Rosner 1983), id="ESD",o
autor recomenda que haja mais de 25 observações para que o teste tenha poder
máximo, mas que 15 amostras são suficientes para uma boa aproximação dos
resultados. Entre cinco e 14 amostras o teste pode ser realizado, mas seus
resultados devem ser analisados com cautela. Menos de cinco amostras o teste
não pode ser feito.
    Para a realização do generalized ESD test (Rosner 1983), id="ESD", o
autor recomenda que as observações tenham uma distribuição normal. É
recomendado antes de usar o teste, verificar se as amostras se aproximam da
distribuição normal.
    VALUE:
    A função retorna uma lista com outras três listas, cada lista é
referente a um método de análise do argumento "id". Quando um dos métodos
não for usado, ou não apresentar outliers a lista referente estará
vazia,isto é, NULL.

        list1 : contém os resultados do método "box" e terá o número de
elementos igual ao comprimento de "quant", nomeados com o nome da variável.
Quando uma variável não tiver outliers identificado, a posição na lista
ficará vazia.

        list2 : contém os resultados do método "z" e terá o número de
elementos igual ao comprimento de "quant", nomeados com o nome da variável.
Quando uma variável não tiver outliers identificado, a posição na lista
ficará vazia.

        list3 : contém o número de elementos igual ao comprimento de
"quant", nomeados com o nome da variável. Dentro de cada elementoda lista,
há a tabela gerada pelo método "ESD" que terá os números de outliers, o
valor estatítico e valor crítico. Quando o valor estatístico supera o valor
crítico significa que há outliers. A função retorna no console o número de
possíveis outliers, a lista é apenas para conferência  dos valores
estatísticos e para publicação. Se uma variável não tiver outliers
identificado, a posição na lista ficará vazia.
```

```
    WARNING:
    Evite dar nome extensos às linhas, por exemplo, com mais de três
dígitos. Não indique rownames para o data frame usado na função, ao invés,
deixe o R usar a posição das linhas na tabela. A função rodará perfeitamente
com qualquer rownames, mas na hora da contrução dos gráficos, nomes extensos
causam muita poluição visual e atrapalham a identificação visual dos
outliers.



    EXAMPLES:
       #input sem rownames
       x= read.table("dados.csv", header = T,sep=",", as.is = T)

       out=id.outliers(x, 6:30, 4, "ALL", 18, "ALL", "MED", TRUE)
       out2=id.outliers(data, 9:15)



    NOTE:
    IGLWICZ, Boris; HOAGLIN, David. Volume 16: How to Detect and Handle
Outliers. IN: MYKYTKA, Edward F.(ed.), The ASQC Basic References in Quality
Control: Statistical Techniques, 1993.

    ROSNER, Bernard. Percentage Points for a Generalized ESD Many-Outlier
Procedure. Technometrics, 25(2), pp. 165-172, 1983.

    ZUUR, Alain F.; IENO, Elena N.; ELPHICK, Chris S. A protocol for data
exploration to avoid common statistical problems. Methods in Ecology and
Evolution, v. 1, n. 1, p. 3-14, 2010.
   AUTHOR:
   DALAPICOLLA, J. 2016. id.outliers: função para identificação de outliers
em dados numéricos. Disponível em:
<http://ecologia.ib.usp.br/bie5782/doku.php?id=bie5782:01_curso_atual:alunos
:trabalho_final:jdalapicolla:start>
```

---

# Código da Função

```
id.outliers = function (x, quant, group=0, id="box", NUMBER=10,
visual="box", res="LOW", csv=FALSE)
{
  if (group!=0) #Para quando houver subgrupos para analisar
  {
    fatores=as.factor(unique(x[,group])) #extrair os valores únicos da
coluna que sera usada como subgrupo e transforma-los em fatores
    for (fator in fatores)#fazer as analises para cada fator, em ciclo
    {
      df= x[x[,group]==fator,] #extrair todas as linhas do data frame
```

```
original que contem o fator de interesse e salvar cada uma delas como 'df'
      resultados = etapa2 (df, quant, fator, id, NUMBER, csv) #usar a funcao
auxiliar 'etapa2' para a identificacao dos outliers e salvar a lista
resultante em 'resultados'
      etapa3 (df, quant, fator, visual, res) # usar a funcao auxiliar
'etapa3' para a elaboracao dos graficos
    }
  }
  else #Quando nao houver subgrupos para analisar
  {
    df=x #considerar a tabela toda como df
    fator="tabela_completa" #considerar o fator com esse nome
    resultados = etapa2 (df, quant, fator, id, NUMBER, csv) #funcao auxiliar
para identificacao dos outliers
    etapa3 (df, quant, fator, visual, res)#funcao auxiliar para a
identificacao dos outliers
  }
  return(resultados) #retorna uma lista dos os possiveis outliers
identificados
}

#####################################################################
######### FUNCOES AUXILIARES ##################################

mudarSubDir = function (fator, subfolder)
  {#funcao auxiliar para mudar o diretorio antes de salvar graficos ou csv
      if (dir.exists(file.path(getwd(), fator)) == FALSE) #se a pasta com o
nome do fator nao existir
        {
        dir.create(fator, showWarnings = FALSE)# crar a pasta com o nome do
fator
        }
      setwd(file.path(getwd(), fator)) #mudar para a pasta criada
      if (dir.exists(file.path(getwd(), subfolder)) == FALSE) #se o
subfolder nao existir
        {
        dir.create(subfolder, showWarnings = FALSE)# criar subfolder
        }
      setwd(file.path(getwd(), subfolder)) #mudar o diretorio para o
subfolder
  }

#####################################################################
###########

rval = function(y) #funcao auxiliar para calcular o ESD
  {
  ares = abs(y - mean(y, na.rm = TRUE))/sd(y, na.rm = TRUE)
  tab = data.frame(y, ares)
  r = max(tab$ares)
```

```
  list(r, tab)
  }

#########################################################################
###########

etapa2= function (df, quant, fator, id="box", NUMBER=10, csv=FALSE)
{
  resultado.box=list()#criar listas para salvar os resultados dos tres
metodos
  resultado.z=list()
  resultado.ESD=list()

  if (id=="box" | id=="ALL")#se o metodo for 'box'
    {
      fatorSemObs=TRUE #variavel que vai indicar a existencia ou nao de
outliers
      loop=0 #variavel que vai indicar a qual e o ciclo do loop
      for (i in quant)#ciclo para cada coluna quantitativa indicada no
inicio da funcao
        {
          loop=loop+1#incremento para o numero de loop
          if (length(df[,i])>=5) #se o numero de amostras para tal coluna
for menor do que 5 nao e possivel indicar os outliers, vai para o proximo
'i'
              {
                graph=boxplot(df[i], data=df, main= colnames(df[i]),
na.omit(df[i])) #cria o boxplot em um objeto, uma lista dentro dele chamado
'out' tem os valores dos possiveis outliers
                posi.out= which (df[,i] %in% graph$out) #descobrir em qual
posicao da tabela esta esses valores de outliers
                nobs=length(graph$out) #numero de outliers encontrados
                if (nobs != 0) #se o numero de outliers for diferente de 0
                  {
                  fatorSemObs=FALSE#muda o valor da variavel indicativa
                  out.lines= as.data.frame(matrix(NA, nrow= nobs,
ncol=ncol(df))) #cria uma tabela para salvar os resultados
                  colnames(out.lines)= colnames(df) #dar os nomes para as
colunas da tabela
                      for (j in 1:nobs) #para cada observacao de outliers
                        {
                        out.lines[j,]= df[posi.out[j], ] #salvar a linha
inteira da tabela original "df" que contenha o valor de outlier na tabela de
resultados
                        }
                  resultado.box[[loop]]= out.lines#salvar a tabela de
resultados no objeto lista final na posicao indicada pelo loop. 1ºciclo
posicao 1, 2ºciclo posicao 2, etc.
                  names(resultado.box)[loop]= colnames(df[i]) #dar o nome
para cada tabela salva na lista
                  if (csv==TRUE) #se foi pedido para salvar csv
```

```
                        {
                        mudarSubDir(fator, "box") #usar funcao auxiliar para
mudar o diretorio
                        write.table(out.lines,
file=paste(colnames(df[i]),"csv", sep="."), sep=",") #salvar resultado como
csv
                        setwd('../..')#voltar ao diretorio original
                        }
                    }
                }
          else #se nao tiver mais 4 amostras
                {
                  print(paste(colnames(df[i])," do agrupamento ",fator," nao
possui numero de amostras suficientes para realizar as analises (n>=5)",
sep="")) #print a mensagem do erro
                  next #ir para o proximo ciclo
                }
        }
      if (fatorSemObs==TRUE) print (paste(fator, "nao tem outliers para
metodo 'box'", sep=" "))#se nao teve outliers identificados imprimir essa
mensagem
    }
  if (id=="z" | id=="ALL")#se for o metodo 'z'
    {
      fatorSemObs=TRUE#variavel que vai indicar a existencia ou nao de
outliers
      loop=0#variavel que vai indicar a qual e o ciclo do loop
      for (i in quant)#ciclo para cada coluna quantitativa indicada no
inicio da funcao
          {
            loop=loop+1#incremento para o ciclo do loop
            obser=sapply(df[,i], as.numeric)#salvar valores da coluna do
data frame original como 'numeric'
            mod.z= abs((0.6745*(obser-median(obser, na.rm=TRUE)))/
mad(df[i], na.rm=TRUE))#calculo da estatistica
            if (any(mod.z > 3.5, na.rm = TRUE)==TRUE)#se o valor da
estatistica for maior que 3.5, ha outliers
            {
              fatorSemObs=FALSE#mudar variavel indicativa de outliers
              positions=which(mod.z> 3.5)#descobrir as posicoes dos outliers
no vetor numerico. e a mesma na tabela original
              out.lines=df[positions,]#salvar as linhas correspondentes as
posicoes em um novo arquivo
              resultado.z[[loop]]= out.lines#salvar arquivo em uma lista
              names(resultado.z)[loop]= colnames(df[i])#nomear elemento da
lista com o nome da coluna do arquivo original
              if (csv==TRUE) #salvar em csv se isso foi pedido
                {
                  mudarSubDir(fator, "z")
                  write.table(out.lines, file=paste(colnames(df[i]),"csv",
```

```
sep="."), sep=",")
                  setwd('../..')
                }
            }
          }
    if (fatorSemObs==TRUE) print (paste(fator, "nao tem outliers para metodo
'z'", sep=" "))#se nao teve outliers identificados imprimir essa mensagem
    }
  if (id=="ESD"| id=="ALL") #se for o metodo 'ESD'
    {
      fatorSemObs=TRUE
      loop=0
      for (j in quant)
        {
          loop=loop+1
          #y=as.vector(df[j][-1,])
          y=sapply(df[,i], as.numeric)
          y=y[!is.na(y)] #retirar os NA da tabela
          n = length(y) #comprimento do vetor
          alpha = 0.05
          NUMBER2= min(NUMBER,ceiling(n/2)) #numero maximo de outliers
testado nunca podera ser maior do que 50% dos dados. Por exemplo é pedido
que se teste 20 outleirs, para um subgrupo com n=50 não tem problema na
fórmula do teste. Mas se um dos vários subgrupos tiver só 6 amostras
(acontece com os meus dados), e pede para retirar 20 outleirs, a fórmula
dará erro (NaN's produced). O teste para ser eficaz precisa ter n maior que
15 segundo os autores.
          lam = 1:NUMBER2 #cria objeto para salvar o valor de lam
          R = 1:NUMBER2 #cria objeto para salvaro valor de R
              if(n>=5) #se tiver numero de amostra sufuciente
                {
                    for (i in 1:NUMBER2)#calcular o teste estatistico,
retirei da referência que pus na proposta, só fiz algumas mudancas. Essa
parte do codigo nao e de minha autoria
                        {
                            if(i==1)
                                {
                                  rt = rval(y) #funcao auxiliar de autoria
dos autores do algoritmo
                                  R[i] = unlist(rt[1])
                                  tab = data.frame(rt[2])
                                  newtab = tab[tab$ares!=max(tab$ares),]
                                }
                            else if(i!=1)
                                {
                                  rt = rval(newtab$y)
                                  R[i] = unlist(rt[1])
                                  tab = data.frame(rt[2])
                                  newtab = tab[tab$ares!=max(tab$ares),]
                                }
                            #Computar o valor critico.
```

```
                            p = 1 - alpha/(2*(n-i+1))
                            t = qt(p,(n-i-1))
                            lam[i] = t*(n-i) / sqrt((n-i-1+t**2)*(n-i+1))
                          }
                }
              else #se o numero de amostra for pequeno
                {
                  print(paste(colnames(df[j]),"  do agrupamento ",fator," nao
possui numero de amostras suficientes para realizar as analises (n>4)",
sep=""))#imprimir o aviso
                  next#ir para o proximo 'i'
                }
          newtab = data.frame(c(1:NUMBER2),R,lam) #criar um tabela com o
resultado
          names(newtab)=c("No. Outliers","Test Stat.", "Critical
Val.")#acrescentar os nomes das colunas
          resultado.ESD[[loop]]= newtab#salvar tabela na lista
          names(resultado.ESD)[loop]= colnames(df[j]) #salvar o nome da
coluna da tabela original usada para o calculo
          if (any(newtab[2] > newtab[3], na.rm = TRUE)==TRUE)#se houve algum
valor que ultrapassou o ponto critico
              {
                fatorSemObs=FALSE #mudar a variavel de indicacao de outliers
                numb.out=which(newtab[2] > newtab[3]) #descobrir quais
numeros ultrapassaram
                print(paste(colnames(df[j]),"  do agrupamento ",fator,"
possui ", max(numb.out), " possivel(is) outlier(s)", sep="")) #imprimir a
mensagem informando o numero de possiveis ouliers
                if (csv==TRUE) #salvar em csv se foi requisitado
                    {
                      mudarSubDir(fator, "ESD")
                      write.table(newtab, file=paste(colnames(df[j]),"csv",
sep="."), sep=",")
                      setwd('../..')
                    }
              }
    }
    if (fatorSemObs==TRUE) print (paste(fator, "nao tem outliers para metodo
'ESD'", sep=" ")) #imprimir essa mensagem se nao ha outliers
    }
  return (list(resultado.box, resultado.z, resultado.ESD)) #retornar os
resultados dos tres metodos, se um deles nao for escolhido a lista retornara
como NULL
}

##############################################################################
############

etapa3 = function (df, quant, fator, visual= "boxplot", res="LOW")
  {
```

```
      #funcao para elaboracao dos graficos
      #parametros definidos como padrao, para o 'LOW'
      width = 1000
      height = 480*ceiling(length(quant)/2)
      resol = 150
      if (res=="MED")#parametros para 'MED'
          {
          width = 2000
          height = 960*ceiling(length(quant)/2)
          resol = 300
          }
      if (res=="HIGH")#parametros para 'HIGH'
          {
            width = 4000
            height = 1920*ceiling(length(quant)/2)
            resol = 300
          }
      if (visual=="boxplot" | visual=="ALL") #para graficos boxplot
          {
            mudarSubDir(fator, "boxplot")#funcao auxiliar para mudar o
diretorio antes de salvar o grafico
            jpeg(filename = paste("graph_boxplot_", res, ".jpg", sep=""),
width = width, height = height, units = "px", pointsize = 12, quality = 100,
bg = "white", res = resol, family = "") #abrir o dispositivo e padronizar o
formato jpeg para salvar o grafico
            par(mfrow= c(ceiling(length(quant)/2), 2)) #dividir a janela
grafica
            for (k in quant) #Ciclo para todas as variaveis quantitativas,
identificadas pela posicao, um grafico por variavel
              {
                  if (length(df[,k])>=5)#se houver mais de cinco amostras,
contruir o grafico
                  {
                    graph=boxplot(df[k], data=df, main= colnames(df[k]),
na.omit(df[k]))#desenhar o grafico
                    posi.out= which (df[,k] %in% graph$out)#saber a posicao
dos outliers
                      if(length(posi.out)>0) #se tiver outliers
                          {text(graph$group, graph$out,posi.out, pos = 4,
col= "red")} #identifica-los no grafico com o numero da posicao da linha na
tabela original 'x'
                  }
                  else #se o numero de amostra for pequeno, nao realizar a
analise
                  {print(paste(colnames(df[k])," do agrupamento ",fator,"
nao possui numero de amostras suficientes para realizar as analises (n>4)",
sep=""))#imprimir essa mensagem
                    next#ir para o proximo k
                  }
              }
            par(mfrow= c(1,1))#voltar ao padrão original da janela grafica
```

```
            dev.off() #desligar o dispositivo
            setwd('../..')#voltar para o diretorio original
          }
      if (visual=="pontos" | visual=="ALL") #para graficos dotchart
          {
            mudarSubDir(fator, "pontos")
            jpeg(filename = paste("graph_pontos_", res, ".jpg", sep=""),
width = width, height = height, units = "px", pointsize = 12, quality = 100,
bg = "white", res = resol, family = "") #padronizar o formato jpeg para
salver o gráfico
            par(mfrow= c(ceiling(length(quant)/2), 2)) #dividir a janela
grafica
                for (k in quant) #Ciclo para todas as variaveis
quantitativas, identificadas pela posição
                  {
                      if (length(df[,k])>=5)
                      {
                      dotchart(as.numeric(df[,k]), main= colnames(df[k]),
pch=16)
                      text(df[,k],
1:length(as.vector(df[,k])),rownames(df[k]), pos=4, cex=1, col="red") #nome
das linhas nos graficos
                      }
                      else
                      {print(paste(colnames(df[k])," do agrupamento
",fator," nao possui numero de amostras suficientes para realizar as
analises (n>4)", sep=""))
                      next
                      }
                  }
            par(mfrow= c(1,1))#voltar ao padrão original da janela grafica
            dev.off() #desligar o dispositivo
            setwd('../..') #voltar para diretorio original
          }

      if (visual=="biplot" | visual=="ALL") #para graficos com duas
variaveis
          {
              mudarSubDir(fator, "biplot") #usar funcao auxiliar para mudar
o diretorio antes de salvar os graficos
              jpeg(filename = paste("graph_biplot_", res, ".jpg", sep=""),
width = width, height = height, units = "px", pointsize = 12, quality = 100,
bg = "white", res = resol, family = "") #padronizar o gráfico
              par(mfrow= c(ceiling(length(quant)/2), 2)) #dividir a janela
grafica
              biplot=quant[-length(quant)] #retirar a posicao do ultimo
'quant'
                  for(k in biplot)
                      {
                         if (length(df[,k])>=5) #se tiver amostras
```

```
suficientes
                              {
                                plot(df[,k], df[,k+1], xlab= colnames(df[k]),
ylab= colnames(df[k+1])) #desenhar os graficos
                                text(df[,k], df[,k+1],rownames(df[k]), pos=4,
cex=1, col="red")#colocar a posicao das linhas nos pontos
                              }
                            else #se não tiver amostras suficientes
                              {print(paste(colnames(df[k])," do agrupamento
",fator," nao possui numero de amostras suficientes para realizar as
analises (n>=5)", sep=""))
                              next
                              }
                        }
            par(mfrow= c(1,1))#voltar ao padrão original da janela grafica
            dev.off() #desligar o dispositivo
            setwd('../..') #voltar ao diretorio original
          }
        }

#############################################################################
```

---

# Arquivos da Função

Função
Dados para Input
Help da Função

---