

Daniel Magalhães Lima

Médico Veterinário formado pela FMVZ-USP



Doutorando no Laboratório de Epidemiologia e Bioestatística, Departamento de Medicina Veterinária Preventiva e Saúde Animal da Faculdade de Medicina Veterinária e Zootecnia da USP.

Tem como tema da tese o estudo do impacto econômico de estratégias de controle da brucelose e tuberculose bovinas através de modelagem matemática.

[LEB - Laboratório de Epidemiologia e Bioestatística](#)

Propostas de Trabalho Final

Proposta 1

Sumário Rápido

Através de um simples comando (`sumario(df)`) a função realiza um sumário geral em um banco de dados, identificando variáveis quantitativas e qualitativas e plotando gráficos e tabelas de acordo com o requerimento da variável. O sumário contará com avaliação da normalidade através de gráficos e medidas de simetria e curtose, medidas de tendência e de dispersão, uma combinação de n de gráficos básicos à escolha do usuário (histogramas com curva normal baseada, boxplot, qqplots, cleveand dotplot, correlogramas com intervalos de confiança) e tabelas de contingência.

```
sumario(data.frame = , by = NULL, graphs = c('histograma', 'boxplot', 'qqplot', 'clevelandplot', 'correlograma'), ...)
```

Proposta 2

Medidas de risco

A função pode ter como entrara 3 tipos de objetos: vetores com variáveis qualitativas associadas a ocorrência de um evento, uma tabela de contingencia 2x2 ou um objeto do tipo `lm/glm`. A partir de um objeto `lm` ou `glm` a função seleciona os coeficientes da regressão e calcula as medidas de risco escolhidas pelo usuário, intervalo de confiança das medidas e plot destes intervalos.

```
risco(x = , y = NULL, medidas = c(OR, RR, RT, RA, RP))
```

Comentários Vitor

Daniel, suas propostas precisam melhorar um pouco. Qual o objetivo da primeira proposta? fazer várias análises pra ver no que dá não é uma boa estratégia de análise de dados. Além disso, você implementaria as análises ou só chamaria

funções pré-existentes? O mesmo vale para a proposta B. Se você for implementar os códigos, eu iria com a B, ou tentaria um plano C Vitor Rios

Trabalho Final

Segue aqui o arquivo:

[final_daniel.r](#)

E aqui o código:

```
# Criando a função e definindo os argumentos
exploratoria <- function(df, by = NULL, to.factor = NULL, bin2factor = T){
  # Carregando o pacote que faz o correlograma
  require(corrgram, quietly = T)
  # Carregando o pacote que faz os gráficos
  require(ggplot2, quietly = T)
  #Extraindo os nomes das colunas para futura indexação
  nomes <- colnames(df)
  # Conferindo o argumento bin2factor
  if (bin2factor == T) {
    # Evitando um aviso causado pela comparação de vetores de tamanhos
    # possivelmente diferentes.
    suppressWarnings(
      # Iniciando o for para a coerção de variáveis binárias para
      # fatores
      for(i in 1:ncol(df)){
        # Conferindo valores únicos no vetor
        check.bin <- sort(unique(df[,nomes[i]]))
        # Conferindo quem é binário e ...
        if (all(check.bin == c(0,1)) == T) {
          # ... se for binário mudar para fator.
          df[,i] <- as.factor(df[,nomes[i]])
          # ... se não for...
        } else {
          # ... segue o jogo!
          next()
        }
      }
    )
  }
  # Conferindo se o argumento 'to.factor' está preenchido e...
  if (!is.null(to.factor)){
    # ... se estiver extraímos as posições nas quais as variáveis foram
```

apontadas.

```
# Este objeto servirá como contador do for logo daqui a pouco.
cont.factor <- which(nomes %in% to.factor)
# Abrindo o for para a coerção para fator
for(i in cont.factor){
  # Coercionando as variáveis selecionadas para fator.
  df[,i] <- as.factor(df[,i])
}
}
# Identificando as variáveis do tipo fator para futura indexação
fatores <- colnames(df[sapply(df, is.factor)])
# Identificando as variáveis do tipo numerica para futura indexação
numericas <- colnames(df[sapply(df, is.numeric)])
#####
##### Função Multiplot #####
##### desenvolvida por Winston Chang #####
#####
# disponível em:
#
http://www.cookbook-r.com/Graphs/Multiple\_graphs\_on\_one\_page\_\(ggplot2\)/
# Embora esta função esteja aqui dentro da minha não fui eu quem a
implementou
# me dei a liberdade de não comentá-la linha a linha, pois, além de o
autor
# ter feito uma série de comentários, eu não domino o objetivo de cada
linha
# utilizada aí.
multiplot <- function(..., plotlist=NULL, file, cols=1, layout=NULL) {
  library(grid)
  # Make a list from the ... arguments and plotlist
  plots <- c(list(...), plotlist)
  numPlots = length(plots)
  # If layout is NULL, then use 'cols' to determine layout
  if (is.null(layout)) {
    # Make the panel
    # ncol: Number of columns of plots
    # nrow: Number of rows needed, calculated from # of cols
    layout <- matrix(seq(1, cols * ceiling(numPlots/cols)),
                      ncol = cols, nrow = ceiling(numPlots/cols))
  }
  if (numPlots==1) {
    print(plots[[1]])
  } else {
    # Set up the page
    grid.newpage()
    pushViewport(viewport(layout = grid.layout(nrow(layout),
ncol(layout))))
    # Make each plot, in the correct location
    for (i in 1:numPlots) {
      # Get the i,j matrix positions of the regions that contain
this subplot
```

```
matchidx <- as.data.frame(which(layout == i, arr.ind =
TRUE))
print(plots[[i]], vp = viewport(layout.pos.row =
matchidx$row,
layout.pos.col =
matchidx$col))
}
}
}
#####
##### Função sumario.base #####
##### desenvolvida por mim mesmo #####
#####
# O objetivo de fazer uma função com uma série de avaliações básicas da
# análise descritiva é poder utilizá-la em diferentes cenários, ex:
# utilizando como entrada um simples vetor, ou através de alguma função
da
# família apply ou até mesmo em conjunto com funções da família de
pacotes
# da filosofia do tidyverse (apesar das tentativas de utilizar este
último
# aqui na função não consegui dominar a aplicação da filosofia.)
# Iniciando a função e definindo os argumentos
sumario.base <- function(x){
  # Extraindo o primeiro quartil
  q1 <- quantile(x, probs = 0.25, na.rm = T)[[1]]
  # Extraindo o terceiro quartil
  q3 <- quantile(x, probs = 0.75, na.rm = T)[[1]]
  # Construindo o data.frame de saída da função
  data.frame(
    # Calculando a média
    media = round(mean(x, na.rm = T),2),
    # Calculando a desvio padrão
    desvio = round(sd(x, na.rm = T),2),
    # Verificando o valor mínimo
    minimo = min(x, na.rm = T),
    # alocando o primeiro quartil
    q1 = q1,
    # Calculando a mediana
    mediana = median(x, na.rm = T),
    # Calculando o terceiro quartil
    q3 = q3,
    # Calculando o valor máximo
    maximo = max(x, na.rm = T),
    # Calculando o intervalo inter-quartil
    iiq = (q3 - q1),
    # Contando os NAs
    na.cont = sum(is.na(x)),
    # Relativizando os NAs
    na.percent = sum(is.na(x))/length(x),
```

```

        # Contando os zeros
        zero.cont = sum(x == 0, na.rm = T),
        # Relativizando os zeros
        zero.percent = sum(x == 0, na.rm = T)/length(x)
    )
}
# Criando as listas que compõe a saída da função...
# ... gráficos de variáveis numéricas...
graficos.num <- list()
# ... de variáveis categóricas...
graficos.fac <- list()
# ... e tabelas.
tabelas.fac <- list()
# Não vi nenhum sentido para calcular um correlograma com base em uma
# variável categorica, portanto o gráfico é construído com base na
totalidade
# dos dados e neste momento da função.
# Calculando a matrix de correlação e desenhando o correlograma
corrgram(df[,numericas], type = 'data',
         lower.panel = 'panel.cor', upper.panel = 'panel.pts')
# Salvando o gráfico
correlograma <- recordPlot()

# Construi a função em dois grandes blocos: Sem variáveis categóricas e
com
# variáveis categóricas.
# Abaixo começa a mais simples: sem variável categórica
#####
##### Sem fatores #####
#####

# Saídas Gráficas -----
# Conferindo o argumento
if(is.null(by)){
    # iniciando o for para a construção de gráficos das variáveis
numericas
    for(i in 1:length(numericas)){
        # criando o plot, definindo o df de trabalho e o eixo y
        bp <- ggplot(df, aes_string(y = numericas[i]))+
            #definindo o eixo x
            aes(x = numericas[i])+
            # definindo o tipo do gráfico
            geom_boxplot()+
            # aparando uma legenda repetitiva
            labs(x = NULL)
        # criando o plot, definindo o df e o eixo x
        histo <- ggplot(df, aes_string(x = numericas[i]))+
            # definindo o tipo do grafico
            geom_histogram()
        # criando o plot, definindo o df de trabalho e a origem das
amostras

```

```
qq <- ggplot(df, aes_string(sample = numericas[i]))+
  # definindo o tipo do gráfico
  stat_qq()
# criando o plot, definindo o df de trabalho e o eixo x
clev <- ggplot(df,aes_string( x = numericas[i]))+
  # definindo o eixo y
  aes(y = row.names(df))+
  # definindo o tipo de gráfico
  geom_point()+
  # aparando legendas desnecessárias
  theme(axis.text.y = element_blank())
# plotando todo mundo junto
multiplot(bp, qq, histo, clev, cols = 2)
# salvando o gráfico
plot.num <- recordPlot()
# alocando o gráfico na lista
graficos.num[[i]] <- plot.num
# nomeando o gráfico
names(graficos.num)[i] <- numericas[i]
}
# iniciando a construção dos gráficos das variáveis categóricas e a
# construção das tabelas
for(i in 1:length(fatores)){
  # criando o plot, definindo o df de trabalho e o eixo x
  barras <- ggplot(df, aes_string(x = fatores[i],
                                   fill = fatores[i]))+
    # definindo o tipo de gráfico e selecionando, por default a
    # contagem como eixo y
    geom_bar()
  # Aqui, apesar de plotar apenas um gráfico precisei utilizar o
  # multiplot pois assim foi mais fácil plotar o gráfico de dentro
  # função.
  multiplot(barras)
  # Salvando o gráfico...
  plot.fac <- recordPlot()
  # ... alocando ele na lista e...
  graficos.fac[[i]] <- plot.fac
  # ... nomeando-o.
  names(graficos.fac)[i] <- fatores[i]
  # Tabelaando os dados, guardando a tabela na lista e...
  tabelas.fac[[i]] <- table(df[fatores[i]])
  # ... nomeando a lista.
  names(tabelas.fac)[[i]] <- fatores[i]
}
}
```

da

Saídas Numéricas -----

aplicando a função sumario.base no data.frame e fazendo a transposição

```

# para que fique mais legível. O objeto sumario.num é um dos componentes
# da saída da função.
sumario.num <- as.data.frame(t(sapply(df[,numericas],sumario.base)))
# Por aqui termina o trabalho quando não há variável indexadora e começa
# a segunda parte.
#####
##### Com Fatores #####
#####
# conferindo o argumento by. Se este existir...
if(!is.null(by)){
# Sidas Graficas -----
# ... começa a construção dos gráficos. A partir das variáveis
numéricas.
  for(i in 1:length(numericas)){
    # criando o plot, definindo o df de trabalho, eixo y e
agrupamento
    # com base no argumento by
    bp <- ggplot(df, aes_string(x = by, y = numericas[i]))+
      # definindo o tipo do gráfico
      geom_boxplot()
    # criando o plot, definindo o df de trabalho e o eixo x...
    histo <- ggplot(df, aes_string(x = numericas[i]))+
      # ...definindo o tipo de gráfico e...
      geom_histogram()+
      # ... fazendo o facet com base no 'by'.
      # A disposição foi selecionada para possibilitar a
comparação
      # da melhor maneira possível
      facet_grid(df[,by] ~ .)
    # criando o plot, definindo o df de trabalho, definindo as
amostras...
    qq <- ggplot(df, aes_string(sample = numericas[i]))+
      # ... definindo o tipo de gráfico e...
      stat_qq()+
      # ... fazendo o facet.
      facet_grid(df[,by] ~ .)
    # criando o plot, definindo o df de trabalho, definindo o eixo
x,...
    cleve <- ggplot(df, aes_string( x = numericas[i]))+
      # ... o eixo y...
      aes(y = row.names(df))+
      # ... o tipo de gráfico ...
      geom_point()+
      # retirando marcações indesejáveis e...
      theme(axis.text.y = element_blank())+
      # fazendo o facet.
      facet_grid(df[,by] ~ .)
    # plotando todo mundo junto, ...
    multiplot(bp, qq, histo, cleve, cols = 2)
    # ... salvando o gráfico,...
    plot.num <- recordPlot()
  }
}

```

```
# ... alocando na devida lista e...
graficos.num[[i]] <- plot.num
# nomeando.
names(graficos.num)[i] <- numericas[i]
}
# para que não haja um gráfico 'by' vs 'by' bolei esta estratégia:
# criei este contador para o for que vem a seguir e...
cont.fac <- 1:length(fatores)
# ... tirei do vetor a posição da variável 'by'.
cont.fac <- cont.fac[c(-which(fatores == by))]
# começando o for para a construção dos gráficos das categóricas
for(i in cont.fac){
  # criando o plot, definindo o df de trabalho, eixo x, ...
  barras <- ggplot(df, aes_string(x = fatores[i],
                                   # colorindo e ...
                                   fill = by,
                                   # agrupando por 'by'
                                   group = by))+
    # definindo o tipo de gráfico e a posição das barras.
    geom_bar(position = position_dodge())
  # plotando,...
  multiplot(barras)
  # ... salvando,...
  plot.fac <- recordPlot()
  # ... alocando e...
  graficos.fac[[i]] <- plot.fac
  # ... nomeando o gráfico.
  names(graficos.fac)[i] <- fatores[i]
  # Criando e alocando as tabelas de contingencia e...
  tabelas.fac[[i]] <- table(df[,by], df[,fatores[i]])
  # ... nomeando as tabelas
  names(tabelas.fac)[[i]] <- paste(fatores[i], by, sep = 'VS')
}
#####
##### Saídas Numéricas #####
#####
# começando a criação do df que levará o sumário numérico.
sumario.num <- NULL
# Abrindo um for para a confecção do sumario
for(i in 1:length(numericas)){
  # aplicando a função sumario.base através da indexação por 'by'
  # em um tapply. O objeto temporário é uma lista...
  tmp <- tapply(df[, numericas[i]], df[,by], sumario.base)
  # ... que é destrinchada em uma matriz e alocada em um df.
  tmp2 <- data.frame(matrix(unlist(tmp), nrow = length(tmp),
byrow=T))

  # O df é atualizado a cada iteração do for.
  sumario.num <- rbind(sumario.num, tmp2)
}
# Começa a arrumação do df...
```

```
# ... primeiro nomeando as colunas,...
colnames(sumario.num) <- colnames(tmp[[1]])
# ... depois inserindo os nomes das variáveis,...
sumario.num$variavel <- rep(numericas, each =
length(levels(df[,by])))
# ... e então os grupos às quais a linha pertence.
sumario.num$grupo <- rep(levels(df[,by]), times = length(numericas))
# Finalizando com a facilitação da visualização.
sumario.num <- sumario.num[,c(13:14, 1:12)]
}
#####
##### Finalizando #####
#####
# Preparando a saída da função
# Construindo a lista com os objetos que compõe a saída e...
saida <- list(sumario.num, graficos.num, graficos.fac, tabelas.fac,
correlograma)
# nomeando-os.
names(saida) <- c('sumario.num','graficos.num','graficos.fac',
'tabelas',
                'correlograma')
# Apresentando os resultados e fechando a função!
return(saida)
}
```

Help da função

Segue aqui o arquivo:

[help_exploratoria.txt](#)

E aqui o texto.

exploratoria()

package:BIE5782

R Documentation

Faz um análise exploratória básica.

Description:

A função recebe um banco de dados e avalia, automaticamente e através dos argumentos, quais são as variáveis categóricas e quais são as numéricas. Com base nesta relação a função plota diversos gráficos, tabelas de contingência e sumários estatísticos.

Usage:

```
exploratoria(df, by, to.factor, bin2factor)
```

```
# default:  
# exploratoria(df, by = NULL, to.factor = NULL, bin2factor = T)
```

Arguments:

df Um data.frame.
by Uma variável indexadora do seu data.frame. Passada entre "".
to.factor Um vetor contendo os nomes das variáveis as quais o usuário deseja coercionar para fator dentro da função.
bin2factor Argumento lógico que variáveis apresentadas como binárias sejam interpretadas como fator.

Details:

Value:

sumario.num Um data.frame que é composto de medidas de tendencia central, medidas de dispersão, contagem de zeros e de NAs para cada variável e suas possíveis categorias.
graficos.num Lista de gráficos das variáveis classificadas como numéricas.
graficos.fac Lista de gráficos das variáveis classificadas como categóricas
tabelas Lista que contém as tabelas com as distribuições das variáveis categóricas
correlograma Correlograma com coeficiente de correlação das variáveis numericas.

Warning:

Note:

Author(s):

Daniel Magalhães Lima

References:

função multiplot() retirada deste livro:
[http://www.cookbook-r.com/Graphs/Multiple_graphs_on_one_page_\(ggplot2\)/](http://www.cookbook-r.com/Graphs/Multiple_graphs_on_one_page_(ggplot2)/)

See Also:

Examples:

```
set.seed(5782)
```

```
banco <- data.frame(  
  grupo = sample(c('a', 'b'), 100, replace = T),  
  medida1 = rnorm(100, 10, 2),  
  medida2 = sample(c(0:5, NA), 100, replace = T),  
  sexo = sample(0:1, 100, replace = T),  
  repeticao = sample(1:3, 100, replace = T)  
)
```

```
# 0 basico:
```

```
teste <- exploratoria(df = banco)
```

```
View(teste$sumario.num)
```

```
teste$tabelas
```

```
# Usando os argumentos extras:
```

```
teste2 <- exploratoria(df = banco, by = 'grupo', to.factor =  
'repeticao')
```

```
View(teste2$sumario.num)
```

```
teste2$tabelas
```

From:

<http://ecor.ib.usp.br/> - **ecoR**

Permanent link:

http://ecor.ib.usp.br/doku.php?id=05_curso_antigo:r2017:alunos:trabalho_final:daniel.magalhaes.lima:start



Last update: **2020/08/12 06:04**