

Trabalho Final

Código

```
lmanal = function(dataf,var.res = colnames(dataf)[1]){
  require("MuMIn") #requer o pacote MuMIn que contem a funcao AICc
#####PREMISSAS#####
#
  if(!is.data.frame(dataf)){ #verifica se dataf e um data.frame
    stop(dataf,"\t nao e data frame") #para o scrip se dataf nao for
data.frame
  }
#####ORGANIZANDO OS
DADOS#####
  s = dim(dataf) #tamanho do data.frame
  #reorganiza o data frame com var.res na primeira coluna
  if(var.res!=colnames(dataf)[1]){ #verifica se a primeira coluna e a
variavel resposta
    if(is.character(var.res)){#verifica se a variavel resposta e string
      teste_char=rep(NA,s[2])#cria um vetor para verificar se var.res e um
nome valido
      for(ii in 1:s[2]){#comeca o loop para teste de cada coluna
        teste_char[ii]=identical(var.res,colnames(dataf)[ii])# teste
      }
      if(sum(teste_char)==1){#teste se alguma coluna contem var.res
        flag=which(colnames(dataf)==var.res) #posicao da variavel resposta
      }
      else{#caso contrario para a funcao
        stop("Escolha um nome ou numero de coluna valido")
      }
    }
    else if(is.numeric(var.res)){ #se var.res for o numero da coluna
      if(var.res<=s[2] && var.res>0){#caso a var.res seja um numero valido
        flag=var.res#retira aposicao de ver.res
        var.res=colnames(dataf)[flag] #atribui a var.res o nome da variavel
resposta
      }
      else{#caso contrario para a funcao
        stop("Escolha um nome ou numero de coluna valido")
      }
    }
    else{#caso var.res nao seja numero nem caractere
      stop("var.res deve ser o nome ou numero da coluna da variavel
resposta")
    }
    if(flag==s[2]){#caso var.res seja a ultima coluna
      dataf = data.frame(dataf[flag],dataf[1:(flag-1)])#organiza o data
frame com var.res na primeira coluna
```

```
}else{#caso seja em qualquer outra coluna
  dataf =
data.frame(dataf[flag],dataf[1:(flag-1)],dataf[(flag+1):s[2]])#organiza o
data frame com var.res na primeira coluna
}
}
if(class(dataf[,1])=="factor" || class(dataf[,1])=="logical"){# se var.res
for fator ou logico lm() nao funciona
  stop("A variavel resposta nao pode ser logica ou fator")#para a funcao
}
#####RETIRANDO
NAS#####
if(sum(is.na.data.frame(dataf))!=0){ #retira os dados NA
  linhas = unique(c(which(is.na.data.frame(dataf),arr.ind = T)[,1]))#cria
um vetor contendo as linhas com NA
  linhas_str = paste0(paste0(linhas[1:length(linhas)-1], ", ", collapse =
""),#transforma o vetor em string
                      linhas[length(linhas)],".",collapse = "")# com
virgulas e ponto final
  message(as.character(sum(is.na.data.frame(dataf))), #manda a mensagem ao
usuario dizendo as linhas
          " linhas contendo NA retiradas: ", #que serao retiradas
          linhas_str)
  dataf = na.omit(dataf) #retira os na's
}
#####PREPARANDO O MODELO
COMPLETO#####
var.nomes = colnames(dataf) #nomes das variaveis
#cria o string que sera avaliado
var.nomes.list=as.list(rep(NA,s[2]-1))#cria uma lista vazia para receber
os nomes das colunas
for(iii in 1:(s[2]-1)){#loop para preencher a lista com as combinações
necessarias para testar as possibilidades
  if(iii!=s[2]-1){#caso nao seja a ultima coluna
    var.nomes.list[[iii]]=c(var.nomes[(iii+1):s[2]])#nao e a ultima
coluna
  }
  else{
    var.nomes.list[[iii]]=c(var.nomes[s[2]])#ultima coluna
  }
}
termos_char = data.frame(do.call(expand.grid,var.nomes.list))#cria a
lista com as combinacoes possiveis com as variaveis
termos_char = t(termos_char) #transpoe o data frame
ndim=dim(termos_char)[2] #retira o numero de colunas
mod.vars = as.list(rep(NA,ndim)) #listas vazias que serao preenchidas
com as
mod.formula=mod.vars #variaveis que vao compor a formula e as formulas
em caracter
for(I in 1:ndim){#loop que completa as listas
```

```

    mod.vars[[I]]=sort(unique(c(termos_char[,I])))#retira as variaveis
redundantes
    mdim=length(mod.vars[[I]])#retira o numero de variaveis preditoras nos
modelos
    if(mdim==1){#caso seja um modelo com apenas uma variavel preditora
        mod.formula[[I]]=paste0(var.res,"~",mod.vars[[I]])#cria a formula
    }
    else if(mdim==2){#caso seja um modelo com duas variaveis preditoras
mod.formula[[I]]=paste0(var.res,"~",mod.vars[[I]][1],"+",mod.vars[[I]][2])#c
ria a formula
    }
    else{#caso seja um modelo com mais de duas variaveis preditoras
        mod.formula[[I]]=paste0(var.res,"~",#cria a formula
                                paste0(mod.vars[[I]][1],
                                paste0("+",mod.vars[[I]][2:mdim],
collapse=""),collapse=""),collapse="")
    }
}
mod.formula=unique(mod.formula)#retira as formulas de modelos
redundantes
odim=length(mod.formula)#retira o numero de modelos a serem testados
aicc.vec = rep(NA,odim)#vetor que sera preenchido com os valores de AICc
for(II in 1:odim){#loop que preenche o vetor
    aicc.vec[II]=AICc(lm(mod.formula[[II]],data=dataf))#calcula do modelo
e respectivo AICc
}
best.fit=lm(mod.formula[[which.min(aicc.vec)]],data=dataf)#carrega o
melhor modelo em best.fit
par(mfrow=c(2,2))#divide o ambiente grafico em 4
plot(best.fit)#plota o melhor modelo
print(summary(best.fit))#entrega as propriedades do melhor modelo
invisible(best.fit)#retorna o modelo em uma variavel caso o usuario
deseje
}

```

Help

lmanal()

R Documentation

Description:

Function that calculates the best linear model from all combinations of the variables from a data frame using the Second-order Akaike Information Criterion (AICc)

Usage:

```
lmanal = function(dataf,var.res = colnames(dataf)[1])
```

Arguments:

`dataf` a data frame class object containing the variables on which the linear model will be constructed

`var.res` character name or numeric value of the column of the data frame which the model will predict

Details:

`var.res` has to be a continuous value

Value:

if attributed to a variable returns a `lm` class object containing the linear model with the lower AICc value

prints the summary of the `lm` object

plots the `lm` object in a 2 by 2 graphic environment

Warning:

column names must not have any spaces or odd characters

Note:

uses the package "MuMIn" to load the AICc function

Author(s):

Luan Sayeg Michelazzo

References:

Burnham KP, Anderson DR. Practical use of the information-theoretic approach. In Model Selection and Inference 1998 (pp. 75-117). Springer, New York, NY

Burnham KP, Anderson DR, Huyvaert KP. AIC model selection and multimodel inference in behavioral ecology: some background, observations, and comparisons. Behavioral ecology and sociobiology. 2011 Jan 1;65(1):23-35.

Wagenmakers EJ, Farrell S. AIC model selection using Akaike weights. Psychonomic bulletin & review. 2004 Feb 1;11(1):192-6.

See Also:

`lm()`, "MuMIn" package

Examples:

```
l = lmanal(attitude)
```

```
lmanal(attitude,"complaints")  
lmanal(attitude,2)
```

From:

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

Permanent link:

http://ecor.ib.usp.br/doku.php?id=05_curso_antigo:r2019:alunos:trabalho_final:luan.michelazzo:final 

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