Código B

```
#########################
# Pensa o que eu penso? #
#########################


##### Código da função #####

twit= function(dist=list(...), con=list(...), coord=TRUE, main=c(NULL),
label=c("Indivíduo"), slabel=c("Distância"), lcol=c("white"), hcol=
c("red"), tcol=c("black"), ncols=1) # criando o nome da função, os
argumentos e seus respectivos defaults
{
  if ((length(grep("ggplot2", library()))==0)=="TRUE") # teste lógico que
procura nos pacotes instalados o pacote "ggplot2" que será necessário para
rodar a função
  {
    stop("Para essa função é necessário instalar o pacote 'ggplot2' (use a
função install.packages()). É aconselhável antes de tentar rodar a função
novamente, reiniciar o programa do R.") # caso o pacote não seja encontrado
uma mensagem de aviso é lançada no console
  }
  if ((length(grep("reshape2", library()))==0)=="TRUE") # teste lógico que
procura nos pacotes instalados o pacote "reshape2" que será necessário para
rodar a função
  {
    stop("Para essa função é necessário instalar o pacote 'reshape2' (use a
função install.packages()). É aconselhável antes de tentar rodar a função
novamente, reiniciar o programa do R.") # caso o pacote não seja encontrado
uma mensagem de aviso é lançada no console
  }
  library(ggplot2)
  library(reshape2)
  if (class(dist) != "list" || class(con) != "list") # teste lógico para ver
se os objetos 'dist' e 'con' são listas
  {
    stop("Os objetos dos argumentos 'dist' e 'con' devem ser listas") #
mensagem mostrada caso o teste dê verdadeiro
  }
  if (length(dist) != length(con)) # teste lógico para verificar se 'dist' e
'con' tem o mesmo tamanho
  {
    stop("O número de matrizes e/ou dataframes de 'dist' não é igual ao
número de vetores de 'con'") # mensagem mostrada caso o teste dê verdadeiro
  }
#############################################################################
  # a função multiplot já existe, para referência ver:
```

```
#http://www.cookbook-r.com/Graphs/Multiple_graphs_on_one_page_%28ggplot2%29/
  #http://www.peterhaschke.com/r/2013/04/24/MultiPlot.html
  multiplot= function(..., plotlist=NULL, file, cols=ncols, layout=NULL)
  {
    library(grid)
    plots <- c(list(...), plotlist)
    numPlots = length(plots)
    if (is.null(layout))
    {
      layout <- matrix(seq(1, cols * ceiling(numPlots/cols)),
                       ncol = cols, nrow = ceiling(numPlots/cols))
    }
    if (numPlots==1)
    {
      print(plots[[1]])
    }
    else
    {
      grid.newpage()
      pushViewport(viewport(layout = grid.layout(nrow(layout),
ncol(layout))))
      for (i in 1:numPlots)
      {
        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))
      }
    }
  }
  ##########################################################################
  ciclos = length(dist) # cria um objeto com o número de ciclos (i.e plots
gerados a partir das listas fornecidas)
  ##########################################################################
  # as próximas linhas definem alguns argumentos para os plots dos heatmaps,
como por exemplo, os nomes dos eixos, cores, etc... , caso não seja
fornecida uma lista para os respectivos argumentos, ou seja fornecido apenas
um caracter.
  argumentos= list(main, label, slabel, lcol, hcol, tcol) # cria uma lista
com os argumentos que definem alguns parâmetros gráficos dos plots dos heat
maps
  for(j in 1:6) # ciclo que verifica o comprimento de cada argumento da
lista 'argumentos'
  {
    if (length(argumentos[[j]]) == 1) # teste lógico para verificar se o
tamanho do argumento na posição [[j]] é igual a 1, se for:
    {
      argumentos[[j]]= rep(argumentos[[j]], ciclos) # sobrescreve o
argumento da posição [[j]] da lista com uma repetição (número = ciclos) do
caracter original daquele argumento
    }
```

```
  }
  # sobrescreve os argumentos com as repetições estipuladas acima
  main= argumentos[[1]] # para o argumento 'main'
  label= argumentos[[2]] #para o argumento 'label'
  slabel= argumentos[[3]] # ...
  lcol= argumentos[[4]]
  hcol= argumentos[[5]]
  tcol= argumentos[[6]]
  all.plots= list() # cria um uma lista onde serão guardados os plots (heat
maps) de cada ciclo do for[k]
  for (k in 1:ciclos) # inicia o ciclo para plotar os diferentes heat maps
que medem as distâncias entre os indivíduos e os conceitos avaliados
    local( # cada ciclo será realizado num ambiente novo, de forma que as
informações dos plots não se sobreescrevam
      {
        k=k # cria um objeto com o valor do ciclo em questão
        individuos = length(con[[k]]) # cria um objeto com o número de
conceitos avaliados, a partir da posição [[k]] da lista 'con' (ou seja, o
número de individuos que foram avaliados)
        if (coord==TRUE) # teste para o argumento 'coord', caso seja TRUE os
dados são de coordenadas geográficas dos entrevistados
        {
          mat.coord= matrix(unlist(dist[k]), length(dist[[k]][,1]), 2) #
recria uma matriz a partir da seleção da posição [k] da lista "dist"
          mat.dist= as.matrix(dist(mat.coord, diag=T, upper=T)) #cria uma
matriz de distância entre os indivíduos, incluíndo a diagonal e o triângulo
superior
          diag(mat.dist)=NA # garante que a diagonal dessa matriz seja de
NAs
        }
        if (coord==FALSE) # teste para o argumento 'coord', caso seja FALSE
os dados já se referem a alguma medida de distância entre os entrevistados
        {
          mat.dist= as.matrix(unlist(dist[[k]]))#, individuos, individuos) #
recria uma matriz a partir da seleção da posição [k] da lista "dist"
          diag(mat.dist)=NA # garante que a diagonal dessa matriz seja de
NAs
        }
        mat.con= matrix(NA, individuos, individuos) # cria uma matriz de NAs
com o número de linhas e colunas iqual ao que seria o número dos indivíduos
entrevistados
        for(l in 1:length(con[[k]])) # inicia um ciclo para preencher os
conceitos da matriz 'mat.con' para as respectivas posições da lista 'con'
        {
          mat.con[l,]= con[[k]][l] # toda a linha daquele ciclo [l] é
preenchida com o seu respectivo conceito na posição [l] do conjunto [[k]] da
lista 'con'
          mat.con[l, l]= NA # as posições da diagonal são preenchidas com
NA, uma vez que não interessa comparar o conceito de uma pessoa com ela
mesma
        }
```

```
        mat.dist.melt= melt(mat.dist) # cria um objeto com a matriz
'mat.dist' convertida em um dataframe que será necessário para o plotar as
cores do heat map
        mat.con.melt= melt(t(mat.con)) # cria um objeto com a matriz
'mat.con' convertida em um dataframe que será necessário para o plotar os
valores do heat map
        all.plots[[k]] <<- # atribui o plot abaixo para a posição [[k]] da
lista 'all.plot'
          ggplot(mat.dist.melt, aes(Var1, Var2)) + # cria uma área de
plotagem dividida em células que representam as combinações dos diferentes
indivíduos avaliados
          scale_x_discrete(limits=c(1:individuos)) + # plota os valores no
número de indivíduos no eixo x
          scale_y_reverse(breaks=seq(1, individuos, 1)) + # inverte o eixo y
para que o plot fique mais parecido com o modelo usual de plotar matrizes de
correlação e coloca os valores no número de indivíduos
          geom_tile(aes(fill = mat.dist.melt$value), color="black") + #
preenche a cor de fundo das células de acordo com valores de distância entre
os indivíduos e traça linhas pretas ao redor de cada célula
          scale_fill_gradient(slabel[k], low = lcol[k], high = hcol[k]) + #
muda as cores de fundo para uma escala de cores específica indicada pelos
argumentos 'lcol' e 'hcol' e muda o nome da legenda de cores para 'slabel',
sempre usando a posição [k] dos argumentos
          geom_text(label = mat.con.melt$value, na.rm= TRUE, color= tcol[k])
+ # acrescenta os valores dos conceitos de cada indivíduo na cor
especificada no argumento 'tcol'
          labs(title= main[k], x = label[k], y= label[k]) + # acrescenta o
nome do título e dos eixos, paras as respectivas posições [k]
          theme_classic() # retira o cinza de fundo da área de plotagem
      }
    )
  multiplot(plotlist = all.plots) # usa a função 'multiplot' para plotar
todos os heat maps de uma vez
  cat("Esta função engloba uma outra função chamada 'multiplot', para
referência ver
http://www.cookbook-r.com/Graphs/Multiple_graphs_on_one_page_%28ggplot2%29/
ou http://www.peterhaschke.com/r/2013/04/24/MultiPlot.html") # apresenta uma
mensagem no console sobre o uso da função 'multiplot'
} # ufa fim.....
```