visualR

```
### Nota: após a finalização do código, o código foi reformatado utilizando
a função "reformat code" do RStudio.
### Por conta disso, algumas linhas não estão comentadas, mas cada código
usado está devidamente comentado.
## Cria a função visualR.
visualR <-
  # Estabelece os parâmetros iniciais para a função.
  function(text = NULL, # vetor opcional com texto que será inserido no
diagrama. Por padrão é NULL.
           color = "multi", # Estabelece a cor do diagrama. Por padrão é
"multi"
           line = 2, # Estabelece o tipo de linha do diagrama. Por padrão é
2 (seta começo->fim)
           words = FALSE, # Opcional, estabelece a quantidade de palavras
que o usuário irá escrever.
           arrows = FALSE, # Opcional, estabelece a quantidade de
setas/linhas que o usuário irá desenhar.
           rectangle = TRUE, # Estabelece se um retângulo será desenhado ou
não ao redor das palavras.
           font = "sans", # Estabelece o tipo de fonte do diagrama
           arrow.col = NULL){  #Opcional, estabelece uma cor única para a
seta/linha
    ## Verificação das condições
    # Vê se as condições para a função foram satisfeitas
    if (is.null(text) == FALSE) { # Se texto não é nulo
      if (is.character(text) == FALSE) { #Caso o vetor text não seja
character
        t.format <- class(text) #Grava a classe atual de text
        text <- as.character(text) #Converte text em character
        cat(paste( #Informa ao usuário a conversão de text em character
(informando o tipo anterior)
          c(
            "TEXT convertido de",
            t.format,
            "para classe character. \n",
            "Resultado da conversão: \n",
            text,
            "\n"
          ),
          sep = " "
        ))
      }
      if (length(text) < 2) { #Verifica se o comprimento de text é menor que
2, e caso sim, para a função.
        stop("TEXT deve ser um vetor de tamanho maior ou igual a 2. \n")
#Para e retorna mensagem de erro
      }
```

```
      if (max(nchar(text)) > 10) { #Verifica se alguma palavra do vetor text
tem mais de 10 caracteres
        stop(
          "Número de caracteres excedido em TEXT. O número máximo de
caracteres por palavra é 10. \n" #Para e retorna mensagem de erro.
        )
      }
      if (is.numeric(words)) { #Verifica se WORDS foi definido pelo usuário
na presença do vetor text
        cat( #Apenas comunica ao usuário que WORDS será ignorado nesse caso
          "WORDS não é um argumento quando TEXT está presente. Número de
valores em TEXT será usado. \n"
        )
      }
    }
    if (is.numeric(words) == FALSE & is.null(text)) { #Verifica se words e
text estão ausentes.
      #Nesse caso o usuário terá um limite de 200 palavras e poderé parar
quando quiser.
      cat(#Apenas comunica ao usuário esse fato.
        "WORDS e TEXT não especificados. \n O usuário terá um limite de 200
palavras, podendo parar quando quiser. \n"
      )
    }
    if (is.numeric(words) & is.null(text)) { #Verifica se words está
presente e text ausente.
      if (words < 2) { # Se sim, verifica se words é um número maior ou
igual a 2
        stop("WORDS deve ser um valor maior que 2 \n") # Caso words <2, para
a função
      }
      if (words != as.integer(words)) { # Verifica se words é um número
inteiro
        stop("WORDS deve ser um número inteiro \n") # Caso não seja, para a
função
      }
    }
    if (is.numeric(arrows)) { # Verifica se arrows está presente
      if (arrows <= 0) { # Se sim, verifica se arrows é um número maior que
0
        stop("ARROWS deve ser um valor maior que 0 \n") # Caso arrows <0,
para a função
      }
      if (arrows != as.integer(arrows)) { # Verifica se arrows é um número
inteiro
        stop("ARROWS deve ser um número inteiro \n") # Caso não seja, para a
função
      }
    }
    ## Criação do espaço gráfico
```

```
    # Estabelece as configurações para o espaço gráfico
    par(
      mar = c(1, 1, 1, 1), # Estabelece margens mínimas
      family = font # Estabelece a família de fontes do diagrama a partir do
objeto font
    )
    # Plota o gráfico vazio, para criar o espaço onde o diagrama será
desenhado
    plot(
      0, # x0,y0
      50, # x1,y1
      ann = FALSE, # Remove anotações
      type = "n", # Tipo de gráfico "vazio"
      bty = "n", # Remove o quadrado em volta do gráfico
      plt = c(0, 1, 0, 1), # Coordenadas da região de plotagem como fração
da região de figura atual.
      xaxt = "n", # Remove o eixo X
      yaxt = "n" # Remove o eixo Y
    )
    ## Estabelece a quantidade de palavras para os diferentes casos (cria o
objeto qt com essa quantidade)
    if (words) { #Caso words tenha sido estabelecido define qt como a
quantidade de palavras que o usuário escolheu.
      qt <- words
    }
    if (is.character(text)) {#caso o vetor text esteja presente, usa o
comprimento e text como quantidade de cliques (cada palavra será um clique)
      qt <- length(text)
    }
    if (is.character(text) == FALSE & isFALSE(words)) {# Caso nem words nem
text tenham sido estabelecidos, dá um limite de 200 palavras para o usuário.
      qt <- 200
    }
    ## Criação de cores aleatórias
    # Para a criação de cores aleatórias, a quantidade de palavras é usada
como parâmetro definindo quantas cores serão criadas.
    # Assim, caso arrows tenha sido estabelecido e seja maior que a
quantidade de palavras será necessário gerar cores para as setas adicionais.
    if (arrows > qt) {# caso arrows seja maior que qt
      qt.col <- arrows # Cria o objeto qt.col com o valor de arrows
    }
    else { #Se arrows não é maior que qt
      qt.col <- qt #Cria o objeto qt.col com o mesmo valor de qt
    }
    if (color == "multi") { #Cria as cores para a opção "multi" (cores
aleatórias)
      random.c <- TRUE #Cria o objeto random.c com o valor TRUE (será usado
para identificar o uso de multi-cores)
      colors <- #Cria o objeto colors
        data.frame( #Um dataframe com três colunas (r,g,b), cada uma com uma
repetição de NAs pelo valor de qt.col
```

```
          r = rep(NA, qt.col),
          g = rep(NA, qt.col),
          b = rep(NA, qt.col)
        )
      for (i in 1:qt.col) { #Cria um ciclo de 1 a qt.col
        random <- seq(from = 0.3, #Cria o objeto random com uma sequência de
0.3 (para evitar cores claras) a 1
                      to = 1,
                      length.out = qt.col) #O comprimento dessa sequência
será o valor de qt.col
        colors[i, 1] <- sample(random, 1) #Pega um valor aleatório do objeto
random e coloca na linha i da coluna 1 de colors
        colors[i, 2] <- sample(random, 1) #Pega um valor aleatório do objeto
random e coloca na linha i da coluna 2 de colors
        colors[i, 3] <- sample(random, 1) #Pega um valor aleatório do objeto
random e coloca na linha i da coluna 3 de colors
      }
    }
    if (color == "grayscale") { #Cria as cores para a opção "grayscale"
(cores aleatórias)
      random.c <- TRUE #Cria o objeto random.c com o valor TRUE (será usado
para identificar o uso de multi-cores)
      colors <- #Cria o objeto colors
        data.frame( #Um dataframe com três colunas (r,g,b), cada uma com uma
repetição de NAs pelo valor de qt.col
          r = rep(NA, qt.col),
          g = rep(NA, qt.col),
          b = rep(NA, qt.col)
        )
      for (i in 1:qt.col) { #Cria um ciclo de 1 a qt.col
        random <- seq(from = 0.1,  #Cria o objeto random com uma sequência
de 0.1 a 0.7 (para evitar cores muito escuras)
                      to = 0.7,
                      length.out = qt.col) #O comprimento dessa sequência
será o valor de qt.col
        samp <- sample(random, 1) #Cria o objeto samp com um valor aleatório
de random
        colors[i, 1] <- samp #Coloca o valor de samp na linha i da coluna 1
de colors
        colors[i, 2] <- samp #Coloca o valor de samp na linha i da coluna 1
de colors
        colors[i, 3] <- samp #Coloca o valor de samp na linha i da coluna 1
de colors
      }
    }
    if (color != "multi" & color != "grayscale") { #Caso color não seja do
tipo "multi" ou "grey.scale", será uma cor fixa estabelecida pelo usuário
      random.c <- FALSE #cria o objeto random.c com o valor FALSE
      colors <- #Cria o objeto colors com o valor do objeto color
        color
```

```
    }
    if (is.null(arrow.col) == FALSE) { # Verifica se foi estabelecida uma
cor única para as setas/linhas
      arrow.c <- TRUE #Se sim, cria o objeto arrow.c com o valor TRUE (será
usado como marcador)
    }
    else{
      arrow.c <- FALSE #Se não, cria o objeto arrow.c com o valor FALSE
    }
    ## Inicia a inserção dos objetos no diagrama
    for (i in 1:qt) { # Cria um ciclo de 1 a qt (quantidade de palavras)
      cat("Clique onde você deseja o texto \n") # Imprime a mensagem para o
usuário
      point <- locator(1) # Cria o objeto point com o valor de locator (1
clique)
      if (random.c) { # Caso random.c seja TRUE
        col <- rgb(colors$r[i], colors$g[i], colors$b[i], 1) # Cor do
retângulo/linha será rgb pegando a posição i de cada coluna. Alpha 1.
      }
      else { # Caso random.c não seja TRUE
        col <- colors # Cor do retângulo/linha será o valor de colors (fixo)
      }
      if (is.null(text) == FALSE) { # Caso o vetor text esteja presente
        text.w <- strwidth(text[i]) # Grava no objeto text.w o comprimento
da palavra na posição i do vetor text
        text.h <- strheight(text[i]) # Grava no objeto text.h a altura da
palavra na posição i do vetor text
      }
      if (is.numeric(words) & is.null(text)) { # Se words foi estabelecido e
text não está presente
        text.r <- readline("Escreva seu texto (max 10 caracteres) \t") #
Cria o objeto text.r com o texto inserido pelo usuário
        if (nchar(text.r) > 10) { # Verifica se o texto inserido pelo
usuário tem mais de 10 caracteres
          text.r <- # Caso tenha mais de 10 caracteres, dá uma nova chance
do usuário escrever o texto.
            readline("Número de caracteres excedido. Escreva um novo texto
(max 10 caracteres) \t")
          if (nchar(text.r) > 10) { # Caso tenha mais de 10 caracteres
novamente
            stop("Número de caracteres excedido") # Para a função e imprime
a mensagem
          }
        }
        text.w <- strwidth(text.r) # Grava no objeto text.w o comprimento da
palavra de text.r
        text.h <- strheight(text.r) # Grava no objeto text.h a altura da
palavra de text.r
      }
      if (is.numeric(words) == FALSE & is.null(text)) { #Se words e text
estão ausentes
```

```
        text.r <- # Cria o objeto text.r com o texto inserido pelo autor
          #Dá a opção do usuário escrever <END> para finalizar a etapa de
digitação de texto
          readline("Escreva seu texto (max 10 caracteres) ou <END> para
finalizar as palavras \n")
        if (nchar(text.r) > 10) { # Verifica se o texto inserido pelo
usuário tem mais de 10 caracteres
          text <- # Caso tenha mais de 10 caracteres, dá uma nova chance do
usuário escrever o texto.
            readline("Número de carcteres excedido. Escreva um novo texto
(max 10 caracteres) \n")
          if (nchar(text.r) > 10) {  # Caso tenha mais de 10 caracteres
novamente
            stop("Número de caracteres excedido") # Para a função e imprime
a mensagem
          }
        }
        text.w <- strwidth(text.r) # Grava no objeto text.w o comprimento da
palavra de text.r
        text.h <- strheight(text.r) # Grava no objeto text.h a altura da
palavra de text.r
        if (text.r == "<END>") { # Se o usuário digitou <END>
          ## Desenho das setas, caso o usuário tenha digitado <END>
          for (i in 1:qt) { # Inicia um ciclo de 1 a qt
            if (random.c) { # Caso random.c seja TRUE
              col <- rgb(colors$r[i], colors$g[i], colors$b[i], 1) # Cor do
retângulo/linha será rgb pegando a posição i de cada coluna. Alpha 1.
            }
            else { # Caso random.c não seja TRUE
              col <- colors # Cor do retângulo/linha será o valor de colors
(fixo)
            }
            if (arrow.c) { #Se arrow.c for TRUE
              col <- arrow.col #Substitui o valor de col pelo valor de
arrow.col (inserido pelo usuário)
            }
            cat("Clique no ponto de início e término da seta/linha \n") #
Imprime a mensagem para o usuário desenhar a linha
            ## Setas
            if (line <= 3) { # Se a linha for do tipo 1 a 3
              point <- locator(2) # Cria o objeto point com os valores de
locator (dois cliques)
              arrows( # Cria uma seta
                point$x[1], #Posição x1 da seta, com o valor da posição 1 de
x de point
                point$y[1], #Posição y1 da seta, com o valor da posição 1 de
y de point
                point$x[2], #Posição x2 da seta, com o valor da posição 2 de
x de point
                point$y[2], #Posição y2 da seta, com o valor da posição 2 de
```

```
y de point
                length = 0.1, # Define o comprimento das linhas da ponta da
seta
                angle = 30, # Define o ângulo da seta
                code = line, # Define o tipo de seta (1,2 ou 3 - ver help de
arrows para mais informações)
                col = col, # Define a cor da seta com o valor de col
                lwd = 2 # Define a espessura da linha da seta
              )
            }
            ## Linha com pontos na ponta
            if (line == 4) { # Se a linha for do tipo 4
              point <- locator(2) # Cria o objeto point com os valores de
locator (dois cliques)
              segments(point$x[1], # Posição x1 da seta, com o valor da
posição 1 de x de point
                       point$y[1], # Posição y1 da seta, com o valor da
posição 1 de y de point
                       point$x[2], # Posição x2 da seta, com o valor da
posição 2 de x de point
                       point$y[2], # Posição y2 da seta, com o valor da
posição 2 de y de point
                       col = col, # Define a cor da linha com o valor de col
                       lwd = 2) # Define a espessura da linha
              points(point$x[1], # Cria um ponto na posição 1 de x de point
                     point$y[1], # e posição 1 de y de point
                     pch = 19, # do tipo 19
                     col = col) # e cor com o valor de col
              points(point$x[2], # Faz o mesmo usando o valor da posição 2
de x de point
                     point$y[2], # e posição 2 de y de point
                     pch = 19,
                     col = col)
            }
            ## Linha simples
            if (line == 5) { # Se a linha for do tipo 5
              point <- locator(2) # Cria o objeto point com os valores de
locator (dois cliques)
              segments(point$x[1], # Posição x1 da seta, com o valor da
posição 1 de x de point
                       point$y[1], # Posição y1 da seta, com o valor da
posição 1 de y de point
                       point$x[2], # Posição x2 da seta, com o valor da
posição 2 de x de point
                       point$y[2], # Posição y2 da seta, com o valor da
posição 2 de y de point
                       col = col, # Define a cor da linha com o valor de col
                       lwd = 2) # Define a espessura da linha
            }
            finish <- # Cria o objeto finish com o texto escrito pelo
usuário
```

```
              readline(
                "Deseja desenhar mais uma linha? \n Pressione enter para
continuar ou digite N para parar \n"
              )
            if (finish == "N") { # Se o valor de finish for N
              return("Usuário terminou o diagrama.") # Finaliza o diagrama e
imprime a mensagem
            }
          }
        }
      }
      if (rectangle) { # Caso rectangle seja TRUE
        x0 <- point$x - text.w # Cria o objeto x0 com o valor de x em point,
menos o valor de text.w
        x1 <- point$x + text.w # Cria o objeto x1 com o valor de x em point,
mais o valor de text.w
        y0 <- point$y - text.h # Cria o objeto y0 com o valor de y em point,
menos o valor de text.h
        y1 <- point$y + text.h # Cria o objeto y1 com o valor de y em point,
menos o valor de text.h
        rect(x0, # Cria um retângulo com os valores de x0,x1,y0,y1 e cor com
o valor de col
             y0,
             x1,
             y1,
             col = col)
      }
      if (is.null(text) == FALSE) { # Se text está presente
        text(point$x, point$y, labels = text[i]) # Insere o texto na posição
x e y de point, com o texto da posição i do vetor text
      }
      else { # Se text está ausente
        text(point$x, point$y, labels = text.r)  # Insere o texto na posição
x e y de point, com o texto do objeto text.r (inserido pelo usuário)
      }
    }
    ## Desenho das setas/linhas
    if (is.numeric(arrows)) { # Se arrows está presente
      qt <- arrows # Substitui o valor de qt pelo valor de arrows
    }
    for (i in 1:qt) { # Cria um ciclo de 1 a qt
      if (random.c) { # Caso random.c seja TRUE
        col <- rgb(colors$r[i], colors$g[i], colors$b[i], 1) # Cor do
retângulo/linha será rgb pegando a posição i de cada coluna. Alpha 1.
      }
      else { # Caso random.c não seja TRUE
        col <- colors # Cor do retângulo/linha será o valor de colors (fixo)
      }
      if (arrow.c) { #Se arrow.c é TRUE
        col <- arrow.col #Substitui o valor de col pelo valor de arrow.col
```

```
(inserido pelo usuário)
      }
      cat("Clique no início e término da seta/linha \n") # Imprime mensagem
para o usuário desenhar a seta/linha
      ## Setas
      if (line <= 3) { # Se a linha for do tipo 1 a 3
        point <- locator(2) # Cria o objeto point com os valores de locator
(dois cliques)
        arrows( # Cria uma seta
          point$x[1], #Posição x1 da seta, com o valor da posição 1 de x de
point
          point$y[1], #Posição y1 da seta, com o valor da posição 1 de y de
point
          point$x[2], #Posição x2 da seta, com o valor da posição 2 de x de
point
          point$y[2], #Posição y2 da seta, com o valor da posição 2 de y de
point
          length = 0.1, # Define o comprimento das linhas da ponta da seta
          angle = 30, # Define o ângulo da seta
          code = line, # Define o tipo de seta (1,2 ou 3 - ver help de
arrows para mais informações)
          col = col, # Define a cor da seta com o valor de col
          lwd = 2 # Define a espessura da linha da seta
        )
      }
      ## Linha com pontos na ponta
      if (line == 4) { # Se a linha for do tipo 4
        point <- locator(2) # Cria o objeto point com os valores de locator
(dois cliques)
        segments(point$x[1], # Posição x1 da seta, com o valor da posição 1
de x de point
                 point$y[1], # Posição y1 da seta, com o valor da posição 1
de y de point
                 point$x[2], # Posição x2 da seta, com o valor da posição 2
de x de point
                 point$y[2], # Posição y2 da seta, com o valor da posição 2
de y de point
                 col = col, # Define a cor da linha com o valor de col
                 lwd = 2) # Define a espessura da linha
        points(point$x[1], # Cria um ponto na posição 1 de x de point
               point$y[1], # e posição 1 de y de point
               pch = 19, # do tipo 19
               col = col) # e cor com o valor de col
        points(point$x[2], # Faz o mesmo usando o valor da posição 2 de x de
point
               point$y[2], # e posição 2 de y de point
               pch = 19,
               col = col)
      }
      ## Linha simples
      if (line == 5) { # Se a linha for do tipo 5
```

```
        point <- locator(2) # Cria o objeto point com os valores de locator
(dois cliques)
        segments(point$x[1], # Posição x1 da seta, com o valor da posição 1
de x de point
                 point$y[1], # Posição y1 da seta, com o valor da posição 1
de y de point
                 point$x[2], # Posição x2 da seta, com o valor da posição 2
de x de point
                 point$y[2], # Posição y2 da seta, com o valor da posição 2
de y de point
                 col = col, # Define a cor da linha com o valor de col
                 lwd = 2) # Define a espessura da linha
      }
    }
    return("Usuário terminou o diagrama.") # Retorna a mensagem para o
usuário informando a finalização do diagrama
  }
```