Código A

```
################################
# Animação dos Desvios Mínimos #
################################


##### Código da função #####

anim.msd= function(x, y, lim=1, int=0.05, time=1, cor= "tot",
name="animation.gif",  xlabel="X", ylabel="Y") # criando o nome da função,
os argumentos e seus respectivos defaults
{
  if ((length(grep("animation", library()))==0)=="TRUE") # teste lógico que
procura nos pacotes instalados o pacote "animation" que será necessário para
rodar a função
  {
    stop("Para essa função é necessário instalar o pacote 'animation'(use a
função install.packages()) e o software 'ImageMagick' (disponível em
http://www.imagemagick.org/script/index.php). Antes de tentar rodar a função
novamente, reinicie o programa do R.") # caso o pacote não seja encontrado
uma mensagem de aviso é lançada no console
  }
  if(length(x) != length(y)) # teste lógigo para a saber se os vetores x e y
possuem o mesmo tamanho
  {
    stop("x e y não tem o mesmo tamanho") # caso o resultado seja TRUE, uma
mensagem de aviso é lançada no console
  }
  if(as.numeric(table(is.na(x & y))["FALSE"]) < length(x)) # teste lógico
que verifica se o número de "FALSE" em uma tabela que conta os NAs é menor
do que o tamanho do objeto x
  {
    stop("retire os NA's dos vetores x e/ou y") # caso o resultado do teste
seja verdadeiro existem NAs nos dados e uma mensagem de aviso é lançada no
console
  }
  else # se nenhum dos problemas acima for detectado, a função entra em seu
ciclo normal
  {
    library(animation) # carrega o pacote "animation"
    coef= coef(lm(y ~ x)) #calcula e guarda em um objeto os coeficientes do
modelo linear de y em função de x
    intervalo= seq(coef[2]-lim, coef[2]+lim, by=int) # cria uma sequência
guardada no objeto "intervalo" a partir do coeficiente de inclinação da
regressão com - e + o valor estipulado no argumento "lim" com intervalos de
valor do argumento "int", representando quais inclinações aparecerão na
animação
```

```
    tam.seg= matrix(NA, length(intervalo), length(y)) #cria uma matriz para
guardar o tamanho dos segmentos que representam os desvios para cada reta
testada
    saveGIF( # função que salva os plots e depois converte em GIF, precisa
também do software "Imagemagick" como colocado anteriormante
      {
        for(i in 1:length(intervalo)) # ciclo para plotar as diferentes
retas
        {
          plot(y ~ x, xlab=xlabel, ylab=ylabel) #plota as observações com o
nome dos eixos
          points(mean(x), mean(y), col="red", cex=1.5, pch=19) #coloca o
ponto de fulcro
          abline(mean(y)-intervalo[i]*mean(x), intervalo[i]) # acrescenta a
reta que passa pelo fulcro e possui a inclinação do "intervalo" daquela
rodada (i)
          y1= (mean(y)-intervalo[i]*mean(x)) + (intervalo[i] * x) # calcula
os valores de y da reta para cada x
          tam.seg[i, ]= abs(y - y1) #calcula o tamanho dos desvios (y
observado em relação ao y esperado da reta) e guarda os valores em cada
linha da matriz
          if(cor=="ind") # teste lógico para o argumento "cor==ind", se for
verdadeiro para plotar a cor dos segmentos, cada um terá seu tamanho
comparado com ele mesmo na rodada anterior
          {
            for(j in 1:length(x)) # ciclo para colocar os segmentos dos
desvios
            {
              if(i==1) # se for a primeira rodada
              {
                cores= rep("black", length(x)) # os segmentos terão a cor
preta
              }
              else # nas outras rodadas as cores serão determinadas por um
teste lógico
              {
                cores= tam.seg[i, ] >= tam.seg[i-1, ] # teste para saber se
o segmento aumentou, permaneceu igual ou diminuiu de tamanho
                cores[cores==TRUE]= "red" # se verdadeiro a cor será
vermelha
                cores[cores==FALSE]= "green" # se falso a cor será verde
              }
              segments(x[j], y[j], x[j], y1[j], col=cores[j]) # acrescenta
as linhas dos desvios para o modelo para cada observação com as cores
determinadas pelos testes acima
            }
          }
          if(cor=="tot") # teste lógico para o argumento "cor==tot", se for
verdadeiro para plotar a cor dos segmentos, será comparada a soma do tamanho
dos segmentos de uma rodada com a soma da rodada anterior
```

```
          {
            for(l in 1:length(x)) # ciclo para colocar os segmentos dos
desvios
            {
              if(i==1) # se for a primeira rodada
              {
                cores= rep("black", length(x)) # os segmentos terão a cor
preta
              }
              else # nas outras rodadas as cores serão determinadas por um
teste lógico
              {
                cores= rep(sum(tam.seg[i, ]) >= sum(tam.seg[i-1, ]),
length(x)) # teste para saber se a soma dos segmentos aumentou, permaneceu
igual ou diminuiu de tamanho
                cores[cores==TRUE]= "red" # se verdadeiro a cor será
vermelha
                cores[cores==FALSE]= "green" # se falso a cor será verde
              }
              segments(x[l], y[l], x[l], y1[l], col=cores[l]) # acrescenta
as linhas dos desvios para o modelo para cada observação com as cores
determinadas pelos testes acima
            }
          }
        }
        plot(y ~ x, xlab=xlabel, ylab=ylabel) # plota novamente as
observações
        points(mean(x), mean(y), col="red", cex=1.5, pch=19) #coloca o ponto
de fulcro
        abline(lm(y ~ x), lwd=2, col="blue") # acrescenta a reta da
regressão linear (i.e. a de melhor ajuste)
      }, movie.name=name, interval=c(rep(time, length(intervalo)), 10),
nmax=50, ani.width=600, ani.height=600) # fecha o ciclo da animação e
determina o nome que será dado, a duração entre os "frames" e o tamanho da
janela que será criada
  }
  return(summary(lm(y ~ x))) # retorna um sumário da regressão linear de y
em função de x
} # fim da função
```