Função die

```
die <- function(x, fp, ali, tc, nev, vb, save) #Cria a função die com os
argumentos "x", "fp", "ali", "tc", "nev" e "vb".
{
  #VERIFICANDO PARÂMETROS
  if(x != round(x) | x <= 0)  #Verifica se x é um número inteiro positivo
{
    stop("x precisa ser número inteiro positivo") #Interrompe a função
mostra mensagem
}
if(nev != round (nev) | nev <= 0) #Verifica se nev é um número inteiro
positivo
{
    stop("nev precisa ser número inteiro positivo") #Interrompe a função
mostra mensagem
}
if(length(x) != length(fp)) #Verifica se o comprimento dos vetores "x" e
"fp" são iguais
{
   stop("dimensões de x, e fp devem ser iguais") #Interrompe a função mostra
mensagem
}
if(length(x) != length(ali) | length(x) != length(tc) | length(x) !=
length(nev)) #Verifica se o comprimento dos vetores "x" e "fp" são iguais
{
   warning("Ciclagem de valores ocorrida") #Continua a função e mostra
mensagem de "warning"
}
if(length(nev) != 1) #Verifica se o vetor "nev" tem comprimento igual a 1
{
   stop("número de eventos de alimentação precisa ser único") #Para a função
e mostra mensagem
}
if(vb != "s" && vb != "n") # Verifica se "vb" foi inserido corretamente
{
  stop("vb deve ser s (sim), ou n (não)") #Pára a função e mostra mensagem
}
if(save != "s" && vb != "n") # Verifica se "vb" foi inserido corretamente
{
  stop("save deve ser s (sim), ou n (não)") #Pára a função e mostra mensagem
}
df <- data.frame(x, fp, ali, tc, nev) #Constroi um data frame com os
argumentos numéricos da função.

  #MONTANDO DATA FRAMES
df1 <- matrix(nrow = length(x), ncol = nev+1) #Cria uma matriz vazia cum
número de linhas igual ao número de
                                                #grupos e número de colunas
```

```
um maior que o número de eventos de alimentação
df1[,1] <- fp #Guarda o peso inicial de cada grupo na primeira coluna da
matriz
for(i in 2:ncol(df1)) #Cria um ciclo da segunda coluna até o número final de
colunas da matriz
{
  df1[, i] <- df1[,i-1] + (df1[,i-1]*tc) # Guarda em cada coluna o peso
estimado de cada grupo acrescido
}                                        # da espectativa de engorda
df2 <- matrix(nrow = length(x), ncol = nev) #Cria uma matri resposta com
número de linhas igual ao número de grupos
                                                #e número de colunas igual ao
número de eventos
df2[,1] <- df1[,1]*ali #Guarda na primeira coluna o peso total a ser
oferecido na alimentação
for(j in 2: nev) #Cria um ciclo para guardar os resultados dos cálculos
{
  df2[,j] <- round(df1[,j+1]*ali) #Calcula o peso do alimento a ser
oferecido à partir das estimativas de peso
 }
df2 <- cbind(df2, df[,1]) #Adiciona ao data frame resposta uma coluna com as
quantidades de cada
                            #tipo de alimentação
nc <- paste("alimentação", 1:nev, sep = " ") #Guarda no objeto "nc" a
repetição dos nomes das colunas
                                                 #por evento de alimentação
nl <- paste("grupo", fp, sep = " ") #Guarda no objeto "nc" a repetição dos
nomes das colunas
                                       #por evento de alimentação
colnames(df2) <- c(nc,"quantidade") #Adiciona o nome das colunas do data
frame resposta
rownames(df2) <- nl #Adiciona o nome das linhas do data frame resposta
#MONTANDO CURVAS DE CRESCIMENTO

time <- 0:nev #Cria o objeto "time" com o número de eventos de alimentação

par(las = 1, tcl = 0.3) #Parâmetros para montagem dos gráficos

if(vb == "s") #Verifica se a curva desejada é "Von Bertalanffy" ou não
{
df3 <- matrix(nrow = length(x), ncol = nev+1) #Cria matriz para armazenar o
resultado do cálculo de crescimento

df3[,1] <- fp #Guarda os valores de peso inicial na primeira coluna
for(k in 2:ncol(df3)) #Cria ciclo para armazenar o resultado calculado nas
colunas seguintes
{
  df3[,k] <- df1[,ncol(df1)]*(1-exp(-tc*(time[k]-0))) + fp #Cálculo da
equação de "Von Bertalanffy"
}
```

```
 for(f in 1:nrow(df3)) #Cria ciclo para plotar os gráficos
{
t <- df3[f,] #Cria um vetor "t" com os valores do peso para cada grupo
plot(time, t, pch = 19, cex = 1.3, xlab = "Evento de alimentação") #Plota o
gráfico
                                                                    #de
crescimento pelo número de eventos
lines(time, t, lty = 1, lwd = 2) #Adiciona a curva de crescimento aos pontos
do gráfico
 }
} else
{
for(j in 1:nrow(df1)) #Cria ciclo para plotar os gráficos
{
v <- df1[j,] #Cria um vetor "t" com os valores do peso para cada grupo
plot(time, v, pch = 19, cex = 1.3, xlab = "Evento de alimentação") #Plota o
gráfico de crescimento
                                                                    #pelo
número de eventos
lines(time, v, lty = 1, lwd = 2) #Adiciona a curva de crescimento aos pontos
do gráfico
}
}

if(save == "s") #Verifica se deve salvar o data frame
{
write.table(df2, "die.csv", sep = ";", col.names = NA) #Salva o data frame
resposta em formato csv
}

return(df2) #Devolve o data frame resposta

}
```

From:
http://ecor.ib.usp.br/ - **ecoR**

Permanent link:
http://ecor.ib.usp.br/doku.php?id=05_curso_antigo:r2018:alunos:trabalho_final:hlucascs:fun3

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