Função Final

Help

```
canadian                package:unknown                R Documentation


Cálculo do Canadian Water Quality Index (CWQI)


Descrição:


A função calcula o CWQI e realiza análises temporais a partir de dados
secundários fornecidos
pelo Portal da Qualidade das Águas da ANA (Agência Nacional das Águas). As
planilhas fornecidas
são compostas por valores de diversos parâmetros (e.g. turbidez, oxigênio
dissolvido, condutividade,
sólidos dissolvidos) para determinado periodo, em determinado ponto de
coleta.


Uso:

canadian(dados, limites, periodo, parametro)



Argumentos:



dados                   data.frame com os dados. As linhas são os meses de
observação e as colunas são os
                        valores dos parâmetros. A primeira coluna deve ser
"Mes", com o periodo da observação
                        no formato "ano/mes" (ex.: 2009/01)

limites                 vetor numérico com os limites para cada parâmetro (de
acordo com a legislação vigente
                        aplicável), na ordem em que aparecem nas colunas da
planilha, da esquerda para a direita

periodo                 período para o cálculo do índice. O formato de
entrada pode ser inserido de duas maneiras,
                        ambas com o uso de aspas: "aaaa/mm" ou
"aaaa/mm:aaaa/mm"
```

```
parametro              nome do parâmetro entre aspas desejado para a análise
temporal. Deve estar escrito da
                       mesma forma que no data.frame


Detalhes:


O usuário deve modificar a planilha de dados secundários de forma que as
colunas restantes sejam apenas
'Mes' na primeira coluna e os parâmetros de qualidade da água nas colunas
restantes. 'Mes' deve estar no
formato "ano/mes" (aaaa/mm). Após as modificações e retirada de dados
faltantes (NA) deve-se importar a pla_
nilha para o R, criando o data.frame 'dados'. O indice varia de 0 a 100:

                                          CWQI    Avaliação
                                          95-100  Excellent
                                          80-94   Good
                                          60-79   Fair
                                          45-59   Marginal
                                          0-44    Poor

Valor:


Caso o usuário insira apenas um mês no argumento 'periodo' (formato
"aaaa/mm"), a função retorna o valor de
CWQI. Caso o usário insira um período entre meses ou anos "aaaa/mm:aaaa/mm",
a função retorna gráfico com os
valores de CWQI em função do periodo selecionado. A função também retorna um
gráfico de análise temporal do
parâmetro inserido no argumento "parametro", para todo o periodo contemplado
no data.frame.

Avisos:

A função não trabalha com dados faltantes (NA ou NaN). O usuário deve
adequar sua planilha de dados antes
de importa-la para o R. Caso o usuário insira apenas um "ano/mes" em
'periodo', a análise temporal de 'para_
metro' não é executada. O cálculo do CWQI é realizado independentemente da
quantidade e dos parâmetros dispo_
níveis no data.frame. Os argumentos 'periodo' e 'parametro' devem ser
colocado entre aspas.


Autor:

   Augusto Bitencourt - Instituto de Biociências USP
```

```
   aug.bitencourt@gmail.com


Referências:

Davies, J.M. (2006). Application and tests of the Canadian Water Quality
Index for assessing changes
in water quality in lakes and rivers of central North America. Lake and
Reservoir Management, 22(4), 308–320.


Exemplos:

dados <- read.csv("exemplo.txt", sep="\t", dec=".")
limites<- c(250,0.05,3,0.0002,0.025,10,1,6,7,500,40)
# Utilizando limites para águas doces de classe I, segundo art.14 da
Resolução 357/CONAMA
canadian(dados, limites, "1989/02:1990/11", "Turbidez")
canadian(dados, limites, "1993/04", "Cromo.Total")
canadian(dados, limites, "2000/01:2002/01", "Nitritos")
```

Código

```
canadian <- function(dados, limites, periodo, parametro)
{
  if(missing(dados)) stop("insira dados")                       # Caso o
usuário nao insira o data.frame 'dados' a função emite um aviso
  if(missing(limites)) stop("insira os limites dos parâmetros") # Caso o
usuário nao insira o vetor 'limites' a função emite um aviso
  if(missing(periodo)) stop("insira um período")                # Caso o
usuário nao insira um periodo, a função emite um aviso
  if(class(dados)!="data.frame")                                # Caso
'dados' não seja um data.frame
  {
    stop ("dados precisa ser da classe data.frame")            # A função
emite um aviso
  }
  if(class(limites)!="numeric")                                 # Caso
'limites' não seja um vetor de valores numéricos
  {
    stop ("limites precisa ser da classe numérica")            # A função
emite um aviso
  }
  if(dim(dados)[2]-1!= length(limites))                         # Caso o
número de parametros do data.frame 'dados' seja diferente do tamanho do
vetor 'limites'
  {
    stop("Insira limites para todos os parâmetros presentes no data.frame")
# A função emite um aviso solicitando a correção
```

```
  }
  if (nchar(periodo) == 7)                                  # Caso o número
de caracteres de 'periodo' seja 7 (formato aaaa/mm)
  {
    dados.periodo <-  dados[dados$Mes == periodo,]          # Apenas uma
linha de 'dados' será utlizada para o cálculo de CWQI (sem uso do 'for'),
que será armazenada em 'dados.periodo'
  }
  if (nchar(periodo) == 15)                                 # Caso o número
de caracteres de 'periodo' seja 15 (formato aaaa/mm:aaaa/mm)
  {
    mes1 <-substr(periodo, 1,7)                             # Os caracteres
de 1 a 7 de periodo (primeiro 'mes/ano') são armazenados no objeto 'mes1'
    mes2 <- substr(periodo, 9,15)                           # Os caracteres
de 9 a 15 de periodo (segundo 'mes/ano') são armazenados no objeto 'mes2'
    vetor <- (1:dim(dados)[1])                              # Uma sequência
de 1 até o número de linhas do data.frame 'dados' é armazenada no objeto
'vetor'
    datas <- dados$Mes                                      # A coluna 'Mes'
do data.frame 'dados' é armazenada no objeto 'datas'
    tabela <- data.frame(vetor, datas)                      # Cria-se o
data.frame 'tabela' com 'vetor' e 'datas'
    dados.periodo <- dados[c(tabela$vetor[tabela$datas ==
mes1]:tabela$vetor[tabela$datas == mes2]),] # Os dados dos meses estipulados
pelo usuário no argumento 'periodo' são armazenados no data.frame
'dados.periodo'
  }
  dados.periodo$Mes <- as.character(dados.periodo$Mes)      # Transforma a
coluna 'Mes' de 'dados.periodo' em caracter
  grafico <- rep(NA, dim(dados.periodo)[1])                 # Cria-se o
objeto 'grafico', que irá armazenar os resultados de CWQI para cada mês
  for (k in 1:dim(dados.periodo)[1])                        # Ciclo para o
cálculo do CWQI para todos os meses selecionados no argumento 'periodo'
   {
#########################################################################
#############################################################
    ################################################ F1 consiste na divisão
entre o número de parâmetros que excederam os limites         ##
    ######### Cálculo de F1 (Scope) ################  estabelecidos na
legislação ("falharam" pelo menos 1 vez) e o total de parâmetros   ##
#########################################################################
#############################################################

    falha.par <- rep(NA,dim(dados.periodo)[2]-1)      # Criação do objeto
'falha.par' que contem valores 1 (parâmetro "falhou") ou 0 (parâmetro não
falhou). '-1' é utilizado para descontar a coluna 'Mes'
    for(a in 2:dim(dados.periodo)[2])                 # 'for' para preencher
'falha.par' com 0 ou 1. Inicia com a=2 pra começar a partir da segunda
coluna (onde está o primeiro parãmetro do data.frame)
    {
```

```
      for (b in 1:length(limites))                    # 'for' para inserir o
valor de 'limites' do respectivo parâmetro
      {
        limite <- limites[b]                          # Armazena o limite do
parâmetro em 'limite'
        if(sum(dados.periodo[k,a] > limite ) == 0)    # Teste lógico para
detectar se o limite foi ultrapassado na coluna do parâmetro
        {falha.par[b]=0}                              # Parâmetro não
"falhou": armazena '0' em falha.par
        else {falha.par[b]=1}                         # Parametro "falhou":
armazena '1' em falha.par
      }
    }
    total <- sum(falha.par)                           # Soma total dos
parâmetros que falharam(soma dos valores '1')
    F1 = (total/length(falha.par)) * 100              # Cálculo de F1
##########################################################################
############################################################
    ########################################################## F2 consiste na
divisão entre o número de observações que excederam os limites  ##
    ############## Cálculo de F2 (Frequency) #############           de seu
parâmetro e o total de observações do data.frame              ##
##########################################################################
############################################################
    falha.valor <- rep(NA,dim(dados.periodo)[2]-1)   #  Cria o objeto
'falha.valor' que contem valores 1 (valor "falhou") ou 0 (valor não
"falhou"). '-1' é utilizado para descontar coluna 'Mes'
    for(c in 2:dim(dados.periodo)[2])                 # 'for' para preencher
'falha.valor' com 0 ou 1. Incia com c=2 para começar a partir da segunda
coluna (onde está o primeiro parãmetro do data.frame)
    {
      for (d in 1:length(limites))                    # 'for' para inserir o
valor de 'limites' do  respectivo parâmetro
      {
        limite <- limites[d]                          # Armazena o limite do
parâmetro em 'limite'
        if(sum(dados.periodo[k,c] > limite ) == 0)    # Teste lógico para
detectar se o limite foi ultrapassado pelo valor
        {falha.valor[d]=0}                            # Valor não "falhou":
armazena '0' em falha.valor
        else {falha.valor[d]=1}                       # Valor "falhou":
armazena '1' em falha.valor
      }
    }
    total <- sum(falha.valor)                         # Soma total das
observações(valores) que falharam (soma dos valores '1')
    F2 = total/length(falha.valor)                    # Cálculo de F2
##########################################################################
############################################################
    ##########################################################################
Contabiliza a quantidade de valores que ultrapassaram os limites esta_ ##
```

```
    ########################## Cálculo de F3 (Amplitude) ################
belecidos pela legislação, para mais ou para menos                ##
#############################################################################
#############################################################
    somas.acima <- rep(NA, dim(dados.periodo)[2]-1)        # Objeto para
armazenar a expressao ((valor/limite do parâmetro)-1) de todos parâmetros
que falharam para CIMA (mais que o permitido)
    for(e in 2:dim(dados.periodo)[2])                      # 'for' para
preencher 'somas.acima'. Começa com valor '2' para pular a coluna "Mês"
    {
      for (f in 1:length(limites))                         # 'for' para
inserir o valor de 'limites' do  respectivo parâmetro
      {
        limite <-limites[f]                                # Armazena o
limite do parâmetro em 'limite'
        if(sum(dados.periodo[k,e] > limite ) == 0)         # Teste lógico:
caso o valor não ultrapasse o limite máximo do parâmetro
        {somas.acima[f]=0}                                 # Armazena '0' em
somas.acima
        else {somas.acima[f]= (dados.periodo[1,e]/limite)-1} # Caso valor
ultrapasse o limite máximo do parâmetro, armazena a expressão em somas.acima
      }
    }
    somas.abaixo <- rep(NA, dim(dados.periodo)[2]-1)       #  Objeto para
armazenar a expressao ((limite/valor do parâmetro)-1) de todos parametros
que falharam para BAIXO (menos que o permitido)
    for(g in 2:dim(dados.periodo)[2])                      # 'for' para
preencher 'somas.abaixo'. Começa com valor '2' para pular a coluna "Mês"
    {
      for (h in 1:length(limites))                         # 'for' para
inserir o valor de 'limites' do  respectivo parâmetro
      {
        limite <-limites[h]                                # Armazena o
limite do parâmetro em 'limite'
        if(sum(dados.periodo[k,g] < limite ) == 0)         # Teste lógico:
caso o valor não ultrapasse o limite mínimo do parâmetro
        {somas.abaixo[h]=0}                                # Armazena '0' em
somas.abaixo
        else {somas.abaixo[h]= (limite/dados.periodo[1,g])-1} # Caso valor
ultrapasse o limite mínimo do parâmetro, armazena a expressão em
somas.abaixo
      }
    }
    excursoes = sum(sum(somas.acima) + sum(somas.abaixo))  # Cálculo do
objeto 'excursões', dado pela soma das expressoes ((valor/limite do
parâmetro)-1)
                                                           # (para as
variáveis que falharam para CIMA) e ((limite/valor do parâmetro)-1) (para as
variáveis
                                                           # que falharam
```

```
para BAIXO)
    NSE = excursoes/dim(dados.periodo)[2]-1                # Cálculo do
objeto NSE (Normalized Sum of Excursions)
    F3 = (NSE/(0.01*NSE)+0.01)                             # Cálculo de F3
##########################################################################
#########################
##########################################################################
#########################
    ################################## Cálculo do Indice CWQI
########################################
##########################################################################
#########################
##########################################################################
#########################
    CWQI = 100 - (sqrt((F1)^2+(F2)^2+(F3)^2))/1.732     #  Cálculo do ìndice
CWQI
    grafico[k] <- CWQI                                  #  Armazena o valor
de CWQI na posição "k" do objeto "grafico"
  }                                                     #  Fecha o 'for'
para o cálculo dos indice de cada periodo selecionado
      if(nchar(periodo)==7)                             #  Se o número de
caracteres de 'periodo' for 7 (formato aaaa/mm)
      {return(CWQI)                                     #  A função retorna
apenas o valor de CWQI
      }
  else{                                                 #  Se o número de
caracteres de 'periodo' for 15
    tabela <- data.frame(dados.periodo$Mes,grafico)         # Cria o
data.frame contendo dados.periodo$Mes e os dados dos índices calculados em
'grafico'
    indices <- tabela$grafico                               # Armazena os
valores de CWQI calculados no objeto 'indices'
    meses <- tabela$dados.periodo.Mes                       # Armazenas os
meses do período selecionado em 'meses
    X11()                                                   # Abre janela
gráfica
    par(mar=c(7,4,4,2))                                     # Altera valor
das margens dos gráficos
    plot(indices ~ meses,                                   # Plota
'indices' em função de 'meses'
         las =2,                                            # Altera
disposição horizontal/vertical dos valores nos eixos
         ylab="CWQI",                                       # Altera rótulo
do eixo y
         xlab=NULL,                                         # Gráfico será
construído sem rótulo no eixo x
         main="Canadian Water Quality Index"                # Altera título
do gráfico
         )
    mtext("Meses",side=1,line=5)                            # Insere texto
no gráfico, abaixo do eixo x
```

```
      }
  if(missing(parametro)) stop("Insira um parâmetro para a análise temporal")
# Caso o usuário não insira 'parametro' para análise temporal, a função
emite um aviso
  parametros <- names(dados)[-1]                  # Cria o objeto
'parametros' como o nome dos parâmetros do data.frame
  vetor2 <- seq(1:length(parametros))             # Cria o objeto 'vetor2',
com sequência numérica
  tabela2 <- data.frame (vetor2, parametros)      # Cria o data.frame com os
objetos 'vetor2' e 'parametros'
  x11(width=18, height=10)                        # Cria janela gráfica
  par(mar=c(5,4,4,2))                             # Altera o tamanho das
margens do próximo gráfico
  return(plot(dados[,tabela2$vetor2[tabela2$parametros == parametro]] ~
dados$Mes,    # Plota gráfico da análise temporal do parametro escolhido em
função de dados$Mes
            las=2,                                    # Altera disposição
horizontal/vertical dos eicos
            cex.axis=0.6,                             # Muda tamanho dos
valores dos eixos
            ylab=parametro,                           # Rótulo do eixo y
será 'parametro'
            xlab=NULL,                                # Gráfico será
construído sem rótulo no eixo x
            main="Análise Temporal",                  # Muda o título do
gráfico
            font.axis=4                               # Altera a fonte dos
eixos
             )
         )
}                                                   # Fecha a função
canadian()
```

Exemplo

exemplo.txt

Referência

Davies, J.M. (2006). Application and tests of the Canadian Water Quality Index for assessing changes in water quality in lakes and rivers of central North America. Lake and Reservoir Management, 22(4), 308–320.