código

```
#########################################################################
# Função abs.qpcr()-Quantificação absoluta de genes por PCR quantitativo #
#########################################################################
abs.qpcr=function(ctp, cta, mtd="reaction", cdp=NULL, dlf=NULL, dls=NULL,
vol=NULL, tpl=NULL, mcs=1, cda=NULL)
## A função tem 10 argumentos e nenhum deles é realmente opcional. O que
## mais se aproxima disso é o argumento 'cda' que só é utilizado no método
## 'ngdna'. Todos os outros argumentos devem ser preenchidos ou na entrada
## da função, ou de forma interativa quando requisitado. Não entrar com
## qualquer uma das informações causa erro nos cálculos da função.
  {
    if (missing(ctp)) {
# Caso os dados de cycle threshold (Ct) do padrão estejam faltando...
      stop("Sem dados de 'cycle threshold' da curva padrão. Impossível
continuar.")
# A função para e dá uma mensagem de erro.
      } else if (!is.matrix(ctp) && NCOL(ctp) != 1) {
# Caso os dados de Ct estejam presentes mas em formato diferente de matriz
# ou vetor numérico...
        stop("Os dados de 'cycle threshold' da curva padrão devem estar em
formato de matriz ou de vetor numérico. Impossível continuar.")
# A função para e dá uma mensagem de erro.
        } else if (is.matrix(ctp)) {
# Caso seja uma matriz...
          y=apply(ctp,2,mean)
# Tiramos a média das réplicas e temos os valores de Ct do padrão (a forma
# de entrada dos dados está explicada na página de ajuda).
          } else {
# Caso seja um vetor...
            y=ctp
# Estes já devem ser os valores de Ct do padrão.
            }
#########################################################################
    if (missing(cta)) {
# Caso os dados de cycle threshold (Ct) das amostras estejam faltando...
      stop("Sem dados de 'cycle threshold' das amostras. Impossível
continuar.")
# A função para e dá uma mensagem de erro.
      } else if (!is.matrix(cta) && NCOL(cta) != 1) {
# Caso os dados de Ct estejam presentes mas em formato diferente de matriz
# ou vetor numérico...
        stop("Os dados de 'cycle threshold' das amostras devem estar em
formato de matriz ou de vetor numérico. Impossível continuar.")
# A função para e dá uma mensagem de erro.
        } else if (is.matrix(cta)) {
# Caso seja uma matriz...
```

```
          k=apply(cta,2,mean)
# Tiramos a média das réplicas e temos os valores de Ct das amostras (a
# forma de entrada dos dados está explicada na página de ajuda).
          } else {
# Caso seja um vetor...
            k=cta
# Estes já devem ser os valores de Ct das amostras.
            }
##############################################################################
    metodos = c("reaction", "ngdna")
# Define os métodos de cálculo do numero de cópias de genes alvo.
    mtd = match.arg(mtd, metodos)
# Permite a definição do método com parte do nome.
    if (mtd %in% "ngdna" && is.null(cda))
# Obriga que o argumento 'cda' seja preenchido caso o método utilizado
# seja 'ngdna', caso contrário...
      stop("Para obter o numero de cópias por nanograma de DNA, você deve
fornecer a concentração de DNA das amostras.")
# A função para e dá uma mensagem de erro.
    if (is.null(cdp) || is.null(dlf) || is.null(dls) || is.null(vol) ||
is.null(tpl)) {
# Se estiver faltando pelo menos uma das informações necessárias para o
# cálculo escolhido e não houver default definido...
      readline(prompt="Faltam alguns dados para completar a análise. Vou
requisitá-los um a um. Quando estiver pronto, aperte [enter] para continuar:
")
# A função avisará que faltam informações e aguardará o usuário apertar
# 'enter' para continuar.
      }
##############################################################################
    if (!is.null(cdp)) {
# Caso o argumento 'cdp' não seja nulo...
      stopifnot(is.numeric(cdp), length(cdp) == 1)
# Ele tem que ser um numero ou a função irá parar indicando qual condição
# não foi atendida.
      } else {
# Caso ele seja nulo...
        cdp = as.numeric(readline(prompt="Informe a concentração de DNA
(ng/µl) do padrão: "))
# A função irá requisitar um valor numérico para este argumento.
        }
    if (!is.null(dlf)) {
# Caso o argumento 'dlf' não seja nulo...
      stopifnot(is.numeric(dlf), length(dlf) == 1)
# Ele tem que ser um numero ou a função irá parar indicando qual condição
# não foi atendida.
      } else {
# Caso ele seja nulo...
        dlf = as.numeric(readline(prompt="Informe o fator de diluição: "))
# A função irá requisitar um valor numérico para este argumento.
```

```
        }
##########################################################################
    if (!is.null(dls)) {
# Caso o argumento 'dls' não seja nulo...
      stopifnot(is.integer(dls), length(dls) == length(y))
# Ele tem que ser um vetor de valores integrais com tamanho idêntico a 'y'
# (isso porque essas diluições deram origem aos valores de y).
      } else {
#Caso ele seja nulo...
        dls1=as.numeric(readline(prompt="Informe a primeira diluição
utilizada: "))
# São requisitados o primeiro valor da diluição...
        dls2=as.numeric(readline(prompt="Informe a última diluição
utilizada: "))
# O último valor da diluição...
        dls=c(dls1:dls2)
# E ambos são concatenados para formar a sequência de diluições utilizadas
# para produzir a curva padrão. Esse passo é explicado no help da função
# de forma detalhada na sessão 'notes'.
        }
##########################################################################
    if (!is.null(vol)) {
# Caso o argumento 'vol' não seja nulo...
      stopifnot(is.numeric(vol), length(vol) == 1)
# Ele tem que ser um numero ou a função irá parar indicando qual condição
# não foi atendida.
      } else {
# Caso ele seja nulo...
        vol = as.numeric(readline(prompt="Informe o volume de amostra (µl)
utilizada na reação de qPCR: "))
# A função irá requisitar um valor numérico para este argumento.
        }
    if (!is.null(tpl)) {
# Caso o argumento 'tpl' não seja nulo...
      stopifnot(is.numeric(tpl), length(tpl) == 1)
# Ele tem que ser um numero ou a função irá parar indicando qual condição
# não foi atendida.
      } else {
# Caso ele seja nulo...
        tpl = as.numeric(readline(prompt="Informe o valor (em pares de
bases) do template formado pelo vetor plasmidial + gene de interesse: "))
# A função irá requisitar um valor numérico para este argumento.
        }
    if (!is.null(mcs))
# Caso o argumento 'mcs' não seja nulo...
      stopifnot(is.numeric(mcs), length(mcs) == 1)
# Ele tem que ser um numero ou a função irá parar indicando qual condição
# não foi atendida. Se ele for nulo, seu valor é definido por default.
    if (max(k) > max(y) || min(k) < min(y))
# Caso existam valores de 'k' fora dos limites de máximo e mínimo da curva
# padrão 'y'...
```

```
      warning("Sua análise possui valores de 'cycle threshold' das amostras
fora da curva padrão.")
# Isso não impedirá a análise, mas avisará o usuário da ocorrência. Esse
# fator deve ser levado em consideração na interpretação dos resultados.
##########################################################################
######################### Cálculo da Curva Padrão ########################
##########################################################################
    l=cdp*(dlf^-dls)
# Multiplica-se a concentração de DNA da amostra original pelo fator de
# diluição elevado ao negativo de cada diluição utilizada para gerar a
# curva padrão. Isso gerará as concentrações escalonadas.
    m=(l*vol)*(10^-9)
# Multiplicam-se os valores acima pelo volume de amostra padrão utilizado
# na reação de qPCR, obtendo-se a quantidade total de DNA na reação. Na
# mesma equação o valor é convertido para a unidade 'gramas'.
    n=(tpl*665)/(6.02214e+23)
# Multiplica-se o numero de bases de um template por 665 (peso molecular
# de um par de bases nitrogenadas), e obtemos o peso de cada template em
# Daltons. Divide-se esse valor pelo Numero de Avogadro para obter-se o
# peso, em gramas, do template formado pelo vetor plasmidial mais o gene
# de interesse.
    o=m/n
# Divide-se a massa total de DNA pela massa total de um template para
# obter o total de templates na reação.
    x=log(o*mcs)
# Multiplicando o numero de templates pela quantidade de genes em cada
# template obtemos o numero de genes em cada reação para o padrão. O
# logarítimo neperiano desses valores será utilizado para obter os
# coeficientes linear e angular da curva padrão.
##########################################################################
    r2=summary(lm(y~x))$r.squared
# Determina o coeficiente de determinação da regressão da curva padrão.
    if (r2 < 0.98)
# Caso ela não supere o valor de consenso da literatura para as reações de
# qPCR...
      warning("O coeficiente de determinação (R²) ficou abaixo de 0.98.")
# O usuário será notificado. Isso não impede o cálculo do numero de genes,
# mas deve ser levada em consideração na interpretação dos resultados.
    a=coef(lm(y~x))[1]
# O coeficiente angular da regressão define a eficiência da reação de PCR.
    if (a > -3.1) {
# Caso a eficiência fique acima de 110%, dado pelo coeficiente -3.1...
      warning("A eficiência da reação de qPCR ficou acima do esperado,
indicando que seu padrão pode conter inibidores.")
# O usuário será notificado.
      } else if (a < -3.6) {
# Também se a eficiência ficar abaixo de 90%, dado pelo coeficiente -3.6,
# a reação de qPCR pode ter sido comprometida por diversos fatores...
        warning("A eficiência da reação de qPCR ficou abaixo do esperado.")
# E o usuário será notificado para que possa levar isso em consideração
```

```
# na interpretação dos resultados.
        }
#########################################################################
################## Cálculo do numero de cópias na reação ################
#########################################################################
    b=coef(lm(y~x))[2]
# Definido o coeficiente linear...
    p=(k-a)/b
# Pode-se então aplicá-los para quantificar o numero de genes representado
# pelos valores de Ct das amostras.
    q=exp(p)
# Esse passo inverte a passagem de x a log e determina o numero absoluto
# de genes na reação de qPCR de cada amostra.
    if (mtd %in% "reaction") { # Caso o método de cálculo seja 'reaction'.
      message("Numero de cópias do gene alvo na reação calculado com
sucesso.")
# A função informa que o cálculo ocorreu com sucesso...
      return(q)
# E retorna a quantidade de cópias do gene nas reações.
#########################################################################
############# Cálculo do numero de cópias por nanograma de DNA ##########
#########################################################################
      } else if (mtd %in% "ngdna") {
# Caso o método seja 'ngdna'...
        q1 = q/(cda*vol)
# A quantidade de cópias na reação é dividida pela concentração de DNA das
# amostras corrigida pelo volume de amostra utilizada para a reação,
# obtendo-se o numero de genes por ng de DNA extraído do substrato.
        message("Numero de cópias por nanograma de DNA calculado com
sucesso.")
# O usuário é notificado que este foi o cálculo realizado...
        return(q1)
# E retorna os resultados.
        }
  }
# Fim da função.
#########################################################################
#########################################################################
#########################################################################
```