

## Script da função cqr() | Com que roupa?

```
#a funcao ja ira com padroes preenchidos nas variveis, permitindo o uso na
forma simples cqr() sem especificar nenhum argumento.
cqr <- function(D= (as.numeric(format(Sys.time()+3600, "%d")) -
as.numeric(format(Sys.Date(), "%d"))), Df=
(as.numeric(format(Sys.time()+18000, "%d")) - as.numeric(format(Sys.Date(),
"%d"))), H= as.character(format(Sys.time()+3600, "%H")), Hf=
as.character(format(Sys.time() + 14400, "%H")), L= NULL , Dt , Ht , T01=18,
T02=32, Tj1=24, Tj2=36, TP1=10, TP2=32){ #funcao com argumentos default (mas
dinamicos, por que acompanham o horario em tempo real) ja definidos
  #checar pacote, linha derivada do codigo de fmachado em
http://bie5782.138098.n3.nabble.com/baixar-e-instalar-pacotes-faz-parte-da-funcao-RESOLVIDO-td701125.html
  if(sum(rownames(installed.packages())=="rwunderground")==0){ #verificando
a existencia de um pacote do o nome rwunderground
    stop("O pacote \"rwunderground\" não foi encontrado, para rodar essa
função por favor instale-o. Instruções:
https://github.com/ALShum/rwunderground ou use o comando
install.packages(\"rwunderground\")\" #mensagem caso o pacote não exista
}
  #Caso o pacote esteja presente
  else{ #se o pacote estiver presente (se nao estiver ausente)
    #Abrir o pacote para obtencao de dados meteorologicos
    library("rwunderground") #abrindo o rwunderground
    #Configurar a chave para acessar o API do serviço RWunderground
    rwunderground::set_api_key("27bb942b36a2f0a8") #inserindo a chave API
que permite o uso dos dados (supostamente pessoal, mas seria inviavel todos
terem uma chave propria para rodar a funcao)
    ###checagem de argumentos###
    if(D>11 | D<0 | is.character(D) ){#D deve estar entre 0(hoje) e ultimo
dia (hoje+11), respeitando o limite de 10 dias de previsão que o
rwunderground fornece
      stop("O valor de D (dia inicial da previsão) deve ser numérico e
inteiro e indicar algum dos próximos 10 dias, seguindo a lógica de 0 ser
hoje, 1 amanhã e assim por diante ") #se o argumento D não cumprir as
exigencias, a funcao para
    }
    if(Df>11 | Df<0 | Df<D | is.character(Df) ){ #Df deve estar entre
0(hoje) e ultimo dia (hoje+11), além de ser maior do que o D
      stop("O valor de Df (dia final da previsão) deve ser maior ou igual ao
valor de D, além de numérico e inteiro e indicar algum dos próximos 10 dias,
seguindo a lógica de 0 ser hoje, 1 amanhã e assim por diante.") #se o
argumento Df não cumprir as exigencias, a funcao para
    }
    if(H>23 | H<0 ){ #H deve estar entre 00 e 23 horas, respeitando os
limites do formato de horario que utilizamos no nosso planeta
      stop("O valor de H (hora inicial da previsão) deve ser um número
inteiro contido entre 00 e 23, recomenda-se o uso de aspas para números
entre 00 e 09") #se o argumento H não cumprir as exigencias, a funcao para e
```

essa mensagem é exibida

```
}
  if(Hf>23 | Hf<0 | strptime(Hf,"%H")+(Df*60*60*24)<
strptime(H,"%H")+(D*60*60*24) ){ #Hf deve estar entre 00 e 23, e deve ser
cronologicamente depois de H
  stop("O valor de H deve ser um número inteiro contido entre 00 e 23,
Hf deve ser maior que H, a não ser que Df (Dia final) seja pelo menos um dia
depois de D (Dia inicial)") #se o argumento Hf não cumprir as exigencias, a
funcao para
}

###Não consegui aplicar um teste lógico para a presença de L como
argumento nulo ou não. Algo que para mim seria: if(L != NULL){
  ###message("Para L, nao informe o argumento ou utilize \"L=NULL\" para
ter a localizacao definida automaticamente pelo seu IP, para informar outros
locais, utilize os formatos recomendados no pacote RWunderground
\"https://github.com/ALShum/rwunderground#locations\" informando apenas o
argumento contido entre os parenteses de set_location().")
  #T"x"1 e T"x"2 devem ser números, sendo T"x"1, maior que T"x"2

  if(T02<T01 | is.character(T01) | is.character(T02) ){ #teste hierarquia
lógica T OMBRO
    stop("Ajuste as temperaturas da seção OMBRO: T02 deve ser menor do que
o valor de T01, acima de T01 usa-se a peça mais arejada (regata), abaixo de
T02 a peça mais fechada (blusa), e entre elas, a peça de roupa intermediária
(camiseta)") #Mensagem de erro de não cumprimento das regras entre os
elementos T0
  }
  if(TJ2<TJ1 | is.character(TJ1) | is.character(TJ2) ){ #teste hierarquia
lógica T JOELHO
    stop("Ajuste as temperaturas da seção JOELHO: TJ2 deve ser menor do
que o valor de TJ1, acima de TJ1 usa-se a peça mais arejada (shorts), abaixo
de TJ2 a peça mais fechada (calça), e entre elas, a peça de roupa
intermediária (bermuda)") #Mensagem de erro de não cumprimento das regras
entre os elementos TJ
  }
  if(TP2<TP1 | is.character(TP1) | is.character(TP2) ){ #teste hierarquia
lógica T PÉ
    stop("Ajuste as temperaturas da seção PÉ: TP2 deve ser menor do que o
valor de TP1, acima de TP1 usa-se a peça mais arejada (chinelo), abaixo de
TP2 a peça mais fechada (bota), e entre elas, a peça de roupa intermediária
(tênis)") #Mensagem de erro de não cumprimento das regras hierarquicas entre
os elementos TP
  }
  tbl <- hourly10day(set_location(), use_metric = TRUE) #Previsao do tempo
em si (sera obtido um "Tibble"), em graus celsius (esse passo pode tomar
cerca de 10s ou mais)
  tbl$data_corrig <- tbl$date-25200 #Corrigindo a data/hora para o atraso
de 7h(25200 segundos) da TimeZone e criando a coluna 'data_corrig'
  df <- data.frame(tbl$data_corrig,tbl$feelslike, tbl$rain, tbl$uvi)
```

```
#extraíndo do tibble as variáveis que desejamos usar e transformando em
data.frame (poderia, da mesma forma, usar subset() e continuar o objeto com
um tibble, sem problemas)
#Datas inicio/final no formato para busca, strptime mantém o formato de
dois dígitos mesmo para os números(H e Hf) de 00 a 09 a segunda parte faz o
acréscimo de dias (em segundos)
inicio <- strptime(H,"%H")+(D*60*60*24) #Data de inicio com base em H e
D
final <- strptime(Hf,"%H")+(Df*60*60*24) #Data final com base em Hf e Df

#retornando as posições para indexação do início e do fim na previsão
geral
inic <- grep(inicio, df$tbl.data_corrige) #posicao das data de inicio no
data.frame df
fin <- grep(final, df$tbl.data_corrige) #posicao das data final no
data.frame df

#em caso de troca, a posicao sera definida, analogamente por troc <-
grep(troc, prev$tbl.data_corrige) - prev sera o data.frame contendo so a
previsao que nos importa (ou seja, entre inic e fin)#
#Editando o data.frame de previsao (df) com base nas datas/horarios
escolhidos em relacao as datas corrigidas
prev <- df[inic:fin,] #prev, novo objeto so com o intervalo entre as
datas de inicio e final

#retornando as posições para indexação do início e do fim na previsão
geral COM BASE EM PREV
inicip <- grep(inicio, prev$tbl.data_corrige) #como inic so que desta vez
associado a prev, e não a df, sendo assim compatível com troc
finp <- grep(final, prev$tbl.data_corrige) #como fin so que desta vez
associado a prev, e não a df, sendo assim compatível com troc

#Na verdade o inicip e finp surgiram por conta de um erro quando tentava
puxar as indexações com base apenas no df, o que não entendi direito pq o
prev, apesar de "editado" em relacao a df mostrava as mesmas posicoes no
objeto
## Com a previsao estabelecida e editada, agora partirei para as
determinacoes com base nas condicoes climaticas
#sem troca de roupa enquanto a pessoa estiver fora(entre H e Hf)
if(missing(Dt)){ #se não tiver o objeto Dt como argumento....
  m1 <- mean(prev$tbl.feelslike) #fazer media do feelslike do prev
inteiro

  #Nao encontrei um jeito mais inteligente de fazer atribuicoes
multiplicas ou rodar varios testes logicos indepententes em sequencia, entao
optei por colocar o valor da media em 3 objetos novos
  m1o <- m1 #objeto para conjunto teste logico OMBRO
  m1j <- m1 #objeto para conjunto teste logico JOELHO
  m1p <- m1 #objeto para conjunto teste logico PE

  #conjunto teste logico OMBRO
```

```
if(mlo>T01 & mlo<T02) #teste temperatura amena
{
  ombro <- paste ("CAMISETA") #indicar usar camiseta
}
if (mlo<=T01) #teste "frio"
{
  ombro <- paste ("CASACO/BLUSA") # indicar usar blusa
}
if (mlo>=T02) #teste "calor"
{
  ombro <- paste ("REGATA") #indicar usar regata
}
#Conjunto teste logico JOELHO

if (mlj>TJ1 & mlj<TJ2) #teste temp amena
{
  joelho <- paste ("BERMUDA") #indicar usar bemuda
}
if (mlj<=TJ1) #teste para frio
{
  joelho <- paste ("JEANS/CALÇA") #indicar usar calça
}
if (mlj>=TJ2) #teste para calor
{
  joelho <- paste ("SHORTS") #indicar usar shorts
}
#Conjunto teste logico PE
if (mlp>TP1 & mlp<TP2) # teste temperatura amena
{
  pe <- paste ("TÊNIS/SAPATO") #indicar usar tenis
}
if (mlp<=TP1) #teste para frio
{
  pe <- paste ("BOTA") #indicar usar bota
}
if (mlp>=TP2) #teste para calor
{
  pe <- paste ("CHINELO") #indicar usar chinelo
}
}
if(missing(Ht)){ # se o argumento Ht nao existir, nao havera troca

  ##tudo ocorre assim como na falta apenas do Dt, mas nao consegui resumir
Dt e Ht em um unico teste por uma questao de sintaxe

  ml <- mean(prev$tbl.feelslike) #calculo da média sem troca

  #Nao encontrei um jeito mais inteligente de fazer atribuicoes
multiplas ou rodar varios testes logicos indepentes em sequencia, entao
optei por colocar o valor da media em 3 objetos novos
```

```
mlo <- m1 #objeto para conjunto teste logico OMBRO
mlj <- m1 #objeto para conjunto teste logico JOELHO
mlp <- m1 #objeto para conjunto teste logico PE

#Conjunto teste logico OMBRO
if(mlo>T01 & mlo<T02)#teste de temperatura amena
{
  ombro <- paste ("CAMISETA") #indicar usar camiseta
}
if (mlo<=T01) #teste para temperatura "frio"
{
  ombro <- paste ("CASACO/BLUSA") #indicar usar blusa
}
if (mlo>=T02) #teste para calor
{
  ombro <- paste ("REGATA") #indicar usar regata
}
#Conjunto teste logico JOELHO
if (mlj>TJ1 & mlj<TJ2)#teste de temperatura amena
{
  joelho <- paste ("BERMUDA") #indicar uso de bermuda
}
if (mlj<=TJ1) #teste para temperatura "frio"
{
  joelho <- paste ("JEANS/CALÇA") #indicar uso de calça
}
if (mlj>=TJ2) #teste para calor
{
  joelho <- paste ("SHORTS") #indicar uso de shorts
}
#Conjunto teste logico PE
if (mlp>TP1 & mlp<TP2)#teste de temperatura amena
{
  pe <- paste ("TÊNIS/SAPATO") #indicar usar tenis
}
if (mlp<=TP1) #teste para temperatura "frio"
{
  pe <- paste ("BOTA") #indicar usar bota
}
if (mlp>=TP2) #teste para calor
{
  pe <- paste ("CHINELO") #indicar usar chinelo
}
}
#COM troca de roupa (contendo os argumentos Dt & Ht)
else #caso os argumentos e Ht e Dt estejam presentes
{
  troca <- strptime(Ht,"%H")+(Dt*60*60*24) #Dia(Dt) e horario(Ht) da
troca
  troc <- grep(troca, prev$tbl.data_corrig) #posicao da troca no df com
as datas, o mesmo que foi usado como argumento para o teste logico de não
```

```
poder ser antes do horario de inicio/depois do final
  if (troc<inicip | troc>finp) #teste para ver se troc está entre os
limite iniciais e finais de posicao de datas
  {
    stop ("Os argumentos Dt (dia da troca) e Ht (hora da troca) devem
estar presentes, a troca deve estar contida entre o valor de saída (H em D)
e de chegada (Hf em Df)") #mensagem caso a data de troca nao esteja entre o
inicio e o fim
  }

  #calculo das duas medias de sensacao termica (antes (m1) e depois (m2)
da troca)
  m1 <- mean(prev$tbl.feelslike[inicip:troc]) #Obtendo as duas medias, m1
(inicio ate troca)
  m2 <- mean(prev$tbl.feelslike[troc: finp]) #e m2 (troca ate o final)
  #Definicao das pessas de roupa de modo analogo ao cenario sem troca,
mas dessa vez teremos 2 objetos possiveis para cada parte (antes e depois da
troca (ombroT, joelhoT, peT)
  #objetos para conjunto teste logico OMBRO
  m1o <- m1 #temperatura antes da troca
  m2o <- m2 #temperatura depois da troca

  #objetos para conjunto teste logico JOELHO
  m1j <- m1 #temperatura antes da troca
  m2j <- m2 #temperatura depois da troca

  #objetos para conjunto teste logico PE
  m1p <- m1 #temperatura antes da troca
  m2p <- m2 #temperatura depois da troca
  #Conjunto teste logico OMBRO
  if (m1o> T01 & m1o< T02) #teste de temperatura amena antes da troca
  {
    ombro <- paste ("CAMISETA") #indicar usar camiseta antes da troca
  }
  if (m1o<=T01) #teste para temperatura "frio" antes da troca
  {
    ombro <- paste ("CASACO/BLUSA") #indicar usar blusa antes da troca
  }
  if (m1o>=T02) #teste para calor antes da troca
  {
    ombro <- paste ("REGATA") #indicar usar regata antes da troca
  }
  if (m2o>T01 & m2o<T02) #teste de temperatura amena pos-troca
  {
    ombroT <- paste ("CAMISETA") #indicar usar camiseta depois da troca
  }
  if (m2o<=T01) #teste para temperatura "frio" pos-troca
  {
    ombroT <- paste ("CASACO/BLUSA") #indicar usar blusa depois da troca
  }
}
```

```
if (m2o>=T02) #teste para calor depois da troca
{
  ombroT <- paste ("REGATA") #indicar usar regata depois da troca
}
#Conjunto teste logico JOELHO
if (m1j>TJ1 & m1j<TJ2) #teste de temperatura amena antes da troca
{
  joelho <- paste ("BERMUDA") #indicar usar bermuda antes da troca
}
if (m1j<=TJ1) #teste para temperatura "frio" antes da troca
{
  joelho <- paste ("JEANS/CALÇA") #indicar usar calça antes da troca
}
if (m1j>=TJ2) #teste para calor antes da troca
{
  joelho <- paste ("SHORTS") #indicar usar shorts antes da troca
}
if (m2j>TJ1 & m2j<TJ2) #teste de temperatura amena pos troca
{
  joelhoT <- paste ("BERMUDA") #indicar usar bermuda depois da troca
}
if (m2j<=TJ1) #teste para temperatura "frio" pos-troca
{
  joelhoT <- paste ("JEANS/CALÇA") #indicar usar calça depois da troca
}
if (m2j>=TJ2) #teste para calor depois da troca
{
  joelhoT <- paste ("SHORTS") #indicar usar shorts depois da troca
}
#Conjunto teste logico PE
if (m1p>TP1 & m1p<TP2) #teste de temperatura amena antes da troca
{
  pe <- paste ("TÊNIS/SAPATO") #indicar usar tenis antes da troca
}
if (m1p<=TP1) #teste para temperatura "frio" antes da troca
{
  pe <- paste ("BOTA") #indicar usar bota antes da troca
}
if (m1p>=TP2) #teste para calor antes da troca
{
  pe <- paste ("CHINELO") #indicar usar chinelo antes da troca
}
if (m2p>TP1 & m2p<TP2) #teste de temperatura amena antes pos troca
{
  peT <- paste ("TÊNIS/SAPATO") #indicar usar tenis depois da troca
}
if (m2p<=TP1) #teste para temperatura "frio" pos troca
{
  peT <- paste ("BOTA") #indicar usar bota depois da troca
}
if (m2p>=TP2) #teste de temperatura para "calor" antes da troca
```

```
{
  peT <- paste ("CHINELO") #indicar usar chinelo depois da troca
}
}
```

#CHUVA, a coluna rain pode apresentar os valores 0 ou 1, a dinamica da variavel nao esta bem documentada, mas encontrei valores 1 apenas em probabilidades de precipitacao(PoP) maiores do 40%, não é detalhado se a PoP contabiliza outros tipos de precipitacao, mas dado o nome da coluna, achei que RAIN seria uma boa indicadora de chuva

#A função sum e a característica binaria da variavel "rain" criam um conjunto facil de ser trabalhado com o teste logico se o valor é maior do 0 em algum ponto da previsao

```
if (sum(prev$tbl.rain)>0) #se o valor for maior do que 0 (no caso 1, é porque houve pelo menos um ponto (uma hora) em que se indica chuva (1).
{
  pe <- paste("BOTA DE CHUVA") #caso chova, é indicado usar bota
  peT <- paste("BOTA DE CHUVA") #usar bota depois da troca tambem
  chuva <- paste("Provavelmente irá chover, leve um GUARDA CHUVA ou CAPA.") #mensagem caso haja previsao chuva
}
else #caso seja zero, nao ha previsao de chuva
{
  chuva <- paste("NÃO houve previsão de chuva para sua região nestes horários.") #mensagem no caso de nao haja chuva
}
#UV, recomenda-se o uso de protetor solar em quase todas as situacoes, porem limitei o aviso apenas para niveis acima do moderado (6)
if (sum(prev$tbl.uvi)>6)#se o índice uv for maior do que 6
{
  UV <- paste("\n"USE FILTRO SOLAR, se eu pudesse dar só uma dica sobre o futuro seria esta.\" BIAL, Pedro.") #Mensagem de retorno avisando para o uso de protetor solar usando a o meme do pedro bial de um tempo que ainda nao existiam memes
}
else #caso o nível não seja maior do que 6
{
  UV <- paste("O índice UV está dentro ou abaixo dos níveis considerados moderados.") #Mensagem que não haja risco UV
}
#Caso não haja troca de roupa, o retorno será...
if(missing(Dt)){ #dessa vez a logica pode ser encurtada pois ja a linhas que dao conta de suprimir que a funcao fucione sem Ht, entao pude colocar so Dt na condicao
#Compondo a lista para retorno
vestir <- paste("Para a previsão entre ",format(inicio, "%d/%m/%Y %H:%M")," e ",format(final, "%d/%m/%Y %H:%M"),", é indicado o uso de ",(ombro)," ",(joelho)," e ",(pe)," ", sep="") #lista de retorno sem troca
m1 <- paste(round(m1,digits = 1)," °C", sep= "") #Arrendondando o valor da temperatura e adicionando o sinal de Celsius
lista.return <- list(vestir, m1, chuva, UV) #compilando os elementos da
```



```
lista de retorno
  names(lista.return) <- c("O QUE VESTIR", "MÉDIA DA SENSÇÃO TÉRMICA",
"PROBABILIDADE DE CHUVA", "ÍNDICE UV") #dando nome aos elementos da lista de
retorno
}
#Retorno COM troca de roupa.
else{ #caso haja troca de roupa
  #Compondo a lista para retorno
  vestir <- paste("Para a previsão entre ",format(inicio, "%d/%m/%Y
%H:%M")," e ",format(final, "%d/%m/%Y %H:%M"),", é indicado o uso de
",(ombro)," ",(joelho)," e ",(pe)," até a troca de roupa programada para ",
format(troca, "%d/%m/%Y %H:%M"),", posteriormente recomenda-se "
,(ombroT)," ",(joelhoT)," e ",(peT),".", sep="") #lista de retorno com
troca
  m1 <- paste(round(m1,digits = 1)," °C", sep="") #Arrendondando os
valores da médias de temperatura antes da troca e adicionando o sinal de
Celsius
  m2 <- paste(round(m2,digits = 1)," °C", sep="") #Arrendondando os
valores da médias de temperatura depois da troca e adicionando o sinal de
Celsius
  lista.return <- list(vestir, c(m1,m2), chuva, UV) #compilando os
elementos para a lista de return
  names(lista.return) <- c("O QUE VESTIR", "MÉDIA DA SENSÇÃO TÉRMICA
(antes da troca / depois da troca)", "PROBABILIDADE DE CHUVA", "ÍNDICE UV")
#nomeando os itens da lista de return
}
}
#retorno da lista com as peças de roupa, sensação térmica e os avisos de
chuva e intensidade UV
return(lista.return)#FIM
}
```

From:

<http://labtrop.ib.usp.br/> - Laboratório de Ecologia de Florestas Tropicais

Permanent link:

[http://labtrop.ib.usp.br/doku.php?id=cursos:ecor:05\\_curso\\_antigo:r2018:alunos:trabalho\\_final:giovane.improta:script.cqr](http://labtrop.ib.usp.br/doku.php?id=cursos:ecor:05_curso_antigo:r2018:alunos:trabalho_final:giovane.improta:script.cqr)

Last update: 2020/07/27 21:48

