Brewtool

```
brewtool <- function(SG, OG, t.mosto, t.cal = 20, vol, tempo.f = 60, IBU =
FALSE, input) # Cria a funcao "brewtool" com os argumentos "SG", "OG",
"t.mosto", "t.cal" (default = 20), "vol", "tempo.f" (default = 60), "IBU"
(default = FALSE) e "input"
{
  # VERIFICANDO OS PARAMETROS
  if (SG < 1.020 | SG > 1.120) # Caso SG nao esteja entre 1.020 e 1.120,
  {
    stop("SG fora do intervalo permitido, 1.020 < SG < 1.120") # interrrompe
a funcao e exibe a mensagem de erro para o usuario.
  }
  if (OG < 1.020 | OG > 1.120)  # Caso SG nao esteja entre 1.020 e 1.120,
  {
    stop("OG fora do intervalo permitido, 1.020 < OG < 1.120") # interrrompe
a funcao e exibe a mensagem de erro para o usuario.
  }
  if ((nchar(SG) - 2) > 3 | (nchar(OG) - 2) > 3) # Caso o numero de decimais
de SG ou OG seja maior que 3,
  {
    warning("SG e OG devem ser numeros com 3 casas decimais. O valor
digitado foi ajustado para 3 casas.") # exibe a mensagem de alerta
informando o usuario que os valores serao arredondados para 3 casas.
  }
  if (t.mosto < 0 | t.mosto > 85) # Caso t.mosto nao esteja entre 0 e 85,
  {
    stop("t.mosto deve ser um numero entre 0 e 85.") # interrompe a funcao e
exibe a mensagem de erro para o usuario.
  }
  if (t.cal <= 0) # Caso t. cal seja menor que 0,
  {
    stop("t.cal deve ser um numero maior que 0.") # interrrompe a funcao e
exibe a mensagem de erro para o usuario.
  }
  if (vol[1] < 0 | vol[2] < 0) # Caso um dos elementos de vol seja menor que
0,
  {
    stop("o vetor vol deve conter apenas numeros maiores que 0") #
interrompe a funcao e exibe a mensagem de erro para o usuario.
  }
  if (length(vol) > 2) # Caso numero de elementos de vol seja maior que 2,
  {
    warning("O vetor vol possui mais de 2 elementos. Apenas os dois
primeiros serao utilizados (v[1] = vol. pre-fervura; v[2] = vol. pos-
fervura.") # exibe a mensagem de alerta informando o usuario que apenas os
dois primeiros elementos serao utilizados.
  }
  if (vol[1] <= vol[2]) # Caso o primeiro elemento (vol. pre-fervura) seja
```

```
menor que o segundo elemento (vol. pos-fervura),
  {
    warning("Volume pre-fervura menor ou igual ao volume pos-fervura. A taxa
de evaporacao nao podera ser calculada. Verifique a ordem dos elementos do
vetor (vol = c(pre, pos)).") # exibe a mensagem de alerta para o usuario.
  }
  if (tempo.f <= 0) # Caso o tempo de fervura seja menor ou igual a 0,
  {
    stop("tempo.f deve ser um numero inteiro > 0") # interrompe a funcao e
exibe a mensagem de erro para o usuario.
  }
  if (IBU == TRUE) # Caso o argumento IBU seja verdadeiro,
  {
    if (class(input) != "data.frame") # e caso a classe do argumento input
seja diferente de "data.frame",
    {
      stop("O objeto input deve ser da classe 'data.frame'") # interrompe a
funcao e exibe a mensagem de erro para o usuario.
    }
    input.names = colnames(input) # Cria o objeto input.names, contendo os
nomes das colunas de input
    counter = 0 # Cria um objeto para ser usado como contador, com valor
inicial igual a 0
    for (i in 1:length(input.names)) # Ciclo for com contador i que vai de 1
ao tamanho do vetor input.names
    {
      if (input.names[i] == "peso" | input.names[i] == "tempo" |
input.names[i] == "aa") # Caso algun dos elementos do vetor input.names seja
igual a "peso", "tempo" ou "aa",
      {
        counter = counter + 1 # somar 1 ao objeto counter.
      }
    }
    if (counter != 3) # Caso o objeto counter seja diferente de 3,
    {
      stop("As informacoes de input nao estão corretas. Verifique a
nomenclatura das colunas (colunas = 'peso', 'tempo', 'aa').") # interrompe a
funcao e exibe a mensagem de erro para o usuario.
    }else # Caso counter seja igual a 3:
    {
      if (class(input$peso) != "numeric" | class(input$tempo) != "numeric" |
class(input$aa) != "numeric") # Caso a classe das colunas "peso", "tempo", e
"aa" nao seja igual a "numeric"
      {
        stop("As informacoes de input nao estao corretas. As colunas 'peso',
'tempo' e 'aa' devem conter apenas numeros.") # interrompe a funcao e exibe
a mensagem de erro para o usuario.
      }
      if (is.na(sum(input$peso, input$tempo, input$aa)) == TRUE) # Caso a
soma das colunas "peso", "tempo", e "aa" seja NA
```

```
      {
        stop("As informacoes de input nao estao corretas. As colunas 'peso',
'tempo' e 'aa' devem conter o mesmo numero de elementos e nao podem conter
NAs.") # interrrompe a funcao e exibe a mensagem de erro para o usuario.
      }
    }
  }
  # CORRECAO DE SG PELA TEMPERATURA (PRIMEIRO OUTPUT) E AJUSTE DOS OBJETOS:
  SG.cal =
round(SG*((1.00130346-0.000134722124*((t.mosto*9/5)+32)+0.00000204052596*((t
.mosto*9/5)+32)^2-0.00000000232820948*((t.mosto*9/5)+32)^3)/
(1.00130346-0.000134722124*((t.cal*9/5)+32)+0.00000204052596*((t.cal*
9/5)+32)^2-0.00000000232820948*((t.cal*9/5)+32)^3)), 3)  # Cria o objeto
SG.cal corrigindo o valor de SG de acordo com as temperaturas do mosto no
momento da medicao
#+e de calibracao do densimetro, depois arredonda o novo valor para 3 casas
decimais.
#+A formula para o calculo de SG.cal foi retirada de
https://straighttothepint.com/hydrometer-temperature-correction/
  OG = round(OG, 3) # Arredonda o valor de OG para 3 casas decimais
  dif = OG - SG.cal # Cria o objeto dif contendo a diferenca entre OG e
SG.cal
  SG.cal = format(SG.cal, nsmall = 3) # Formata SG.cal para 3 casas decimais
obrigatoriamente
  SG = format(round(SG, 3), nsmall = 3) # Arredonda e formata SG para 3
casas decimais
  OG = format(OG, nsmall = 3) # Formata OG para 3 casas decimais
obrigatoriamente
  tempo.f = round(tempo.f) # Arredonda tempo.f
  if (SG.cal < 1.1) # Caso SG.cal seja menor que 1.1,
  {
    SG.dig = as.numeric(c((unlist(strsplit(SG.cal, split =
NULL)))[4],(unlist(strsplit(SG.cal, split = NULL)))[5])) # Cria o vetor
SG.dig com os 2 ultimos digitos de SG.cal e o converte em um vetor numerico
    SG.dig = round((SG.dig[1]*10)+SG.dig[2]) # Transforma o vetor SG.dig em
um numero inteiro
  }else # Caso seja maior ou igual a 1.1,
  {
    SG.dig = as.numeric(c((unlist(strsplit(SG.cal, split = NULL)))[3],
(unlist(strsplit(SG.cal, split = NULL)))[4],(unlist(strsplit(SG.cal, split =
NULL)))[5])) # Cria o vetor SG.dig com os 3 ultimos digitos de SG.cal e o
converte em um vetor numerico
    SG.dig = round((SG.dig[1]*100)+(SG.dig[2]*10)+SG.dig[3]) # Transforma o
vetor SG.dig em um numero inteiro
  }
  if (OG < 1.1) # Caso OG seja menor que 1.1
  {
    OG.dig = as.numeric(c((unlist(strsplit(OG, split =
NULL)))[4],(unlist(strsplit(OG, split = NULL)))[5])) # Cria o vetor OG.dig
com os 2 ultimos digitos de OG e o converte em um vetor numerico
    OG.dig = round((OG.dig[1]*10)+OG.dig[2]) # Transforma o vetor OG.dig em
```

```
um numero inteiro
  }else # Caso OG seja maior ou igual a 1.1
  {
    OG.dig = as.numeric(c((unlist(strsplit(OG, split = NULL)))[3],
(unlist(strsplit(OG, split = NULL)))[4], (unlist(strsplit(OG, split =
NULL)))[5])) # Cria o vetor OG com os 3 ultimos digitos de OG e o converte
em um vetor numerico
    OG.dig = round((OG.dig[1]*100)+(OG.dig[2]*10)+OG.dig[3]) # Transforma o
vetor OG.dig em um numero inteiro
  }
  SG.pre = round((SG.dig*vol[2])/vol[1]) # Cria o objeto SG.pre contendo os
2 ultimos digitos do valor de SG antes da fervura (formula --> Ci*Vi =
Cf*Vf)

  # TESTES CONDICIONAIS PARA DIF PARA DETERMINACAO DE COMO AJUSTAR SG.CAL
(SEGUNDO OUTPUT):
  if (dif == 0) # Caso OG seja = SG.cal e, portando, dif seja 0,
  {
    print.og = cat(paste("Sua SG pre-fervura foi calculada em ",
format(((SG.pre/1000)+1), nsmall = 3), ".\n", "Sua OG eh igual a ", SG.cal,
" e, portanto, ja eh igual a OG desejada, nada precisa ser feito. Prossiga
com a receita.\n", sep = "")) # Cria o objeto print.og com a mensagem a ser
impressa para o usuario.
  }
  if (dif < 0) # Caso OG seja < SG.cal e, portando, dif seja < 0,
  {
    v.final = round(((SG.dig*vol[2])/OG.dig), 1) # Cria o objeto v.final com
o calculo do volume final necessario para o ajuste da OG
    v.adic = round((v.final - vol[2]), 1) # Cria o objeto v.adic com o
calculo do volume de agua a ser adicionado
    print.og = cat(paste("Sua SG pre-fervura foi calculada em ",
format(((SG.pre/1000)+1), nsmall = 3), ".\n", "Sua OG eh igual a ", SG.cal,
". Para ajusta-la para ", OG,  ", adicione ", v.adic, " litros de agua
esterilizada. Seu volume final sera de ", v.final, " litros.\n", sep = ""))
# Cria o objeto print.og com a mensagem a ser impressa para o usuario
  }
  if (dif > 0) # Caso OG seja > SG.cal e, portando, dif seja > 0,
  {
    if (vol[1] <= vol[2]) # Caso o volume pre-fervura seja igual ou menor
que o volume pos-fervura,
    {
      print.og = cat(paste("Sua SG pre-fervura foi calculada em ",
format(((SG.pre/1000)+1), nsmall = 3), ".\n","Sua OG eh igual a ", SG.cal,
". Como nao foi possivel calcular a taxa de evaporacao, nao sera possivel
ajusta-la para ", OG, ".\n", sep = "")) # Cria o objeto print.og com a
mensagem a ser impressa para o usuario
    }else # Caso o volume pre-fervura seja maior que o volume pos-fervura
    {
      ev = abs(diff(vol))/tempo.f # Cria o objeto ev com o calculo da taxa
de evaporacao
```

```
      v.final.ferv = round(((SG.dig*vol[2])/OG.dig), 1) # Cria o objeto
v.final.ferv com o calculo do volume final necessario para o ajuste da OG
      v.evap = vol[2] - v.final.ferv # Cria o objeto v.evap com o volume a
ser evaporado para o ajuste da OG
      tempo = round(v.evap/ev) # Cria o objeto tempo com o calculo do tempo
adicional de fervura para o ajuste da OG
      print.og = cat(paste("Sua SG pre-fervura foi calculada em ",
format(((SG.pre/1000)+1), nsmall = 3), ".\n", "Sua OG eh igual a ", SG.cal,
". Para ajusta-la para ", OG, ", ferva o mosto por mais ", tempo, " minutos.
Seu volume final sera de ", v.final.ferv, " litros.\n", sep = "")) # Cria o
objeto print.og com a mensagem a ser impressa para o usuario
    }
  }
  # CALCULO DO IBU (TERCEIRO OUTPUT):
  if (IBU == TRUE) # Caso o argumento IBU seja verdadeiro,
  {
    if (SG.pre < 30 | SG.pre > 120) # Caso SG.pre esteja fora do intervalo
entre 30 e 120,
    {
      print.ibu = cat(paste("Sua SG pre-fervura eh um valor fora do
intervalo 1.030 - 1.120. O IBU nao pode ser calculado.\n", sep = "")) # Cria
o objeto print.ibu com a mensagem a ser impressa para o usuario
    }else # Caso SG.pre esteja no intervalo 30-120,
    {
      # Criando os vetores que comporao o data frame a.acidos contendo a
utilizacao de alfa acidos por tempo de fervura para valores de SG pre-
fervura entre 1.030 e 1.120
      # Os valores para a.acidos foram retirados de
https://concerveja.com.br/calcular-ibu/
      SG.a.acidos = c(30, 40, 50, 60, 70, 80, 90, 100, 110, 120) # Criando o
vetor contendo os valores de SG da tabela
      Gravity_Time = c(0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 70,
80, 90, 100, 110, 120) # Tempos de fervura
      SG.30 = c(0.000, 0.055, 0.100, 0.137, 0.167, 0.192, 0.212, 0.229,
0.242, 0.253, 0.263, 0.270, 0.276, 0.285, 0.291, 0.295, 0.298, 0.300, 0.301)
# Valores de utilizacao para SG.pre = 1.030
      SG.40 = c(0.000, 0.050, 0.091, 0.125, 0.153, 0.175, 0.194, 0.209,
0.221, 0.232, 0.240, 0.247, 0.252, 0.261, 0.266, 0.270, 0.272, 0.274, 0.275)
# Valores de utilizacao para SG.pre = 1.040
      SG.50 = c(0.000, 0.046, 0.084, 0.114, 0.140, 0.160, 0.177, 0.191,
0.202, 0.212, 0.219, 0.226, 0.231, 0.238, 0.243, 0.247, 0.249, 0.251, 0.252)
# Valores de utilizacao para SG.pre = 1.050
      SG.60 = c(0.000, 0.042, 0.076, 0.105, 0.128, 0.147, 0.162, 0.175,
0.185, 0.194, 0.200, 0.206, 0.211, 0.218, 0.222, 0.226, 0.228, 0.229, 0.230)
# Valores de utilizacao para SG.pre = 1.060
      SG.70 = c(0.000, 0.038, 0.070, 0.096, 0.117, 0.134, 0.148, 0.160,
0.169, 0.177, 0.183, 0.188, 0.193, 0.199, 0.203, 0.206, 0.208, 0.209, 0.210)
# Valores de utilizacao para SG.pre = 1.070
      SG.80 = c(0.000, 0.035, 0.064, 0.087, 0.107, 0.122, 0.135, 0.146,
0.155, 0.162, 0.168, 0.172, 0.176, 0.182, 0.186, 0.188, 0.190, 0.191, 0.192)
# Valores de utilizacao para SG.pre = 1.080
```

```
      SG.90 = c(0.000, 0.032, 0.058, 0.080, 0.098, 0.112, 0.124, 0.133,
0.141, 0.148, 0.153, 0.157, 0.161, 0.166, 0.170, 0.172, 0.174, 0.175, 0.176)
# Valores de utilizacao para SG.pre = 1.090
      SG.100 = c(0.000, 0.029, 0.053, 0.073, 0.089, 0.102, 0.113, 0.122,
0.129, 0.135, 0.140, 0.144, 0.147, 0.152, 0.155, 0.157, 0.159, 0.160, 0.161)
# Valores de utilizacao para SG.pre = 1.100
      SG.110 = c(0.000, 0.027, 0.049, 0.067, 0.081, 0.094, 0.103, 0.111,
0.118, 0.123, 0.128, 0.132, 0.135, 0.139, 0.142, 0.144, 0.145, 0.146, 0.147)
# Valores de utilizacao para SG.pre = 1.110
      SG.120 = c(0.000, 0.025, 0.045, 0.061, 0.074, 0.085, 0.094, 0.102,
0.108, 0.113, 0.117, 0.120, 0.123, 0.127, 0.130, 0.132, 0.133, 0.134, 0.134)
# Valores de utilizacao para SG.pre = 1.120
      a.acidos = data.frame(Gravity_Time, SG.30, SG.40, SG.50, SG.60, SG.70,
SG.80, SG.90, SG.100, SG.110, SG.120, check.names = FALSE) # Cria o data
frame a.acidos
      IBU.pre = rep(NA, nrow(input)) # Cria o vetor IBU.pre onde serao
armazenados os valores de IBU para cada lupulo de input, para vol[2]
      IBU.pos = rep(NA, nrow(input)) # Cria o vetor IBU.pos onde serao
armazenados os valores de IBU para cada lupulo de input, para v.final
      for (i in 1:length(IBU.pre)) # Cria um ciclo com contador i, de 1 ate
o tamanho do vetor IBU.pre
      {
        tempo.l = tempo.f - input$tempo[i] # Cria o objeto tempo.l com o
tempo de fervura do lupulo[i]
        for (j in 1:length(SG.a.acidos)) # Cria um ciclo com contador j, de
1 ate o tamanho do vetor SG.a.acidos
        {
          if (SG.pre == SG.a.acidos[j]) # Caso SG.pre seja igual a
SG.acidos[j],
          {
            SG.U = names(a.acidos)[j+1] # Cria o objeto SG.U contendo o nome
da coluna j+1 de a.acidos
            for (k in 1:nrow(a.acidos)) # Cria um ciclo com contador k, de 1
ate o numero de linhas de a.acidos
            {
              if (tempo.l == a.acidos$Gravity_Time[k]) # Caso tempo.l seja
igual ao valor da linha k da coluna Gravity_Time de a.acidos,
              {
                U = a.acidos[k, SG.U] # Cria o objeto U contendo o valor de
utilizacao de alfa acidos correspondente a linha k e coluna SG.U
                break # Quebra o ciclo for com contador k
              }else if (tempo.l < a.acidos$Gravity_Time[k]) # Caso tempo.l
seja menor que o valor da linha k da coluna Gravity_Time de a.acidos,
              {
                U1 = a.acidos[k-1, SG.U] # Cria o objeto U1 contendo o valor
de utilizacao de alfa acidos correspondente a linha k-1 e coluna SG.U
                U2 = a.acidos[k, SG.U] # Cria o objeto U1 contendo o valor
de utilizacao de alfa acidos correspondente a linha k e coluna SG.U
                dif.tempo = tempo.l - a.acidos$Gravity_Time[k-1] # Cria o
objeto dif.tempo com a diferenca exata entre tempo.l e o tempo k-1 da coluna
```

```
Gravity_Time de a.acidos
                U = round((U1+((U2 - U1)/10)*dif.tempo), 3) # Cria o objeto
com a interpolacao dos valores U1 e U2
                break # Quebra o ciclo for com contador k
              }else if (tempo.l > 120) # Caso tempo.l seja maior que 120,
              {
                U = a.acidos[19, SG.U] # Cria o objeto U com o valor
correspondente a linha 19 (tempo = 120) e coluna SG.U
                break # Quebra o ciclo for com contador k
              }
            }
          }else if (SG.pre < SG.a.acidos[j]) # Caso o valor de SG.pre seja
menor que SG.a.acidos[j],
          {
            SG.U1 = names(a.acidos)[j] # Cria o objeto SG.U1 com o nome da
coluna j de a.acidos
            SG.U2 = names(a.acidos)[j+1] # Cria o objeto SG.U1 com o nome da
coluna j+1 de a.acidos
            dif.SG = SG.pre - SG.a.acidos[j-1] # Cria o objeto dif.SG com a
diferenca exata entre SG.pre e o valor SG.a.acidos[j-1]
            for (l in 1:nrow(a.acidos)) # Cria um ciclo com contador l, de 1
ate o numero de linhas de a.acidos
            {
              if (tempo.l == a.acidos$Gravity_Time[l]) # Caso tempo.l seja
igual ao tempo l da coluna Gravity_Time de a.acidos,
              {
                U1 = a.acidos[l, SG.U1] # Cria o objeto U1 contendo o valor
de utilizacao de alfa acidos correspondente a linha l e coluna SG.U1
                U2 = a.acidos[l, SG.U2]# Cria o objeto U2 contendo o valor
de utilizacao de alfa acidos correspondente a linha l e coluna SG.U2
                U = round((U1+((U2 - U1)/10)*dif.SG), 3) # Cria o objeto U
com a interpolacao dos valores de U1 e U2
                break # Quebra o ciclo for com contador l
              }else if (tempo.l < a.acidos$Gravity_Time[l]) # Caso tempo.l
seja menor que tempo l da coluna Gravity_Time de a.acidos,
              {
                dif.tempo = tempo.l - a.acidos$Gravity_Time[l-1] # Cria o
objeto dif.tempo com a diferenca exata entre tempo.l e o tempo l-1 da coluna
Gravity_Time de a.acidos
                U1 = a.acidos[l-1, SG.U1] # Cria o objeto U1 contendo o
valor de utilizacao de alfa acidos correspondente a linha l-1 e coluna SG.U1
                U2 = a.acidos[l-1, SG.U2] # Cria o objeto U2 contendo o
valor de utilizacao de alfa acidos correspondente a linha l-1 e coluna SG.U2
                U3 = a.acidos[l, SG.U1] # Cria o objeto U3 contendo o valor
de utilizacao de alfa acidos correspondente a linha l e coluna SG.U1
                U4 = a.acidos[l, SG.U2] # Cria o objeto U4 contendo o valor
de utilizacao de alfa acidos correspondente a linha l e coluna SG.U2
                U.tempo1 = round((U1+((U2 - U1)/10)*dif.SG), 3) # Cria o
objeto U.tempo1 com a interpolacao de U1 e U2
                U.tempo2 = round((U3+((U4 - U3)/10)*dif.SG), 3) # Cria o
objeto U.tempo2 com a interpolacao de U3 e U4
```

```
                U = round((U.tempo1+((U.tempo2 - U.tempo1)/10)*dif.tempo),
3) # Cria o objeto U com a interpolacao de U.SG1 e U.SG2
                break # Quebra o ciclo for com contador l
              }else if (tempo.l > 120) # Caso tempo.l seja maior que 120
              {
                U1 = a.acidos[19, SG.U1] # Cria o objeto U1 contendo o valor
de utilizacao de alfa acidos correspondente a linha 19 (tempo = 120) e
coluna SG.U1
                U2 = a.acidos[19, SG.U2]# Cria o objeto U2 contendo o valor
de utilizacao de alfa acidos correspondente a linha 19 (tempo = 120) e
coluna SG.U2
                U = round((U1+((U2 - U1)/10)*dif.SG), 3) # Cria o objeto U
com a interpolacao de U1 e U2
                break # Quebra o ciclo for com contador l
              }
            }
            break # Quebra o ciclo for com contador j
          }
        }
        IBU.pre[i] = (input$peso[i]*U*input$aa[i]*10)/vol[2] # Atribui o IBU
calculado para o lupulo[i] a IBU.pre[i]
        if (exists("v.adic") == TRUE) # Caso o objeto v.adic tenha sido
criado
        {
          IBU.pos[i] = (input$peso[i]*U*input$aa[i]*10)/v.final # Calcula o
valor de IBU para o lupulo[i] apos o ajuste de OG e atribui a IBU.pos[i]
        }
        if (exists("v.final.ferv") == TRUE) # Caso o objeto v.final.ferv
tenha sido criado
        {
          tempo.l.pos = tempo.l + tempo # Cria o objeto tempo.l.pos com o
tempo de fervura do lupulo[i] + o tempo adicional de fervura
          if (exists("SG.U") == TRUE) # Caso SG.U tenha sido criado,
          {
            for (n in 1:nrow(a.acidos)) # Cria um ciclo com contador n, de 1
ate o numero de linhas de a.acidos
            {
              if (tempo.l.pos == a.acidos$Gravity_Time[n]) # Caso
tempo.l.pos seja igual a a.acidos$Gravity_Time[n],
              {
                U.pos = a.acidos[n, SG.U] # Cria o objeto U.pos contendo o
valor de utilizacao de alfa acidos correspondente ao tempo n e coluna SG.U
                break # Quebra o ciclo for com contador n
              }else if (tempo.l.pos < a.acidos$Gravity_Time[n]) # Caso
tempo.l.pos seja menor que a.acidos$Gravity_Time[n],
              {
                U1.pos = a.acidos[n-1, SG.U] # Cria o objeto U1.pos contendo
o valor de utilizacao de alfa acidos correspondente ao tempo n-1 e coluna
SG.U
                U2.pos = a.acidos[n, SG.U] # Cria o objeto U2.pos contendo o
```

```
valor de utilizacao de alfa acidos correspondente ao tempo n e coluna SG.U
                dif.tempo.pos = tempo.l.pos - a.acidos$Gravity_Time[n-1] #
Cria o objeto dif.tempo com a diferenca exata entre tempo.l.pos e o tempo
n-1 da coluna Gravity_Time de a.acidos
                U.pos = round((U1.pos+((U2.pos - U1.pos)/10)*dif.tempo.pos),
3) # Cria o objeto U.pos com a interpolacao dos valores U1.pos e U2.pos
                break # Quebra o ciclo for com contador n
              }else if (tempo.l.pos > 120) # Caso tempo.l.pos seja maior que
120
              {
                U.pos = a.acidos[19, SG.U] # Cria o objeto U.pos com o valor
correspondente a linha 19 (tempo = 120) e coluna SG.U de a.acidos
                break # Quebra o ciclo for com contador n
              }
            }
          }else # Caso SG.U nao tenha sido criado,
          {
            for (o in 1:nrow(a.acidos)) # Cria um ciclo com contador o, de 1
ate o numero de linhas de a.acidos
            {
              if (tempo.l.pos == a.acidos$Gravity_Time[o]) # Caso
tempo.l.pos seja igual a a.acidos$Gravity_Time[o],
              {
                U1.pos = a.acidos[o, SG.U1] # Cria o objeto U1.pos contendo
o valor de utilizacao de alfa acidos correspondente a linha o e coluna SG.U1
                U2.pos = a.acidos[o, SG.U2]# Cria o objeto U2.pos contendo o
valor de utilizacao de alfa acidos correspondente a linha o e coluna SG.U2
                U.pos = round((U1.pos+((U2.pos - U1.pos)/10)*dif.SG), 3) #
Cria o objeto U.pos com a interpolacao dos valores de U1.pos e U2.pos
                break # Quebra o ciclo for com contador o
              }else if (tempo.l.pos < a.acidos$Gravity_Time[o]) # Caso
tempo.l.pos seja menor que a.acidos$Gravity_Time[o],
              {
                dif.tempo.pos = tempo.l.pos - a.acidos$Gravity_Time[o-1] #
Cria o objeto dif.tempo.pos com a diferenca exata entre tempo.l.pos e o
tempo o-1
                U1.pos = a.acidos[o-1, SG.U1] # Cria o objeto U1.pos
contendo o valor de utilizacao de alfa acidos correspondente ao tempo o-1 e
coluna SG.U1
                U2.pos = a.acidos[o-1, SG.U2] # Cria o objeto U2.pos
contendo o valor de utilizacao de alfa acidos correspondente ao tempo o-1 e
coluna SG.U2
                U3.pos = a.acidos[o, SG.U1] # Cria o objeto U3.pos contendo
o valor de utilizacao de alfa acidos correspondente ao tempo o e coluna
SG.U1
                U4.pos = a.acidos[o, SG.U2] # Cria o objeto U4.pos contendo
o valor de utilizacao de alfa acidos correspondente ao tempo o e coluna
SG.U2
                U.SG1.pos = round((U1.pos+((U2.pos - U1.pos)/10)*dif.SG), 3)
# Cria o objeto U.SG1.pos com a interpolacao de U1.pos e U2.pos
                U.SG2.pos = round((U3.pos+((U4.pos - U3.pos)/10)*dif.SG), 3)
```

```
# Cria o objeto U.SG2.pos com a interpolacao de U3.pos e U4.pos
                U.pos = round((U.SG1.pos+((U.SG2.pos -
U.SG1.pos)/10)*dif.tempo.pos), 3) # Cria o objeto U.pos com a interpolacao
de U.SG1.pos e U.SG2.pos
                break # Quebra o ciclo for com contador o
              }else if (tempo.l.pos > 120) # Caso tempo.l.pos seja maior que
120
              {
                U1.pos = a.acidos[19, SG.U1] # Cria o objeto U1.pos contendo
o valor de utilizacao de alfa acidos correspondente a linha 19 (tempo = 120)
e coluna SG.U1
                U2.pos = a.acidos[19, SG.U2]# Cria o objeto U2.pos contendo
o valor de utilizacao de alfa acidos correspondente a linha 19 (tempo = 120)
e coluna SG.U2
                U.pos = round((U1.pos+((U2.pos - U1.pos)/10)*dif.SG), 3) #
Cria o objeto U.pos com a interpolacao de U1.pos e U2.pos
                break # Quebra o ciclo for com contador o
              }
            }
          }
          IBU.pos[i] = (input$peso[i]*U.pos*input$aa[i]*10)/v.final.ferv #
Calcula o valor de IBU para o lupulo[i] apos o ajuste de OG e atribui a
IBU.pos[i]
        }
      }
      if (dif == 0) # Caso dif seja igual a 0,
      {
        print.ibu = cat(paste("Seu IBU eh igual a ", round(sum(IBU.pre)),
".\n", sep = "")) # Cria o objeto print.ibu contendo a mensagem e o valor de
IBU a serem impressos para o usuario
      }
      if (dif < 0) # Caso dif seja menor que 0,
      {
        print.ibu = cat(paste("Seu IBU atual eh igual a ",
round(sum(IBU.pre)), " e seu IBU apos o ajuste da OG sera igual a ",
round(sum(IBU.pos)), ".\n", sep = "")) # Cria o objeto print.ibu contendo a
mensagem e os valores de IBU a serem impressos para o usuario
      }
      if (dif > 0) # Caso dif seja maior que 0,
      {
        print.ibu = cat(paste("Seu IBU atual eh igual a ",
round(sum(IBU.pre)), ".\n", sep = "")) # Cria o objeto print.ibu com o valor
de IBU total e a mensagem a ser impressa para o usuario
        if (is.na(sum(IBU.pos)) != TRUE) # Caso IBU nao contenha NAs, ou
seja, caso IBU.pos[i] tenha sido calculdo,
        {
          print.ibu.pos = cat(paste("Seu IBU apos o ajuste da OG sera igual
a ", round(sum(IBU.pos)), ".\n", sep = "")) # Cria o objeto print.ibu.pos
com o valor de IBU total apos o ajuste da OG e a mensagem a ser impressa
para o usuario
```

```
          }
        }
      }
    }
}
```