Função: actograma()

```
#Disciplina: Uso da Linguagem R na Análise de Dados Ecológicos.
  #Trabalho Final: Proposta A - "Criando um actograma!".
    #Por Beatriz Foganholi Fernandes - 01 de julho de 2019.
#A distribuição temporal dos ritmos biológicos muitas vezes acontece de forma associada às variações ambientais, como de dia e noite, quente e frio. No entanto, essa relação nem sempre é obrigatória! É possível que um ritmo biológico se mantenha mesmo que todas as variações do ambiente sumam. Por esse motivo, os estudos dos ritmos biológicos, feitos pela chamada Cronobiologia, acontecem associados a diferentes condições ambientais.
#Uma ferramenta muito utilizada para analisar padrões de ritmos de um mesmo indivíduo na Cronobiologia é o actograma, que consiste num empilhamento das atividades registradas dia a dia, com o objetivo de permitir a visualização de padrões na escala diária. Para facilitar ainda mais essa visualização, geralmente são feitos na forma de dupla. Imagine que você tem registros da movimentação de um animal durante 120 horas corridas, e que, se você desenhasse essa movimentação numa fita, ela estaria marcado em rosa como na figura abaixo. Para construir um actograma, basta “cortar” sua fita em pedaços de 24 horas e colocá-las empilhadas em sequência, indo do dia A ao dia E. Por fim, basta duplicar essas fitas e alinhá-las com sua vizinha seguinte para formar um actograma duplo, criando pares AB, BC, CD e DE. como consequência dessa cópia deslocada, a primeira e a última linha do actograma ficam sem par.
#Como esta é uma ferramenta visual muito utilizada na análise dos ritmos biológicos, ao mesmo tempo em que não é disponibilizada por nenhum programa gratuito, minha proposta com a função aqui trazida é gerar actogramas a partir de dados de ritmos biológicos, combinando-as a diferentes possibilidades de regimes de claro-escuro (ou seja, demarcando as durações do que foi dia ou noite para o animal). Além de poder informar quais foram os ciclos ambientais de claro-escuro, esta função permite ao usuário combinar diferentes regimes de ciclos ambientais de temperatura, conhecidos como ciclos quente-frio, representando-os por meio da coloração das barras que representam o ritmo observado.

actograma <- function(data_hora, ritmo, n.linhas, Ei, Ef,
Fi=rep("00:00:00",n.linhas), Ff=rep("00:00:00",n.linhas)) #Cria a função "actograma" com os argumentos especificados no parênteses. Para mais detalhes, verifique a página de ajuda da função!
          {
#Carregando pacotes
          #install.packages("lubridate") #Caso não possua, retire o # e instale o pocote "lubridate" para melhor manipulação dos dados de data-tempo.
          #install.packages("ggplot2")   #Caso não possua, retire o # e instale o pacote "ggplot2" que nos permite criar o actograma combinado ao pacote "gridExtra".
          #install.packages("gridExtra") #Caso não possua, retire o # e instale o pacote "gridExtra" que nos permite criar o actograma sem limite de
```

```
número de dias.
          require(lubridate)            #Carrega o pacote "lubridate"
instalado.
          require(hms)                  #Carrega o pacote base 'hms'
referente à classe de hora, minuto e segundo. Para manipulação de valores de
hora fornecidos pelo usuário.
          require(ggplot2)              #Carrega o pacote "ggplot2"
instalado.
          require(gridExtra)            #Carrega o pacote "gridExtra"
instalado.
#- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
#Verificação de parâmetros
          #Verificação 1: Igualdade de tamanho entre data_hora e ritmo.
          if(length(data_hora)!=length(ritmo))
#Se hora e ritmo possuem comprimentos diferentes:
            {stop("'data_hora' e 'ritmo' possuem comprimentos distintos.")}
#Para e retorna ”'data_hora' e 'ritmo' possuem comprimentos distintos.”.

          #Verificação 2: Transformação correta do 'Time stamp'.
          data_hora <- mdy_hms(data_hora,tz="") #Transforma o objeto
data_hora no formato mês-dia-ano + hora:minuto:segundo + fuso horário. Se
houver valores fora do formato, são transformados em NA.
          if(sum(is.na(data_hora))>0)           #Se um ou mais valores no
objeto data_hora forem transformados em NA:
            {stop("'data_hora' contém dados fora de formato.")} #Para e
retorna ”'data_hora' contém dados fora de formato.”.

          #Verificação 3: Presença de valores negativos em ritmo.
          if(sum(ritmo<0)>0) #Se os valores de ritmo contiverem negativo(s):
            {message("Cuidado, seu dados de ritmo possuem valor(es)
negativo(s). Você talvez queira conferir.")} #Avisa ao usuário "Cuidado, seu
dados de ritmo possuem valor(es) negativo(s). Você talvez queira conferir.".

          #Verificação 4: Valor adequado de n.linhas fornecido.
          n.linhas <- as.integer(n.linhas) #Transforma n.linhas em número
inteiro, possibilitando arredondamentos.
          if(n.linhas <= 0)               #Se n.linhas for negativo:
            {stop("'n.dias' deve ser número inteiro e positivo.")} #Para e
retorna "'n.linhas' deve ser número inteiro e positivo.".

          #Verificação 5: Transformação correta dos horários fornecidos para
ciclos de claro-escuro (obrigatório) e correspondência com n.linhas
escolhido pelo usuário.
          if(length(Ei)!=length(Ef))                                #Se Ei e
Ef possuem comprimentos diferentes:
            {stop("'Ei' e 'Ef' possuem comprimentos distintos.")} #Para e
retorna ”'Ei' e 'Ef' possuem comprimentos distintos.”.
          E.inicio <- as.hms(Ei)
#Transforma o objeto Ei no formato hora:minuto:segundo e guarda no objeto
E.inicio. Se houver valores fora do permitido, por exemplo uma anotação
```

```
"25:00:00", são transformados em NA.
          E.fim <- as.hms(Ef)
#Transforma o objeto Ef no formato hora:minuto:segundo e guarda no objeto
E.fim. Se houver valores fora do permitido, por exemplo uma anotação
"25:00:00", são transformados em NA.
          if(sum(is.na(E.inicio))>0 | sum(is.na(E.fim))>0)        #Se um ou
mais valores nos objetos E.inicio/E.fim forem transformados em NA:
            {stop("Regime(s) de escuro fora de formato.")}        #Para e
retorna ”Regime(s) de escuro fora de formato.”.
          if(length(Ei)<n.linhas & length(Ef)<n.linhas)           #Se Ei e
Ef possuirem comprimento menor que n.linhas escolhido:
            {stop("'Ei' ou 'Ef' possuem comprimentos menores que o número de
linhas escolhido em n.linhas.")} #Para e retorna ”'Ei' ou 'Ef' possuem
comprimentos menores que o número de linhas escolhido em n.linhas.”.
          #Verificação 6: Transformação correta dos horários fornecidos para
ciclos de quente-frio (opcional) e correspondência com n.linhas escolhido
pelo usuário.
          if(length(Fi)!=length(Ff))                              #Se Fi e
Ff foram especificados e possuem comprimentos diferentes:
            {stop("'Fi' e 'Ff' possuem comprimentos distintos.")} #Para e
retorna ”'Fi' e 'Ff' possuem comprimentos distintos.”.
          if(missing(Fi)==T & missing(Ff)==T)                     #Caso Fi e
Ff não sejam especificados pelo usuário:
            {                                                     #Indica as
operações a serem seguidas.
            Fi.c <- rep("00:00:00",n.linhas)                      #Cria Fi.c
com hora zero repetida pelo número de linhas escolhido.
            Ff.c <- rep("00:00:00",n.linhas)                      #Cria Ff.c
com hora zero repetida pelo número de linhas escolhido.
            F.inicio <- as.hms(Fi.c)
#Transforma o objeto Fi.c no formato hora:minuto:segundo.
            F.fim <- as.hms(Ff.c)
#Transforma o objeto Ff.c no formato hora:minuto:segundo.
            } else                                                #Finaliza
operações indicadas. Do contrário (Fi e Ff especificados):
            {                                                     #Indica as
operações a serem seguidas.
            F.inicio <- as.hms(Fi)
#Transforma o objeto Fi informado no formato hora:minuto:segundo e guarda no
objeto F.inicio. Se houver valores fora do permitido, por exemplo uma
anotação "25:00:00", são transformados em NA.
            F.fim <- as.hms(Ff)
#Transforma o objeto Ff  informado no formato hora:minuto:segundo e guarda
no objeto F.fim. Se houver valores fora do permitido, por exemplo uma
anotação "25:00:00", são transformados em NA.
            if(sum(is.na(F.inicio))>0 | sum(is.na(F.fim))>0)      #Se um ou
mais valores nos objetos F.inicio ou F.fim forem transformados em NA:
            {stop("Regime(s) de frio fora de formato.")}          #Para e
retorna ”Regime(s) de frio fora de formato em Fi.”.
            if(length(Fi)<n.linhas | length(Ff)<n.linhas)         #Se Fi e
Ff possuirem comprimento menor que n.linhas escolhido:
```

```
            {stop("'Fi' ou 'Ff' possuem comprimentos menores que o número de
linhas escolhido em n.linhas.")} #Para e retorna "'Fi' ou 'Ff' possuem
comprimentos menores que o número de linhas escolhido em n.linhas.".
            }                                                    #Finaliza
operações indicadas.
#- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
#Agora que todos os parâmetros foram verificados, podemos começar!
#Preparando dados para plotagem do actograma:
          dados <- data.frame(date(data_hora[1:n.linhas]),
as.hms(data_hora[1:n.linhas]), ritmo[1:n.linhas], E.inicio[1:n.linhas],
E.fim[1:n.linhas], F.inicio[1:n.linhas], F.fim[1:n.linhas]) #Cria o
dataframe 'dados' combinando a data extraída de data_hora, a hora extraída
de data_hora, ritmo, E.inicio, E.fim, F.inicio e F.fim, todos indo do
primeiro valor àquele na posição de valor igual a n.linhas. Com isso, o
dataframe dados, que será a base para alimentar nosso actograma, pode ter
número de linhas (e, portanto, de observações consideradas) variável.
          colnames(dados) <- c("Data","Hora","Ritmo","Escuro_inicio",
"Escuro_fim", "Frio_inicio", "Frio_fim") #Altera os nomes das colunas de
'dados' para "Data","Hora","Ritmo","Escuro_início", "Escuro_fim",
"Frio_início" e "Frio_fim".
          dias_existentes <- unique(dados$Data) #Cria o objeto
"dias_existentes" que contém todas as datas de dados, compiladas apenas uma
vez.
          dados$Temperatura <- rep(NA,nrow(dados))
#Adiciona uma coluna Temperatura aos dados, com um NA para cada linha.
          for(a in 1:nrow(dados))
#Cria um ciclo com contador a que vai de 1 ao número de linhas em dados.
          {
#Indica início do ciclo.
          if(dados[a,6]<dados[a,7])                                  #Se
o horário de Frio_inicio da linha for menor que Frio_fim (na escala de 24
horas), isso quer dizer que a duração do frio não inclui a meia noite. Nesse
tipo de padrão, queremos demarcar uma única "faixa" de frio dentro do dia de
24 horas, que está no intervalo Frio_inicio <= x <= Frio_fim. Então:
                {
#Indica as operações a serem seguidas.
                if(dados[a,2] < dados[a,6])                          #Se
a Hora da linha for menor que Frio_inicio:
                  {dados[a,8]<- paste0("quente")}
#Escreve 'quente' na coluna Temperatura da linha correspondente.
                if(dados[a,2] >= dados[a,6] & dados[a,2] <= dados[a,7]) #Se
Hora estiver entre Frio_inicio e Frio_fim:
                  {dados[a,8] <- paste0("frio")}
#Escreve 'frio' na coluna Temperatura da linha correspondente.
                if(dados[a,2] > dados[a,7])                          #Se
Hora for maior que Frio_fim:
                  {dados[a,8] <- paste0("quente")}
#Escreve 'quente' na coluna Temperatura da linha correspondente.
                }
#Finaliza as operações indicadas.
```

```
          if(dados[a,6]>dados[a,7])                                        #Se
o horário de Frio_inicio da linha for maior que Frio_fim (na escala de 24
horas), isso quer dizer que a duração do frio inclui a meia noite (que é o
ponto de quebra do eixo x no actograma). Nesse tipo de padrão, queremos
demarcar duas "faixas" de frio dentro do dia de 24 horas, que estão nos
intervalos Frio_inicio <= x <= 23:59:59 e meia-noite <= x <= Frio_fim.
Então:
               {
#Indica as operações a serem seguidas.
               if(dados[a,2] < dados[a,7])                                 #Se
a Hora da linha for menor que Frio_fim:
                 {dados[a,8]<- paste0("frio")}
#Escreve 'frio' na coluna Temperatura da linha correspondente.
               if(dados[a,2] >= dados[a,7] & dados[a,2] <= dados[a,6])  #Se
Hora estiver entre Frio_fim e Frio_inicio:
                 {dados[a,8] <- paste0("quente")}
#Escreve 'quente' na coluna Temperatura da linha correspondente.
               if(dados[a,2] > dados[a,6])                                 #Se
Hora for maior que Frio_inicio:
                 {dados[a,8] <- paste0("frio")}
#Escreve 'frio' na coluna Temperatura da linha correspondente.
               }
#Finaliza as operações indicadas.
          if(dados[a,6]==dados[a,7])                                       #Por
fim, se o horário de Frio_inicio da linha for igual a Frio_fim, como é o
caso do default para quando o usuário não fornece Fi nem Ff, ou também
possível para o caso de o usuário querer fornecer tanto regime quente-frio
quanto de temperatura constante no mesmo actograma:
                {dados[a,8]<- paste0("constante")}
#Escreve 'constante' na coluna Temperatura da linha correspondente.
          }
#Finaliza o ciclo.
          dados$Temperatura <- as.factor(dados$Temperatura)
#Transforma as inserções feitas em Temperatura em fatores. Isso é importante
para poder colorir o actograma de aoordo com os regimes de temperatura
informados.
#- - -
#Preparando o actograma:
          #Definir o layout e valores máximos de eixos para os actogramas
          linhas.plot <- seq(from=1, to=(length(dias_existentes)+1)) #Cria
sequência indo de 1 ao número de dias total do actograma. Deve haver um a
mais para ser o espaço em branco típico de actogramas (ora na primeira linha
e ora na última).
          mat.layout <- as.matrix(linhas.plot)        #Transforma a
sequência em matriz, para ser usada mais adiante em arrangeGrobs(). A partir
daqui vou utilizar as funções do pacote gridExtra para plotar os gráficos de
cada dia de acordo com a sequência contida em mat.layout. Isso, pois a
função layout() base do R possui um limite de plotagem de no máximo 200
linhas que pode ser inconveniente para o usuário.
          xmax = as.numeric(as.hms("23:59:59"))       #Define um objeto
'xmax' equivalente a 23h:59m:59s de forma numérica para ser operável em
```

```
relação ao eixo x do actograma.
          ymax = max(dados$Ritmo)                      #Define um objeto
'ymax' equivalente valor máximo de Ritmo presente no dataframe dados. Assim,
todas as linhas do actograma terão a mesma altura de acordo com o conjunto
dos dados.
          ymax = as.numeric(ymax)                      #Transforma ymax em
objeto númérico. Pré requisito para scale_y_continuous().
#- - -
          #Criar os plots
                 #Vamos formar um actograma que contém os ritmo plotados em
barras, e os horários de claro/escuro demarcados com retângulo cinza ao
fundo. As diferentes possibilidades de ciclo de temperatura (cíclica ou
constante) serão representadas pelas cores das barras com base no fator
contido na coluna de Temperatura.
             plots1 <- rep(NA,(length(dias_existentes)+1))  #Para a 1ª
"coluna" do actograma, cria o objeto plots1, com (número de
dias_existentes+1) vezes "NA".
             plots1 <- as.list(plots1)              #Transforma plots1 em
lista, para ser usada mais adiante em arrangeGrobs().
             plots1[[1]] <- ggplot() + theme_void() #Cria um gráfico vazio
na posição 1 de plots1, típico de actogramas como o que faremos.
             for(d in 2:(length(dias_existentes)+1))#Cria ciclo com contador
d indo de 2 a dias_existentes+1, para preencher os gráficos abaixo do vazio
criado.
             {                                      #Indica início do ciclo.
             assign(paste0("df",(d-1)),
dados[dados$Data==dias_existentes[d-1],]) #Cria um novo dataframe contendo
apenas os dados da data contida em dias_existentes na posição do contador,
de forma a separar o conjunto de dados por data em novos dataframes
numerados df1, df2, df3 etc. Aqui, queremos que df1 contenha valores da
primeira data de dias_existentes, por isso, utilizamos (d-1).
             if(get(paste0("df",(d-1)))[1,4]<get(paste0("df",(d-1)))[1,5])
#Caso a fase de escuro declarada não contenha meia-noite (quando
Escuro_inicio do df é menor que Escuro_fim do df), cria apenas 1 retângulo
cinza ao fundo. Como Escuro_inicio e Escuro_fim serão iguais para todas as
linhas do dataframe daquele dia, podemos utilizar apenas o valor da primeira
linha [1,x].
                      {
#Indica as operações a serem seguidas.
                      plots1[[d]] <- ggplot(get(paste0("df",(d-1))),
#Cria um plot a partir de df do número do contador-1, armazenando em plots1.
                                            aes(x =Hora, y =Ritmo)
#Utilizando a coluna Hora como x e Ritmo como y.
                                            ) +
#Fecha a função ggplot() e sinaliza o início de uma próxima.
                                     geom_rect(aes(xmin = Escuro_inicio[1],
#Adiciona um retângulo que começa em x no valor de Escuro_inicio.
                                                   xmax = Escuro_fim[1],
#Termina em x no valor de Escuro_fim.
                                                   ymin = -Inf,
```

```
#"Começa" em -infinito em y.
                                                    ymax = Inf),
#"Termina" em infinito em y.
                                                    fill = "grey",
#Preenchido com a cor cinza.
                                                    alpha = 0.5) +
#Com opacidade de 50%.
                                    geom_bar(aes(fill=Temperatura),
#Adiciona gráfico de barras, cor de acordo com Temperatura.
                                             stat="identity",
#Com altura das barras equivalentes aos seus valores em y.
                                             width=1000) +
#Largura das barras = 1000 segundos.
                                    scale_fill_manual(values = c("quente" =
"red3",    #dados quentes em Temperatura serão vermelhos.
                                                                 "frio" =
"blue4",         #dados frios em Temperatura serão azuis.
                                                                 "constante" =
"black")) + #dados constantes em Temperatura serão pretos.
                                    theme_classic() + #Utilizando o tema
pronto classic, que tem um padrão de fundo branco box em L.
                                    theme(plot.margin= unit(c(-2,0,0,0),
"pt")) + #Estabelece margens superior, direita, inferior e esquerda, nessa
ordem, em pontos.
                                    theme(axis.title=element_blank())  +
#Remove título dos eixos.
                                    theme(axis.text=element_blank())  +
#Remove anotações dos eixos.
                                    theme(axis.ticks=element_blank()) +
#Remove marcações dos eixos.
                                    theme(legend.position="none") +
#Remove legenda.
                                    scale_x_continuous(limits =
c(-500,(xmax+500)), #Estabelece limite do eixo x indo de -500
segundos(metade da largura de uma barra, para que valores existentes às
00:00:00 possam aparecer) a xmax+500(pelo mesmo motivo).
                                                       expand = c(0, 0)) +
#Inicia o gráfico na origem zero-zero.
                                    scale_y_continuous(limits = c(0,ymax),
#Estabelece limite do eixo y indo de 0 a ymax.
                                                       expand = c(0, 0))
#Inicia o gráfico na origem zero-zero.
                      }
#Finaliza operações indicadas.
          if(get(paste0("df",(d-1)))[1,4]>get(paste0("df",(d-1)))[1,5])
#Caso a fase de escuro no df do contador (d-1) contenha meia-noite (quando
Escuro_inicio é maior que Escuro_fim), cria 2 retângulos cinzas ao fundo.
                      {
#Indica as operações a serem seguidas.
                      plots1[[d]] <- ggplot(get(paste0("df",(d-1))),
#Cria um plot a partir de df do número do contador-1, armazenando em plots1.
```

```
                                          aes(x =Hora, y =Ritmo)
#Utilizando a coluna Hora como x e Ritmo como y.
                                          ) +
#Fecha a função ggplot() e sinaliza o início de uma próxima.
                                    geom_rect(aes(xmin = -500,
#Adiciona um retângulo que começa em x -500 segundos.
                                                  xmax = Escuro_fim[1],
#Termina em x no valor de E.fim.
                                                  ymin = -Inf,
#"Começa" em -infinito em y.
                                                  ymax = Inf),
#"Termina" em infinito em y
                                                fill = "grey",
#Preenchido com a cor cinza.
                                                alpha = 0.5) +
#Com opacidade de 50%.
                                    geom_rect(aes(xmin = Escuro_inicio[1],
#Adiciona um retângulo que começa em x no valor de Escuro_inicio.
                                                  xmax = xmax,
#Termina em x no valor 23:59:59.
                                                  ymin = -Inf,
#"Começa" em -infinito em y.
                                                  ymax = Inf),
#"Termina" em infinito em y.
                                                fill = "grey",
#Preenchido com a cor cinza.
                                                alpha = 0.5) +
#Com opacidade de 50%.
                                    geom_bar(aes(fill=Temperatura),
#Adiciona gráfico de barras, cor de acordo com Temperatura.
                                             stat="identity",
#Com altura das barras equivalentes aos seus valores em y.
                                             width=1000) +
#Largura das barras = 1000 segundos.
                                    scale_fill_manual(values = c("quente" =
"red3",    #dados quentes em Temperatura serão vermelhos.
                                                                 "frio" =
"blue4",         #dados frios em Temperatura serão azuis.
                                                                 "constante" =
"black")) + #dados constantes em Temperatura serão pretos.
                                    theme_classic() + #Utilizando o tema
pronto classic, que tem um padrão de fundo branco box em L.
                                    theme(plot.margin= unit(c(-2,0,0,0),
"pt")) + #Estabelece margens superior, direita, inferior e esquerda em
pontos.
                                    theme(axis.title=element_blank())  +
#Remove título dos eixos.
                                    theme(axis.text=element_blank())  +
#Remove anotações dos eixos.
                                    theme(axis.ticks=element_blank()) +
```

```
#Remove marcações dos eixos.
                                    theme(legend.position="none") +
#Remove legenda.
                                    scale_x_continuous(limits =
c(-500,(xmax+500)), #Estabelece limite do eixo x indo de -500
segundos(metade da largura de uma barra, para que valores existentes às
00:00:00 possam aparecer) a xmax+500 (pelo mesmo motivo).
                                                       expand = c(0, 0)) +
#Inicia o gráfico na origem zero-zero.
                                    scale_y_continuous(limits = c(0,ymax),
#Estabelece limite do eixo y indo de 0 a ymax.
                                                       expand = c(0, 0))
#Inicia o gráfico na origem zero-zero.
                     }
#Finaliza as operações indicadas.
          if(get(paste0("df",(d-1)))[1,4]==get(paste0("df",(d-1)))[1,5])
#Caso o regime de claro/escuro do df seja de claro constante (quando
Escuro_inicio é igual Escuro_fim), não cria retângulo cinza ao fundo.
                     {
#Indica as operações a serem seguidas.
                     plots1[[d]] <- ggplot(get(paste0("df",(d-1))),
#A partir de df do número do contador-1, armazenando em plots1. Desejamos
que o segundo gráfico de plots1 seja referente ao df1, e assim por diante,
por isso utilizamos (d-1).
                                           aes(x =Hora, y =Ritmo)
#Utilizando a coluna Hora como x e Ritmo como y.
                                           ) +
#Fecha a função ggplot() e sinaliza o início de uma próxima.
                                    geom_bar(aes(fill=Temperatura),
#Adiciona gráfico de barras, cor de acordo com Temperatura.
                                             stat="identity",
#Com altura das barras equivalentes aos seus valores em y.
                                             width=1000) +
#Largura das barras = 1000 segundos.
                                    scale_fill_manual(values = c("quente" =
"red3",    #dados quentes em Temperatura serão vermelhos.
                                                                 "frio" =
"blue4",         #dados frios em Temperatura serão azuis.
                                                                 "constante" =
"black")) + #dados constantes em Temperatura serão pretos.
                                    theme_classic() + #Utilizando o tema
pronto classic, que tem um padrão de fundo branco box em L.
                                    theme(plot.margin= unit(c(-2,0,0,0),
"pt")) + #Estabelece margens superior, direita, inferior e esquerda em
pontos.
                                    theme(axis.title=element_blank())  +
#Remove título dos eixos.
                                    theme(axis.text=element_blank())  +
#Remove anotações dos eixos.
                                    theme(axis.ticks=element_blank()) +
#Remove marcações dos eixos.
```

```
                                    theme(legend.position="none") +
#Remove legenda.
                                    scale_x_continuous(limits =
c(-500,(xmax+500)), #Estabelece limite do eixo x indo de -500
segundos(metade da largura de uma barra, para que valores existentes às
00:00:00 possam aparecer) a xmax+500 (pelo mesmo motivo).
                                                       expand = c(0, 0)) +
#Inicia o gráfico na origem zero-zero.
                                    scale_y_continuous(limits = c(0,ymax),
#Estabelece limite do eixo y indo de 0 a ymax.
                                                       expand = c(0, 0))
#Inicia o gráfico na origem zero-zero.
                      }
#Finaliza as operações indicadas.
           }
#Finaliza o ciclo (1ª coluna do actograma).
          plots2 <- rep(NA,(length(dias_existentes)+1)) #Para a 2ª "coluna"
do actograma, cria o objeto plots2, com (número de dias_existentes+1) vezes
"NA".
          plots2 <- as.list(plots2)                    #Transforma plots2
em lista, para ser usada mais adiante em arrangeGrobs().
          plots2[[(length(dias_existentes)+1)]] <- ggplot() + theme_void()
#Cria um gráfico vazio na última posição de plots2, típico de actogramas
como o que faremos.
          for(d in 1:(length(dias_existentes)))        #Para contador d
indo de 1 ao número de dias existentes (Como nesse caso o gráfico vazio é o
último da coluna, podemos seguir a ordem de d para recuperar os dataframes
df1, df2, df3 etc já criados.).
          {                                            #Indica início do
ciclo.
           if(get(paste0("df",d))[1,4]<get(paste0("df",d))[1,5])
#Caso a fase de escuro declarada no df não contenha meia-noite (quando
Escuro_inicio é menor que Escuro_fim), cria apenas 1 retângulo cinza ao
fundo.
                      {
#Indica as operações a serem seguidas.
                      plots2[[d]] <- ggplot(get(paste0("df",d)),
#Cria um plot do df do número do contador, armazenando em plots2.
                                            aes(x =Hora, y =Ritmo)
#Utilizando a coluna Hora como x e Ritmo como y.
                                            ) +
#Fecha a função ggplot() e sinaliza o início de uma próxima.
                                    geom_rect(aes(xmin = Escuro_inicio[1],
#Adiciona um retângulo que começa em x no valor de Escuro_inicio.
                                                  xmax = Escuro_fim[1],
#Termina em x no valor de Escuro_fim.
                                                  ymin = -Inf,
#"Começa" em -infinito em y.
                                                  ymax = Inf),
#"Termina" em infinito em y.
```

```
                                                    fill = "grey",
#Preenchido com a cor cinza.
                                                    alpha = 0.5) +
#Com opacidade de 50%.
                                    geom_bar(aes(fill=Temperatura),
#Adiciona gráfico de barras, cor de acordo com Temperatura.
                                             stat="identity",
#Com altura das barras equivalentes aos seus valores em y.
                                             width=1000) +
#Largura das barras = 1000 segundos.
                                    scale_fill_manual(values = c("quente" =
"red3",    #dados quentes em Temperatura serão vermelhos.
                                                                 "frio" =
"blue4",         #dados frios em Temperatura serão azuis.
                                                                 "constante" =
"black")) + #dados constantes em Temperatura serão pretos.
                                    theme_classic() + #Utilizando o tema
pronto classic, que tem um padrão de fundo branco box em L.
                                    theme(plot.margin= unit(c(-2,0,0,0),
"pt")) + #Estabelece margens superior, direita, inferior e esquerda em
pontos.
                                    theme(axis.title=element_blank())  +
#Remove título dos eixos.
                                    theme(axis.text=element_blank())  +
#Remove anotações dos eixos.
                                    theme(axis.ticks=element_blank()) +
#Remove marcações dos eixos.
                                    theme(legend.position="none") +
#Remove legenda.
                                    scale_x_continuous(limits =
c(-500,(xmax+500)), #Estabelece limite do eixo x indo de -500
segundos(metade da largura de uma barra, para que valores existentes às
00:00:00 possam aparecer) a xmax+500 (pelo mesmo motivo).
                                                       expand = c(0, 0)) +
#Inicia o gráfico na origem zero-zero.
                                    scale_y_continuous(limits = c(0,ymax),
#Estabelece limite do eixo y indo de 0 a ymax.
                                                       expand = c(0, 0))
#Inicia o gráfico na origem zero-zero.
                     }
#Finaliza operações indicadas.
          if(get(paste0("df",d))[1,4]>get(paste0("df",d))[1,5]) #Caso a fase
de escuro declarada contenha meia-noite (quando Escuro_inicio é maior que
Escuro_fim), cria 2 retângulos cinzas ao fundo.
                     {
#Indica as operações a serem seguidas.
                     plots2[[d]] <- ggplot(get(paste0("df",d)),
#Cria um plot de df do número do contador, armazenando em plots2.
                                           aes(x =Hora, y =Ritmo)
#Utilizando a coluna Hora como x e Ritmo como y.
                                           ) +
```

```
#Fecha a função ggplot() e sinaliza o início de uma próxima.
                                    geom_rect(aes(xmin = -500,
#Adiciona um retângulo que começa em x -500 segundos.
                                                  xmax = Escuro_fim[1],
#Termina em x no valor de Escuro_fim.
                                                  ymin = -Inf,
#"Começa" em -infinito em y.
                                                  ymax = Inf),
#"Termina" em infinito em y
                                                fill = "grey",
#Preenchido com a cor cinza.
                                                alpha = 0.5) +
#Com opacidade de 50%.
                                    geom_rect(aes(xmin = Escuro_inicio[1],
#Adiciona um retângulo que começa em x no valor de Escuro_inicio.
                                                  xmax = xmax,
#Termina em x no valor 23:59:59.
                                                  ymin = -Inf,
#"Começa" em -infinito em y.
                                                  ymax = Inf),
#"Termina" em infinito em y.
                                                fill = "grey",
#Preenchido com a cor cinza.
                                                alpha = 0.5) +
#Com opacidade de 50%.
                                    geom_bar(aes(fill=Temperatura),
#Adiciona gráfico de barras, cor de acordo com Temperatura.
                                             stat="identity",
#Com altura das barras equivalentes aos seus valores em y.
                                             width=1000) +
#Largura das barras = 1000 segundos.
                                    scale_fill_manual(values = c("quente" =
"red3",    #dados quentes em Temperatura serão vermelhos.
                                                                 "frio" =
"blue4",         #dados frios em Temperatura serão azuis.
                                                                 "constante" =
"black")) + #dados constantes em Temperatura serão pretos.
                                    theme_classic() + #Utilizando o tema
pronto classic, que tem um padrão de fundo branco box em L.
                                    theme(plot.margin= unit(c(-2,0,0,0),
"pt")) + #Estabelece margens superior, direita, inferior e esquerda em
pontos.
                                    theme(axis.title=element_blank())  +
#Remove título dos eixos.
                                    theme(axis.text=element_blank())  +
#Remove anotações dos eixos.
                                    theme(axis.ticks=element_blank()) +
#Remove marcações dos eixos.
                                    theme(legend.position="none") +
#Remove legenda.
```

```
                                    scale_x_continuous(limits =
c(-500,(xmax+500)), #Estabelece limite do eixo x indo de -500
segundos(metade da largura de uma barra, para que valores existentes às
00:00:00 possam aparecer) a xmax+500 (pelo mesmo motivo).
                                                       expand = c(0, 0)) +
#Inicia o gráfico na origem zero-zero.
                                    scale_y_continuous(limits = c(0,ymax),
#Estabelece limite do eixo y indo de 0 a ymax.
                                                       expand = c(0, 0))
#Inicia o gráfico na origem zero-zero.
                      }
#Finaliza as operações indicadas.
           if(get(paste0("df",d))[1,4]==get(paste0("df",d))[1,5]) #Caso não
haja fase de escuro declarada (quando Escuro_inicio é igual Escuro_fim), não
cria retângulo cinza ao fundo e entende-se uma sitação de claro constante.
                      {
#Indica as operações a serem seguidas.
                      plots2[[d]] <- ggplot(get(paste0("df",d)),
#Cria um plot de df do número do contador, armazenando em plots2.
                                            aes(x =Hora, y =Ritmo)
#Utilizando a coluna Hora como x e Ritmo como y.
                                            ) +
#Fecha a função ggplot() e sinaliza o início de uma próxima.
                                    geom_bar(aes(fill=Temperatura),
#Adiciona gráfico de barras, cor de acordo com Temperatura.
                                              stat="identity",
#Com altura das barras equivalentes aos seus valores em y.
                                              width=1000) +
#Largura das barras = 1000 segundos.
                                    scale_fill_manual(values = c("quente" =
"red3",    #dados quentes em Temperatura serão vermelhos.
                                                            "frio" =
"blue4",         #dados frios em Temperatura serão azuis.
                                                            "constante" =
"black")) + #dados constantes em Temperatura serão pretos.
                                    theme_classic() + #Utilizando o tema
pronto classic, que tem um padrão de fundo branco box em L.
                                    theme(plot.margin= unit(c(-2,0,0,0),
"pt")) + #Estabelece margens superior, direita, inferior e esquerda em
pontos.
                                    theme(axis.title=element_blank())  +
#Remove título dos eixos.
                                    theme(axis.text=element_blank())  +
#Remove anotações dos eixos.
                                    theme(axis.ticks=element_blank()) +
#Remove marcações dos eixos.
                                    theme(legend.position="none") +
#Remove legenda.
                                    scale_x_continuous(limits =
c(-500,(xmax+500)), #Estabelece limite do eixo x indo de -500
segundos(metade da largura de uma barra, para que valores existentes às
```

```
00:00:00 possam aparecer) a xmax+500 (pelo mesmo motivo).
                                                        expand = c(0, 0)) +
#Inicia o gráfico na origem zero-zero.
                                    scale_y_continuous(limits = c(0,ymax),
#Estabelece limite do eixo y indo de 0 a ymax.
                                                        expand = c(0, 0))
#Inicia o gráfico na origem zero-zero.
                      }
#Finaliza as operações indicadas.
           }
#Indica fim do ciclo (2ª coluna do actograma).
#- - -
                  #Adicionando escalas externas e legenda
                  #Régua lateral em dias:
                  x1 <- rep(1,(length(dias_existentes)+2))
#Cria objeto x1 com valores de 1 para número de dias existentes + 2 (para as
pontas). Esses valores serão a base para criar os traços horizontais da
régua lateral.
                  y1 <-
rev(seq(0,(ymax*(length(dias_existentes)+1)),by=ymax)) #Cria y1 com
sequência reversa de 0 até ymax*(length(dias_existentes)+1), ou seja, de 0
até a altura máxima dos gráficos para cada dia empilhados, intervalados pelo
valor de ymax.
                  regua1 <- data.frame(x1,y1)
#Une x1 e y1 em um dataframe regua1.
                  regua.dia <- ggplot(regua1,aes(x=x1,y=y1)) +
#Cria o gráfico regua.dia, com base em x1 e y1 de regua1.
                                geom_segment(x=0.7,xend=1,
#Adiciona linhas com 0.3 de tamanho, no canto direito.
                                             y=y1,yend=y1,
#Horizontais, com mesmo y cada.
                                             linetype="solid",
#Do tipo linha sólida.
                                             color = "black",
#Na cor preta.
                                             size=0.5)+
#Espessura 0.5.
                                geom_vline(xintercept = 0.7,
#Adiciona segmento vertical no intercepto 0.7 em x.
                                           linetype="solid",
#Do tipo linha sólida.
                                           color = "black",
#Na cor preta.
                                           size=0.5) +
#Espessura 0.5.
                                theme_void() +
#Utilizando o tema pronto void, que tem um padrão vazio.
                                theme(plot.margin=unit(c(-3,0,3,0), "pt"))+
#Estabelece margens superior, direita, inferior e esquerda em pontos.
                                theme(axis.title=element_blank())  +
```

```
#Remove título dos eixos.
                              theme(axis.text=element_blank())  +
#Remove anotações dos eixos.
                              theme(axis.ticks=element_blank()) +
#Remove marcações dos eixos.
                              scale_x_continuous(limits = c(0,1),
#Estabelece limite do eixo x indo de 0 a 1.
                                                 expand = c(0,0)) +
#Inicia o gráfico na origem zero-zero.
                              scale_y_continuous(limits =
c(0,(ymax*(length(dias_existentes)+1))), #Estabelece limite do eixo y indo
de 0 a altura máxima dos gráficos empilhados.
                                                 expand = c(0, 0)) +
#Inicia o gráfico na origem zero-zero.
                              annotate("text", x = 0.15, y =
((ymax*(length(dias_existentes)+1))/2), #Adiciona texto na posição x=0.15 e
y = metade da altura máxima dos gráficos empilhados.
                                       label = "Dias",
#Escreve a palavra 'Dias'.
                                       angle = 90,
#Rotacionada a 90°.
                                       size = 4,
#Com tamanho 4.
                                       family = "serif")
#Fonte 'serif'.
                  for(d in (length(dias_existentes):0))
#Cria ciclo para escrever o número do dia ao lado de cada gráfico, indo do
maior a zero.
                     {
#Inicia o ciclo.
                     regua.dia <- regua.dia +
#Adiciona ao gráfico regua.dia.
                                    annotate("text", x = 0.55, y =
((d*ymax)+(ymax/2)), #Texto em x=0.55 e y= meio da altura do dia na pilha
dos gráficos, que é seu ponto + metade de sua altura máxima.
                                    label = paste0(length(dias_existentes)-
d),#Escreve o número do dia, que é obtido pelo número de dias existentes - o
contador. Isso gera um resultado de escrita 0, 1, 2, 3(...) de baixo para
cima, pois o contador começa escrevendo no alto.
                                    size = 3,
#Com tamanho 3.
                                    family="serif")
#Fonte 'serif'.
                     }
#Finaliza o ciclo.
                  #Régua inferior em horas:
                  x2 <- seq(0,48)
#Cria objeto x2 com valores de 0 a 48.
                  y2 <- rep(1,49)
#Cria objeto y2 com 49 vezes de 1. Esses valores serão a base para criar os
traços verticais da régua inferior.
```

```
                  regua2 <- data.frame(x2,y2)
#Une x2 e y2 em um dataframe regua2.
                  regua.hora <- ggplot(regua2,aes(x=x2,y=y2)) +
#Cria o gráfico regua.hora, com base em x2 e y2 de regua2.
                                geom_segment(x=x2,xend=x2,y=0.7,yend=0.85,
#Adiciona linhas horizontais de altura 0.15.
                                                 linetype="solid", color =
"black", size=0.5)+ #Sólidas, pretas e com espessura 0.5.
                                geom_segment(x=0,xend=0,y=0.85,yend=1,
#Adiciona linha acima do anterior na posição x=0.
                                                 linetype="solid", color =
"black", size=0.5)+ #Sólida, preta e com espessura 0.5.
                                geom_segment(x=24,xend=24,y=0.85,yend=1,
#Adiciona linha acima do anterior na posição x=24.
                                                 linetype="solid", color =
"black", size=0.5)+ #Sólida, preta e com espessura 0.5.
                                geom_segment(x=48,xend=48,y=0.85,yend=1,
#Adiciona linha acima do anterior na posição x=48.
                                                 linetype="solid", color =
"black", size=0.5)+ #Sólida, preta e com espessura 0.5.
                                geom_hline(yintercept = 0.7,
#Adiciona linha horizontal no intercepto 0.7 em y.
                                                   linetype="solid", color =
"black", size=0.5) +  #Sólida, preta e com espessura 0.5.
                                theme_void() +
#Utilizando o tema pronto void, que tem um padrão vazio.
                                theme(plot.margin=unit(c(-1,0,0,3),
"pt"))+#Estabelece margens superior, direita, inferior e esquerda em pontos.
                                theme(axis.title=element_blank())  +
#Remove título dos eixos.
                                theme(axis.text=element_blank())  +
#Remove anotações dos eixos.
                                theme(axis.ticks=element_blank()) +
#Remove marcações dos eixos.
                                scale_x_continuous(limits = c(0,48),
#Estabelece limite do eixo x indo de 0 a 48.
                                                           expand = c(0,0)) +
#Inicia o gráfico na origem zero-zero.
                                scale_y_continuous(limits = c(0,1),
#Estabelece limite do eixo y indo de 0 a 1.
                                                           expand = c(0, 0)) +
#Inicia o gráfico na origem zero-zero.
                                annotate("text", x = (length(x2)/2), y =
0.2, #Adiciona texto com x no meio da régua e y=0.2.
                                                     label = "Hora
(Zt)",size=4,family="serif") + #Escreve 'Hora(Zt)' em tamanho 4 e fonte
'serif', que significa Zeitgeber time na área de Cronobiologia.
                                annotate("text", x = 0.3, y = 0.55,
#Adiciona texto com x=0.3 da régua e y=0.55 (Porque não aparece se for
escrito em x=0 por conta dos limites do gráfico).
```

```
                                        label = "0",size=3,family="serif")
+ #Escreve '0' em tamanho 3 e fonte 'serif'.
                                annotate("text", x = 47.6, y = 0.55,
#Adiciona texto com x=47.6 da régua e y=0.55 (Porque não aparece se for
escrito em x=48 por conta dos limites do gráfico).
                                        label = "48",size=3,family="serif")
#Escreve '48' em tamanho 3 e fonte 'serif'.
                for(d in (c(12,24,36)))
#Cria ciclo para a escrever a hora nas posições 12, 24 e 36.
                  {
#Indica início do ciclo.
                  regua.hora <- regua.hora +
#Adiciona ao gráfico regua.hora.
                                  annotate("text", x = (d), y = 0.55,
#Texto em x no valor do contador e y=0.55.
                                           label = paste0(d),
#Escreve o valor do contador.
                                           size=3,
#Com tamanho 3.
                                           family="serif")
#Com fonte 'serif'.
                  }
#Finaliza o ciclo.
                #Legenda:
                x3 <- seq(0,48)
#Cria objeto x3 com valores de 0 a 48.
                y3 <- rep(1,49)
#Cria objeto y3 com 49 vezes de 1. Esses valores serão a base para criar os
traços verticais da régua inferior.
                regua3 <- data.frame(x3,y3)
#Une x3 e y3 em um dataframe regua3.
                legenda <- ggplot(regua3,aes(x=x3,y=y3)) +
#Cria o gráfico legenda, com base em x3 e y3 de regua3.
                           theme_void() +
#Utilizando o tema pronto void, que tem um padrão vazio.
                           theme(plot.margin=unit(c(-1,0,0,3), "pt"))+
#Estabelece margens superior, direita, inferior e esquerda em pontos.
                           theme(axis.title=element_blank())  +
#Remove título dos eixos.
                           theme(axis.text=element_blank())  +
#Remove anotações dos eixos.
                           theme(axis.ticks=element_blank()) +
#Remove marcações dos eixos.
                           scale_x_continuous(limits = c(0,48),
#Estabelece limite do eixo x indo de 0 a 48.
                                              expand = c(0,0)) +
#Inicia o gráfico na origem zero-zero.
                           scale_y_continuous(limits = c(0,1),
#Estabelece limite do eixo y indo de 0 a 1.
                                              expand = c(0, 0)) +
#Inicia o gráfico na origem zero-zero.
```

```
                              annotate("text", x=6, y=0.7,
#Texto em x=6 e y=0.7.
                                       label = "Regime de fotoperíodo:",
#Escreve "Regime de fotoperíodo:".
                                       size=3,
#Com tamanho 3.
                                       family="serif")  +
#Com fonte 'serif'.
                              annotate("text", x=30, y=0.7,
#Texto em x=30 e y=0.7.
                                       label = "Regime de temperatura:",
#Escreve "Regime de temperatura:".
                                       size=3,
#Com tamanho 3.
                                       family="serif") +
#Com fonte 'serif'.
                              geom_rect(aes(xmin = 13,
#Adiciona um retângulo que começa em x no valor 13.
                                            xmax = 15,
#Termina em x no valor 15.
                                            ymin = 0.55,
#Começa em em y 0.55.
                                            ymax = 0.8),
#Termina em em y 0.8.
                                        colour = "black",
#Contornado em linha na com a cor preta.
                                        fill = "white" ) +
#Preenchido em branco.
                              geom_rect(aes(xmin = 13,
#Adiciona um retângulo que começa em x no valor 13.
                                            xmax = 15,
#Termina em x no valor 15.
                                            ymin = 0.30,
#Começa em em y 0.30.
                                            ymax = 0.55),
#Termina em em y 0.55.
                                        colour = "black",
#Contornado em linha na com a cor preta.
                                        fill = "grey") +
#Preenchido com a cor cinza.
                              geom_rect(aes(xmin = 13,
#Adiciona um retângulo que começa em x no valor 13.
                                            xmax = 14,
#Termina em x no valor 14.
                                            ymin = 0.05,
#Começa em em y 0.05.
                                            ymax = 0.30),
#Termina em em y 0.3.
                                        colour = "black",
#Contornado em linha na com a cor preta.
```

```
                                        fill = "white") +
#Preenchido em branco.
                              geom_rect(aes(xmin = 14,
#Adiciona um retângulo que começa em x no valor 14.
                                            xmax = 15,
#Termina em x no valor 15.
                                            ymin = 0.05,
#Começa em em y 0.05.
                                            ymax = 0.30),
#Termina em em y 0.3.
                                        colour = "black",
#Contornado em linha na com a cor preta.
                                        fill = "grey") +
#Preenchido com a cor cinza.
                              geom_rect(aes(xmin = 37,
#Adiciona um retângulo que começa em x no valor 37.
                                            xmax = 39,
#Termina em x no valor 39.
                                            ymin = 0.55,
#Começa em em y 0.55.
                                            ymax = 0.8),
#Termina em em y 0.8.
                                        colour = "black",
#Contornado em linha na com a cor preta.
                                        fill = "black" ) +
#Preenchido em preto.
                              annotate("text", x=18, y=0.7,
#Texto em x no valor 18 e y=0.7.
                                       label = "C:C",
#Escreve "C:C". (Significa claro constante).
                                       size=3,
#Com tamanho 3.
                                       family="serif") +
#Com fonte 'serif'.
                              annotate("text", x=18, y=0.45,
#Texto em x no valor 18 e y=0.45.
                                       label = "E:E",
#Escreve "E:E". (Significa escuro constante).
                                       size=3,
#Com tamanho 3.
                                       family="serif") +
#Com fonte 'serif'.
                              annotate("text", x=18, y=0.2,
#Texto em x no valor 18 e y=0.2.
                                       label = "C:E",
#Escreve "C:E". (Significa claro-escuro).
                                       size=3,
#Com tamanho 3.
                                       family="serif") +
#Com fonte 'serif'.
                              annotate("text", x=42, y=0.7,
```

```
#Texto em x no valor 42 e y=0.7.
                                        label = "Constante",
#Escreve "Constante".
                                        size=3,
#Com tamanho 3.
                                        family="serif")
#Com fonte 'serif'.
                  if(missing(Fi)==F & missing(Ff)==F)
#Caso haja Fi e Ff declarados:
                  {
#Indica as operações a serem seguidas.
                  legenda <- legenda +
#Adiciona ao gráfico legenda
                              geom_rect(aes(xmin = 37,
#Adiciona um retângulo que começa em x no valor 37.
                                                    xmax = 38,
#Termina em x no valor 38.
                                                    ymin = 0.25,
#Começa em em y 0.25.
                                                    ymax = 0.50),
#Termina em em y 0.50.
                                         colour = "black",
#Contornado em linha na com a cor preta.
                                         fill = "red2") +
#Preenchido em vermelho red2.
                              geom_rect(aes(xmin = 38,
#Adiciona um retângulo que começa em x no valor 38.
                                                    xmax = 39,
#Termina em x no valor 39.
                                                    ymin = 0.25,
#Começa em em y 0.25.
                                                    ymax = 0.50),
#Termina em em y 0.50.
                                         colour = "black",
#Contornado em linha na com a cor preta.
                                         fill = "blue4") +
#Preenchido em azul blue4.
                              annotate("text", x=42, y=0.4,
#Texto em x no valor 42 e y=0.4.
                                        label = "Q:F",
#Escreve "Q:F". (Significa quente-frio).
                                        size=3,
#Com tamanho 3.
                                        family="serif")
#Com fonte 'serif'.
                  }
#Finaliza as operações indicadas

#- - -
          #Desenhar o gráfico final
```

```
          acto1 <- arrangeGrob(grobs=plots1, layout_matrix=mat.layout) #Faz
o arranjo dos objetos gráficos contidos em plots1 de acordo com a mat.layout
e guarda em acto1.
          acto2 <- arrangeGrob(grobs=plots2, layout_matrix=mat.layout) #Faz
o arranjo dos objetos gráficos contidos em plots2 de acordo com a mat.layout
e guarda em acto2.
          grid.arrange(regua.dia,acto1,acto2,regua.hora,legenda,
#Desenha os arranjos seguindo a ordem escrita.
                       widths=c(1.5,10,10),                              #Com
larguras de 1.5(regua.dia) e 10(acto1, acto2 e legenda).
                       heights=c(10,1.5,1.5),                            #Com
alturas de 10(regua.dia, acto1 e acto2) e 1.5(regua.hora e legenda)
                       layout_matrix
=cbind(c(1,NA,NA),c(2,4,5),c(3,4,5)),#Na ordem da matriz
cbind(c(1,NA,NA),c(2,4,5),c(3,4,5)).
                       top="",right="")                                  #Cria
espaço em branco acima e à direita da grade.
#- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
#Retornando os resultados no terminal
num.dias <- cat("Actograma para dados contidos em ",
length(dias_existentes), " dias.\n") #Cria o objeto num.dias com a frase que
indica a quantidade de dias no n.linhas fornecido pelo usuário.

         claro <- rep(NA,length(dias_existentes)) #Cria o objeto claro, com
NA vezes o número de dias existentes.
         escuro <- rep(NA,length(dias_existentes))#Cria o objeto escuro, com
NA vezes o número de dias existentes.
         for(d in 1:(length(dias_existentes)))    #Cria ciclo com contador d
indo de 1 ao número de dias existentes.
             {                                    #Indica início do ciclo.
             escuro[d] <- round(abs((unique(get(paste0("df",(d)))[,5])-
unique(get(paste0("df",(d)))[,4]))/3600)) #Subtrai o horário de
Escuro_inicio de Escuro_fim daquele dia(que devem ser os mesmos para todas
as linhas do df em questão, por isso podemos usar unique), divide por 3600
para transformar em horas, transforma em valor absoluto e arredonda, para
chegar ao valor de duração do escuro em horas. Atribui essa duração à
posição escuro[d] referente a data em que ocorre.
             claro[d] <- 24-escuro[d]             #Calcula a duração do
claro para cada data a partir da complementariedade nas 24h diárias, e
guarda em claro na posição da data correspondente.
             }                                    #Finaliza o ciclo.
         ce <- data.frame(dias_existentes,paste0(claro,":",escuro)) #Cria um
novo dataframe ce, com as datas contidas em dias_existentes e as durações de
horas de claro e de escuro dessa data combinadas, expressas no formato
"(duração de claro em horas):(duração de escuro em horas)", tradicionalmente
escrito na cronobiologia como "C:E".
          colnames(ce) <- c("Data","C:E")         #Altera os nomes das
colunas no dataframe para "Data" e"C:E".
if(missing(Fi)==T & missing(Ff)==T)               #Se não houver Fi ou Ff
declarados:
          {                                       #Indica as operações a
```

```
serem feitas.
          frases <- paste0("Sem regime(s) cíclico(s) de quente-frio
declarado(s).\n","Verifique os regime(s) cíclico(s) de claro-escuro na
tabela gerada.\n")                              #Cria em frases duas frases
lado a lado.
          Regimes <-strsplit(frases, "\n")[[1]]   #Cria o objeto Regimes
para armazenar as frases escritas, separadas em linhas.
          View(ce)                                #Imprime na tela uma
janela com o dataframe ce em forma de tabela.
          } else                                  #Finaliza as operações
indicadas.Do contário:
          {                                       #Indica as operações a
serem feitas.
          quente <- rep(NA,length(dias_existentes))#Cria o objeto quente,
com NA vezes o número de dias existentes.
          frio <- rep(NA,length(dias_existentes)) #Cria o objeto frio, com
NA vezes o número de dias existentes.
          for(d in 1:(length(dias_existentes)))   #Cria ciclo com contador d
indo de 1 ao número de dias existentes.
             {                                    #Indica início do ciclo.
             frio[d] <- round(abs((unique(get(paste0("df",(d)))[,7])-
unique(get(paste0("df",(d)))[,6]))/3600)) #Subtrai o horário de Frio_inicio
de Frio_fim daquele dia (que devem ser os mesmos para todas as linhas do df
em questão, por isso podemos usar unique), divide por 3600 para transformar
em horas, transforma em valor absoluto e arredonda, para chegar ao valor de
duração do frio em horas. Atribui essa duração à posição frio[d] referente a
data em que ocorre.
             quente[d] <- 24-frio[d]              #Calcula a duração do
quente para cada data a partir da complementariedade nas 24h diárias, e
guarda em quente na posição da data correspondente.
             }                                    #Finaliza o ciclo.
          ce.qf <-
data.frame(dias_existentes,paste0(claro,":",escuro),paste0(quente,":",frio))
#Cria um novo dataframe, com as datas contidas em dias_existentes e as
durações de horas de claro e de escuro (C:E) dessa data e as durações das
horas de quente e de frio combinadas expressas no formato "(duração de
quente em horas):(duração de frio em horas)", tradicionalmente escrito na
cronobiologia como "Q:F".
          colnames(ce.qf) <- c("Data","C:E","Q:F")#Altera os nomes das
colunas no dataframe para "Data", "C:E" e "Q:F".
          Regimes <- paste0("Verifique os regime(s) cíclico(s) de claro-
escuro e quente-frio na tabela gerada.") #Cria o objeto Regimes para
armazenar a frase escrita.
          View(ce.qf)                             #Imprime na tela uma
janela com o dataframe ce.qf em forma de tabela.
          }                                       #Finaliza as operações
indicadas.

resultados <- c(num.dias, Regimes)                #Cria o objeto resultados
para armazenar num.dias e Regimes numa lista.
```

```
return(resultados)                                  #Retorna resultados no
terminal.
          }
```