

Jefferson Tiago Silvério da Silva



Aluno de mestrado no Departamento de Fisiologia, IB-USP, no laboratório de Cronobiologia Animal, sob orientação da Prof. Dra. Gisele Oda. Desenvolvo projeto sobre fotoperiodismo e sazonalidade reprodutiva em roedores subterrâneos do gênero *Ctenomys*.

Meus Exercícios

[Meus Exercícios](#)

Projeto Final

Proposta A: Índice de diurnalidade

Contextualização

Em estudos de padrões de atividade e ritmos biológicos o índice de diurnalidade, proposto por [Hoogenboom \(1984\)](#), é usado para medir o quanto um animal está noturno ou diurno. Esse índice é particularmente útil em estudos que avaliam a alteração do padrão de atividade sazonal ou de animais que mudam de noturnos para diurnos em certas situações experimentais. O Índice de diurnalidade é uma proporção entre a quantidade de atividade exibida durante as horas de claro e a atividade durante a fase de escuro. O índice é simétrico ao redor de 0 e pode variar de -1 até 1, onde 1 representa um animal completamente diurno e -1 um animal completamente noturno. Esse índice é calculado pela equação abaixo, onde C_d é a quantidade de atividade diurna e C_n é a quantidade de atividade noturna.

$$D = (C_d - C_n) / (C_d + C_n)$$

O objetivo da função é calcular o índice de diurnalidade de um intervalo de dias qualquer ao longo de um conjunto de dados.

Entrada

`diurnality (datetime, activity, interval = 1, sunrise = NULL, sunset = NULL, lat, long, timezone)`

- `datetime` = Vetor de datahora no formato POSIXct
- `activity` = Vetor numérico
- `interval` = Número inteiro de dias para calculo do índice.
- `sunrise` = Horário de nascer do sol em formato HH:MM.
- `sunset` = Horário de por do sol em formato HH:MM.

- Receber o horário do nascer e do por do sol pelo usuário funcionará bem em casos onde dados não se estendam por um tempo muito longo. Porém, caso o volume de dados seja grande e se estenda por um ano, por exemplo, é necessário calcular a duração do dia a medida que o índice é calculado.
- Nesse caso pensei em usar a função `getSunlightTimes` do pacote `suncalc` para calcular a duração do dia. Dessa maneira, eu adicionaria mais três argumentos na função: **latitude**, **longitude** e **timezone**. Esses são os mesmos argumentos usados pela função `getSunlightTimes`. (Tenho dúvida se essa é uma boa prática)

Pseudocódigo

1. Verificar argumentos:
 1. Se (`sunrise == NULL & sunset == NULL`) então:
 1. Se (`lat == NULL & long == NULL`) então:
 1. Warning ("Fornecer dados para calculo do fotoperíodo")
 2. Senão obtem valores de `sunrise` e `sunset` da função `getSunlightTimes`
1. `ni` = número intervalos possíveis dentro dos dados (total de dias dos dados/interval)
2. For de `i` de 1:`ni`
 1. Atribuir dados que estão dentro do intervalo a um novo vetor `activity.interval`
 2. Recalcula `sunrise` e `sunset` usando `getSunlightTimes`
 3. `nr = length(activity.interval)`
 1. For `j` de 1:`nr`
 1. Horário do registro[`j`] está no intervalo entre `sunrise` e `sunset`?
 2. Se sim `day.act = day.act + atividade do registro[j]`
 3. Se não `night.act = night.act + atividade do registro[j]`
 2. Fim do loop
 4. Calcular índice do intervalo `diurnality[i] = (day.act - night.act)/(day.act + night.act)`
3. Fim do Loop

Saída

- Um vetor com os índices calculados.
- Um gráfico de pontos mostrando a variação da diurnalidade ao longo do tempo, semelhante a:



Referências

- <http://biostat.mc.vanderbilt.edu/wiki/pub/Main/ColeBeck/dateestimes.pdf>
- <https://www.gormanalysis.com/blog/dates-and-times-in-r-without-losing-your-sanity/>
- https://www.r-project.org/doc/Rnews/Rnews_2001-2.pdf
- Hoogenboom, I., Daan, S., Dallinga, J.H. et al. *Oecologia* (1984) 61: 18.

Comentários Lucas Freitas

Acredito que essa primeira proposta seja a mais interessante! Quanto o objetivo tudo OK, vc está transformando uma tarefa necessária e cansativa em algo que uma vez organizado fica extremamente conveniente. Aqui vão minhas pontuações.

Ambas as estratégias de obtenção de dados de sunrise e sunset são válidas. Você pode sim utilizar o pacote suncalc para calcular esses valores dentro da sua função. Não se esqueça de otimizar os warnings no começo para que o usuário saiba que um dos dois conjuntos: (sunrise e sunset) ou (lat, long, null, timezone) deve estar completo e os elementos do outro conjunto devem estar vazios.

Dica: crie duas variáveis dentro da função (bottom.limit upper.limit) e use testes lógicos para guardar os valores de sunset e sunrise conforme a utilização de ou o input dos dados ou sua aquisição com o get SunlightTimes.

Dica 2. Cuidado ao extrair os valores com a função getSunlightTimes. O formato dos horários na variável resposta pode não ser o desejado. Se necessário é possível a extração de elementos de strings com outras funções. (formas de conversão mais simples também devem existir)

O argumento Interval é desnecessário, uma vez que esse valor deve ser o mesmo de o tamanho do vetor de sunrise e/ou sunset (se eu entendi direito).

Não entendi exatamente como os dados estão organizados:

Datetime seria uma matriz em que cada linha representa o início de uma atividade que foi realizada pelo indivíduo observado no formato POSIXct? Ela contém a data e o Horário? Eles estão juntos em uma variável ou cada um em colunas diferentes? Pense nisso na hora de elaborar os cálculos. Activity seria um vetor contendo a duração de tempo de cada uma das atividades iniciadas na respectiva posição em datetime?

De qualquer forma imagino que em algum lugar estão os: Inícios das observações, tempo das observações e os valores referentes a proporção dia e noite na combinação dos dados e isso seria o bastante para o cálculo de apenas um indivíduo.

O que acontece se a atividade começa durante o dia e termina de noite?

Talvez nomear os elementos do vetor resposta, uma vez calculado, seja interessante. Da forma como vc pensou acredito que fique fácil fazer a confecção do gráfico no final com base na variação do índice.

Jefferson

Oi, Lucas. Obrigada pelos comentários. Bom, acho que na minha proposta ficaram faltando explicar algumas coisas. Os dados desses tipos geralmente vêm em formato de texto. O arquivo geralmente tem duas colunas, uma com a data e hora e a outra com a atividade correspondente a aquela. Um exemplo de dados de acelerômetro registrando a atividade a cada 5 minutos, durante 23 dias, seria:

```
datetime    activity
05/03/2019 08:05:30.000 0.075
05/03/2019 08:10:30.000 0.057
05/03/2019 09:05:30.000 0.170
05/03/2019 09:10:30.000 0.210
05/03/2019 09:15:30.000 0.241
```

·
·

```
.  
28/03/2019 22:05:30.000 0.000  
28/03/2019 22:10:30.000 0.001  
28/03/2019 22:05:30.000 0.020  
28/03/2019 22:10:30.000 0.009
```

Então, penso que para tornar a função mais flexível o melhor seria que ambos os argumentos de entrada `datetime` e `activity` sejam vetores. O usuário leria esse conjunto de dados, provavelmente como `dataframe`, trataria como necessário e poderia passar os argumentos para a função tanto indexando o `dataframe` quanto usando vetores separados.

Acho que uma coisa que não ficou muito claro na minha proposta inicial é que nem sempre o índice é calculado por dia. Por exemplo, dentro desse conjunto de dados poderia calcular o índice de diurnidade por dia e no final receberia um vetor com 23 linhas correspondentes ao índice de cada um dos dias. Porém, dependendo da duração do registro ou do objetivo do trabalho, talvez eu queira calcular o índice de diurnidade a cada 5 dias, ou a cada 30 dias. Esse número de dias para o qual o índice vai ser calculado é o que eu nomeei de `interval` nos argumentos.

A ideia, então, é receber o intervalo de dias para o qual o índice vai ser calculado. Dentro desse intervalo, separar o que é atividade diurna e o que é atividade noturna. Por fim, calcular o índice correspondente ao intervalo de dias fornecido pelo usuário.

Proposta B: Plotando Tweets

O twitter é, infelizmente, o meio de comunicação oficial de Bolsonaro. Acompanhar as postagens do presidente, no entanto, é um exercício de paciência. Assim, minha segunda proposta é fazer uma função que receba o nome de um usuário qualquer do twitter e retorne alguns gráficos que sumariam o número de tweets, RTs e curtidas por dia. Além disso, permitir ao usuário fornecer uma ou mais palavras chaves para fazer uma busca específica dentro dos twitts. A ideia é reproduzir algo como feito nesse artigo do [Nexo](#) porém para uso contínuo.

Entrada

```
plot.tweet(user, n = 1000, keywords = NULL)
```

- `user` = nome do usuário no twitter.
- `keywords` = palavras chaves a serem buscadas nos tweets do usuário.
- `n` = número de tweets a serem recuperados, deve ser menor do que 3200.

Pseudocódigo

1. Verificar argumentos
2. Conectar com API do twitter usando o pacote `rtweet`
3. Fazer chamada para recuperar dados do usuário desejado
4. Atribui o resultado da busca à variável `tweets`
5. Plot de Quantidade de tweets, RTs e curtidas em formato de calendário usando `ggplot2`

1. Se `length(keywords) > 1` então:
 1. Cria uma lista `tweets.summary` que recebe na posição `[1]` os valores de média de tweets por dia, média de RTs por tweet e médias de curtidas por tweet
 2. For `i` de `1:length(keywords)`
 1. `tweets.keywords = grep(keyword[i], tweets)`
 2. Plot de quantidade de tweets por palavra chave usando `ggplot2`
 3. `tweets.summary[i+1]` recebe os valores da média de tweets com a palavra chave por dia, média de RTs por tweet e médias de curtidas por tweet
 3. Fim do for
2. Se não:
 1. Cria uma vetor `tweets.summary` que recebe os valores de média de tweets por dia, média de RTs por tweet e média de curtidas por tweet.

Saída

1. Uma lista ou vetor com os valores médios de tweets por dia, RTs por tweet e curtidas por tweet
2. Gráficos do tipo heatmap com a quantidade de tweets, RTs e curtidas por dia

Alguns Links

<http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html#Calendar%20Heat%20Map>

<http://www.roymfrancis.com/calendar-plot-with-ggplot2/>

Comentários Lucas Freitas

Quanto a segunda proposta não entendo nem de twitter nem da função `retweet`. Maaaaaaas, se você conseguir acesso a um arquivo contendo as informações necessárias:

1. todos os Tweets em strings
2. a data da publicação deles
3. o número de Retweets de cada um deles

Acredito que se fizer o proposto pelo pseudocódigo terá sucesso

Peço desculpas se entendi algo errado . Qualquer coisa estou a disposição.

Email: rodriguesdefreitaslucas@gmail.com Boa sorte.

Proposta 1: Função Diurnality

[diurnality.r](#)

```
#####
# Calculo do indice de diurnalidade
```

```
# Jefferson Silva
#####

diurnality = function(datetime, activity, interval = 1, lat = NULL, lon
= NULL, sunrise = NULL, sunset = NULL, graph = TRUE){
  # checa se pacotes necessários estão instalados
  if (!require(suncalc)){
    # Instala pacotes caso não estejam instalados
    install.packages("suncalc")
  }else
  {
    # carrega pacotes necessários
    # suncalc é usado para calcular a duração do dia em determinado
    dia do ano dado as coordenadas do local
    require(suncalc)
  }
  #####
  # Verifica argumentos
  #####
  ##### Datetime deve ser do tipo POSIXct para evitar erros futuros
  if(!inherits(datetime, "POSIXct")){
    # caso não seja a função é parada
    stop("Argumento 'datetime' deve ser da classe POSIXct")
  }

  # Intervalo para calculo do indice deve ser > 1
  if(interval<=0){
    stop("Argumento 'interval' deve ser >= 1")
  }
  if(!is.numeric(activity)){
    stop("Argumento 'activity' deve ser numérico.")
  }
  if(length(activity) != length(datetime)){
    stop("Argumentos 'datetime' e 'activity' devem ter o mesmo
    tamanho.")
  }
  ##### Devem ser preenchido (sunrise e sunset) ou então (lat, lon)
  if(is.null(sunrise) | is.null(sunset)){
    if(is.null(lat) | is.null(lon) ){
      # Nenhum dos conjuntos de argumentos foi fornecido
      stop("É necessário fornecer manualmente o valor de nascer e
      pôr do sol ou as coordenadas do local.")
    }
    else{
      # caso apenas lat lon sejam fornecidas
      cat(">>> A duração do dia será automaticamente calculada
      usando as coordenadas fornecidas.\n")
      # atribui TRUE para uma variavel indicadora, o que significa
      que a duração do dia seá calculada usando o pacote suncalc
      coord = TRUE
    }
  }
}
```

```

    }
  }
  else{
    if(is.null(lat) | is.null(lon)){
      # checa se sunrise e sunset estão no formato correto de
      HH:MM
      if ( is.na(as.POSIXct(paste(Sys.Date(), sunset), format =
"%Y-%m-%d %H:%M")) | is.na(as.POSIXct(paste(Sys.Date(), sunrise),
format = "%Y-%m-%d %H:%M"))) {
        stop("'Sunrise' ou 'Sunset' não estão no formato
      HH:MM.")
      }
      else{
        # caso apenas sunrise e sunset sejam fornecidos
        cat(">>> A duração do dia será feita com base no
horário de nascer e pôr do sol fornecidos.")
      }
      # atribui FALSE, indicando que será usado os valores de
      sunrise e sunset fornecidos pelo usuário
      coord = FALSE
    }
    else{
      # Os dois conjuntos de argumentos foram fornecidos
      stop("Apenas um conjunto de argumentos deve ser fornecido
entre horário de nascer e pôr do sol e as coordenadas.")
    }
  }

#####
## calcula indice ##
#####
# Combina dados de entrada em um dataframe, o que facilita a
manipulação.
df = data.frame(datetime, activity)
# omite NAs
df = na.omit(df)

# Usar coordenadas para calculo da duração do dia?
if (coord == FALSE){
  ## Esse bloco de código cria uma nova coluna no dataframe
  ## A nova coluna 'daylight' indica que o registro
correspondente aquela linha foi realizado durante o dia
  # Para cada linha extrai somente o valor da data, descartando
as horas
  dates = as.Date(df$datetime)
  # concatena a data correspondente daquela linha com o horário de
nascer e pôr do sol
  sunrise = paste(dates, sunrise)
  sunset = paste(dates, sunset)
  # Verifica se o registro foi feito entre as horas de nascer e
por do sol e adiciona a nova coluna 'daylight' ao dataframe

```

```
df$daylight = ifelse(test = df$datetime >= sunrise &
df$datetime <= sunset, yes = TRUE, no = FALSE)

}
else{
  ## Para conseguir dados de nascer e por do sol, temos três
  opções em três pacotes diferentes:
  ## suncalc::getSunlightTimes, maptools::sunriset e
  StreamMetabolism::sunrise.set
  ## Em tempo de execução suncalc::getSunlightTimes foi mais
  rápido do que as outras funções
  # cria novo vetor com as datas de nasce e por do sol para cada
  linha do df
  sun = getSunlightTimes(as.Date(df$datetime), lat = lat, lon =
lon, keep = c("sunrise", "sunset"))
  # Verifica se df está entre as horas de nascer e por do sol e
  adiciona a nova coluna 'daylight' ao dataframe.
  df$daylight = ifelse(test = df$datetime >= sun$sunrise &
df$datetime <= sun$sunset, yes = TRUE, no = FALSE)
}
# Cria uma string de acordo com o intervalo de dias fornecido nos
argumentos
b = paste(as.character(interval),"days")
# Cria um fator que corresponde ao intervalo de cada uma das linhas
do df.
cuts = cut(x = df$datetime, breaks = b)
# Cria uma lista dividida por intervalos
datetime.list = split(x = df, f = cuts)
# Separa atividade que acontece durante dia ou noite e calcula o
índice de diurnidade (Hoogenboom, 1984)
d.index = sapply(datetime.list,
function(x){
  # indexa valores de atividade que acontecem
durante o dia
  actv.day = x$activity[x$daylight]
  # indexa valores de atividade que acontecem
durante a noite
  actv.night = x$activity[!x$daylight]
  # somatória da atividade diurna
  actv.day = sum(actv.day)
  # somatória da atividade noturna
  actv.night = sum(actv.night)
  # calculo do índice de diurnidade
(Hoogenboom, 1984)
  d = (actv.day - actv.night)/(actv.day +
actv.night)
}
)

#####
```

```
## plot ##
#####
if(graph){
  # Prepara variaveis para criar retangulo e label do eixo X
  n = length(d.index)
  date.axis = strptime(names(d.index),"%Y-%m-%d")
  # Define como exibir as legendas
  if(interval >= 30){
    # Para intervalos maiores do que 30 dias exibe o nome do
    mês e o ano.
    date.axis = format(date.axis, "%b\n%Y")
  }
  else{
    # Para intervalos menores do que 30 dias exibe dia e nome
    do mês.
    date.axis = format(date.axis, "%d\n%b")
  }
  # prepara paramentros gráficos
  par(pch=16, tcl=-0.3, bty="l", cex.axis = 0.9, las = 1,
  cex.lab=1, mar=c(4,4,2,2))
  # plota pontos
  plot(d.index,
    main = "",
    xlab = "",
    xaxt = "n",
    ylab = "Diurnality Index",
    ylim = c(-1,1)
  )
  # adiciona eixo X
  axis(1, at=1:n, labels=date.axis, mgp = c(3, 1.5, 0))
  # adiciona retangulo cinza na parte noturna do indice
  rect(xleft = c(-1,-1), ybottom = c(-2,-2), xright = c(n+2,n+2),
  ytop = c(0,0), col = rgb(0,0,0,0.009), lty=3, border = NA)
  # conecta pontos
  lines(x = 1:n, y = d.index, col = rgb(0,0,0,0.2), type = "b")
  # adiciona linha horizontal
  abline(h = 0, lty = 3, col = rgb(0,0,0,0.5))
}

#####
## return ##
#####
return(d.index)

} #FIM DIURNALITY
```

[help_diurnality.txt](#)

diurnality
Documentation

package:unknown

R

CÁLCULO DO ÍNDICE DE DIURNALIDADE

Description:

Função para calcular o índice de diurnidade (Hoogenboom, 1984) usado em estudos de ritmos biológicos. A função pode calcular o índice de diurnidade usando dados de nascer e pôr-do-sol fornecidos pelo usuário ou calcular esses horários automaticamente pelas coordenadas geográficas do local de coleta. Além disso, produz um gráfico para visualização dos resultados.

Usage:

```
diurnality(datetime, activity, interval = 1, lat = NULL, lon = NULL, sunrise = NULL, sunset = NULL, graph = TRUE)
```

Arguments:

datetime: Um vetor da classe POSIXct que corresponde a data e o horário das coletas de dados.

activity: Um vetor numérico que corresponde a atividade exibida em cada ponto de coleta de dados.

interval: Número ≥ 1 que corresponde ao intervalo no qual o índice de diurnidade deveria ser calculado. Como padrão o índice é calculado diariamente.

lat: Latitude em valor numérico para cálculo automático da duração do dia.

lon: Longitude em valor numérico para cálculo automático da duração do dia.

sunrise: Caracter em formato HH:MM. Horário de nascer do sol para cálculo da duração do dia. Caso 'sunrise' e 'sunset' sejam fornecidos a duração do dia se manterá a mesma ao longo do tempo.

sunset: Caracter em formato HH:MM. Horário de pôr-do-sol para cálculo da duração do dia. Caso 'sunrise' e 'sunset' sejam fornecidos a duração do dia se manterá a mesma ao longo do tempo.

graph: Logical. TRUE para exibição do gráfico. FALSE para não exibir o gráfico.

Details:

O índice de diurnidade é calculado de acordo com a atividade exibida durante o dia ou durante a noite. Em casos onde a coleta de dados se estenda por um longo período de tempo, a duração do dia pode ser calculada automaticamente caso os argumentos 'lat', 'lon' sejam fornecidos. Caso a coleta de dados seja feita em ambiente de laboratório com luminosidade controlada, a duração do dia pode ser fornecida manualmente usando os argumentos 'sunrise' e 'sunset'.

Somente um desses conjuntos de argumentos deve ser fornecido.

Value:

Um vetor numérico com os valores dos índices calculados.

Gráfico de pontos com os valores dos índices de diurnalidades calculados em função do tempo.

Warning:

Se algum dos argumentos não for inserido corretamente a função não será executada, retornando uma mensagem de erro para identificação do problema.

Author(s):

Jefferson Silvério
jt.silverio@usp.br

References:

Hoogenboom, I., Daan, S., Dallinga, J.H. et al. *Oecologia* (1984) 61: 18. <https://doi.org/10.1007/BF00379084>

See Also:

suncalc
suncalc::getSunlightTimes

Examples:

```
datetime = seq(ISOdate(2018,7,1), ISOdate(2019,7,1), "5 min") #  
Gera dados de Julho/2018 até Julho/2019 com intervalos de 5 minutos  
activity = rpois(length(datetime), 5) # Gera dados de atividade  
diurnality(datetime = datetime, activity = activity, lat =  
-23.5489, lon = -46.6388) # Coordenadas de SP  
diurnality(datetime = datetime, activity = activity, interval = 5,  
lat = -23.5489, lon = -46.6388) # Coordenadas de SP, calculado a cada 5  
dias  
diurnality(datetime = datetime, activity = activity, interval = 30,  
lat = -23.5489, lon = -46.6388) # Coordenadas de SP, calculado a cada  
30 dias  
diurnality(datetime = datetime, activity = activity, interval = 2,  
lat = -28.8, lon = -66.934) # Coordenadas de La Rioja, Argentina.  
Calculado a cada 2 dias  
diurnality(datetime = datetime, activity = activity, interval = 5,  
sunrise = "07:00", sunset = "18:00") # Horário dos crepúsculos se  
mantém constante, calculado a cada 5 dias
```

From:

<http://ecor.ib.usp.br/> - **ecoR**

Permanent link:

http://ecor.ib.usp.br/doku.php?id=05_curso_antigo:r2019:alunos:trabalho_final:jefferson.tiago.silva:start



Last update: **2020/08/12 06:04**