Tamiris Imaeda Yassumoto

Aluna de Mestrado pelo Departamento de Fisiologia do IB-USP, na área de Cronobiologia, sob orientação da Profa. Dra. Gisele Akemi Oda e coorientação da Dra. Verónica Sandra Valentinnuzzi.

Meu projeto é sobre os efeitos agudos da luz sobre os ritmos circadianos do tuco-tuco (*Ctenomys aff. knighti*), um roedor subterrâneo.

Exercícios

exercicios_aula_1.txt

exercicios_-_aula_4.r

exercicios_aula_5.r

exercicio_aula_7.r

7b._exercicios_de_regressao_multipla.r

exercicios_8.r

Proposta de trabalho final

Proposta A

Contexto

Aparelhos para medição de movimento podem fornecer dados de posição de um animal em relação a eixos x e y, ou de aceleração em relação a eixos x, y e z. A função que eu gostaria de desenvolver geraria um gráfico animado mostrando a mudança de posição do animal nesses eixos.

Input

Data frame com os dados de posição/aceleração em cada eixo. Os argumentos principais serão x, y e z, e eu gostaria de criar também um argumento que permita controlar a velocidade da animação (quantos gráficos/frames por segundo).

Função

Para a criação da função, penso em utilizar um ciclo que construa um gráfico scatterplot 3D (pacote “scatterplot3d”) para cada linha do data frame, ou seja, um gráfico mostrando a posição do animal em relação aos 3 eixos em cada medição. Depois, utilizando o pacote “animation”, esses gráficos seriam salvos em sequência para criar o efeito de animação, gerando um arquivo GIF.

Outras possibilidades

Se a função for simples demais, também pensei na possibilidade de gerar um gráfico de linha de aceleração total ($a = ax^2 + ay^3 + az^2$) pelo tempo (no caso, pediria uma coluna com o tempo no data frame do input). Esse gráfico permitiria visualizar em que horários do dia o animal estaria mais ativo, por exemplo.

Dúvida

Estou com dificuldades para criar um ciclo que construa um gráfico scatterplot por linha do data frame usando o "for". É possível e só preciso pensar um pouco mais, ou devo buscar outra maneira de criar esses ciclos de gráficos?

Proposta B

Contexto

Há situações em que se deseja aplicar o teste t para comparar dois grupos em que cada indivíduo teve suas medições tomadas diversas vezes (por exemplo, temperatura corpórea medida a cada 5min). A função permitiria a aplicação do teste t considerando a média entre as medições do indivíduo (por exemplo, temperatura corporal média de um roedor durante um determinado horário) ou a soma dos valores medidos (por exemplo, número total de revoluções na roda de atividade que foram medidas a cada 5min durante uma hora).

Input

Data frame com as medições de cada indivíduo em uma coluna. Os argumentos seriam o data frame, a opção de soma ou média, de uni ou bicaudal, de pareado ou não, e as opções de gráficos (boxplot ou gráfico de linhas).

Função

Primeiro, haveria um teste para verificar se a distribuição dos dados é normal. Caso não seja, um aviso apareceria para o usuário. Depois, seria calculada a média ou a soma das medidas de cada indivíduo. A partir desses dados, seria realizado o teste t uni ou bi-caudal, pareado ou não. Por fim, com esses dados, seria gerado um gráfico comparando os dois grupos.

Output

Resultado do teste t e gráfico.

> Ambas as propostas são ótimas! Acho a primeira mais legal, só pq evolve mais desafios realmente ligados a programação, e o resultado é mais divertido!
>
> A segunda é interessante, mas hoje em dia vc raramente veria alguém usando um teste t para medidas repetidas, e muito mais gente usando um modelo misto, que trata disso de forma bastante natural.
>
> Quanto à duvida, é super simples usar um for() para isso! Tente fazer isso para

um exemplo bem simples, e se precisar pergunte no forum!

Por mim pode começar a proposta 1, e o gráfico de aclaração pelo tempo parece um acréscimo simples e bem útil!

—-*Ogro*

Oi Tamiris, também sou da opinião de seguir com a proposta 1! Por favor adicione o gráfico :) e boa sorte! — *Sara Mortara*

Código da função

```
 ### Função move3d ###

move3d <- function(x, y, z=NULL, hora=NULL, vel=0.1,
                   doisd.xlab="X axys", doisd.ylab="Y axys",
                   tresd.xlab="X axys", tresd.ylab="Y axys", tresd.zlab="Z
axys",
                   l.xlab="Time", l.ylab="Total acceleration")
  # x, y e z = dados de aceleração ou de posição em cada eixo, podendo estar
contidos em uma matriz ou data frame;
  # hora= horário de cada observação;
  # vel= intervalo de tempo entre cada gráfico/frame na animação (em
segundos);
  # doisd.xlab, doisd.ylab = nome dos eixos x e y em gráfico 2D;
  # tresd.xlab, tresd.ylab, tresd.zlab = nome dos eixos x, y e z em gráfico
3D;
  # l.xlab, l.ylab = nome dos eixos x e y em gráfico de linhas de aceleração
total pelo tempo.
{
  require(scatterplot3d) # requisita o pacote scatterplot3d
  require(animation) # e requisita também o pacote animation (ImageMagick ou
GraphicsMagick deve estar instalado para seu funcionamento correto)
  ### Gráfico 2D ###
  if(is.null(z)) ### Se não houver eixo z, será gerado um gráfico animado
com apenas dois eixos
  {
    saveGIF({     # Salva cada gráfico gerado como um frame da animação GIF
      resulta2 <- vector("list", length(x))     # Cria uma lista vazia com o
mesmo comprimento de x
      for(i in 1:length(x))     # Cria um ciclo que percorre todo o
comprimento dos dados x (e, consequentemente, dos dados y), gerando gráficos
      {
        par(las=1, bty="l", mar=c(5, 5, 4, 2), tcl=0.2, lwd=1.3)     #
Ajusta alguns parâmetros gráficos: rótulos horizontais, tipo de caixa,
```

```
margens, marcas para dentro do gráfico, linha um pouco mais grossa que o
padrão
        graficos2D <- plot(x=x[i], y=y[i],     # Gera gráficos, cada um com
um único ponto de acordo com suas posições nos eixos x e y
                           pch=19,         # Tipo de ponto = 19
                           xlim=c(-max(abs(x))-0.1,max(abs(x))+0.1),
ylim=c(-max(abs(y))-0.1,max(abs(y))+0.1),   # Limites dos eixos x e y = -
(valor máximo dos dados em módulo)-0.1 e +(valor máximo dos dados em
módulo)+0.1
                           xlab=doisd.xlab, ylab=doisd.ylab,     # Nome dos
eixos x e y, que podem ser escolhidos com os argumentos doisd.xlab e
doisd.ylab
                           cex=2)     # Tamanho do símbolo em relação ao
padrão (como argumento de plot, não afeta os textos, apenas o símbolo)
        resulta2[[i]]<-graficos2D    # Os gráficos são guardados dentro da
lista
      }
    },interval=vel, nmax=300)     # O intervalo de tempo entre cada gráfico
na animação (frames por segundo) pode ser escolhido com o argumento vel, e o
número máximo de frames é 300 (um número que, com vel=1, gera uma animação
de 5 minutos)
  }
  ### Gráficos 3D ###
  else ### Se houver eixo z, será feito um gráfico animado com 3 eixos
  {
    saveGIF({ # Salva cada elemento da lista como um frame da animação GIF
      resulta1 <- vector("list", length(x)) # Cria uma lista vazia com o
mesmo comprimento de x
      for(i in 1:length(x)) # Cria um ciclo que percorre todo o comprimento
dos dados x (e, consequentemente, dos dados y), gerando gráficos
      {
        par(las=1, lwd=1.3)   # Ajusta alguns parâmetros gráficos: rótulos
horizontais, linhas um pouco mais grossas que o padrão
        graficos3D <- scatterplot3d(x=x[i], y=z[i], z=y[i],     # Gera
gráficos scatter plot 3D, cada um deles com um único ponto de acordo com
suas posições em x, y e z
                                    pch=19, cex.symbols=2, box=T,     # Tipo
de símbolo =19, símbolo duas vezes maior que o padrão, caixa=TRUE
                                    xlim=c(-
max(abs(x))-0.1,max(abs(x))+0.1), ylim=c(-max(abs(z))-0.1,max(abs(z))+0.1),
zlim=c(-max(abs(y))-0.1,max(abs(y))+0.1), # Limites dos eixos x, y e z = -
(valor máximo dos dados em módulo)-0.1 e +(valor máximo dos dados em
módulo)+0.1
                                    xlab=tresd.xlab, ylab=tresd.zlab,
zlab=tresd.ylab)   # Nome dos eixos x, y e z, que podem ser escolhidos com
os argumentos tresd.xlab, tresd.ylab e tresd.zlab
        resulta1[[i]]<-graficos3D # Os gráficos são guardados na lista
      }
    },interval=vel, nmax=300) # O intervalo de tempo entre cada gráfico na
animação (frames por segundo) pode ser escolhido com o argumento vel, e o
```

```
número máximo de frames é 300 (um número que, com vel=1, gera uma animação
de 5 minutos)
    ### Gráfico de linha da aceleração total pelo tempo ###
    if(is.null(hora)) # Se não houver argumento hora, o gráfico não será
feito, e aparecerá uma mensagem informando sobre a falta do argumento
    {
      cat("\n Adicione o argumento hora para um gráfico de aceleração total
pelo tempo") # Mensagem sobre a falta do argumento hora
    }
    else # Se o argumento hora for definido pelo usuário, o gráfico será
feito
    {
      x11()   # Abre janela gráfica
      cat("\n Total acceleration = sqrt(ax^2 + ay^2 +az^2)")   # Mensagem
com a fórmula utilizada para calcular a aceleração total, que assume que os
dados já passaram por um fator de correção da gravidade
      par(las=1, bty="l", mar=c(5, 5, 4, 2), tcl=0.2, lwd=1.3)    # Ajusta
alguns parâmetros gráficos: rótulos horizontais, tipo de caixa, margens,
marcas para dentro do gráfico, linhas um pouco mais grossas que o padrão
      graf <- plot(x=hora, y=(sqrt((x^2)+(y^2)+(z^2))), type="l", # Cria
gráfico de linha de aceleração total pelo tempo aplicando a fórmula indicada
na mensagem
                   xlab=l.xlab, ylab=l.ylab) # Os nomes dos eixos podem ser
modificados pelo usuário por meio dos argumentos l.xlab e l.ylab
    }
  }
  # Para o caso do número de gráficos/frames gerados ser maior que o limite
de 300
  if((length(x))>300) # Se o comprimento de x, ou seja, o número de gráficos
gerados, for maior que 300, surgirá uma mensagem
  {
    cat("\n Erro: O número de frames não pode ser maior que 300
        \n x, y e z devem ter <= 300 observações") # Mensagem de erro e
justificativa do erro
  }
}
```

Arquivo da função

funcao_move3d.r

Página help

```
move3d                package:unknown                R Documentation

Gráfico animado de mudança da posição ou aceleração de um corpo em relação a
eixos x e y, ou de aceleração em relação a eixos x, y e z
```

```
Description:

     Cria uma animação em GIF que mostra a mudança da posição de um ponto em
relação aos eixos. No caso de haver eixo z e horários de amostragem
uniformes para os dados de aceleração, também é construído um gráfico de
aceleração total pelo tempo.

Usage:

     move3d <- function(x, y, z=NULL, hora=NULL, vel=0.1,
                    doisd.xlab="X axys", doisd.ylab="Y axys",
                    tresd.xlab="X axys", tresd.ylab="Y axys", tresd.zlab="Z
axys",
                    l.xlab="Time", l.ylab="Total acceleration")
Arguments:

x           Dados de aceleração ou de posição
y           Dados de aceleração ou de posição
z           Dados de aceleração ou de posição
hora        Horário das amostragens
vel         Intervalo de tempo entre cada gráfico/frame na animação em
segundos
doisd.xlab  Nome do eixo x em gráfico 2D
doisd.ylab  Nome do eixo y em gráfico 2D
tresd.xlab  Nome do eixo x em gráfico 3D
tresd.ylab  Nome do eixo y em gráfico 3D
tresd.zlab  Nome do eixo z em gráfico 3D
l.xlab      Nome do eixo x em gráfico de linhas de aceleração total pelo
tempo
l.ylab      Nome do eixo y em gráfico de linhas de aceleração total pelo
tempo

Details:

     Requer os pacotes scatterplot3d e animation. Para o funcionamento
correto do pacote animation, ImageMagick ou GraphicsMagick deve estar
instalado no computador.
     O número máximo de gráficos/frames que podem ser gerados é 300. Com
vel=0.1, o padrão, significa uma animação de 50 segundos. Com vel=1,
significa uma animação de 5 minutos.
     Fórmula utilizada para o cálculo da aceleração total = sqrt(ax^2 + ay^2
+az^2). Assume que os dados já passaram por um fator de correção da
gravidade.

Value:

     Gráfico animado, que é salvo com o nome animation.gif no diretório de
trabalho.
     Gráfico opcional de aceleração total pelo tempo, aberto na janela
gráfica.
```

```
Author(s):

     Tamiris Imaeda Yassumoto

References:

     ImageMagick: http://www.imagemagick.org
     GraphicsMagick: http://www.graphicsmagick.org

Examples:

### Exempo 1 ###
### Aceleração de um objeto em relação aos eixos xyz a cada 0.1 segundos ###

ax <- c(-0.13, -0.19, -0.06, 0.13, 0.19, 0.31, 0.56, 0.5, 0.59, 0.56, 0.44,
0.47, 0.47, 0.50, 0.53, 0.47, 0.47, 0.47, 0.47, 0.47, 0.50, 0.38, 0.41,
0.38, 0.34, 0.31, 0.31, 0.20, 0.10, 0)
ay <- c(0.97, 0.91, 0.94, 0.84, 0.81, 0.56, 0.44, 0.28, 0.25, -0.03, 0.13,
0.31, 0.25, 0.19, 0.16, 0.20, 0.20, 0.20, 0.20, 0.20, 0.26, 0.20, 0.30,
0.47, 0.44, 0.44, 0.38, 0.30, 0.32, 0.23)
az <- c(0, 0, 0, -0.16, -0.53, -0.78, -0.84, -1, -0.97, -0.84, -0.81, -0.91,
-0.84, -0.84, -0.84, -0.69, -0.69, -0.69, -0.69, -0.69, -0.72, -0.84, -0.75,
-0.78, -0.84, -0.84, -0.84, -0.78, -0.60, -0.60)
time <- seq(0.1, 3.0, by=0.1)

exemplo <- data.frame(time, ax, ay, az)
move3d(exemplo$ax, exemplo$ay, exemplo$az, hora=exemplo$time, vel=0.1,
       tresd.xlab="Aceleração em x (m/s²)", tresd.ylab="Aceleração em y
(m/s²)", tresd.zlab="Aceleração em z (m/s²)")


### Exemplo 2 ###
### Mudança de posição de um objeto em relação aos eixos x e y a cada 0.2
segundos ###

px <- c(0.5, 0.5, 0.5, 0.5, 0.5, 1, 1.5, 2.5, 3, 3, 3, 2.5, 2, 1.5, 2.5, 3,
3)
py <- c(0.5, 1, 1.5, 2, 2.5, 2.5, 2.5, 2.5, 2.5, 1.5, 1.5, 1.5, 1.5, 1.5,
0.5, 0, 0)

move3d(px, py, vel=0.2, doisd.xlab="Posição x (cm)", doisd.ylab="Posição y
(cm)")
```