Diego Solano Brenes

Mestrando em Ecología Evolutiva e Comportamental, Instituto de Biociências, USP.

Meu projeto de mestrado tem a ver com o efeito que tem a condição alimentar das fêmeas da aranha *Paratrechalea ornata* sobre o investimento em reprodução que faz o macho.

Exercícios

Exercícios

Propostas

Proposta 1 : Matriz de transições

A procura de padrões nas sequências dos comportamentos, é uma das tarefas possíveis durante a descrição da história natural dos animais. Portanto, os etólogos fazem análises da frequência com que um indivíduo muda de um comportamento a algum outro, procurando transições que possam se generalizar e entender como um comportamento é afetado por outros. Esta análise é utilizada também em outras áreas da biologia quando se tem diferentes “estados” e quer se entender a frequência das transições entre eles.

Atualmente existem ferramentas que geram resultados semelhantes ao esperado na minha função, mas todas elas presentam problemas estatísticos ou metodológicos que fazem da minha função uma boa solução para pessoas que desejam analisar seus dados comportamentais com mais de uma réplica (e.g. individuo) e sem assumir distribuições específicas dos dados. A minha função gerará uma matriz de transições com proporções, utilizando só a sequência temporal dos comportamentos. Além disso, testará para cada transição se o valor observado pode se gerar pelo acaso, também se a matriz observada pode se gerar em cenários aleatórios, e gerará um fluxograma com as transições significativas que permitirá observar o padrões das transições de uma forma mais simples.

Entrada

```
    transition (data, ind.matrix= FALSE, nsim=1000, prob=0.05, param=
c(curve = 0.05, relsize = 0.9, lwd = 0.7,box.type = "rect",box.prop = 0.3,
box.lwd = 1, shadow.size = 0,dr=0.001,self.cex =
0.5,arr.lwd=0.7,arr.type="triangle",arr.pos=0.45,
arr.length=0.15,arr.width=0.1, cex=0.7,box.cex=0.8))
```

data= data frame com duas variáveis categóricas, a variável na primeira posição deve ser a identidade das réplicas (e.g. individuo) e a variável na segunda posição, a sequência temporal dos comportamentos.

Tabela 1. Exemplo de possível data frame que poderia se analisar com a função transition. Na

primeira coluna o usuário deve colocar a identidade dos indivíduos e na segunda coluna os comportamentos feitos em ordem temporal.

ind. matrix= argumento lógico com default FALSE. Se individual= TRUE, retorna um array com uma matriz para cada réplica, com o número de eventos por transição.

nsim= argumento que indica o número de permutações utilizadas para gerar cenários nulos e obter a probabilidade de que as frequências de transições entre estados sejam geradas pelo acaso.

prob= Limite superior de probabilidade das transições para ser plotadas no fluxograma.

param= vetor de possíveis parâmetros para gerar o fluxograma só com as transições que não poderem acontecer pelo acaso.

Requerimentos

O usuário deve ter o pacote diagram instalado.

Data deve ser um data frame com duas variáveis.

As duas variáveis dentro do data frame devem ser variáveis categóricas.

As duas variáveis dentro do data frame devem ter o mesmo comprimento.

As duas variáveis não podem ter NA.

Pseudo-código

1. Fazer um if para = 1) confirmar que data é um data frame com duas colunas, se não, colocar uma mensagem; 2) confirmar se as variáveis têm a mesma quantidade de filas, se não, colocar uma mensagem “As variáveis devem ter a mesma quantidade de filas”;3) confirmar que o usuário tem o pacote diagram instalado, se não, colocar uma mensagem.
2. Fazer as duas colunas do data frame, variáveis categóricas para ter acesso ao número de fatores dentro de cada uma das variáveis (número de réplicas e número total de estados na população de dados)
3. Gerar um array vazio para guardar os dados gerados. O array terá n matrizes de acordo ao número de réplicas, com número de filas e colunas igual ao número de estados na população.
4. Fazer 3 for aninhados que determinem o número de vezes que dentro da réplica k o estado j na posição x mudou ao estado i na posição x+1.
5. Guardar os resultados no array nas posições j,i,k do array.
6. Somar as celdas na mesma ubicação das diferentes matrizes
7. Dividir cada celda por a soma por fila para obter a proporção de ocorrência de cada transição de estados.
8. Obter o valor da variância dos valores dentro da matriz de proporções.
9. Gerar um cenário nulo por indivíduo onde o número de transições entre estados seja aleatório, somar eles para obter um cenário comparável com o observados, simular ele nsim vezes.
10. Obter a probabilidade de obter um número de eventos em cada transição igual ou maior da observada
11. Obter a variância para cada cenário nulo.

12. Obter a probabilidade como a quantidade de vezes que em um cenário nulo, a variância é igual ou maior que o valor observado.
13. Obter uma matriz com só as transições significativas
14. Gerar um gráfico utilizando a função plotmat do pacote diagram utilizando a matriz com só as transições significativas

Saída

Se ind.matrix=FALSE, uma lista com: 1) uma matriz com as proporções das transições entre comportamentos (fila=de, coluna=para), 2) o valor da probabilidade de obter uma matriz com a variação observada pelo acaso, 3) uma matriz com as probabilidades por transição de acontecer pelo acaso, 4) um fluxograma só com as transições que forem significativamente diferentes dos cenários nulos

Se ind.matrix= TRUE, uma lista com: 1) uma matriz com as proporções das transições entre comportamentos (fila=de, coluna=para), 2) o valor da probabilidade de obter uma matriz com a variação observada pelo acaso, 3) uma matriz com as probabilidades por transição de acontecer pelo acaso, 4) um fluxograma só com as transições que forem significativamente diferentes dos cenários nulos, 5) um array com uma matriz por réplica com os valores observados por transição

Proposta 2: Simulando evolução por mutação

Em educação muitas vezes é necessário exemplificar processos evolutivos complexos. Na minha graduação a gente simulava fuerzas evolutivas com grãos de feijão e demorava muito tempo. Especialmente a simulação de mutação era uma das simulações mais difíceis de compreender. Minha ideia é gerar uma ferramenta que simule as mudanças ao longo do tempo das frequências alélicas de uma população que é afetada por uma mutação. Esta função permitirá modificar características iniciais da população e características da mutação. Desta forma, será possível comparar graficamente como a evolução de uma população varia segundo as características iniciais.

Entrada

muta.evol (n=100, gen=30, freq.a=0.49, freq.b=0.5, freq.m=0.01, rate= 0.01, dom=TRUE, del=TRUE).

n= número de indivíduos dentro da população.

gen= número de gerações.

freq.a= frequência alélica inicial para o alelo dominante com default = 0.50.

freq.b=frequência alélica para o alelo recessivo com default = 0.49.

freq.m=frequência alélica para o alelo mutado com default = 0.01.

rate= taxa de mutação do alelo afetado por geração com default= 0.01 .

dom=argumento lógico com default= TRUE que indica se a mutação é uma mutação no alelo dominante, se dom= FALSE, a mutação será uma mutação no alelo recessivo.

del= argumento lógico com default= TRUE que indica se a mutação é deletéria (o indivíduo morre quando apresenta o fenótipo da mutação), se del=FALSE a mutação será neutra (o indivíduo não

morre quando apresenta o fenótipo da mutação).

Requerimentos

freq.a + freq.b + freq.m = 1 n e gen devem ser valores maiores que 0.

Pseudo- código

1. Fazer um if para confirmar que as três frequências sumam 1, se não colocar um stop com uma mensagem
2. Fazer um if para confirmar que n e gen são maiores que 0, se não colocar um stop com uma mensagem
3. Gerar um data frame vazio com três colunas para guardar as frequências alélicas geradas. O data frame terá uma longitude = gen+1.
4. Se dom= TRUE e del= TRUE
5. Fazer um for de 1 até gen
6. Dentro de for
7. Calcular a frequência para cada genótipo inicial da população utilizando as fórmula do equilíbrio de Hardy Weinberg.
8. Tirar uma amostra com reemplazo de tamanho = n que tenha as diferentes genótipos em proporções iguais às calculadas no item 7.
9. Calcular as frequências alélicas dessa geração e guardar no data frame gerado no item 3
10. Eliminar os indivíduos que tenham presente o alelo m
11. Calcular a nova proporção de alelos a, b e m depois de eliminar os indivíduos mutantes
12. Sumar para a freq.m o valor de rate y subtrair o valor de rate à freq.a
13. Utilizar esta nova frequência alélica como frequência inicial para a próxima simulação.
14. Se dom= TRUE e del= FALSE
15. Fazer um for de 1 até gen
16. Dentro de for
17. Calcular a frequência para cada genótipo inicial da população utilizando as fórmula do equilíbrio de Hardy Weinberg.
18. Tirar uma amostra com reemplazo de tamanho = n que tenha as diferentes genótipos em proporções iguais às calculadas no item 7.
19. Calcular as frequências alélicas dessa geração e guardar no data frame gerado no item 3
20. Calcular a nova proporção de alelos a, b e m
21. Sumar para a freq.m o valor de rate y subtrair o valor de rate à freq.a
22. Utilizar esta nova frequência alélica como frequência inicial para a próxima simulação.
23. Se dom= FALSE e del= TRUE
24. Fazer um for de 1 até gen
25. Dentro de for
26. Calcular a frequência para cada genótipo inicial da população utilizando as fórmula do equilíbrio de Hardy Weinberg.
27. Tirar uma amostra com reemplazo de tamanho = n que tenha as diferentes genótipos em proporções iguais às calculadas no item 7.
28. Calcular as frequências alélicas dessa geração e guardar no data frame gerado no item 3
29. Eliminar os indivíduos que tenham presente o alelo m
30. Calcular a nova proporção de alelos a, b e m depois de eliminar os indivíduos mutantes
31. Sumar para a freq.m o valor de rate y subtrair o valor de rate à freq.b
32. Utilizar esta nova frequência alélica como frequência inicial para a próxima simulação.
33. Se dom= TRUE e del= FALSE

34. Fazer um for de 1 até gen
35. Dentro de for
36. Calcular a frequência para cada genótipo inicial da população utilizando as fórmula do equilíbrio de Hardy Weinberg.
37. Tirar uma amostra com reemplazo de tamanho = n que tenha as diferentes genótipos em proporções iguais às calculadas no item 7.
38. Calcular as frequências alélicas dessa geração e guardar no data frame gerado no item 3
39. Calcular a nova proporção de alelos a, b e m
40. Sumar para a freq.m o valor de rate y subtrair o valor de rate à freq.b
41. Utilizar esta nova frequência alélica como frequência inicial para a próxima simulação.
42. Plotar as frequências alélicas num gráfico de três linhas, uma para cada alelo, tendo como valores no eixo y, os valores da frequência alélica e no eixo x, a gerações.

Saída

O gráfico de linhas gerado no item 42.

Vitor Rios

Diego, gostei da proposta A, mas precisa de alguns detalhes pra melhorar o entendimento, principalmente no seu objeto de entrada "data". A estrutura dele não está clara. Sugiro fortemente que você tente estruturar ele numa tabela pra entender melhor a estrutura que ele precisa ter. O que vc quer dizer variáveis discretas? Cada réplica é o id de um indivíduo? Como você vai representar a sequência temporal? Uma string (cadeia de caracteres) de siglas de comportamento? O que você quer dizer por filas, o número de linhas? E o número de estados seria o número de comportamentos? Se você vai colocar a sequencia dentro de uma única célula, na forma de uma string, vai ter que pensar em como acessar esses valores dentro da string pra construir a matriz de transição. A lógica do seu for me parece correta, assim, como os calculos das frequencias. Seria interessante uma opção de retornar a matriz de transição na forma de uma grafo orientado, uma rede onde os nós são os estados e as arestas as transições, com a espessura da aresta proporcional à frequencia da transição, acho que essa representação visual fica melhor de interpretar do que uma tabela

A sua proposta B está bem fechada, mas me parece pouco desafiadora, eu seguiria com a proposta A Quando fizer as alterações, me envie um email para eu olhar novamente Vitor Rios

Trabalho final

```
###FUNÇÃO
transition=function(data,ind.matrix=FALSE,n.sim=1000, prob=0.05,curvat =
0.05, t.quad = 0.9, c.tipo = "rect",c.prop = 0.3, c.lwd = 1, t.sombra =
0,s.linha=0.001,propia.int = 0.5,t.linha=0.7,l.tipo="triangle",l.pos=0.45,
l.comp=0.15,l.larg=0.1, t.txt=0.7,caix.txt=0.8)
{
###REQUERIMENTOS
#Data deve ser um data frame
```

```
if (class(data)!="data.frame")
  {stop("O objeto data deve ser da clase data.frame")}
#Data deve ter mínimo duas colunas
if(ncol(data)<2)
  {stop("O objeto data deve conter mínimo duas variáveis")}
#Coluna 1 deve ter mesmo tamanho que Coluna 2
if (length(data[,1])!=length(data[,2]))
  {stop("As variáveis devem ter o mesmo comprimento")}
#Nenhuma coluna dever ter NA
if(sum(is.na(data))!=0)
  {stop("O objeto data não deve conter NA")}
#Deve ter instalado o pacote diagram
if(require(diagram)!=TRUE)
  {stop("A função transition precisa do pacote diagram")}
#valores categóricos nas variáveis
if(class(data[,1])!="factor"|class(data[,2])!="factor")
  {warning("As variáveis forem trasformadas a fatores")}
### OBTENDO AS MATRIZES DE TRANSIÇÕES GERAL
## Transformando as variáveis a fatores
comporta=data
comporta$ind= as.factor(data[,1])
comporta$comp= as.factor(data[,2])
## Gerando um array para guardar o número de trancisões entre comportamentos
por indivíduo.
freq.bruta=array(NA,
dim=c(length(levels(comporta$comp)),length(levels(comporta$comp)),length(lev
els(comporta$ind))),dimnames =
list(levels(comporta$comp),levels(comporta$comp),levels(comporta$ind)))
## Gerando as matrices de transições dos dados observados
  for(k in 1:length(levels(comporta$ind)))
  {
    #Fazendo um subset por indivíduo para evitar transições entre
comportamentos de dois indivíduos diferentes
    indi=subset(comporta,subset = comporta$ind==levels(comporta$ind)[k])
    for (j in 1:length(levels(comporta$comp)))
    {
      #Averiguando a posição do comportamento j
      pos=which(indi$comp==levels(comporta$comp)[j])
      for(i in 1:length(levels(comporta$comp)))
      {
        #Soma das vezes que o comportamento j na posição pos passou ao
comportamento i na posição pos+1
soma.freq=sum((indi$comp[pos+1]==levels(comporta$comp)[i]),na.rm=TRUE)
        #Guardando os valores da soma no array freq.bruta
        freq.bruta[j,i,k]=soma.freq
      }
    }
  }
  ##Gerando uma matriz com as somatorias das transições entre comportamentos
  sum.freq=apply(freq.bruta, c(1,2),sum)
```

```
  ##Gerando uma matriz com as proporções observadas por linha das transições
de cada comportamento e colocando nomes as colunas e linhas
  prop=round(sum.freq/apply(sum.freq,1,sum),2)
  row.names(prop)=levels(comporta$comp)
  colnames(prop)=levels(comporta$comp)
  ### GERANDO CENÁRIOS NULOS
  ## Gerando um array para guardar os dados simulados por indivíduo
  s.freq.bruta=array(NA,
dim=c(length(levels(comporta$comp)),length(levels(comporta$comp)),length(lev
els(comporta$ind))))
  ##Gerando um array para guardar a somatoria das transições das matrices
simuladas por indivíduo
   c.nulo=array(NA,
dim=c(length(levels(comporta$comp)),length(levels(comporta$comp)),n.sim))
  ##Simulando nsim cenários com sequências aleatórias dos comportamentos
utilizando os mesmos comportamentos observados por indivíduo.
   for(o in 1:n.sim)
   {
     for(l in 1:length(levels(comporta$ind)))
     {
       #Fazendo um subset por indivíduo para evitar transições entre
comportamentos de dois indivíduos diferentes
       s.indi=subset(comporta,subset =
comporta$ind==levels(comporta$ind)[l])
       #Bagunçando os comportamentos para gerar cenários onde a sequência
dos comportamentos é aleatoria
       bagunca=sample(s.indi$comp)
       for (m in 1:length(levels(comporta$comp)))
       {
         #Averiguando a posição do comportamento m
         s.pos=which(bagunca==levels(comporta$comp)[m])
         for(n in 1:length(levels(comporta$comp)))
         {
           #Soma das vezes que o comportamento m na posição pos passou ao
comportamento n na posição pos+1
s.soma.freq=sum((bagunca[s.pos+1]==levels(comporta$comp)[n]),na.rm=TRUE)
            #Guardando os valores da soma no array s.freq.bruta
           s.freq.bruta[m,n,l]=s.soma.freq
         }
       }
     }
     # Gerando uma matriz geral com a soma do numero de transições para cada
comportamento
     soma= apply(s.freq.bruta, c(1,2), sum)
     #Guardando a matriz em c.nulo
     c.nulo[,,o]=soma
   }
   ###AVERIGUANDO AS PROBABILIDADES DE OBTER VALORES GERADOS PELO ACASO
MAIORES O IGUAIS AOS OBSERVADOS
   ##Gerando um array para guardar as matrices com TRUE y FALSE quando em um
cenário nulo o número de eventos por transição foi maior ou igual ao
```

```
observado
   dif.cnulo=array(NA,
dim=c(length(levels(comporta$comp)),length(levels(comporta$comp)),n.sim))
   for (r in 1:n.sim)
   {
     for(q in 1:length(levels(comporta$comp)))
     {
       for(p in 1:length(levels(comporta$comp)))
       {
         #Gerando uma matriz de TRUE y FALSE de quando em um cenario nulo o
número de eventos por transição foi maior ou igual ao observado
         diferencas=c.nulo[p,q,r]>=sum.freq[p,q]
         #Guardando matrices no array dif.cnulo
         dif.cnulo[p,q,r]=diferencas
       }
     }
   }
##Obtendo a probabilidade que em um cenario nulo o numero de eventos por
transição foi maior ou igual ao observado

probabilidade= round(apply(dif.cnulo, c(1,2), sum)/n.sim,3)
#Colocando nomes nas linhas e as colunas de probabilidade
row.names(probabilidade)=levels(comporta$comp)
colnames(probabilidade)=levels(comporta$comp)
##Averiguando a probabilidade de obter uma variacao dos dados em cenários
nulos maiores o iguais do observado (prevendo que matrices de transições com
algum padrao vai ter muitos zeros e valores altos, enquanto matrices sem
padroes vao ter valores semelhantes)
#Gerando um vetor com os devios padrao dos cenarios nulos mais o desvio
padrao da matriz observada
dp.nulo=c(apply(c.nulo, 3, sd),sd(sum.freq))
#Obtendo a frequencia de ter desvios padroes iguais ou maiores ao observado
dp.prob=sum(dp.nulo>=sd(sum.freq))/n.sim


###FAZENDO FLUXOGRAMA

  ##Obtendo a localizacao das transições com uma probabilidade maior ou
igual a prob
  no.sig=which(probabilidade>prob)
  ##Fazendo um objeto igual a prop para modificar ele
  prop.ns=prop
  ##Remplazando as transições com uma probabilidade maior ou igual a 0.05
por NA
  prop.ns[no.sig]=NA
  ##Fazendo um objeto igual a prop.ns para modificar ele
  prop.ns2=prop.ns
  ##Identificando e eliminando as colunas da matriz prop.ns2 que só tem NA
  id.col.solo.na=apply( is.na(prop.ns2), 2, all )
  mat.sem.colna <- prop.ns2[,!id.col.solo.na]
```

```
  ##Identificando e eliminando as linhas da matriz prop.ns2 que só tem NA
  id.fila.solo.na=apply( is.na(mat.sem.colna), 1, all )
  mat.sem.na=mat.sem.colna[!id.fila.solo.na,]
  ##Invertindo a matriz porque a função plotmat grafica de coluna até linha.
  mat.graf=t(mat.sem.na)
  ##Plotando a matriz só com os comportamentos com transições
singnificativas
  plotmat(mat.graf,curve = curvat , relsize = t.quad ,box.type = c.tipo
,box.prop = c.prop, box.lwd = c.lwd  , shadow.size =t.sombra
,dr=s.linha,self.cex = propia.int,arr.lwd= t.linha,arr.type=
l.tipo,arr.pos=l.pos , arr.length= l.comp ,arr.width=l.larg, cex=
t.txt,box.cex= caix.txt)


###FAZENDO O RETURN SEGUNDO ind.matrix
if(ind.matrix==TRUE)
{return(list(Proporções.observadas=prop,Probabilidade.matriz=dp.prob,Probabi
lidade.por.transição=probabilidade,Eventos.por.indivíduo=freq.bruta))}
else
{return(list(Proporções.observadas=prop,Probabilidade.matriz=dp.prob,Probabi
lidade.por.transição=probabilidade))}
}
```

Documentação

```
Generação de matriz de transições, fluxograma e testes probabilísticos



Description

Gera uma matriz quadrada com a frequência das transições entre
comportamentos. A função utiliza permutações para 1) testar variação da
matriz observada pode ser gerada o superada em cenários de transições
aleatórias; e 2) testar se o valor por transição pode ser gerado o superado
em cenários de transições aleatórias. Além disso, a função gera um
fluxograma utilizando só as transições significativas.
Usage:

transition (data, ind.matrix=FALSE, n.sim=1000, prob=0.05, curvat=0.05,
t.quad=0.9, linha=0.7, c.tipo = "rect",
            c.prop=0.3, c.lwd=1, t.sombra=0, s.linha=0.001, propia.int=0.5,
t.linha=0.7, l.tipo="triangle",
            l.pos=0.45, l.comp=0.15, l.larg=0.1, t.txt=0.7, caix.txt=0.8 )
```

```
Arguments:

data: data frame que deve conter como mínimo duas variáveis. A primeira
coluna deve ser a identidade dos
      indivíduos, e a segunda variável deve ser a sequência temporal dos
comportamentos. A função faz fatores os
      valores das duas variáveis.

ind.matrix: retorna um array com matrizes de transições por indivíduo com o
número de eventos por transição.

n.sim: um valor que indica o número de simulações de cenários nulos com
transições aleatórias entre os
       comportamentos.

prob: um valor entre 0 e 1 que indica o maior valor de probabilidade das
transições representadas no fluxograma.

curvat: um valor especificando a curvatura das flechas nas linhas das
transições do fluxograma.

t.quad: um valor com o tamanho da área ocupada pelo gráfico do fluxograma.

c.tipo: Forma das caixas (“rect”, “ellipse”, “diamond”, “round”, “hexa”,
“multi”) do fluxograma.

c.prop: um valor com proporção do comprimento/largura das caixas do
fluxograma.

c.lwd: largura das linhas da borda das caixas do fluxograma.

t.sombra: um valor que indica o tamanho relativo da sombra das caixas do
fluxograma.

s.linha: um valor que indica suavidade das linhas de transições no
fluxograma, valores menores fazem as linhas mais suaves.

propia.int: um valor que indica o tamanho das linhas que indicam a
permanência em um comportamento no fluxograma.

t.linha: largura das linhas das transições do fluxograma.

l.tipo: tipo de flecha nas linhas das transições do fluxograma ("curved",
"triangle", "circle", "ellipse", "T", "simple").

l.pos: valor de 0 até 1 que indica a posição da flecha nas linhas de
transições do fluxograma.

l.comp: comprimento das flechas nas linhas de transições no fluxograma.
```

```
l.larg: largura das flechas nas linhas de transições no fluxograma.

t.txt: tamanho do texto que indica a frequência das transições no
fluxograma.
caix.txt: tamanho do texto dentro das caixas no fluxograma.



Details:

O número total de comportamentos no data define o número de linhas e
consequentemente das colunas da matriz. Se ind. matrix é TRUE, a função
retorna o array com as matrizes por indivíduo.
Para fazer o fluxograma, a função utiliza a função plotmat com a
possibilidade de modificar só alguns argumentos dessa função, o resto de
argumentos mantém os valores por default da função original. O número de
comportamentos representados no fluxograma varia segundo o número de
transições frequentes entre eles e o valor de prob desejado. Quando o valor
de prob é 1, o fluxograma representa todos os comportamentos e as transições
observadas.
Quando curvat é 0, as flechas nas linhas do fluxograma são diretas.

Segundo a forma colocada em c.tipo as caixas do fluxograma podem tomar as
seguintes formas:
“rect”: um rectângulo
“ellipse”: uma elipse
“diamond”: um diamante
“round”: um rectângulo com as esquinas redondeadas
“hexa”: um hexágono
“multi”: uma figura multigonal

Segundo o tipo de forma de flecha selecionada em l.tipo, a flecha pode tomar
os seguintes aspectos:
"curved": flechas com linhas curvadas hacia dentro
"triangle": flecha com forma de triángulo
"circle": um circulo
"ellipse": uma elipse
"T": uma linha transversal da linha da transição
"simple": flecha utilizada por outras funções de R.

l.pos determina a posição da flecha dentro da linha de transição, valores
perto de 0 colocaram a flecha perto do comportamento inicial, o valor 0.5
colocará a flecha no ponto central entre os dos comportamentos e valores
perto de 1, perto do comportamento final.



Value:

Uma lista contendo:

Proporções.observadas: matriz quadrada com o estado inicial nas linhas e o
```

```
final nas colunas. Os valores dentro da
                        matriz representa a frequência das transições do
comportamento na linha com todos os
                        comportamentos nas colunas. Os valores forem obtidos
com a somatória dos eventos observados
                        por indivíduo para cada transição, dividida por a
somatória total dos eventos por linha.
                        Portanto, as somatórias das frequências por linha da
matriz, devem somar 1.

Probabilidade.matriz: Probabilidade de obter em n.sim matrizes com o ordem
aleatório dos comportamentos, desvios
                       padrões iguais o maiores do desvio padrão da matriz de
eventos por transição observada.

Probabilidade.por.transição: Probabilidade por transição de obter em n.sim
simulações com o ordem aleatório dos
                             comportamentos, quantidades de eventos por
transição iguais o maiores dos eventos
                             observados.

Eventos.por.indivíduo: um array com uma matriz por indivíduo com o número de
eventos observados por transição.


Author(s):

   Diego Solano-Brenes <diegosb04@gmail.com>


See Also:

plotmat

Examples:

###Exemplo
x1= as.factor(sample(rep(letters[1:13],length=100), replace = T))
x2= as.factor(rep(1:5, each=20))
data= data.frame(x2,x1)
transition(data)
```