

Fernando Henrique A. Farache



Doutorando em Entomologia, Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto, USP.

Projeto: Taxonomia e sistemática do gênero Neotropical *Idarnes* Walker 1843 (Chalcidoidea: Sycophaginae). Neste projeto pretendo realizar revisão taxonômica e analisar filogeneticamente *Idarnes*, o gênero mais diverso de vespas não-polinizadoras associadas a *Ficus* na região Neotropical. Além disso, pretendo analisar aspectos de co-especiação entre o grupo e suas plantas hospedeiras.

exec

Proposta de Trabalho Final

Principal

Função: Seleção natural de palavras.

Para exemplificar o funcionamento da seleção natural, pretendo criar uma função na qual inserimos as letras de uma palavra na forma de vetor (eg. c("r","o","m","a")). Esta função funcionará em dois modos:

Modo acaso(argumento modo="acaso"): a função irá reconstruir a palavra re-amostrando todas as letras do vetor até formá-la;

Modo seleção natural(argumento modo="selec"): a função irá reamostrar todas as letras, porém, quando conseguir acertar alguma das letras, esta letra será mantida e as outras continuarão sendo re-mostradas.

Em ambos os modos, o número de gerações (loops) necessários para formar a palavra serão contados. Assim, poderão ser comparados os números de repetições necessários para formar as palavras dos dois modos.

Esta função tem como objetivo exemplificar de forma didática o processo evolutivo, que não atua de forma completamente aleatória (como exemplificado no "modo acaso") e sim atua selecionando as características que aumentam o "fitness" do indivíduo e passando-as para a próxima geração, podendo assim formar estruturas complexas (como exemplificado no "modo seleção natural").

Referências: Dawkins, R. (1976). *The Selfish Gene*. Oxford: Oxford University Press.
ISBN 0-19-286092-5.

Dawkins, R. (1996). *Climbing Mount Improbable*. New York: W. W. Norton & Company.
ISBN 0-393-31682-3.

Comentários

Idéia muito interessante! Seria legal produzir um plot para visualizar o resultado, como o número de palavras diferentes em relação à palavra alvo por geração, por ex.

Só lembrando que o algoritmo do Dawkins não é esse, que apresenta sistema de “tranca”. O algoritmo original simulava “organismos” que eram palavras, e que apresentavam reprodução e mutação aleatória com uma dada probabilidade. Ou seja, era possível apresentar um afastamento do alvo no algoritmo do Dawkins. No que você propôs, não é. Seria igualmente possível fazer algo desse tipo. Um pouco mais trabalhoso, mas eu acho que muito mais verossimilhante.

— [Fabio de A. Machado](#) 2011/04/06 17:54

Paulo: gostei muito desta forma, pois já é um ótimo desafio. Fazer os gráficos sugeridos pelo Fabio tb é um acréscimo legal e viável.

Plano B

Plano B

Função: organização de matrizes de interação parasitóide-hospedeiro ou polinizador-planta.

Com esta função pretendo criar matrizes de interação entre parasitas ou polinizadores (colocados como colunas) com as plantas as quais eles estão associados (colocados como linhas). Nesta função seriam inseridos os dados brutos, com várias amostras/observações por hospedeiro (e possivelmente diferentes localidades). A função geraria com esses dados uma lista de matrizes. Nesta lista, cada matriz representaria a comunidade de uma localidade, no formato[localidade, hospedeiro, especie]. Quando mais de uma amostra for colocada para cada hospedeiro, a função teria a opção de fornecer a média de indivíduos observados por hospedeiro, a soma do total de indivíduos por hospedeiro ou somente uma matriz de presença e ausência.

Referências: Dorman, C. F. , Fründ, J., Blüthgen, N. & Gruber, B. (2009). Indices, Graphs and Null Models: Analyzing Bipartite Ecological Networks. *The Open Ecology Journal*. **2**, 7-24

Comentários

Parece factível, mas um pouco simples. Talvez fosse interessante concatenar com alguns diagnósticos das redes ecológicas, por exemplo.

— [Fabio de A. Machado](#) 2011/04/06 18:08

Página de Ajuda

selec

package:nenhum

R Documentation

Seleção de palavras por reamostragem exemplificando seleção natural.

Description:

Calcula o número de reamostragens necessárias para formar a palavra desejada de duas formas: amostrando os caracteres aleatoriamente ou mantendo os caracteres que forem selecionados em uma geração e reamostrando os outros caracteres até formar a palavra.

Usage:

```
selec(x,modo="selec",tab.plot=FALSE)
```

Arguments:

x: Vetor de caracteres, formado pelas letras da palavra, minúsculas e sem nenhum tipo de acentuação ou "ç".

modo: define a forma como as palavras serão amostradas: "aleat" para amostragens aleatórias e "selec" para manter caracteres iguais aos caracteres de mesma posição do vetor x

tab.plot: se TRUE fornece tabelas ou gráficos de resumo dos resultados. mais informações em details.

...: qualquer argumento a repassar para a função plot.

Details:

Reamostra caracteres de um vetor do mesmo comprimento de x das seguintes formas:

Se modo="aleat", os caracteres do vetor aleatório são reamostrados até todos serem selecionados ao mesmo tempo e na mesma ordem.

Se modo="selec", os caracteres do vetor são reamostrados. Se algum dos caracteres for igual ao caractere na mesma posição do vetor x, ele será mantido e os outros caracteres serão reamostrados. Este processo continuará até todos os caracteres de x serem obtidos.

O objetivo dessa função é exemplificar de forma didática o processo evolutivo, mostrando que ele não atua de forma completamente aleatória (modo = aleat), mas sim através da seleção das características que otimizam o 'fitness' do indivíduo. Através das pequenas mudanças, uma estrutura complexa (exemplificada por uma

palavra, ou seja, o vetor x) pode ser formada muito mais rápido que pelo puro acaso (modo = selec).

Value:

A função conta o número de rodadas (gerações) necessário para se obter o vetor x através das reamostragens do vetor de mesmo comprimento criado.

Se `tab.plot = TRUE` e `modo = aleat`, a função irá retornar uma tabela mostrando quantas das rodadas (gerações) obtiveram cada quantidade de letras e um gráfico com o número de acertos para cada geração.

Se `tab.plot = TRUE` e `modo = selec`, será criado um gráfico mostrando o decaimento do número de letras que estão sendo remostradas por geração.

No caso de `modo = "aleat"` com muitas letras, os gráficos podem demorar muito para ser plotados, o que pode gerar erro.

Warnings:

A função não aceita maiúsculas, acentuação qualquer caractere que não esteja contido no vetor `letters`

O `modo = aleat` é muito devagar e leva muito tempo para recriar palavras longas. É aconselhável começar por palavras menores (2 ou 3 letras) e depois adicionar letras uma a uma. Quando inserir palavras maiores, utilizar `tab.plot = FALSE`, pois se o número de gerações é grande o gráfico pode levar muito tempo para ser criado, ou algum problema pode ocorrer na criação. Além disso, gráficos gerados para o `modo = aleat` podem ser muito poluídos e difíceis de compreender, dependendo do número de gerações necessárias.

Author(s):

Fernando Henrique Antonioli Farache

fhafarache@gmail.com

References:

Dawkins, R. (1976). The Selfish Gene. Oxford: Oxford University Press. ISBN 0-19-286092-5.

Dawkins, R. (1996). Climbing Mount Improbable. New York: W. W. Norton & Company. ISBN 0-393-31682-3.

See Also:

'letters'

Examples:

```
x<- sample(letters,3) # cria um vetor de caracteres aleatório
selec(x,modo="aleat",tab.plot=FALSE)
selec(x,modo="aleat",tab.plot=TRUE)
selec(x,modo="selec",tab.plot=FALSE)
selec(x,modo="selec",tab.plot=TRUE)
```

```
x<- sample(letters,4)
selec(x,modo="aleat",tab.plot=FALSE) # com 4 letras já pode demorar um
pouco.
selec(x,modo="selec",tab.plot=FALSE)
```

```
x<-
c("I","n","c","o","n","s","t","i","t","u","c","i","o","n","a","l","i","s","s",
"i","m","a","m","e","n","t","e")
# erro: a função não aceita maiúsculas.
```

```
x<-
c("i","n","c","o","n","s","t","i","t","u","c","i","o","n","a","l","i","s","s",
"i","m","a","m","e","n","t","e")
```

```
selec(x,modo="selec",tab.plot=TRUE)
# o modo selec funciona rápido com palavras bem grandes
```

```
# comparando modos aleat e selec
x<-c("o","l","a")
selecionado<-rep(NA,10)
```

```
for(i in 1:10){
selecionado[i]=selec(x,modo="selec",tab.plot=FALSE)
}
selecionado #resultado da função modo selec repetida sobre o vetor 10 vezes

#20 aleat de "o" "l" "a"
aleatorio<-rep(NA,10)
for(i in 1:10){
aleatorio[i]=selec(x,modo="aleat",tab.plot=FALSE)
}
aleatorio #resultado da função modo aleat repetida sobre o vetor 10 vezes;
pode demorar um pouco.

summary(data.frame(selecionado,aleatorio))
boxplot(selecionado,aleatorio)
```

Código da Função

```
selec<-function(x,modo="selec",tab.plot=FALSE,...){

#Aviso para a presença de maiúsculas, acentos e cedilha
  if(length(na.exclude(match(x,letters)))!=length(x))
    {warning("Vetor apresenta caracteres não avaliáveis",immediate.=TRUE)}
  else{

#modo aleatório sem gráficos
  if(modo=="aleat" & tab.plot==FALSE){    if(length(x)>=4){
    cat("Aviso: vetores de caracter muito grandes podem demorar muito para
ser reamostrados. Testar vetores de tamanhos menores e adicionar caracteres
um a um. Consultar ajuda.\n")}
  z<-sample(letters,length(x),replace=TRUE) #cria o primeiro vetor
  i=0 #cria o i inicial (contagem de gerações)
  while (sum(z==x)!=length(x)){ # fazer repetições até o vetor aleatório
  igualar ao vetor inserido
  i=i+1 #soma mais uma geração
  z<-sample(letters,length(x),replace=TRUE) #reamostra o vetor de letras
  }
  return(i)
}

#modo aleatório com gráfico
  if(modo=="aleat" & tab.plot==TRUE){
    if(length(x)>=4){
      cat("Aviso: vetores de caracter muito grandes podem demorar muito para
ser reamostrados. Testar vetores de tamanhos menores e adicionar caracteres
um a um. Para vetores grandes, preferir <tab.plot=FALSE>. Consultar
```

```
ajuda.\n")}\n
z<-sample(letters,length(x),replace=TRUE)\n
i=1\n
a<-sum(z==x) #quantifica o número de letras acertadas\n
while (sum(z==x)!=length(x)){\n
  i=i+1\n
  z<-sample(letters,length(x),replace=TRUE)\n
  a<-c(a,sum(z==x))} #quantifica o número de letras acertadas por geração e\n
concatena ao número de letras acertadas nas gerações passadas\n
plot(a,xlab="Geracoes",ylab="N. de caracteres obtidos",main="Caracteres\n
obtidos por geracao",bty="l",col="blue",cex=0.5,...) #Cria gráfico do número\n
de letras acertadas por geração\n
return(list("Geracoes necessarias"=i,"Tabela de geracoes por\n
acertos"=table(a)))\n
}\n\n
#modo seleção natural\n
if(modo=="selec"){ \n
  z<-sample(letters,length(x),replace=TRUE)\n
  x<-x[-(x==z)==0] # definindo que no caso de qualquer elemento de z ser igual\n
a x, o elemento será removido da reamostragem\n
  i=1\n
  a<-length(x) #comprimento de x será observado, de forma a calcular o número\n
de letras encontradas no decorrer das gerações\n
  while(length(x)!=0){\n
    i=i+1\n
    z<-sample(letters,length(x),replace=TRUE)\n
    x<-x[-(x==z)==0]\n
    a<-c(a,length(x))#número de letras encontradas no decorrer das gerações\n
  }\n\n
#selec com gráfico\n
if(tab.plot==TRUE){\n\n
  plot(a,xlab="Geracao", ylab="Caracteres diferentes de x",main="Aproximacao a\n
x",type="s",,bty="l",col="blue",...)\n
  return(i)}\n\n
#selec sem gráfico\n
if(tab.plot==FALSE){\n
  return(i)}\n
}}}
```

Arquivo da função

[selec.r](#)

Arquivo help

[selec.rtf](#)

From:

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

Permanent link:

http://ecor.ib.usp.br/doku.php?id=05_curso_antigo:r2011:alunos:trabalho_final:fernando:start 

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