Carlos Henrique Tonhatti

cenciado em Ciências Biológicas - Unicamp. Mestrando em genética e evolução no IB Unicamp.

Atualmente trabalho com genética de populações naturais, buscando padrões na diversidade genética das mesmas que possam explicar os processos pelos quais estas populações passaram recentemente.

Meus Exercícios

Linque para a página com os meus exercícios resolvidos

- exer1.r
- exer2.r
- exer3.r
- exer4.r
- exer5.r
- exer6.r
- exer7ff.r
- exer8.r

Trabalho Final

Plano A

Uma função que carregue dados de sequências de DNA no formato fasta de arquivos e estime os indices de diversidade molecular como numero de sítios polimórficos, diversidade de nucleotidios, diversidade haplotipica, etc. Tal como ja existe na programa Arlequin. A saida será um arquivo texto com os dados de todas as populações.

Comentários

Daniel (Musgo): Você pretende construir uma função que faça a leitura do formato fasta como parte da sua proposta? Esse pode ser um desafio, pois, até onde sei, não há caracteres separando cada elemento da sequência de dados neste formato, nesse caso seria mais complexo do que um simples **'read.table'** com os argumentos certos. Do contrário, você poderia procurar a função chamada **'read.fasta'** do pacote **seqinr** que faria esta parte de leitura por você. Com isso você poderia concentrar o seu tempo estimando os índices. Não seria interessante fazer um output com gráficos explicativos? Talvez uma analise de cluster, juntando as sequências mais próximas?

=Resposta ao comentario=

Eu to planejando usar a função **read.dna** que ja existe e faria o trabalho mais complicado, e depois pegar umas funções que ja existem mas que tão espalhadas por varios pacotes (nuc.div, seg,sites, etc) construir algumas outras que não achei em nenhum pacote.

Assim o input seria as sequencias de uma população e a saida uma tabela com os dados de tabulados usando o **write.table**. A ideia dos graficos é uma boa.

COmentários das Respostas aos COmentários

Acho que está bem dimensionado. Pode atacar o plano A! — *Alexandre Adalardo de Oliveira 2012/04/03 21:16*

Plano B

Criar uma função de ordenação de uma coluna de um data.frame mantendo as correspondencias entre as linhas. Sem usar a funçao **order()** ou **sort()** ja implementadas no R. To pensando em usar a ideia do quick sort ou bubble sort.

```
Entrada: um data.frame, qual coluna ordenar
Saida: o mesmo data.frame ordenado pela coluna.
```

Códigos

A função do plano A

```
# Trabalho final ##
## carlos henrique tonhatti
##########################
require(ape)
require(pegas)


# Carregar os dados
mol.index <- function(nomearq,formato="fasta"){

require(ape)
require(pegas)

pop<-read.dna(nomearq, format=formato)


resul <- list()

#tamanho da amostra (numero de individuos
n.ind<-length(labels(pop))
    resul$individuos <-n.ind

#comprimento das seq
sitios <- length(pop[1,])
```

```
    resul$sitios<- sitios

# sitios polimorficos
sitios.seg.p <- seg.sites(pop) # devolve a localização dos sitios
polimorficos
    resul$sitios.seg.pos <-sitios.seg.p
sitios.seg<-length(seg.sites(pop)) # o numero de sitios
    resul$sitios.seg <- sitios.seg

#numero de haplotipos
hap<-haplotype(pop, labels=NULL)
    resul$haplotipos <- hap
hap.l <- attr(hap,"index")
nh <- length(hap.l)
    resul$nh <- nh


#diversidade genetica
#heterozigosidade esperada
freq <- rep(NA,length(hap.l))
for(i in 1:length(hap.l)){
    freq[i] <- length(hap.l[[i]])
    }
    resul$freq.hap <- freq

HE <- heterozygosity(freq, variance=T)
    resul$diver.haplo <- HE

#diversidade nucleotidica
#pi
nd <- nuc.div(pop, variance=T)
    resul$diver.nuc <-nd


# thetaS
thes <- theta.s(s=sitios.seg,n=n.ind, variance= T)
thep <- theta.pi<- nuc.div(pop)*sitios
    resul$theta.s <- thes
    resul$theta.pi <- c(thep,nd[2])

         return(resul)}


plot.mol.index <- function(lista){
    temp <-rep(1,lista$sitios.seg)
    par(mfrow=c(2,1))

    plot(lista$sitios.seg.pos,temp,ann=FALSE, axes=F,cex=1,
         col="blue",pch=20,
         xlim=c(0,lista$sitios))
        mtext("Posição dos sitios segregantes", cex=1.5)
```

```
        axis(1)

    plot(lista$haplotipos)
        mtext("Frequência dos haplótipos",cex=1.5)
}
```

Página de ajuda: Plano A

```
mol.index                     package:<none>                     R
Documentation

Indices moleculares

Description:
    Retorna os indices moleculares a partir de um conjunto de sequencias de
DNA.


Usage:
   mol.index(nomearq,formato="fasta")

   plot.mol.index(x)

Arguments:

    nomearq é o nome da arquivo que contem as sequencias. As sequencias
devem estar alinhadas.
    formato é o formato do arquivo das sequencias. Por padrão é fasta, mais
detalhes veja read.dna.
         x é um lista gerada por mol.index.

Details:

    mol.index retorna uma lista dos seguintes paramentros:numero de
individuos(individuos), comprimento das sequencias(sitios), posição dos
sitios segregantes(stios.seg.pos), numero de sitios segregantes
(sitios.seg),lista de haplotipos(haplotipos),numero de
haplotipos(nh),frequencia de haplotipos(freq.hap),diversidade haplotipica
(diver.haplo), diversidade nucleotidica(diver.nuc), theta s (theta.s), theta
pi (theta.pi).

Value:

     Lista com os parametros estimados o um conjunto de sequencias.

Author:
    Carlos Henrique Tonhatti (carlostomate@gmail.com)

References:
        Nei, M. (1987) _Molecular evolutionary genetics_. New York:
```

```
     Columbia University Press.

See Also:
    pegas,read.dna,ape,seqinr

Example:

## Criando os dados em fasta
    cat("> No305",
    "NTTCGAAAAACACACCCACTACTAAAANTTATCAGTCACT",
    "> No304",
    "ATTCGAAAAACACACCCACTACTAAAAATTATCAACCACT",
    "> No306",
    "ATTCGAAAAACACACCCACTACTAAAAATTATCAATCACT",
    "> No306",
    "ATTCGAAAAACACACCCACTACTAAAAATTATCAATCACT",
    file = "exdna.txt", sep = "\n")

#Executando a função

a<-mol.index("exdna.txt")
plot.mol.index(a)
```

Função plano B

Por segurança acabei fazendo o plano B antes do A. Não funcionou muito bem (ele trava com um conjunto de dados maior) mas seguindo a ideia do “Publish your computer code: it is good enough” coloco ele aqui.

```
"Tsort" Tomate Sort.
```

```
Tsort = function(x,y,cresc=TRUE){
        n = length(x[,y])

    if(cresc==TRUE){
            for(i in 1:(n-1)){
                    for(j in 1:(n-i)){
                            if(x[j,y] > x[j+1,y]){
                                    aux = x[j,]
                                    x[j,] = x[j+1,]
                                    x[j+1,] = aux
                            }
                    }
            }
            return(x)
    }
    else
        {
            for(i in 1:(n-1)){
                    for(j in 1:(n-i)){
                            if(x[j,y] < x[j+1,y]){
```

```
                                    aux = x[j,]
                                    x[j,] = x[j+1,]
                                    x[j+1,] = aux
                                }
                            }
                        }
                        return(x)
                    }

}
```

```
Tsort           package:nenhum            R Documentation

 Ordenação de data.frame e matrizes

Descrição:
    Ordena uma matriz ou data.frame usando uma coluna como referencia sem
perder a correspondencia entre as linhas.

Uso:

    Tsort(x,y,cresc=TRUE)

Argumentos
    x: matriz ou data.frame.
    y: index da coluna de referencia.
    cresc: Sentido de ordenação, cresc= TRUE ordem crescente, FALSE ordem
descrecente.

Detalhes:
      A ordenação é feita utilizando o algoritmo de bolha (bubble sort) que
não é eficiente com um grande conjunto de dados.

Valor:


Avisos:
    A coluna de referencia não pode ser um fator.

Autor:
    Carlos Henrique Tonhatti
    carlostomate@gmail.com

Referencias:

Veja tambem:
    order(), sort()

Exemplo:
```

```
# exemplo 1

numero<-c(4,5,6,7,2)
letra <- c("q","r","s","t","v")
df<-data.frame(numero,letra)

Tsort(df,1,cresc=FALSE)
```

codigos

mol.index Tsort