Mariana Vidal

Doutoranda do Departamento de Ecologia da Universidade de São Paulo, desenvolvendo o projeto “Fragmentação de hábitats na Mata Atlântica e a persistência de redes de interação entre plantas e frugívoros”, sob orientação do Prof. Dr. Paulo Roberto Guimarães Júnior.

Meus exercícios

Exercício 1 Exercício 2 Exercício 3 Exercício 4 Exercício 5 Exercício 6 Exercício 7 Exercício 8 Exercício 9

Proposta de Trabalho Final

Principal

A ideia é desenvolver uma função que identifique inconsistências entre diferentes bases de dados, que deveriam compartilhar as mesmas informações. Seria uma função útil em trabalhos que utilizam dados secundários, coletados por diferentes pesquisadores e que lidam com arquivos de diferentes procedências, modificados segundo diversos critérios. Imagino conseguir com esta função um novo objeto, com todos os registros “faltantes” nas planilhas comparadas.

Dentro de uma mesma base de dados, é importante conferir os nomes científicos das espécies. Gostaria de incluir na função um mecanismo que gerasse um objeto com todos os nomes de espécies (aves e plantas) em ordem alfabética, facilitando a identificação de possíveis erros de digitação. Além disso, pretendo identificar registros duplicados em uma mesma base de dados, casos que estarão listados em um novo objeto gerado pela função.

Para desenvolver esta ambiciosa função, pretendo empregar uma extensa base de dados de interação entre aves frugívoras e plantas, que utilizarei em meu projeto de doutorado. Na verdade, tenho em mãos duas bases de dados:

(1). Dados coletados por um certo pesquisador;

(2). Teoricamente, seria a mesma base de dados anterior (1), incluindo dados coletados por outros pesquisadores.

Em uma primeira avaliação das bases de dados, notei que existem variadas inconsistências entre elas e identificá-las é fundamental para futuras análises.

Comentários

De fato identificar inconsistencia entre bases de dados distintas é bastante útil. A função me parece possível sim, provavelmente irá involver muitas operações lógicas. Uma sugestão é ir além: não

apenas fazendo o diagnóstico, mas dando a lista de dados comuns, já organizados em uma mesma tabela. Não me parece ser muito mais complicado do que a proposta original, e tornaria a função útil não apenas para diagnóse, mas para a manipulação dos dados propriamente ditos.

— *Fabio de A. Machado 2011/04/06 17:07*

Alternativa

Resolver, numericamente, um sistema de equações diferenciais que modelam a dinâmica populacional de plantas e aves mutualistas em uma paisagem composta por um número infinito de manchas idênticas (Fortuna & Bascompte 2006). Neste modelo, as populações locais (nas manchas) das diferentes espécies de aves e plantas possuem taxas de colonização e extinção definidas aleatoriamente, amostradas de uma distribuição uniforme.

Comentários

Visto que o R já tem pacotes que resolvem equações diferenciais, acho que a proposta fica bem simples. A proposta original parece mais interessante.

— *Fabio de A. Machado 2011/04/06 17:17*

Trabalho Final

Página de Ajuda

```
data.check                package:unknown                R Documentation

Identificacao de inconsistencias em bases de dados (data frames) e,
futuramente, entre duas bases de dados distintas.

Description:

Retorna os valores unicos (sem repeticao) de colunas dos data frames em
ordem crescente, permitindo a identificacao de possiveis erros de digitacao;
retorna tambem possiveis linhas repetidas encontradas nos data frames,
facilitando a consulta e verificacao dos dados. No futuro, retornara ainda
as linhas presentes em uma base de dados e ausentes na outra.

Usage:

data.check(a, b, unicos=1, lista_col_uni=NULL, repetidos=1,
lista_col_rep=NULL)

Arguments:
```

a,b objetos que guardam os dois data frames a serem comparados (a e b). A funcao exige que ambos os data frames sejam inseridos. A importacao dos data frames para os objetos a e b deve ser feita com o argumento as.is=TRUE, da funcao read.table().
unicos define as colunas cujos valores unicos (sem repeticao) serao retornados. Se unicos=0, nao serao retornados os valores unicos de nenhuma coluna; se unicos=1, serao retornados os valores unicos de todas as colunas, de ambos os data frames; se unicos=2, serao retornados apenas os valores unicos das colunas especificadas pelo usuario, por meio do argumento lista_col_uni (ver a seguir).
lista_col_uni lista com dois vetores numericos, que contem as colunas cujos valores unicos se deseja obter: o primeiro vetor se refere ao data frame a e o segundo, ao data frame b. Deve ser especificado apenas se unicos=2.
repetidos define as colunas que serao consideradas na busca por linhas repetidas. Se repetidos=0, nao sera feita a busca por linhas repetidas; se repetidos=1, serao retornadas as linhas exatamente identicas, ie, aquelas que possuem elementos iguais com relacao a todas as colunas; se repetidos=2, serao retornadas apenas as linhas com elementos iguais nas colunas especificadas pelo usuario, por meio do argumento lista_col_rep (ver a seguir).
lista_col_rep lista com dois vetores numericos, que contem as colunas que serao consideradas ao se buscar linhas repetidas: o primeiro vetor se refere ao data frame a e o segundo, ao data frame b. Deve ser especificado apenas se repetidos=2.

Details:

A funcao gera valores unicos para todas as colunas (argumento "unicos=1") ou para colunas especificadas pelo usuario (argumento "unicos=2") por meio da funcao unique(), organizando as informacoes geradas em arquivos .txt separados para cada coluna, em ordem crescente.
Na busca por possiveis linhas repetidas em um data frame, a funcao compara todas as linhas, par a par, quanto a todos os seus elementos (argumento "repetidos"=1) ou quanto aos elementos presentes em colunas determinadas (argumento "repetidos=2"), retornando um arquivo .txt com as linhas repetidas no diretorio de trabalho.
Alem dos arquivos .txt, a funcao retorna os resultados em objetos na area de trabalho (veja a seguir).

Value:

Objetos gerados na area de trabalho:

Objetos da classe "list" contendo os valores unicos encontrados nos objetos a e b:
  Se unicos=1: "v.unicos.a", "v.unicos.b"
  Se unicos=2: "v.unicos.a.c", "v.unicos.b.c"

Objetos da classe "data.frame" contendo as linhas repetidas encontradas nos objetos a e b:

```
  Se repetidos=1: "reg.duplicados.a", "reg.duplicados.b"
  Se repetidos=2: "reg.duplicados.ac", "reg.duplicados.bc"

Arquivos .txt gerados no diretorio de trabalho:

Se unicos=1 ou unicos=2: sera gerado um arquivo .txt para cada coluna de
cada data frame, contendo seus respectivos valores unicos. Os nomes dos
arquivos gerados terao a seguinte estrutura:
  Para unicos=1: "unicos1_a ou b_nome da coluna.txt"
  Para unicos=2: "unicos2_a ou b_nome da coluna.txt"

Se repetidos=1 ou repetidos=2: sera gerado um unico arquivo .txt para cada
data frame, contendo as linhas repetidas. Os nomes dos arquivos gerados
terao a seguinte estrutura:
  Para repetidos=1: "registros_repetidos_a ou b.txt"
  Para repetidos=2: "registros_repetidos_a ou b_colunas_especificas.txt"

Warning:



Note:

 A funcao foi pensada de modo a incluir a comparacao entre data frames,
etapa ainda nao implementada. Esta intencao justifica a necessidade dos dois
data frames (a e b) na funcao, assim como a presenca das opcoes de nao
realizar as consultas de valores unicos e linhas repetidas.

Author(s):

Mariana Morais Vidal
marimvidal@yahoo.com.br

References:


See Also:

Para guardar os data frames nos objetos a e b: read.table (as.is=TRUE)

Examples:

library(datasets)
a<- OrchardSprays
b<- cars
  data.check(a,b,unicos=1,repetidos=1)
  data.check(a,b,unicos=2,repetidos=0,lista_col_uni=list(c(1,2,4),c(1,2)))
  data.check(a,b,unicos=0,repetidos=2,lista_col_rep=list(c(1,4),c(1)))
```

Código da Função

```
########## Funcao R ##########

##### Nome #####
#data.check

##### Objetivo #####
#identificar inconsistencias em data frames e entre data frames

##### Argumentos #####
#a e b (data frames a serem comparados)
#unicos=1 (retorna valores unicos de todas as colunas de ambos os data
frames); ou =0; ou=2 (requer lista com 2 vetores com os numeros de colunas
especificas de a e b).
#lista_col_uni=list(c(...),c(...)) -> apenas se unicos=2
#repetidos=1 (retorna os registros repetidos em ambos os data frames,
comparando todas as colunas); ou=0; ou=2 (requer lista com 2 vetores com os
numeros de colunas especificas de a e b que devem ser utilizadas na
comparacao. O retorno contera todas as colunas).
#lista_col_rep=list(c(...),c(...)) -> apenas se repetidos=2
#faltantes=

########## Valores Unicos ##########
#unicos=1 (retorna valores unicos de todas as colunas de ambos os data
frames); ou =0; ou= 2 (lista com 2 vetores com os numeros de colunas
especificas de a e b).

data.check<-
function(a,b,unicos=1,repetidos=1,lista_col_uni=NULL,lista_col_rep=NULL)
{

#Caso o usuario queira os valores unicos de todas as colunas (de ambos os
data frames): unicos=1#

    if (unicos==1)
    {
    v.unicos.a<- list(NA) #Data frame a -> criando uma lista com NAs para
guardar os resultados do for a seguir
    for (j in 1:ncol(a)) #para cada coluna de a,
    {
        unique(a[,j]) #retornar os valores unicos, sem repeticao
        v.unicos.a[[j]]<- unique(a[,j])     #guarda-los no objeto
"v.unicos.a"
        write.table(sort(v.unicos.a[[j]]),file=paste("unicos1_a_",names(a)
[j],".txt",sep=""),sep="\t",col.names=names(a)        [j]) #e salva-los em
ordem crescente em um arquivo .txt
    }
    names(v.unicos.a)<- names(a) #colocando os nomes para cada grupo da
lista
```

```
    v.unicos.a<<- v.unicos.a
    ##write.table(v.unicos.a,file="valores_unicos_a.txt",sep="\t") #nao
consegui salvar a lista como txt
    v.unicos.b<- list(NA) #Data frame b -> criando uma lista com NAs para
guardar os resultados do for a seguir
    for (k in 1:ncol(b)) #para cada coluna de b,
    {
        unique(b[,k]) #retornar os valores unicos, sem repeticao
        v.unicos.b[[k]]<- unique(b[,k]) #guarda-los no objeto "v.unicos.b"
        write.table(sort(v.unicos.b[[k]]),file=paste("unicos1_b_",names(b)
[k],".txt",sep=""),sep="\t",col.names=names(b)        [k]) #e salva-los em
ordem crescente em um arquivo .txt
    }
    names(v.unicos.b)<- names(b) #colocando os nomes para cada grupo da
lista
    v.unicos.b<<- v.unicos.b
    ##write.table(v.unicos.b,file="valores_unicos_b.txt",sep="\t") #nao
consegui salvar a lista como txt
    cat("Valores unicos de a e b em arquivos .txt gerados no diretorio de
trabalho\n")
    }

#Caso o usuario nao queira os valores unicos de nenhuma coluna:#

    if (unicos==0)
    {
        cat("Nao foi feita a compilacao dos valores unicos: argumento unicos
= 0\n")
    }

#Caso o usuario queira os valores unicos de colunas determinadas (de ambos
os data frames): unicos=2. O usuario tera que inserir uma lista com 2
vetores, no argumento "lista_col_uni": list(c(...),c(...))#
    if (unicos==2)
    {
    v.unicos.a.c<- list(NA) #Data frame a -> criando uma lista com NAs para
guardar os resultados do for a seguir
    for (w in 1:length(lista_col_uni[[1]])) #para as colunas de a
especificadas no primeiro vetor da lista "unicos" (do 1o ao ultimo valor),
    {
        unique(a[,lista_col_uni[[1]][w]]) #retornar os valores unicos, sem
repeticao, para o w-esimo elemento do primeiro vetor
        v.unicos.a.c[[w]]<- unique(a[,lista_col_uni[[1]][w]]) #guarda-los no
objeto "v.unicos.a.c"
        write.table(sort(v.unicos.a.c[[w]]),file=paste("unicos2_a_",names(a)
[lista_col_uni[[1]][w]],".txt",sep=""),sep="\t",col.names=names(a)
[lista_col_uni[[1]][w]]) #e salva-los em ordem crescente em um arquivo .txt
    }
    names(v.unicos.a.c)<- names(a)[lista_col_uni[[1]]] #colocando os nomes
para cada grupo da lista
```

```
    v.unicos.a.c<<- v.unicos.a.c
    v.unicos.b.c<- list(NA) #Data frame b -> criando uma lista com NAs para
guardar os resultados do for a seguir
    for (y in 1:length(lista_col_uni[[2]])) #para as colunas de b
especificadas no segundo vetor da lista "unicos" (do 1o ao ultimo valor),
    {
        unique(b[,lista_col_uni[[2]][y]]) #retornar os valores unicos, sem
repeticao, para o y-esimo elemento do segundo vetor
        v.unicos.b.c[[y]]<- unique(b[,lista_col_uni[[2]][y]]) #guarda-los no
objeto "v.unicos.a.c"
        write.table(sort(v.unicos.b.c[[y]]),file=paste("unicos2_b_",names(b)
[lista_col_uni[[2]][y]],".txt",sep=""),sep="\t",col.names=names(b)
[lista_col_uni[[2]][y]]) #e salva-los em ordem crescente em um arquivo .txt
    }
    names(v.unicos.b.c)<- names(b)[lista_col_uni[[2]]] #colocando os nomes
para cada grupo da lista
    v.unicos.b.c<<- v.unicos.b.c
    cat("Valores unicos de a e b em arquivos .txt gerados no diretorio de
trabalho\n")
    }

########## Registros Repetidos ##########

## Argumento repetidos=1: retorna os registros repetidos em ambos os data
frames, comparando as linhas par a par, quanto ao conteudo em todas as
colunas.

    if(repetidos==1)
    {
        #Data frame a:
        fixa.a<- seq(1:nrow(a)) #fixando uma linha
        duplos.a<-array(NA, dim=c(nrow(a),ncol(a),nrow(a))) #criando um
array para guardar todos os testes logicos, em que a dimensao 1 sao as
linhas, a dimensao 2 sao as colunas e a dimensao 3 sao as linhas fixas.
Assim, na 3a dimensao temos cada linha sendo comparada com todas as outras.
        for (z in 1:length(fixa.a)) #compare uma dada linha fixa
        {
            for (x in 1:nrow(a)) #com todas as linhas
            {
            logico.a<- a[fixa.a[z],]==a[x,]
            duplos.a[x,,z]<- logico.a #e guarde os resultados no array
"duplos.a"
            #Eu queria usar o lower.tri aqui, pois os valores se repetem,
mas nao consegui... Inicialmente, eu queria que o segundo for fosse: for(x
in z+1:nrow()), ja que nao e necessario fazer as comparacoes novamente com
as linhas anteriores, mas tambem nao deu certo.
            }
        }
        somas.a<- apply(duplos.a,MARGIN=c(1,3),FUN=sum) #calcule a soma de
todas as linhas (dimensao 1) em cada camada (dimensao 3), pois quero saber
as linhas com soma (numeros de TRUE) igual ao numero de colunas, o que
```

```
mostra as linhas que sao identicas.
        iguais.a<-which(somas.a==ncol(a),arr.ind=TRUE) #retorna as posicoes
do array em que as linhas sao exatamente iguais, ie, para todas as colunas
as duas linhas comparadas sao identicas.
        iguais.a2<- iguais.a[,1]!=iguais.a[,2]
        sum(iguais.a2)
        if (sum(iguais.a2)==0)
        {
            cat("Nao ha registros repetidos em a\n")
        }
        if(sum(iguais.a2)>0)
        {
            iguais.a3<- iguais.a[iguais.a[,1]!=iguais.a[,2]]
            reg.duplicados.a<- a[unique(iguais.a3),]
            reg.duplicados.a$Cod_Linha<- row.names(reg.duplicados.a)
            reg.duplicados.a<<-reg.duplicados.a
            #order.a<- order(reg.duplicados.a$nome da coluna)# so e possivel
ordenar depois, de acordo com cada data frame
            #reg.duplicados.a<- reg.duplicados.a[order.a,]
write.table(reg.duplicados.a,file="registros_repetidos_a.txt",sep="\t",row.n
ames=FALSE)
            cat("Registros repetidos em a em arquivo .txt no diretorio de
trabalho\n")
        }
        #Data frame b:
        fixa.b<- seq(1:nrow(b)) #fixando uma linha
        duplos.b<-array(NA, dim=c(nrow(b),ncol(b),nrow(b))) #criando um
array para guardar todos os testes logicos, em que a dimensao 1 sao as
linhas, a dimensao 2 sao as colunas e a dimensao 3 sao as linhas fixas.
Assim, na 3a dimensao temos cada linha sendo comparada com todas as outras.
        for (u in 1:length(fixa.b)) #compare uma dada linha fixa
        {
            for (v in 1:nrow(b)) #com todas as linhas
            {
            logico.b<- b[fixa.b[u],]==b[v,]
            duplos.b[v,,u]<- logico.b #e guarde os resultados no array
"duplos.b"
            }
        }
        somas.b<- apply(duplos.b,MARGIN=c(1,3),FUN=sum) #calcule a soma de
todas as linhas (dimensao 1) em cada camada (dimensao 3), pois quero saber
as linhas com soma (numeros de TRUE) igual ao numero de colunas, o que
mostra as linhas que sao identicas.
        iguais.b<-which(somas.b==ncol(b),arr.ind=TRUE) #retorna as posicoes
do array em que as linhas sao exatamente iguais, ie, para todas as colunas
as duas linhas comparadas sao identicas.
        iguais.b2<- iguais.b[,1]!=iguais.b[,2]
        sum(iguais.b2)
        if (sum(iguais.b2)==0)
        {
```

```
            cat("Nao ha registros repetidos em b\n")
        }
        if(sum(iguais.b2)>0)
        {
            iguais.b3<- iguais.b[iguais.b[,1]!=iguais.b[,2]]
            reg.duplicados.b<- b[unique(iguais.b3),]
            reg.duplicados.b$Cod_Linha<- row.names(reg.duplicados.b)
            reg.duplicados.b<<- reg.duplicados.b
            #order.b<- order(reg.duplicados.b$nome da coluna)# ordenar
depois, de acordo com as colunas do data frame
            #reg.duplicados.b<- reg.duplicados.b[order.b,]
write.table(reg.duplicados.b,file="registros_repetidos_b.txt",sep="\t",row.n
ames=FALSE)
            cat("Registros repetidos em b em arquivo .txt no diretorio de
trabalho\n")
        }
    }

## Argumento repetidos=0:
    if(repetidos==0)
    {
        cat("Nao foi feita a busca por possiveis registros repetidos:
argumento repetidos = 0\n")
    }

## Argumento repetidos=2. Exige uma lista com 2 vetores com os numeros de
colunas especificas de a e b que devem ser utilizadas na comparacao entre
pares de linhas. O retorno contera todas as colunas.
    if(repetidos==2)
    {
    #Data frame a:
        fixa.ac<- seq(1:nrow(a)) #fixando uma linha
        duplos.ac<-array(NA,
dim=c(nrow(a),length(lista_col_rep[[1]]),nrow(a))) #criando um array para
guardar todos os testes logicos, em que a dimensao 1 sao as linhas, a
dimensao 2 sao as colunas e a dimensao 3 sao as linhas fixas. Assim, na 3a
dimensao temos cada linha sendo comparada com todas as outras.
        for (c in 1:length(fixa.ac)) #compare uma dada linha fixa
        {
            for (d in 1:nrow(a)) #com todas as linhas
            {
            logico.ac<-
a[fixa.ac[c],lista_col_rep[[1]]]==a[d,lista_col_rep[[1]]]
            duplos.ac[d,,c]<- logico.ac #e guarde os resultados no array
"duplos.ac"
            #Eu queria usar o lower.tri aqui, pois os valores se repetem,
mas nao consegui... Inicialmente, eu queria que o segundo for fosse: for(x
in z+1:nrow()), ja que nao e necessario fazer as comparacoes novamente com
as linhas anteriores, mas tambem nao deu certo.
            }
        }
```

```
        somas.ac<- apply(duplos.ac,MARGIN=c(1,3),FUN=sum) #calcule a soma de
todas as linhas (dimensao 1)em cada camada (dimensao 3), pois quero saber as
linhas com soma (numeros de TRUE) igual ao numero de colunas, o que mostra
as linhas que sao identicas.
        iguais.ac<-which(somas.ac==length(lista_col_rep[[1]]),arr.ind=TRUE)
#retorna as posicoes do array em que as linhas sao exatamente iguais, ie,
para todas as colunas as duas linhas comparadas sao identicas.
        iguais.ac2<- iguais.ac[,1]!=iguais.ac[,2]
        sum(iguais.ac2)
        if (sum(iguais.ac2)==0)
        {
            cat("Nao ha registros repetidos em a\n")
        }
        if(sum(iguais.ac2)>0)
        {
            iguais.ac3<- iguais.ac[iguais.ac[,1]!=iguais.ac[,2]]
            reg.duplicados.ac<- a[unique(iguais.ac3),]
            reg.duplicados.ac$Cod_Linha<- row.names(reg.duplicados.ac)
            reg.duplicados.ac<<- reg.duplicados.ac
            #order.ac<- order(reg.duplicados.ac$nome da coluna)# so e
possivel ordenar depois, de acordo com cada data frame
            #reg.duplicados.ac<- reg.duplicados.ac[order.ac,]
write.table(reg.duplicados.ac,file="registros_repetidos_a_colunas_especifica
s.txt",sep="\t",row.names=FALSE)
            cat("Registros repetidos em a em arquivo .txt no diretorio de
trabalho\n")
        }
    #Data frame b:
        fixa.bc<- seq(1:nrow(b)) #fixando uma linha
        duplos.bc<-array(NA,
dim=c(nrow(b),length(lista_col_rep[[2]]),nrow(b))) #criando um array para
guardar todos os testes logicos, em que a dimensao 1 sao as linhas, a
dimensao 2 sao as colunas e a dimensao 3 sao as linhas fixas. Assim, na 3a
dimensao temos cada linha sendo comparada com todas as outras.
        for (e in 1:length(fixa.bc)) #compare uma dada linha fixa
        {
            for (f in 1:nrow(b)) #com todas as linhas
            {
            logico.bc<-
b[fixa.bc[e],lista_col_rep[[2]]]==b[f,lista_col_rep[[2]]]
            duplos.bc[f,,e]<- logico.bc #e guarde os resultados no array
"duplos.bc"
            }
        }
        somas.bc<- apply(duplos.bc,MARGIN=c(1,3),FUN=sum) #calcule a soma de
todas as linhas (dimensao 1) em cada camada (dimensao 3), pois quero saber
as linhas com soma (numeros de TRUE) igual ao numero de colunas, o que
mostra as linhas que sao identicas.
        iguais.bc<-which(somas.bc==length(lista_col_rep[[2]]),arr.ind=TRUE)
#retorna as posicoes do array em que as linhas sao exatamente iguais, ie,
```

```
para todas as colunas as duas linhas comparadas sao identicas.
        iguais.bc2<- iguais.bc[,1]!=iguais.bc[,2]
        sum(iguais.bc2)
        if (sum(iguais.bc2)==0)
        {
            cat("Nao ha registros repetidos em b\n")
        }
        if(sum(iguais.bc2)>0)
        {
            iguais.bc3<- iguais.bc[iguais.bc[,1]!=iguais.bc[,2]]
            reg.duplicados.bc<- b[unique(iguais.bc3),]
            reg.duplicados.bc$Cod_Linha<- row.names(reg.duplicados.bc)
            reg.duplicados.bc<<- reg.duplicados.bc
            #order.bc<- order(reg.duplicados.bc$nome da coluna)# ordenar
depois, de acordo com as colunas do data frame
            #reg.duplicados.bc<- reg.duplicados.bc[order.bc,]
write.table(reg.duplicados.bc,file="registros_repetidos_b_colunas_especifica
s.txt",sep="\t",row.names=FALSE)
            cat("Registros repetidos em b em arquivo .txt no diretorio de
trabalho\n")
        }
    }
}
```

Arquivos da Função

Função data.check
Help da função data.check