

Tauana Junqueira da Cunha



tauana.cunha@usp.br • tauanajc@gmail.com

Recém graduada em Ciências Biológicas pelo IB USP. Realizei minha Iniciação Científica no Instituto Oceanográfico, sob orientação do Prof. Paulo Sumida. No Projeto “Macrofauna associada à macroalga *Dictyota* spp. (Phaeophyceae, Dictyotaceae) no Recife Sebastião Gomes e no Arquipélago dos Abrolhos, Brasil”, fiz o primeiro levantamento taxonômico da fauna fital da região de Abrolhos e estudei o padrão de distribuição temporal e espacial da macrofauna.

Atualmente estou no Laboratório de Poliquetologia do Dpto. de Zoologia (IB USP) trabalhando com a taxonomia da família Sabellidae na minha segunda IC intitulada “Sabellidae (Polychaeta) da Baía da Ilha Grande, Rio De Janeiro, Brasil” sob orientação do Prof. João Nogueira.

[Currículo Lattes](#)

Exercícios

[exec](#)

Trabalho Final

Plano A - Concluído

Criar um conjunto de funções que dê como output as coordenadas de uma lista de localidades. Em muitos trabalhos pode ser interessante apresentar a distribuição dos organismos com os quais se está trabalhando ou informar a origem do material utilizado. Entretanto, as informações disponíveis nem sempre são fáceis de serem compreendidas ou encontradas (como rótulos de organismos depositados em museus, que na maioria das vezes não são renovados e ficam com nomes desatualizados; ou artigos não tão recentes que fornecem a localização de seu material apenas com nomes pouco conhecidos por outras pessoas).

Tendo como base uma tabela de localidades com os nomes das cidades, pretendo criar um conjunto de funções que busque esses nomes numa tabela bem completa de localidades e coordenadas, a princípio no Brasil (tabela referência de coordenadas do país baixada do GEONet Names Server, mais especificamente, desse [link](#)). Essa base inclui nomes de localidades afastadas e nomes antigos que já estão fora de uso, mas que muitas vezes são a única informação disponível. Atualmente, o usuário precisa inserir o nome da localidade, uma de cada vez no campo SEARCH>país escolhido>Name>Search database, o que retorna os registros correspondentes.

A ideia é que a função procure nessa base a partir dos dados fornecidos e, não encontrando uma correspondência perfeita, traga as melhores possibilidades para que o usuário escolha dentre elas (isso poderia ser um argumento de função com TRUE e FALSE, permitindo essa escolha ou retornando a melhor opção escolhida pela própria função). O output sairá num arquivo de texto com as

coordenadas correspondentes a cada entrada. Dando certo, isso poupará um enorme esforço manual de busca.

Ainda, se for viável, pretendo utilizar esse output como base para a montagem de mapas com as distribuições das localidades. Esse [site](#) poderá ser útil.

Plano B

Automatizar uma rotina de análises multivariadas como as que utilizei na minha Iniciação Científica. A partir de dados de quantidades de espécimes e da unidade de medida do substrato, pretendo criar um conjunto de funções que generalize sua utilização para a obtenção da densidade de organismos e, a partir dessa variável, sejam rodadas várias análises multivariadas usualmente empregadas em estudos de comunidades, como nMDS e ANOSIM, passando pela criação da matriz de dissimilaridades, transformações dos dados e por testes que avaliem se os dados atendem aos pressupostos dos demais testes que serão utilizados. Os outputs serão arquivos de textos com os resultados de cada análise e gráficos bem formatados.

Comentários das propostas (Leo)

Acho a proposta A interessante. A proposta B é mais complexa e talvez de difícil execução dentro do prazo proposto. Com relação à proposta A, dois desafios seriam permitir que dentro da base de dados a função retorne caracteres aproximados (que não sejam idênticos), e um algoritmo que classifique as melhores localidades por algum critério (similaridade). Resolvendo estas duas tarefas a plotagem dos registros em um mapa é trivial e pode ser implementada facilmente.

Página de Ajuda / Help

geo.coordinates

package:nenhum

R Documentation

Coordenadas geográficas no Brasil. Mapa de distribuição.

Description:

geo.coordinates procura pelos nomes de localidades brasileiras no banco de dados GEONet Names Server (GNS).

É possível escolher a unidade das coordenadas geográficas que serão obtidas e, caso escolhida a decimal, a função utiliza elementos gráficos para auxiliar na desambiguação de localidades.

A função ainda produz um mapa final com a distribuição das localidades (para coordenadas em decimal). Necessária a instalação dos pacotes `sp` e `maptools`.

Usage:

```
geo.coordinates(x, output="Output", unit="dec", type=F, short=F,  
generic=F, class=F, elev=F, note=F)
```

Arguments:

x caráter. Interpretado como nome do arquivo de entrada se houver apenas um elemento. Deve ser uma tabela em formato de texto com apenas uma coluna contendo cabeçalho e os nomes das localidades de interesse. Ver Examples.

Interpretado como um vetor contendo os nomes das localidades se tiver mais de um elemento.

output caráter. Nome do arquivo que será produzido. Não colocar a extensão. Ver Value.

unit caráter. Unidade das coordenadas geográficas. Default: dec (decimal). Permite a construção de mapas temporários para auxiliar na determinação das localidades e de um mapa final com a distribuição dos pontos de interesse.

Opções são dms (graus, minutos, segundos) ou mgrs (Military Grid Reference System). Não permitem a construção de mapas. Ver Value para interpretação dos resultados.

type lógico. Classificação do nome da localidade. Se TRUE, é inserido como coluna adicional no output. Ver Value para legenda dos resultados.

short lógico. Abreviação do nome da localidade. Se TRUE, é inserido como coluna adicional no output.

generic lógico. Descrição da localidade (como Rio, Fazenda etc). Se TRUE, é inserido como coluna adicional no output.

class lógico. Classificação da localidade. Se TRUE, é inserido como coluna adicional no output. Ver Value para legenda dos resultados.

elev lógico. Altitude em metros. Se TRUE, é inserido como coluna adicional no output.

note lógico. Observações do banco de dados sobre a localidade. Se TRUE, é inserido como coluna adicional no output.

Details:

É recomendada a utilização de `unit="dec"` (unidade decimal das coordenadas) para que seja possível a construção de mapas temporários que ajudem na identificação das localidades. Para tal, os pacotes `sp` e `maptools` devem ser previamente instalados.

Por padrão, os argumentos `type`, `short`, `generic`, `class`, `elev` e `note` não são considerados na função. De acordo com o interesse nas informações, uma coluna para cada um deles pode ser adicionada inserindo TRUE nos respectivos argumentos.

`geo.coordinates` salva o banco de dados baixado do GEONet Names Server e os shapefiles do mapa do Brasil numa pasta chamada "zip" automaticamente criada dentro do diretório de trabalho. Manter a pasta "zip" torna a função mais

rápida para utilização consecutiva.

Value:

geo.coordinates retorna um data.frame com os seguintes componentes nas colunas:

original	nomes originais das localidades (fornecidos no argumento x)
full	nomes como encontrados no banco de dados GNS
lat_dec	latitude em graus decimais ($\pm$ dd.dddddd); sem sinal (+) = Norte, sinal negativo (-) = Sul.
long_dec	longitude em graus decimais ($\pm$ dd.dddddd); sem sinal (+) = Leste, sinal negativo (-) = Oeste.
lat_dms	latitude em graus, minutos e segundos ($\pm$ ddmmss); sem sinal (+) = Norte, sinal negativo (-) = Sul.
long_dms	longitude em graus, minutos e segundos ($\pm$ ddmmss); sem sinal (+) = Leste, sinal negativo (-) = Oeste.
coord_mgrs	coordenadas no sistema militar, sistema alfa-numérico de valores únicos para cada localidade do globo.
type	Classificação do nome da localidade. Legenda: C = Conventional N = Approved NS = Non-roman script D = Unverified DS = Unverified non-roman script H = Historic HS = Historic non-roman script P = Provisional PS = Provisional non-roman script V = Variant or alternate VA = Anglicized variant VS = Variant non-roman script
short	Abreviação dos nomes das localidades.
generic	Descrição da localidade (como Rio, Fazenda etc).
class	Classificação da localidade. Legenda: A = Administrative region feature type P = Populated place feature type V = Vegetation feature type L = Locality or area feature type U = Undersea feature type R = Streets, highways, roads, or railroad
feature type	T = Hypsographic feature type H = Hydrographic feature type S = Spot feature type
elev	Altitude em metros.
note	Observações encontradas no banco de dados sobre a localidade.

Mais detalhes em: http://earth-info.nga.mil/gns/html/gis_countryfiles.html

O `data.frame` é apresentado na tela ou pode ser salvo em um objeto. Seu conteúdo é salvo em uma tabela no diretório de trabalho em formato de texto (.txt separado por tabulação).

Se escolhido `unit="dec"`, a função também exporta para o diretório de trabalho um mapa do Brasil (em formato pdf) com as localidades plotadas de forma numérica, com os índices correspondentes aos nomes das localidades nas linhas da tabela produzida.

Warning:

As coordenadas obtidas são originadas do banco de dados GNS. `geo.coordinates` exibe uma mensagem de aviso quando uma localidade de interesse não é encontrada no banco. Entretanto essas localidades não são inseridas na tabela final produzida.

Para localidades com múltiplos registros no banco de dados, as opções são exibidas para que o usuário faça a escolha crítica da localidade apropriada.

As demais informações que podem ser extraídas (altitude, classificação etc) podem não estar presentes para todas as localidades e devem ser analisadas pelo usuário de forma crítica.

Author:

Tauana Junqueira da Cunha
tauanajc@gmail.com
tauana.cunha@usp.br

References:

GEONet Names Server. National Geospatial-Intelligence Agency (NGA). Último acesso em 23 de abril de 2012. Disponível em:
<http://earth-info.nga.mil/gns/html/index.html>

Hijmans, R. 2012. DIVA-GIS. Último acesso em 23 de abril de 2012. Disponível em: <http://www.diva-gis.org/>

Download do Banco de Bados do Brasil e outros países em:
<http://earth-info.nga.mil/gns/html/namefiles.htm>

Download do mapa (shapefile) de divisões políticas do Brasil em:
<http://www.diva-gis.org/datadown>

"Toponymic information is based on the Geographic Names Database, containing official standard names approved by the United States Board on Geographic Names and maintained by the National Geospatial-Intelligence Agency. More information is available at the Maps and Geodata link at www.nga.mil. The National Geospatial-Intelligence Agency name, initials, and seal are protected by 10 United States Code Section §445."

See Also:

`readShapePoly` do pacote `maptools` (também necessário pacote `sp`) para a construção do mapa.

Examples:

```
geo.coordinates(c("Casa Branca", "Sorocaba", "Tapiraí")) # Fornecendo x  
como vetor.
```

```
# Baixe o arquivo "Localidades Teste.txt" para rodar os exemplos a seguir  
e salve-o no diretório de trabalho que será usado no R.
```

```
geo.coordinates(x="Localidades Teste.txt", output="Coordenadas Teste",  
unit="dec", class=T) # Gera tabela com coordenadas em decimais e  
classificação das localidades e mapa nos arquivos Coordenadas Teste.txt e  
Coordenadas Teste_Map.pdf
```

```
coordenadas.teste <- geo.coordinates(x="Localidades Teste.txt",  
unit="dms") # Guarda o data.frame produzido no objeto coordenadas no  
workspace do R e gera tabela de coordenadas em graus, minutos, segundos  
salva no arquivo Output.txt
```

Código da Função geo.coordinates

```
geo.coordinates <- function(x, output="Output", unit="dec", type=F, short=F,  
generic=F, class=F, elev=F, note=F)  
{  
  
if(missing(x)) # Se nao fornece arquivo de entrada  
{stop("Você deve fornecer um arquivo de localidades para a função  
(argumento x).\nConsulte o help para saber como deve ser o arquivo.")}  
  
# Caso o usuario digite errado a unidade de coordenada desejada:  
if(!(unit %in% c("dms", "dec", "mgrs"))) #  
if(unit=="dms"|unit=="dec"|unit=="mgrs") - opcao  
{stop("É necessário escolher umas das opções de unidade de coordenadas no  
argumento unit: dms, dec ou mgrs. Confira a grafia")}  
  
#####  
##### Importando e organizando database #####  
  
if(!(any("zip"==dir(getwd())))) # Para nao criar pasta zip de novo caso ja  
exista  
{dir.create("zip")}  
  
work.dir <- getwd() # Salvar o diretorio de trabalho  
setwd("zip") # Mandar para pasta criada para a descompressao de arquivos  
  
if(!(any("br.zip"==dir(getwd())))) # Para nao perder tempo baixando caso ja  
o tenha feito  
{download.file("http://earth-info.nga.mil/gns/html/cntyfile/br.zip",  
"br.zip")  
unzip("br.zip")}
```

```
database <- read.delim("br.txt", fileEncoding="utf8") # Lendo Banco de Dados

setwd(work.dir) # Voltando ao diretorio de trabalho

database <- database[,c(4,5,6,7,8,10,16,18,20,21,23,28)] # Apenas colunas de
interesse
colnames(database) <- c("lat_dec", "long_dec", "lat_dms", "long_dms",
"coord_mgrs", "class", "elev", "type", "short", "generic", "full", "note") #
Reduzindo nome das colunas

### Trocando tudo para letras minusculas nas colunas full e generic
for(m in 10:11)
{
  database[,m] <- tolower(database[,m])
}

### Substituindo acentuacao nas colunas full e generic
letras.1 <- c("á", "à", "ã", "â", "ä", "é", "è", "ê", "í", "ó", "ò", "õ",
"ô", "ú", "ü", "ñ", "ç")
letras.2 <- c("a", "a", "a", "a", "a", "e", "e", "e", "i", "o", "o", "o",
"o", "u", "u", "n", "c")

for(l in 1:length(letras.1))
{
  for(v in 10:11)
  {
    tempo <- grep(letras.1[l], database[, v])
    if(length(tempo)!=0)
      {database[tempo, v] <- gsub(letras.1[l], letras.2[l], database[tempo,
v])}
  }
}

##### Fim do Importando e organizando database #####
#####
#####
##### Importando e organizando dados #####

if(length(x)==1) # Se da o caminho (um elemento)
{dados.original <- scan(x, "character", sep="\n", skip=1)}

else # Se ja da vetor (mais de um elemento)
{dados.original <- x}

## Minusculas
dados <- tolower(dados.original) # Tambem em minusculas para evitar
problemas na comparacao de caracteres

## Sem acentos
for(l in 1:length(letras.1))
{
```

```
dados[grep(letras.1[l], dados)] <- gsub(letras.1[l], letras.2[l],
dados[grep(letras.1[l], dados)])
}

##### Fim do Importando e organizando dados #####
#####
#####
##### Importando shapefiles para mapa #####

setwd("zip") # Direcionando para a pasta de descompressao

if(!(any("BRA_adm.zip"==dir(getwd())))) # So baixar o mapa se o shapefile
ainda nao existir
{download.file("http://gadm.org/data/shp/BRA_adm.zip", "BRA_adm.zip")
unzip("BRA_adm.zip")}

if(unit=="dec") # Abrindo pacotes para criacao de mapa (para unidade decimal
de coordenadas)
{require("sp")
require("maptools")
brazil.shape <- readShapePoly("BRA_adm1",
proj4string=CRS(as.character(NA)))}

setwd(work.dir) # Voltar ao diretorio de trabalho

##### Fim do Importando shapefile para mapa #####
#####
#####
##### Definindo argumentos #####

colunas <- c(11) # Coluna minima que constara no output: o nome das
localidades

# A escolha dos argumentos determina quais colunas serao adicionadas no
output da funcao:

if(unit=="dec")
{colunas <- c(colunas,1,2)}
if(unit=="dms")
{colunas <- c(colunas,3,4)}
if(unit=="mgrs")
{colunas <- c(colunas,5)}

if(type==T)
{colunas <- c(colunas, 8)}

if(short==T)
{colunas <- c(colunas, 9)}

if(generic==T)
```



```
{colunas <- c(colunas, 10)}

if(class==T)
  {colunas <- c(colunas, 6)}

if(elev==T)
  {colunas <- c(colunas, 7)}

if(note==T)
  {colunas <- c(colunas, 12)}

##### Fim do Definindo argumentos #####
#####
##### Buscando dados no database #####

nlines <- length(dados) # Numero de observacoes/localidades do usuario
parcial <- list() # Lista vazia onde colocar o resultado da busca

## Cruzando tabelas e salvando resultados numa lista com listas
for(b in 1:nlines)
{
  parcial[[b]] <- database[grep(dados[b], database$full), colunas]
}

## Opcoes de resultados encontrados
locais.zero <- integer() # Criando onde guardar os indices de quem nao foi
encontrado
result <- data.frame(full=character(), lat=numeric(), long=numeric()) #
Criando data frame onde jogar os resultados
nome.original <- character() # Para criar coluna com nomes originais no
data.frame final

for(i in 1:length(parcial)) # Percorrendo a tabela de resultados
{
  if(dim(parcial[[i]])[1]==0) # Se encontra uma localidade sem
correspondente no database
  {
    locais.zero <- c(locais.zero, i) # Salva o indice da localidade no
objeto
  }

  if(dim(parcial[[i]])[1]==1) # Se encontra exatamente um registro
  {
    result <- rbind(result, parcial[[i]]) # Joga o registro num data frame
que constituirá o output final
    nome.original <- c(nome.original, dados.original[i]) # Para coluna de
nomes originais
  }

  if(dim(parcial[[i]])[1]>1) # Se encontra mais de uma coordenada para uma
```

```
mesma localidade
{
  show <- data.frame(indice=1:dim(parcial[[i]])[1], parcial[[i]]) #
data.frame de possiveis localidades
  cat("\n")
  write.table(show, quote=F, row.names=FALSE) # Mostra as opcoes
  if(unit=="dec")
  {plot(brazil.shape) # Plota mapa
  text(show[,4:3], labels=as.character(show$indice), col=2)} # Plotando as
possibilidades para ajudar na escolha
  temp <- readline(prompt="\nHá mais de uma opção para essa localidade
(tabela acima).\n0 mapa pode ajudar na identificação (só para unidade
decimal de coordenadas).\nEscolha o registro correto e digite o índice
correspondente\n(digite zero para nenhuma das opções):") # Pedes para
escolher a opcao que quer
  while(!(any(temp==(0:dim(parcial[[i]])[1])))) # Se digitar algo que nao
uma das opcoes, pede de novo!
  {
    cat("\nEscolha uma das opções fornecidas, indicadas pelo índice
numérico:\n\n")
    write.table(show, quote=F, row.names=FALSE)
    temp <- readline(prompt="\nHá mais de uma opção para essa localidade
(tabela acima).\n0 mapa pode ajudar na identificação (só para unidade
decimal de coordenadas).\nEscolha o registro correto e digite o índice
correspondente\n(digite zero para nenhuma das opções):")
  }
  if(temp==0) # Se encontra uma localidade sem correspondente no database
  {
    locais.zero <- c(locais.zero, i) # Salva o indice da localidade no
objeto
  }
  else
  {result <- rbind(result, show[show$indice==temp, c(-1)]) # Salva a
escolha no data.frame final
  nome.original <- c(nome.original, dados.original[i])
  }
}

rownames(result) <- 1:dim(result)[1] # Enumerando as linhas do output
result$original <- nome.original # Criando coluna com nomes originais
result <- result[c(length(result), 1:(length(result)-1))] # Jogando a coluna
de nomes originais para o comeco da tabela

##### Fim do Buscando dados no database #####
#####
#####
##### Salvando e reportando os resultados #####

write.table(x=result, file=paste(output, "txt", sep="."), quote=F, sep="\t")
# Salvando o resultado num arquivo na pasta de trabalho
```

```
if(unit=="dec") # Salvando o mapa resultante na pasta de trabalho
{pdf(paste(output, "Map.pdf", sep="_"))
map <- plot(brazil.shape) # Plotando mapa
text(result[,4:3], labels=rownames(result), col=2) # Plotando dados
dev.off()}

if(length(locais.zero)>0) # Avisando quais localidades nao foram encontradas
{cat("\nAs seguintes localidades não foram encontradas no GEONet Names
Server: ", paste(dados.original[locais.zero],collapse=", "), "\nVocê pode
tentar checar a grafia dos nomes.\n")}

# Avisando o que aconteceu com os resultados
cat("\nA tabela com as coordenadas foi salva em: ", paste(output, "txt",
sep="."), "\nSe você utilizou decimal como unidade das coordenadas
geográficas, um mapa com as localidades encontradas foi salvo em: ",
paste(output, "Map.pdf", sep="_"), "\n\n")
return(result)

} # Fim da Funcao geo.coordinates()! Ufa!
```

Arquivos para Download

Caso haja desconfiguração dos caracteres, tente abrir com um Encoding diferente ou escreva para tauanajc@gmail.com que eu envio em outro formato.

[Código geo.coordinates \(UTF8\)](#)

[Help geo.coordinates \(UTF8\)](#)

[Arquivo Teste \(UTF8\)](#)

Próximo Passo para Dominar o Mundo

Alterar a função, acrescentando um argumento de escolha de país, para que ela baixe o banco de dados de qualquer país no GEONet Names Server!

MUA HA HA HA

Só não colocarei o banco de dados do mundo inteiro porque o arquivo é gigantesco e demoraria muuuuito para fazer o download. Melhor conquistar o mundo país por país...

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

Permanent link:
http://ecor.ib.usp.br/doku.php?id=05_curso_antigo:alunos2012:alunos:trabalho_final:tauana.cunha:start&rev=1597223093

Last update: 2020/08/12 06:04