Eric Y Kataoka

Estudante de mestrado em Botânica, Instituto de Biociências/USP, Laboratório de Sistemática Vegetal, sob orientação da Profa. Lúcia G. Lohmann.

Trabalho com revisão taxonômica, reconstrução filogenética e biogeografia de um gênero de liana Neotropical (*Martinella* Baill.) que pertence à mesma família dos Ipês: árvores que colorem as ruas de São Paulo com suas florações no inverno 😃

Meu projeto de pesquisa está inserido em um projeto mais amplo, de natureza integrativa, e que busca compreender a evolução e estruturação da biota Amazônica e seu ambiente.

Exercícios do curso resolvidos

exec

Trabalho final

Proposta A - "Limpeza de dados para estudos biogeográficos"

Contextualização

Vivemos atualmente na era de 'Big Data'. Basicamente, 'Big Data' pode ser resumido no que chamam de 3Vs: volume, velocidade e variedade dos dados. Nas Ciências Biológicas, repositórios de dados de ocorrência de espécies como o Global Biodiversity Information Facility (GBIF) e o brasileiro SpeciesLink (SpLink), são exemplos importantes de iniciativas com a missão de abrigar e disponibilizar uma enorme quantidade de dados de biodiversidade de forma gratuita e colaborativa. A utilização de dados oriundos desses repositórios requer um passo essencial que consiste na limpeza desses dados, que deve ser anterior à qualquer análise. No entanto, devido ao grande volume de dados, esse processo toma muito tempo, e está sujeito a erros. Uma alternativa é automatizar essa etapa por meio da aplicação de funções no R, por exemplo.

Com isso, a minha proposta A é criar uma função que faça a limpeza dos dados provenientes de repositórios como o GBIF e SpLink, para Plantas, com foco em estudos biogeográficos, por exemplo, Antonelli et al. (2015)

Descrição

Tarefas-padrão da função:

```
1. Concatenar gênero com epíteto específico
2. Eliminar registros sem nome do gênero & epíteto específico - registros
apenas com nome do gênero são mantidos
3. Verificação dos nomes de espécies comparando com um banco de dados
nomenclatural(1) para buscar o nome aceito. Este passo é fundamental, pois
um mesmo táxon pode ter mais de um nome, mas apenas um deles é aceito
(validamente publicado). Empregarei funções, ou modificações destas, do
pacote taxize.
```

Adicionalmente, uma sequência de tarefas de limpeza e visualização de dados poderão ser implementadas nesta função, dependendo dos argumentos fornecidos pelo usuário, a saber:

```
1. Argumento 1 - Eliminar registros que já estejam classificados como 'known
coordinate issues'
2. Argumento 2 - Eliminar registros com valores de coordenadas igual a zero
3. Argumento 3 - Elaboração de um mapa de distribuição para verificação
visual dos pontos de ocorrência
```

Objeto de entrada: arquivo .csv contendo dados de espécies, e que tenham colunas nomeadas de acordo com o padrão DarwinCore, que é utilizado pelo GBIF e SpLink, por exemplo.

Objeto de saída: arquivo .csv contendo as alterações feitas pelo(a) usuário(a) ao dataframe original, mapa de distribuição.

Pacotes com os quais trabalharei para desenvolver esta função

```
rgbif
taxize
speciesgeocodeR
```

(1) International Plant Names Index, The Plant List, Tropicos

Comentários Julia

Oi Eric,

Achei bastante interessante a função e com uma boa aplicabilidade.

Algumas questões importantes porém, para garantir que seja viável:

1- Você tem alguma familiaridade com o pacote speciesgeocodeR ? Provavelmente para executar as funções dele devem ser necessários alguns argumentos relacionados a DATUM das coordenadas geográficas ou algo do tipo, talvez isso deva entrar como argumento na função, ou ser extraído do dataframe de entrada caso ele obtenha esta informação. Esse é um exemplo, mas acho importante checar de que terá o domínio para lidar com as funções do pacote.

2- Acho legal antes de batermos este martelo você checar como os

mapas são gerados para que a questão gráfica em si não emperre o resto da função de realizar suas tarefas. Mas acredito que isso não será um problema.

3- Como será feita esta etapa de checar se o nome da espécie está de acordo com o banco de dados presente na internet (vi que você colocou três referências) ? A função irá acessar o site ? Essa pode ser a parte mais complicada talvez, dependendo do formato que o site foi estruturado.

Acho que seria legal esclarecermos esses pontos para garantir que será viável. Mas a ideia me parece bem legal e desafiadora em um certo nível.

Qualquer coisa meu e-mail é juliambmolina@gmail.com

Bjs

Resposta aos comentários

Olá, Julia,

Muito obrigado pelas considerações a respeito da minha proposta.

Vamos aos pontos:

1. Eu não tenho familiaridade com o speciesgeocodeR, mas li um pouco da documentação do pacote, e acredito que será viável empregá-lo na minha função.

2. Você se refere aos pacotes requeridos para gerar mapas? Eu tive a oportunidade de fazer um curso de análise de dados espaciais usando o R. Penso que essa parte da função não será um impedimento...

3. Isso mesmo! A ideia é que a função faça uma busca em um dos três sites que utilizei como exemplo de repositórios de informações sobre nomenclatura de taxons de plantas. Para ser sincero, essa é a parte que vai requerer mais estudo para que eu consiga executar. A minha ideia é partir do pacote taxize, que tem uma função que faz coisas muito similares.

Obrigado novamente!

Saudações

Proposta B (Alternativa) - Classificação morfológica de folhas

Contextualização

Em estudos de taxonomia vegetal, descrever a forma dos diferentes órgãos da planta é um grande

desafio. As plantas possuem grande plasticidade fenotípica, principalmente nas folhas. Com isso, uma forma padronizada e reproduzível para descrever a forma das folhas é essencial para os estudos taxonômicos. Uma referência clássica, muito utilizada por taxonomistas de plantas para descrever a forma das folhas foi proposta por Leo J. Hickey (1973). De acordo com essa proposta, as categorias são definidas com base na posição do eixo de maior largura da folha e na razão entre comprimento e largura da lâmina foliar no ponto de maior largura, como ilustrado na figura extraída de Hickey (1973):

As linhas tracejadas indicam o eixo de maior largura da folha.

Descrição

A proposta B de função fará a classificação da forma foliar de acordo com Hickey (1973) nas seguintes categorias:

- oblonga
- elíptica
- ovada
- obovada
- forma especial

Obs.: Vale ressaltar que cada uma dessas categorias possuem sub-divisões (pelo menos cinco em cada categoria), que serão incorporadas à função, mas que não apresento aqui para simplificar a apresentação da proposta.

Objeto de entrada: dataframe com os dados organizados da seguinte maneira:

Medidas de comprimento e largura em milímetros (mm).

A segunda coluna do dataframe de entrada só poderá ter quatro diferentes string de caracteres, que correspondem à posição do eixo de maior largura da folha, e deverão estar de acordo com o exemplo fornecido acima. Assim, a função fará uma verificação desta coluna. Caso haja alguma inconsistência, retornará uma mensagem ao usuário(a) apontando o problema e pedindo correção.

Objeto de saída: dataframe com a classificação da forma da lâmina foliar para cada espécie.

Comentários Julia

Achei esta proposta menos elaborada, mas que seria viável caso os argumentos fossem mais bem trabalhados ou explicitados aqui dando alguma flexibilidade ao usuário.

Veja se entendi certo : A função irá utilizar as três informações no data frame para categorizar a folha certo ? Então por exemplo, uma largura de intervalo X a Y, associada a um comprimento de um intervalo Z a W, e uma posição por exemplo “mediana” do maior eixo (axis) corresponderá a uma certa classificação. E por ai vai.. A função terá um garabito embutido nela é isso ?

Acho válido mas poderia trabalhar melhor os argumentos.

A princípio a proposta A me parece mais promissora porém precisamos garantir aqueles pontos levantados no meu comentário para a proposta A.

Bjs

Resposta aos comentários

Obrigado novamente pelos comentários, Julia.

Vamos aos esclarecimentos...

A função irá utilizar as três informações no data frame para categorizar a folha certo ? Isso mesmo! Mas a ideia é que essa classificação é baseada na razão entre comprimento e largura da folha **&** na posição do eixo de maior largura da folha.

A função terá um garabito embutido nela é isso ? Sim. Por exemplo, uma folha com razão comprimento/largura de 2:1 **&** o eixo de maior largura na porção do meio da folha, será classificada como uma folha oblonga.

Concordo que a proposta A esteja mais bem elaborada e mais interessante e desafiadora (). E também não vejo muito claramente como eu poderia aprimorar a proposta B, já que o que eu pretendo fazer está praticamente todo descrito na proposta.

Abraço!

Comentários Julia - 09/junho

Oi Éric,

Acho que pode prosseguir com a proposta A então. Se já tem familiaridade com dados espaciais fico mais tranquila. E sobre o pacote que não utilizou ainda acho que deve achar orientações no StackOverflow ou então não exite em escrever no forum assim que puder. Pode me escrever aqui ou no meu e-mail também, nunca utilizei o pacote mas estamos aqui para aprender juntos. É uma proposta desafiadora mas bem completa e bem legal!

Cheque bem essa parte de acessar os sites (se possível hoje) para não prometer algo que no final pode virar um problema maior do que você esperava. De uma olhada nos pacotes RCurl e XML ou o rvest, talvez ai encontre algo.

Aconselho:

- Utilizar somente um site ver se consegue acessar o link do site que contem os nomes de espécies direto pelo R (as vezes é um link que não pode ser acessado sem antes passar por outra pagina do site. Ai você provavelmente utilizaria a função 'getURL' ou o 'getForm' e em seguida o 'htmlTreeParse' para transformar essa página da internet

salva em um objeto em uma leitura HTML.

- Cheque bem se acessar essas páginas destes sites (o que você escolher) são acessíveis em HTML ou outro formato e viabilizam seu trabalho. Tente fazer um teste rápido rodando algumas linhas de script fora da função mesmo.

- Aqui tem um tutorial interessante de recuperar pesquisas feitas no google ... Acho que já ajuda a traçar um caminho para algo análogo ao que você quer ;) https://ryouready.wordpress.com/2009/01/01/r-retrieving-information-from-google-with-rcurl-package/

Sugestões:

- A consulta a nomenclatura na internet pode até mesmo ser um argumento opcional. As vezes a pessoa que utilizará a função não quer que essa consulta seja feita porque já sabe que tem espécies novas que não estão na lista do site ou seja lá o que for.

- Sua função poderia retornar uma lista dos nomes que foram encontrados diferentes do que existe na internet. Se quiser até já trocar para o nome correto (retirado da internet) no banco de dados da pessoa e salvar em um novo arquivo retornando uma mensagem de quais nomes foram modificados.

Bom trabalho!

Resposta aos comentários

Oi, Julia,

Muito obrigado por todas as sugestões!

No momento estou tentando acessar os dados do site que pretendo consultar na minha função, mas não estou tendo muito êxito. Tenho algumas alternativas já em mente do que eu poderia usar para contornar esse problema, e qualquer coisa eu te escrevo.

Abraços!

Comentários Julia - 18/06

Oi Eric, tudo bem?

Como estão caminhando as coisas ?

Qualquer coisa lembre de nos mandar um e-mail ou dar um grito no forum ;) Sua proposta é muito legal mas sabemos que envolve um nível considerável de complexidade.

Beijos!

FUNÇÃO - UpCleanPlantBiogeo

Arquivos que serão empregados para o exemplo da função

Arquivo 1. sample_splink.csv
Arquivo 2. sample_gbif.csv

Arquivos adicionais

Código da função. function_code_upcleanplantbiogeo_v-final.r
Help da função. help-upcleanplantbiogeo.txt

Código da função

```
############## FUNCAO UpCleanPlantBiogeo ##############

### Esta funcao atualiza os nomes de especies de plantas recuperadas de
repositorios de dados de biodiversidade mundiais, faz limpeza de dados de
ocorrencia (coordenadas geograficas) e plota os pontos de ocorrencia em um
mapa. Para maiores detalhes, consulte o help da funcao.

UpCleanPlantBiogeo = function(x, from="spLink", cleanMap=TRUE)
#Funcao UpCleanPlantBiogeo, argumentos: x=data.frame com os dados de
entrada, from="spLink" ou "GBIF" indicando origem dos dados, cleanMap para
realizar limpeza dos dados de ocorrência e plotar os pontos de ocorrencia em
um mapa.
    {

    if(is.data.frame(x) == FALSE)        #Verifica a condicao de que arquivo
de entrada eh um data.frame, senao a funcao nao eh executada
        {
        stop("O objeto fornecido nao eh um data.frame, corrija e tente
novamente")        #caso o arquivo de entrada nao seja um data.frame,
retorna mensagem de erro ao usuario
        }

    if(from == "spLink" & cleanMap == TRUE)        #Teste condicional para
os argumentos da funcao, se from="spLink" e cleanMap = TRUE (padrao),
executa os seguintes comandos
        {
        #indexar o dataframe de entrada, pois contem muito mais informacoes
do que o necessario
```

```
        data1 = x[,c("family", "genus", "species", "subspecies",
"longitude", "latitude")]        #Cria o objeto data1, contendo apenas as
colunas de interesse do data.frame de input
        #eliminando registros sem nenhuma identificacao
        data2 = data1[!(data1$genus==""),]        # Cria o objeto data2
contendo apenas os registros com informacao nas linhas da coluna $genus que
nao sejam espacos em branco

        #inserindo "sp." nos registros que contem apenas nomes de generos
        data2$species[data2$species==""] = "sp."        #Na coluna $species,
sempre que houver um espaco em branco, este sera substituido por "sp."

        #concatenando as colunas 'genus', 'species' e 'subspecies' em uma
nova coluna, chamada "scientificname"
        data2$scientificname = paste(data2$genus, data2$species,
data2$subspecies)
        #verificando os nomes contidos na coluna "scientificname" -
sinonimias e nomes aceitos, empregando a funcao plantminer()
        library(taxize)        #carrega o pacote taxize
        data3 = plantminer(data2$scientificname, from="tpl")    #executa a
funcao plantminer do pacote taxize

        #a funcao plantminer() gera um output que nao mantem os registros
identificados somente ate o nome de genero
        #para contornar isso, a seguinte operacao eh necessaria. Busca, na
coluna $note, pelo string de caracteres "not full name", que eh
correspondente, em termos de posicao, a coluna $name, e entao coloca a
informacao da celula da coluna $original.search que tiver o string de
caracteres = "sp." estritamente e armazena na coluna $name.
        data3$name[grep("not full name", data3$note)] =
data3$original.search[grep("\\bsp.\\b", data3$original.search)]

        #incorporando as informacoes de coordenadas geograficas no output do
plantminer
        data3$longitude = data2$longitude        #criando uma nova coluna
para armazenar os dados de coordenadas geograficas no objeto de output do
plantminer()
        data3$latitude = data2$latitude        #criando uma nova coluna para
armazenar os dados de coordenadas geograficas no objeto de output do
plantminer()

        #alguns nomes nao sao encontrados na busca, pois devem ser
identificacoes com nomes invalidos, o proximo comando elimina esses
registros do data frame
        data4 = data3[!(data3$name==""),]        #Cria o objeto data4, com
todas as colunas do data3, e elimina as linhas que contenham espaços em
branco na coluna $name do data3
        #Indexando o dataframe, pois as informacoes necessarias sao: coluna
"name", longitude e latitude
        data5 = data4[,c("name", "longitude", "latitude")]        #Cria o
```

```
objeto data5, contendo apenas tres colunas

        #eliminando os registros sem dados de coordenadas geograficas
        dataF = data5[!(is.na(data5$longitude))|!(is.na(data5$latitude)),]
#cria o data.frame que será retornado na tela do usuário, eliminando os
registros que nao possuem dados de coordenadas geograficas
        write.csv(dataF, file="UpCleanPlantBiogeo_output1.csv")
#salva um arquivo csv no diretório de trabalho contendo os dados limpos
        #organizando um 'report' com o numero de registros inicial, numero
de registros final, o numero de registros que foram eliminados por nao
possuirem identificacao e coordenadas geograficas
        i.recs = dim(data1)[1]        #armazena o tamanho, em numero de
linhas, do objeto data1 em i.recs (initial number of records)
        f.recs = dim(dataF)[1]        #armazena o tamanho, em numero de
linhas, do objeto dataF em f.recs (final number of records)
        noID = dim(data1)[1] - dim(data2)[1]        #calcula o numero de
registros sem identificacao que foram eliminados durante o processo de
limpeza e checagem por nomes aceitos
        noCoords = dim(data2)[1] - dim(dataF)[1]        #calcula o numero de
registros sem informacao de coordenadas geograficas que foram retiradas
        UpCleanReport = c(i.recs, f.recs, noID, noCoords)        #organiza
as informacoes para o report no objeto vetorial UpCleanReport
        names(UpCleanReport) = c("initial # of records", "final # of
records", "# of unidentified records", "# of records without geo
coordinates")        #Nomeia cada elemento do vetor

        #plotando os pontos de ocorrencia em um mapa para verificacao visual
        library(maptools)        #Carrega um dos pacotes disponiveis para
lidar com dados espaciais no R
        data(wrld_simpl)        #Carrega o dado wrld_simpl, que contem um
shapefile da Terra, um mapa mundi
        plot(wrld_simpl, xlim = c(min(dataF$longitude),
max(dataF$longitude)), ylim = c(min(dataF$latitude),max(dataF$latitude)),
axes=TRUE)        #Abre um dispositivo grafico e plota um recorte do mapa,
de acordo com a extensao dos proprios pontos de ocorrencia do data.frame,
assim, facilita a visualizacao dos pontos de ocorrencia no mapa
        points(dataF$longitude, dataF$latitude, pch=19, col="black",
cex=0.5)        #Plota os pontos de ocorrencia no mapa
        }
    if(from == "spLink" & cleanMap == FALSE)        #Teste condicional para
os argumentos da funcao, se from="spLink" e cleanMap = FALSE, executa os
seguintes comandos
        {
        #indexar o dataframe de entrada, pois contem muito mais informacoes
do que o necessario
        data1 = x[,c("family", "genus", "species", "subspecies",
"longitude", "latitude")]        #Cria o objeto data1, contendo apenas as
colunas de interesse do data.frame de input
        #eliminando registros sem nenhuma identificacao
        data2 = data1[!(data1$genus==""),]        # Cria o objeto data2
contendo apenas os registros com informacao nas linhas da coluna $genus que
```

```
nao sejam espacos em branco

        #inserindo "sp." nos registros que contem apenas nomes de generos
        data2$species[data2$species==""] = "sp."        #Na coluna $species,
sempre que houver um espaco em branco, este sera substituido por "sp."

        #concatenando as colunas 'genus', 'species' e 'subspecies' em uma
nova coluna, chamada "scientificname"
        data2$scientificname = paste(data2$genus, data2$species,
data2$subspecies)
        #verificando os nomes contidos na coluna "scientificname" -
sinonimias e nomes aceitos, empregando a funcao plantminer()
        library(taxize)        #carrega o pacote taxize
        data3 = plantminer(data2$scientificname, from="tpl")    #executa a
funcao plantminer do pacote taxize

        #a funcao plantminer() gera um output que nao mantem os registros
identificados somente ate o nome de genero
        #para contornar isso, a seguinte operacao eh necessaria. Busca, na
coluna $note, pelo string de caracteres "not full name", que eh
correspondente, em termos de posicao, a coluna $name, e entao coloca a
informacao da celula da coluna $original.search que tiver o string de
caracteres = "sp.", estritamente, e armazena na coluna $name.
        data3$name[grep("not full name", data3$note)] =
data3$original.search[grep("\\bsp.\\b",data3$original.search)]
        #alguns nomes nao sao encontrados na busca, pois devem ser
identificacoes com nomes invalidos, o proximo comando elimina esses
registros do data frame
        dataF = data3[!(data3$name==""),]        #Cria o objeto data4, com
todas as colunas do dataF, e elimina as linhas que contenham espaços em
branco na coluna $name do data3
        write.csv(dataF, file="UpCleanPlantBiogeo_output2.csv")
#salva um arquivo csv no diretorio de trabalho contendo os dados limpos
        #organizando um 'report' com o numero de registros que foram
eliminados por nao possuirem identificacao e coordenadas geograficas
        i.recs = dim(data1)[1]        #armazena o tamanho, em numero de
linhas, do objeto data1 em i.recs (initial number of records)
        f.recs = dim(dataF)[1]        #armazena o tamanho, em numero de
linhas, do objeto dataF em f.recs (final number of records)
        noID = dim(data1)[1] - dim(data2)[1]        #calcula o numero de
registros sem identificacao que foram eliminados durante o processo de
limpeza e checagem por nomes aceitos
        UpCleanReport = c(i.recs, f.recs, noID)        #organiza as
informacoes para o report no objeto vetorial UpCleanReport
        names(UpCleanReport) = c("initial # of records", "final # of
records", "# of unidentified records")        #Nomeia cada elemento do vetor
        }

    if(from=="GBIF" & cleanMap == TRUE)        #Teste condicional para os
argumentos da funcao, se from="GBIF" e cleanMap = TRUE (padrao), executa os
```

```
seguintes comandos
        {
        #indexar o dataframe de entrada, pois contem muito mais informacoes
do que o necessario
        data1 = x[,c("family", "genus", "specificEpithet", "lon", "lat")]
#Cria o objeto data1, contendo apenas as colunas de interesse do data.frame
de input
        #eliminando registros sem nenhuma identificacao
        data2 = data1[!is.na(data1$genus),]        # Cria o objeto data2
contendo apenas os registros com informacao nas linhas da coluna $genus que
nao sejam NA

        #inserindo "sp." nos registros que contem apenas nomes de generos
        data2$specificEpithet[is.na(data2$specificEpithet)] = "sp."
#Na coluna $specificEpithet, sempre que houver um NA, este sera substituido
por "sp."

        #concatenando as colunas 'genus' e 'species' em uma nova coluna,
chamada "scientificname"
        data2$scientificname = paste(data2$genus, data2$specificEpithet)
        #verificando os nomes contidos na coluna "scientificname" -
sinonimias e nomes aceitos, empregando a funcao plantminer()
        library(taxize)        #carrega o pacote taxize
        data3 = plantminer(data2$scientificname, from="tpl")    #executa a
funcao plantminer do pacote taxize

        #a funcao plantminer() gera um output que nao mantem os registros
identificados somente ate o nome de genero
        #para contornar isso, a seguinte operacao eh necessaria. Busca, na
coluna $note, pelo string de caracteres "not full name", que eh
correspondente, em termos de posicao, a coluna $name, e entao coloca a
informacao da celula da coluna $original.search que tiver o string de
caracteres = "sp.", estritamente, e armazena na coluna $name.
        data3$name[grep("not full name", data3$note)] =
data3$original.search[grep("\\bsp.\\b",data3$original.search)]
        #incorporando as informacoes de coordenadas geograficas no output do
plantminer
        data3$longitude = data2$lon        #criando uma nova coluna para
armazenar os dados de coordenadas geograficas no objeto de output do
plantminer()
        data3$latitude = data2$lat        #criando uma nova coluna para
armazenar os dados de coordenadas geograficas no objeto de output do
plantminer()
        #alguns nomes nao sao encontrados na busca, pois devem ser
identificacoes com nomes invalidos, o proximo comando elimina esses
registros do data frame
        data4 = data3[!(data3$name==""),]        #Cria o objeto data4, com
todas as colunas do data3, e elimina as linhas que contenham espaços em
branco na coluna $name do data3
        #Indexando o dataframe, pois as informacoes necessarias sao: coluna
"name", longitude e latitude
```

```
        data5 = data4[,c("name", "longitude", "latitude")]        #Cria o
objeto data5, contendo apenas tres colunas

        #eliminando os registros sem dados de coordenadas geograficas
        dataF = data5[!(is.na(data5$longitude))|!(is.na(data5$latitude)),]
#cria o data.frame que será retornado na tela do usuário, eliminando os
registros que nao possuem dados de coordenadas geograficas
        write.csv(dataF, file="UpCleanPlantBiogeo_output3.csv")
#salva um arquivo csv no diretório de trabalho contendo os dados limpos

        #organizando um 'report' com o numero de registros que foram
eliminados por nao possuirem identificacao e coordenadas geograficas
        i.recs = dim(data1)[1]        #armazena o tamanho, em numero de
linhas, do objeto data1 em i.recs (initial number of records)
        f.recs = dim(dataF)[1]        #armazena o tamanho, em numero de
linhas, do objeto dataF em f.recs (final number of records)
        noID = dim(data1)[1] - dim(data2)[1]        #calcula o numero de
registros sem identificacao que foram eliminados durante o processo de
limpeza e checagem por nomes aceitos
        noCoords = dim(data2)[1] - dim(dataF)[1]        #calcula o numero de
registros sem informacao de coordenadas geograficas que foram retiradas
        UpCleanReport = c(i.recs, f.recs, noID, noCoords)        #organiza
as informacoes para o report no objeto vetorial UpCleanReport
        names(UpCleanReport) = c("initial # of records", "final # of
records", "# of unidentified records", "# of records without geo
coordinates")        #Nomeia cada elemento do vetor
        #plotando os pontos de ocorrencia em um mapa para verificacao visual
        library(maptools)        #Carrega um dos pacotes disponiveis para
lidar com dados espaciais no R
        data(wrld_simpl)        #Carrega o dado wrld_simpl, que contem um
shapefile da Terra, um mapa mundi
        plot(wrld_simpl, xlim = c(min(dataF$longitude),
max(dataF$longitude)), ylim = c(min(dataF$latitude),max(dataF$latitude)),
axes=TRUE)        #Abre um dispositivo grafico e plota um recorte do mapa,
de acordo com a extensao dos proprios pontos de ocorrencia do data.frame,
assim, facilita a visualizacao dos pontos de ocorrencia no mapa
        points(dataF$longitude, dataF$latitude, pch=19, col="black",
cex=0.5)        #Plota os pontos de ocorrencia no mapa
        }

    if(from=="GBIF" & cleanMap == FALSE)        #Teste condicional para os
argumentos da funcao, se from="GBIF" e cleanMap = FALSE, executa os
seguintes comandos
        {
        #indexar o dataframe de entrada, pois contem muito mais informacoes
do que o necessario
        data1 = x[,c("family", "genus", "specificEpithet", "lon", "lat")]
#Cria o objeto data1, contendo apenas as colunas de interesse do data.frame
de input
        #eliminando registros sem nenhuma identificacao
```

```
        data2 = data1[!is.na(data1$genus),]        # Cria o objeto data2
contendo apenas os registros com informacao nas linhas da coluna $genus que
nao sejam NA

        #inserindo "sp." nos registros que contem apenas nomes de generos
        data2$specificEpithet[is.na(data2$specificEpithet)] = "sp."
#Na coluna $specificEpithet, sempre que houver um NA, este sera substituido
por "sp."

        #concatenando as colunas 'genus' e 'species' em uma nova coluna,
chamada "scientificname"
        data2$scientificname = paste(data2$genus, data2$specificEpithet)
        #verificando os nomes contidos na coluna "scientificname" -
sinonimias e nomes aceitos, empregando a funcao plantminer()
        library(taxize)        #carrega o pacote taxize
        data3 = plantminer(data2$scientificname, from="tpl")    #executa a
funcao plantminer do pacote taxize

        #a funcao plantminer() gera um output que nao mantem os registros
identificados somente ate o nome de genero
        #para contornar isso, a seguinte operacao eh necessaria. Busca, na
coluna $note, pelo string de caracteres "not full name", que eh
correspondente, em termos de posicao, a coluna $name, e entao coloca a
informacao da celula da coluna $original.search que tiver o string de
caracteres = "sp.", estritamente, e armazena na coluna $name.
        data3$name[grep("not full name", data3$note)] =
data3$original.search[grep("\\bsp.\\b",data3$original.search)]
        #alguns nomes nao sao encontrados na busca, pois devem ser
identificacoes com nomes invalidos, o proximo comando elimina esses
registros do data frame
        dataF = data3[!(data3$name==""),]        #Cria o objeto dataF, com
todas as colunas do data3, e elimina as linhas que contenham espaços em
branco na coluna $name do data3
        write.csv(dataF, file="UpCleanPlantBiogeo_output4.csv")
#salva um arquivo csv no diretório de trabalho contendo os dados limpos
        #organizando um 'report' com o numero de registros que foram
eliminados por nao possuirem identificacao e coordenadas geograficas
        i.recs = dim(data1)[1]        #armazena o tamanho, em numero de
linhas, do objeto data1 em i.recs (initial number of records)
        f.recs = dim(dataF)[1]        #armazena o tamanho, em numero de
linhas, do objeto dataF em f.recs (final number of records)
        noID = dim(data1)[1] - dim(data2)[1]        #calcula o numero de
registros sem identificacao que foram eliminados durante o processo de
limpeza e checagem por nomes aceitos
        UpCleanReport = c(i.recs, f.recs, noID)        #organiza as
informacoes para o report no objeto vetorial UpCleanReport
        names(UpCleanReport) = c("initial # of records", "final # of
records", "# of unidentified records")        #Nomeia cada elemento do vetor
        }

    return(list(dataF, UpCleanReport))        #Retorna uma lista contendo
```

```
(1) os dados limpos e (2) o report com o numero de registros processados

    }
```

Help da função

```
UpCleanPlantBiogeo                package:unknown               R
Documentation

UpCleanPlantBiogeo processes data gathered from global biodiversity data
repositories such as the Global Biodiversity Information Facility (GBIF) and
speciesLink (spLink) for plant biogeographical studies.
It Updates plant taxon names, according to the accepted names on The Plant
List's database, and Cleans species occurrence data (coordinates).

Description:

    UpCleanPlantBiogeo takes a dataframe containing plant species occurrence
data, checks for species names synonymy and accepted names on The Plant
List's database, generating an updated dataframe with the accepted names.
    Depending on the user's choice, the function also cleans geographic
coordinates (i.e., removes records without geographic coordinates, and with
value = 0) and plots them on a map for visual inspection, in order to readly
spot potential outliers and/or records with erroneous geographic
coordinates.

Usage:

    UpCleanPlantBiogeo(x, from = "spLink", cleanMap = TRUE)

Arguments:
    x            data.frame, downloaded either from GBIF or speciesLink
(spLink).
    from            character (required). Specify from which repository the
data (x) has been retrieved: from = "GBIF" or from = "spLink". Default =
"spLink".
    cleanMap        logical. If TRUE, cleans geographic coordinates that are
missing from each record (row) and plots a map for visual inspection of
species occurrence points. If FALSE, does not perform any of the steps
previously described.

Details:
    The input data.frame must contain columns named "family", "genus",
"species", "specificEpithet", "subspecies", "lon", "longitude", "lat",
"latitude",  it will depend on the data source. Data from the GBIF must have
at least columns named: "family", "genus", "specificEpithet", "lon", "lat".
```

```
Data from spLink must have at least columns named "family", "genus",
"species", "subspecies", "longitude", "latitude".
    Input files must be downloaded either from speciesLink
(http://www.splink.org.br/index?lang=pt) or GBIF (http://www.gbif.org/). The
function will only work on GBIF data downloaded via gbif() function from
dismo package.
    The mapping step is only going to work properly if geographic
coordinates are in decimal degrees, which is the default geographic
coordinate format provided by the GBIF and spLink. So make sure the input
file contains the correct format of geographic coordinates.
    Before running UpCleanPlantBiogeo, make sure to install the following
packages: taxize and maptools
Value:
    An output file will be saved on the working directory in csv format.
    The function returns a list containing (1) a dataframe with all the
accepted names, and (2) a vector containing a very simple report with the
number of records that have been removed from the original dataframe for
being unidentified records and/or without geographic coordinates, depending
on the arguments provided to the function.
    A map will be returned as a graphical device on the user's screen.

Warning:

    The function will stop and print an error message if the input file is
not a data.frame.

    The function will only work on GBIF data downloaded via the gbif()
function from dismo package.
    Geographic coordinates must be in decimal degrees, otherwise the mapping
step will be compromised.

Author(s):

    Eric Yasuo Kataoka, erickataoka@usp.br

References:
    1. Global Biodiversity Information Facility - GBIF website
(http://www.gbif.org)
    2. speciesLink website (http://splink.cria.org.br )

See Also:

    speciesgeocodeR, taxize, dismo and rgbif packages

Examples:
    # Remember: make sure you have installed taxize and maptools packages
before running UpCleanPlantBiogeo
    ### Example1, data downloaded from spLink, default arguments
    ## Not run:
    sample.spLink = read.csv("sample_spLink.csv", header = TRUE, sep = ",",
dec = ".", as.is=T)
```

```
    UpCleanPlantBiogeo(sample.spLink, from = "spLink")
    ## End (Not run)
    ### Example 2, data downloaded from spLink, from = "spLink" and cleanMap
= FALSE
    ## Not run:
    sample.spLink = read.csv("sample_spLink.csv", header = TRUE, sep = ",",
dec = ".", as.is=T)
    UpCleanPlantBiogeo(sample.spLink, from = "spLink", cleanMap = FALSE)
    ## End (Not run)
    ### Example3, data downloaded from GBIF, default arguments
    ## Not run:
    sample.GBIF = read.csv("sample_GBIF.csv", header = TRUE, sep = ",", dec
= ".", as.is=T)
    UpCleanPlantBiogeo(sample.GBIF, from = "GBIF")
    ## End (Not run)
    ### Example4, data downloaded from GBIF, from = "spLink" and cleanMap =
FALSE
    ## Not run:
    sample.GBIF = read.csv("sample_GBIF.csv", header = TRUE, sep = ",", dec
= ".", as.is=T)
    UpCleanPlantBiogeo(sample.GBIF, from = "GBIF", cleanMap = FALSE)
    ## End (Not run)
```

From:
http://ecor.ib.usp.br/ - **ecoR**

Permanent link:
http://ecor.ib.usp.br/doku.php?id=05_curso_antigo:r2017:alunos:trabalho_final:ericyasuok:start

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