EDUARDA ROMANINI

Mestre em Ecologia e Recursos Naturais pela Universidade federal de São Carlos (UFSCar) e doutoranda no programa de pós graduação em Ecologia da Universidade de São Paulo(USP).

O título da minha tese é: Influência da dinâmica dos usos da terra e das variações sazonais de biomassa agrícola sobre a qualidade da água no estado de São Paulo, sob orientação do Prof. Dr. Leandro R. Tambosi.

Meus exercícios

Link para os meus exercícios resolvidos: Exercícios

Proposta de Trabalho Final

Plano A - Detecção de mudanças de usos da terra

Contextualização

Alterações no uso e ocupação do solo podem afetar ciclos biogeoquímicos – como parâmetros da qualidade da água (Uriarte et al., 2011) , movimentação de espécies – por exemplo, cross habitat spillover de aves (Boesing, Nichols, Metzger, 2017) , regulação da temperatura do ar (Rocha e Fialho, 2010), emissão de gases (Neto et al., 2011), entre tantos outros exemplos. No entanto, diferentes tipos de usos da terra podem ter diferentes influências no parâmetro analisado e a dinâmica da mudança de um uso específico para outro uso de interesse, também pode tem um efeito diferencial.

Nesse sentido, uma função que consiga selecionar na paisagem mudanças entre usos da terra mostra-se relevante, podendo evidenciar quais locais passaram por essas mudanças e/ou auxiliar na escolha de locais de estudos para aqueles que desejam analisar efeitos de mudanças de usos mais específicas. Esta função seleciona as mudanças de uso de interesse, dentro da paisagem que contem

diferentes locais dentro dela, de acordo com um limiar de mudança (exemplo: ter um limiar de mudança de 30%, podendo ser de diminuição ou aumento da porcentagem do uso da terra de interesse na paisagem estudada).

Mais especificamente, a entrada da função é um data.frame que contenha na primeira coluna os nomes dos locais (que estão inseridos numa paisagem maior); na segunda, os tipos de uso da terra; na terceira, as porcentagens iniciais de cada uso naquele local (no ano j) e, nas colunas seguintes, as porcentagens de cada uso para os m anos seguintes. Sendo assim, o data.frame precisa ter um número de colunas maior ou igual a quatro. A função retorna o nome dos locais nos quais se encontrou o limiar de mudança estabelecido, bem como as porcentagens totais de aumento ou diminuição do uso da terra.

Se possível (caso haja tempo e eu consiga aprender melhor sobre os pacotes que trabalham com dados espaciais no R), a função também poderá olhar para os rasters ou vetores e selecionar as mudanças de uso de interesse, dentro da paisagem, retornando os locais encontrados num mapa e também um data.frame com as porcentagens dos usos nos dois momentos que foram analisados. Nesta função, podem ser selecionadas diferentes mudanças de interesse, mas a entrada dos anos acontece de dois a dois (um ano inicial e um final). Neste caso, o usuário pode inserir arquivos raster ou vetor, selecionar quais as mudanças de interesse, definir o limiar de mudança, bem como o número que ele deseja obter de locais com aquela mudança específica (o R dividiria a paisagem em um mosaico de locais, de formato quadrangular).

Tanto para data.frame como para dados espaciais, os locais encontrados e retornados pela função são escolhidos de maneira aleatória, dentre os locais que possuem as características desejadas dentro da paisagem. Caso o usuário deseje ter como retorno todos os locais, o argumento referente ao número de locais não deve ser especificado.

Planejamento da Função

Entrada: mudancas(data, mud(i to n), anos (j to m), limiar, num, total = FALSO)

- data = data.frame com porcentagens de usos da terra para cada local, em diferentes anos;
- mud (i to n) = usos da terra de interesse;
- anos (j to m) = anos em que se deseja procurar a mudança de uso da terra, se não especificado, função compara todas as colunas;
- limiar = porcentagem mínima de mudança de uso da terra a ser procurada;
- num = numero de locais dentro da paisagem que se deseja ter como retorno, se não especificado, função irá entender que o usuários quer como retorno todos os locais encontrados;
- total = FALSO = argumento lógico (verdadeiro ou falso), que tem como default ser FALSO e indica se o usuário quer comparar o primeiro ano de estudo com o último (útil quando mais de dois anos estão sendo comparados;
- se possível = função também terá argumentos dados1 e dados2, que se referem aos rasters ou vetores do ano incial e final, respectivamente. Se dados1 e dados2 estão presentes, data=NULL, ou seja, data.frame não é necessário.

Verificando os parâmetros

- o data-frame possui o número de colunas necessárias? → se não, retorna mensagem “Número de colunas insuficiente” e para função;
- existe NA nas colunas? → se sim, retorna aviso “NAs presentes” e para função;
- mud (i to n) são classes de uso existentes nos dados inseridos? → se não, retorna mensagem “erro nos tipos de uso” e para função;
- limiar está especificado e é um número entre 0 e 1? → se não, retorna mensagem “limiar deve estar especificado no intervalo 0 ≤ limiar ≤ 1” e para função;
- num está especificado? → se não, retorna mensagem “num não especificado, resultado representa todos os locais encontrados”;
- num é um número dentro do número de divisões possíveis naquela paisagem? → se não, retorna mensagem “erro no número de locais escolhidos”;
- se possível inserir dados1 e dados2: dado1 e dado2 são da mesma origem (raster ou vetor)? → se não, retorna mensagem dizendo “dados possuem origens diferentes” e para função.

Pseudo-código

- verificar todos os parâmetros;
- usando “for”, com ciclos de 1 até o número n de mudanças de interesse, para cada uma das mudanças definidas, dentro de um “for” com ciclos de 1 até o número m de anos estudados, para cada local especificado no data.frame de entrada:

- o nome do local será sempre o indexador;

- olhar a cada transição de ano (ou seja, de uma coluna para a coluna seguinte), se houve a(s) mudança(s) de interesse (olha linha por linha se a porcentagem de uso se manteve e/ou se houve mudança de acordo com o limiar estipulado);

- caso “num” esteja especificado, realizar um processo aleatório para escolha dos locais que serão estudados, de acordo com o número de locais do argumento num;

- caso o número de locais encontrados seja menor ou igual ao estabelecido, retornar mensagem “todos os locais encontrados foram escolhidos, pois o número de locais é menor que o estabelecido” ou “todos os locais encontrados foram escolhidos, pois o número de locais é igual ao estabelecido”;

- criar um data.frame como “return”.

Se possível também trabalhar com os dados espaciais:

- carregar os pacote necessário para ler e trabalhar com dados espaciais no R;
- usar “for”, com ciclos de 1 até o número n de mudanças de interesse, para cada uma das mudanças definidas:

- transformar usos de interesse nos dados em 1 e 2 e demais usos em NA;

- plotar, em uma mesma janela, um mapa só com os dados do uso inicial na paisagem, outro com os dados de usos finais e um com o locais onde a mudança foi detectada (figura apenas para visualização, na será salva);

- procurar na paisagem as mudanças com o limiar determinado;

- definir, aleatoriamente, os locais encontrados que serão retornados na saída (caso argumento “num” tenha sido especificado);

- caso o numero de locais encontrados seja menor ou igual ao estabelecido, retornar mensagem “todos os locais encontrados foram escolhidos, pois o número de locais é menor que o estabelecido” ou “todos os locais encontrados foram escolhidos, pois o número de locais é igual ao estabelecido”;

- criar um data.frame como “return”;

- retornar também um mapa com os locais escolhidos na paisagem (este mapa será salvo como imagem).

Saída

- Para os data.frame, um novo data.frame com os nomes dos locais escolhidos em uma coluna, na coluna seguinte o uso da terra de interesse e nas demais colunas as porcentagens de mudança (para cada ano em que a mudança foi detectada). Se argumento “num” é especificado, função retorna apenas os locais aleatoriamente escolhidos.
- Caso seja possível trabalhar com dados espaciais: Summary para cada local escolhido, com as porcentagens das mudanças de interesse (quanto, em porcentagem, mudou de um ano para o outro) e mapas com os locais escolhidos.

EXEMPLO

Digamos que o usuário da função tenha a seguinte tabela de usos da terra:

Ex 1: Caso o usuário queira encontrar, em sua paisagem, locais em que cana-de-açúcar tenha passado por um limiar de mudança de 40% (seja ganho ou perda de porcentagem de área), a função compara os anos de estudo e mostra como resultado quais os locais passaram por essa mudança.

Os argumentos de entrada seriam “mud = cana” e “limiar = 40”.

No caso, a função encontra que no local 1 ocorreu essa mudança entre 2005 e 2015; no local 3, entre os anos 95 e 2005; e no local 2, não teve a mudança que eu procuro. A função retorna uma tabela com as porcentagens de mudança para os locais que tiveral mudanças maiores ou iguais ao limiar, em pelo menos um dos anos de estudo analisado.

Ex 2: Caso o usuário queira encontrar locais que tenham passado por um limiar de mudança maior ou igual a 30%, tanto para cana como para pasto, a função compara os diferentes anos e procura essa mudança específica.

Os argumentos de entrada seriam “mud = pasto, cana” e “limiar = 30”.

Neste caso, a função encontra que no local 1, para pasto e cana, a mudança ocorreu entre os anos

2005 e 2015. No local 3, ambas as mudanças especificadas ocorreram entre os anos 1995 e 2005, enquanto no local 2 essa mudança não foi encontrada. A função retorna uma tabela com os locais em que a mudança foi encontrada e a porcentagem de mudanças totais dos usos da terra de interesse.

Em abos exemplos, caso o argumento “num” estivesse especificado, a tabea seria de acordo com a escolha aleatória dos locais, porém, o argumento “num” não foi especificado e função retorna todos os locais encontrados.

Oi Eduarda,

A proposta A parece bastante interessante, mas tenho problemas para entender exatamente o que a sua função visa calcular, e mais importante ainda, a estrutura dos objetos que ela receberá como input. Do jeito que está, parece que o data frame de entrada teria mais do que duas dimensões, e vale muito a pena considerar como tais dados serão representados. Antes de continuar gostaria que você reavaliasse esses detalhes tecnicos.

Por favor lembre-se que o aceite da proposta B é obrigatorio, então, veja meus comentários sobre a mesma na seção a seguir. Próximas conversas terão lugar pelo email e uma vez a gente consiga um consenso você deverá fazer as mudanças que tenham lugar no seu wiki.

—*Gustavo A. Ballen*

Plano B - Necessidades alimentares de animais de estimação

Contextualização

A forma e a fase da vida de um animal levam a diferenças em suas necessidades nutricionais (Lazzarotto, 1999), por exemplo, as necessidades energéticas dos filhotes de cães variam com a idade e a raça, sendo que esses requerimentos diminuem conforme o aumento da idade (Lazzarotto, 2001). As necessidades energéticas determinam diretamente a quantidade de alimento que deve ser ingerida por um animal, sendo importante entender melhor a exigência energética de cada um deles (Parreira, 2007).

Da mesma forma que as necessidades energéticas variam para a mesma espécie em diferentes fases da vida e para raças diferentes da mesma espécie, ela também varia entre espécies. Cães e gatos

possuem necessidades nutricionais bastante distintas, com os gatos precisando, por exemplo, de maior número de refeições ao longo do dia (Revista veterinária, 2018).

Nesse sentido, uma função que ajude os amantes de animais a programarem a quantidade correta de alimento a ser dada para cada um dos seus pets é bastante útil. A função calcula a quantidade energética vital para cada indivíduo, a partir da espécie, porte no animal e idade. A função retorna, baseada na necessidade energética, a quantidade de alimento que precisa ser fornecida, com qual frequência diária e quanto se deve comprar do alimento para um determinado período, previamente estipulado por um argumento da função.

Planejamento da função

Entrada: alimento (data, spp, dias)

- data = data.frame que contenha na primeira coluna o número ou nome referente ao animal, na segunda o porte do animal (pequeno, médio ou grande) e na terceira coluna a idade (filhote, adulto, gestante, lactante ou idoso), sendo que cada linha representa um animal diferente;
- spp = cão ou gato;
- dias = para quantos dias pretende-se calcular a quantidade de alimento a ser comprado, pode variar de 1 a 31.

Verificando os parâmetros

- data possui três colunas e duas ou mais linhas com as categoriais aceitas para a função? → se não, retorna mensagem “erros no data.frame” e para função;
- spp é “cão” ou “gato”? → se espécie for diferente, escrever “espécie não reconhecida” e para função;
- o dias é um valor inteiro, da classe numeric e está no intervalo entre 1 e 31? → se não, retorna “problemas com o argumento dias, verificar limite 1 ≤ dias ≤ 31” e para função.

Pseudo-código

- a função tem uma tabela associada a ela em que estão presentes informações a respeito das necessidades nutricionais dos animais e quantidade respectiva de alimento (ração), de acordo com os parâmetros fornecidos no data.frame de entrada;
- a função se inicia com um if(cão)/if(gato);
- em seguida faz um ciclo “for”, que vai de um até o número de linhas do data.frame de entrada, sendo que para cada linha ele calcula as necessidades nutricionais e a quantidade de alimento, de acordo com a tabela que está inserida na função;
- este ciclo tem como “return” um data frame;
- no final, a função soma todas as linhas referentes a quantidade de ração diária de todos os animais e multiplica pelo número no argumento “num”, retornando o valor total de ração que deves se comprada para o período estipulado.

Saída

- lista com um data.frame e um valor final;
- data.frame especifica, em cada linha, as necessidades energéticas diárias de cada animal, a quantidade de ração a ser fornecida e em quantas porções ao longo do dia;
- valor final vem com a mensagem “Você deve comprar X quantidade de ração para atender as necessidades nutricionais dos seus animais por Y dias”, sendo que X é resultado dos cálculos da função e Y é igual ao argumento “dias”.

Bibliografia

Boesing, A.L.; Nichols, E. & Metzger, J.P. (2017). ‘Land use type, forest cover and forest edges modulate avian cross-habitat spillover’, Journal of Applied Ecology, 55(3), pp. 1252-1264.

Lazzarotto, J.J. (1999). ‘Revisão de literatura: relação entre aspectos nutricionais e obesidade em pequenos animais’, R. Un. Alfenas, 5, pp. 33-35.

Lazzarotto, J.J. (2001). ‘Nutrição e alimentação de filhotes de cães’, Revista da FZVA, 7/8(1), pp. 157-162.

Neto, M.A. et al. (2011). ‘Emissão de gases do efeito estufa em diferentes usos da terra no bioma Cerrado’, Revista Brasileira de Ciência do Solo, 35(1), pp. 63-76.

Parreira,, P.R. (2007). ‘Aspectos fundamentais da determinação da exigência energética de cães domésticos’, Rev. Acad, 5(4), pp. 415-422.

Revista Veterinária. ‘Alimentação e manejo nutricional de cães e gatos’. < http://www.revistaveterinaria.com.br/2014/04/14/alimentacao-e-manejo-nutricional-de-caes-e-gatos/> . Acessada em três de maio de 2018.

Rocha, V.M. & Fialho, E.S. (2010). ‘Uso da terra e suas implicações na variação termo-higrométrica ao longo de um transeto campo-cidade no município de Viçosa-MG’, Revista de C. Humanas, 10(1), pp. 64-77.

Uriarte, M. et al. (2011) ‘Influence of land use on water quality in a tropical landscape: A multi-scale analysis’, Landscape Ecology, 26(8), pp. 1151–1164. doi: 10.1007/s10980-011-9642-y.

> Oi Eduarda,
>
> A proposta B tem um jeito bem de calculadora mesmo, e acho que não aproveita as mútiplas ferramentas de programação que o R disponibiliza pra você. Veja que sua proposta A foi bastante elaborada e interessante; a proposta B deve ser da mesma qualidade caso aconteça algum problema com a proposta A. Por favor pense numa outra proposta pra começar uma vez as duas sejam aceitas. Próximas conversas terão lugar pelo email e uma vez a

> gente consiga um consenso você deverá fazer as mudanças que tenham lugar no seu wiki.
>
> —*Gustavo A. Ballen*

PROPOSTA FINAL

Com o auxílio do monitor, decidi seguir com a proposta A. Conforme desenvolvimento da função, adquiri novos conhecimentos e percebi que não seria possível seguir exatamente o que estava descrito no pseudocódigo da proposta. Esse pseudocódigo orientou os passos iniciais, mas a função tornou-se mais completa (e complexa), como pode ser visto no script abaixo.

Isto posto, segue:

1. Código da função

```
mudanca<-function(data, mud = NULL, anos = NULL, total = FALSE, limiar,
num= "todos") # Cria a função
{
  ##VERIFICANDO OS PARÂMETROS
  if(ncol(data) < 4)  # Verifica o número de colunas
  {
    stop("número de colunas insuficiente") # Se o número de colunas for
menor que 4 (número mínimo de colunas que a função precisa ter para
funcionar), a função para
  }
  if(sum(!complete.cases(data)) > 0) # Verifica a presença de NA. A
função "!complete.cases" indica aonde tem valores faltantes (ou seja,
refere-se aos NA); a função "sum" mostra o total de valores faltantes
(número total de NA), logo, como nenhum NA pode estar presente, se essa
soma for maior que zero, a função para
  {
    NAS<- which(is.na(data), arr.ind=TRUE) # Essa linha e a próxima
servem para mostrar aonde estão os NA da tabela
    cat("linhas com NA = ", NAS[,1], "\n colunas com NA = ", NAS[,2],
"\n") # Sequencia da linha anterior, para mostrar a localização dos NA
na tabela
    stop("NAs presentes") # Se pelo menos um NA estiver presente, a
função para
  }
  if(length (limiar) == 0 | limiar < 0 | limiar> 100) # Verifica se o
argumento "limiar" esta ausente ou dentro do intervalo esperado (como
```

```
deve ser uma porcentagem de uso, os valores de limiar esperado devem
estar entre zero e cem)
  {
    stop ("Limiar deve estar especificado no intervalo 0 <= limiar <=
100")  # Caso uma das restrições seja encontrada, a função deve parar e
retornar mensagem de erro
  }
  ####PASSO 1: CRIA UMA NOVA TABELA, APENAS COM OS USOS DE INTERESSE
  if(length(mud) == 0) # Analisa se o argumento mud foi fornecido pelo
usuário
  {
    data.usos <- data # Cria uma nova tabela, igual à tabela original,
apenas para manter o nome de objeto que será utilizado nos passos
seguintes e a função continua analisando todos os usos presentes na
tabela original
  }else{              # Caso o argumento mud seja especificado
    if(any(mud %in% data[,2] == FALSE)) # Verifica se o que foi
estipulado no argumento mud está de acordo com os tipos de uso da terra
possíveis (usos existentes nos dados inseridos), que estão presentes
sempre na coluna 2 do data.frame inserido (por isso uso a indexação data
[,2]); caso tenha qualquer valor lógico igual a "FALSE", a função deve
parar
    {
      cat("Este(s) uso(s) da terra não está(ão) incluso(s) nos tipos
possíveis =", sep = ", ", setdiff(mud,data[,2]), "\n") # Mostra quais
são os elementos que diferem dos tipos de uso do solo que são possíveis
de serem usados
      stop("Erros nos tipos de uso") # Para a função e retorna mensagem
de erro
    }
    # Caso a condicional acima não seja atendida, função continua, agora
somente para os usos de interesse:
    data.usos<-data[0,] # Cria uma tabela vazia, apenas com os nomes das
colunas iguais aos da tabela de entrada
    for(i in mud) # Faz um ciclo para cada um dos usos presente no
argumento mud
    {
      data.usos<-rbind(data.usos,data[which(data[2]==i),]) #adiciona uma
nova linha, a cada ciclo, na tabela criada apenas com os nomes das
colunas: une, a cada ciclo, todas as colunas das linhas que contenham os
usos de interesse
    }
  }
  #### PASSO 2: CRIA UMA TABELA COM OS ANOS DE INTERESSE
  if(length(anos)== 0) # Caso o usuário não forneça o argumento anos,
todos os anos da tabela original serão analisados (comparando um ano com
o ano anterior)
  {
    data.anos<-data.usos   # De maneira semelhante ao primeiro passo, é
criada uma nova tabela igual à tabela de usos (feita no passo 1), apenas
para manter o nome de objeto que será utilizado nos passos seguintes
```

```
    anos<- colnames(data.anos[3:length(data.anos)]) # Cria um vetor com
o nome das colunas referente aos anos de estudo, fundamental para os
próximos passos da função
  }else{        # Caso o argumento anos seja especificado
    data.anos<-data.usos[,1:2]  # Cria uma nova tabela, apenas com as
colunas 1 e 2 da tabela, sendo que essas colunas refere-se aos locais e
usos de interesse
    for (i in 1:length(anos)) # Faz um ciclo que se repite de acordo com
o número de anos de interesse
    {
      data.anos<-cbind(data.anos,Ano = data.usos[,anos[i]]) # A cada
ciclo, apenas as linhas das colunas referentes aos anos de interesse são
adicionadas
    }
  }
  #### PASSO 3: CALCULA AS DIFERENCAS DE PORCENTAGEM DE USO ENTRE OS
ANOS
  # Comandos para quando total é igual a FALSO ou VERDADEIRO
  data.mud.t<-data.anos   # Cria uma nova tabela (data.frame) para
adicionar as colunas referentes à mudanças de usos da terra, sem alterar
o data.frame original (data.anos, criado no passo anterior, que será
importante para passos seguintes)
  for(i in 3:(length(data.anos[1,])-1))    # Subtrai os valores de
porcentagem de um uso da terra de um ano (coluna i+1) pela porcentagem
do mesmo uso  no ano anterior (coluna i). Logo, o ciclo acontece entre a
primeira coluna que tem valores de porcentagem de uso para um ano de
estudo (que sempre será a coluna de número 3) e o número colunas total
menos 1 (subtrai o valor 1 pois dentro do for{} o calculo é feito
subtraindo o valor da coluna i+1 pelo valor da coluna i, se o valor 1
não fosse subtraído do total de colunas da tabela, o número de ciclos
seria maior do que o máximo de colunas)
  {
    data.mud.t[,paste("Porcentagem de mudança",anos[i-1],anos[i-2])]<-
data.anos[,i+1]-data.anos[,i]   # Adiciona novas colunas a cada ciclo.
As colunas recebem o nome "Porcentagem de mudança" junto com os anos que
estão sendo analisados. Em cada coluna será feita a subtração da
porcentagem do uso de interesse no maior ano (i+1) pela porcentagem
desse uso no ano anterior (i)
  }
  if(total == TRUE)   # Se o argumento total for verdadeiro (default é
ele ser falso), uma nova linha de comando será adicionada para calcular
a mudança total nas porcentagens de uso da terra
  {
    data.mud.t[,paste("Porcentagem de mudança total")]<-
data.anos[,length(data.anos)]-data.anos[,3] # Este é o passo adicional
caso argumento total seja verdadeiro, diferente do default: uma nova
coluna comparando a mudança total (subtraindo porcentagem de uso do
último ano pela porcentagem de uso do ano inicial) é adicionada à tabela
  }
  ####PASSO 4: SELECIONA OS LOCAIS COM OS LIMIARES DE INTERESSE
```

```
  # Cria nova tabela, com todas as linhas, mas só as colunas de ganho
(as colunas das porcentagens de usos por ano são eliminadas)
  data.ganhos<-data.mud.t[,-(3:(length(anos)+2))]  # Dados sobre os anos
começam na coluna 3. Soma-se o valor 2 ao argumento anos para saber até
qual coluna vão os dados de porcentagem de uso por ano (as duas
primeiras colunas são referentes ao local e ao uso e devem permanecer)
  data.limiar<-data.ganhos[0,]  # Nova tabela, sem linhas e apenas com
os nomes das colunas
  for(i in 3:length(data.limiar)) # Nesse ciclo somente as linhas que
possuam o limiar de interesse são selecionadas e adicionadas à tabela
  {
    data.limiar<-
rbind(data.limiar,data.ganhos[which(data.ganhos[,i]>=limiar |
data.ganhos[,i]<=-limiar),])  # Seleciona os valores de ganho de uso
(maiores ou iguais ao limiar de interesse) ou perda de uso (menores ou
iguais ao correspondente negativo do limiar)
  }
  data.limiar<- unique(data.limiar) # Exclui todas as possíveis linhas
repetidas
  for(i in 3:length(data.limiar))  # Para adicionar um tracinho ("-")
nas células que aparecem na tabela gerada anteriormente, mas que não
estão de acordo com o limiar, este ciclo vai da primeira coluna com
dados de mudança de uso da terra, até a última coluna que tenha essa
informação
  {
    data.limiar[which(data.limiar[,i]<limiar & data.limiar[,i]>-
limiar),i] <- "-"  # Adiciona um tracinho para os valores que são
menores que o limiar e maiores que o limiar negativo (esses valores
podem aparecer porque a linha toda é adicionada, mesmo que o valor de
uma das colunas desta linha não esteja de acordo com o limiar)
  }
  #### PASSO 5 - FAZ UMA TABELA COM OS LOCAIS SELECIONADOS
ALEATORIAMENTE
  data.limiar[,1]<- as.factor(data.limiar[,1])  # Transforma para fator,
o que permite usar os locais como níveis(levels) no momento da escolha
aleatória dos locais
  if(num == "todos" | num<=0 |num>=length(levels(data.limiar[,1]))) # Se
o número de locais desejados não for especificado ou for especificado,
mas menor ou igual a zero ou maior ou igual ao número total de locais
encontrados com o limiar de interesse, a tabela final conterá todos os
locais encontrados no passo 4 e a seleção aleatória de locais com o
limiar de interesse não será feita
  {
    data.final<-data.limiar #Cria nova tabela, apenas para manter o nome
que será utilizado no passo seguinte
    data.final<- data.final[order(data.final[,1]),] # Organiza a tabela
em ordem crescente, de acordo com o nome dos locais
    cat("Argumento num não especificado ou >= ao número de locais
encontrados com a mudança de uso(s) de interesse, dentro do limiar
especificado, portanto todos os locais possíveis estão representados",
"\n")
```

```
  }else{            # Caso o número de locais seja especificado, maior
que zero e menor que o número total de locais encontrados
    (locais.aleat<- sample(levels(data.limiar[,1]),num))    # Cria vetor
com locais (que devem sempre estar na coluna 1) selecionados
aleatoriamente
    data.final<- data.frame(nrow(length(locais.aleat)),
ncol(length(data.limiar)))  # Cria uma nova tabela com número de colunas
da tabela do passo anterior (com os dados de mudança de usos dentro do
limiar especificado) e que terá apenas as linhas dos locais selecionados
aleatoriamente
    for(i in locais.aleat) # Faz ciclagem para cada um dos locais
selecionados aleatoriamente
    {
      data.final<-
rbind(data.final,data.limiar[which(data.limiar[1]==i),])  # Adiciona os
locais que foram aleatoriamente selecionados à tabela criada
anteriormente
      data.final<- data.final[order(data.final[,1]),]  # Ordena a tabela
em ordem crescente, de acordo com o nome dos locais que foram
aleatoriamente selecionados
    }
  }
  ####PASSO 6 - SALVA A TABELA NO DIRETÓRIO DE TRABALHO E RETORNA O
DATA.FRAME NA ÁRA DE TRABALHO
  write.table(data.final, file = "Mudanças_de_uso_de_interesse.csv", sep
= ",", dec = ".", row.names = F) # Salva a tabela final, como arquivo
.csv, no diretório de trabalho, com o nome Mudanças_de_uso_de_interesse.
  Mudanca_usos <<- data.final
  return(data.final) # Retorna a tabela (data.frame) final na tela da
área de trabalho
}
```

2. Help da função

```
mudanca                package:unknown                R Documentation


DETECÇÃO DE MUDANÇAS DE USOS DA TERRA


Description:

     Esta função detecta locais nos quais, através dos anos, usos da
terra tenham passado por algum limiar de mudança (determinado pelo
usuário). Para isso, a função manipula a tabela de entrada e retorna uma
nova, apenas com os locais nos quais os limiares de mudança foram
```

```
encontrados.


Usage:

mudanca (data, mud = NULL, anos = NULL, total = "FALSE", limiar, num=
"todos")


Arguments:

data    Objeto, da classe data.frame, que contenha na primeira coluna os
locais de estudo, na segunda, os usos da terra e, a partir da terceira
coluna, as porcentagens de uso da terra para cada ano de estudo.

mud Vetor, da classe “character”, que define quais são os usos da terra
de interesse (devem estar entre as opções de uso contidas na coluna dois
do data.frame de entrada). Caso não seja especificado (mud = NULL, que é
o default deste argumento), a função analisa todos os usos da terra.

anos    Vetor, da classe “character” e definido em ordem numérica
crescente, que define os anos de estudo que se deseja comparar no
cálculo do valor de mudança do uso da terra, sendo que um ano sempre
será comparado com o ano anterior. Quando este argumento não é
especificado (anos = NULL, default deste argumento), todos os anos são
comparados (um ano com o anterior a ele).

total   Argumento lógico (TRUE ou FALSE), que tem como default ser FALSE
e indica se o usuário quer comparar o primeiro ano de estudo com o
último (útil quando mais de dois anos estão sendo comparados). Caso seja
TRUE, o último ano de interesse é comparado com o primeiro e retorna a
porcentagem de mudança total do uso de interesse.

limiar  Além do data.frame de entrada, este é o único argumento que,
obrigatoriamente, deve ser definido pelo usuário e determina o limiar de
mudança que está sendo procurado. A função, então, procurará valores
maiores ou iguais ao limiar, indicando aumento (ganho) da porcentagem do
uso, ou, ainda, valores menores ou iguais ao correspondente negativo do
limiar, evidenciando diminuições (perdas) daquele uso da terra. Este
argumento deve ser um número inteiro no intervalo 0<= limiar <= 100.


num Número de locais que o usuário deseja ter como retorno. Este deve
ser um número inteiro, maior que zero e menor que o total de locais
existentes. Como padrão (default), a função retorna todos os locais
encontrados em que os usos da terra passaram pelo limiar de mudança. De
maneira semelhante, caso o valor do argumento acabe sendo maior ou igual
ao total de locais encontrados (de acordo com limiar), a função também
retorna todos os locais. Se num for especificado e de acordo com
intervalo 0 <num <total de locais encontrados, a função escolhe
aleatoriamente os locais que aparecerão no data.frame de retorno.
```

Details:

Para esta função, os dois argumentos obrigatórios são o data.frame de entrada e o limiar. Se apenas estes dois argumentos forem fornecidos, a função procura limiares de mudanças para todos os usos da terra presentes na coluna 2, para todos os anos de estudo, não calcula a mudança total e retorna todos os locais encontrados. Dessa forma, a utilização mais ampla da função é: mudanca(data, limiar).

Dentre as utilizações possíveis, esta função é adequada para aqueles usuários que querem determinar locais de estudo, dentro das possibilidades em uma paisagem, que tenham passado por determinadas mudanças específicas de uso da terra. Sendo assim, a função pode auxiliar nos delineamentos amostrais de pesquisadores diversos, especialmente aqueles que estão inseridos na área de Ecologia da Paisagem.

Value:

Tabela (data.frame) que contem os locais selecionados (primeira coluna), os usos de interesse que tiveram a mudança de acordo com o limiar (segunda coluna) e a porcentagem total de mudança (a partir da terceira coluna). As colunas que contenham informação sobre a porcentagem de mudança serão referentes à comparação entre um ano e o anterior a ele, sendo que os anos comparados aparecem no nome da coluna. Os locais nos quais o limiar de mudança não foi atingido terão um tracinho (“-“) na respectiva célula. A tabela de resultados, além de aparecer na área de trabalho, também é salva no diretório de trabalho.

Warning:

A função é interrompida e retorna mensagens de erros se:

1. O data.frame de entrada não possuir número adequado de colunas (>=4);
2. Existir qualquer NA no data.frame de entrada (neste caso, uma mensagem com os locais – numero da linha e da coluna – dos NA também é mostrada);
3. O argumento limiar não for especificado ou estiver fora dos limites possíveis (menor que zero ou maior que 100);
4. Se o argumento mud tiver classes de usos diferentes das possíveis (determinado pelos tipos de uso da coluna 2 do data.frame de entrada) e o(s) argumento(s) diferente(s) é(são) mostrado(s) na tela.

Caso o argumento num não seja especificado, ou seja maior ou igual ao número de locais possíveis, uma mensagem avisa que isto ocorreu e os locais no data.frame de saída representam todos os locais possíveis.

```
Author(s):

     Eduarda Romanini
     e-mail: eduardaromanini.bio@gmail.com



See Also:

     Funções: which(), rbind(), cbind()


Examples:
## Observação: dados criados pela autora da função.


     ####Exemplos com a tabela de entrada correta

    ## Criando o data.frame:

LOCAL<- rep(1:6, each = 5)
USOS<- rep(c("pasto", "cana","silvicultura","perene", "urbano"), 6)
ano1<-
c(70,10,0,10,10,20,60,10,5,5,50,30,0,10,10,50,30,0,10,10,70,10,0,10,10,2
0,40,25,0,15)
ano2<-
c(50,30,10,0,10,20,50,10,10,10,40,30,0,15,15,40,30,0,15,15,20,50,15,0,15
,20,50,15,0,15)
ano3<-
c(40,20,30,0,10,20,50,15,0,15,10,70,0,5,15,10,70,0,5,15,20,50,15,0,15,70
,10,0,10,10)
ano4<-
c(10,70,10,0,10,20,40,25,0,15,10,70,0,0,20,10,70,0,0,20,10,70,0,0,20,10,
70,0,0,20)

data.inicial<- data.frame(LOCAL, USOS, ano1, ano2, ano3, ano4,
stringsAsFactors = FALSE)
colnames(data.inicial)<- c("Local", "Usos", "1985", "1995", "2005",
"2015")


    ## Utilização mais ampla da função, apenas com o data.frame de
entrada e o limiar:

mudanca(data.inicial, limiar = 20)  # Limiar de mudança de 20%.


    ## Exemplos utilizando mais argumentos:

mudanca(data.inicial, mud= c("cana", "silvicultura", "urbano"), anos =
c("1985", "1995", "2015"), total = TRUE, limiar = 10, num = 3) # Limiar
```

```
de 10% e escolhendo aleatoriamente 3 locais.


mudanca(data.inicial, mud= c("cana", "pasto"), anos = c("1985", "1995",
"2005"), total = TRUE, limiar = 40, num = 5)  ## Aparece a mensagem de
que todos os locais estão representados, uma vez que o argumento num tem
um número maior que o de locais encontrados dentro do limiar de mudança
especificado (limiar = 40%).



     #### Exemplos com erros:

    ## Argumento mud tem classes erradas:
mudanca(data.inicial, mud= c("cana", "silcultura", "urno"), anos =
c("1985", "1995", "2015"), total = TRUE, limiar = 10, num = 3)  # Alguns
dos usos estão escritos de maneira errada: mensagem de erros nos tipos
de uso, indicando que os usos “silcultura” e “urno” não são tipos
possíveis.


    ## Data.frame não possui número de colunas suficiente:

data.col <- data.inicial[,-(4:6)] # Cria um novo data.frame com número
de colunas errado.

mudanca(data.col, mud= c("cana", "silvicultura"), anos = c("1985",
"1995", "2005"), total = TRUE, limiar = 30, num = 3)  # Mensagem de
erro: número de colunas insuficiente.

    ## Data.frame possui Nas:

NAS<-
c(10,70,10,NA,10,20,40,NA,0,15,10,70,0,0,20,10,NA,0,0,20,10,70,0,0,20,10
,70,0,0,NA) # Cria vetor com NA

data.NA <- cbind(data.inicial, NAS) # Adiciona nova coluna à tabela
inicial.
colnames(data.NA)<- c("Local", "Usos","1975", "1985", "1995", "2005",
"2015") # Dá nome as colunas.

mudanca(data.NA, mud= c("cana", "pasto"), anos = c("1975", "1995",
"2015"), total = TRUE, limiar = 30, num = 3) # Mensagem de erro sobre a
presença de NAs e a localização deles no data.frame d entrada.



     #### Exemplo com mais dados sobre anos e locais:

install.packages("data.table") # Instala pacote com função que permite
```

```
ler dados diretamente da internet.
library(data.table) # Abre o pacote.


data.maior<-
fread("http://ecologia.ib.usp.br/bie5782/lib/exe/fetch.php?media=bie5782
:01_curso_atual:alunos:trabalho_final:eduardaromanini.bio:exemplofuncao2
.csv", header = TRUE, sep = ";", stringsAsFactors = FALSE)  # Carrega
arquivo .csv que está em uma página da internet.

data.exemplo<- data.frame(data.maior) # Transforma objeto para classe
data.frame.
colnames(data.exemplo)<- c("Local", "Usos", "1985", "1990", "1995",
"2000", "2005", "2010", "2015") # Dá nome para as colunas.



mudanca(data.exemplo, mud= c("cana", "silvicultura", "pasto"), anos =
c("1985", "1990", "2000", "2010", "2015"), total = TRUE, limiar = 30,
num = 6) # Limiar de 30%, com escolha aleatória de 6 locais.
```

3. Dados que serão necessários para um dos exemplos da função

Exemplo com mais dados

Observação: Não é necessário fazer download deste arquivo, o próprio exemplo carrega os dados diretamente do link da internet.