Alfredo Acosta

Doutorando em Epidemiología aplicada as zoonoses, FMVZ, USP

O título da minha tese é: '*Avaliação do sistema de vigilância para peste suína clássica do Equador. Proposta de sistema de vigilância baseado em risco*', orientado pelo professor Fernando Ferreira.

http://leb.fmvz.usp.br/pt-br

Meus exercicios

Linque para a página com os meus exercícios resolvidos Exercicios de Alfredo Acosta

Proposta de trabalho final

I Plano A: Reporte padronizado de vacinação

Contextualização

A Agencia de regulação e controle fito zoo sanitário e uma organização adscrita ao Ministério de Agricultura do Equador, nas suas competências tem a execução e fiscalização do Projeto de erradicação de Peste suína clássica (PSC). A PSC e uma das principais doenças infecciosas dos suínos domésticos, não tem cura e eventualmente causa a morte do animal infetado, sua erradicação se baseia em prevenção mediante vacinação e remoção sistemática dos animais infetados.

No Equador se tem implementado um proceso de vacinação obrigatória para seu controle, para isso se tem um sistema de registro de atestados de vacina, numa base de dados (PostgreSQL) com uma interface desenvolvida em código PHP, reportes dinâmicos ainda não foram desenvolvidos. Atualmente se baseiam em análise de reportes excel mediante filtros e tabela dinâmicas tanto nos funcionários públicos, como professionais autorizados que tem aceso aos reportes.

Para o controle do processo, são definidos no inicio do ano, metas baseadas na população existente e vacinada (baseados na mais alta vacinação histórica registrada), são registrados os atestados de vacina (50.000 por ano), e o respetivo seguimento e controle em cada órgão estadual (24 províncias) quem a sua vez fazem o controle nos municípios (221 cantones) e 1024 parroquias (mínima unidade territorial). 330 Fiscais agropecuários Veterinários e Agrônomos fiscalizam o avance das metas usando reportes da cobertura de vacina em cada unidade territorial, e uns 100 credenciados privados para o que é usado vários reportes de avance de cobertura, semanais, mensuais e anuais (Exemplo1).

Ao introduzir um banco de dados (que se encontra padronizado desde o servidor), a função

responderia a pergunta Qual é o status (porcentagem de avanço) de imunização neste período e neste território. Mostrando um reporte PDF com a informação proporcionada e analisada.

Planejamento da função

Entrada: vac(`arquivo`, `territorio`, `ordem`)

- `arquivo` = Nome do arquivo em formato excel (class: dataframe).
- `territorio` = Análise territorial (class: list, 3 fatores [provincia, canton ou parroquia]).
- `ordem` = Alfabético ou ranking (class: list, 2 fatores [alf, rank]).

Verificando os parâmetros:

- arquivo = É um arquivo .xlsx com 20 colunas validadas os nomes originais das colunas. Se não escreve: “O arquivo deve ser o original baixado desde reportes do sistema GUIA”.
- territorio = Somente 3 opções são aceitas para análise territorial. Se não escreve: “Só podem ser feitos analises para Provincia, Cantón ou Parroquia (pro, can ou par são aceitos)”
- order = Somente duas opções. Se não escreve: “Só aceito alfab = alfabético ou rank = ranking”.

- Um objeto meta fornecido antecipadamente debe rá vir como fonte de dados dentro da função.

Pseudo-código

1. Avalia se o arquivo fornecido é um xlsx com o número de colunas (20) especificado e os nomes de colunas padronizadas.
2. Importa o arquivo como um dataframe de nome `dados`.
3. Cria um data frame do nome `resultv` com N fileiras, e 3 colunas UT (executado, meta, porcentagem), prenche tudo com NAs.
4. Calcula o número de N fileiras de acordo com o UT número de unidades territoriais do parâmetro territorio + um com o total.
5. Cria um objeto `porcentagem` e salva a divisão da soma do numero de animais vacinados do objeto dados pela soma do numero de animais vacinados no objeto meta por cada UT. Valida a existência do nome de cada UT antes de fazer a divisão.
6. Salva do objeto `result` com um match por nomes da UT meta com dados.
7. Ordena os valores do objeto `result` de acordo com o parâmetro order alfabético das UT ou numérico da porcentagem.
8. Cria um objeto `range` com a resta do tempo em dias do primeiro e última data encontradas na coluna data de registro e salva tambem as duas datas.
9. Cria um objeto `data` com data e hora do fim do processo.
10. Cria um objeto `timestamp` com o tempo de análise desde a importação ate a geração do PDF.

Saída:

- Cria um gráfico de barras no eixo X os nomes da UT, eixo Y a `porcentagem`.
- Exporta em PDF o nome do archivo proporcionado, os parâmetros de entrada da função, o

objeto range, o dataframe result o gráfico a data do reporte e o timestamp.

Links

http://www.agrocalidad.gob.ec/proyecto-de-control-y-erradicacion-de-peste-porcina-clasica-ppc/
https://guia.agrocalidad.gob.ec/agrodb/ingreso.php

Exemplo 1

II Plano B: Mapa dinâmico de movimentação animal

Contextualização

O controle da movimentação animal e um processo mandatório padronizado ao redor do mundo no qual todos os animais de produção (bovinos, suínos, aves, equinos e outros), para se mobilizar precisam de um atestado no qual indique sua condição sanitária.

Os atestados de mobilização são salvos, mas em geral não se tem visualização padrado dos dados registrados. Uma análise gráfica feita com mapas das densidades por localização da origem e destino dos animais poderia ser bem útil para analisar tendências de movimentações ou lugares de maior densidade. Isto para melhorar controles baseados nas tendências deses movimentos ou simplesmente para saber espacialmente onde foram movimentados.

Mapas são feitos a través de programas de geoprocessamento como ARCGIS ou QGIS mas, um mapa com densidades como no proporcionado no exemplo(2) demora em fazer unos 20-30 minutos em pessoal experiente ou dificilmente seriam feitos pelos propietários dos animais.

A função pretende traçar o movimento de animais de produção em diferentes escalas espaciais, fornecendo ao usuário de um resumo estatístico e mapas em escala apropriada registrando as movimentações dos animais.

A estatística descritiva `mstats` vai analisar o : total de atestados, animais, lotes, media de animais por atestado e media de animais e lotes por atestado por unidade territorial, summary do número de animais.

Planejamento da função

Entrada: `mov(arquivo,divter,dir,tipo,especie)`

- `arquivo` = Nome do arquivo em formato csv (class: dataframe).
- `divter` = Divisão territorial (class: list, 3 fatores [d1, d2 ou d3]).
- `dir` = Direção do movimento (class: list, 2 fatores [from, to]).
- `tipo` = Animais ou lotes movimentados (class: list, 2 fatores [animais, lotes])
- `especie` = Tipo de espécie de animais (class: list, [aves, bovinos, suínos, caprinos]

Verificando os parâmetros:

- arquivo = csv contendo:
    1. Número de atestado.
    2. Data do atestado.
    3. 3 divisões territoriais origem e 3 divisões territoriais destino (estado, microrregião, municipio), (states, counties, municipalities) ou (Provincias, cantones, parroquias).
    4. Número de animais movimentados.
    5. Especie movimentada.

Se não escreve: “O arquivo deve ter os parametros requeridos”.

- divter = Somente 3 opções são aceitas para análise territorial. Se não escreve: “as opções de territorios disponíveis são t1, t2 ou t3”.
- dir = Somente duas opções. Se não escreve: “só analiso origem ou destino”.
- tipo = Somente duas opções. Se não escreve: “analiso unicamente animais ou lotes”.
- especie = Quatro opções. Se não escreve: “as opcões animais disponveis são aves, bovinos, suínos, caprinos”.

Pseudo-código:

1. Avalia se o arquivo fornecido é um csv com 9 colunas e os nomes de colunas padronizadas (indicadas no help).
2. Importa o arquivo como um dataframe de nome `dados`.
3. Cria um data frame do nome `resultm`, colnames [Divisão_territorial, n_animais, n_lotes, Código_divisão].
4. Prenche `resultm` com o resultado da soma de todos os animais consolidados ao número de divisão territorial escolhida, se o parâmetro tipo é animais.
5. Prenche `resultm` com o resultado da conta única do número de atestado consolidados ao número de UT escolhido, se o parâmetro tipo é lote.
6. Prenche `resultm` com o código da divisão territorial[t1], fazendo uma comparação do nome de divisão territorial contida no arquivo shp com o nome da divisão territorial do `resultm[Divisão_territorial]`.
7. Prenche `resultm` com o código da divisão territorial[t2], fazendo uma comparação do nome de divisão territorial contida no arquivo shp com o nome da divisão territorial do `resultm[Divisão_territorial t1+t2]`.
8. Prenche `resultm` com o código da divisão territorial[t3], fazendo uma comparação do nome de unidade territorial contida no arquivo shp com o nome da unidade territorial do `resultm[Unidade_Territorial t1+t2+t3]`.
9. Cria um data.frame do nome `mstats` com informação descritiva e estatísticas incluindo as informações indicadas na contextualização.
10. Carrega o shape com unidades territoriais de acordo com a escolha do usuario de `divter` e salva como objeto `shp`.
11. Cria um objeto `grapm` e representa graficamente os dados de `resultm` num mapa com coloração de escala de 4 níveis de cor, baseado no shape.
12. Cria um objeto `range` com a resta do tempo em dias do primeiro e última data encontrada na coluna data do df. Salva tambem as duas datas.
13. Cria um objeto `data` com data e hora do fim do processo.
14. Cria um objeto `timestamp` com o tempo de análise desde a importação ate a geração do PDF.

Saída:

- Exporta o grafico `gramp` com as unidades territoriais preenchidas de acordo com a escala de 4 níveis de cor por densidade de número de animais ou lotes movimentados `resultm` para a unidade territorial selecionada.
- Exporta em PDF o nome do arquivo proporcionado, os parâmetros de entrada da função, o objeto range, o objeto grapm a `data`, `mstats` e o `timestamp`.

III Referências

Exemplo 2

---

nome	projeto	email	telefone
Alfredo		alfredoacosta@usp.br	~~~~~

> Olá Alfredo,
>
> Você estruturou bem ambas as propostas de funções, apresentou bem os argumentos de entrada e as saídas das mesmas. Achei muito legal que ambas as funções pretendem verificar que os parâmetros de entrada estejam nos formatos certos e retornem mensagem de erro para os usuários. E também achei bem legal os pseudo-códigos. Contudo, ambas as propostas têm uma deficiência crítica, e é que elas são muito específicas. Da forma como você as contextualiza e as apresenta dá para entender que suas propostas irão apenas resolver os problemas destacados por você, não mais disso. Um ponto importante da função que vocês devem criar é a capacidade de resolver problemas amplos, dentre os quais problemas na sua área de atuação devem ser apenas um exemplo do que a função pode fazer. Ou seja, mesmo que sua função se encaixe dentro de um contexto acadêmico específico, a aplicabilidade da tarefa que desenvolverá deve abranger diversos problemas da área. Sugiro você dar uma olhada de novo aos “Passos para uma boa proposta” explicitados para o Trabalho final. Veja que não estou pedindo para você trocar suas propostas, de fato gostei mais da B por exemplo. Mas reformule suas propostas de forma que a aplicabilidade das funções seja mais abrangente. — *Gustavo Agudelo 2018/05/10 11:34*

> — *Alexandre Adalardo de Oliveira 2018/05/18 10:39* Caro Alfredo,
>
> Concordo com o Gustavo que ambas as propostas estão muito bem colocadas e

apresentadas. Apesar de serem específicas a um problema, são desafios interessantes de implementação de atividades repetitiva que pode ser útil. Pode seguir com aquela que tiver mais interesse.

Mapa dinâmico de movimentação animal

funcao_mov.r

```
mov <- function(arquivo,divter,dir,tipo,map) {

  t1i <- Sys.time() #00 Contando o tempo da funcao
  #Requiring package
  require(ggplot2, quietly = F)
  require(readxl, quietly = F)
  require(raster, quietly = F)
  require(tidyverse, quietly = F)
  require(leaflet, quietly = F)
  require(webshot, quietly = F)
  require(htmlwidgets, quietly = F)
  # A Validacao de argumentos ----
  if(divter != "t1" && divter != "t2" && divter !="t3") #01 Verifica se
divter foi inserido corretamente
  {
    stop("divter so pode ser t1, t2 ou t3") #2 indica os valores permitidos
  }
  if(dir != "o" && dir != "d") #3 Verifica se dir foi inserido corretamente
  {
    stop("dir so pode ser 'o' ou 'd' (origem ou destino)" ) #4 Indica os
valores permitidos
  }
  if(tipo != "a" && tipo !="l") #5 Verifica se tipo foi inserido
corretamente
  {
    stop("tipo so pode ser 'a' ou 'l' (animais ou lotes)") #6 indica os
valores permitidos
  }
  if(map != "q" && map != "qq") # 7 Verifica se opcoes do mapa foram
inseridas corretamente
  {
    stop("map so pode ser 'q', 'qq'(quantidade, quartil)") #8 indica os
valores permitidos
  }
```

```
  #B Validacoes no arquivo ----
  m <- readxl::read_excel(path = arquivo,
                          col_types = c("text", "text", "text",
                                        "text", "text", "text",
                                        "text", "numeric","date"))# 9 carrego
o arquivo fornecido para validacoes
  m <<- m
  #10 crio um dataframe com as clases correctas para comparacao
  a <- data.frame(as.character("a"), as.character("b"), as.character("c"),
                  as.character("d"), as.character("e"), as.character("f"),
                  as.character("g"), as.numeric('5'), as.Date("2018-01-01"),
                  stringsAsFactors = FALSE)
  if(!sapply(a, class) == sapply(m, class)) #11 Compara as variaveis do
arquivo fornecido com as do exemplo
  {
    stop("Suas 9 variavels, tem que ser no formato especificado na ajuda")
#12 Indica as variaveis
  }
  #C Somando e agregando animais dependendo dos territorios origen ----
  if(divter=="t1" && dir=="o" && tipo=="a") #13 Especificando argumentos
para somas
  {
    result <- aggregate(x= m[ ,8], by=m[ ,2], FUN=sum) #14 somatoria de
animais t1 e origem     }
  }
  if(divter=="t2" && dir=="o" && tipo=="a") #15 Especificando argumentos
para somas
  {
    result <- aggregate(x= m[ ,8], by=m[ ,3], FUN=sum) #16 somatoria de
animais t2 e origem
  }
  if(divter=="t3" && dir=="o" && tipo=="a") #17 Especificando argumentos
para somas
  {
    result <- aggregate(x= m[ ,8], by=m[ ,4], FUN=sum) #18 somatoria de
animais t3 e origem
  }
  #D Somando e agregando animais dependendo dos territorios destino ----
  if(divter=="t1" && dir=="d" && tipo=="a") #19 Especificando argumentos
para somas
  {
    result <- aggregate(x= m[ ,8], by=m[ ,5], FUN=sum) #20 somatoria de
animais t1 e destino
  }
  if(divter=="t2" && dir=="d" && tipo=="a") #21 Especificando argumentos
para somas
  {
    result <- aggregate(x= m[ ,8], by=m[ ,6], FUN=sum)#22 somatoria de
animais t2 e destino
  }
  if(divter=="t3" && dir=="d" && tipo=="a") #23 Especificando argumentos
```

```
para somas
  {
    result <- aggregate(x= m[ ,8], by=m[ ,7], FUN=sum)#24 somatoria de
animais t3 e destino
  }
  #E Contando e agregando lotes dependendo dos territorios origen ----
  if(divter=="t1" && dir=="o" && tipo=="l") #25 Especificando argumentos
para conta de lotes
  {
    result <- aggregate(x= m[ ,8], by=m[ ,2], FUN=NROW) #26 Contagem dos
lotes t1 e origem
  }
  if(divter=="t2" && dir=="o" && tipo=="l") #27 Especificando argumentos
para conta de lotes
  {
    result <- aggregate(x= m[ ,8], by=m[ ,3], FUN=NROW) #28 Contagem dos
lotes t2 de origem
  }
  if(divter=="t3" && dir=="o" && tipo=="l") #29 Especificando argumentos
para conta de lotes
  {
    result <- aggregate(x= m[ ,8], by=m[ ,4], FUN=NROW) #30 Contagem dos
lotes t3 de origem
  }
  #F Contando e agregando lotes dependendo dos territorios destino ----
  if(divter=="t1" && dir=="d" && tipo=="l") #31 Especificando argumentos
para conta de lotes
  {
    result <- aggregate(x= m[ ,8], by=m[ ,5], FUN=NROW) #32 Contagem dos
lotes t1 de origem
  }
  if(divter=="t2" && dir=="d" && tipo=="l") #33 Especificando argumentos
para conta de lotes
  {
    result <- aggregate(x= m[ ,8], by=m[ ,6], FUN=NROW) #34 Soma os animais
t2 de origem
  }
  if(divter=="t3" && dir=="d" && tipo=="l") #35 Especificando argumentos
para conta de lotes
  {
    result <- aggregate(x= m[ ,8], by=m[ ,7], FUN=NROW) #36 Soma os animais
t3 de origem
  }
  # E Fazendo mapa para t1
  if(divter == 't1')#37 Se e t1 muda o colname para t1
  {
    colnames(result) <- c("t1","cantidad") #38 muda o nome da coluna para
nao confundir
    ec1 <- getData("GADM", country = "ecuador", level = 1)#39 baixa o mapa
para t1
```

```
    #40 elimino carateres especiais do resultado para poder fazer match
correcto entre dados e mapa
    result$t1 <- gsub("á","a", result$t1)
    result$t1 <- gsub("é","e", result$t1)
    result$t1 <- gsub("í","i", result$t1)
    result$t1 <- gsub("ó","o", result$t1)
    result$t1 <- gsub("ú","u", result$t1)
    #41 elimino caracteres especiais do data frame
    ec1@data$NAME_1 <- gsub("á","a", ec1@data$NAME_1)
    ec1@data$NAME_1 <- gsub("é","e", ec1@data$NAME_1)
    ec1@data$NAME_1 <- gsub("í","i", ec1@data$NAME_1)
    ec1@data$NAME_1 <- gsub("ó","o", ec1@data$NAME_1)
    ec1@data$NAME_1 <- gsub("ú","u", ec1@data$NAME_1)
    #42 Comparo dados do mapa com os dos dados do result de obter match
mantenho o valor se nao aplicar NA
    ### e importante conhecer os NA, que sao territorios que nao tem dados.
    result <- data.frame(t1 = setdiff(ec1@data$NAME_1, result$t1),
                         cantidad = NA,
                         stringsAsFactors = FALSE) %>% bind_rows(result)
    # 43 Crio no data do shape as quantidades para usarlas na projeccao do
mapa
    ec1@data$cantidad <- result$cantidad[match(ec1@data$NAME_1, result$t1)]
    ec1 <<- ec1 #44 envia o shape file para o global enviroment
  }
  if(divter == 't1' && map == "q")#45 Se e mapa solicitado e por quantidade
  {
    #46 meus colores de paleta calculados por numero de animais
    mypal <- colorNumeric(palette = "viridis", na.color = "#ffffff", domain
= ec1@data$cantidad)
    map <- leaflet() %>% #47 Apertura ambiente grafico html
      addProviderTiles("OpenStreetMap.Mapnik") %>%#48 Apertura open maps
      setView(lat = -1.7, lng = -78.5, zoom = 7) %>%#49 Localiza o mapa
      addPolygons(data = ec1, stroke = FALSE, smoothFactor = 0.5,
fillOpacity = 0.7, #50 Agrega poligonos e define opacidades
                  fillColor = ~ mypal(ec1@data$cantidad),#51 Prence cores do
shape de acordo a quantidade
                  popup = paste("Terr: ", ec1$NAME_1, "<br>",#52 Pop-up
dados de territorio
                                "Quantidade: ", ec1@data$cantidad, "<br>"))
%>%#53 Pop-up quantidade
      addLegend(position = "bottomleft", pal = mypal, values =
ec1@data$cantidad,#54 Posicao da legenda, paleta e dados
                title = "Quantidade", opacity = 0.7)# 55 Legenda e opacidade
  }
  if(divter == 't1' && map == "qq")# 56 Se o mapa e solicitado por quartis
  {
    mypal <- colorQuantile(palette = "viridis", na.color = "#ffffff", domain
= ec1@data$cantidad)#57 muda as cores de paleta para calculo com quartis
    #58 Usa as mesmas opcoes para os graficos dos outros territorios
    map <- leaflet() %>% #58 Apertura ambiente grafico html
      addProviderTiles("OpenStreetMap.Mapnik") %>%#59 Apertura open maps
```

```
      setView(lat = -1.7, lng = -78.5, zoom = 7) %>% #60 Localiza o mapa
      addPolygons(data = ec1, stroke = FALSE, smoothFactor = 0.5,
fillOpacity = 0.7, #61 Agrega poligonos e define opacidades
                  fillColor = ~mypal(ec1@data$cantidad), #62 Prence cores do
shape de acordo a quantidade
                  popup = paste("Terr: ", ec1$NAME_1, "<br>", #63 Pop-up
dados de territorio
                                "Quant: ", ec1@data$cantidad, "<br>")) %>%
#64 Pop-up quantidades
      addLegend(position = "bottomleft", pal = mypal, values =
ec1@data$cantidad,#65 Posicao da legenda, paleta e dados
                title = "Quartis", opacity = 0.7)#66 Lenda e opacidade
  }
  # F Fazendo mapa para t2
  if(divter == 't2')#67 Se e t1 muda o colname para t1
  {
    colnames(result) <- c("t2","cantidad") #68 muda o nome da coluna para
nao confundir
    ec2 <- getData("GADM", country = "ecuador", level = 2)#69 baixa o mapa
para t2
    #70 elimino carateres especiais do resultado t2 para poder fazer match
correcto entre dados e mapa
    result$t2 <- gsub("á","a", result$t2)
    result$t2 <- gsub("é","e", result$t2)
    result$t2 <- gsub("í","i", result$t2)
    result$t2 <- gsub("ó","o", result$t2)
    result$t2 <- gsub("ú","u", result$t2)
    #71 elimino carateres especiais do mapa para poder fazer match correcto
entre dados e mapa
    ec2@data$NAME_2 <- gsub("á","a", ec2@data$NAME_2)
    ec2@data$NAME_2 <- gsub("é","e", ec2@data$NAME_2)
    ec2@data$NAME_2 <- gsub("í","i", ec2@data$NAME_2)
    ec2@data$NAME_2 <- gsub("ó","o", ec2@data$NAME_2)
    ec2@data$NAME_2 <- gsub("ú","u", ec2@data$NAME_2)
    #72 Comparo o data do mapa com os dos dados e de nao encontrar algum
aplicar NA
    result <- data.frame(t2 = setdiff(ec2@data$NAME_2, result$t2),
                         cantidad = NA,
                         stringsAsFactors = FALSE) %>% bind_rows(result)
    #73 Envio cantidades dato para o shape
    ec2@data$cantidad <- result$cantidad[match(ec2@data$NAME_2, result$t2)]
    ec2 <<- ec2 #74 envia o mapa para o global enviroment
  }
  if(divter == 't2' && map == "q")#75 Se e mapa solicitado e por quantidade
  {
    #76 Colores de paleta calculados por numero de animais
    mypal <- colorNumeric(palette = "viridis", na.color = "#ffffff", domain
= ec2@data$cantidad)
    #77 Usa as mesmas opcoes para os graficos dos outros territorios desde
#47
```

```
    map <- leaflet() %>% #78 Apertura ambiente grafico html
      addProviderTiles("OpenStreetMap.Mapnik") %>% #79 define proveedor open
maps
      setView(lat = -1.7, lng = -78.5, zoom = 7) %>% #80 Localiza o mapa
      addPolygons(data = ec2, stroke = FALSE, smoothFactor = 0.5,
fillOpacity = 0.7, #81 Agrega poligonos e define opacidades
                  fillColor = ~mypal(ec2@data$cantidad),#82 Prence cores do
shape de acordo a quantidade
                  popup = paste("Terr: ", ec2@data$NAME_2, "<br>",#83 Pop-up
dados de territorio
                                "Quant: ", ec2@data$cantidad, "<br>")) %>%
#84 Pop-up quantidades
      addLegend(position = "bottomleft", pal = mypal, values =
ec2@data$cantidad,#85 Posicao da legenda, paleta e dados
                title = "Quantidade", opacity = 0.7) #86 Lenda e opacidade
  }
  if(divter == 't2' && map == "qq") #87 Se o mapa e solicitado por quartis
  {
    mypal <- colorQuantile(palette = "viridis", na.color = "#ffffff", domain
= ec2@data$cantidad) #88 muda os colores de paleta para calculo com quartis
    map <- leaflet() %>% #88 Abre ambiente grafico html
      addProviderTiles("OpenStreetMap.Mapnik") %>% #89 Apertura open maps
      setView(lat = -1.7, lng = -78.5, zoom = 7) %>% #90 Localiza o mapa
      addPolygons(data = ec2, stroke = FALSE, smoothFactor = 0.5,
fillOpacity = 0.7, #91 Agrega poligonos e define opacidades
                  fillColor = ~mypal(ec2@data$cantidad), #100 Prence cores
do shape de acordo a quantidade
                  popup = paste("Terr: ", ec2@data$NAME_2, "<br>", #101 Pop-
up dados de territorio
                                "Quant: ", ec2@data$cantidad, "<br>")) %>%
#102 Pop-up quantidades
      addLegend(position = "bottomleft", pal = mypal, values =
ec2@data$cantidad,#103 Posicao da legenda, paleta e dados
                title = "Quartis", opacity = 0.7) #104 Lenda e opacidade
  }
  # G Fazendo mapa para t3
  if(divter == 't3') #105 Se e t1 muda o colname para t1
  {
    colnames(result) <- c("t3","cantidad") #106 muda o nome da coluna t3
para nao confundir
    ec3 <- getData("GADM", country = "ecuador", level = 3) #107 baixa o mapa
para t3
    #108 elimino carateres especiais do resultado t3 para poder fazer match
correcto entre dados e mapa
    result$t3 <- gsub("á","a", result$t3)
    result$t3 <- gsub("é","e", result$t3)
    result$t3 <- gsub("í","i", result$t3)
    result$t3 <- gsub("ó","o", result$t3)
    result$t3 <- gsub("ú","u", result$t3)
    #109 elimino carateres especiais do mapa para poder fazer match correcto
entre dados e mapa
```

```
    ec3@data$NAME_3 <- gsub("á","a", ec3@data$NAME_3)
    ec3@data$NAME_3 <- gsub("é","e", ec3@data$NAME_3)
    ec3@data$NAME_3 <- gsub("í","i", ec3@data$NAME_3)
    ec3@data$NAME_3 <- gsub("ó","o", ec3@data$NAME_3)
    ec3@data$NAME_3 <- gsub("ú","u", ec3@data$NAME_3)
    #110 Comparo o data do mapa com os dos dados e de nao encontrar algum
aplicar NA
    result <- data.frame(t3 = setdiff(ec3@data$NAME_3, result$t3),
                         cantidad = NA,
                         stringsAsFactors = FALSE) %>% bind_rows(result)
    #111 Envio cantidades dato para o shape
    ec3@data$cantidad <- result$cantidad[match(ec3@data$NAME_3, result$t3)]
    ec3 <<- ec3#112 envia o mapa para o global enviroment
  }
  if(divter == 't3' && map == "q")#113 Se e mapa solicitado e por quantidade
  {
    #114 meus colores de paleta calculados por numero de animais
    mypal <- colorNumeric(palette = "viridis", na.color = "#ffffff", domain
= ec3@data$cantidad)
    map <- leaflet() %>% #115 Apertura ambiente grafico html
      addProviderTiles("OpenStreetMap.Mapnik") %>% #116 Apertura open maps
      setView(lat = -1.7, lng = -78.5, zoom = 7) %>% #117 Localiza o mapa
      addPolygons(data = ec3, stroke = TRUE, weight = 0.3, smoothFactor =
0.5, fillOpacity = 0.4, #118 Agrega poligonos e define opacidades
                  fillColor = ~mypal(ec3@data$cantidad),#119 Prence cores do
shape de acordo a quantidade
                  popup = paste("Terr: ", ec3@data$NAME_3, "<br>",#120 Pop-
up dados de territorio
                                "Quant: ", ec3@data$cantidad, "<br>"))
%>%#121 Pop-up quantidades
      addLegend(position = "bottomleft", pal = mypal, values =
ec3@data$cantidad,#122 Posicao da legenda, paleta e dados
                title = "Quantidade", opacity = 0.4) #123 Legenda e opacidad
  }
  if(divter == 't3' && map == "qq")# 124 Se o mapa e solicitado por quartis
  {
    mypal <- colorQuantile(palette = "viridis", na.color = "#ffffff", domain
= ec3@data$cantidad)#125 muda os colores de paleta para calculo com quartis
    map <- leaflet() %>% #126 Apertura ambiente grafico html
      addProviderTiles("OpenStreetMap.Mapnik") %>% #127 Apertura open maps
      setView(lat = -1.7, lng = -78.5, zoom = 7) %>% #128 Localiza o mapa
      addPolygons(data = ec3, stroke = TRUE, weight = 0.3, smoothFactor =
0.5, fillOpacity = 0.4, #129 Agrega poligonos e define opacidades
                  fillColor = ~mypal(ec3@data$cantidad), #130 Prence cores
do shape de acordo a quantidade
                  popup = paste("Terr: ", ec3@data$NAME_3, "<br>", #131 Pop-
up dados de territorio
                                "Quant: ", ec3@data$cantidad, "<br>")) %>%
#132 Pop-up quantidades
      addLegend(position = "bottomleft", pal = mypal, values =
```

```
ec3@data$cantidad,#133 Posicao da legenda, paleta e dados
                 title = "Quartis", opacity = 0.4) #134 Lenda e opacidade
  }
  #G Mostra dados no console
  cat("mov (1) -- Os territorios sem valores numericos (NA's) sao:") #135
Etiqueta
  cat("\n") #136 line break
  print(result[is.na(result$cantidad), ]) #137 Mostra dados do #135
  result <<- result #138 Envia para o global enviroment os valores
calculados
  print(map)#139 Mostra o mapa na janela grafica
  t1f <- Sys.time() #140 Registra o tempo final da corrida da funcao
  tt1 <- t1f - t1i #141 Diferenca de tempos entre inicial e final
  cat("\n") #142 line break
  cat("mov (2) -- O tempo total utilizado na analise foi: ") #143 Etiqueta
  print(round(tt1,1)) #144 Mostra no console, tempo utilizado na corrida da
funcao
  cat("\n") #144 line break
  cat("mov (3) -- Presione nos territorios do mapa para saber os valores
numericos.") #145 Etiqueta
  cat("\n")#146 line break
  cat("mov (4) -- Um arquivo (txt), imagem (png) e um site (html) foram
salvos na pasta de trabalho com mais informacoes") #147 Etiqueta
  cat("\n") #148 line break
  cat("mov (5) -- Obrigado por usar mov() !! ") #149 Etiqueta
  cat("\n") #150 line break
  #H Exporta mapa como png
  saveWidget(map, "mov_mapa.html", selfcontained = FALSE) #150 Cria um temp
html para salvar uma imagem
  webshot::webshot("mov_mapa.html","mov_mapa.png", zoom = 2)#151 Salva a
imagem como png
  # I Exporta arquivo txt com resumo dos analises
  sink("mov_dados.txt") #152 funcao para escrever texto
  cat("RESUMO ANALISE DE MOVIMENTACAO ANIMAL")#153 Etiqueta
  cat("\n") #154 line break
  cat("\n")#155 line break
  cat("\n") #156 line break
  cat("RESUMO DO BANCO CONFERIDO") # 157 Etiqueta
  cat("\n")# 158 line break
  cat("Data do primeiro registro encontrado: ") #159 etiqueta
  range <- summary(m$`Fecha Inicio Vigencia`) #160 Resumen de datas do
arquivo
  class(range) <-class(m$`Fecha Inicio Vigencia`) #161 asigno a mesma clase
do df para que se mostre como data (POSIXct)
  print(range[1]) #162 Mostro a data menor
  cat("\n")# 163 line break
  cat("Data do ultimo registro encontrado: ") #164 Etiqueta
  print(range[6]) #165 Mostro a date maior
  cat("\n") #166 line break
  cat("Faixa de atestados analizados: ") #167 Etiqueta
  print(round(range[6]-range[1]),2) #168 Tempo de diferenca entre atestados
```

```
registrados
  cat("\n") #169 line break
  cat("Numero de atestados analizados: ") #170 Etiqueta
  print(length(m$`Número Certificado`))#171 Quantidade de GTA analizadas
  cat("\n")#172 line break
  cat("Numero de animais movimentados: ") #173 Etiqueta
  print(sum(m$Cantidad)) #174 Quantidade de animais movilizados
  cat("\n") #175 line break
  cat('Media de animais por atestado: ') #176 Etiqueta
  round(sum(m$Cantidad)/length(m$`Número Certificado`),0)#177 Promedio
animais por atestado
  cat("\n")#178 line break
  print("Resumo do numero de animais movimentados: ") #179 Etiqueta
  summary(m$Cantidad) #180 Resumo do numero de animais movilizados
  cat("\n")#181 line break
  cat("\n")#182 line break
  cat("RESUMO DOS RESULTADOS") #183 Etiqueta
  cat("\n")#184 line break
  cat("Resumo das quantidades movimentadas por territorios (NA's=territorios
sem informacao): ")
  cat("\n") #185 line break
  print(summary(result$cantidad)) #186 resultados
  cat("\n")#187 line break
  cat("\n")#188 line break
  cat("\n")#189 line break
  cat("Data de execucao do analisis:") #190
  print(t1f <- Sys.time()) #191 Registra o tempo final da corrida da funcao
  cat("\n")#192 line break
  cat("\n")#193 line break
  cat("\n")#194 line break
  cat("Nome dos territorios sem informacao") #195 etiqueta
  print(result[is.na(result$cantidad), ]) #196 Territorios sem informacoes
  cat("\n")#197 line break
  cat("Territorios ordenados com informacao") #198 etiqueta
  print(count(result[!is.na(result$cantidad), ]))# 199 Imprime os que nao
sao NA
  print(result[order(result$cantidad,decreasing=TRUE), ]) #200 Territorios
ordenados por seu numero de animais
  cat("\n")#201 line break
  sink() #202 fecha dispositivo
} #203 fim da funcao

mov("arquivo.xlsx",divter="t1",dir="d",tipo = "l", map = "qq")
```

Help

```
mov                          package:unknown                  R Documentation

Mapa dinâmico de movimentação animal

-Description

Esta função gera caraterização espacial dos animais e lotes de animais
movimentados em
origem ou destino. A função traça o movimento de animais de produção em tres
escalas
espaciais territoriais, fornecendo uma visualização HTML dinâmica e
interativa dos
registros de movimentação no nivel nacional e local.

-Usage

function(arquivo,divter,dir,tipo,map)

-Arguments

x      Nome do arquivo em formato xlsx, contendo os registros a serem
analizados em
       9 colunas.
divter Um caracter indicando a divisão territorial na cual sera plotados os
dados.
       "t1", "t2" ou "t3". Sendo as divisões territoriais dos países em
geral nesta
       ordem de maior a menor.
dir    Um caracter indicando ao mapa a forma de representar a direção da
movimentação.
       Sendo "o" origem ou "d" para destino.
tipo   Um caracter indicando se seram plotados o numero de atestados ou
lotes de animais
       ou os animais contidos neses lotes. Sendo "a" animais e "l" para
lotes.
map    Um caracter indicando o de representação por quantidade ou por
quartis. Sendo "q"
       para quantidade ou "qq" para quartis.

-Detais

O arquivo fornecido deve ser .xlsx e ter 9 colunas: código único, territorio
origem 1,
territorio origem 2, territorio origem 3, territorio destino 1, territorio
destino 2,
territorio destino 3, quantidade e data.
```

```
colnames exemplo ex(cod,ot1, ot2, ot3, dt1, dt2, dt3, num, data).

Os mapas utilizados são disponibilizados pela GADM Database of Global
Administrative Areas,
[[www.gadm.org/]], usando os 3 niveis gerais correspondentes com t1, t2 e
t3.

-Value

Mapa Web iterativo
Reporte txt con resumo do análise total de atestados, animais, lotes, media
de animais por
atestado e media de animais e lotes por atestado por unidade territorial, se
indicam os
territorios sem informação.
Arquivo de imagem png do mapa.

-Author(s)
Alfredo Acosta alfredoacosta@usp.br

-Examples:

  mov(x,divter="t1",dir="d",tipo = "l", map = "q")

  mov(x,divter="t2",dir="o",tipo = "a", map = "qq")
```

banco_de_dados.rar