Fernanda Bocalini

Bióloga, com graduação e mestrado pela Universidade de São Paulo. Atualmente, sou aluna do primeiro ano de doutorado do Programa de Pós-Graduação em Zoologia do IBUSP. Meu projeto de doutorado é sobre biogeografia e filogeografia comparada das aves do Centro de Endemismo Pernambuco (CEP), utilizando Elementos Ultra-Conseravados (UCEs) do genoma. Tenho experiência em zoologia com ênfase em ornitologia, taxonomia, sistemática, biogeografia e evolução.

exec

Trabalho Final

Proposta A

Função genpop.map()

Esta função foi pensada para ser utilizada em estudos de genética de populações e filogeografia, seus resultados poderiam auxiliar na formulação modelos biogeográficos a serem testados posteriormente. O objetivo desta função é construir um mapa de distribuição que fosse capaz de representar a proximidade genética de cada uma das populações estudadas, por meio da na coloração das manchas de distribuição, sendo que populações mais próximas geneticamente teriam coloração mais intensa e as menos relacionadas teriam manchas mais claras. Como numa variação de vermelho intenso até amarelo claro.

Para construir esta função, seriam necessários 4 argumentos principais:

- uma matriz com o valores de fst entre cada par de população que se deseja comparar. Pensei na possibilidade dessa matriz de valores de fst ser substituída por uma matriz de sequências alinhadas, assim a função que calcula o fst poderia ser incluída dentro dessa função.
- um data frame com as seguintes colunas: a) Nome da espécie junto com o número de identificação do indivíduo; b) valor da latitude decimal de cada indivíduo; c) valor da longitude decimal de cada indivíduo; d) Nome da população que o indivíduo pertence, mesmo nome utilizado na matriz do fst (eg. “Sul da Bahia”, “Centro Pernambuco”, “Belém”, etc).
- Um argumento chamado “alvo”, seria a coluna população do data.frame (da classe fator), indexando-se qual seria a população de referência para comparações do valor do fst (que é sempre entre pares de populações).Por exemplo, eu estudo a biogeografia das aves do Centro de Endemismo Pernambuco, portanto, essa população seria escolhida como “alvo” das comparações dos valores de fst. Caso esse argumento não seja preenchido, a função não irá rodar.
- o argumento “shape”, em que o usuário poderá carregar o shape do mapa desejado, caso seja deixado em branco haverá um shape default.

Cada ponto será plotado no mapa, haverá a união dos pontos mais externos de cada população para formação de manchas de distribuição, de acordo com a população a qual cada ponto pertence. Por fim, cada mancha será preenchida com a cor correspondente, de acordo com a proximidade genética de cada população com a população alvo. Pares de populações com menor fst terão cores mais fortes

e aqueles com fst maior cores mais fracas.

O resultado dessa função será este mapa e uma tabela com os valores do fst par a par, no caso de utilização de sequências.

Exemplo de utilização da função:

genpop.map(matrix, data.frame, alvo = data.frame[data.frame$pop==“centro Penambuco”,])

Proposta B

Função map.quantile()

Na taxonomia é de extrema importância a análise das variações morfológicas entre indivúduos de diferentes populações de uma espécie, podendo essas populações serem consideradas subespécies ou não. A visualização dessas variações em um mapa de distribuição é essencial para se verificar se há coerência geográfica nessa variação, ou seja, se é um padrão exibido pela maioria dos indivíduos de uma população (variação fixada por um padrão de ancestralidade e descendência) ou se trata-se apenas de uma variação individual sem qualquer significado evolutivo. Assim,é possível a delimitação de unidades taxonômicas operacionais e após o acúmulo de evidências (morfológicas, morfométricas, genéticas, etc.) pode-se elevar populações diagnosticáveis a espécies plenas.

A função map.quantil() tem como objetivo mapear as variações morfológicas nos indivíduos de uma população, esta variação poderá ser qualquer variável contínua, como comprimento da asa,comprimento da cauda ou até mesmo comprimento de onda (no caso de diferentes colorações).

Para entrar na função os argumentos serão:

- um data frame com 5 colunas: a primeira com o número de identificação do indivíduo, a segunda com a latitude decimal, a terceira com a longitude decimal, a quarta com as medidas de cada indivíduo e a quinta o nome da população daquele indivíduo (podendendo ser usado como referência as distribuições das subespécies ou regiões biogeográficas). As colunas deverão estar nessa ordem para a função possa indexá-las
- o argumento “shape”, em que o usuário poderá carregar o shape do mapa desejado, caso seja deixado em branco haverá um shape default.

A função irá plotar os pontos de ocorrência de cada indivíduo, unir os externos da distribuição dos pontos de acordo com a população a qual eles pertencem. Posteriormente, irá calcular os quartis da variável medida, levando em conta todos os dados (sem distinção por população). Esses quartis serão utilizados para agrupar as medidas dos indivíduos em 3 categorias, os pontos dos 3 quartis serão plotados no mapa com cores diferentes e se possível agrupados em manchas. Assim, poderíamos verificar como a variação da variável está distribuída, se é aleatória ou se há relação com as populações definidas previamente. A função poderá retornar este mapa e um teste anova entre as populações (ou teste t, se forem apenas 2 populações).

> Oi, Fernanda,
>
> Legal, ambas as suas propostas são super legais e parecem factíveis. Algumas dicas em cada uma para ver se elas não ficam mais claras.

Proposta A:

- Pode ser interessante que a própria função veja se o que foi dado como argumento foi a matriz de fst ou se foi as sequências alinhadas. Caso seja a sequência, antes de continuar, a função primeiro calcula a matriz. Assim, você fica com as duas opções em uma função, que fica mais flexível.
- Não entendi a necessidade da primeira coluna do dataframe. A função serviria para populações diferentes de uma única espécie, não? Ou ela conseguiria diferenciar entre espécies? Se sim, não ficou claro como.
- Aliás, será que o argumento de entrada tem que ser um dataframe? E se for vetores de coordenadas e um vetor de populações?
- Será que no argumento "alvo" é necessário chamar a coluna com um indexador? Você não consegue pensar em um jeito de escrever o código em que o argumento é direto o nome da população?
- Não seria legal um argumento em que usuário pode escolher as cores? Como *default*, pode ser essas cores que você sugeriu, mas se o usuário quiser mudar, pode ser interessante, tornando a função mais flexível e adaptável às necessidades do usuário.

Proposta B

- De novo, aqui não ficou claro para mim a necessidade da primeira coluna do dataframe, com a identificação da espécie…
- De novo, será que precisa ser um dataframe o objeto de entrada?
- O cálculo dos quartis e como isso será plotado no mapa não ficou claro… Cada ponto de cada indivíduo será plotado com uma cor, de acordo com o quartil a qual pertence, é isso?

Dito tudo isso, gosto mais da primeira proposta, pois tem essa primeira opção de colocar as sequências ou não. Sugiro que invista em detalhar a primeira proposta de acordo com as dicas acima! Pode escrever aqui abaixo mesmo (e sempre deixe a original aqui na página) e eu vou comentando por aqui também.

Qualquer coisa pode me mandar um email.

—*Diana Garcia*

Olá Diana, tive problemas com meu computador e só pude responder agora.

Vou seguir com a proposta A, vou seguir sua sugestão e fazer com que a função identifique se foi dada uma matriz com as sequências alinhadas (calculando o fst na mesma função) ou uma matriz com os fsts já calculados para cada par de população. Assim, a função irá identificar qual informação que está sendo dada e prosseguir com os cáculos necessários. A necessidade de se ter uma coluna que identifique o indivíduo seria para que pudesse ser feita uma conexão entre o ponto da coordenada geográfica e o indivíduo daquele ponto, mas ele realmente pode ser descartado, já que deve estar na ordem. Concordo que não há a necessidade do arquivo de entrada ser um data frame, vou seguir com a ideia dos vetores de coordenas e populações, também o argumento alvo poderia apenas chamar o nome da população do vetor anterior. Achei interessante a idéia do usuário poder escolher as cores e vou incluí-la na função.

Proposta A reformulada

O objetivo desta função é construir um mapa de distribuição que fosse capaz de representar a proximidade genética de cada uma das populações estudadas, por meio da na coloração das manchas de distribuição, sendo que populações mais próximas geneticamente teriam coloração mais intensa e as menos relacionadas teriam manchas mais claras.

- O primeiro argumento da função será uma matriz (x) que poderá ser de duas formas a escolha do usuário: com os valores de fst entre cada par de população que se deseja comparar ou uma matriz de sequências alinhadas no formato fasta. Caso o usuário coloque uma matriz de sequências alinhadas, o cálculo do fst será feito dentro da função.
- O segundo argumento (coord) será uma matriz contendo a latitude na coluna 1 e a longitude na coluna 2
- O terceiro argumento (pop) será vetor ou fator dando a atribuição da população de cada linha do primeiro argumento (x).
- O quarto argumento será o argumento alvo, que irá identificar qual população do argumento anterior será utilizado para as comparações. O default será a população que aparece primeiro no argumento pop.
- O quinto argumento será o argumento shape, em que o usuário poderá carregar o shape do mapa desejado, caso seja deixado em branco haverá um shape default.
- O último argumento será o argumento cor, em caso o usuário deseje utilizar cores diferentes do padrão default da função, ele poderá carregar um vetor com as cores desejadas do mais escuro (populações mais próximas) para o mais claro (populações mais distantes)

> Oi, Fernanda,
>
> Ótimo, acho que vai ficar bem legal sua função! Boa sorte e qualquer coisa grita (via fórum de preferência! Hehe)!
>
> —*Diana Garcia*

Na função final os argumentos coord e pop foram agrupados em um mesmo argumento, chamado dados. O argumento shape foi substituído pelo argumento mapa e não há um mapa default, dado que os pontos podem ser de origem de qualquer parte do globo, por isso é melhor que o usuário defina qual mapa gostaria de utilizar.

Código da Função

```
popgen.map <- function(x, dados, alvo, mapa, cor="cor")
{
  library(hierfstat) # pacote para calcular o fst
  library(ggmap) # pacote dependente do ggplot2
  library(ggplot2) # pacote com ferramentas gráficas, para plotar heatmap
  library(RColorBrewer) # pacote com as cores que serão plotadas nos mapas
  if (is.null(alvo))
  {
    stop("selecionar população alvo") # a função não roda se o usuário não
```

```
selecionar a população alvo
  }
  if (is.null(x))
  {
    stop("argumento x está faltando") # a função não roda se o usuário não
definir o argumento x
  }
  if (is.null(dados))
  {
    stop("o argumento dados está faltando") # a função não roda se o usuário
não definir o argumento x
  }
  if (is.null(mapa))
  {
    stop("o argumento mapa está faltando") # a função não roda se o usuário
não definir o mapa
  }
  if (class(x)== "data.frame" & cor=="cor") # se x é as sequências alinhadas
e a cor é a default
  {
    fst <- pairwise.neifst(x) # calcula o fst par a par de cada população
presente na primeira coluna do data.frame
    tabela.fst <- data.frame(fst)  # transforma o resultado do cáculo de fst
de matriz para data frame
    pop.alvo <- tabela.fst[alvo] # indexa a coluna com os valores da
população de interesse
    pop.alvo[is.na(pop.alvo)] <- 0 # substitui os NA por 0 na tabela de
resultados, para plotar
    pop.alvo[pop.alvo < 0] <- 0 # substitui os valores menores que 0 por
zero, pois é a mesma coisa para o valor de fst
    pop.alvo.v <-as.vector(pop.alvo[,alvo]) # transforma em vetor os valores
de fst
    levels.pop <- as.vector(levels(dados$Pop)) # faz um vetor com os nomes
das populações
    dados.alvo <- data.frame(levels.pop, pop.alvo.v) # junta o vetor
população com os resultados de fst
    dados.fst <- merge(dados, dados.alvo, by.x="Pop", by.y= "levels.pop")  #
dá o valor de fst referente a cada indivíduo da tabela dados
    dados.fst[,"pop.alvo.v"] <- as.factor(dados.fst[,"pop.alvo.v"])  #
transformando o valor de fst em fator para que possa ser plotado no mapa
    cor<- c(colorRampPalette(brewer.pal(9,"YlOrRd"))(nrow(dados.alvo)))  #
define a paleta de cores que será usada
    cor<- cor[nrow(dados.alvo):1] # inverte o vetor de cores para que ele
seja plotado da cor mais escura para a cor mais clara
    mapa +
      stat_bin2d(aes(x = Long, y = Lat, fill = pop.alvo.v), size = 1, bins =
30, alpha = 0.8, data = dados.fst, binwidth = c(1, 1)) +
scale_fill_manual(values= cor) # plota em um mapa os valores de fst com
cores diferentes
  }
  if (class(x)== "matrix" & cor=="cor") # se x é os dados de fst já
```

```
calculados e a cor é default
  {
    fst <- x # os valores de fst serão dados pela matrix dos valores par a
par inserida pelo usuário
    tabela.fst <- data.frame(fst)
    pop.alvo <- tabela.fst[alvo] # indexa a coluna com os valores da
população de interesse
    pop.alvo[is.na(pop.alvo)] <- 0 # substitui os NA por 0 na tabela de
resultados, para plotar
    pop.alvo[pop.alvo < 0] <- 0 # substitui os valores menores que 0 por
zero, pois é a mesma coisa para o valor de fst
    pop.alvo.v <-as.vector(pop.alvo[,alvo]) # transforma em vetor os valores
de fst
    levels.pop <- as.vector(levels(dados$Pop)) # faz um vetor com os nomes
das populações
    dados.alvo <- data.frame(levels.pop, pop.alvo.v) # junta o vetor
população com os resultados de fst
    dados.fst <- merge(dados, dados.alvo, by.x="Pop", by.y= "levels.pop")  #
dá o valor de fst referente a cada indivíduo da tabela dados
    dados.fst[,"pop.alvo.v"] <- as.factor(dados.fst[,"pop.alvo.v"])  #
transformando o valor de fst em fator para que possa ser plotado no mapa
    cor<- c(colorRampPalette(brewer.pal(9,"YlOrRd"))(nrow(dados.alvo)))  #
define a paleta de cores que será usada
    cor<- cor[nrow(dados.alvo):1] # inverte o vetor de cores para que ele
seja plotado da cor mais escura para a cor mais clara
    mapa +
      stat_bin2d(aes(x = Long, y = Lat, fill = pop.alvo.v), size = 1, bins =
30, alpha = 0.8, data = dados.fst, binwidth = c(1, 1)) +
scale_fill_manual(values= cor) # plota em um mapa os valores de fst com
cores diferentes
  }
  if (class(x)== "data.frame" & cor!= "cor") # se x é as sequências
alinhadas e a cor foi esclhida pelo usuário
  {
    fst <- pairwise.neifst(x) # calcula o fst par a par de cada população
presente na primeira coluna do data.frame
    tabela.fst <- data.frame(fst)
    pop.alvo <- tabela.fst[alvo] # indexa a coluna com os valores da
população de interesse
    pop.alvo[is.na(pop.alvo)] <- 0 # substitui os NA por 0 na tabela de
resultados, para plotar
    pop.alvo[pop.alvo < 0] <- 0 # substitui os valores menores que 0 por
zero, pois é a mesma coisa para o valor de fst
    pop.alvo.v <-as.vector(pop.alvo[,alvo]) # transforma em vetor os valores
de fst
    levels.pop <- as.vector(levels(dados$Pop)) # faz um vetor com os nomes
das populações
    dados.alvo <- data.frame(levels.pop, pop.alvo.v) # junta o vetor
população com os resultados de fst
    dados.fst <- merge(dados, dados.alvo, by.x="Pop", by.y= "levels.pop")  #
```

```
dá o valor de fst referente a cada indivíduo da tabela dados
    dados.fst[,"pop.alvo.v"] <- as.factor(dados.fst[,"pop.alvo.v"])  #
transformando o valor de fst em fator para que possa ser plotado no mapa
    mapa +
      stat_bin2d(aes(x = Long, y = Lat, fill = pop.alvo.v), size = 1, bins =
30, alpha = 0.8, data = dados.fst, binwidth = c(1, 1)) +
scale_fill_manual(values= cor) # plota em um mapa os valores de fst com
cores diferentes
  }
  else # se x é matriz dos dados de fst já calculados e a cor foi escolhida
pelo usuário
  {
    fst <- x # os valores de fst será a matrix dos valores par a par
inserida pelo usuário
    tabela.fst <- data.frame(fst)
    pop.alvo <- tabela.fst[alvo] # indexa a coluna com os valores da
população de interesse
    pop.alvo[is.na(pop.alvo)] <- 0 # substitui os NA por 0 na tabela de
resultados, para plotar
    pop.alvo[pop.alvo < 0] <- 0 # substitui os valores menores que 0 por
zero, pois é a mesma coisa para o valor de fst
    pop.alvo.v <-as.vector(pop.alvo[,alvo]) # transforma em vetor os valores
de fst
    levels.pop <- as.vector(levels(dados$Pop)) # faz um vetor com os nomes
das populações
    dados.alvo <- data.frame(levels.pop, pop.alvo.v) # junta o vetor
população com os resultados de fst
    dados.fst <- merge(dados, dados.alvo, by.x="Pop", by.y= "levels.pop")  #
dá o valor de fst referente a cada indivíduo da tabela dados
    dados.fst[,"pop.alvo.v"] <- as.factor(dados.fst[,"pop.alvo.v"])  #
transformando o valor de fst em fator para que possa ser plotado no mapa
    mapa +
      stat_bin2d(aes(x = Long, y = Lat, fill = pop.alvo.v), size = 1, bins =
30, alpha = 0.8, data = dados.fst, binwidth = c(1, 1)) +
scale_fill_manual(values= cor) # plota em um mapa os valores de fst com
cores diferentes
  }
}
```

Help da Função

```
popgen.map                package:unknown                R Documentation

Função que constrói um heatmap de acordo com os dados de fst de diferentes
populações
Description:
A função irá utilizar dados de sequências alinhadas no formato data frame
(mesmo formato aceito pelo pacote hierstat), para primeiramente calcular o
```

```
fst par a par entre as populações e posteriormente plotar em um mapa um
heatmap em que seja possível visualizar a divergência genética entre as
populações, sempre com relação a uma população de referência, escoolhida
pelo usuário.
Usage:
    popgen.map(x, dados, alvo, mapa, cor="cor")
Arguments:
x       poderá assumir duas formas a escolha do usuário, poderá ser um data
frame das sequências alinhadas, estes dados devem ser da classe data.frame,
este data frame deve conter a população de origem da sequência na primeira
coluna e os genótipos dos indivíduos nas demais columas. Ou o usuário pode
escolher colocar os dados do fst já calculados dentro de uma matriz.
dados   um data frame com os dados correspondentes aos indivíduos das
sequências carregadas no argumento x. Deverá conter 3 colunas, uma com os
dados da latitude decimal de cada indivíduo, com o nome "Lat", outra com os
dados de Longitude decimal com o nome "Long" e outra com o nome da população
a qual o indivíduo pertence com o nome "Pop".

alvo    nome da população de referência com as quais as comparações dos
valores do fst serão feitas. Se o nome tiver espaços substituir por pontos
como Rio de Janeiro = Rio.de.Janeiro.

mapa    um objeto da classe ggplot produzido pelas funções get_map, ggmap e
ggplot (pacotes ggmap e ggplot2).

cor     um vetor de cores em que o número de cores deverá ser igual ao
número de populações diferentes que serão analisadas. Haverá valor default
de cores retirado da paleta "YlOrRd" do pacote RColorBrewer.
Details:
Esta função requer a utilização dos pacotes: "hierfstat", "ggmap", "ggplot2"
e "RColorBrewer".

O cáculo do FST é feito por meio da função pairwise.neifst() do pacote
"hierstat", que estima o FST par a par de acordo com Nei (1987).

O mapa deve ser feito antes de executar a função utilizando as funções
get_map(), ggmap() e ggplot() (pacotes "ggmap" e "ggplot2"). Mais detalhes
no exemplo.

Para plotar o heatmap no mapa é utilizada a função stat_bin2d() do pacote
"ggplot2", que utiliza as colunas do argumento dados.
Value:

comp1 : um heatmap representando os valores de FST para a par de cada
população com relação a população de referência identificada previamente e
os valores de FST e suas cores correspondentes na legenda.

Warning:
Deve-se confirmar se todos os pacotes necessários são compatíveis com a
versão do R utilizada e se todos foram carregados corretamente.
```

```
Deve-se confirmar se o nome das colunas do argumento dados são os mesmos que
estão na decrição deste argumento.

Deve-se confirmar se o nome da população alvo teve os espaços substituídos
por pontos.

Author(s):
Fernanda Bocalini
Contact Info: fernanda.bocalini@usp.br
References:
Nei, M. (1987) Molecular Evolutionary Genetics. Columbia University Press

Examples:
## 1) Lendo e transformando as sequências alinhadas:

library(ape) #pacote para ler as sequências no formato fasta
rhopias_nd2 <- read.dna("rhopias_ali.fas",format="fasta") # lê as sequências
alinhadas do artigo Batalha-Filho & Myiaki (2016)
write.csv(rhopias_nd2, file="rhopias_data.csv") # transformando DNAbin em
data.frame

data.dna <- read.csv("rhopias_data.csv", header = TRUE)  # le o data.frame
criado a partir das sequências alinhadas
#
substituir o nome ou número do espécime
#
pelo nome da população que pertence

dados <- read.table("rhopias2.csv",header = T, sep = ",") # le os dados
(Coordenadas e população)

dados <- subset(dados, select = c("Lat", "Long", "Pop")) # indexando as
colunas de interesse
data.dna[,1] <- dados$Pop # substituindo o nome da espécie e número do
genBank pelo nome da população na primeira coluna

## 2) Criando o mapa do Brasil:

library(ggmap) # pacote para baixar o mapa do google maps
library(ggplot2) # pacote para plotar o mapa
brasil <- get_map(location = "brazil", zoom = 4, maptype = "terrain") #
baixa o mapa de uma localidade no googlemap
BrasilMap <- ggmap(brasil, base_layer = ggplot(aes(x = Long, y = Lat), data
= dados)) # plota o raster produzido pela função get_map()

## Utilizando a função sem modificar as cores:

popgen.map(x=data.dna, dados=data, alvo="Bahia", mapa=BrasilMap, cor="cor")

## Modificando as cores
red <- c(colorRampPalette(brewer.pal(9,"Reds"))(6)) #criando um vetor de
```

```
cores

popgen.map(x=data.dna, dados=data, alvo="Bahia", mapa=BrasilMap, cor=red)
```

Arquivos utilizados nos exemplos

Sequências alinhadas de ND2 de *Rhopias gularis* segundo Batalha-Filho & Miyaki (2016). rhopias_ali.fas.txt apagar a extensão .txt (arquivos .fas são proibidos no wiki)

Dado de latitude, longitude e população de *Rhopias gularis* segundo Batalha-Filho & Miyaki (2016). rhopias2.csv

Referência

Batalha-Filho, H., & Miyaki, C. Y. (2016). Late Pleistocene divergence and postglacial expansion in the Brazilian Atlantic Forest: multilocus phylogeography of Rhopias gularis (Aves: Passeriformes). Journal of Zoological Systematics and Evolutionary Research.