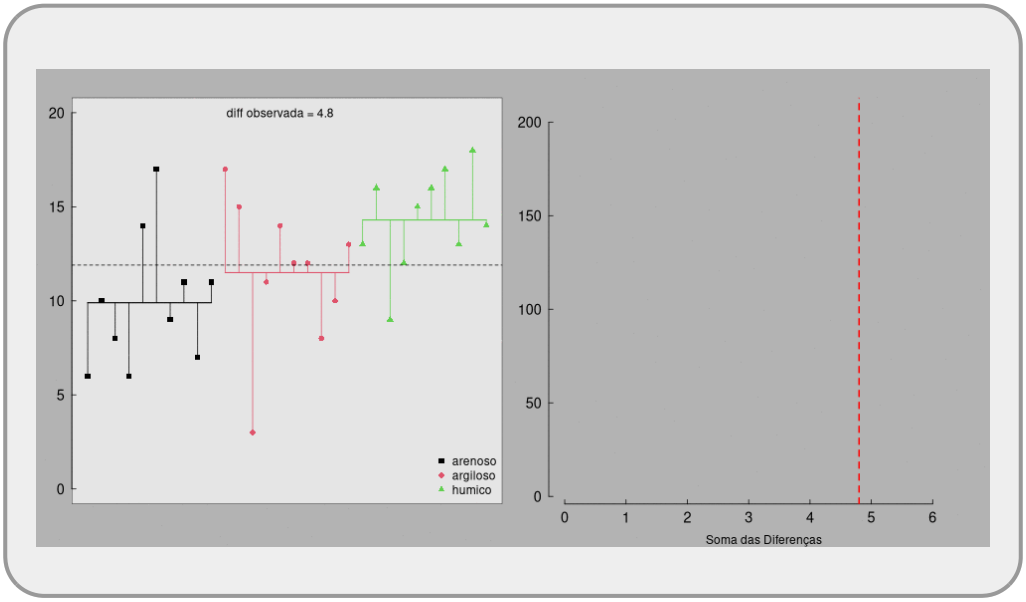
- Tutorial
- Exercícios
- Apostila

8. Reamostragem e Simulação



Os métodos de Monte Carlo são procedimentos de simulação computacional para soluções de problemas complexos. Eles são utilizados em principalmente três campos da matemática: otimização, integração numérica e aleatorização de amostras de funções probabilísticas. Aqui vamos focar em sua base mais simples: a aleatorização ou permutação para gerar distribuições probabilísticas do cenário nulo e calcular a probabilidade do resultado obtido ter sido gerado pelo acaso e/ou calcular o intervalos de confiança de alguma estatística de interesse. Especificamente para o testes de hipótese por aleatorização ou randomização dos dados, temos ao menos as seguintes etapas:

1. Definir a estatística de interesse (EI)
2. Calcular a estatística de interesse a partir dos dados
3. Estabelecer o cenário nulo
4. Simular cenário nulo
5. Calcular a EI no cenário nulo (pseudovalor)
6. Produzir a distribuição dos pseudovalores
7. Posicionar a EI observada na distribuição dos pseudovalores
8. Calcular o p-valor

Para esses testes iremos precisar apenas de duas instrumentações poderosas que já utilizamos em outros tópicos da linguagem R: a função `sample()` e controle de fluxo com ciclos de iteração usando o `for()`.

Função sample

A função `sample()` amostra aleatoriamente elementos de um objeto `x`. Se não utilizarmos nenhum argumento, a função irá embaralhar os elementos do objeto, ou seja, montar um vetor de mesmo tamanho com os elementos alocados aleatoriamente. Para montar vetores de tamanhos diferentes do original precisamos indicar o tamanho do vetor resultado com o argumento `size`. Quando colocamos o argumento `replace = TRUE` os elementos do vetor `x` são amostrados com reposição, ou seja, podem ser amostrados mais do que uma vez, sendo que no padrão FALSE, cada elemento pode ser amostrado apenas um vez. Por fim, o argumento `prob` recebe um vetor de valores de mesmo tamanho que `x` e que definem a probabilidade de amostrar cada elemento do vetor original. Por exemplo, `prob = c(0.5, 1, 1.5, 2)`, significa que o elemento `x[4]` tem o dobro de probabilidade de ser amostrado do que `x[2]` e quatro vezes mais que o `x[1]`.

Vamos criar um vetor a partir do objeto LETTERS, com letras de “A” a “J” e aplicar a função `sample` nele.

```
vetor <- rep(LETTERS[1:10])
vetor
sample(vetor)
sample(vetor, replace = TRUE)
sample(vetor, 40, replace = TRUE)
sample(vetor, prob =
c(0.1,0.2,0.05,0.05,0.2,0.1,0.05,0.05,0.1,0.1),replace=TRUE)
sample(vetor, prob = c(1,2,0.5,0.5,2,1,0.5,0.5,1,1), replace = TRUE)
```

Revisitando o teste de hipótese

Agora vamos revisitar os dados de Chacal Dourado e a pergunta se há diferença no tamanho de mandíbulas entre machos e fêmeas, onde exemplificamos o teste de hipótese no 6a. Teste de Hipótese.

```
macho=c(120,107,110,116, 114, 111, 113,117,114,112)
femea=c(110,111,107, 108,110,105,107,106,111,111)
macho
femea
sexo=rep(c("macho", "femea"), each=10)
sexo
mf=c(macho,femea)
mf
macho.m=mean(macho)
macho.m
femea.m=mean(femea)
femea.m
macho.m-femea.m
dif.mf=diff(tapply(mf,sexo,mean))
dif.mf
```

PERGUNTAS:

- Essa diferença entre as médias é significativa?
- Qual minha incerteza ao afirmar que essas médias são diferentes?

Se a variação encontrada é devido à variações não relacionadas ao sexo, é possível gerar essa diferença permutando os dados. Caso isso seja verdade encontraremos frequentemente diferenças iguais ou maiores que a observada.

No código abaixo estamos aleatorizando o vetor `mf` em relação ao vetor `sexo` e calculando a estatística de interesse novamente a partir dessa simulação, e gerando o pesudovalor em seguida. Repetimos esse procedimento algumas vezes:

```
s1.mf=sample(mf)
s1.mf
diff(tapply(s1.mf,sexo,mean))
##+1
s2.mf=sample(mf)
s2.mf
diff(tapply(s2.mf,sexo,mean))
##+2
diff(tapply(sample(mf),sexo,mean))
##+3
diff(tapply(sample(mf),sexo,mean))
##+1000
### e agora? fazer na mão 1000 vezes? ###
```

Criando ciclos de eventos

Para repetir esse procedimento muitas vezes utilizamos um controle de fluxo com a função `for()`, que tem a seguinte estrutura:

Ciclos de iteração

```
for(var in seq)
{

}
```

Onde:

- `var` é o nome sintético para uma variável
- `seq` vetor de valores que será assumido por `var`
- `{}` expressões de procedimentos a serem repetidos

Antes de iniciar os ciclos de iteração é desejável criar o objeto que irá armazenar os resultados de cada ciclo. Note que no caso abaixo criamos o objeto `result` e incluímos na sua primeira posição o valor de diferença observada entre os tamanhos médios de mandibulas de machos e fêmeas. Note também que, a variável de iteração vai assumir os valores de 2 a 1000 nos ciclos.

```
result <- rep(NA, 1000)
result[1] <- diff(tapply(mf, sexo, mean))
for(i in 2:1000)
    {
    dif.dados <- diff(tapply(sample(mf), sexo, mean))
    result[i] <- dif.dados
    }
```

Um primeiro passo é fazer um gráfico com esse vetor de resultados:

```
hist(result)
abline(v = result[1], col = "red")
abline(v = result[1]*-1, col = "red")
```

Cálculo do P

Duas perguntas distintas podem ser colocadas nesse teste de hipótese. Se há diferença entre os tamanhos ou se um tamanho é maior (menor) que outro, como já vimos no teste de hipótese.

Para o cálculo do p-valor da afirmação que há diferença temos:

```
bicaudal=sum(result>=result[1]| result<=(result[1]*-1))
bicaudal
length(result)
p.bi=bicaudal/length(result)
p.bi
```

Para se os machos são maiores que as fêmeas:

```
unicaudal=sum(result>=result[1])
unicaudal
p.uni=unicaudal/length(result)
p.uni
```

Simula T

No tutorial de 6a. Teste de Hipótese também utilizamos uma função que automatiza esse teste de hipótese por simulação chamada

simulaT}}preservefilenames::simulaT.r

. Para relembrar, baixe a função e refaça o teste:

```
x11(width = 10, height = 10)
source("simulaT.r")
simulaT(macho, femea, teste = "maior", anima = TRUE)
```

As funções não precisam ser consideradas procedimento abstrato no qual não temos acesso. Uma função similar a essa foi criada durante a aula no curso de 2012, com o código que aprenderam neste

tópico. Abra o arquivo da função em um editor de texto e reconheça todos os comando que estão nas linhas de código da função. Na próxima aula iremos entender como incorporar um procedimento em uma função. O primeiro passo é saber executar o procedimento em linhas de código, como fizemos no início do tutorial.

Bootstrap

Bootstrap é outro método de simulação computacional para calcular a imprecisão associada a uma estimativa da população estatística. O procedimento é bastante simples, amostramos com reposição o mesmo número de elementos do vetor de dados e recalculamos a estimativa de interesse. Baseado na premissa que nossa amostra é representativa da nossa população estatística, conseguimos calcular os intervalos de confiança das estimativas.

No exemplo abaixo utilizaremos os mesmos dados anteriores para exemplificar o procedimento bootstrap para calcular o intervalo de confiança da média dos machos do chacal dourado.

Primeiro vamos ver novamente esses dados e sua média:

```
macho
macho.m
```

Agora, fazemos um aleatorização deste vetor e calcular novamente a média:

```
mean(sample(macho))
```

Essa média não é diferente da anterior, porque mudar a posição dos valores não afeta a estimativa da média. No entanto, se usarmos uma reamostragem com reposição (amostrar um valor e depois retorná-lo, antes de amostrar o próximo), permite que os valores já amostrados apareçam novamente na nova amostra. Vamos fazê-lo:

```
smacho <- sample(macho, replace = TRUE)
mean(smacho)
mean(sample(macho, replace = TRUE))
mean(sample(macho, replace = TRUE))
```

Perceba que as últimas linhas de comando produzem valores diferentes apesar de serem as mesmas. Esse processo é similar ao que usamos para fazer amostras de uma distribuição conhecida com o *rnorm()* e *rpois()*, só que agora os valores passíveis de serem amostrados são apenas aqueles presentes nos nossos dados. Se repetirmos esse procedimento muitas vezes e guardarmos os resultados de cada simulação de amostras com reposição, teremos um conjunto de pseudo-valores que representam a distribuição do nosso parâmetro e portanto, podemos calcular o intervalo de confiança que desejarmos a partir dessa distribuição. Como repetimos uma operação muitas vezes no R? Usando novamente os ciclos produzidos pela função `for(… in …)`, vamos fazer então 100 simulações:

```
nsim <- 100
resulta <- rep(NA,nsim)
for(i in 1:nsim)
 {
```

```
  resulta[i] <- mean(sample(macho, replace = TRUE))
 }
resulta
```

Agora precisamos calcular o intervalo de confiança, chamado **intervalo bootstrap**, para o limite que interessa (95%, 99%...). Vamos calcular para um intervalo de 90%. Uma forma de faze-lo é ordenando os valores e olhado quais valores estão nos extremos com 5% de cada lado.

```
sort(resulta)
sort(resulta)[6] ## o valore que deixa as 5 menores de fora
sort(resulta)[95] ## o valore que deixa os 5 maiores de fora
```

Podemos também usar a função `quantile()` definindo os quantis de interesse:

```
quantile(resulta, prob=c(0.05, 0.95))
```

Definitivamente, fazer só 100 simulações, não parece adequado. Existem muitos arranjos possíveis de 10 elementos reamostrados com reposição[1)]. Refaça o código com 1000 (mil) iterações e recalcule o intervalo.

Tesourinha e a deriva continental

Para fechar nosso tutorial vamos reproduzir uma análise mais complexa, que foi publicada em um artigo na Nature em 1966 (*Geographical Distribution of the Dermaptera and the Continental Drift Hypothesis*) e descrita no primeiro capítulo do livro do Manly sobre permutação. A ideia era verificar se a ocorrência de tesourinhas (*Dermaptera*) estava mais correlacionada com a distribuição dos continentes atual ou antes da deriva continental. A informação de interesse é a correlação da ocorrência de tesourinha entre diferentes regiões biogeográficas: Eurasia, África, Madagascar, Oriente, Austrália, Nova Zelândia, América do Sul e América do Norte. Valores positivos próximos a 1 representam composições de comunidades muito parecidas, valores próximos a -1 representam composição muito distintas. Vamos reconstruir essa matriz no objeto `data.coef`:

```
data.coef<-matrix(c(NA, .30, .14, .23, .30, -0.04, 0.02, -0.09, NA, NA,
.50,.50, .40, 0.04, 0.09, -0.06, NA, NA, NA, .54, .50, .11, .14, 0.05,
rep(NA, 4), .61, .03,-.16, -.16, rep(NA, 5), .15, .11, .03, rep(NA, 6), .14,
-.06, rep(NA, 7), 0.36, rep(NA, 8)), nrow=8, ncol=8)
rownames(data.coef) <- c("Eur_Asia", "Africa", "Madag", "Orient", "Austr",
"NewZea", "SoutAm", "NortAm")
colnames(data.coef) <- c("Eur_Asia", "Africa", "Madag", "Orient", "Austr",
"NewZea", "SoutAm", "NortAm")
data.coef
```

Foram usadas nesse estudo outras duas matrizes de distância, a primeira representando o número de eventos de dispersão de longas distâncias necessários para a conexão de populações na configuração atual dos continentes e a outra na configuração antes da deriva continental, entre as mesmas regiões biogeográficas.

```
dist.atual<-matrix(c(NA,1,2,1,2,3,2,1, NA, NA, 1,2,3,4,3,2, NA, NA,
```

```
NA,3,4,5,4,3, rep(NA, 4),1,2,3,2, rep(NA, 5), 1,4,3, rep(NA, 6), 5,4,
rep(NA, 7), 1, rep(NA, 8)), nrow=8, ncol=8)
dist.atual
dist.deriva<- matrix(c(NA,1,2,1,2,3,2,1, NA, NA, 1,1,1,2,1,2, NA, NA,
NA,1,1,2,2,3, rep(NA, 4),1,2,2,2, rep(NA, 5), 1,2,3, rep(NA, 6), 3,4,
rep(NA, 7), 1,  rep(NA, 8)), nrow=8, ncol=8)
# colocando nomes nas matrizes
rownames(dist.atual) <- colnames(dist.atual)<- c("Eur_Asia", "Africa",
"Madag", "Orient", "Austr", "NewZea", "SoutAm", "NortAm")

colnames(dist.deriva)<- rownames(dist.deriva)<-  c("Eur_Asia", "Africa",
"Madag", "Orient", "Austr", "NewZea", "SoutAm", "NortAm")
# olhando as matrizes
dist.atual
dist.deriva
```

A primeira parte da análise dos dados é calcular a correlação entre a matriz de similaridade taxonômica e de eventos de dispersão (atual e antes da deriva). Para isso, calculamos o coeficiente de correlação de *Pearson* entre as matrizes. Esse valor irá nos dizer se duas matrizes estão correlacionadas. A correlação pode ser positiva (até +1) se variações nos elementos de uma matriz levam a variações na mesma direção dos elementos correspondentes na outra , negativa quando em direção contrária (até -1), ou podem ser não relacionadas (≅0).

$$ r = \frac{\sum_1^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_1^n{(x_i-\bar{x})^2}}\sqrt{\sum_1^n{(y_i-\bar{y})^2}}}$$

```
cor12<-cor(as.vector(data.coef), as.vector(dist.atual), use="complete.obs")
cor13<-cor(as.vector(data.coef), as.vector(dist.deriva), use="complete.obs")
cor12 ## correlação com a distancia atual
cor13 ## correlação com a distancia antes da deriva
```

Ambos os valores de correlação estão nos dizendo que, quanto maior a distância geográfica mais diferente é a composição de espécies de tesourinha. Além disso, que a correlação com as distâncias antes da deriva é mais forte. No caso, valores maiores em módulo, já que a relação é de correlação negativa (aumento da distância diminui a similaridade florística).

Agora vamos calcular se esse valores de correlação poderiam ser atribuídos ao acaso. Para isso, vamos fazer a permutação de uma das matrizes e calcular o coeficientes de Pearson, após essa permutação. A permutação é simples, vamos mudar as colunas e linhas de lugares de maneira a aleatorizar os valores, mas manter a estrutura subjacente ao dados. Uma maneira de fazer é:

```
data.sim<-data.coef # copia da matriz que será aleatorizada
data.sim

# preenchendo o triangulo superior da matriz com os dados correspondentes do
triangulo inferior
data.sim[upper.tri(data.sim)] <- t(data.coef)[(upper.tri(data.coef))]

data.sim # olhando a matriz
data.sim[8:1, 8:1] # uma matriz baguncada mas que mantem certa estrutura
sim.pos<-sample(1:8) # posicoes permutadas
```

```
sim.pos
data.sim<-data.sim[sim.pos, sim.pos] # aqui uma matriz verdadeiramente
permutada
cor12.sim<-cor(as.vector(data.sim), as.vector(dist.atual),
use="pairwise.complete.obs")
cor13.sim<-cor(as.vector(data.sim), as.vector(dist.deriva),
use="pairwise.complete.obs")
cor12.sim
cor13.sim
cor12 ## correlação observada com a distancia atual
cor13 ## correlação observada com a distancia antes da deriva
```

Para reproduzir muitas vezes o procedimento acima, vamos colocá-lo dentro de um ciclo de iteração, não sem antes criar o objeto para guardar todos os valores que queremos.

```
res.cor=data.frame(sim12=rep(NA, 5000), sim13=rep(NA,5000))
str(res.cor)
res.cor[1,]<-c(cor12, cor13)
str(res.cor)
for(s in 2:5000)
    {
        sim.pos<-sample(1:8)
        data.sim<-data.sim[sim.pos, sim.pos]
        res.cor[s,1]<-cor(as.vector(data.sim), as.vector(dist.atual),
use="pairwise.complete.obs")
        res.cor[s,2]<-cor(as.vector(data.sim), as.vector(dist.deriva),
use="pairwise.complete.obs")
    }
```

Por fim, vamos avaliar os resultados e calcular o p-valor:

```
str(res.cor)
par(mfrow=c(2,1))
hist(res.cor[,1])
abline(v=res.cor[1,1], col="red")
hist(res.cor[,2])
abline(v=res.cor[1,2], col="red")
#### calculando o P ###########
p12=sum(res.cor[,1]<= res.cor[1,1])/(dim(res.cor)[1])
p12
p13=sum(res.cor[,2]<= res.cor[1,2])/(dim(res.cor)[1])
p13
```

Um fase muito importante é a interpretação dos resultados de testes como esse, que não está no escopo deste curso. De qualquer forma, consegue imaginar a conclusão do artigo para esse resultado?

Para saber mais

Veja a aba da apostila deste mesmo tópico. Ali apresentamos outros conceitos. Dois livros são muito importantes e lançaram as bases das análises de Monte Carlo na ecologia:

- Manly B. F. J., 1997 Randomization, bootstrap and Monte Carlo methods in biology. 2nd Ed., Chapman and Hall, London
- GOTELLI, N. J. & G. R. GRAVES. 1996. Null models in ecology. Washington and London, Smithsonian Institution Press

Caso tenham interesse pelo assunto sugiro iniciar por eles. O livro do Gotelli está esgotado, mas o autor disponibiliza o PDF em seu site.

1)
se não estou equivocado o número de arranjos é 10^{10}, mas como no nosso cálculo a ordem dos elementos não importa, esse número é efetivamente menor

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

Permanent link:
http://ecor.ib.usp.br/doku.php?id=02_tutoriais:tutorial9:start&rev=1655847119

Last update: **2022/06/21 18:31**