- Apostila
- Tutorial
- Exercícios

8. Reamostragem e Simulação

O R é uma ferramenta poderosa para simular situações e compará-las com padrões obeservados. O assunto é extenso, e aqui apresentaremos apenas exemplos simples das modalidades mais usadas dem ecologia. Para quem quiser se se aprofundar, um bom começo é Manly (1997 [1)]).

A Função ''sample''

Esta função reamostra os elementos de um vetor:

```
> meu.vetor
 [1] "A" "A" "B" "B" "C" "C" "D" "D" "E" "E"
> meu.vetor.bagunçado <- sample(meu.vetor)
> meu.vetor.bagunçado
 [1] "C" "A" "A" "D" "B" "E" "E" "C" "D" "B"
```

Por *default* a função retorna um novo vetor de mesmo comprimento, com os elementos permutados ao acaso. Portanto, é uma amostragem **sem reposição**. A função tem argumentos que definem se a amostragem é ou não com reposição, o tamanho do vetor resultante, entre outros. Consulte a página de ajuda.

Exercícios

Exercício: *Quem Precisa da Caixa Econômica?*

Você pode simular sorteios das loterias com a função sample. Consulte a ajuda da função para descobrir como simular um sorteio como o feito para o jogo da sena.

Reamostragens sem Reposição: Permutações

Nesse procedimentos apenas “embaralhamos” os elementos de um ou mais vetores, o que podemos fazer gerando um vetor de mesmo tamanho, cujos elementos são amostrados ao acaso **sem reposição**. Esse procedimento é chamado de permutação ao acaso.

Testes de Permutação

Uma das aplicações simples das permutações ao acaso são testes de comparação de médias entre

grupos, como o teste t, como mostramos abaixo.

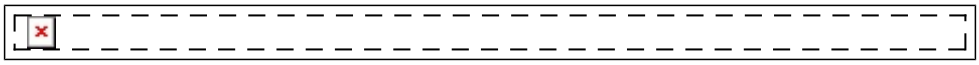
Os dados que usaremos são numero de besouros no conteúdo estomacal de lagartos *Phrynosoma brevirostrae* machos e femeas (Manly,1997):

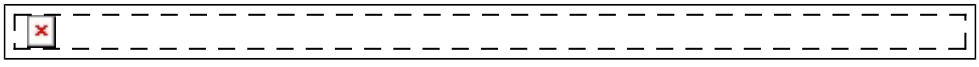
```
<code>
> lagartos <- data.frame(sexo=factor(rep(c("m","f"),c(24,21))),
+
ncoleop=c(256,209,0,0,0,44,49,117,6,0,0,75,34,13,0,90,0,32,0,205,332,0,31,0,
+
0,89,0,0,0,163,289,3,843,0,158,443,311,232,179,179,19,142,100,0,432))
```

Uma análise exploratória dos dados mostra que as médias amostrais são muito diferentes, mas as variâncias também, e a variação é muito grande, devido ao grande número de animais sem sinais de besouros no estômago:

```
> tapply(lagartos$ncoleop,lagartos$sexo,summary)
$f
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    0.0     0.0   142.0   170.6   232.0   843.0

$m
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00    0.00   22.00   62.21   78.75  332.00

##Médias
> medias <- tapply(lagartos$ncoleop,lagartos$sexo,mean)
> medias
        f         m
170.57143  62.20833
##Diferença absoluta entre as médias
> dif.obs <- abs( diff( as.vector(medias) ) )
> dif.obs
[1] 108.3631

##Variâncias
> tapply(lagartos$ncoleop,lagartos$sexo,var)
        f         m
43533.557  8855.911
```

Os gráficos de quantis téoricos indicam um forte desvio em relação à distribuição normal:





Como os dados não atendem as premissas de um teste t, uma alternativa é substituí-lo por um teste de permutação. Como a hipótese nula é que as duas amostras vêm de populações com a mesma média, ela pode ser simulada permutando-se ao acaso os valores entre os sexos.

Calculamos então um índice de diferenças entre as médias das amostras que deve ser similar ao obtido, caso a hipótese nula esteja correta. Repetindo esse procedimento milhares de vezes, podemos estimar a chance de um valor igual ou maior que o observado ter ocorrido mesmo se as duas amostras vêm de populações com médias diferentes.

A permutação é feita com a função `sample`, que pode ser repetida com um *loop*, por meio da função `for`:

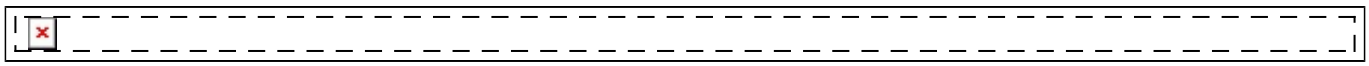
```
> ##cria um vetor para armazenar os resultados
> results <- c()
> ##Permuta os valores das medidas, calcula a diferença absoluta entre as
médias e
> ##armazena no vetor "results". Repete a operação mil vezes
> for(i in 1:1000){
+ results[i] <- abs( diff(
tapply(sample(lagartos$ncoleop),lagartos$sexo,mean) ) )
+ }
```

Em apenas 15 das mil permutações a diferença absoluta das médias foi igual ou maior do que a observada:

```
> sum(results >= dif.obs)
[1] 15
```

Logo, a diferença observada é pouco provável sob a hipótese nula, o que nos permite rejeitá-la. Um gráfico de densidade probabilística dos valores das permutações ilustra isso:

```
> plot(density(results),xlab="Diferença Absoluta das Médias",ylab="Freq
Relativa", main="")
> abline(v = dif.obs, col="red")
```



Modelos Nulos em Ecologia de Comunidades

Modelos nulos em ecologia de comunidades [2)] muitas vezes envolvem permutação de elementos de uma matriz, em geral de ocorrência ou abundância de espécies por local.

Vamos imaginar uma dessas matrizes, com seis espécies (linhas) e seis locais (colunas):

```
> cc2
    [,1] [,2] [,3] [,4] [,5] [,6]
sp1    1    1    0    0    0    0
sp2    1    1    1    0    0    0
sp3    0    0    1    1    0    0
sp4    0    0    1    1    1    0
sp5    0    0    0    0    1    1
sp6    0    0    0    0    1    1
> #Numero de especies por local
```

```
> S.obs <- apply(cc2,2,sum)
> S.obs
[1] 2 2 3 2 3 2
> ##Maximo de especies em coexistencia
> max(S.obs)
[1] 3
```

Podemos nos perguntar se há um limite ao número de espécies que podem coexistir. Se isso ocorre, o número máximo de espécies em co-ocorrência deve ser menor do que se distribuirmos as espécies ao acaso pelos locais. Há vários cenários nulos possíveis, mas ficaremos aqui com permutação dos locais em cada espécie ocorreu.

Para isso, basta permutar os valores dentro de cada linha:

```
> apply(cc2,1,sample)
     sp1 sp2 sp3 sp4 sp5 sp6
[1,]   1   0   0   1   0   1
[2,]   0   1   0   0   1   0
[3,]   0   1   1   1   0   0
[4,]   0   0   0   1   1   0
[5,]   0   1   0   0   0   1
[6,]   1   0   1   0   0   0
```

Ops! a função `apply` retorna sempre seus resultados em colunas, transpondo a matriz original. Para evitar isso, transpomos o resultado, voltando as espécies para as linhas:

```
> t(apply(cc2,1,sample))
    [,1] [,2] [,3] [,4] [,5] [,6]
sp1    0    1    0    0    1    0
sp2    1    1    0    0    1    0
sp3    0    1    0    1    0    0
sp4    0    1    0    0    1    1
sp5    1    0    0    0    0    1
sp6    1    0    1    0    0    0
```

Agora basta repetir essa permutação muitas vezes e calcular o número máximo de espécies em co-ocorrência. Para esse cálculo, podemos criar uma função:

```
> ##Uma funcao para calcular o maximo de especies em co-ocorrencia
> max.c <- function(x){ max(apply(x,2,sum)) }
> max.coc <- max.c(cc2)
> max.coc
[1] 3
```

E então repetimos as permutações 1000 vezes, inserindo-as em um *loop*:

```
##Um vetor para guardar os resultados
> results <- c()
## Mil simulações em um loop
> for(i in 1:1000){
+   result.c[i] <- max.c( t(apply(cc2,1,sample)))
```

```
+ }
```

O número de randomizações que teve um máximo de espécies por local menor ou igual ao observado foi alto (21,6%), portanto a observação de que no máximo três espécies ocorrem no mesmo local não é evidência de que haja um limite a esse número:

```
> sum(result.c<=max.coc)
[1] 216
```

Reamostragem com Reposição: Bootstrap

Como a maioria das boas idéias, o conceito geral do bootstrap é muito simples:

> *Se aceitamos que nossa amostra é a melhor informação disponível que temos sobre uma população estatística, podemos simular uma nova coleta de dados reamostrando os elementos de nossa amostra, com reposição.*

A aplicação mais simples deste princípio é estimar intervalos de confiança por reamostragem. Para exemplificar, usaremos o conjunto de dados BCI, um dataframe de abundâncias de espécies de árvores na parcela permanente de Barro Colorado nas (linhas), por subparcelas de 1 ha:

```
##Esse objeto de dados está no pacote "vegan"
> data(BCI, package="vegan")
##Visão de três linhas e três colunas
> BCI[1:3,25:28]
  Brosimum.guianense Calophyllum.longifolium Casearia.aculeata
Casearia.arborea
1                  0                        0                 0
1
2                  0                        2                 0
1
3                  0                        0                 0
3
```

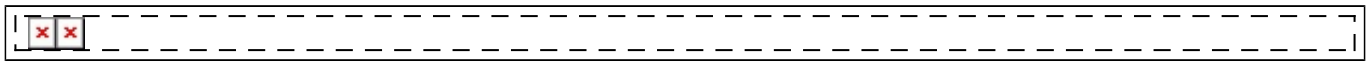
Vamos criar uma função para calcular o índice de Shannon a partir de um vetor de abundâncias de espécies, e aplicar a cada parcela:

```
> H <- function(x){
+   y <- x[x>0]
+   pi <- y/sum(y)
+   -sum(pi*log(pi))
+ }
> H.BCI <- apply(BCI,1,H)
```

O vetor H.BCI contém os índices de Shannon de cada parcela de 1 ha. Sua distribuição não parece ser normal:

```
> hist(H.BCI,xlab="Índice de Shannon")
```

```
> qqnorm(H.BCI);qqline(H.BCI)
```



Vamos simular uma amostra deste conjuntos de parcelas, e estimar o intervalo de confiança da diversidade média por parcela:

```
> ##Uma amostra ao acaso de 20 parcelas
> bci.sample <- BCI[sample(1:nrow(BCI),20),]

> ##Estimativas da média e intervalo de confiança baseados na distribuição
de t de Student:
> t.test(H.am)

        One Sample t-test

data:  H.am
t = 114.8506, df = 19, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 3.783456 3.923914
sample estimates:
mean of x
 3.853685
```

A função `t.test` quando aplicada a um vetor de números sem nenhum outro argumento retorna o teste t com a hipótese nula de que o vetor é uma amostra de uma população com distribuição normal e média zero. O intervalo de confiança da média da população de onde veio a amostra também é calculado.

No caso, esse intervalo não inclui o zero, o que rejeita a hipótese nula. Mas o que interessa aqui é que o intervalo inclui a média da população de parcelas de onde veio a amostra, que é :

```
> mean(H.BCI)
[1] 3.82084
```

Se a amostra vem de uma população de valores com distribuição normal, o intervalo deve incluir a média da população em 95% das vezes que uma amostragem for realizada [3)].

Como a premissa de normalidade parece não ser verdadeira, uma solução é calcular um intervalo por *bootstrap*. Para isso, basta re-amostrar com reposição a amostra, e calcular a estatística de interesse, no caso a média dos índices de Shannon de cada parcela:

```
##Um vetor para armazenar os resultados
> results <- c()
##Reamostrando 1000 vezes com reposição
> for(i in 1:1000){ results[i] <- mean( sample(H.am,replace=T) )}
```

A diferença entre a média da amostra original e a média das aleatorizações estima o viés, no caso muito pequeno:

```
> mean(results)
[1] 3.853739
> mean(H.am) - mean(results)
[1] -5.396726e-05
```

O intervalo de confiança por quantis a 95% está compreendida entre o valor abaixo do qual há apenas 2,5% das randomizaçôes e o valor acima do qual restam apenas 2,5% das randomizações. Portanto, são os quantis a 2,5% e 97,5% dos mil valores simulados. Usamos a função `quantile` para obtê-los:

```
> quantile(results, probs=c(0.025,0.975))
    2.5%    97.5%
3.788261 3.912513
```

No caso, não houve diferença importante entre os intervalos de confiança estimados por *bootstrap* e pela distribuição de t de Student.

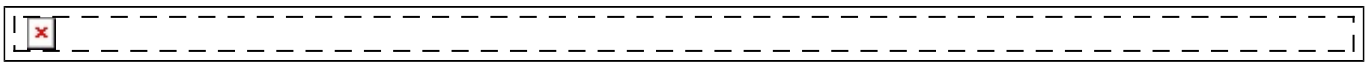
Simulações com Distribuições Teóricas

O R gera valores amostrados de distribuições teóricas de probabilidades (ver Distribuições Estatísticas: Funções no R), o que pode ser usado em simulações de diversas maneiras.

Um delas é avaliar a eficácia de um procedimento, como a força de um teste de significância em diferentes situações. O exemplo a seguir avalia a força de um teste t comparando duas amostras tomadas de duas populações com distribuição binomial negativa.

Para ilustrar a forma dessas distribuições, podemos fazer o gráfico dos valores teóricos de probabilidade, no intervalo de zero a 25:

```
> x <- 0:25
> y1 <- dnbinom(x,mu=2, size=2)
> y2 <- dnbinom(x,mu=3, size=2)
> plot(y1~x, type="b", xlab="Valores",ylab="probabilidade",
+      ylim=c(0,max(c(y1,y2))), pch=17 )
> points(y2~x, col="blue", type="b")
> legend(8,0.2,c("média=2, var=4","média=3, var=7,5"),lty=1,
+        col=c("black","blue"), pch=c(17,1))
```



As premissas do teste t são que as amostras vêm de populações com distribuições normais que têm variâncias iguais (pelo menos aproximadamente). A distribuição binomial negativa tem uma forma muito diferente da normal, e nela a variância é uma função da média. Assim, duas distribuições binomiais negativas com médias diferentes, como as ilustradas acima, terão variâncias diferentes. Além disso, a distribuição binomial negativa é **discreta**, enquanto a normal é **contínua**.

Em resumo, usar o teste t para comparar as médias de amostras tomadas de populações com distribuição binomial negativa implica em desconsiderar as premissas do teste, o que pode comprometer sua força. Mas como avaliar isso? Podemos simular amostras tomadas de populações

com distribuição binomial negativa com médias diferentes, aplicar o teste t e contar o número de simulações em que o teste indicou uma diferença significativa entre as médias, a 5%.

Para simular a comparação de duas amostras de tamanho dez, usando um *loop* temos:

```
> ##Um vetor fator com 20 elementos, para representar cada elemento das duas
amostras
> tratamento <- factor(x=rep(c("a","b"),each=10))
> ##Um vetor para armazenar os resultados da simulação
> results <- c()
> ##Dez mil simulações, com um loop e a função t.test para realizar o teste
t
> for(i in 1:10000){
+   variaveis <- c(rnbinom(n=10,size=2,mu=2),rnbinom(n=10,size=2,mu=3))
+   results[i] <- t.test(variaveis~tratamento)$p.value
+ }
> ##Quantas das simulações indicaram uma diferença significativa das médias?
> sum(results<0.05)
[1] 1148
```

O teste detectou corretamente que as amostras vêm de populações com médias diferentes em 1148 das dez mil simulações, o que estima a força do teste neste caso em $\beta = 0{,}1148$ [4)].

Vamos agora avaliar o que acontece se tomamos amostras de populações com distribuição normal, mantendo as mesmas variâncias:

```
> for(i in 1:10000){
+    variaveis <- c(rnorm(n=10,mean=2,sd=2),
+                   rnorm(n=10,mean=3,sd=sqrt(7.5)))
+    results[i] <- t.test(variaveis~tratamento)$p.value
+  }
> sum(results<0.05)
[1] 1389
```

Como poderíamos esperar, a força estimada foi maior, pois uma das premissas do teste agora é satisfeita. No entanto, a premissa de igualdade de variâncias ainda não é cumprida. Se simulamos amostras com variâncias iguais à média das duas variâncias usadas nas simulações anteriores, temos um ganho na força do teste, embora pequeno:

```
> ## Desvio-padrão médio correspondente às variâncias de 7,5 e 4
> sd1 <- mean(sqrt(c(7.5,4)))
> ## Verificando o valor
> sd1
[1] 2.369306
> for(i in 1:10000){
+    variaveis <- c(rnorm(n=10,mean=2,sd=sd1),
+                   rnorm(n=10,mean=3,sd=sd1))
+    results[i] <- t.test(variaveis~tratamento)$p.value
+  }
> sum(results<0.05)
```

```
[1] 1419
```

Mesmo nesta situação ideal com todas as premissas cumpridas, em apenas 14% das tentativas o teste detectará a diferença que existe entre as populações. Isto acontece pelo pequeno tamanho amostral. As simulações podem prosseguir para avaliar o efeito de diferentes tamanhos amostrais.

Exercícios

Exercício: *Que simulação é esta?*

Descubra que simulação é feita com código abaixo:

```
> m1 <- matrix(data=rnbinom(n=10000,size=2,mu=2),nrow=10,ncol=1000)
> m2 <- matrix(data=rnbinom(n=10000,size=2,mu=3),nrow=10,ncol=1000)
> matrizona <- rbind(m1,m2)
> tratamento <- factor(x=rep(c("a","b"),each=10))
> signif.t <-
function(valores,fator){(t.test(formula=valores~fator))$p.value}
> sum(apply(X=matrizona,MARGIN=2,FUN=signif.t,fator=tratamento)<=0.05)
```

1)

Manly B. F. J., 1997 Randomization, bootstrap and Monte Carlo methods in biology. 2nd Ed., Chapman and Hall, London

2)

ver GOTELLI, N. J. & G. R. GRAVES. 1996. Null models in ecology. Washington and London, Smithsonian Institution Press

3)

Obs: é um erro comum pensar que o intervalo de confiança é fixo, e a média tem 95% de chance de estar dentro dele. Na verdade, a média populacional é fixa, pois é um parâmetro. O intervalo varia a cada vez que você repetir a amostragem, sendo uma *estimativa por intervalo*

4)

Este valor irá variar a cada repetição da simulação, pois novos valores serão sorteados, mas ficará em torno do obtido. Para repetir uma simulação e obter os mesmos valores defina a mesma semente de números aleatórios com a função `set.seed`.