PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO
Departamento de Engenharia Industrial
Rua Marquês de São Vicente, 225
22453-900 – Rio de Janeiro
Brasil

# ENG1536 – Inferência Estatística

# Laboratório – Guia de Estudo 5

Neste estudo você vai aprender uma das possibilidades mais interessantes do R: como criar novas funções que são incorporadas ao vocabulário do R. Muitas (não todas) das funções na instalação padrão do R são escritas com expressões como as que você aprendeu a criar nos estudos anteriores. Quando esse é o caso, é possível ler a definição da função digitando o seu nome sem parênteses e depois e pressionando ENTER. Por exemplo, uma função muito importante em estatística que estudaremos no futuro é `lm` (*linear model*), que serve para estimar regressões lineares. Digite `lm` na linha de comando para ver o código completo que a define. Não se espera que você compreenda o conteúdo do código.

A criação de funções novas em R utiliza **expressões agrupadas**. Uma expressão agrupada consiste numa sequência de expressões separadas por ponto-e-vírgula, envolvida por chaves. Quando uma expressão agrupada é interpretada pelo sistema, ele avalia todas as expressões que o grupo contém e mostra ao final para o usuário apenas o valor da última expressão no grupo. Digite:

```
{ sqrt(2) ; x <- 2*3 ; x^2 }
```

Veja que o resultado dessa expressão agrupada é “36”, que é o resultado da terceira expressão no grupo, mas tendo sido realizada a designação na segunda expressão. A primeira expressão que calcula a raiz quadrada de 2 é inócua. Verifique também que o objeto `x` criado dentro da expressão agrupada continua existindo na área de trabalho. No âmbito de funções criadas pelo usuário, essa permanência dos objetos não existe, como veremos a seguir.

Para criar uma função nova em R, designamos a um novo objeto o resultado da função chamada `function`, que tem como argumentos os mesmos argumentos da função que estamos criando, os quais serão manipulados pela nova função segundo uma expressão agrupada. Um exemplo tornará bem claro o procedimento. Suponha que queiramos criar uma nova função chamada `medpot` que calcule a média aritmética dos elementos de um vetor numérico elevados a uma potência inteira definida pelo usuário. Essa função tem dois argumentos: o vetor cujos elementos serão usados no cálculo e a potência à qual os elementos serão elevados. Veja como criaríamos essa função:

```
medpot <- function(v,n) { vpot <- v^n; mean(vpot) }
```

A partir do comando acima, o R reconhecerá o nome `medpot` como sendo o de uma função que demanda dois argumentos (chamados `v` e `n`) e que devolve a média dos elementos do primeiro argumento (`v`) elevados ao segundo argumento (`n`), como queríamos. Experimente digitando: `medpot(1:4,2)` e checando o resultado igual a 7,5. Um detalhe importante da avaliação das expressões que definem uma função é a impermanência dos objetos criados ali. Dentro da expressão agrupada que define a função `medpot` um objeto chamado `vpot` é criado. Esse objeto é temporário (ou "local") e o R não se lembrará dele depois da execução da função. Digite no console `vpot` para comprovar.

Quando um valor específico será usado com muita frequência em algum argumento da função que você está criando, você pode definir aquele valor como padrão para o argumento. Assim, o usuário não precisará digitá-lo quando quiser o valor padrão. Basta designar o valor desejado para o argumento, na chamada de `function`. Por exemplo, suponha que a função `medpot` deverá ser mais usada para calcular a média dos quadrados dos elementos do vetor. Você poderia definir a função assim: `medpot <- function(v,n=2) { vpot <- v^n; mean(vpot) }`. Veja o que acontece agora se você digitar apenas `medpot(1:4)`.

É claro que o resultado de uma função não precisa se restringir a um número. Por exemplo, abaixo criamos uma função chamada `trimhist` que constrói o histograma de um vetor fornecido pelo usuário, excluídos dele os números mais do que dois desvios padrões distantes da média:

```
trimhist <- function(x) { xtrim <- x[x > mean(x)-2*sd(x)]; xtrim <- xtrim[xtrim <
mean(x)+2*sd(x)] ; hist(xtrim, main="Observações centrais") }
```

A grande vantagem da criação de funções está no fato de que elas não são esquecidas pelo sistema, enquanto o usuário preservar a área de trabalho, gravando-a ao fim da sessão de trabalho. Quando você acionar o R em outra ocasião, as funções que você criou estarão disponíveis. Num certo sentido, é como se o R fosse capaz de adquirir um vocabulário novo, adaptado para as suas necessidades de trabalho. Você pode, inclusive, usar as funções que você criou antes dentro de expressões agrupadas quando você criar outras funções. As possibilidades são limitadas apenas pela sua imaginação.

Como você sabe, a instalação básica do R pode ser enriquecida com pacotes desenvolvidos por voluntários no mundo inteiro. Esses pacotes consistem essencialmente em bibliotecas de funções que compartilham algum tema em comum e que foram criadas, muitas delas, como você aprendeu aqui. Quem sabe um dia você não participará desse esforço conjunto e contribuirá com uma biblioteca de funções criadas por você?

**Lista de exercícios 5**

(1) Sem digitar no R, preveja qual será o resultado das seguintes expressões agrupadas:

(a) `{ rep(5:2,each=2) ; 2*3 ; rnorm(10) }`

(b) `{ sqrt(2) ; x^2 }`

(2) Quando uma função é tão simples que possa ser resolvida com apenas uma expressão, você não precisa usar chaves na sua definição. Redefina a função `medpot` sem usar chaves.

(3) Modifique a função `trimhist` de maneira que o usuário possa especificar o número de desvios padrões que determina o limite de inclusão dos elementos do vetor. Faça com que o padrão continue sendo 2 desvios padrões.

(4) Pesquise na Internet o como é calculado o coeficiente de assimetria de uma amostra aleatória. Crie uma função em R chamada `acoef` que calcule esse coeficiente de um vetor fornecido pelo usuário. Depois, use a função que você criou para comparar a assimetria de uma amostra aleatória de 1.000 números normais com uma amostra de mesmo tamanho de uma v.a. exponencialmente distribuída.

(5) Em Matemática, assim como no R, usamos todo o tempo operadores binários, tais como os operadores das quatro operações aritméticas básicas: $+$, $-$, $\times$ e $\div$. Estude, na seção 10.2 do manual *An Introduction to R*, como são criados novos operadores binários em R. Depois, crie um operador chamado `%doido%` que calcule a raiz quadrada do número à sua esquerda dividida pelo cubo do número à sua direita.

* * *