Programação Funcional em R: aplicações sobre dados imutáveis

Nelson Seixas dos Santos

Núcleo de Ciência de Dados e Computacional em Economia e Finanças
Faculdade de Ciências Econômicas
Universidade Federal do Rio Grande do Sul

1 de novembro de 2024

Sumário

Introdução

Problema

Como saber o valor das variáveis de um programa em cada instante da sua execução?

Contexto do problema

- A correção de um programa depende de o algoritmo de solução prover a solução correta para o problema especificado.
- A execução do algoritmo sob o paradigma procedural implica em mudanças constantes no valores de todas as variáveis de um programa (chamado de estado do programa) que dificulta o acompanhamento e avaliação da correção do algoritmo.
- Além disso, caso haja variáveis globais, a mudança em seus valores pode ser ocasionada por diferentes funções, dificultando ainda mais o trabalho de verificação da correção do programa.

Solução

- Escrever o algoritmo de modo que ele possa ser representado exclusivamente pela aplicação e composição de funções puras[1], eliminando todas as variáveis globais.
- Como consequência deste procedimento, o resultado da execução do programa depende apenas, em cada estágio da execução, dos valores entrada em cada função;

[1] Funções puras ou funções matemáticas cujos valores de saída dependem exclusivamente dos seus valores de entrada, isto é, dos seus parâmetros

Solução (cont.)

- A estrutura de controle de seleção é substituída por uma função normalmente denominada filter, mas que, no pacote básico do GNU R, é substituída pelas funções ifelse() e switch().
- A estrutura de controle de iteração é substituída por uma função normalmente chamada de map, mas que, no pacote básico do GNU R, é chamada de apply.

Solução (cont.)

- A execução do programa não consistirá da alteração de valores de variáveis, mas da aplicação de funções. Por isso, as variáveis poderão ser, no mais das vezes, imutáveis;
- Esta forma de compor os programas consiste no paradigma de Programação Funcional.
- A linguagem S e seu dialeto R foram desenhados com foco em propiciar o uso de programação funcional.

As referências básicas desta apresentação são [Wickham (2019)] e [Rodrigues (2020)].

Funções em R

Funções: definição

Uma função é um objeto que recebe um valor de entrada e fornece um valor de saída. Propriedades:

- Servem para armazenar trechos de código que serão reutilizados em diversas partes do programa.
- Facilitam a leitura do programa.
- Evita a repetição do código da mesma em diversas partes do programa.
- Implementa o princípio de programação denominado pela sigla em inglês DRY ("**D**on't **R**epeat **Y**ourself")

.

Escrevendo uma função

- Sintaxe de definição de funções

```
nome_da_função <- function(a,b){
          <comando 1>
          <comando 1>
          .
          .
          .
          <comando N>
          return(valor_calculado)
}
```

Escrevendo uma função em R: exemplo

Função que soma os valores da sua entrada

```
# Definição da função
soma <- function(a,b){
        return(a+b)
}

# Chamada da função
soma(2, 3)
```

Atribuição de Variáveis

A função assign

Em programação funcional, normalmente não se faz atribuição de variável, mas o GNU R possibilita a atribuição de modo funcional, usando a sintaxe a seguir:

```
assign('x', 3)
```

No exemplo acima, o valor 3 foi atribuído (associado) ao objeto x.

Funções de Estrutura de Seleção

Função ifelse: sintaxe

A sintaxe da função ifelse(b,v,f) é tal que:

- b é uma expressão booleana
- v e f são vetores.
- A função atribuirá o valor v, se b for verdadeira e o valor f se b for falsa. Esta função é bem semelhante ao if do MS-Excel

.

Função ifelse: exemplo

O uso da função

```
a <- c(5,7,2,9)
ifelse(a %% 2 == 0,"even","odd")
```

Função switch

A sintaxe da função switch(expressão, caso 1, caso 2, ..., caso N).

Veja o exemplo de uso da função switch()

Exemplo: uso da função switch para decisão

```
# Entrada
print("Digite um número de 1 a 5")
x <- scan(what=integer(), nmax = 1)

# Processamento
resultado <- switch(x, "1o", "2o", "3o", "4o", "5o")

# Saída
print(paste("Você ficou em", resultado, "lugar"))
```

Funções de Estrutura de Repetição

Função apply

Retorna um vetor, um array ou uma lista de valores a partir da aplicação de uma função às margens de um array ou de uma matriz.

A sintaxe da função é apply(x, margem, função), onde x é um array, margem é linha ou coluna e função é a função que se deseja que seja aplicada na linha ou coluna do array.

Margem pode ser 1 para coluna, 2 para linha ou o vetor c(1,2) para linhas e colunas.
A referência é link

Exemplo: uso do apply

Eleve ao quadrado todos os termos da matriz de números de 1 a 30,.

```
z <- array(1:30, dim=c(5,6))
print(z)
quadrado <- function(matriz){
matriz**2
}
y <- apply(z, c(1,2), quadrado)
print(y)
```

Função lapply

Retorna uma lista de mesmo tamanho de um dado vetor ou lista, onde uma função dada é aplicada elemento a elemento.

A sintaxe da função é lapply(x, função), onde x é um vetor ou lista e função é a função que se deseja que seja aplicada a x.
A referência é link

Exemplo: uso do lapply

Eleve ao quadrado todos os termos do vetor de números de 1 a 30,

```
z <- 1:30
print(z)
quadrado <- function(vetor){
vetor**2
}
y <- lapply(z, quadrado)
print(y)
```

Função sapply

Esta função é igual à função lapply, mas simplifica o resultado para ele ser do mesmo tipo do dado de entrada. Assim, se a entrada x for um vetor sapply retornará um vetor.

Exercício

Experimente usar por si só no exemplo anterior, substituindo lapply por sapply e compare os resultados.

Considerações finais

Considerações finais

A linguagem S e seu dialeto GNU R são essencialmente funcionais. Por isso, escrever código funcional é mais natural nesta linguagem e não raro apresenta um desempenho maior.

Apresentou-se uma introdução à programação funcional em R onde se propós a substituição das estruturas de controle de seleção(decisão) e iteração(repetição) pelas respectivas funções de R.

O desenvolvimento posterior no uso do R é escrever apenas funções puras para facilitar o uso de processamento paralelo se for necessário reduzir em muito o tempo de execução do código R.

Mais isso fica para uma próxima oportunidade...

Referências

Referências


RODRIGUES, Bruno. *Modern R with the tidyverse*, Chapter 8. LeanPub: 2020. Disponível em link.


WICKHAM, Hadley. *Advanced R, 2nd. Edition*. Boca Raton, FL: CRC Press, 2019. Disponível em link.