

Classes S3 em R

Programação orientada a objetos simplificada para cientistas de dados



Eduardo Ogasawara

eduardo.ogasawara@cefet-rj.br


<https://eic.cefet-rj.br/~eogasawara>

Introdução às Classes S3

O sistema S3 é a forma mais popular e acessível de criar classes em R. Sua simplicidade o torna ideal para cientistas de dados que estão começando com programação orientada a objetos.

Características principais:

- Implementação simples e intuitiva
- Base para a maioria das classes nativas do R
- Flexível e fácil de estender

 **Conceito-chave:** Uma classe S3 é essencialmente uma lista com um atributo especial de classe

```
# Cria objeto da classe "polygon"
obj <- list(n = n)
```

```
# Define a classe
attr(obj, "class") <- "polygon"
```

Criando o Construtor

O construtor é uma função fundamental que cria e inicializa objetos da sua classe. Ele garante que os objetos sejam criados de forma consistente e com validação adequada.

Validação de Entrada

Sempre verifique os parâmetros antes de criar o objeto para evitar estados inválidos

Criação do Objeto

Construa a lista com os atributos necessários para representar a classe

Definição da Classe

Marque o objeto com o nome da classe usando `attr()` ou `class()`

```
polygon <- function(n) {  
  if(n <= 0) stop("número de vértices deve ser maior que zero")  
  obj <- list(n = n)  
  attr(obj, "class") <- "polygon"  
  return(obj)  
}
```

Implementando Herança

A herança permite criar classes especializadas que reutilizam e estendem funcionalidades de classes existentes. No sistema S3, isso é feito através da adição de múltiplas classes ao vetor de classes.

Vantagens da herança:

- Reutilização de código
- Hierarquia lógica de classes
- Especialização de comportamento

```
rectangle <- function(w, h) {  
  obj <- polygon(4)  
  obj$w <- w  
  obj$h <- h  
  class(obj) <- append("rectangle", class(obj))  
  return(obj)  
}
```

Note que `class(obj)` retorna um vetor: primeiro "rectangle", depois "polygon"

INTERFACE EXISTENTE

Implementando Interfaces



Identifique a Interface

Escolha uma função genérica existente como `print()`



Crie o Método

Implemente usando a convenção `nome.classe`



Teste o Comportamento

Verifique se o método é chamado corretamente

Ao implementar métodos para funções genéricas existentes, você garante que seus objetos se comportem de forma consistente com o ecossistema R.

```
print.polygon <- function(obj) {  
  cat(obj$n, "\n")  
}  
  
print.rectangle <- function(obj) {  
  cat(obj$w, ",", obj$h, "\n")  
}
```

Exemplo Prático: Interface print()

Método para Polygon

Exibe o número de vértices de forma simples e direta

```
print.polygon <- function(obj) {  
  cat(obj$n, "\n")  
}
```

Método para Rectangle

Mostra as dimensões específicas: largura e altura

```
print.rectangle <- function(obj) {  
  cat(obj$w, ",", obj$h, "\n")  
}
```



Polimorfismo em ação: A mesma função `print()` se comporta diferentemente dependendo da classe do objeto

Criando Suas Próprias Interfaces

Criar funções genéricas permite definir comportamentos que podem ser especializados por diferentes classes.

01

Defina a Interface Genérica

Use `UseMethod()` para criar a função genérica

02

Implemente o Método Padrão

Crie `nome.default` para comportamento geral

03

Especialize por Classe

Implemente `nome.classe` para comportamentos específicos

```
area <- function(obj) {  
  UseMethod("area")  
}  
  
area.default <- function(obj) {  
  return(0)  
}  
  
area.rectangle <- function(obj) {  
  return(obj$w * obj$h)  
}
```

Anatomia de uma Interface Customizada



Função Genérica

`UseMethod("area")` delega a chamada para o método apropriado baseado na classe do objeto



Método Padrão

`area.default()` retorna 0 para objetos que não têm implementação específica de área



Método Específico

`area.rectangle()` calcula a área multiplicando largura por altura

```
area <- function(obj) { UseMethod("area") }  
area.default <- function(obj) { return(0) }  
area.rectangle <- function(obj) { return(obj$w * obj$h) }
```


Encontrando Interfaces Publicadas

Antes de criar uma nova interface, verifique se ela já existe. Isso evita duplicação e garante compatibilidade com o ecossistema R.

Benefícios de usar interfaces existentes:

- Consistência com pacotes padrão
- Menos código para manter
- Melhor interoperabilidade

```
methods(class="default")
```

Este comando lista todos os métodos disponíveis para uma classe específica, ajudando você a descobrir quais interfaces já estão implementadas.

Explorando as Classes na Prática

Vamos ver como os objetos se comportam quando usamos as classes e métodos que criamos:

Criação dos Objetos

```
a <- 3  
p <- polygon(5)  
r <- rectangle(3, 10)
```

1

2

Usando print()

```
print(a) # [1] 3  
print(p) # 5  
print(r) # 3 , 10
```

Calculando Áreas

```
print(area(a)) # [1] 0  
print(area(p)) # [1] 0  
print(area(r)) # [1] 30
```

3



Observe: Cada classe se comporta de forma diferente, mas a interface permanece consistente e intuitiva

Hora de Praticar!



1

Explore o Exemplo

Acesse o repositório GitHub e execute o código completo para entender cada componente

github.com/eogasawara/R/tree/main/14-S3-Classes

2

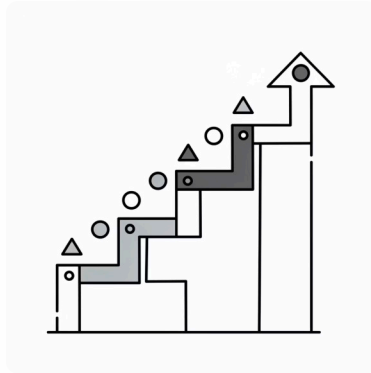
Exercício Proposto

Desafio: Implemente as classes `square` (quadrado) e `hexagon` (hexágono) e calcule suas áreas

- `Square`: herde de `rectangle`
- `Hexagon`: crie a partir de `polygon`
- Implemente os métodos `area()`

Dica: Área do hexágono regular = $(3\sqrt{3}/2) \times \text{lado}^2$

O que vem a seguir: Fundamentos da Linguagem R



Você já domina:

- Variáveis e tipos de dados
- Estruturas de controle de fluxo
- Funções e programação modular
- Estruturas de dados complexas

Leitura de Dados Reais

Importação e manipulação de arquivos CSV, Excel e bancos de dados

Pipelines de Dados

Construção de fluxos de transformação com dplyr e tidyverse

Análise Estatística

Testes, modelagem e visualização avançada de dados

Integração com Ecossistemas

Conexão com Python, SQL e ferramentas de ciência de dados

O próximo curso aprofunda seus conhecimentos e prepara você para trabalhar com dados reais em projetos profissionais e acadêmicos.

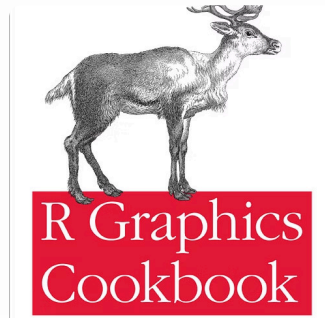
Referências



Hands-on Programming

Aprenda R criando suas próprias funções e simulações

<https://rstudio-education.github.io/hopr/index.html>



R Graphics Cookbook

Domine visualizações de dados em R

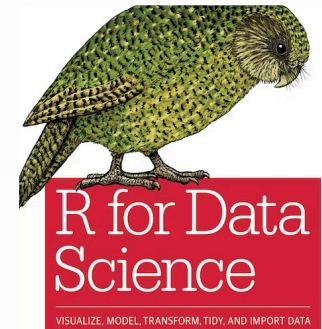
<https://r-graphics.org>



R Packages

Desenvolva seus próprios pacotes R

<https://r-pkgs.org/index.html>



R for Data Science

Guia completo para ciência de dados

<https://r4ds.had.co.nz>