Objekty

Michal Kvasnička

R je ve své podstatě objektově orientovaný jazyk. Na rozdíl od jiných jazyků však objekty nejsou vlastní součástí jazyka, ale jsou vytvořeny v rámci jazyka. V důsledku toho (a dlouhého vývoje R a jeho předchůdce, jazyka S) dnes v R existuje několik různých systémů objektů, z nichž nejvýznamnější jsou systémy S3 a S4. V této lekci se podíváme na nejdůležitější základy systému S3, který je nejpoužívanější a nejjednodušší. Další systémy jsou popsané v AdvR.

Základní pojmy (pro laiky)

Každý objekt má určitou třídu. Třída (class) popisuje strukturu dat objektu (v terminologii OOP atributy), vztah k ostatním třídám objektů (v rámci dědičnosti) a to, jaké funkce (metody) se volají při práci s objektem.

Např. objekt třídy "člověk" může obsahovat atributy jako jméno, příjmení, výšku, váhu apod. Pro tuto třídu mohou být definovány nějaké funkce, např. funkce, která spočítá BMI. Objekt třídy "manažer" je logicky svázán s objektem třídy "člověk" – obsahuje stejná data (může obsahovat i další, např. seznam podřízených). Stejně tak může využívat funkce se stejnými jmény (které však mohou být implementovány odlišně).

Logickou vazbu tříd "člověk" a "manažer" popisuje dědičnost: třída "manažer" je potomek třídy "člověk". Dědičnost vytváří hierarchii: potomci dědí vlastnosti a chování předků. Pokud potomek nemá implementovanou nějakou funkci, pak se použije funkce jeho předka.

Systém S3

Systém S3 používá poměrně neobvyklý přístup k objektovému programování, tzv. "generic function OOP". Tento přístup je jednoduchý (až primitivní), ale funguje velmi dobře a používá jej drtivá většina balíků v R.

V normálních OO jazycích patří objektu nebo třídě jak data, tak metody (funkce). V R však objektu patří jen data a jméno třídy; metody patří tzv. generické funkci. Nejjednodušší způsob, jak pochopit strukturu objektu, je nějaký objekt vytvořit.

Vytvoření třídy a objektu

S3 nemá formální definici tříd proměnných. Objekt se vytvoří tak, že se vytvoří nějaký základní typ objektu (většinou seznam) a nastaví se mu atribut `class` na hodnotu jména třídy:

```
foo <- list()         # vytvoří prázdný seznam
class(foo) <- "foo"   # přiřadí mu třídu "foo"
```

Alternativně to jde provést naráz pomocí funkce `structure()`, která nastavuje danému objektu atributy:

```
foo <- structure(list(), class = "foo")
```

Nyní je proměnná `foo` objekt třídy "foo":

```
class(foo)
```

```
## [1] "foo"
```

R nemá žádný mechanismus, který by kontroloval, zda je datová struktura objektu správná. Již vytvořenému objektu je možné přidat nebo ubrat datové sloty (technicky obvykle prvky seznamu) nebo změnit jeho třídu. Někdy se to hodí; nikdy to však nedělejte, pokud opravdu dobře nevíte, co děláte, jinak vznikne velmi těžko předvídatelné chování a těžko dohledatelné chyby.

Aby se zajistilo, že je struktura objektu správná, je vhodné vytvořit konstruktor – funkci, která vytváří objekt:

```
human <- function(name, height, weight)
    structure(list(name = name, height = height, weight = weight),
              class = "human")
adam <- human("Adam", 173, 63)
```

(Funkce jako `numeric()` jsou také konstruktory.)

Dědičnost se zajistí tak, že atribut `class` je vektorem jmen několik tříd – zleva doprava od potomků k předkům:

```
manager <- function(name, height, weight, rank)
    structure(list(name = name, height = height, weight = weight, rank = rank),
              class = c("manager", "human"))
eva <- manager("Eve", 169, 52, "CEO")
```

Eva nyní dědí vlastnosti člověka:

```
class(eva)
```

```
## [1] "manager" "human"
```

```
inherits(eva, "manager")
```

```
## [1] TRUE
```

```
inherits(eva, "human")
```

```
## [1] TRUE
```

Protože je většina objektů v S3 postavená pomocí seznamů, můžete získat hodnotu jednotlivých slotů objektů pomocí `$`:

```
str(eva)
```

```
## List of 4
##  $ name  : chr "Eve"
##  $ height: num 169
##  $ weight: num 52
##  $ rank  : chr "CEO"
##  - attr(*, "class")= chr [1:2] "manager" "human"
```

```
eva$rank
```

```
## [1] "CEO"
```

Nová metoda

Hlavní pointa systému S3 spočívá v tom, že stejně pojmenovaná funkce volá pro každý typ objektu jiný kód – přesně uzpůsobený danému typu dat. Funkcím, které to dokážou, se říká generické funkce. Příkladem generické funkce je funkce `print()`, která tiskne různé informace v závislosti na tom, jaká je třída objektu, na který je zavolaná.

Třída "human" zatím nemá definovanou žádnou metodu pro generickou funkci `print()`, proto zavolá metodu pro seznam:

```
print(adam)
```

```
## $name
## [1] "Adam"
##
## $height
## [1] 173
##
## $weight
## [1] 63
##
## attr(,"class")
## [1] "human"
```

Když vytvoříte novou třídu objektu, můžete vytvořit novou metodu ke generické funkci tak, že vytvoříte funkci, jejíž jméno je `jmeno_genericke_funkce.jmeno_tridy`:

```
print.human <- function(x)
    cat("*** Human ***",
        paste("Name:", x$name),
        paste("Height:", x$height),
        paste("Weight:", x$weight),
        sep = "\n")
print(adam)
```

```
## *** Human ***
## Name: Adam
## Height: 173
## Weight: 63
```

Protože třída "manager" je potomkem třídy "human", dědí její chování. To znamená, že pokud tato třída nemá definovanou svou metodu `print()`, pak zavolá metodu svého předka, třídy "human":

```
eva  # implicitně se volá funkce print
```

```
## *** Human ***
## Name: Eve
## Height: 169
## Weight: 52
```

Nová generická funkce

Novou generickou funkci jde vytvořit takto:

```
bmi <- function(x)          # generická funkce
    UseMethod("bmi")
bmi.human <- function(x)    # metoda pro třídu "human"
    x$weight / (x$height / 100) ^ 2
bmi.manager <- function(x)  # metoda pro třídu "manager"
    "classified"
```

Nyní můžeme zjistit, jaký mají Adam a Eva BMI:

```
bmi(adam)
```

```
## [1] 21.04982
```

```
bmi(eva)
```

```
## [1] "classified"
```

(Bohužel je informace o BMI manažerů tajná.)

Generická funkce může mít i implicitní metodu, která se použije v případě, že pro daný typ není žádná metoda k dispozici. Tato metoda se jmenuje `default`:

```
bmi.default <- function(x)
    "Unknown class"
bmi(1)
```

```
## [1] "Unknown class"
```

(Bez definování implicitní metody by předchozí řádek skončil chybou.)

Práce s objekty

Ve většině případů nebudete vytvářet vlastní objekty – stačí, když budete schopní zacházet s objekty, které vrátí funkce, které budete používat.

Jak poznat objekt S3

Poznat, že je něco objekt v systému S3 není úplně snadné. Jsou tři možnosti:

```
is.object(eva) & !isS4(eva)
```

```
## [1] TRUE
```

```
pryr::otype(eva)  # funkce z balíku pryr
```

```
## [1] "S3"
```

Nebo se samozřejmě podíváte do dokumentace funkce, kterou používáte.

Struktura objektu

Většina objektů systému S3 je postavená nad seznamy, takže jejich strukturu zjistíte funkcí `str()` a složky získáte pomocí `$`:

```
str(eva)
```

```
## List of 4
##  $ name  : chr "Eve"
##  $ height: num 169
##  $ weight: num 52
##  $ rank  : chr "CEO"
##  - attr(*, "class")= chr [1:2] "manager" "human"
```

```
eva$height
```

```
## [1] 169
```

Které metody jsou k dispozici

Metody, které patří dané generické funkci vrátí funkce `methods()`:

```
head(methods("print"))  # jen prvních 6 metod (zajistí funkce head())
```

```
## [1] "print.acf"     "print.AES"     "print.anova"   "print.aov"
## [5] "print.aovlist" "print.ar"
```

Metody, které jsou k dispozici pro nějakou třídu:

```
methods(class = "human")
```

```
## [1] bmi   print
## see '?methods' for accessing help and source code
```

Jména konkrétních metod lze zjistit také pomocí funkce `apropos()`, když se požádá, aby hledala regulární výraz pro cokoli, co končí ".třída_objektu":

```
apropos(".*\\.human")
```

```
## [1] "bmi.human"   "print.human"
```

Nápověda k metodám

Když hledáte nápovědu ke generické funkci, nemusíte se dozvědět vše, co potřebujete. Pokud chcete nápovědu k tomu, jak se generická funkce chová pro daný objekt, hledejte dokumentaci k jeho metodě.

Později se budeme bavit o tom, jak dělat ekonometrii v R. Výsledkem odhadu lineárního modelu je objekt třídy "lm". Pro tento objekt existuje mnoho generických funkcí jako je `print()`, `summary()` a `plot()`. Pokud tento objekt vytisknete pomocí funkce `plot()`, vykreslí několik diagnostických grafů. Metoda má mnoho parametrů, které však nezjistíte z dokumentace generické funkce `plot()`, nýbrž z dokumentace metody `plot.lm()`.

Více informací

Detailnější informace o objektových systémech S3, S4 a ReferenceClass najdete v AdvR, kap. 7.