NĚKTERÉ ASPEKTY TVORBY KNIHOVEN V JAZYCE R

Ladislav Beránek
Katedra informatiky, PF, JCU – Jihočeská Universita, České Budějovice, Jeronýmova 10,
371 15 České Budějovice
beranek@pf.jcu.cz

ABSTRAKT:
Jazyk R je vývojové prostředí pro statistické zpracování dat a pro jejich grafické znázorňování. Je volně dostupný pod licencí GNU a je rozšířen zejména v komunitě pracovníků zabývajících se statistikou a v akademické sféře. V příspěvku jsou ukázány některé postupy při psaní knihoven jazyka R se zaměřením na použití volání funkcí napsaných v C v prostředí operačního systému Windows. Příspěvek vychází z vlastních zkušeností s tímto jazykem při tvorbě knihovny pro analýzu dat z hmotnostních spektrometrů a klade si za cíl ukázat některé aspekty tohoto specifického jazyka.

KLÍČOVÁ SLOVA:
jazyk R, zpracování dat, externí funkce

1. Úvod

Tento příspěvek je orientován prakticky. Seznamuje se systémem, který je rozšířený v akademickém prostředí a v komunitě statistiků, a pomocí jednoduchých příkladů ukazuje základní možnosti využití funkcí naprogramovaných v C v prostředí R. Tento postup slouží ke zrychlení výpočtů a umožňuje využití knihoven napsaných v jiných programech (C, Fortran) tak, jak je to obvyklé například v prostředí Matlab nebo jeho GNU verzi Octave.

2. Jazyk R

Jazyk R je systém nebo prostředí určené pro analýzu dat. Umožňuje provádět statistické výpočty a následnou grafickou prezentaci výsledků. Inspirací pro jeho vznik byl programovací jazyk S [1], který by1 vyvíjen u firmy AT&T v Bell Laboratories. Systém R vznikl na University of Auckland, dnes se na jeho vývoji podílí vývojový tým lidí z celého světa. Je volně šiřitelný pod GNU licencí.
Součástí systému R je programovací jazyk, prostředí a překladač. Také je umožněn přístup k funkcím systému a je zde možnost spouštění programů uložených v souboru. V programovacím jazyku lze použít rekurzi a nechybí ani podpora objektově orientovaného programování. Samozřejmostí jsou podmínky, cykly a vstupní a výstupní operace.
Systém R používá statický paměťový model. To znamená, že při svém startu požádá operační systém o paměť určité velikosti a za běhu ji už nemůže zmenšovat ani zvětšovat. V této paměti si pak zajišťuje vlastní správu. Při startu programu do této paměti umisťuje funkce a data načtená ze souboru operačního systému. Paměť je díky vlastní správě používána velmi účelně. Funkce a data, které jsou součástí systému nebo vznikají při jeho používání, nazýváme objekty.Všechny objekty, i nově vznikající, jsou během práce se systémem uchovávány v paměti a jsou ukládány do záložních souborů operačního systému, kde jsou připraveny pro budoucí použití. Mnoho funkcí je součástí základního systému. Další objekty nebo objekty související s určitou oblastí bývají někdy umístěny v externích knihovnách nazvaných package, které se připojují podle potřeby. Součástí systému R je jedenáct knihoven a mnoho dalších lze získat a dodatečně připojit.

Aktuálně je systém R vyvíjen pro operační systémy UNIX, MS Windows a MacOS. Za zmínku stojí, že mnoho funkcí je napsáno opět v jazyku systému R. Z důvodů rychlého vývoje a vzhledem k maximálnímu využití nových poznatků a metod není zaručena zpětná kompatibilita se staršími verzemi.
Prostředí obsahuje vlastní interpret jazyka R, ve kterém lze připravit jak dávkové soubory, tak definovat nové funkce. Tyto funkce mohou být interpretovány buď přímo z textové podoby souborů nazývaných R-soubory nebo z předzpracované podoby pomocí knihoven, tzv. balíčků (packages). Jazyk R je tedy integrovaná sada programových příslušenství pro analýzu dat a grafické znázorňování. Mezi jinými nabízí:

- rozsáhlý soubor nástrojů pro statistické zpracování dat a pro grafiku,
- jazyk sloužící pro vyjádření statistických modelů a jako nástroj pro lineární a nelineární statistické modely,
- grafické prostředky pro analýzu dat na obrazovce nebo pro tisk,
- je volně dostupný pod licencí GNU na www stránkách [1] včetně balíčků.

Důležitou součástí jazyka R jsou dnes knihovny (balíčky), které obsahují vždy určitý obor statistických metod nebo nástrojů numerické matematiky zpracovaný uceleným způsobem včetně dokumentace a příkladů.

3. Nástroje pro kompilaci funkcí napsaných v C

V rámci projektu vytvoření algoritmů pro zpracování hmotnostních spekter jsme použili jazyk R. Je k dispozici zdarma (na rozdíl od Matlabu, který by byl také vhodný), je kolem něho velká komunita jak vývojářská tak i uživatelská, a dále obsahuje spoustu knihoven (balíčků). Naším cílem je vytvořit knihovnu (balíček) algoritmů, které by byly použitelné pro identifikaci biomarkerů ze spekter naměřených pomocí hmotových spektrometrů. Termínem biomarker označujeme protein (nebo jeho část), jehož množství se statisticky mění s intenzitou nebo stupněm studovaného biologického procesu, např. nemoci. Ve spektrech se biomarker projeví píkem (nebo množinou píků). Vzhledem k variabilitě biologických procesů není jednoznačná identifikace biomarkerů z hmotových spekter jednoduchou úlohou.
Některé funkce, které jsme napsali v R, však byly pomalé a tak jsme se pokusili napsat je v jazyce C jako externí funkce. Tyto externí funkce jsou potom volány z prostředí R. Výpočty se tímto způsobem zrychlily. U maticových operací jsme však tento způsob nepoužili (ani netestovali). Spolehli jsme se na tvrzení vývojového týmu R [1], podle kterého mohou být maticové operace v R rychlejší než stejné operace přepsané do C, protože R používá optimalizovanou verzi BLAS pro provádění maticových operací [7].
Při práci v prostředí LINUXu je použití procedur napsaných v jazyce C a volaných z prostředí R jednoduché. V prostředí LINUXu, který implicitně obsahuje potřebné nástroje, stačí napsat funkce v C a zkompilovat je použitím příkazu:

R CMD SHLIB *jméno_funkce.c*

Tento příkaz, který používá standardní kompilátory pro C, PERL a různé utility UNIXu, vytvoří soubor nazvaný *jméno_funkce.so*, který může být dynamicky volán z R (dříve či později) použitím funkce dyn.load.
Ve Windows je to však komplikovanější. Postup není přesně nikde popsán a tak bylo nutné prozkoumat podrobně postup v LINUXu a aplikovat ho ve Windows. Zde je nutné použít různé kompilátory a nástroje potřebné pro vytvoření dynamicky volané knihovny (DLL).
Jedná se o:

- množinu nástrojů, které jsou původně utilitami určenými pro UNIX,
- PERL verze pro Windows,

- GNU kompilátor pro Windows, např. MinGW [4] nebo Cygwin [5].

MinGW zahrnuje kompilátory GNU C, C++ a Fortran77. Protože R je kompilován za použití GNU C kompilátoru, budou soubory kompilované tímto kompilátorem kompatibilní. Navíc GNU kompilátor použitý v prostředí Windows má dobrou reputaci, uvádí se, že výsledkem jeho použití je kód, který je rychlejší než kompilátor Microsoft [6]. Po instalaci Perl a MinGW je třeba udělat úpravy v nastavení prostředí. Ve vlastnostech systému je třeba doplnit systémovou proměnnou path o:

C:\rtools; c:\mingw\bin; c\Perl\bin; c:\R\R-2.2.1\bin;

4. Psaní programu v C pro prostředí R

Jestliže chceme používat funkce napsané v jazyce C pro prostředí R, musí mít tyto funkce několik důležitých vlastností:

- C funkce volané R musí všechny vracet void. To znamená, že musí vracet výsledek výpočtu v jejich argumentech,
- všechny argumenty do C funkce předávány jako odkazy, což znamená, že se předává ukazatel na skalár nebo pole. Je proto nutné při programování funkcí v C dát si na tuto skutečnost pozor, může to být zdroj chyb,
- Každý soubor obsahující C kód, který má být volán v R musí zahrnovat záhlaví R.h. Na začátku souboru by tak měl být řádek #include<R.h>.

Jestliže chceme použít speciální funkce (např. distribuční funkce), pak musí záhlaví obsahovat řádek *Rmath.h.*

Funkci napsanou v C zkompilujeme z prostředí R. Je to mnohem vhodnější, než volat kompilátor C přímo. Systém R po této kompilaci bude vědět, kde jsou umístěny hlavičkové soubory a knihovny. Pokud tedy máme napsaný zdrojový soubor (knihovnu potřebných funkcí) v C, např. soubor jménem *baseline.c*, zkompilujeme ho z příkazové řádky pomocí příkazu:

R CMD SHLIB *baseline.c*

Tímto příkazem se vytvoří DLL se jménem *baseline*, který může být volán z prostředí R. Jestliže chceme, aby se naše knihovna jmenovala jinak (například *nova_knihovna*), použijeme příkaz:

R CMD SHLIB -o *nova_knihovna baseline.c*

Zkompilovanou knihovnu můžeme volat z prostředí R pomocí funkce dyn.load:

> dyn.load("*baseline.dll*")

Poté už máme v prostředí R zpřístupněny všechny funkce napsané v souboru baseline.c.

4.1 Příklad programu

Abychom uvedli jednoduchý příklad, uvedeme funkci, která zapíše na obrazovku „Hello, world". Přestože je tento příklad velmi jednoduchý, ukazuje podstatu výše uvedených principů.

V R můžeme tento program napsat velmi jednoduše. Nazveme ho *hello1*:

```
hello1 = function(n) {
            for (i in 1:n) {
            cat("Hello, world!\n")
        }
}
```

Nyní můžeme zavolat v prostředí R funkci *hello1*:

```
> hello1(5)
Hello, world!
Hello, world!
Hello, world!
Hello, world!
Hello, world!
>
```

Nyní napíšeme program stejný program v C a provedeme příkazy, abychom ho mohli volat v R. Nejprve vytvoříme soubor napsaný C, který bude provádět potřebné příkazy výpisu daného textu na obrazovku. Tento zdrojový soubor nazveme *hello.c*. Soubor *hello.c* může vypadat takto:

```
#include <R.h>

void hello(int *n)
{
int i;
 for ( i = 0; i < *n; i++) {
          Rprintf("Hello, world!\n");
    }
  }
```

Je vidět, že soubor neobsahuje funkci main(). Je to proto, že nepíšeme celý program, ale pouze funkci v C. Funkce Rprintf stejná, jako je printf v C s tím rozdílem, že Rprintf posílá svůj výstup do prostředí R, takže můžeme vidět výstup funkce hello.c, když máme spuštěno prostředí R.
Odpovídající příkazy v R, aby bylo možné volat soubor *hello.c,* musíme zapsat následujícím způsobem:

```
hello2 = function(n) {
.C("hello",as.integer(n))
}
```

Funkce **.**C vytváří interface pro funkce napsané v C a zkompilované dále uvedeným způsobem z příkazové řádky a prostředím R. První argument v **.**C je řetězec obsahující jméno funkce v C, která se má volat. Další argumenty jsou argumenty pro volání této funkce. V uvedeném příkladě má funkce napsaná v C má jen jeden argument, počet tisku řetězce „Hello world". Je to proměnná n a je označena jako typ integer. Typy a pořadí proměnných si musí odpovídat.
Před použitím v prostředí R musíme soubor s funkcí napsanou v C ještě zkompilovat. Soubor *hello.c* zkompilujeme příkazem z příkazové řádky:

R CMD SHLIB hello.c

Toto vytvoří DLL s názvem *hello.dll*. Nyní v prostředí R napíšeme:

>dyn.load("hello.dll")

Poté již můžeme volat funkci *hello2:*

>hello2(5)

Na konzoli se objeví:
Hello, world!
Hello, world!
Hello, world!
Hello, world!
Hello, world!
[[1]]
[1] 5

Hodnoty v hranatých závorkách je hodnota zadávaná ve funkci **.**C funkci. Ta také vrací seznam všech argumentů, které jsou na vstupu do **.**C funkce.

Další příklad ukazuje volání funkce napsané v C ze systému R pomocí funkce **.**C, kdy jsou použity různé typy vektorů:

Program useC2 napsaný v C:

```
#include <R.h>
/* useC2.c      */
void useC(int *i, double *d, char **c, int *l) {

i[0] = 11;
d[0] = 2.33;
c[1] = "g";
l[0] = 0;

}
```

Opět provedeme kompilaci z příkazové řádky:

R CMD SHLIB useC2.c

V prostředí R nyní provedeme:
Nejprve zapíšeme vstupní vektory:

```
> i = 1:10                                  #vektor celých čísel
> d = seq(length = 3, from = 1, to = 2)     # vektor reálných čísel
> c = c("a", "b", "c")                      # vector znaků
> l = c("TRUE","FALSE")                     #vektor logických hodnot
```

Jejich výpis vypadá takto:

```
> i
[1]  1  2  3  4  5  6  7  8  9  10

> d
[1]  1.0  1.5  2.0

> c
[1] "a"  "b"  "c"

>l
[1] "TRUE"  "FALSE"
```

Nyní zavoláme funkci useC2

```
> dyn.load(useC2.dll")
> out = .C("useC", i1 = as.integer(a), d1 = as.numeric(d), c1 = as.character(c), l1 = as.logical(l))
```

Nyní vypíšeme v prostředí R proměnnou out:

```
> out
$d1
        [1] 11  2  3  4  5  6  7  8  9  10

$d1
        [1]  2.33  1.50. 2.00

$c1
        [1] "g"  "b"  "c"

$l1
        [1] "FALSE"  "FALSE"
```

5. Závěr

V příspěvku je popsána možnost tvorby externích knihoven se zaměřením na volání funkcí napsaných v jazyce C ze systému R pomocí funkce **.**C. Systém R obsahuje i další funkce, které zajišťují možnost volání funkcí napsaných v C ze systému R. Je to funkce **.**Call a **.**External. Jejich použití je obdobné funkci **.**C, poskytují však větší možnosti. Pro demonstraci je však ukázka funkce **.**C přehlednější.

Volání procedur napsaných v jazyce C v prostředí R je vhodné z důvodů:

- rychlosti,
- efektivní správy paměti,
- použití existujících knihoven jazyka C.

Při užití funkcí napsaných v C pro použití v prostředí R je však třeba vzít v úvahu:

- nutnost použití vnitřních funkcí jazyka C určených pro použití v prostředí R (kompletní seznam těchto funkcí je uveden v [2],
- nutnost použití ukazatelů jak pro skaláry, tak i pro vektory.

Systém (nebo také jazyk) R je velmi specifický a jeho použití je především v oblasti statistických výpočtů a v oblasti vývoje statistických numerických a grafických metod. Jazyk R je, pokud umíme využít všech jeho vlastností (a jedna z nich je v příspěvku demonstrována), velmi silný nástroj, se kterým lze mnohdy velmi náročné úkoly vyřešit za zlomek času, než by to trvalo s využitím klasického komerčního softwaru.

LITERATURA

[1] http://www.r-project.org/.
[2] Writing R Extensions, R Development Core Team, http://www.r-project.org/.
[3] The R language definition, R Development Core Team, http://www.r-project.org/.
[4] http://www.mingw.org/
[5] http://www.cygwin.com/
[6] http://www.willus.com/ccomp_benchmark.shtml
[7] http://math-atlas.sourceforge.net/