Tvorba modelů v jazyce S Plus

Jiří MILITKÝ ,Katedra textilních materiálů, Technická universita, 461 17 Liberec, Česká Republika

1 ÚVOD

Moderní metody v řadě vědních oborů využívají koncepce tvorby modelů s využitím matematicko statistických metod. To klade speciální požadavky na software, které by mělo umožnit jednoduše definovat různé typy modelů a usnadnit jejich analýzu. Většina známých programových balíků těmto požadavkům vyhovuje jen pro nejjednodušší typy modelů. Pro případy, kdy je třeba pracovat s modely interaktivním způsobem zatím vhodné software představují různé statistické jazyky, z nichž vyniká jazyk S, jehož rozšířením je objektivně orientovaný statistický systém S Plus. Účelem tohoto sdělení je ukázat některé možnosti systému S Plus, zejména v oblasti analýzy dat a statistického modelování. Pro podrobnější informace o vlastním jazyce lze využít referenční příručku [2]. V knize [3] jsou uvedeny některé speciální modelovací techniky obsažené v systému S Plus. Přehled obecných, klasických i moderních technik analýzy dat a modelování je uveden v knize [1]. Pro první seznámení se systémem S Plus lze využít také práce [4].

2 VÝVOJ SYSTÉMU S PLUS

S Plus se dá označit jako interaktivní programovatelné prostředí pro analýzu dat, grafiku, statistické modelování a další výpočty. Představuje rozšířenou, upravenou a vylepšenou versi statistického jazyka S, který byl vyvinut v Bellových laboratořích Beckerem, Chambersem a Wilksem. První verze jazyka S byla vytvořena v roce. 1984. V roce 1988 byl tento jazyk zcela přepracován a nazván "New S" [2]. K další výrazné změně došlo v r. 1991, kdy byl podroben výrazné revizi a znovu označen jako S. Ve všech těchto variantách se dodával včetně zdrojových programů vytvořených zejména v programovacím jazyce C a částečně FORTRAN. I když měly tyto verze všechny funkce, chybělo zejména okolí a navíc nebylo možno použít operační systém DOS (standardně je tento software budován pod operačním systémem UNIX).

K revolučním změnám došlo, když tento jazyk převzala k dalšímu vývoji firma Statistical Sciences Inc. ze Seattlu. Tato firma provedla řadu vylepšení týkajících se zejména okolí a systému nápovědy a nazvala výsledný produkt S Plus. Toto programovatelné okolí (pro DOS verze 2.*) se již dodává v kompilované formě bez zdrojových kódů. Obsahuje však řadu funkcí psaných přímo v jazyce S, které lze číst, upravovat a zahrnovat do S Plus. Jde tedy o představitele moderních jazyků, které jsou psány sami v sobě. Verze S Plus z r. 1992 obsahuje kolem 140 000 řádek programu, z nichž asi 3/4 jsou v kompilované formě. Pro DOS jde o systém, který je pouze příkazově orientovaný.

Výrazného zlepšení komfortu obsluhy bylo docíleno ve verzích 3.*, které jsou vytvořeny jako řádná aplikace pod Microsoft Windows 3.1. Pro primární interakci se používá příkazového okna. Některé standardní funkce jsou v roletových nabídkách a je plně využito možností ukládání textů i obrázků v rámci Windows. Také systém nápovědy je zpracován podle standardní koncepce Windows.

3 PRÁCE SE SYSTÉMEM

Technicky představuje S Plus funkčně orientovaný jazyk s jednoduchou syntaxí. Elementární příkazy (zadávané za řádkovým ukazatelem >, který se objevuje na obrazovce automaticky) jsou buď výrazy nebo přiřazení. Pokud se zadá výraz jako příkaz provede se jeho vyhodnocení a zobrazení, ale hodnota výrazu se nikam nezaznamená.

Např.

```
> sqrt((2+3)*5)
```

zobrazí hodnotu 5 a očekává další příkazy.

Při přiřazení se také vyhodnotí výraz, ale zapíše se do proměnné, aniž se zobrazí. Přiřazovacími operátory jsou <- nebo _.

Např.

```
> cp <- sqrt ((2+3)*5)
```

neprovede nic pouze se do proměnné c zapíše číslo 5. Pro zobrazení výsledku se pak musí zapsat cp, kdy se na obrazovce objeví 5. Výrazy mohou být i dosti komplikované.

Pro frekventované výpočetní postupy a funkce může uživatel definovat vlastní funkce. Tyto funkce lze používat podobně jako již zabudované funkce, nebo je mohou nahradit. Demonstrujme možnosti S Plus na jednoduché úloze výpočtu variačního koeficientu pro prvních pět celých čísel. Nejdříve vytvoříme vektor data obsahující čísla 1 až 5 pomocí jednoduchého přiřazovacího příkazu

```
> data <- c(1:5).
```

A. Výpočet využívající pouze příkazy

```
>    mean (data) / sqrt(var(data))
```

a dostaneme přímo výsledek 1.8973

B. Výpočet využívající přiřazení

```
> m <- mean(data)
> v <- var(data)
> cv <- m/sqrt(v)
> cv
```

a výsledkem je opět 1.8973

C. Vytvoření uživatelem definované funkce

```
> cv <- function (y) {
> m <- mean(y); v<- var(y)
> cv  <- m/sqrt (v)
> }
```

Pro konkrétní výpočet se zavolá funkce cv s argumentem data

```
cv(data)
```

a v proměnné cv je výsledek. Je zřejmé, že funkce cv definovaná v odst C. se dá použít zcela obecně, např. v dalších výrazech atd.

Jednotlivé výrazy mohou být seskupovány do skupin uzavřených v hranatých závorkách

```
{ expr1, expr2 .. exprn}
```

Výsledkem je hodnota exprn. Tyto skupiny mohou být opět součástí vyšších výrazů atd.

Jazyk S obsahuje také příkazy umožňující tvorbu programů (ve tvaru velké funkce), které jsou blízké příkazům jazyka C. Mezi dva základní příkazy patří cykl a logické

rozhodování. Rozhodovací příkaz má strukturu

If (expr 1) expr 2 else expr 3

kde expr 1 je příkaz, jehož výsledkem je logická proměnná. Příkaz cyklu má tvar

for (Index in expr1) expr2

kde index je označení (formální) indexu, expr1 je vektorový výraz (často sekvence typu 1:5) a expr2 obsahuje index.

Např. pro sumaci všech prvků obsažených ve vektoru data lze použít kombinace obou výše uvedených příkazů

> for (i in 0 : length (data)) {if (i == 0) s <- 0 else s <- s + data [i] }

V proměnné s je nyní suma. Pro sumace je v jazyce S také funkce sum().

Operátory

Základní aritmetické operátory pro sčítání, odčítání, násobení a mocnění jsou:

+ - * / ^.

Logické porovnávací operátory jsou:

< > <= >= == !=

pro "menší než", "větší než", "menší nebo rovno", "větší nebo rovno", "rovno" a "nerovno".

Logické operátory jsou:

& | !

pro operace typu "and", "or", a "not"

Operátor ":" umožňuje vytvoření sekvence s krokem 1 mezi libovolnou dvojicí čísel. Příkaz

> c(4.1 : 1.1)

vytvoří vektor s prvky 4,1 3,1 2,1 a 1,1

Operátor %/% umožňuje celočíselné dělení a operátor % % určí zbytek po dělení prvního operátoru druhým.

Data mohou být extrahována z datových objektů (viz dále) s využitím indexů ve tvaru

data [index].

Indexy mohou být dvou základních typů:

a) celá čísla. Pak se objeví příslušný element vektoru "data". Např. data [2:4,7] zobrazí druhou, třetí, čtvrtou a sedmou hodnotu vektoru data

b) logické hodnoty. Pak se vyberou jen ty hodnoty, pro které je výsledkem logická jednička (TRUE). Např.

> data [data < 0]

zobrazí pouze záporné prvky vektoru data. Je možné použít také přiřazení. Např. záporné prvky vektoru data budou nahrazeny nulami při použití příkazu

> data [data <0] <- 0

Výhodou jazyka S je to, že se v anomálních situacích snaží o určení nejvhodnějšího výsledku. Např. při operacích, kdy operandy jsou nestejné délky se provádí cyklické opakování kratšího tak, až se docílí stejné délky jako u delšího operandu. Příkaz

> data + 4

tedy znamená, že ke každému prvku vektoru data se připočte konstanta 4. Podobně pokud dojde k problému nestejného typu operátoru (logické, numerické) provede se převod na stejný typ, který je více informativní.

Manipulace s daty

Jazyk S pracuje s datovými strukturami označenými jménem. Nejjednodušší datovou

strukturou je vektor, který představuje uspořádanou skupinu čísel (znaků). Např. vektor vec obsahující čísla 1, 5, 9, 7 se vytvoří pomocí funkce

```
> vec <- c(1, 5, 9, 7)
```

Výsledkem funkce c(....) je tedy převedení jejích argumentů do složek vektoru. Další možností je interaktivní zadávání složek z klávesnice pomocí funkce scan().Začíná se příkazem

```
> vec <- scan()
```

Po zadání té funkce se objeví ukazatel 1: a očekává se zadání čísel. Po každém zmáčknutí klávesy RETURN se objeví číslo o jednotku vyšší. Ukončení zadání se provede stisknutím klávesy RETURN bez předchozího zadání čísla. Obrazovka má pak např. tvar

```
1 : 1
2 : 5
3 : 9 7
4 :
>
```

Funkce scan() se dá použít také ke čtení souborů ve formátu ASCII uložených v počítačovém médiu. Např. příkaz

```
> vec <- scan("c:\\SP\\ vec.txt")
```

přečte soubor vec.txt z disku c, knihovny SP a přiřadí jeho obsah vektoru vec.

Pokud je třeba naplnit vektor od čísla (č 1) do čísla (č 2) s krokem (k), případně délkou (l), použije se příkaz

```
seq(č1, č2,k,l),
```

resp.

```
seq(from=č1,to=č2,by=k,length=l)
```

Příbuznou funkcí je rep(..), která umožňuje různé typy replikace zadaných vektorů. Nejjednodušší je

```
> s1 <- rep(vec,times=5)
```

kdy vektor s1 obsahuje 5 po sobě následujících vektorů vec. Protože jazyk S umožňuje manipulaci s logickými proměnnými lze konstruovat logické jedničky (TRUE) a logické nuly (FALSE). Logické vektory se tvoří z číselných vektorů pomocí logických operátorů. Např.

```
> s1 <- vec < 5
```

nastaví prvek s1[i] na logickou jedničku, pokud vec[i] < 5. Jinak bude nastavena logická nula.

Pokud nejsou některé složky vektoru známy (chybějící hodnoty) přiřazuje se jim hodnota NA. (Každá operace s NA má za následek hodnotu NA). Funkce is.na() vrací logický vektor s hodnotami (TRUE) na místech, kde jsou v původním vektoru NA. Vektor může být tvořen také znakovými řetězci

```
> vez <- c ("x1", "x3", "x7")
```

Vektory mohou být použity v různých aritmetických výrazech (operace se provádí "prvek po prvku") nebo mohou být argumentem různých funkcí a modelů.

Komplikovanější datovou strukturou jsou pole a matice obsahující různé vektory dat stejného typu. Každé pole má vektor rozměrů (s prvky tvořenými celými kladnými čísly). Hodnoty prvků vektoru rozměrů udávají horní limity pro jednotlivé indexy pole (spodní limita je vždy 1). Předpokládejme, že máme vektor vec s 12 prvky. Přiřazení

```
> dim(vec) <- c(2, 2, 3)
```

umožňuje zpracovávat vektor vec jako třírozměrné pole 2x2x3. Jednodušší je použití funkcí array() pro obecné pole a matrix() pro dvourozměrné pole.
Např.

```
> Z <- matrix(Z, 3, 4)
```

převede vektor Z na matici 3x4.

Při plnění polí se používá konvence FORTRANU, tj. první index se mění nejrychleji a poslední nejpomaleji. Také pole a matice mohou být použity jako argumenty aritmetických výrazů (operace se provádějí po prvcích). V případě matic se provádí maticové násobení pomocí operátoru % * %.

Matice lze tvořit z vektorů a submatic pomocí příkazu

```
cbind()
```

tj. horizontálním (po sloupcích) spojováním nebo pomocí příkazu

```
rbind()
```

tj. vertikálním (po řádcích) spojováním. Také pro extrakci prvků z matic a polí se používají hranaté závorky.

Zobecněním pojmu pole je seznam, který je vlastně složen z různých typů vektorů a polí. Může tedy obsahovat numerické, logické a znakové vektory, pole, funkce atd). Jednotlivé složky jsou očíslovány a mohou být také pojmenovány. Řada funkcí jazyka S poskytuje výsledky jako seznam. Pro selekci i-té položky seznamu SE se používá dvojích hranatých závorek, tedy SE [[i]].

Speciálně pro potřeby modelování byla vytvořena datová struktura "data frame" (dále se označuje jako frame). Objekt frame obsahuje řádky a sloupce dat podobně jako matice s tím, že sloupce mohou být různého typu (znakové, numerické atd). Jak sloupce, tak řádky mohou být pojmenovány. Tyto datové struktury se tvoří z vektorů a matic pomocí funkce

```
data.frame().
```

Pro nelinenární regresi se používá struktura "parametrizovaný frame" obsahující vektory x-ových a y-nových hodnot měřených bodů, názvy regresních parametrů a jejich první odhady. Parametrizovaný frame se tvoří pomocí funkce parameters ().

Např. chceme připravit strukturu parametrizovaný frame pro odhad parametrů modelu

```
y = alfa + b * exp (kapa*x)
```

Nechť x-ové složky experimentálních bodů jsou ve vektoru xs a y-nové složky ve vektoru ys. Pak strukturu my.frame vytvoříme pomocí příkazů

```
> my.frame <- data.frame (x=xs, y=ys)
```

První odhady přidáme jako seznam

```
> parameters (my.frame) <- list (alfa=10,b=0.5,kapa=-0.01)
```

Je zřejmé, že použití datových struktur frame a parametrizovaný frame umožňuje jednotně zadávat data pro řadu typů statistických modelů.

Grafika

Grafické možnosti jsou jednou z velkých předností jazyka S. Obecně je třeba na začátku práce s grafy definovat nejdříve zařízení pro zobrazení grafů (např. win.graph() u S Plus for Windows) a pak lze zadávat grafické příkazy.

Zajímavou funkcí umožňující kreslení více obrázků na jednu stránku je par(), ve které se zadává parametr mfrow. Tento parametr definuje umístění obrázků po "řádcích". Např. dva obrázky pod sebou se definují příkazem

```
> par(mfrow = c(2,1))
```

Základní funkce pro zobrazení jsou:

plot (x,y..)	**kreslí rozptylový diagram odpovídající vektorům x a y**
points (x,y..)	**přidává body k existujícímu rozptylovém diagramu**
lines (x,y..)	**přidává čáry k existujícímu rozptylovému diagramu**
texts (x,y)	**přidává text od bodu (x, y)**
abline(a,b)	**zakreslí čáru se směrnicí b a úsekem a**
abline(h=c)	**zakreslí horizontální čátu ve výšce C**
abline (v=c)	**zakreslí vertikální čátu ve výšce c**
abline(lsfit(x,y))	**zakreslí regresní přímku určenou MNČ**
abline(lmsreg(x,y))	**zakreslí regresní přímku pro robustní metodu LMS.**

Např.

```
> plot(x,y)
> lines(spline(x,y))
```

zobrazí body a zakreslí klasický interpolační spline.

Je možnétaké interakce v grafech pomocí identifikace bodů a přidávání informací, případně dynamická manipulace s datovými strukturami. Např. pro 3D rotaci vybraných tří sloupců matice slouží příkaz

```
spin()
```

Příkaz

```
brush()
```

umožňuje interaktivní práce v grafech obsahujících všechny párové rozptylové diagramy.

Mezi speciální typy grafických funkcí patří

barplot()	sloupcový diagram
boxplot()	krabicový graf
hist()	histogram
dotchart()	tečkový diagram
pie()	koláčový graf
qqnorm()	Q-Q graf pro normalitu
density()	jádrový odhad hustoty pravděpodobnosti

Pro vícerozměrná data lze použít grafy jako

contour()	vrstevnicový graf
persp()	3D perspektivní pohled na zadaný povrch
faces()	Chernovovy "tváře"
pairs()	všechy párové rozptylové diagramy
symbols()	symbolové zobrazení
stars()	"hvězdicové" zobrazení
plclust()	zobrazení pro hierarchické shluky
image()	zobrazení úrovní v barvách

Pro data ve formě časových řad jsou k dispozici také speciální grafické funkce jako

tsplot()	zobrazení časové řady
monthplot()	sezónní komponenty časových řad
acf()	graf autokorelační funkce
spectrum()	periodogram

Statistické funkce

Existuje celá řada statistických funkcí pro odhady parametrů a testy. Některé poskytují pouze jednu hodnotu, některé více hodnot a některé dokonce pole nebo matici v závislosti na dimensi argumentu. Např.

var(M) určí buď rozptyl (pro vektor M) nebo kovarianční matici (pro matici M)
quantil() určí kvantily pro zadané pravděpodobnosti
mean(),median() určí průměr resp. medián
summary() určí sumární statistické informace (minimum, dolní kvartil, medián, průměr, horní kvartil, maximum)
t.test() Počítá různé typy t-testů

Např. pro testaci hypotézy Ho:průměr=1 se použije příkaz

```
> t.test(x,mu=1)
```

Seznam klasických testů zahrnuje jak parametrické (t, Chí kvadrát, Fisherův), tak i neparametrické (Friedman, Wilcox, Kruskall).

Regulační diagramy

S Plus obsahuje řadu funkcí pro zpracování regulačních diagramů. Je možno konstruovat tyto základní typy diagramů

Typ	Statistika	Popis
xbar	průměr	průměr spojité proměnné
s	směr.odchylka	směr.odchylka spojité proměnné
R	prozmezí	rozmezí spojité proměnné
np	počet	počet zmetků
p	podíl	podíl zmetků
c	počet	počet vad na jednotku
u	počet	počet vad na nestejně velké jednotky

Při konstrukci regulačních diagramů se postupuje ve dvou krocích:

1. vytvoří se "QCC" objekt ze známých procesních dat (v době, kdy byl proces pod kontrolou)
2. vytvoří se vlastní kontrolní karta pro nová data s využitím objektu "QCC".

Např.

```
> data <- matrix(10 + rnorm(100),ncol=5)
```

vytvoří matici o 20-ti řádcích a 5 sloupcích obsahujících náhodná čísla kolem 10.
Objekt QCC má pak např.tvar

```
> ob.d <- qcc(data,type="xbar")
```

Shewhartův regulační diagram pro data v objektu ob.d se vytvoří příkazem

```
> shewhart(ob.d)
```

a diagram kmulativních součtů příkazem

```
> cusum(ob.d)
```

Matematické funkce

S Plus obsahuje tyto základní funkce

sqrt()	odmocnina
abs()	absolutní hodnota
sin(),cos(),tan()	trigonometrické funkce (rad)
asin(),acos(),atan()	inverzní funkce
sinh(),cosh(),tanh()	hyberbolické funkce
asinh(),acosh(),atanh()	inverzní funkce
exp(),log()	exponenciála a přirozený logaritmus
log 10()	logaritmus při základu 10
gamma(),lgama()	gamma funkce a její logaritmus

Každá funkce se vyčísluje prvek po prvku svého argumentu.

Maticové výpočty

Uvažujme matici M. Její transpozice se provádí příkazem

```
> t(M)
```

a diagonální prvky se určí pomocí příkazu

```
> diag(M)
```

Stopa čtvercové matice se počítá jako kombinace funkcí sum a diag

```
> sum(diag(M))
```

Podobně se konstruuje funkce determinant s využitím funkce eigen

```
> det <- function(x) prod(eigen(x)$values)
```

Pro vlastní výpočet se pak použije příkaz **det(M)**. Zde **prod()** je součin a **eigen(x)$value** jsou vlastní čísla matice x. Podobně **eigen$ectors(x)** jsou vlastní vektory matice x. Pro Choleského rozklad matice M na tvar **M = RTR**, kde **R** je horní trojúhelníková matice se provádí pomocí funkce

```
> chol(M)
```

Q-R dekompozice se provádí s využitím funkce

```
> qr(M)
```

a SVD pomocí funkce

```
> svd (M).
```

Pro práci s maticemi a poli existují speciální funkce **apply()** a **sweep()**. Fukce **apply()** provádí aplikaci zadané funkce na definovanou část dat a funkce **sweep()** provádí operaci nad určenou částí dat .

Uvažujme matici M (nxm) obsahující jako řádky pozorování a jako sloupce proměnné. Určení vektoru průměrů pro všechny proměnné se provede např. takto

```
> uprum <- apply( M, 2, mean)
```

a výpočet centrované matice matc pomocí funkce

```
> matc <- sweep( M, 2, uprum, "-")
```

Výhodou těchto funkcí je to, že lze pracovat s obecnými vícerozměrnými poli dat a provádět operace přes zvolené indexy.

Derivace a integrály

Pro výpočet integrálu zadané funkce v zadaném intervalu se používá funkce **integrate**, kde první složka je zadaná funkce

```
> integrate(cos,0,pi)
```

Pro symbolické derivace je možno použít funkce D.

```
> D(expression(4*cos (2* x)), "x")
```

Optimalizace

Pro hledání kořenů funkcí a lokálních extrémů je možno použít zabudované příkazy

polyroot() ...hledá kořeny polynomického modelu
uniroot() ... hledá kořen jednodimensionální funkce v zadaném intervalu
peaks() .....hledá lokální maxima v zadaných diskrétních bodech
optimize() ..hledá extrém jednorozměrné funkce v zadaném intervalu
ms() ..... hledá lokální minimum vícerozměrné funkce
nlminb() .....hledá lokální minimum vícerozměrné funkce s omezeními na parametry
nls()..... hledá lokální minimum součtu čtverců
nlregb()..... minimalizuje součet čtverců s omezením

Pravděpodobnostní výpočty

S plus má funkce pro realizaci pravděpodobnostních výpočtů a generaci náhodných čísel pro řadu rozdělení. Každá z těchto funkcí začíná písmenem

r ... pokud jde o generátor náhodných čísel
p ... pokud jde o distribuční funkci
d ... pokud jde o hustotu pravděpodobnosti
q ... pokud jde o kvantilovou funkci

Další znaky přísluší zvolenému pravděpodobnostnímu rozdělení

beta ... beta rozdělení (dva parametry)
binom .. binomické rozdělení
cauchy .. Cauchyho rozdělení
chsq .. Chí square rozdělení (stupněvolnosti)
exp .. exponenciální rozdělení
f ... Fisherovo rozdělení (df1, df2)
gamma .. Gamma rozdělení
geom ... geometrické rozdělení
hyper .. hypergeometrické rozdělení
lnorm ... lognormální rozdělení
logist .. logistické rozdělení
ubincom . negativní binomické rozdělení
norm .... normální rozdělení
pois ... Poissonovo rozdělení
t ..... Studentovo rozdělení (df)
unif .... rovnoměrné rozdělení
weibull .. Weibullovo rozdělení
wilcox .. Wilcoxonovo rozdělení

Např. generace 50 pseudonáhodných čísel z normálního rozdělení se střední hodnotou 5 a rozptylem 12 se provede příkazem

```
> rnorm(50,5,12)
```

95 % ní kvantil Studentova rozdělení s 10 stupni volnosti se určí příkazem

```
> qt(0.95, 10)
```

4 STATISTICKÉ MODELY

Výhodou jazyka S plus je, že má zabudované možnosti analýzy celé řady různých statistických modelů jako jsou

- lineární modely - příkazy **lm()**,**lmsreg()**
- modely ANOVA - příkaz **anova()**
- zobecněné lineární modely - příkaz **glm()**
- lokální regrese - příkaz **loess()**
- aditivní modely - příkazy **ace()**,**avas()**
- regresní stromy - příkaz **tree()**
- nelineární modely - příkazy **nls()**,**ms()**

Všechny tyto modely jsou zpracovávány stejným způsobem.

a) vytvoří se datová struktura frame nebo parametrizovaný frame obsahující data
b) zavede se objekt definující strukturu modelu.

Při definici modelů se (s vyjímkou nelineární regrese) používá kompaktní vyjádření kde mají základní operátory (+ - * :) speciální význam. Označme symbolem F_1 a F_2 dva různé výrazy, faktory nebo funkce. Pak zápis $y \sim F_1$ znamená, že proměnná y je modelována jako F_1 . Základí operátory pak mají tento význam :

$y \sim F_1 + F_2$	jsou zahrnuty oba výrazy (faktory)
$y \sim F_1 - F_2$	z F_1 je vypuštěno vše co je v F_2
$y \sim F_1 : F_2$	je zahrnuta interakce obou faktorů
$y \sim F_1 * F_2$	je ekvivalentní $1 + F_1 + F_2 + F_1 : F_2$
$y \sim F_1 \hat{} m$	jsou zahrnuty interakce až do řádu m

Pomocí funkce I() je možné vrátit oprátorům jejich původní význam. Na obou stranách definice modelu mohou být ještě různé trnsformace.

Příkladem objektů definujících strukturu modelů jsou:

y~ x1 + x2	(lineární model $y = a_0 + a_1 x_1 + a_2 x_2$)
y~ -1 + x	(model přímky procházející počátkem)
y~ poly (x, 3)	(model kubického polynomu)
1/y~ poly(x1 ,2) + log(x2)	(model $y = 1/(a_0 + a_1 x_1 + a_2 x_1^2 + a_3 \log(x_2))$)
y~ x1 + x2 + I(x1 * x2)	(lineární model $y = a_0 + a_1 x_1 + a_2 x_2 + a_3 x_1 * x_2$)

Pro nelineární regresi se modely zapisují v běžné syntaxi. Tedy

y ~ alfa+ b0* exp(beta * x)

označuje exponenciální model s parametry alfa, b0 a beta. Pomocí funkce **formula ()** lze vytvořený model uložit do objektu.

Pro vlastní analýzu většiny typů modelů platí stejná syntaxe jako např. pro lineární regresi MNČ, kdy se přiřadí objekt lm s parametry model a data frame zvolenému objektu

fit <- lm(model, data frame)

V objektu fit jsou pak uloženy výsledky regrese. Tento objekt sám může být argumentem pro různé funkce určující specifické výsledky nebo různé odvozené charakteristiky regrese. Tak funkce **summary ()** zobrazuje základní informace o výsledku regesní analýzy. Funkce **coef()** zobrazuje odhady parametrů, funkce **rezid()** zobrazuje rezidua a funkce **plot()**

zobrazuje graf reziduí proti predikci a hodnot y_i proti predikci. Pomocí funkce **lm.influence()** lze získat dhady parametrů a reziduálních rozptylů pro regresi, kde jsou postupně vynechávány jednotlivé body resp. diagonální prvky projekční matice. To umožňuje poměrně jednoduše počítat různé charakteristiky vlivných bodů založené na principu jejich vynechávání.

Tak např. plně studentizovaná rezidua $t_i = e_i/(s_i * \sqrt{(1-H_i)})$, (kde e_i jsou kasická rezidua, s_i jsou reziduální rozptyly pro případ, že při výpočtu regresních koeficientů byl vynechán i-tý bod a H_i jsou diagonální prvky projekční matice) lze vypočítat pomocí příkazů

```
> infl <- lm.influence(fit)
> ei   <- residuals(fit)
> si   <- infl$sigma
> hi   <- infl$hat
> ti   <- ei/(si*(1-hi)^.5)
```

Zde např. infl$sigma označuje, že se z objektu infl vybírá položka sigma (vektor). Pro určení indexů podezřelých bodů (data nechť jsou uložena ve frame "dat") postačí použít příkaz

```
> rownames(dat)[ti^2>10 ]
```

Pokud používáme grafické znázornění charakteristik vlivných bodů lze použít pro jejich identifikaci funkce **identify()**.

Z dalších funkcí uveďme pouze **summary.aov()** umožňující realizaci analýzy rozptylu (testaci významnosti příspěvků jednotlivých regresorů), **lm.sensitivity()** pro určení citlivosti odhadů na malé změny v datech (multikolinearita), **predict()** pro určení predikcí v místech definovaných uživatelem, **update()** pro postupné přidávání a **drop()** pro postupné vylučování regresorů.

Objekt **lm()** může ještě obsahovat další parametry definující váhy jednotlivých bodů a numerickou metodu řešení přeurčené soustavy rovnic (je možný QR rozklad a Choleskiho rozklad příp. SVD). Obdobně lze provádět analýzu dalších typů statistických modelů.

5 ZÁVĚR

Z tohoto zdaleka ne úplného výčtu je patrné, že S plus je poměrně mohutným nástrojem, jehož možnosti rostou s tím jak se uživatel zdokonaluje v jeho jazyce. Lze s ním pracovat jako s interaktivním prostředím, nebo programovat vlastní aplikace. To ho předurčuje jako nástroj pro ty, kteří řeší skutečně praktické problémy. V současné době má možnosti interface také s jazykem MATHEMATICA, což dále rozšiřuje jeho využitelnost i pro další typy vědecko technických výpočtů.

6 LITERATURA

[1] Meloun M., Militký, J., Forina, M.: Chemometrics in Analytical Chemistry, vol 1, 2, Ellis Horrwood 1993,1994

[2] Becker R.A., Chambers, J.M., Wilks, A.R.: The New S Language,, Wadsworth Brooks 1988

[3] Chambers, J.M., Hasties, T.J.: Statistical Models in S Wadsworth Brooks 1992

[4] Ripley, B.: Introductory Guide to S-Plus, Res.Rept., Oxford University 1992