R pro samouky
stručný úvod

Michal Kulich

3. března 2004

Webová stránka pro výuku GLM: `http://www.karlin.mff.cuni.cz/~kulich/vyuka/glm/index.html`

Charakteristiky R:

- Interaktivní „výpočetní prostředí" pro statistické výpočty a grafiku
- Dialekt jazyka S volně šiřitelný pod GNU GPL
- Funguje pod Windows, Unixem, MacOS, všemi versemi Linuxu
- Řádkový interpret, objektově orientovaný
- Kompatibilní s S-Plus (až na ε)
- Snadno rozšiřovatelný (rozšiřující knihovny na WWW)

Reference:

- *Hlavní webová stránka projektu:* `http://www.r-project.org/` (informace, manuály, FAQ)
- *Hlavní depositář:* `http://cran.at.r-project.org/` (instalace, zdrojový kód, rozšiřující knihovny ke stažení)

Technika práce s R:

- *Nastartování:* Ve Windows 2000 například z menu Start/Programs/R/R 1.8.1 (závisí na instalaci a OS)
- *Volba pracovního direktoráře:* V pracovním direktoráři se vyhledávají a ukládají soubory dat a výsledků. Pod Windows lze nastavit pomocí menu File/Change dir.... Taktéž lze použít funkci `setwd()`. Nastavení pracovního direktoráře zobrazí funkce `getwd()`.
- *Uvádění cest (paths):* Pokud je třeba uvést se jménem souboru i cestu, použijeme i pod Windows buď obyčejné lomítko, nebo dvě zpětná lomítka. Píšeme tedy `c:/data` nebo `c:\\data`, nikoli `c:\data`.
- *Konsolové okno:* Slouží pro vstup příkazů z klávesnice a pro textový výstup. Předchozí příkazy lze vyvolat pomocí klávesy ↑.

- *Grafické okno:* Slouží pro grafický výstup. Lze vytisknout nebo uložit na disk ve zvoleném formátu.
- *Editace programů:* Lze použít externí editor a přenášet příkazy do R přes Copy–Paste. Celý program lze natáhnout a spustit příkazem `source("program.r")`.
- *Ukončení práce:* Práce s R se ukončuje příkazem `q()`. Následuje otázka, zdali se mají všechny existující objekty uložit na disk. Odpovíte-li „Y", vše se uloží do souboru `.RData` ve vašem pracovním direktoráři.

Nápověda:

- *Nápověda pro daný příkaz/funkci:* Vyvolá se v konsoli funkcí `help`, například `help(q)` pro ukončovací funkci `q()`.
- *Hypertextová nápověda s vyhledáváním:* Vyvolá se v konsoli pomocí `help.start()`. Nastartuje webový prohlížeč na úvodní stránce nápovědy. Přístup k nápovědě pro jednotlivé funkce, tříděné podle knihoven, jest skrze *Packages* a výběrem knihovny na následující stránce. Běžné příkazy a funkce jsou v knihovně *Base*. Navíc jsou zde k disposici čtyři manuály v angličtině a řada odkazů.

Vytváření, vypisování a mazání objektů:

R je objektově orientovaný jazyk: proměnné, data, výsledky, funkce, příkazy i jazykové konstrukce jsou objekty. Jedna funkce může vykonávat značně rozdílnou činnost v závislosti na typu objektů, které má zpracovat.

Jako názvy objektů jsou přípustné řetězce obsahující písmena, číslice a znak „." (tečka). Velká a malá písmena se rozlišují, tj. `barel` a `Barel` označují dva různé objekty.

Objekty se vytvářejí přiřazením. Základní přiřazovací operátor je `<-`, tj. posloupnost dvou znaků „<" a „−".

Jména existujících objektů lze vypsat funkcí `ls()`. Jména plus některé další informace vypisuje i funkce `ls.str()`. Existující objekty lze smazat funkcí `rm()`.

Komentované příklady:

`> X <- 14`	přiřazení objektu X
`> x <- sqrt(8)`	$x \neq X$
`> z <- x+X`	
`> z`	výpis hodnoty `z`
`[1] 16.82843`	výsledek
`> z+x^3`	spočti jednoduchý výraz
`[1] 39.45584`	
`> ls()`	výpis jmen existujících objektů
`[1] "x" "X" "z"`	

```
> ls.str()                                  výpis s více podrobnostmi
x : num 2.83
X : num 14
z : num 16.8
> zajic <- 22.21
> bazant <- 14.72
> ls()
[1] "bazant" "x" "X" "z" "zajic"
> ls(pattern="^z")                          výpis objektů začínajících na „z"
[1] "z" "zajic"
> ls(pattern="z")                           výpis objektů obsahujících „z"
[1] "bazant" "z" "zajic"
rm(x)                                       smaž objekt x
rm(list=ls(pattern="^z"))                   smaž všechny objekty začínající na „z"
rm(list=ls())                               smaž všechny objekty (opatrně!)
```

Volání funkcí

Většina funkcí v R má mnoho argumentů, ale zadávají se jen některé. Argumenty funkcí v R totiž mohou mít předdefinované hodnoty nebo mohou zůstat zcela nespecifikované. Pokud se volá funkce bez argumentů, je třeba uvést prázdné závorky; jinak se pouze vypíše definice funkce (zkuste `ls` versus `ls()`). Argument je možné identifikovat jménem, tak jako `ls(pattern="z")`, anebo pořadím. Např. `ls(2)` je totéž jako `ls(name=2)`, neboť `name` je první argument funkce `ls`. Jaké má určitá funkce argumenty, jaké jsou jejich předdefinované hodnoty a jaké mají argumenty význam, to vše je vysvětleno v nápovědě. Stačí například zadat `help(ls)`.

Datové typy

R rozlišuje čtyři základní datové typy (*mode*s): numerický, znakový, komplexní a logický. Typ objektu zjistíme funkcí `mode()`. Chybějící hodnoty jsou representovány kódem `NA` (*Not Available*). Speciální numerické hodnoty jsou `Inf`, `-Inf` a `NaN` ($+\infty$, $-\infty$, *Not A Number*). Znakové objekty se skládají z řetězců, které je nutno zadávat v dvojitých uvozovkách (`"Toto je retezec"`). Logické objekty obsahují konstanty `TRUE` a `FALSE` (psáno samými velkými písmeny!).

Datové struktury

Základní datové struktury v R jsou mj. vektor (*vector*), matice (*matrix*), pole (*array*), datová tabulka (*data frame*) a seznam (*list*).

Vektory:

```
> b <- c(1,5,8,1,5)                         Vytvoření vektoru z čísel
> b                                         Výpis
[1] 1 5 8 1 5
```

```
> b[4]
[1] 1
```
Hodnota daného prvku

```
> b <- 5:14
> b
 [1] 5 6 7 8 9 10 11 12 13 14
```
Aritmetická posloupnost s krokem 1

```
> b[3:6]
[1] 7 8 9 10
```
Subvektor

```
> b<7
[1] TRUE TRUE FALSE FALSE FALSE FALSE
FALSE FALSE FALSE FALSE
```
Logická operace na vektor

```
> b[b>9]
[1] 10 11 12 13 14
```
Hodnoty splňující podmínku

```
> z <- rep("+",4)
```
Vytvoření vektoru identických prvků (zde znakových)

```
> z <- c(z,rep("-",4),rep("+",2))
> z
[1] "+" "+" "+" "+" "-" "-" "-" "-" "+"
"+"
```
Ukázka lepení vektorů

```
> b[z=="-"]
[1] 9 10 11 12
```
Prvky jednoho vektoru na místech, kde druhý vektor splňuje podmínku

```
> q <- seq(1,8,by=0.05)
```
Aritmetická posl. se zvoleným krokem

```
> length(q)
```
Délka vektoru

Matice:

```
> mat <- cbind(b,rep(1,10))
```
Vytvoření matice ze sloupců

```
> mat <- rbind(b,rep(1,10))
```
Vytvoření matice z řádků

```
> mat <- matrix(c(1:6),c(11:16),ncol=3)
```
Vytvoření matice z prvků po sloupcích

```
> mat <- matrix(1:12,ncol=3,byrow=T)
```
Vytvoření matice z prvků po řádcích

```
> dim(mat)
[1] 4 3
```
Dimense matice

```
> mat[1,3]
```
Prvek matice

```
> mat[3,]
```
Řádek matice

```
> mat[,2]
```
Sloupec matice

```
> mat[1:3,2:3]
```
Submatice

Pole (*array*):

```
> pole <- array(1:24,c(3,2,4))
```
Vytvoření 3-rozměrného pole

```
> dim(pole)
[1] 3 2 4
```
Dimense pole

```
> pole[2,1,3]
```
Přístup k prvkům pole

```
> pole[2,,]
```
Přístup k vrstvám pole

Seznam (*list*):

Seznam je datová struktura, jejíž složky mohou mít různý typ i různou délku (rozměr). Každá složka seznamu má své jméno. Výsledky většiny statistických analýz včetně regresních modelů jsou objekty typu seznam.

`> l <- list(cislo=15,vektor=q,logic=(b<7),`	Vytvoření seznamu
`+ znaky=z)`	
`> l$cislo`	Přístup ke složkám jménem
`[1] 15`	
`> l[[1]]`	Přístup ke složkám pořadím
`[1] 15`	
`> l$logic[6:8]`	Přístup k prvkům složek
`[1] T F F`	
`> l[[2]][3]`	Příklad řetězení odkazů
`[1] 1.100000`	
`> names(l)`	Jména složek seznamu

Datová tabulka (*data frame*):

Datová tabulka je speciální případ seznamu, jehož složky jsou vektory různého typu, ale stejné dělky. Datovou tabulku lze považovat za zobecněnou matici, jejíž řádky odpovídají pozorováním a sloupce veličinám.

`> a <- data.frame(x1=c(rep(1,8),rep(0,8)),`	Vytvoření datové tabulky
`+ pohl=rep(c("Muz","Zena"),c(8,8)))`	
`> names(a)`	Výpis jmen veličin
`> names(a)[1] <- "Skupina"`	Změna jména veličiny
`> names(a)`	
`> a$pohl`	Přístup k veličinám jménem
`> attach(a)`	Zavedení datové tabulky k aktivní práci
`> search()`	Seznam aktivních datových tabulek a knihoven
`> Skupina[3]`	Nyní je možný přímý přístup k veličinám (bez $)
`> Skupina[3] <- 0`	Oprava dat
`> a$Skupina[3]`	V datové tabulce je stále původní hodnota
`> a$Skupina[3] <- 0`	Nyní to již bude opraveno
`> detach(a)`	Ukončení práce s datovou tabulkou
`> a <- read.table(file='fname.asc')`	Vytvoření dataframu z ascii souboru

Faktor:

Faktor je speciální forma znakového vektoru. Má atribut `levels` (úrovně) a používá se pro diskrétní veličiny. Pro faktor je možné předem nastavit kódování parametrů použité později při zařazení faktoru do regresního modelu.

```
> fac <- factor(rep(c(0,1,9),c(8,3,2))      Vytvoření (konverse) faktoru
> levels(fac)                               Výpis úrovní
> levels(fac) <- c("Chlap","Zenska","???")  Změna úrovní
> is.factor(a$pohl)                         Veličina pohl je již faktor
> levels(a$pohl)                            Výpis úrovní
> a$pohl
> codes(a$pohl)                             Uvnitř jsou numerické kódy
> a$pohl <- C(a$pohl,treatment)             Nastavme GLIMovou parametrizaci
                                            pro modelování faktoru pohl
```

Operátory

Operátory aplikované na vektory, matice nebo pole se provádějí po složkách. Výsledkem je opět vektor, matice nebo pole.

	Aritmetické		Relační		Logické
`+`	Sčítání	`<`	menší	`!`	negace
`-`	Odčítání	`>`	větší	`&`	a zároveň
`*`	Násobení	`<=`	menší nebo rovno	`|`	nebo
`/`	Dělení	`>=`	větší nebo rovno		
`^`	Mocnění	`==`	rovná se		
`%%`	Modulo	`!=`	nerovná se		
`%/%`	Celočíselné dělení				

Elementární funkce

Skalární: Přijímají za argument skalár, vektor, matici. Provádějí se po složkách. Příklady: `sqrt`, `exp`, `log`, `log10`, `gamma`, `abs`, `round`, `trunc`.

```
> mat <- matrix(seq(1,by=0.5,length=12),ncol=4,byrow=T)
> sqrt(mat)
> round(exp(-mat),4)
> round(mat*mat)
```

Vektorové: Přijímají za argument vektor nebo matici. Matice se transformuje na vektor po sloupcích. Příklady: `length`, `sum`, `prod`, `min`, `max`, `range`, `mean`, `median`, `quantile`, `var`, `cor`, `cumsum`, `cumprod`.

```
> v <- seq(1,by=0.5,length=12)
> sum(v)                                    Součet
> min(v)                                    Minimum
> range(v)                                  Vektor minima a maxima
> mean(v)                                   Průměr
> var(v)                                    Výběrový rozptyl
> quantile(v,c(0.3,0.5))                    Výběrové kvantily
> cumsum(v)                                 Kumulativní součty
```

Maticové: Přijímají za argument matici.

```
> mat1 <- matrix(c(5,0,-1,4),nrow=2)
> mat2 <- matrix(c(2,-2,0,1),nrow=2)
> t(mat1)                          Transposice
> diag(mat1)                       Diagonála
> mat1%*%mat2                      Maticový součin
> solve(mat1)                      Inverse
> kronecker(mat1,mat2)             Kroneckerův součin
> eigen(mat1)                      Vlastní čísla a vektory
> apply(mat1,2,sum)                Vektor sloupcových součtů
> apply(mat1,1,mean)               Vektor řádkových průměrů
```

Definice a volání funkcí

```
f <- function(x=1,n=2)
{
     p <- x^n
     return(p)
}
```

Místo `return(p)` může být pouze `p`.

Volání:

```
 > a <- f(7.4,3)                  se dvěma argumenty
 > a <- f(7.4)                    pouze s 1. argumentem (2. je předdefinovaný)

 > a <- f()                       oba argumenty předdefinované
 > a <- f(n=0.5,x=2)              explicitní pojmenování argumentů (nezáleží na pořadí)
```

Kontrolní příkazy a smyčky

`if` (*cond*){*expressions*}
`if` (*cond*){*exp1*} `else` {*exp2*}
`while` (*cond*) {*expr*}
`for` (*variable* `in` *range*) {*expr*}
`ifelse`(*vec.cond,vec.expr.1,vec.expr.2*)

Příklady: Tři alternativní definice funkce sgn

```
sgn1 <- function(x)
{
for (i in 1:length(x))
    {if (x[i]>0)
```

```
       x[i] <- 1
    else
       if (x[i]<0) x[i] <- -1}
x
}

sgn2 <- function(x)
{
ifelse(x>0,1,ifelse(x<0,-1,0))
}

sgn3 <- function(x)
{
x[x>0] <- 1
x[x<0] <- -1
x
}
```

Příkazu `for` se lze většinou vyhnout. R má řadu funkcí, které zpracovávají celé objekty najednou bez nutnosti chodit po složkách. Velmi užitečné jsou funkce `apply()`, `tapply()`, `sapply()` a `lapply()`.

Funkce:	*Vstup:*	*Činnost:*
`apply()`	Matice, pole	Proveď danou funkci na daný rozměr/sekci matice/pole *Př:* `apply(x,2,sum)` vektor sloupcových součtů matice `x`
`tapply()`	Vektor	Rozděl vektor na skupiny a spočítej funkci pro každou skupinu *Př:* `tapply(y,list(pohl,rasa),mean)` vytvoř tabulku průměrů veličiny `y` pro každou kombinaci pohlaví a rasy
`lapply()`	Seznam	Proveď danou funkci na každý prvek seznamu, výsledek je seznam *Př:* viz `help(lapply)`
`sapply()`	Seznam	Proveď danou funkci na každý prvek seznamu; výsledek je matice/vektor *Př:* viz `help(lapply)`

Načtení dat z externího souboru

R má specializované funkce pro načítání dat z textových souborů a jejich konversi na datovou tabulku nebo matici. Funkce `read.table()` vytvoří z textového souboru datovou tabulku se všemi atributy. Má speciální varianty `read.csv()`, `read.csv2()`, `read.delim()`, `read.delim2()` pro soubory s hodnotami oddělenými čárkami, středníky a tabulátory, a s desetinnými tečkami i čárkami. CSV (*comma-separated values*) formát je nejvhodnější

datový formát pro konversi mezi R a Excelem (a jinými programy). Funkce `scan()` taktéž načítá textové soubory, ale umožňuje větší kontrolu nad konversí než `read.table()`. Data v pevném formátu (*fixed-width fields*) se načítají funkcí `read.fwf()`.

Export dat z R do textových souborů provádí např. funkce `write.table()`.

Práce s chybějícími hodnotami