Jak na jazyk R

instalace a základní příkazy

Vladislav Bína
Arnošt Komárek
Lenka Komárková

2006

Co je R?

R je jazyk a zároveň prostředí vhodné pro provádění statistických výpočtů a tvorbu grafických výstupů. Je to volně šiřitelný nástroj, který vychází z velmi podobného jazyka S. Umožňuje vytvářet profesionální grafy a obrázky. Relativně snadno umožňuje přidávat další nástroje, uživatelé mají možnost svou práci maximálně zjednodušovat a prostředí si upravit podle svých představ. Do systému lze doinstalovat velké množství balíčků, které jeho možnosti dále rozšiřují.

Prostředí R je možné používat ve všech verzích Microsoft Windows (počínaje Windows 95), ale také v systémech Linux a MacOS X.

Více o jazyku R na stránkách: `http://www.r-project.org/`

Instalace prostředí R

Program samotný a některé přídavné balíčky lze před instalací zkopírovat ze školního disku K:, konkrétně `K:\PED\KMIH\verejny\kmm\Komarek\software`. Kde naleznete adresáře se dvěma instalacemi pro různé operační systémy: jednak pro Windows (`Rko-Windows`) a jednak pro linux (`Rko-Linux`). Kromě toho je tento software ke stažení na stránkách **R Project**, které naleznete na výše uvedené adrese (`http://www.r-project.org/`), budete-li následovat odkaz **CRAN** a vyberete-li umístění, odkud chcete program stahovat. Je vhodné vybrat umístění (CRAN Mirror) v blízké lokalitě, zvolíte-li rakouskou Wirtschaftsuniversitaet Wien, je možné použít přímo odkaz: `http://cran.at.r-project.org/`.

Zde vyberete systém, který běží na vašem počítači a pod nímž prostředí R poběží, tedy zřejmě prostředí Windows. Dále následujete-li odkaz **base**, můžete stáhnout soubor `README.R-*`[1] a vlastní instalaci programu (v případě systému Windows) `R-*-win32.exe`[2].

Pokud máte instalaci programu uloženou v počítači a máte oprávnění na daném počítači instalovat, můžete `R-*-win32.exe` spustit[3].

[1] Místo hvězdičky aktuální číslo verze - například: `README.R-2.3.1`.

[2] Zde opět místo hvězdičky aktuální číslo verze - například: `R-2.3.1-win32.exe`.

[3] I když to obecně nelze doporučit, lze prostředí R bez potíží již nainstalované (obvykle v adresáři `Program Files\R`) překopírovat do jiného počítače a začít jej používat.

- Nejdříve se vás program zeptá na jazyk, který má použít při instalaci.
- Po odsouhlasení se objeví obvyklý **Průvodce instalací**, kde stisknete tlačítko **Další >**.
- Poté odsouhlasíte podmínky licenční smlouvy a opět stisknete **Další >**.
- Nyní je potřeba volit složku, kam bude program nainstalován, obvykle stačí ponechat implicitní předvyplněnou cestu, jen v případě, že tam nemáte povolený zápis, zvolte jiný adresář na vám přístupném disku (v učebně například disk D:).
- V dalším dialogu pak zvolte součásti, které se mají instalovat, předvolené můžete ponechat nebo vybrat **Full installation**.
- Poté zvolíte jméno složky v nabídce **Start**, opět lze ponechat.
- A konečně zvolíte další úkoly, které se mají provést, opět není obvykle potřeba měnit, jen v případě potíží s oprávněními lze tato zaškrtnutí zrušit.
- Pak se spustí samotná instalace a po jejím proběhnutí zbývá jen stisknout tlačítko **Dokončit** a poté prostředí R spustit.

Rozšiřující balíčky lze nalézt na stránkách **CRAN** v sekci: **Packages**, ale o tom v následující části.

Instalace rozšiřujících balíčků

Balíčky ve formě **zip** archivů lze obecně získat opět například ze stránek `http://cran.at.r-project.org/` pod odkazem **Packages**.

Nicméně balíčky, které budou potřebné v rámci výuky naleznete opět na školním disku K:, ve výše zmíněném adresáři s instalací R, konkrétně pro windows: `K:\PED\KMIH\verejny\kmm\Komarek\software\Rko-Windows\baliky`

Pokud je již máte na některém z disků přístupném v počítači, spusťte si prostředí R a pro instalaci balíčků lze použít na liště menu **Packages** položku **Install package(s) from local zip files ...** Poté vyhledáte cestu, kde jsou rozšiřující balíčky uloženy (tedy na školních počítačích to bude výše zmíněná

složka `K:\PED\KMIH\verejny\kmm\Komarek\software\Rko-Windows\baliky`) a se stisknutou klávesou **Shift** klikněte levým tlačítkem myši na první a poslední balíček v této složce. Jsou-li označeny všechny balíčky, stiskněte **Otevřít** a instalace balíčků proběhne. Tímto by vaše Rko mělo být připraveno k práci.

Malá vsuvka: R Commander

Prostředí R můžete již nyní začít používat a psát příkazy v **R Console**, tedy příkazovém okně, které se objeví po spuštění. Nicméně je tu ještě možnost používat nadstavbu, kterou jste nainstalovali v rozšiřujícím balíčku **Rcmdr** neboli **R Commander**. Spustíte jej tak, že v okénku **R Console** zapíšete příkaz `library(Rcmdr)` (a odešlete jej k vykonání klávesou **Enter**).

Zobrazí se vám okno **R Commanderu**, v horní části (**Script Window**) můžete zapisovat příkazy, které poté vykonáte, umístíte-li kurzor na příslušný řádek a stisknete tlačítko **Submit** (případně můžete označit více příkazů najednou a použít **Submit**). V dolní části **Output Window** se pak zobrazují výsledky příkazu.

Použití je tedy v zásadě obdobné jako u prosté **R Console**, s tím rozdílem, že zadávané příkazy a jim příslušející výstup jsou v oddělených oknech a oboje lze snadno uložit. Kromě toho **R Commander** usnadňuje mnoho dalších úkolů, jako je načítání a editování dat a jiné operace, které jsou přístupné v „klikací“ formě přímo z jeho menu, aniž by bylo nutné přesně si pamatovat příkaz.

Ještě poznamenejme, že ne vždy je jeho použití úplně bezproblémové. Například příkaz pro vykreslení histogramu při použití **R Commanderu** špatně zobrazuje graf (přinejmenším ve verzi používané nyní na fakultě). Proto pokud budete zaskočeni zvláštním chováním, vyzkoušejte stejný postup ještě jednou v **R Consoli**.

V dalším textu nebudeme užití **R Commanderu** předpokládat, nicméně téměř vše by mělo fungovat bez problému i v něm.

Jak začít aneb R jako kalkulátor

Pokud používáte **R Console**, zapisujete jednotlivé příkazy na řádek začínající > a po dopsání jej odešlete stiskem klávesy **Enter**. V **R Commanderu** je to obdobné (viz výše). Můžete tedy začít zkoušet následující příkazy a jejich varianty.

`3+5-2`	Provede a zobrazí výsledek.
`((3*15 + 5)/5)`	Přednost operací lze usměrňovat závorkami.
`2^4`	Čtvrtá mocnina ze dvou.
`sqrt(16)`	Výpočet odmocniny.
`exp(-1)`	Funkční hodnoty exponenciely: e^{-1}.
`sin(3.14)`	Výpočet sinu, argument se zadává v radiánech.
`log(5-4)`	Přirozený logaritmus.
`log10(0.1)`	Dekadický logaritmus.
`log(25, base=5)`	Jiný základ, zde základ 5.
`rnorm(10)`	Deset náhodných čísel z normálního rozdělení.
`max(c(1,-2,9,2))`	Vybere maximální hodnotu z uvedených, viz vektory.
`min(runif(5))`	Nejmenší číslo z pěti náhodných s rovnom. rozdělením.
`round(pi,3)`	Zaokrouhlení čísla π na tři desetinná místa.
`11 %/% 5`	Celočíselné dělení, výsledek 2.
`11 %% 5`	Zbytek po dělení, výsledek 1.

Nápověda

K informacím o funkcích vedou následující příkazy:

`help(rnorm)`	Zobrazí nápovědu k funkci `rnorm`.
`?rnorm`	Jen stručnější varianta téhož.
`help.search(trigonometric)`	Prohledává systém nápovědy.
`?Trig`	Nápověda k trigonometrickým funkcím.
`help.start()`	Spustí HTML verzi nápovědy.
`apropos(plot)`	Hledá příkazy mající ve jméně daný text.
`ls()`	Vypíše již vytvořené proměnné.
`rm(aa)`	Zruší proměnnou `aa`.

Proměnné

Samotné výpočty jsou nedostatečné, často je potřeba proměnnou uložit.

`x <- 12`	Uloží do proměnné `x` hodnotu `12`.
`x = 12`	Stejné jako `x <- 12`.
`x`	Zobrazí uloženou hodnotu v proměnné `x`.

Proměnné mohou být různých typů, kromě číselných například řetězce (uzavřené v uvozovkách nebo apostrofech) nebo logické (viz kapitola Logické operátory).

Vektory a posloupnosti

Mnohdy jednotlivé proměnné nedostačují, jazyk R často používá vektory a posloupnosti hodnot, jak pro výstup tak pro zadávání dat. Ostatně jejich potřeba vyplývá už ze samotné podstaty statistických problémů.

`vec <- c(10,1, -4)`	Uloží do proměnné `vec` hodnoty `10`, `1` a `-4`.
`vec`	Zobrazí uložené hodnoty vektoru `vec`.
`seq(1,5)`	Vytvoří vektor přirozených čísel od `1` do `5`.
`1:7`	Opět vektor přirozených čísel od `1` do `7`.
`1.23:5`	Aritmet. posloupnost s diferencí 1 mezi `1.23` a `5` s výsledkem: `1.23 2.23 3.23 4.23`.
`v2 <- seq(1,9,by=2)`	Vektor lichých čísel od `1` do `9` v proměnné `v2`.
`v2`	Zobrazení s výsledkem: `1 3 5 7 9`.
`v2[4]`	Čtvrtá položka vektoru `v2`.
`v2[3:5]`	Třetí až pátá položka vektoru `v2`.
`v2[c(1,3,4)]`	První, třetí a čtvrtá položka vektoru `v2`.
`seq(1,3,by=.1)`	Vektor čísel od `1` do `3`, následující je o `0.1` větší.
`seq(1,7,length.out=5)`	Vektor délky `length.out=5` z čísel od `1` do `7`, s výsledkem: `1.0 2.5 4.0 5.5 7.0`.
`rep(1:4,2)`	Zopakuje dvakrát posloupnost `1:4` s výsledkem: `1 2 3 4 1 2 3 4`
`rep(1:4,each=2)`	Zopakuje dvakrát každé číslo posloupnosti `1:4` s výsledkem: `1 1 2 2 3 3 4 4`
`numeric(7)`	Vytvoří nový vektor obsahující sedm nul.

Délku vektoru lze zjistit pomocí příkazu `length()`, tedy např.: `length(v2)`.

Pracovní složka

Dříve než se pokusíme o načítání nebo ukládání dat, je potřeba si vyjasnit, jaká složka je aktuálně zvolena jako pracovní, případně ji změnit. Ulehčí nám to situaci a nebudeme muset vypisovat kompletní cestu k souborům na disku postupně přes jednotlivé složky.

`getwd()`	Zjistíme aktuální pracovní složku.
`setwd('D:/rko')`	Změníme pracovní složku na zadanou `D:\rko`.
`dir()` nebo `list.files()`	Jména souborů v aktuální nebo zadané složce.

Načítání a ukládání dat, datové rámce

V této části budeme načítat `data` se souboru `car.dat`, který je na školním disku `K:` ve složce `K:\PED\KMIH\verejny\kmm\Komarkova\statistika\R_materialy`. Zkopírujete-li si tento soubor například na disk `D:` do složky `D:\rko`, můžeme používat následující příkazy[4]:

```
data <- read.table("D:\\rko\\car.dat")
```
Načte obsah souboru do proměnné `data`, nerozezná, že první řádek obsahuje názvy sloupečků a nerozezná od sebe jednotlivé sloupce.

`data` Zobrazí načtená data.

```
data <- read.table('D:/rko/car.dat', header = TRUE)
```
Načte obsah souboru do proměnné `data`, hlavička je správně rozeznána jako jména proměnných, vše je však stále jen v jednom sloupci.

[4]Cesta k souboru se zapisuje jako proměnná typu řetězec, tedy buďto do uvozovek nebo do apostrofů. Použití lomítek u cesty k souboru se může jevit poněkud nezvyklé, pokud použijete zpětné lomítko, je potřeba jej zdvojit, protože zpětné lomítko je v řetězcích používáno jako speciální znak (viz níže). Můžete použít místo zpětného lomítka obyčejné – jako v linuxu – a načtení bude rovněž bez problémů.

```
data <- read.table("D:/rko/car.dat", header=TRUE, sep=";")
```
Načte obsah souboru do proměnné `data`, v uvozovkách u parametru `sep` je uveden znak, který odděluje sloupce v souboru. Pokud není zadán znak oddělovače, předpokládá se, že oddělují mezery.

```
data<-read.table("D:/rko/car.dat",header=T,sep=";",na.strings="chybi")
```
Načte obsah souboru do proměnné `data`, v uvozovkách u parametru `na.strings` je uveden řetězec, který v datovém souboru označuje, že v něm chybí hodnota. Lze uvést i více takových řetězců ve tvaru vektoru. Kromě toho lze v tomto příkazu použít parametr `dec`, který označuje oddělovač celé a desetinné části čísla (např. desetinná tečka: `dec='.'`).

Výsledkem posledního uvedeného příkazu jsou hodnoty uložené v proměnné `data`, která je datovým rámcem (neboli rámcem zastřešujícím několik proměnných z načteného souboru), obsah může vypadat takto:

```
                       typ   druh     nahon kon.sila spotr.mesto spotr.dalnice delka sirka
1         Chevrolet.Aveo.4dr osobni    predni       NA         8.4           6.9   424   168
2    Mercedes-Benz.CL500.2dr osobni     zadni      302        14.7           9.8   498   185
3 Ford.F-150.Supercab.Lariat pickup ctyrkolka      300        16.8          13.1    NA    NA
4         GMC.Sierra.HD.2500 pickup ctyrkolka      300          NA            NA    NA    NA
```

Přičemž k jednomu sloupečku lze přistupovat pomocí příkazu `data$delka`:

```
424 498  NA  NA
```

To samé provede příkaz `data[,"delka"]`, případně příkaz `data[,7]`, pokud potřebujete např. jen první dvě hodnoty použijete `data$delka[1:2]` nebo `data[1:2,"delka"]`.

Kromě toho existují ještě další varianty příkazu `read.table` určené přímo načítání CSV souborů:

- `read.csv()`
  Načítaný soubor obsahuje hlavičku se jmény proměnných `header=TRUE`, počítá s oddělovačem `sep=","` a s desetinnou tečkou `dec="."`.
- `read.csv2()`
  Uvažuje soubor s hlavičkou obsahující jména proměnných `header=TRUE`, počítá s oddělovačem `sep=";"` a s desetinnou čárkou `dec=","`.

Datové rámce můžeme jednoduše vytvářet přímo v R. Treba takto:

```
jmena <- c("Adam", "Barbara", "Chris", "David")
```

```
vek <- c(33, 19, 27, 48)
vyska <- c(1.81,1.77,1.66,2.03)
lide <- data.frame(jmena,vek,vyska)
```

V datovém rámci `lide` jsou nyní sloupce:

```
    jmena vek vyska
1    Adam  33   1.81
2 Barbara  19   1.77
3   Chris  27   1.66
4   David  48   2.03
```

Můžeme jednoduše přidat další sloupec: `lide$vaha <- c(82,91,58,81)`. To lze udělat i následujícím příkazem `lide <- transform(lide, bmi=vaha/vyska^2)` nebo můžeme s pomocí tohoto příkazu převést údaj o výšce například na centimetry `lidecm <- transform(lide, vyska = vyska*100)`. Přičemž k transformaci můžeme vybrat libovolnou funkci, tedy například přirozený logaritmus `log()` nebo Z-transformaci `scale()`. Lze také vybrat jen některé sloupce `lide[,c('vek','vaha')]` nebo vytvořit nový rámec např. jen s údaji o věku, výšce a váze: `lide2 <- subset(lide, select=vek:vaha)`.

Uložit datový rámec lze pomocí příkazu `write.table()`, který má obdobné parametry jako `read.table()`, tedy např.:

```
write.table(lide, "D:/rko/lide.csv", sep=';', row.names=F)
```

Používáte-li delší dobu proměnné z jednoho datového rámce, může být vhodné použít funkci `attach()`, která umožní přistupovat k proměnným datového rámce přímo. Funkce `detach()` vše uvede do původního stavu.

```
attach(lide)
jmena[vyska > 2]
jmena[(vaha >= 80) & (vaha <= 90)]
detach(lide)
```

(Ukázka zjišťuje jména lidí vyšších než dva metry a s váhou mezi 80 a 90 kilogramy včetně.)

Základní informace a charakteristiky

Informace o datových rámcích a jejich jednotlivých proměnných a jejich základní statistické charakteristiky lze získat pomocí následujících příkazů:

`names(lide)`	Zjištění (změna) jmen proměnných v datovém rámci.
`summary(lide)`	Základní charakteristiky jednotlivých proměnných.
`min(lide$vek)`	Minimální hodnota, které nabývá proměnná.
`max(lide$vek)`	Maximální hodnota, které nabývá proměnná.
`range(lide$vek)`	Vrací minimum a maximum.
`mean(lide$vek)`	Aritmetický průměr hodnot.
`median(lide$vek)`	Výběrový medián ze zadaných hodnot.

`quantile(lide$vek)`	Zjistí výběrové kvartily.
`quantile(lide$vek,probs=c(.1,.9))`	Vypočítá zadané kvantily.
`var(lide$vaha)`	Vypočítá rozptyl zadaných hodnot.
`sd(lide$vaha)`	Směrodatná odchylka.
`cov(lide$vaha, lide$vyska)`	Vypočítá kovarianci.
`cor(lide$vaha, lide$vyska)`	Vypočítá koeficient korelace.

Kategoriální veličiny

Často je potřeba explicitně říci, která veličina má být považována za kategoriální. Vytvoříme-li například veličinu `lide$pohlavi <- c('M','Z','M','M')`, víme, že tato veličina je kategoriální a nabývá pouze dvou hodnot. Tento fakt ale musíme přímo specifikovat. To lze příkazem:

```
lide$pohlavi <- factor(lide$pohlavi, labels=c('muz','zena'))
```

Nyní, když si necháte vypsat vektor `lide$pohlavi`, zjistíte, že kromě jednotlivých položek vektoru je ve výpisu již i seznam hodnot, kterých proměnná nabývá:

```
[1] muz  zena muz  muz
Levels: muz zena
```

Konvertovat spojitou proměnnou v diskrétní lze pomocí příkazu `cut()`:

```
lide$nad30 <- cut(lide$vek, breaks = c(0,30,100))
```

V proměnné `lide$nad30` je

```
[1] (30,100] (0,30]   (0,30]   (30,100]
Levels: (0,30] (30,100]
```

Předcházejícím výpisem jsme získali kategorizované hodnoty vektoru a zároveň úrovně, kterých veličina nabývá. Pokud chceme zjistit pouze úrovně kategoriální

veličiny, použijeme příkaz `levels(lide$nad30)`, ale můžeme je tímto příkazem i změnit:
`levels(lide$nad30)=c('mladsi','starsi')`

Vytvořit vektor hodnot kategoriální veličiny lze i pomocí příkazu `gl()`, například:
`gl(2,3,12,labels = c('A','N'))`

Chceme tedy dvě hodnoty (A a N), které se mají vždy třikrát zopakovat a pak použít následující, celé opakovat tolikrát, aby vektor měl délku dvanáct, výsledkem bude:

```
 [1] A A A N N N A A A N N N
Levels: A N
```

Tabulky četností

Zobrazení absolutních a relativních četností umožňují příkazy `table()` a `prop.table()`, pokud vám nevyhovuje způsob zápisu vícerozměrných tabulek, můžete vyzkoušet příkaz `ftable()`.

```
attach(lide)
tabulka <- table(pohlavi,nad30)
tabulka
reltab <- prop.table(tabulka)
detach(lide)
```

Podmíněné četnosti lze vypočítat pomocí `prop.table(tabulka,1)`, což jsou podmíněné četnosti v závislosti na pohlaví (nálepka pohlaví). Zatímco `prop.table(tabulka,2)` nám vypočte podmíněné četnosti obou pohlaví, pokud známe věkovou kategorii (nálepkou je veličina `nad30`).

Marginální četnosti získáme z již vypočtených absolutních nebo relativních četností pomocí příkazu `margin.table(tabulka)`, který sečte všechny hodnoty v tabulce. Pokud přidáme další parametr, získáme součty vzhledem k zadanému indexu, tedy `margin.table(tabulka,1)` zjistí četnosti pro obě pohlaví, atd.

Kumulativní četnosti lze počítat pomocí funkce `cumsum()`.

Obecně lze pracovat s tabulkami pomocí příkazu `apply()`, který umožňuje aplikovat na jednotlivé dimenze matice různé funkce (viz též níže u funkce `tapply()`). Chceme-li provést sčítání po řádcích nebo maxima ve sloupcích, použijeme:

```
apply(tabulka, 1, sum)
apply(tabulka, 2, max)
```

Podmíněné charakteristiky

Charakteristiky kvantitativní proměnné podmíněné hodnotou kvalitativního znaku lze získat pomocí funkce `tapply()`. Následující zjišťuje minimální výšku ve dvou věkových kategoriích nebo průměrnou výšku, respektive dolní a horní kvartil výšek obou pohlaví.

```
tapply(lide$vyska, lide$nad30, min)
tapply(lide$vyska, lide$pohlavi, mean)
tapply(lide$vyska, lide$pohlavi, quantile, c(0.25,0.75))
```

Kromě toho lze jako třetí argument použít i maximum `max`, rozptyl `var`, směrodatnou odchylku `sd`, medián `median`, součet hodnot `sum`, setřídění `sort` a další funkce.

Logické operátory

Logické výrazy nabývají jen dvou hodnot: `TRUE` nebo `FALSE`. Jsou důležité například pro výběr dat, je-li splněna určitá podmínka.

```
x <- c(44,10,15,24,32,12,51)
x[x > 20]
subset(x, x <= 30)
```

Často potřebujeme spojit dohromady více podmínek, chceme-li například jména všech lidí, kteří měří alespoň 1.7 m a méně než 1.8 m nebo jména těch, kteří váží do 60 kg včetně nebo nad 90 kg, poslední dvě konstrukce vypisují věk všech lidí kromě Chrise. Všimněte si symbolu `==`, který je porovnáním dvou hodnot, nikoliv přiřazením jako jednoduché rovnítko.

```
lide$jmena[(lide$vyska > 1.7) & (lide$vyska < 1.8)]
lide$jmena[(lide$vaha <= 60) | (lide$vaha > 90)]
lide$vek[!(lide$jmena == 'Chris')]
lide$vek[lide$jmena != 'Chris']
```

Grafy

K vykreslování absolutních a relativních četností kategoriálních veličin můžete použít příkazy:

```
pie(table(lide$pohlavi))
barplot(table(lide$pohlavi))
barplot(prop.table(table(lide$pohlavi)))
```

K zobrazení spojitých veličin lze využít krabičkový diagram nebo je kategorizovat a vykreslit histogram:

```
boxplot(lide$vyska)
hist(lide$vek)
hist(lide$vek, breaks = c(0,20,40,60))
hist(lide$vek, breaks = c(0,20,40,60), freq=FALSE)
```

Přičemž druhý z histogramů má zvolené dělící body, nejsou vypočítány automaticky. V prvních dvou jsou zobrazeny absolutní četnosti, zatímco třetí zobrazuje empirickou hustotu.

Ke zobrazení vztahu mezi dvěma kvantitativními veličinami lze použít funkci `plot()`, která vykreslí mimo jiné scatterplot, ale můžete ji využít k vykreslení funkční závislosti:

```
plot(lide$vyska,lide$vaha)
plot(x,log(x+1),type='l')
```

Pokud chceme funkci plot využít k vykreslení pravděpodobnostní funkce, použijeme příkaz `plot(,type='h')`, k vykreslení distribuční a kvantilové funkce `plot(,type='s')`.

K vykreslování normálního QQ-grafu lze použít funkci `qqnorm()`.

K vykreslení více grafů do jednoho obrázku lze použít funkci `par`, zde se uvádí do hodnoty parametru, v kolika řádcích a v kolika sloupcích mají být grafy uspořádány. Všechny funkce pro vykreslování grafů mají samozřejmě mnoho dalších parametrů. Některé zde uvedeme v několika ukázkách:

```
par(mfrow=c(1,3))
pie(table(lide$pohlavi), col=c("yellow","magenta"))
xx <- prop.table(table(lide$pohlavi))
lab <- sprintf("%s: %s%%", levels(lide$pohlavi), xx*100)
```

```
pie(xx, labels = lab, col=c("red","cyan"))
barplot(xx, legend = levels(lide$pohlavi), col=c('red','blue'))

par(mfrow=c(2,2))
hist(lide$vek, breaks = c(0,20,40,60), xlab = 'Vek [roky]',
     ylab = 'Absolutni cetnost', main = 'Vekove kategorie',
     col = c('blue','orange','green'), border = 'gray')
boxplot(lide$vyska, ylab = "vyska [m]", main = "Boxplot vysky")
abline(h=2, col = "magenta")
x <- 0:10
plot(x,log(x+1),type='l',col="brown")
abline(h = 1, v = exp(1) - 1, col = "pink")
prumer <- mean(lide$vyska)
odchylka <- sd(lide$vyska)
qqnorm(lide$vyska, main="QQ-graf pro vysku",
    xlab="Teoreticke kvantily N(0,1)", ylab="Vyberove kvantily")
abline(a=prumer, b=odchylka, col="blue")
```

Ukládání obrázků

Jednoduchý způsob je kliknout na již vykresleném obrázku (grafu) pravým tlačítkem a zvolit v menu, které se objeví. Lze obrázek kopírovat do schránky jako bitmapu nebo jako metafile a poté obrázek vložit například v Malování. Můžete jej přímo jako metafile nebo postscript uložit nebo vytisknout (pokud máte nainstalovaný PDFCreator, můžete takto vytvořit i postscript).

Kromě toho existuje postavený přímo na příkazech jazyka R. K ukládání vektorových obrázků slouží příkazy `pdf()` a `postscript()`, je potřeba zadat jméno souboru a rozměry obrázku v palcích.

```
pdf(file='D:/obr1.pdf', width = 5, height = 5)
xx <- prop.table(table(lide$pohlavi))
lab <- sprintf("%s: %s%%", levels(lide$pohlavi), xx*100)
pie(xx, labels = lab, col=c("red","cyan"))
dev.off()
```

Příkaz `dev.off()` uzavírá a ukončuje ukládání do souboru. Místo příkazu `pdf()` lze použít: `postscript(file='D:/obr2.ps', width = 5, height = 5)`. Podobně se k ukládání bitmapových obrázků používá příkazů `jpeg()`, `bmp()` a `png()`. U bitmapových obrázků se velikost udává v bodech.

Náhodné výběry a rozdělení

Náhodný výběr 7 ze 49 hodnot provedeme pomocí funkce:
`sample(1:49,7)`
Kromě toho lze pomocí parametru `replace=T` zvolit výběr s opakováním a pomocí `prob=` zadat vektor pravděpodobností jednotlivých hodnot:
`sample(c(0,1),100,replace=T,prob=c(0.1,0.9))`

Práce s rozděleními pravděpodobnosti se realizuje vždy pomocí čtveřice příkazů. Příkaz začínající na `d` označuje hustotu pravděpodobnosti nebo pravděpodobnostní funkci. Příkaz začínající na `p` je distribuční funkce, na `q` začíná kvantilová funkce a konečně písmenem `r` se označují generátory čísel s příslušným rozdělením. Je vždy potřeba zadat parametry daného rozdělení. Tedy například rovnoměrné rozdělení:

```
x <- seq(-2,6,0.01) #pomocný vektor
plot(x, punif(x,0,4),type='l') # graf distribuční funkce
plot(x, dunif(x,0,4),type='l') # graf hustoty pravděpodobnosti
q <- seq(0,1,0.1)
plot(q, qunif(q,0,4),type='l') # kvantilová funkce
runif(10,0,4) # deset čísel z daného rozdělení
```

Obdobně to funguje pro normální rozdělení `pnorm()`, `dnorm()`, `qnorm()` a `rnorm()`, zde parametry jsou střední hodnota a rozptyl. Binomické rozdělení má příkazy `pbinom()`, `dbinom()`, `qbinom()` a `rbinom()`:

```
x <- 0:10
setiny <- seq(0,1,0.01)
par(mfrow=c(2,2))
plot(x, dbinom(x,10,1/4), main="Pravdep. fce", type="h")
plot(x, pbinom(x,10,1/4), main="Distrib. fce", type="s")
plot(setiny,qbinom(setiny,10,1/4),main="Kvantilova fce",type="s",)
```

Kde 10 byl počet pokusů a 1/4 je pravděpodobnost úspěchu v každém pokusu.

Příslušnou čtveřici funkcí naleznete samozřejmě pro mnoho dalších rozdělení, ze spojitých například Studentovo t-rozdělení (`*t()`), pro F-rozdělení (`*f()`), pro Chi-kvadrát rozdělení (`*chisq()`), exponenciální (`*exp()`) a další, z diskrétních třeba: geometrické (`*geom()`), multinomické (`*multinom()`) nebo Poissonovo (`*pois()`). Kde hvězdička je místo čtveřice písmen `d`, `p`, `q` a `r`.

Funkce balíčku vsePackage

Abychom funkce tohoto balíčku uvedli do provozu, je potřeba balíček nahrát, pokud jste jej správně instalovali (viz výše), stačí jej nahrát příkazem `library(vsePackage)`. K dispozici jsou funkce počítající bodové a intervalové odhady:

```
estim.mean(lide$vyska, conf.level=0.95)
estim.var(lide$vyska, conf.level=0.99)
```

U obou příkazů lze zajistit výpočet jednostranných intervalů pomocí parametru `type = 'less'` (levostranný) nebo `type = 'greater'` (pravostranný). Kromě toho jsou v tomto balíčku k dispozici některé statistické testy a množství dalších funkcí.

Statistické testy

Jazyk R a přídavný balík `vsePackage` dává možnost používat (kromě jiných) následující testy:

Normalita:
Shapirův-Wilkův test:

```
shapiro.test(x)
```

Jednovýběrové, parametrické:
Asymptotický test o střední hodnotě (vsePackage):

```
asymp.mean.test(x, mu =, alternative =, conf.level =)
```

T-test o střední hodnotě normálního rozdělení:

```
t.test(x, mu=, alternative =, conf.level =)
```

Testy o směrodatné odchylce a rozptylu normálního rozdělení (vsePackage):

```
onesample.var.test(x, sd =, alternative =, conf.level =)
onesample.var.test(x, var =, alternative =, conf.level =)
```

Jednovýběrové, neparametrické:
Test o poloze spojitého rozdělení:

```
wilcox.test(x, mu =, alternative =, conf.level =)
```

Dvouvýběrové, parametrické:
F-test na porovnání rozptylů (normalita obou výběrů):

```
var.test(x, y, ratio =)
var.test(y factor, ratio =)
```

Dvouvýběrový T-test, shodné rozptyly (normalita obou výběrů):

```
t.test(x, y, mu =, var.equal = T)
t.test(y factor, mu =, var.equal = T)
```

Dvouvýběrový T-test (zobecněný, Welch), různé rozptyly (normalita obou výběrů):

```
t.test(x, y, mu =, var.equal = F)
t.test(y factor, mu =, var.equal = F)
```

Dvouvýběrové, neparametrické:
Dvouvýběrový Wilcoxonův test (spojité výběry):

```
wilcox.test(x, y, mu =)
wilcox.test(y factor, mu =)
```

Kolmogorovův-Smirnovův test:

```
ks.test(x, y)
```

Párové, parametrické a neparametrické:
Párový t-test (normalita rozdílu párů):

```
t.test(x, y, mu =, paired = T)
```

Párový Wilcoxonův test (spojitost a přibližná symetrie rozdílu párů):

```
wilcox.test(x, y, mu =, paired = T)
```

Obecně lze oboustranný test zvolit pomocí `alternative='two.sided'`, levostranný pomocí volby `alternative='greater'` a konečně pravostranný volbou `alternative='less'`. U dvouvýběrových testů můžeme buď to zadávat dva vektory s výběry nebo použít zápis `y~factor`, kde se pomocí faktorizované (viz výše) kategoriální veličiny `factor` se dvěma hodnotami rozděluje do dvou výběrů vektor `y`.

Ošetření chybějících hodnot

Chybějící hodnoty lze u většiny funkcí ošetřit pomocí parametru `na.rm = T`, pokud potřebujeme kompletní páry například u výpočtu koeficientu korelace, použijeme `cov(x, y, use = 'pairwise.complete.obs')`. Nebo můžeme z vektoru `y` odstranit chybějící hodnoty pomocí funkce `is.na()` a negace: `bez.na <- y[!is.na(y)]`.