Základy práce s R

Arnošt Komárek

8. října 2007

R je volně šiřitelná verze systému S, který byl vyvinut Johnem Chambersem a kolegy v Bell Laboratories. Komerční a dle mého názoru v mnoha ohledech horší variantou systému S (a to nejen s ohledem na cenu) je program S-PLUS dodávaný firmou *Insightful Corporation*. Na tomto místě uvedu několik základních informací o Rku, zejména kde ho získat jak ho nainstalovat a jak provést některé základní obecné operace.

1 Stažení a instalace

Program R lze možno zdarma stáhnout a dále distribuovat (při dodržení podmínek GNU GPL licence) z libovolného zrcadla dostupného přes `http://www.R-project.org`. V tomto textu popíši postup pro uživatele operačního systému Wokna. R je samozřejmě dostupné i pro další operační systémy jako Linux nebo MacOS. Předpokládám však, že uživatelé ne-Wokenních operační systémů jsou dostatečně zkušení v ovládání počítače, že si snadno poradí sami. Linux uživatelům vřele doporučuji nestahovat předkompilovanou verzi R, ale stáhnout si přímo zdrojový kód a R si zkompilovat pomocí asi 3 příkazů svépomocí. Tak tedy postup pro Wokenáře:

1. Jděte na stránku `http://www.R-project.org`;
2. Klikněte na odkaz `CRAN` v *Download* sekci na levé straně stránky;
3. Zvolte si vhodné (ne příliš vzdálené zrcadlo), osobně mám dobré zkušenosti se zrcadlem rakouským (Austria);
4. V sekci *Precompiled Binary Distributions* klikněte na `Windows (95 and later)`;
5. Pro stažení základního R klikněte na odkaz `base`;
6. Stáhněte soubor `R-2.6.0-win32.exe` (číslo 2.6.0 odráží číslo aktuální verze a s časem se mění);
7. Na svém počítači klikněte na `R-2.6.0-win32.exe` a postupujte podle instrukcí (odbobně jako při instalaci jakéhokoliv jiného programu);
8. Na ploše počítače se patrně objeví ikona znázorňující modré písmeno R.

1.1 Přídavné balíky

Do R lze podle potřeby doinstalovat též některý(é) z asi 500 přídavných balíků (viz odkaz *Contributed packages* na `CRAN`). U počítače s přímým (a rozumně rychlým) připojením na Internet lze toto učinit pomocí příkazu `install.packages`, např.

```
> install.packages("Rcmdr")
```

nainstaluje balík nazvaný *Rcmdr* (a též všechny další balíky, na nichž je *Rcmdr* závislé).

1. Stiskněte `Enter` a měli byste vidět modré zprávy o průběhu instalace.
2. Pokud se Vám vše podařilo, příkaz

   ```
   > library(Rcmdr)
   ```

   by měl způsobit otevření jakéhosi dalšího okna.

2 Začínáme, kde najít pomoc či nápovědu?

Program R spustíte kliknutím na jeho ikonu. Na začátku každé práce doporučuji nastavit pracovní adresář tak, aby program věděl, kde brát např, požadovaná data anebo abyste Vy věděli kde se budou ukládat např. vytvořená grafy. Předpokládejme, že chceme mít pracovní adresář `C:\Moje\Veci`. Jako pracovní ho nastavíme pomocí

```
> setwd("C:/Moje/Veci")
```

Více o nastavení pracovního adresáře se dočtete v oddíle 4.4. K tomu, aby se Vám otevřelo okno, v němž lze v horní polovině editovat příkazy a v dolní polovině vidět výsledky, musíte si připojit přídavný balík *Rcmdr* (nejprve ho musíte nainstalovat – viz výše). Toto učiníte pomocí příkazu

```
> library(Rcmdr)
```

Dříve než začnete, doporučuji se alespoň zběžně podívat na dostupné příručky (v angličtině): klikněte na `Help` v horní čá sti spuštěného R (ve Woknech) a zvolte si oblí beny formá t (*Html help* nebo *Manuals (in PDF)*). Zejména příručka – *An Introduction to R* by neměla uniknout Vaší alespoň zběžné pozornosti.

Nápověda jak základního R tak přídavných balíků (packages) je (v závislosti na snaze autora balíku) obvykle na poměrně vysoké úrovni. K použití HTML nápově dy stačí kliknout na `Help` lištu (ve Woknech), zvolit *Html help* a poté kliknout na `Packages` v sekci *Reference*. Tato akce Vám podá seznam všech nainstalovaných balíků (povšimněte si, ze mnohé balíky jsou naistalovány standardně se základním R). Nápovědu k jakémukoliv příkazu programu R získáte jeho vyhledáním v podsekci přísluš ného balíku. Přehled těch nejdůležitějších balíků následuje.

`base` základní příkazy programu R (matematické operace a funkce, práce s vektory a maticemi, manipulace s daty, ...);

`graphics` základní příkazy pro vytváření obrázků a grafických výstupů;

`stats` funkce k prová dění základních statistických výpočtů, testů, ...

Pokud víte jak se příkaz, pro nějž hledáte nápovědu jmenuje (např. `mean`), získáte nápovědu pomocí otazníku a názvu příkazu z příkazové řá dky R:

```
> ?mean
```

nebo pomocí pří kazu `help`:

```
> help(mean)
```

2.1 Několik základních operátorů a příkazů

- Operátor přiřazení má tvar `<-` (menšítko a pomlka) anebo = (rovnítko).

```
> x <- 10
> x

[1] 10

> x = 10
> x

[1] 10
```

- Vektor vytvoříme pomocí funkce `c` (combine).

```
> x <- c(1, 2, 3, 4, 5)
> x

[1] 1 2 3 4 5
```

- Vektor mající tvar aritmetické posloupnosti vytvoříme pomocí funkce `seq` (sequence), vektor ve tvaru aritmetické posloupnosti s krokem 1 též pomocí operátoru `:` (dvojtečka).

```
> x <- seq(1, 5, by = 1)
> x

[1] 1 2 3 4 5

> x <- seq(1, 5, length = 5)
> x

[1] 1 2 3 4 5

> x <- 1:5
> x

[1] 1 2 3 4 5
```

- Matici vytvoříme pomocí funkce `matrix`. Data do matice se vyplňují standardně po sloupcích. Chceme-li je vyplňovat po řádcích, musíme nastavit argument `byrow` funkce `matrix` na `TRUE`.

```
> x <- matrix(seq(1, 11, by = 2), nrow = 2, ncol = 3)
> x
```

```
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    3    7   11
```

```
> x <- matrix(seq(1, 11, by = 2), nrow = 2, ncol = 3, byrow = TRUE)
> x
```

```
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    7    9   11
```

3 Načtení dat

K praktické analýze vlastních dat je potřeba tato nějakým způsobem Rku dodat. Program je schopem číst z pevného disku (či jiného média) data uložená v mnoha různých formátech. Vzhledem k tomu, že většina programů (SAS, Excel, ...) umožňuje export dat do obyčejného ASCII (txt) souboru, probereme zde detailněji načítání ASCII souborů ve víře, že pomocí exportního mezikroku bude čtenář schopen načítat soubory uložené v téměř jakémkoliv jiném formátu. Pro přímé načítání dat uložených v jiném než ASCII formátu odkazujeme na *R Data Import/Export* příručku dostupnou po kliknutí na `Help` lištu (zde je mimo jiné popsán podrobně import dat z Excelu).

Základním příkazem pro čtení obdélníkových dat je příkaz `read.table`. Uvedeme několik příkladů jeho použití.

3.1 Datový soubor s daty oddělenými mezerou

Představme si, že máme následující datový soubor.

Soubor `kamaradi.dat`:

```
Tohle je soubor vytvoreny v den vyroci objevu parniho stroje
Autor souboru je neznamy a nenese zadnou odpovednost za data
v souboru obsazena
id  pohlavi  vyska  vaha
1   muz      173    75
2   zena     165    50
3   muz      185    80
4   muz      181    100
5   zena     158    67
```

Data načteme a poté prohlédneme pomocí:

```
> kamaradi <- read.table("kamaradi.dat", skip = 3, header = TRUE)
> kamaradi
```

```
  id pohlavi vyska vaha
1  1     muz   173   75
2  2    zena   165   50
```

```
3  3     muz   185   80
4  4     muz   181  100
5  5    zena   158   67
```

Argument `skip` zde roven hodnotě 3 udává, že na začátku souboru jsou 3 řádky, které neobsahují žádnou datovou informaci. Argument `header`, zde roven `TRUE` na druhou stranu říká, že datový soubor kromě vlastních dat obsahuje též informaci o jménech jednotlivých proměnných na prvním řádku (po přeskočení tří řádků s komentářem).

3.2 Datový soubor s jiným oddělovačem než mezerou

Představme si nyní, že výše uvedený soubor má poněkud jiný tvar (datové položky jsou odděleny středníky namísto mezer).

Soubor `kamaradi2.dat`:

```
Tohle je soubor vytvoreny v den vyroci objevu parniho stroje
Autor souboru je neznamy a nenese zadnou odpovednost za data
v souboru obsazena
id;  pohlavi;  vyska;  vaha
1;   muz;      173;    75
2;   zena;     165;    50
3;   muz;      185;    80
4;   muz;      181;    100
5;   zena;     158;    67
```

Soubor načteme a data poté prohlédneme pomocí:

```
> kamaradi2 <- read.table("kamaradi2.dat", skip = 3, header = TRUE,
+     sep = ";")
> kamaradi2
```

```
  id  pohlavi vyska vaha
1  1     muz   173   75
2  2    zena   165   50
3  3     muz   185   80
4  4     muz   181  100
5  5    zena   158   67
```

Fakt, že data jsou oddělena středníky jsme sdělili funkci `read.table` pomocí argumentu `sep`. Jsou-li vaše data oddělena jiným znakem (znaky), stačí změnit hodnotu tohoto argumentu.

3.3 Datový soubor bez jmen proměnných

V některých situacích není informace o názvech jednotlivých proměnných součástí datového souboru a musíme si tedy jednotlivé proměnné pojmenovat sami jako v následujícím případě.

Soubor `kamaradi3.dat`:

```
Tohle je soubor vytvoreny v den vyroci objevu parniho stroje
Autor souboru je neznamy a nenese zadnou odpovednost za data
v souboru obsazena
1:  muz:   173:  75
2:  zena:  165:  50
3:  muz:   185:  80
4:  muz:   181:  100
5:  zena:  158:  67
```

Soubor načteme a data poté prohlédneme pomocí:

```
> kamaradi3 <- read.table("kamaradi3.dat", skip = 3, header = FALSE,
+     sep = ":")
> colnames(kamaradi3) <- c("id", "pohlavi", "vyska", "vaha")
> kamaradi3

  id  pohlavi vyska vaha
1  1      muz   173   75
2  2     zena   165   50
3  3      muz   185   80
4  4      muz   181  100
5  5     zena   158   67
```

Argument `header` je nyní roven `FALSE`, pomocí funkce `colnames` pojmenujeme jednotlivé sloupce dat. Povšimněte si též, že jednotlivá data byla nyní v datovém souboru oddělena dvojtečkou.

3.4 Data s chybějícími pozorováními

V reálném životě se nám ne vždy podaří získat všechna data, která jsme si předsevzali získat. Datový soubor poté obsahuje chybějící data, obvykle nějak označená. V následujícím souboru se nepodařilo získat výšku jedné a hmotnost dvou osob.

Soubor `kamaradi4.dat`:

```
Tohle je soubor vytvoreny v den vyroci objevu parniho stroje
Autor souboru je neznamy a nenese zadnou odpovednost za data
v souboru obsazena
id  pohlavi  vyska  vaha
1   muz      173    75
2   zena     165    xx
3   muz      xx     80
4   muz      181    xx
5   zena     158    67
```

Data načteme a poté prohlédneme pomocí:

```
> kamaradi4 <- read.table("kamaradi4.dat", skip = 3, header = TRUE,
+     na.strings = "xx")
```

```
> kamaradi4
```

```
  id pohlavi vyska vaha
1  1     muz   173   75
2  2    zena   165   NA
3  3     muz    NA   80
4  4     muz   181   NA
5  5    zena   158   67
```

Způsob, jakým jsou chybějící pozorování kódována v datovém souboru jsme funkci `read.table` sdělili pomocí jejího argumentu `na.strings` (zde kombinace dvou písmen ‚x'). Povšimněme si, že R kóduje chybějící pozorovaní pomocí kombinace písmen ‚NA' *(not available)*.

3.5 Data s chybějícími pozorováními podruhé

Při exportu dat z jiných systémů je často (téměř) nemožné reprezentovat chybějící pozorování jinak než mezerou. V takovém případě musíme jednotlivé položky dat oddělit něčím jiným než standardní mezerou (např. středníkem) a poté je snadno načteme jako v následujícím případě.

Soubor `kamaradi5.dat`:

```
Tohle je soubor vytvoreny v den vyroci objevu parniho stroje
Autor souboru je neznamy a nenese zadnou odpovednost za data
v souboru obsazena
id;  pohlavi;  vyska;  vaha
1;   muz;      173;    75
2;   zena;     165;
3;   muz;      ;       80
4;   muz;      181;
5;   zena;     158;    67
```

Soubor načteme a data poté prohlédneme pomocí:

```
> kamaradi5 <- read.table("kamaradi5.dat", skip = 3, header = TRUE,
+     sep = ";", na.strings = "")
> kamaradi5
```

```
  id  pohlavi vyska vaha
1  1      muz   173   75
2  2     zena   165   NA
3  3      muz    NA   80
4  4      muz   181   NA
5  5     zena   158   67
```

3.6 Datový soubor se sloupci pevné šířky

V praxi je možné se setkat též se soubory, která mají data uložena ve sloupcích pevné (známé) šířky (fixed width formatted). V takovémto formátu jsou často uložena data obsahující větší množství

textové informace. Soubory tohoto typu načítáme pomocí funkce `read.fwf` z balíku `utils` (tento balík není nutné explicitně připojovat k R, nebot' je připojen automaticky).

Soubor `kamaradi6.dat`:

```
Tohle je soubor vytvoreny v den vyroci objevu parniho stroje
Autor souboru je neznamy a nenese zadnou odpovednost za data
v souboru obsazena
1  muz   173  75
2  zena  165  *
3  muz   *    80
4  muz   181  *
5  zena  158  67
```

Soubor načteme a data poté prohlédneme pomocí:

```
> kamaradi6 <- read.fwf("kamaradi6.dat", skip = 3, header = FALSE,
+     widths = c(3, 8, 6, 5), sep = "", na.strings = "*")
> colnames(kamaradi6) <- c("id", "pohlavi", "vyska", "vaha")
> kamaradi6
```

```
  id pohlavi vyska vaha
1  1     muz   173   75
2  2    zena   165   NA
3  3     muz    NA   80
4  4     muz   181   NA
5  5    zena   158   67
```

Šířka jednotlivých sloupců byla udána pomocí argumentu `widths` funkce `read.fwf`.

3.7 Načítání nepravidelných či rozsáhlých dat

Vzhledem k tomu, že funkce `read.table` při načítání dat převádí nejprve celý soubor do jednoho dlouhého řetězce a poté tento řetězec kouskuje na jednotlivá data, může se stát, že velké datové soubory (několik desítek tisíc pozorování) narazí při načítání na nedostatek operační paměti vašeho počítače. V takovém případě je lepší použít funkci `scan` pro načtení dat. Tato funkce čte data sekvenčně tak jak jsou uložena v souboru a je tedy vhodná též k načítání souborů s nepravidelně uspořádanými, t.j. neobdélníkovými daty. Pro ilustraci zde uvedeme způsob načtení souboru `kamaradi5.dat` pomocí `scan`. Pro pokročilejší použití této funkce odkazuji na její nápovědu.

```
> kamaradi5a <- as.data.frame(scan("kamaradi5.dat", skip = 4, what = list(numeric(0),
+     character(0), numeric(0), numeric(0)), sep = ";", na.strings = ""))
> colnames(kamaradi5a) <- scan("kamaradi5.dat", skip = 3, nlines = 1,
+     what = character(0), sep = ";")
> kamaradi5a
```

```
  id    pohlavi   vyska   vaha
1  1        muz     173     75
```

```
2  2      zena    165     NA
3  3       muz     NA     80
4  4       muz    181     NA
5  5      zena    158     67
```

Povšimněte si, že funkci `scan` musíme sdělit pomocí argumentu `what` jaký typ dat budeme načítat. Dále si povšimněte jak jsme funkci `scan` zkombinovali s funkcí `as.data.frame` a poté pomocí dalšího volání funkce `scan` načetli jména proměnných.

4 Ukládání obrázků do souboru

Abychom vytvořené obrázky mohli zahrnout do vlastních dokumentů (zpráv o analýze apod.) musíme být schopni obrázky uložit v souboru vhodného formátu. Nejdůležitějšími formáty, které R ko nabízí jsou

- postscript (`.ps`)
- portable document format (`.pdf`)
- bitové mapy typu `jpg` a `png`

4.1 Postscript

Obrázek ve formátu `.ps` uložíme pomocí příkazu `postcript`. Nejdůležitějšími argumenty tohoto příkazu jsou

file jméno souboru, do kterého chceme obrázek uložit;

width, height šířka a výška obrázku v palcích (1inch = 2,54cm);

horizontal `TRUE` či `FALSE` udávající jakou orientaci pro obrázek chceme. Ležatá orientace (landscape), pokud `horizontal=TRUE`, stojatá orientace (portrait), pokud `horizontal=FALSE`;

Například hypotetický krabicový graf uložíme do souboru nazvaném `mujboxplot.ps` následujícím způsobem (příkaz `dev.off()` sděluje R ku, že už jsme skončili s kreslením a výsledek má být uložen).

```
> postscript(file = "mujboxplot.ps", width = 6.5, height = 5, horizontal = FALSE)
> par(mfrow = c(1, 1), bty = "n")
> boxplot(spotr.mesto ~ druh, data = Auta93, main = "Spotreba ve meste")
> dev.off()
```

4.2 Portable document format

Obrázek ve formátu `.pdf` uložíme pomocí příkazu `pdf`. Nejdůležitějšími argumenty tohoto příkazu jsou

file jméno souboru, do kterého chceme obrázek uložit;

width, height šířka a výška obrázku v palcích (1inch = 2,54cm).

Hypotetický krabicový graf uložíme do souboru nazvaném `mujboxplot.pdf` následujícím způsobem.

```
> pdf(file = "mujboxplot.pdf", width = 6.5, height = 5)
> par(mfrow = c(1, 1), bty = "n")
> boxplot(spotr.mesto ~ druh, data = Auta93, main = "Spotreba ve meste")
> dev.off()
```

4.3 Bitové mapy

Soubory typu `.jpg`, respektive `.png` vytvoříme příkazy `jpeg`, respektive `png`. Jejich nejdůležitějšími argumenty jsou

filename jméno souboru, do kterého chceme obrázek uložit;

width, height šířka a výška obrázku v pixelech;

quality (pouze pro `jpeg`) číslo mezi 0 a 100 udávající kvalitu bitové mapy. Nižší hodnoty vedou k větší kompresi a horší kvalitě.

Hypotetický krabicový graf uložíme do souboru nazvaného `mujboxplot.jpg`, resp. `mujboxplot.png` následujícími způsoby.

```
> jpeg(filename = "mujboxplot.jpg", width = 1000, height = 700, quality = 75)
> par(mfrow = c(1, 1), bty = "n")
> boxplot(spotr.mesto ~ druh, data = Auta93, main = "Spotreba ve meste")
> dev.off()
```

```
> png(filename = "mujboxplot.png", width = 1000, height = 700)
> par(mfrow = c(1, 1), bty = "n")
> boxplot(spotr.mesto ~ druh, data = Auta93, main = "Spotreba ve meste")
> dev.off()
```

4.4 Kde najdu uložený soubor?

Soubor naleznete v pracovním adresáři. Ale který to je? Na tuto otázku vám odpoví funkce `getwd` (get working directory):

```
> getwd()
```

```
[1] "/home/komari/teach/mff_2007/Rko"
```

Pokud se vám pracovní adresář nelíbí, můžete ho změnit příkazem `setwd` (set working directory):

```
> puvodni.dir <- getwd()
> setwd("/home/komari/teach/")
```

Byl adresář změněn?

```
> getwd()
```

```
[1] "/home/komari/teach"
```

Ano, byl. Ale já chci zpět můj původní adresář:

```
> setwd(puvodni.dir)
```

Poznámka pro uživatele Woken. Ve jménech adresářů používáme běžná a nikoliv zpětná lomítka, t.j. adresář `C:\komari\work` nastavíme jako pracovní příkazem:

```
> setwd("C:/komari/work")
```

5 Souhrnný přehled nejdůležitějších příkazů

5.1 Základní elementy

Příkazy

`ls()` nebo `objects()`	vypiš seznam objektů definovaných a dostupných na pracovní ploše
`rm(object)`	vymaž `object` z pracovní plochy
`search()`	vypiš co všechno je prohledáváno a v jakém pořadí, když se hledá nějaký objekt

Jména proměnných

Kombinace písmen, číslic a teček. Nesmí začínat číslicí. Nedoporučuje se začínat jméno proměnné tečkou. Rozlišují se velká a malá písmena, tj. objekt pojmenovaný `krabicka` je něco jiného než objekt pojmenovaný `Krabicka`.

Přiřazovací příkazy

$<-$ nebo $=$	přiřad' hodnotu proměnné
$->$	přiřazení „doprava“
$<<-$	globální přiřazení (ve funkcích)

5.2 Operátory

Aritmetické operátory

$+$	sčítání
$-$	odčítání
$*$	násobení
$/$	dělení
$\wedge$	umocňování
%/%	celočíselné dělení (div)
%%	zbytek po celočíselném dělení (mod)

Logické operátory a operátory vztahu

Výsledkem těchto operátorů je vždy logická hodnota `TRUE` anebo `FALSE`.

==	je rovno?
!=	není rovno?
<	je menší než?
>	je větší než?
<=	je menší než nebo rovno?
>=	je větší než nebo rovno?
`is.na(x)`	je `x` chybějící hodnota?
&	logické A SOUČASNĚ (AND)
|	logické NEBO (OR)
!	logická negace (NOT)

5.3 Vektory a datové typy

Generování vektorů s nějakou strukturou

`numeric(25)`	vektor o 25 nulách
`character(25)`	vektor s 25 prázdnými znaky, tj. `""`
`logical(25)`	logický vektor s 25 elementy rovnými `FALSE`
`seq(-4, 4, 0.1)`	vektor aritmetické posloupnosti -4, $-3{,}9$, $-3{,}8, \ldots, 3{,}9, 4$
`1:10`	vektor aritmetické posloupnosti 1, 2, ..., 10, to samé jako příkaz `seq(1, 10, 1)`
`c(5, 7, 9, 13, 1:5)`	vytvoř vektor spojením složek uvnitř `c` (*concatenation*), zde vektor 5, 7, 9, 13, 1, 2, 3, 4, 5
`rep(1, 10)`	vektor, kde se 1 opakuje 10×
`gl(3, 2, 12)`	faktorový vektor o 3 úrovních, opakuj každou úroveň v blocích o velikosti 2, a to až do celkové délky vektoru 12, zde tedy 1, 1, 2, 2, 3, 3, 1, 1, 2, 2, 3, 3

Přetypování vektorů

`as.numeric(x)`	přetypuj `x` na numerický vektor
`as.character(x)`	přetypuj `x` na znakový vektor
`as.logical(x)`	přetypuj `x` na logický vektor (obsahující pouze `TRUE` a `FALSE`)
`factor(x)`	vytvoř faktor (kategoriální veličinu) z `x`

5.4 Datové soubory (data frames)

`data.frame(height=c(165, 185), weight=c(90, 65))`	vytvoř data frame se dvěma pojmenovanými veličinami
`data.frame(height, weight)`	ulož dříve vytvořené vektory jako dva sloupce v data framu
`dfr$var`	vyber proměnnou (sloupec) `var` z data framu `dfr`
`attach(dfr)`	polož data frame `dfr` do vyhledávací cesty, k jednotlivým proměnným lze potom přistupovat i bez `$`
`detach(dfr)`	odstraň data frame z vyhledávací cesty

5.5 Numerické funkce

Matematické

`log(x)`	přirozený logaritmus x
`log10(x)`	dekadický logaritmus x
`exp(x)`	exponenciální funkce e^x
`sin(x)`	sinus x
`cos(x)`	kosinus x
`tan(x)`	tangens x
`asin(x)`	arcus-sinus x
`acos(x)`	arcus-kosinus x
`atan(x)`	arcus-tangens x
`mix(x)`	minimum z vektoru x
`min(x1, x2, ...)`	minimum z několika vektorů, výsledkem je jedno číslo
`max(x)`	maximum z vektoru x
`max(x1, x2, ...)`	maximum z několika vektorů, výsledkem je jedno číslo
`range(x)`	to samé jako `c(min(x), max(x))`
`pmin(x1, x2, ...)`	paralelní (po složkách) minimum z několika stejně dlouhých vektorů
`pmax(x1, x2, ...)`	paralelní (po složkách) maximum z několika stejně dlouhých vektorů
`length(x)`	počet složek vektoru
`sum(complete.cases(x))`	počet nechybějících složek ve vektoru

Statistické

`mean(x)`	aritmetický průměr
`sd(x)`	směrodatná odchylka
`var(x)`	rozptyl
`median(x)`	medián
`quantile(x, p)`	kvantily
`cor(x, y)`	korelace

5.6 Indexace/vybírání

`x[1]`	první element
`x[1:5]`	podvektor obsahující prvních pět elementů
`x[c(2,3,5,7,11)]`	podvektor obsahující 2., 3., 5., 7. a 11. element
`x[y <= 30]`	výběr podvektoru pomocí logického výrazu
`x[sex=="male"]`	výběr podvektoru pomocí faktorové proměnné
`i <- c(2,3,5,7,11); x[i]`	výběr podvektoru pomocí číselné proměnné
`l <- (y<=30); x[l]`	výběr podvektoru pomocí logické proměnné

Matice a datové soubory

`m[4,]`	čtvrtý řádek
`m[,3]`	třetí sloupec
`dfr[dfr$promenna<=30]`	částečný datový soubor
`subset(dfr, subset=(promenna<=30))`	to samé jako předcházející příkaz

5.7 Pravděpodobnostní rozdělení

Normální rozdělení

`dnorm(x)`	hustota $\mathcal{N}(0, 1)$
`pnorm(x)`	distribuční funkce $\mathcal{N}(0, 1)$, $\mathrm{P}(X \leq x)$
`qnorn(p)`	p-kvantil $\mathcal{N}(0, 1)$, x: $\mathrm{P}(X \leq x) = p$
`rnorm(n)`	n pseudonáhodných standardně normálně rozdělených hodnot

Diskrétní rozdělení – pravděpodobnostní funkce

`dbinom(x, n, p)`	binomické rozdělení s `n` pokusy a pravděpodobností úspěchu `p`
`dgeom(x, prob)`	geometrické rozdělení s pravděpodobností úspěchu `p`
`dnbinom(x, size, prob)`	negativně binomické rozdělení s pravděpodobností úspěchu `p` a počtem `size` úspěchů na které čekáme
`dhyper(x, m, n, k)`	hypergeometrické rozdělení (výběr bez vracení), kde `m` je počet bílých koulí v urně, `n` počet černých koulí v urně a `k` počet koulí tažených z urny
`dpois(x, lambda)`	Poissonovo rozdělení se střední hodnotou `lambda`
`dmultinom(x, size, prob)`	multinomické rozdělení
`dsignrank(x, n)`	rozdělení Wilcoxonovy jednovýběrové statistiky ve výběru o rozsahu `n`
`dwilcox(x, m, n)`	rozdělení Wilcoxonovy dvouvýběrové statistiky pro výběry o rozsahu `m` a `n`

Spojitá rozdělení s oborem hodnot $\mathbb{R}$ – hustoty

`dnorm(x, mean, sd)`	normální rozdělení se střední hodnotou `mean` a směrodatnou odchylkou `s`
`dt(x, df)`	Studentovo t rozdělení s `df` stupni volnosti
`dcauchy(x, location, scale)`	Cauchyho rozdělení (zobecnění t_1 rozdělení)
`dlogis(x, location, scale)`	logistické rozdělení

Spojitá rozdělení s oborem hodnot $\mathbb{R}^+$ – hustoty

`dexp(x, rate)`	exponenciální rozdělení se střední hodnotou `1/rate`
`df(x, n1, n2)`	Fisherovo-Snedeckorovo F rozdělení se stupni volnosti `n1` a `n2`
`dchisq(x, df)`	χ^2 rozdělení s `df` stupni volnosti
`dlnorm(x, mean, sd)`	log-normální rozdělení, tj. $\log(X) \sim \mathcal{N}(\texttt{mean}, \texttt{sd}^2)$
`dweibull(x, shape, scale)`	Weibullovo rozdělení
`dgamma(x, shape, rate)`	gamma rozdělení se střední hodnotou `shape/rate`

Spojitá rozdělení s oborem hodnot rovným intervalu v $\mathbb{R}$ – hustoty

`dunif(x, min, max)`	rovnoměrné rozdělení na intervalu (`min`, `max`)
`dbeta(x, a, b)`	beta rozdělení na intervalu (`a`, `b`)

Stejné značení jako u normálního rozdělení, tj. p-q-r, platí pro hustoty, kvantilové funkce a funkce generující pseudonáhodná čísla.

5.8 Standardní statistické metody

Kvantitativní (spojitá) odezva

`t.test`	jedno a dvouvýběrový t test
`pairwise.t.test`	párový t test, resp. mnohonásobné porovnávání
`cor.test`	test o korelačním koeficientu pro normálně rozdělená data
`var.test`	porovnání dvou rozptylů pro normálně rozdělená data (F test)
`lm(y ~ x)`	jednoduchá regrese `x` na `y`
`lm(y ~ f)`	je-li `f` faktor, jednoduchá analýza rozptylu (*one-way ANOVA*)
`lm(y ~ f1 + f2)`	jsou-li `f1` a `f2` faktory, analýza rozptylu se 2 faktory (*two-way ANOVA*)
`lm(y ~ f + x)`	je-li `f` faktor, analýza kovariance (*ANCOVA*)
`lm(y ~ x1 + x2 + x3)`	vícerozměrná regrese
`bartlett.test`	Bartlettův test na porovnání k rozptylů

Kvantitativní (spojitá) odezva – neparametrické metody

`wilcox.test`	jedno a dvouvýběrový Wilcoxonův test
`kruskal.test`	Kruskalův-Wallisův test (neparametrická jednoduchá ANOVA)
`friedman.test`	Friedmanova neparametrická ANOVA s dvěma faktory
`cor.test(method = "kendall")`	test o nulovosti Kendallova τ
`cor.test(method = "spearman")`	test o nulovosti Spearmanova ρ

Diskrétní odezva

`binom.test`	binomický test (včetně znaménkového testu)
`prop.test`	test porovnávající pravděpodobnosti úspěchu (*proportions*) ve dvou výběrech
`prop.trend.test`	test pro trend v pravděpodobnostech úspěchu
`fisher.test`	Fisherův exaktní faktoriálový test v malých kontingenčních tabulkách
`chisq.test`	χ^2 test nezávislosti v kontingenční tabulce
`glm(y ~ x1 + x2 + x3, binomial)`	logistická regrese
`glm(count ~ f1 + f2 + f3, poisson)`	poissonovská regrese (log-lineární model)