Grafická prezentace a numerické modelování geochemických dat

Krátký kurz jazyka R

Vojtěch Janoušek,

vojtech.janousek@geology.cz

CRAN: http://www.r-project.org

GCDkit: http://www.gcdkit.org

GCDkit Blog: http://blog.gcdkit.org/

Williams and Holland's Law:

If enough data is collected, anything may be proven by statistical methods

Shaw's Principle:

Build a system that even a fool can use, and only a fool will want to use it

Naeser's Law

You can make it foolproof, but you can't make it damnfoolproof

Software pro geochemické přepočty a grafy

CRAN:
http://www.r-project.org

Stránka tohoto kurzu:
http://petrol.natur.cuni.cz/~janousek/Rkurz/

Vybrané citace:

PETRELLI M., POLI G., PERUGINI D. & PECCERILLO A. 2005. PetroGraph: A new software to visualize, model, and present geochemical data in igneous petrology. *Geochemistry Geophysics Geosystems* **6:** 15 pp.

~~RICHARD L.R. 1995. MinPet: Mineralogical and petrological data processing system, version 2.02. MinPet Geological Software, Québec, Canada.~~

SIDDER G. B. 1994. Petro.calc.plot, Microsoft Excel macros to aid petrologic interpretation: *Computers & Geosciences,* **20:** 1041–1061.

STORMER J. C. & NICHOLLS J. 1978. XLFRAC: a program for the interactive testing of magmatic differentiation models. *Computers & Geosciences* **4:** 143–159.

SU Y. J., LANGMUIR C. H. & ASIMOW P. D. 2003. PetroPlot: A plotting and data management tool set for Microsoft Excel. *Geochemistry Geophysics Geosystems* **4:** 14 pp.

WANG X., MA W., GAO, S. & KE, L. 2008. GCDPlot: An extensible Microsoft Excel VBA program for geochemical discrimination diagrams. *Computers & Geosciences,* **34:** 1964–1969.

ZHOU J. & LI X. 2006. GeoPlot: An Excel VBA program for geochemical data plotting. Computers and Geosciences **32:** 554–560.

1.1 Spreadsheety (MS Excel, Quattro Pro …)

Výhody:

- Rozšířenost (každý to má)
- Nulové dodatečné náklady
- Snadnost ovládání (skoro každý s tím umí)

Nevýhody:

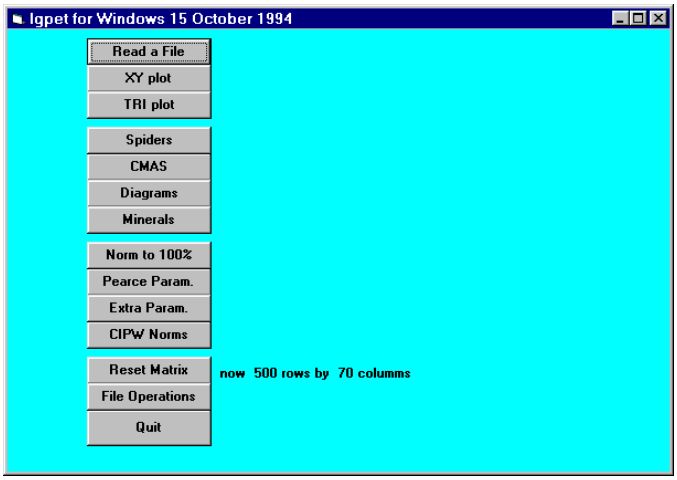
- Nedostupnost aplikací (až na výjimky – např. Sidder, 1994; Zhou & Li, 2006)
- Nepřehlednost + možnost narušení primárních dat
- Neefektivnost při opakovaném používání
- Nedostatečná kvalita grafického výstupu

1.2 Speciální software

1.2.1 MinPet

CITACE: Richard (1995) *http://www.minpet.com*
OS: Windows 3.1/95/98 (Visual Basic)
DISTRIBUCE: Komerční (CAN$ 1000)
VSTUP: DBF, ASCII
VÝSTUP: DBF, ASCII
GRAFIKA: WMF
MOŽNOSTI: Poměrně rozsáhlý program s nepříliš pochopitelným ovládáním a složitým importem dat. Silnou stránkou jsou krystalochemické přepočty minerálů (v současnosti asi bezkonkurenčně nejlepší na trhu). *MinPet* zahrnuje i řadu klasifikačních a geotektonických diagramů, plus spider diagramy. Pro spider diagramy umožňuje výběr celé řady normalizačních hodnot a dokonce vynášení polí ukazující rozptyl dat pro jednotlivé skupiny. Lze vynášet také uživatelské binární a ternární diagramy. Asi nejpokročilejší možnosti jednoduché statistiky (včetně korelačních koeficientů mezi jednotlivými prvky). Neobsahuje žádné možnosti geochemického modelování, z norem zahrnuje pouze CIPW. Velké mínus byla naprosto neadekvátní cena, teď zřejmě ani nejde zakoupit.

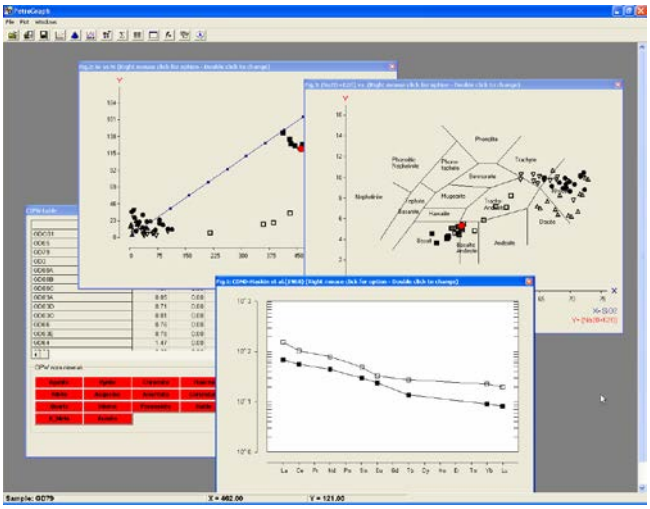
1.2.2 IgPet



CITACE: *http://www.rockware.com/product/overview.php?id=102*
OS: MS Windows (Visual Basic)
DISTRIBUCE: Komerční (US$ 199)
VSTUP: DBF, ASCII
VÝSTUP: DBF, ASCII
GRAFIKA: WMF

MOŽNOSTI: Poměrně malý a elegantní program s jednoduchým ovládáním. Obsahuje poměrně omezený výběr klasifikačních a geotektonických diagramů, o něco lepší výběr spider diagramů. Mimoto lze vynášet i uživatelské binární a ternární diagramy. Specialita je interaktivní identifikace vynesených bodů a hlavně možnost zobrazení regresních linií a trendů, vznikajících jako důsledek celé řady petrogenetických procesů (mj. frakční krystalizace, AFC, binární míšení, zone refining…). Zvláštní modul umožňuje inverzní modelování frakční krystalizace na hlavních prvcích (zadává se složeni koncových členů a frakcionujících minerálů, výstup jsou jejich proporce a stupeň frakční krystalizace). Z norem zahrnuje pouze CIPW.

1.2.3 PetroGraph



CITACE: Petrelli *et al.* (2004) *http://accounts.unipg.it/~maurip/SOFTWARE.htm*
OS: Windows 1998/2000/XP (VisualBasic)
DISTRIBUCE: Freeware
VSTUP: XLS, ROC (IgPet), PEG (PetroGraph formát: v principu textové soubory)
VÝSTUP: –
GRAFIKA: WMF, clipboard

MOŽNOSTI: *PetroGraph* je pěkný freewareový program pro vynášení analýz vyvřelých hornin (binární, ternární a spider diagramy). Specialitou je interaktivní identifikace jednotlivých vzorků v grafech a skutečně silnou stránkou je vynášení regresních trendů způsobených širokým spektrem petrogenetických procesů (např. frakční krystalizací, parciálním tavením, AFC a binárním míšením). Další možností je inverzní modelování frakční krystalizace (za použití algoritmu Stormera & Nicholse, 1978). Slabinou jsou přepočty, k dispozici je pouze CIPW norma. Také výstup výsledků je problematický.

1.2.4 Speciální software – souhrn

Výhody:

- Zavedené standardy
- *Obvykle* jednoduché a intuitivní ovládání (menu)

Nevýhody:

- Nedostatečná dokumentace použitých algoritmů („black box“)
- Neúplnost + špatná modifikovatelnost/rozšiřitelnost
- Grafický výstup není v publikační kvalitě
- [Konverze vstupních/výstupních dat]
- [Cena]

Tj. – chybí skutečně všestranný a modifikovatelný nástroj pro statistiku, grafiku a petrogenetické modelování!

Základy programovacího jazyka R

Vybrané citace:

BECKER R.A., CHAMBERS J.M. & WILKS A.R. 1988. The New S Language. Chapman & Hall, London.

GRUNSKY, E. C., 2002. R: a data analysis and statistical programming environment – an emerging tool for the geosciences: *Computers & Geosciences*, **28**: 1219–1222.

HORNIK K. 2013. The R FAQ. Accessed November 18, 2013, at URL http://cran.r-project.org/doc/FAQ/R-FAQ.html

IHAKA R. & GENTLEMAN R. 1996. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, **5:** 299–344.

JANOUŠEK V., FARROW C. M. & ERBAN V., 2006. Interpretation of whole-rock geochemical data in igneous geochemistry: introducing Geochemical Data Toolkit (GCDkit). *Journal of Petrology* **47:** 1255–1259.

JANOUŠEK, V., FARROW, C. M., ERBAN, V. & TRUBAČ, J. 2011. Brand new Geochemical Data Toolkit (GCDkit 3.0) – is it worth upgrading and browsing documentation? (Yes!). *Geologické výzkumy na Moravě a ve Slezsku* **18:** 26–30.

VENABLES, W.N. & RIPLEY, B.D. 1999. Modern Applied Statistics with S-PLUS. Springer, New York.

VENABLES W.N., SMITH D. M. & R DEVELOPMENT CORE TEAM 2013. An Introduction to R. Notes on R: A Programming Environment for Data Analysis and Graphics. Version 3.0.2 (25 September 2013). Accessed November 18, 2013, at URL http://cran.r-project.org/doc/manuals/R-intro.pdf

1.3 R = vhodná alternativa speciálního software a spreadsheetů

R = programovací jazyk pro statistické výpočty a počítačovou grafiku, navržen Ihakou & Gentlemanem (1996), Dept. of Statistics, Auckland, NZ

- od roku 1997 vyvíjí **R Core Team** "of about a dozen people", spolupráce po Internetu + CRAN (Comprehensive R Archive Network)
- syntaxe založena na komerčně úspěšném jazyce S, vyvinutém v AT & T Bell Laboratories (Becker *et al*. 1988), nyní distribuuje jako **S Plus** firma Insightful
- určen pro Wintel, Unix a Mac OS, verze 1.0 byla zveřejněna 29. února 2000, aktuální verze je 3.0.2 (září 2013)

Výhody:

- Cena (= nulová), jednoduchá instalace, malý soubor
- Jednoduchý vstup dat (není pevná struktura, mohou chybět některé hodnoty = *NA*)
- Zabudované aritmetické, databázové a statistické funkce
- Kvalitní grafický výstup v řadě formátů (PostScript, WMF, PNG, JPG…)
- Přehlednost (strukturovaný jazyk)
- Možnost použití také v interaktivním režimu

- Rozšiřitelnost (libraries/packages)
- Přenositelnost kódu

Nevýhody:

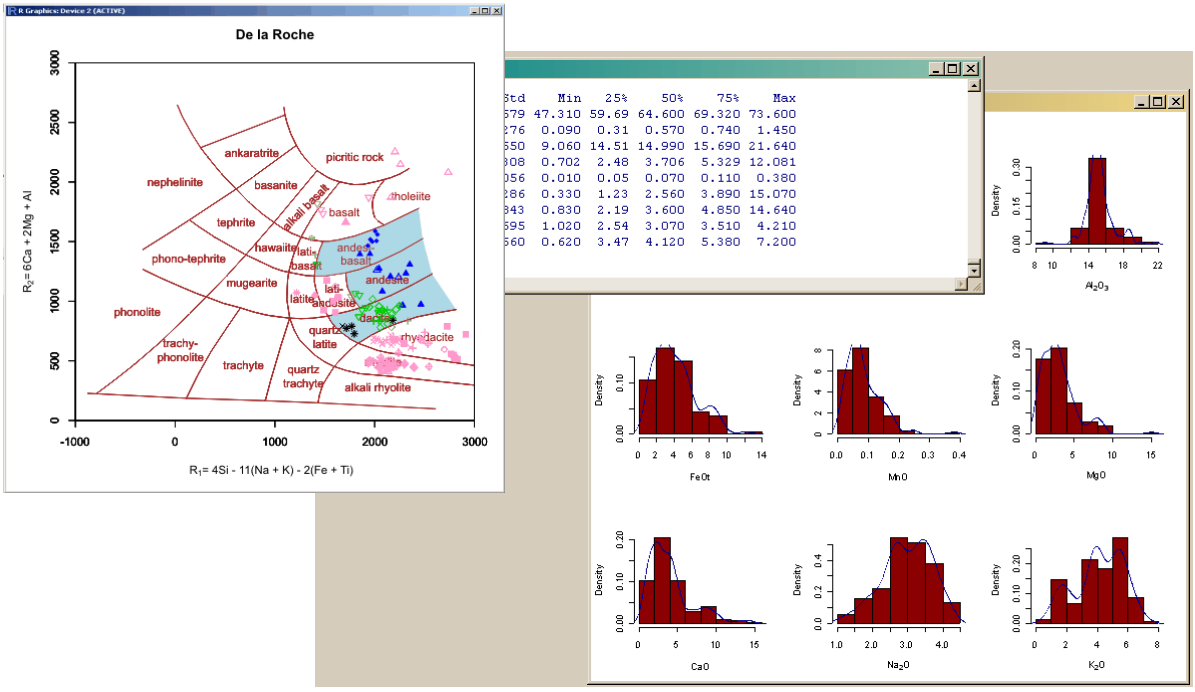
- Nový SW — dosud není příliš rozšířený
- Nezvyklá a složitá syntaxe ("steep learning curve")
- Pro seriózní práci je potřeba znalost programování
  (= psychologická bariéra pro většinu geologů)

Řešení?

GCDkit…..

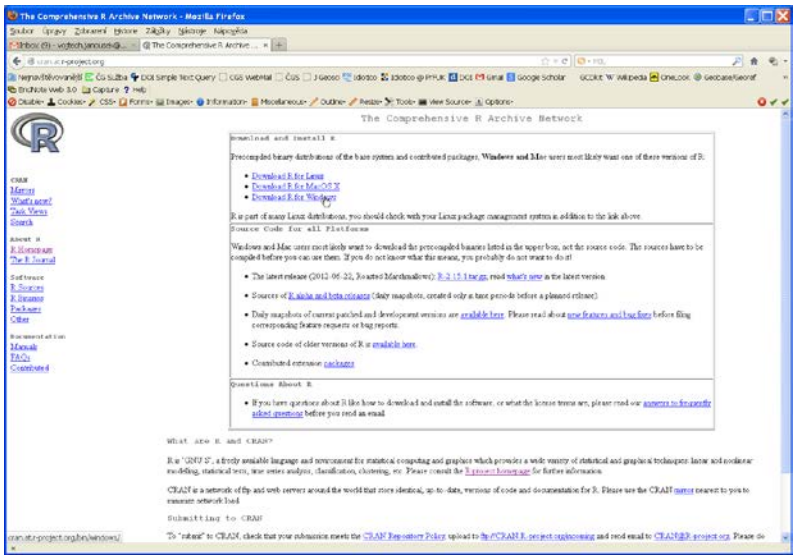
- grafické rozhraní k některým grafickým a statistickým funkcím v R
- nové speciální geochemické přepočty (např. normy, interpretace Sr–Nd izotopických dat) a grafy (šablony klasifikačních a geotektonických diagramů)
- pro normální ovládání není potřeba programování, ale možnost vkládání příkazů jazyka R je zachována
- modularita, snadná rozšiřitelnost, dostupnost (freeware), časem snad nebude záviset na platformě (verze pro Linux a Mac)

http://www.gcdkit.org



1.4 Instalace R/GCDkit

(běží pouze v MS Windows)

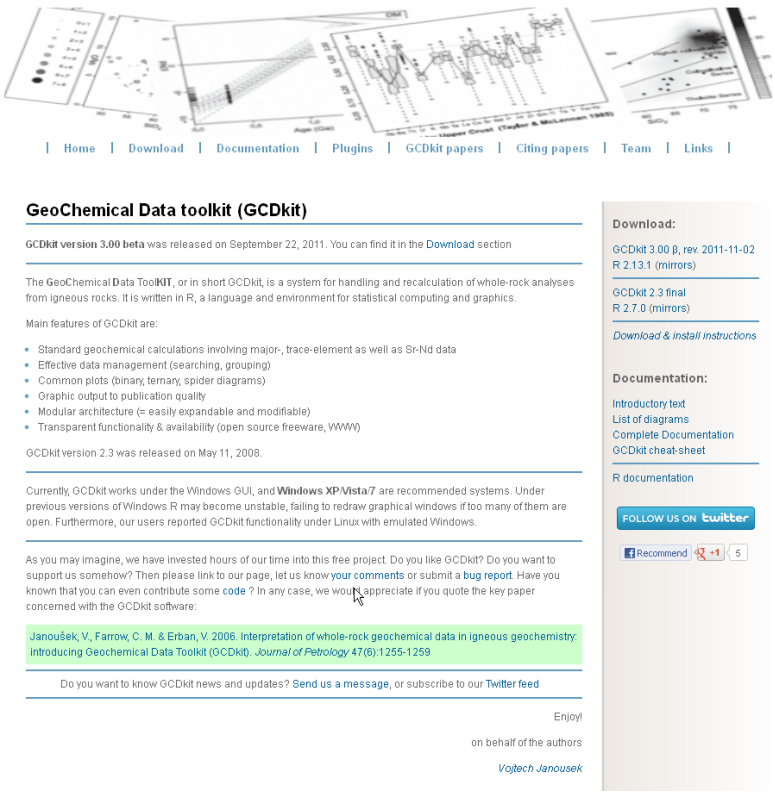


Instalace R

- Log in jako administrátor
- Stáhnout příslušnou verzi ze stránky *http://www.gcdkit.org* nebo CRAN (*http://cran.r-project.org*), jméno souboru obsahuje číslo verze XXX
- DŮLEŽITÉ! Aktuální verze GCDkitu, 3.0, byla vyvinuta v R 2.13.2. Korektní funkce v jiné verzi nemůže být garantována.
- Spustit R-XXX-win.exe, vybrat požadované položky k instalaci a cílový adresář (budeme ho označovat dále jako %RHOME%)
- !!! Nezapomenout zvolit instalaci SDI (multiple windows) interface

Instalace GCDkitu

- Stáhnout instalaci GCDkit ze *http://www.gcdkit.org*
- Double clicknout stažený EXE soubor a postupovat podle instrukcí.
- POZNÁMKA: instalační adresář nelze změnit, je to *%RHOME%\library\GCDkit*.
- nemazat adresář s daty ani soubor '*.Rprofile'* v něm, jinak zástupce na desktopu přestane fungovat.



1.5 spuštění a ukončení R překladače

Spuštění

Po dvojitém kliknutí se otevře okno *Console* určené pro vstup příkazů a výstup jejich výsledků. Kromě toho může být otevřeno jedno nebo více oken s grafickým výstupem.

Ukončení

```
> q()
```

File|Exit

1.6 Help + dokumentace

Překladač R je distribuován se soubory nápovědy v řadě formátů – čistý text (ASCII), Windows Help File (HLP), HTML (pro zobrazení v WWW prohlížeči), Latex a nově též Wiki. Kromě toho, přibaleno je několik manuálů ve formátu PDF nebo HTML.

Textový mód

```
> help(plot)
> ?plot
```

Příbuzné příkazy

```
> apropos(plot)
```

Příklady

```
> example(plot)
```

Pro HTML help:

Help|R language (html)

```
> help.start()
```

PDF dokumentace (Acrobat Reader):

Help|Manuals

1.7 Příkazy

Prostředí jazyka R lze využívat jednak v **přímém režimu,** kdy jsou příkazy vkládané do příkazové řádky přímo vykonávány. Jako alternativa se nabízí napsat celý R program do textového souboru a spustit ho v **dávkovém režimu**. Příkazy (funkce) v R jsou vkládány buď po jedné, každá na zvláštní řádek, nebo oddělené středníkem; složitější příkazy vyžadují

použití složených závorek { }. Každý příkaz je následován závorkami s parametry (a nebo prázdnými, pokud žádné parametry nejsou požadovány). Podobně jako v Unixu, také v R se rozlišují malá/velká písmena. Příkazy jsou v malých písmenech a systém rozlišuje velká a malá písmena v názvech proměnných (!).

1.7.1 Přímý režim

```
> 1+1
[1] 2
> (15+6)*3
[1] 63
```

Numerické hodnoty mohou být přiřazeny do proměnné pomocí operátoru „**<-**“, ne “**=**”!

```
> x<-5
```

V přímém režimu je hodnota proměnné zobrazena, pokud je napsáno její jméno:

```
> x
[1] 5
```

Pokud je vložen neúplný příkaz, R zobrazí prompt “+”a poskytne tak šanci vložit, co chybí:

```
> (15+6
> +  )*3
[1] 63
```

Jazyk R průběžně ukládá historii příkazů, jak byly vloženy. Tyto mohou být znovu vyvolány pomocí kláves se šipkami nahoru a dolů, a případně modifikovány a znovu odeslány, pokud si to uživatel přeje.

1.7.2 Dávkový režim

Program v jazyce R (R-script) může být také napsán předem ve formě textového souboru (ASCII). Využit k tomu může být prakticky jakýkoli textový editor, i když pro delší programy je výhodné, pokud nabízí číslování řádek, kontrolu R syntaxe a závorek. (jako freeware PSPad, *http://www.pspad.com/* nebo shareware WinEdt, *http://www.winedt.com*).

Běžně používané přípony R skriptů jsou “*r*” nebo “*R*”. Takový skript pak může být spuštěn z menu *File|Source R code* nebo pomocí příkazu `source`:

```
> source("myprogram.r")
```

Běžící program může být zastaven klávesou `Esc` nebo z menu *(Misc|Stop current computation).* Program může obsahovat komentáře uvozené znakem “`#`”:

```
> # Můj komentář
```

1.8 Objekty

1.8.1 Zacházení s objekty uloženými v paměti

R je objektově orientovaný jazyk, zahrnující objekty řady typů. Nejdůležitější **typy objektů** jsou: vectors, arrays (2D = matrices), factors, data frames, lists, functions

Pro zobrazení seznamu objektů, uložených v paměti, slouží příkaz *(Misc|List objects).*

```
> ls()
```

Odstranění nežádoucích objektů je pomocí funkce `rm`:

```
> rm(x,y,junk)
```

1.8.2 Atributy

Každý objekt může mít několik vlastností, zvaných atributy. Dva nejdůležitější jsou *length* an *mode*(s) (některé objekty jich mohou mít více): *logical*, *numeric*, *complex* nebo *character*.

```
> mode(10)
[1] "numeric"
```

1.9 Numerické vektory (numerical vectors)

1.9.1 Přiřazení

Přiřazení více položek vektoru je pomocí funkce *c*:

```
> x <- c(10.4, 5.6, 3.1, 6.4, 21.7)
> y <- c(x, 0, x)
> y
 [1] 10.4  5.6  3.1  6.4 21.7  0.0 10.4  5.6  3.1  6.4 21.7
```

1.9.2 Vektorová aritmetika

Pro výpočty lze použít jeden z operátorů:

```
  +   -    *    /    ^
```

1.9.3 Názvy

Každý vektor může mít atribut *names*, které mohou být nastaveny jako v následujícím příkladu (samozřejmě délka vektoru a jmen pro něj si musí odpovídat!):

```
> x<-c(3,15,27)
> names(x)<-c("Peter","Paul","Mary")
> names(x)
[1] "Peter" "Paul"  "Mary"
> x
```

```
Peter  Paul  Mary
    3    15    27
```

1.9.4 Generování sekvencí čísel

Pravidelné sekvence s krokem 1 nebo -1 mohou být vytvářeny pomocí operátoru ":"

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

operátor " : ", má nejvyšší prioritu

Obecnější použití má funkce `seq`:

seq (from, to, by)

```
> seq(from=30,to=-15,by=-2)
> seq(30, -15, -2)
[1]  30  28  26  24  22  20  18  16  14  12  10  8  6  4  2  0 -2 -4 -6
[20] -8 -10 -12 –14
```

rep (x, times)

Zopakuje argument `x` požadovaným počtem opakování (`times`)

```
> x<-c(3,9)
> rep(x, 5)
[1] 3 9 3 9 3 9 3 9 3 9
```

1.9.5 Funkce

Nejdůležitější funkce pro manipulaci numerických vektorů jsou:

abs(x)	Absolutní hodnota
sqrt(x)	Druhá odmocnina
log(x)	Přirozený logaritmus
log10(x)	Dekadický logaritmus
log(x,base)	Logaritmus se základem base
exp(x)	Exponenciální funkce
sin(x) cos(x) tan(x)	Trigonometrické funkce
min(x)	Minimum
max(x)	Maximum
range(x)	Totální rozsah prvků v x; odpovídá c(min(x),max(x))
length(x)	Počet prvků (= length) vektoru
rev(x)	Reverzní pořadí prvků vektoru
sort(x)	Seřadí prvky vektoru (vzestupně)
rev(sort(x))	Seřadí prvky vektoru (sestupně)
round(x,n)	Zaokrouhlí prvky vektoru x na n desetinných míst
sum(x)	Suma prvků x
mean(x)	Aritmetický průměr prvků x

1.10 Textové vektory (character vectors)

paste (x, y,..., sep="")

spojí dva textové řetězce v jeden, mezi nimi je řetězec specifikovaný v *sep*

```
> paste("Richard","Lionheart",sep=" the ")
[1] "Richard the Lionheart"
```

substring (x, start, stop)

extrahuje část řetězce *x* od pozice *first* do *last*

```
> x<-c("Utah","Vermont","Washington")
> substring(x,1,4)
[1] "Utah" "Verm" "Wash"
```

1.11 Logické vektory

Logické vektory mají prvky, které mohou nabývat jen dvou hodnot:

TRUE (T), FALSE (F)

Logické operátory

< <= > >= == (rovná se) != (nerovná se)

```
> x<-c(1,12,15,16,13,0)
> x > 13
[1] FALSE FALSE TRUE TRUE FALSE FALSE
```

Výsledek může být přiřazen do vektoru s módem logical:

```
> x<-c(1,12,15,16,13,0)
> temp<-x > 13
> temp
[1] FALSE FALSE TRUE TRUE FALSE FALSE
```

Kombinace logických podmínek c1 a c2: and (&), or (|), not (!) popř. závorky:

```
> x<-c(1,12,15,16,13,0)
> c1<-x>10
> c2<-x<15
> c1
[1] FALSE TRUE TRUE TRUE TRUE FALSE
> c2
[1] TRUE TRUE FALSE FALSE TRUE TRUE
> c1 & c2 # logical "and"
[1] FALSE TRUE FALSE FALSE TRUE FALSE
> c1 | c2 # logical "or"
[1] TRUE TRUE TRUE TRUE TRUE TRUE
```

```
> !c1 # negation
[1] TRUE FALSE FALSE FALSE FALSE TRUE
```

1.12 Chybějící hodnoty (NA, NaN)

R přiřazuje chybějícím datům speciální NA (not available). Některé operace, které za určitých okolností neposkytují smysluplné výsledky, dávají hodnotu NaN (not a number)

```
> sqrt(-15)
[1] NaN
```

Dělení nulou dává $+\infty$ (Inf).

```
> 1/0
[1] Inf
```

is.na (x)

Testuje dostupnost jednotlivých prvků vektoru x (tj. zda se nerovnají NA/NaN):

```
> x<-c(5,9,-4,12,-6,-7)
> is.na(sqrt(x))
[1] FALSE FALSE TRUE FALSE TRUE TRUE
```

1.13 Indexace vektorů

Index vektory jsou několika typů:

1. logické

```
> x[x > 10] # všechny prvky x, které jsou větší než 10
> x[!is.na(x)] # všechny prvky x, které jsou k dispozici
```

2. numerický vektor s pozitivními hodnotami

```
> x[1:10] # prvních deset prvků
> x[c(1,5,15)] # 1., 5. a 15. prvek
```

3. numerický vektor s negativními hodnotami

```
> x[-(1:5)] # všechny prvky bez prvních pěti
```

4. textový vektor se jmény prvků

```
> x[c("Peter","Paul","Mary")]
```

Cvičení 1.1

R obsahuje celou řadu datových souborů, které lze použít pro demonstraci jeho možností. Objekt islands *je vektor, v němž jsou uloženy rozlohy ostrovů a kontinentů, jejichž plocha přesahuje 10 000 čtverečních mil. Než s ním začneme pracovat, je třeba ho zpřístupnit následujícím příkazem:*

```
>  data(islands)
```

Úkoly:

- vypište obsah celého vektoru
- zobrazte rozlohu Luzonu
- jaká je průměrná hodnota celého souboru
- který kontinent je největší a jakou má rozlohu
- které kontinenty/ostrovy mají rozlohu přes 5000 čtverečních mil
- zobrazte jména 15 nejmenších a největších kontinentů/ostrovů
- za předpokladu, že jde o britské míle, přepočtěte data na km^2 (1 sq mi = 2.59 km^2)

Řešení:

```
> islands

> islands["Luzon"]
Luzon
   42

> mean(islands)
[1] 1252.729

> names(islands)[islands==max(islands)]
[1] "Asia"

> max(islands)
[1] 16988

> names(islands)[islands>5000]
[1] "Africa"  "Antarctica"  "Asia"  "North America"
[5] "South America"

> z<-sort(islands)
> z[1:15]
       Vancouver           Hainan  Prince of Wales            Timor
              12               13               13               13
          Kyushu           Taiwan      New Britain      Spitsbergen
              14               14               15               15
    Axel Heiberg         Melville      Southampton Tierra del Fuego
              16               16               16               19
           Devon            Banks            Celon
              21               23               25
> z[(length(z)-14):length(z)]
      Britain        Honshu       Sumatra        Baffin    Madagascar
           84            89           183           184           227
       Borneo    New Guinea     Greenland     Australia        Europe
          280           306           840          2968          3745
   Antarctica South America North America        Africa          Asia
        5500          6795          9390         11506         16988

> islands*2.59
```

1.14 Matice (matrices)

Matice jsou dvourozměrné objekty, které se liší od *data frames* tím, že obsahují prvky pouze jednoho typu (daného atributem ‚mode'), např. jenom čísla.

1.14.1 Vytvoření matice

Matice je vytvořena příkazem:

matrix (data, nrow, ncol, byrow=FALSE)

matice o `nrow` řádcích a `ncol` sloupcích, vyplní se hodnotou `value` (implicitně se zaplňuje po sloupcích, pokud není specifikováno `byrow=TRUE`!). Například:

```
> x<-matrix(1:12,3,4)
> x
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
```

1.14.2 Funkce

Užitečné funkce pro manipulaci matic jsou shrnuty v následující tabulce:

`nrow(x)`	Počet řádků
`ncol(x)`	Počet sloupců
`rownames(x)`	Název řádků
`colnames(x)`	Název sloupců
`rbind(x,y)`	Spojí dvě matice se stejným počtem sloupců *ncol* (nebo vektory stejné délky) po řádcích
`cbind(x,y)`	Totéž, ale po sloupcích
`t(x)`	Transponuje matici
`apply (x, margin, fun)`	Aplikuje funkci *fun* na matici *x* po řádcích (margin = 1) nebo sloupcích (margin = 2)
`x%*%y`	Násobení matic

1.14.3 Indexace matic

Prvky matice jsou uváděny v pořadí [`řádek, sloupec`]. Pokud není uvedeno nic pro `řádek` nebo sloupec, znamená to bez omezení (všechny prvky). Příklady:

```
> x[1,] # první řádek
> x[,c(1,3)] # první a třetí sloupec
> x[1:3,-2] # všechny sloupce (mimo druhého) řádků 1-3
```

Cvičení 1.2

Objekt state *shrnuje informace o jednotlivých členských státech USA. V matici* state.x77 *o 50 řádcích a 8 sloupcích jsou uložena následující data (informace z Help souboru)*

- `Population:` population estimate as of July 1, 1975
- `Income`: per capita income (1974)
- `Illiteracy`: illiteracy (1970, percent of population)
- `Life Exp`: life expectancy in years (1969–71)

- `Murder`: murder and non-negligent manslaughter rate per 100,000 population (1976)
- `HS Grad`: percent high-school graduates (1970)
- `Frost`: mean number of days with minimum temperature below freezing (1931–1960) in capital or large city
- `Area`: land area in square miles

Úkoly:

- vypište názvy dostupných proměnných (= názvy sloupců)
- kolik bylo vražd v Nevadě?
- vypište všechna data pro Nevadu a Texas
- spočtěte celkový počet obyvatel USA
- vypište pět států s nejnižší negramotností
- spočtěte průměrné hodnoty všech proměnných

Řešení:

```
> data(state) # nejprve zpřístupníme data pro další práci

> colnames(state.x77)
[1] "Population" "Income"     "Illiteracy" "Life Exp"   "Murder"
[6] "HS Grad"    "Frost"      "Area"

> state.x77["Nevada","Murder"]
[1] 11.5

> state.x77[c("Nevada","Texas"),]
       Population Income Illiteracy Life Exp Murder HS Grad Frost   Area
Nevada        590   5149        0.5    69.03   11.5    65.2   188 109889
Texas       12237   4188        2.2    70.90   12.2    47.4    35 262134

> sum(state.x77[,"Population"])
[1] 212321
> sort(state.x77[,"Illiteracy"])[1:5]
        Iowa       Nevada South Dakota        Idaho       Kansas
         0.5          0.5          0.5          0.6          0.6

> apply(state.x77,2,mean)
Population     Income Illiteracy   Life Exp     Murder    HS Grad      Frost
 4246.4200 4435.8000     1.1700    70.8786     7.3780    53.1080   104.4600
      Area
70735.8800
```

1.15 Seznamy (lists)

Seznamy *(lists)* jsou uspořádané soubory jiných objektů, označovaných jako komponenty (*components)*. Tyto komponenty mohou být kombinací objektů nejrůznějšího typu. Seznamy jsou vlastně jakési kontejnery, obsahující velmi volná seskupení vektorů, data framů, matic,

funkcí a jiných seznamů. Komponenty jsou číslovány a mohou být adresovány na základě jejich pořadí, uváděném v dvojitých hranatých závorkách "[[]]". Navíc každá komponenta může být pojmenována a potom odkazována jménem ve formě `list_name$component_name`. Jména komponent mohou být zkracována až na minimální délku, umožňující jejich jedinečnou identifikaci. Seznam se utváří příkazem:

list.name<-list (component1=,component2=…)

Nejjednodušší je demonstrovat použití seznamů na reálném příkladu:

```
> x1<-c("Lučkovice","9 km E of Blatná","disused quarry")
> x2<-"Lučkovice melamonzonite"
> x3<-c(47.31,1.05,14.94,2.23,7.01, 8.46,10.33)
> names(x3)<-c("SiO2","TiO2","Al2O3","Fe2O3","FeO","MgO","CaO")
> WR<-list(ID="Gbl-4",Locality=x1,Rock=x2,major=x3)
> WR
$ID
[1] "Gbl-4"

$Locality
[1] "Lučkovice"       "9 km E of Blatná" "disused quarry"

$Rock
[1] "Lučkovice melamonzonite"

$major
 SiO2  TiO2 Al2O3 Fe2O3   FeO   MgO   CaO
47.31  1.05 14.94  2.23  7.01  8.46 10.33
```

Některé příklady indexace:

```
> WR[[1]]
[1] "Gbl-4"
> WR$Rock # or WR$Roc, WR$R, WR[[3]] etc.
[1] "Lučkovice melamonzonite"
> WR[[2]][3]
[1] "disused quarry"
> WR$major[c("SiO2","Al2O3")]
 SiO2 Al2O3
47.31 14.94
```

1.16 Faktory

Faktor je vektorový objekt, který se používá pro klasifikaci jiného vektoru téže délky.Nabývá jen omezeného počtu diskrétních hodnot.

1.16.1 Základní použití faktorů

Vektor x se konvertuje na faktor pomocí příkazu:

factor (x)

Nejlépe je použití faktorů vysvětlit na příkladu. Datový soubor *ToothGrowth* ukazuje délku zubů morčat v závislosti na dávce vitamínu C (mg) podaném buď ve formě pomerančového džusu (OJ) nebo kyseliny askorbové (VC). Obsahuje tři sloupce:

[,1] len	Tooth length
[,2] supp	Supplement type (VC or OJ).
[,3] dose	Dose in milligrams

Nejprve vytvoříme faktor *metoda*, ukazující způsob podání vitamínu C:

```
> data(ToothGrowth)
> metoda<-factor(ToothGrowth[,"supp"])
```

Možné hodnoty faktoru *metoda* vypíše příkaz `levels()`

```
> levels(metoda)
[1] "OJ" "VC"
```

Pokud chceme spočíst průměrnou délku zubů každé ze skupin zvlášť, použijeme funkce

tapply (x, index, fun):

kde `x` je vektor, `index` je faktor (nebo list dvou faktorů) a `fun` a funkce, která se má vyhodnotit

```
> tapply(ToothGrowth[,"len"],metoda,mean)
      OJ       VC
20.66333 16.96333
> davka<-factor(ToothGrowth[,"dose"])
> tapply(ToothGrowth[,"len"],list(forma=metoda,mg=davka),mean)
```

```
     mg
forma    0.5    1    2
   OJ 13.23 22.70 26.06
   VC  7.98 16.77 26.14
```

Cvičení 1.3

Pokud budeme pokračovat ve zpracování datového souboru s morčaty, můžeme v detailu sledovat závislost délky zubů na dávce vitamínu C.

Úkoly:

- Jaké byly možné dávky vitamínu C?
- Spočtěte průměrnou délku zubů morčat pro různé dávky vitamínu C – existuje nějaká závislost?
- Kolik morčat dostalo jednotlivé dávky?

Řešení:

```
> y<-factor(ToothGrowth[,"dose"])
> levels(y)
[1] "0.5"  "1"  "2"

> tapply(ToothGrowth[,"len"],y,mean)
   0.5      1      2
10.605 19.735 26.100

> tapply(ToothGrowth[,"dose"],y,length)
0.5   1   2
 20  20  20
```

1.16.2 Konverze numerických vektorů na faktory

cut (x, breaks, labels)

Funkce `cut()` rozdělí hodnoty vektoru `x` na řadu dílčích intervalů a oklasifikuje každou jeho položku podle toho, do nějž padne. Parametr `breaks` buď definuje hraniční hodnoty nebo udává požadovaný počet těchto intervalů. Parametr `labels` může obsahovat jména jednotlivých skupin. Následující příklad využívá data o růstu zubů morčat – pokusíme se slovně popsat délku zubů pro jednotlivá měření:

```
> data(ToothGrowth)
> max(ToothGrowth[,"len"])
 [1] 33.9
# Rozdelme tedy data na ctyri skupiny, 0-10,10-20, 20-30 a 30-40
> delka<-cut(ToothGrowth[,"len"],breaks=10*(0:4),labels=c("Kratke",
"Normalni","Dlouhe","Superdlouhe"))
> delka
 [1] Kratke      Normalni    Kratke      Kratke      Kratke      Kratke
 [7] Normalni    Normalni    Kratke      Kratke      Normalni    Normalni
[13] Normalni    Normalni    Dlouhe      Normalni    Normalni    Normalni
[19] Normalni    Normalni    Dlouhe      Normalni    Superdlouhe Dlouhe
[25] Dlouhe      Superdlouhe Dlouhe      Dlouhe      Dlouhe      Dlouhe
[31] Normalni    Dlouhe      Normalni    Kratke      Normalni    Kratke
[37] Kratke      Kratke      Normalni    Kratke      Normalni    Dlouhe
[43] Dlouhe      Dlouhe      Normalni    Dlouhe      Dlouhe      Dlouhe
[49] Normalni    Dlouhe      Dlouhe      Dlouhe      Dlouhe      Dlouhe
[55] Dlouhe      Superdlouhe Dlouhe      Dlouhe      Dlouhe      Dlouhe
Levels:  Kratke Normalni Dlouhe Superdlouhe
```

1.16.3 Kontingenční tabulky (frequency tables)

table (f1,f2)

Funkce `table` sestaví kontingenční tabulku na základě dvou faktorů stejných délek, `f1` a `f2`.

Pokud si připravíme faktor `metoda`, ukazující způsob podání vitamínu C:

```
>  metoda<-factor(ToothGrowth[,"supp"])
```

a z předchozí práce máme již připravený faktor `delka`, který bude hodnotit délku zubů, můžeme sestavit kontingenční tabulku ukazující distribuci délek zubů v závislosti na formě podání vitamínu C. Pro tento účel použijeme funkci `table()`:

```
> table(metoda,delka)

      delka
metoda Kratke Normalni Dlouhe Superdlouhe
  OJ       5        7     17           1
  VC       7       13      8           2
```

Cvičení 1.4

Úkoly:

- vytvořte kontingenční tabulku délky zubů morčat v závislosti na dávce vitamínu C

Řešení:

```
> max(ToothGrowth[,"dose"])
 [1] 2
> davka<-factor(cut(ToothGrowth[,"dose"],breaks=seq(0,2,by=0.5)))
> table(davka,delka)

         delka
davka     Kratke Normalni Dlouhe Superdlouhe
  (0,0.5]     12        7      1           0
  (0.5,1]      0       12      8           0
  (1.5,2]      0        1     16           3
```