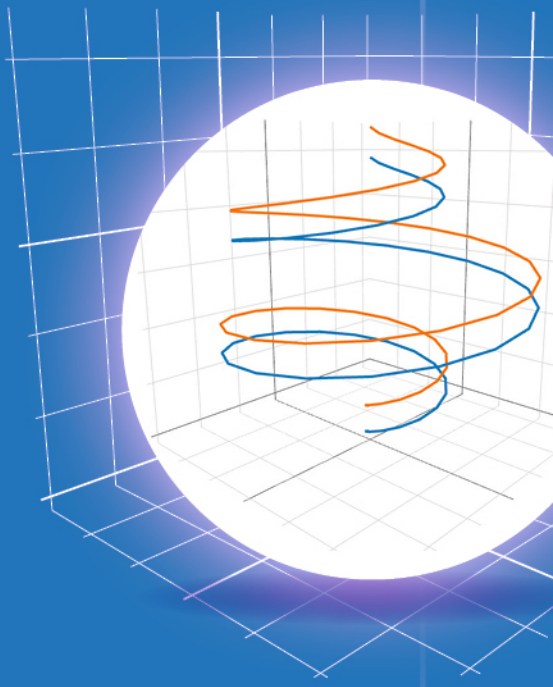
EDICE
myslíme v ...

knihovna programátora

- Přehled programovacího jazyka R pro začátečníky i pokročilé
- Vysvětlení datových typů, základů programování a zpracování dat
- Tvorba statických i interaktivních grafů dle jejich typu
- Detailní návody pro úpravy vzhledu 2D i 3D grafů
- Ukázkový příklad od zpracování dat po vytvoření grafů



Jazyk R a tvorba grafů

KATEŘINA NOVÁKOVÁ, PETR VESELÝ

GRADA

EDICE

myslíme v ...

knihovna programátora

Jazyk R
a tvorba grafů

KATEŘINA NOVÁKOVÁ, PETR VESELÝ

GRADA
Publishing

QUANTITATIVE
consulting

Kateřina Nováková, Petr Veselý

Jazyk R a tvorba grafů

Vydala Grada Publishing, a.s.
U Průhonu 22, Praha 7
obchod@grada.cz, www.grada.cz
tel.: +420 234 264 401
jako svou 8264. publikaci

Jazyková úprava Martina Mojzesová
Sazba Jaroslav Kolman
Počet stran 256
První vydání, Praha 2021
Vytiskla TISKÁRNA V RÁJI, s.r.o., Pardubice

ISBN 978–80–271–4570–6 (ePub)
ISBN 978–80–271–4569–0 (pdf)
ISBN 978–80–271–3137–2 (print)

Obsah

Předmluva

Volně šiřitelný programovací jazyk R je jedním z celosvětově nejrozšířenějších nástrojů pro statistické a matematické analýzy a modelování. Nabízí přehledné uživatelské prostředí, obsahuje širokou a neustále rozšiřovanou škálu různých metod a nadstaveb od základních matematických výpočtů po složité statistické analýzy, zpracování velkých objemů dat, modely založené na umělé inteligenci nebo tvorbu webových aplikací. Nezanedbatelnou výhodou je rovněž skutečnost, že R lze využívat na všech obvyklých operačních systémech.

Tato kniha je určena všem začínajícím i zkušenějším uživatelům tohoto jazyka, kteří potřebují vytvářet přehledné grafické výstupy svých analýz a výpočtů. Jejím účelem je přehledná prezentace možností základních grafických balíčků bez nutnosti ztrácet čas dohledáváním informací v různých internetových zdrojích.

První kapitola se zabývá jazykem R obecně a je určena jeho začínajícím a méně zkušeným uživatelům. Je proto koncipována tak, aby i čtenář bez předběžných znalostí tohoto jazyka po jejím přečtení mohl s R začít efektivně pracovat a aplikovat způsoby vizualizace výsledků popsané v dalších kapitolách. Je zde podán přehledný návod na samotnou instalaci R, vysvětlen způsob práce v tomto jazyce a prezentovány jeho nejčastěji používané nástroje včetně tvorby programovacích skriptů a nejpoužívanějších možností reprezentace dat.

Další kapitoly podrobně pokrývají základní možnosti R. Na příkladech je popsána tvorba mnoha typů 2D a 3D grafů a rovněž různých speciálních grafů, jako jsou například finanční grafy a interaktivní grafy. Největší pozornost je přitom věnována nejpoužívanějším nástrojům pro tvorbu grafů, které představují balíčky *ggplot2* a *plotly*.

Závěrečná kapitola obsahuje ukázku práce s R na jednoduchém příkladu analýzy dat. Tato kapitola je určena především méně zkušeným uživatelům a jejím smyslem je názorné propojení teoretického výkladu práce s R v první kapitole a popisu tvorby grafů v dalších kapitolách.

V celé knize je výklad důsledně doplňován ukázkami kódu, aby měl čtenář možnost si vše reálně ověřit na svém počítači. Tyto praktické příklady jsou uváděny ve zvláštních rámečcích, které jsou dvojího typu. Rámečky s bílým pozadím představují výstupy na konzoli R přesně v té podobě, v jaké je uživatel uvidí. Pro větší přehlednost jsou přitom příkazy zadané uživatelem odlišeny barvou písma od automaticky generovaných výstupů a upozornění. Rámečky druhého typu mají šedé pozadí a obsahují ukázky kódu pro vytváření grafů. Tyto rámečky jsou číslovány a je na ně odkazováno v textu. Výstupy těchto rámečků jsou graficky prezentovány v té podobě, v jaké je uživatel uvidí po spuštění daného kódu. U každého grafu je zároveň uveden odkaz na příslušný rámeček s kódem.

1 Úvod do R

R je programovací jazyk a prostředí sloužící pro matematické modelování, statistické analýzy a jejich vizualizaci. V této oblasti se jedná o jeden z nejrozšířenějších a současně nejefektivnějších nástrojů, jehož velkou výhodou je dostupnost bez poplatků.

Tato úvodní kapitola podávající informace o práci s jazykem R je sestavena tak, aby i bez předchozí znalosti tohoto nástroje umožnila jeho aktivní a efektivní využívání. Text proto pokrývá instalaci R a jeho vývojového prostředí RStudia, popis základních funkcí a jejich používání, základní datové objekty, jejich tvorbu a práci s nimi, datové importy a exporty a nástroje pro statistické zpracování dat. Nechybí ani základy programování včetně vytváření vlastních funkcí, použití for a while cyklů či podmíněných příkazů typu if-else. Na závěr je uveden seznam užitečných externích odkazů k tvorbě grafů pomocí zde využitých balíčků.

Poznamenejme, že čtenář by měl věnovat zvláštní pozornost vysvětlení rozdílů mezi širokým a dlouhým formátem dat v sekci 1.7.3. U většiny typů grafů se totiž syntaxe kódu pro jejich vytvoření i samotný grafický výstup liší podle formátu zpracovávaných dat.

Pro lepší demonstraci práce s R jsou v této kapitole uváděny ukázkové kódy a jejich výstupy v rámečcích s bílým pozadím. Tyto rámečky přesně kopírují vstupy a výstupy zobrazené na konzoli. Modře je označena část kódu, kterou píše uživatel, černým písmem je zobrazen výstup, který poskytne R, a oranžově jsou psána upozornění, která jsou v konzoli rovněž barevně odlišena od zbytku výstupu. Tyto ilustrace tak umožňují lépe pochopit způsob práce v prostředí R a čtenář si podle nich může na svém počítači sám zkoušet vysvětlované funkčnosti. Je-li kód napsán přes více řádků, znaménka + na začátku řádků není potřeba do konzole psát, R je přidává automaticky. V rámečcích jsou uváděna jen pro úplnost jako přesná kopie obsahu konzole.

1.1 R, RStudio a jejich instalace

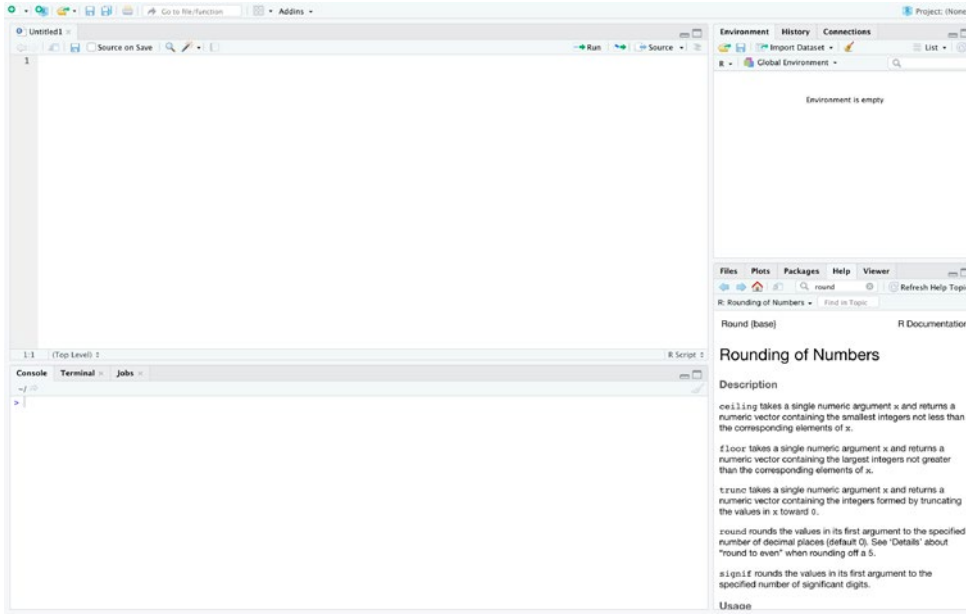
Programovací jazyk R vychází z jazyka S a považuje se za jeho jinou implementaci. Za jeho vznikem stojí Ross Ihaka a Robert Gentleman z Aucklandské univerzity na Novém Zélandu. Název jazyka je odvozen z počátečních písmen křestních jmen obou vývojářů a současně se jedná o jazykovou narážku na jazyk S [3]. První verze R se objevila v srpnu v roku 1993 a v roce 1995 bylo R zpřístupněno ostatním uživatelům jako free a open-source software. Navíc byl vytvořen R Core Team, který má na starost další vývoj a management jazyka. [14].

R je možné používat z příkazového řádku. Nicméně existují i prostředí, ve kterých je práce s tímto jazykem pohodlnější a preferována. Nejznámějším je RStudio, které bylo poprvé vydáno v únoru 2011. Hlavním vývojářem byl Hadley Wickham také z Aucklandské univerzity na Novém Zélandu. RStudio má jak volnou licenci, tak placené komerční varianty. [2].

R i RStudio jsou dostupné jak pro MS Windows, tak pro macOS a Linux [4]. Při instalaci je důležité nezaměnit pořadí a nejprve nainstalovat R a až poté RStudio. Stažení R je možné na stránkách www.r-project.org [13], kde v sekci "Getting Started" najdeme odkaz "download R" (nebo ekvivalentní odkaz "CRAN" v levém svislém menu). Otevře se nové okno na stránce cra-

n.r-project.org, kde je potřeba vybrat zemi. Po kliknutí na odkaz uvedený pro Českou republiku zvolíme svůj operační systém a poté stáhneme instalační soubor (pro MS Windows ho najdeme pod odkazem "base", u macOS je potřeba kliknout na odkaz odpovídající verzi macOS na našem počítači). Dále postupujeme dle pokynů průvodce instalací. Po dokončení instalace by už bylo možné R začít používat, přesto pro pohodlnější a přehlednější práci doporučujeme instalovat ještě RStudio.

RStudio lze stáhnout na stránce www.rstudio.com [9]. Na horní liště této stránky se nachází odkaz "Download,", který přesměruje na stránku s nabízenými licencemi. Pro běžné používání stačí zvolit bezplatnou licenci RStudio Desktop. Dále se postupuje dle pokynů průvodce instalací. Po dokončení obou instalací stačí otevřít RStudio, které se automaticky propojí s už staženým R. Obrázek 1.1 poskytuje náhled prázdného prostředí RStudia.



Obrázek 1.1: RStudio

1.2 Základy práce s R a vytváření skriptů

R je tzv. command driven software – uživatel napíše kód do konzole a klávesou Enter požádá R o jeho přeložení a vykonání. Je tak možné využívat R jako kalkulačku, kdy se do konzole napíše matematický výraz a stiskem Enter dostane uživatel výsledek. V konzoli je vhodné všimnout si znaménka > před blikajícím kurzorem. Jeho přítomnost značí, že je R připraveno k vykonání dalšího příkazu, jinak je zaneprázdněno. Zaneprázdnění lze identifikovat i z červené šesé tihranné ikony svítící v pravém horním rohu konzole. Kliknutím na tuto ikonu se provádění příslušného kódu přeruší.

Zobrazování výstupů

Zobrazené výstupy zadaných příkazů začínají téměř vždy číslem v hranaté závorce. Toto číslo značí pozici prvního zobrazovaného elementu ve výstupním objektu. U jednořádkového výstupu, např. po zadání výrazu 2+4, tak bude v hranaté závorce číslo 1. Kdybychom však ope-

rovali například s vektory a výstupem byl vektor, který by se nevešel na jeden řádek konzole, pak by se automaticky zalomil na další, na jehož počátku by bylo v hranaté závorce uvedeno pořadové číslo prvního elementu zobrazeného na tomto řádku. V následující ukázce jsou pro ilustraci tohoto principu uvedeny dva příkazy a jejich výsledky spolu se všemi symboly, které se v konzoli zobrazují. Druhý příkaz v rámečku představuje druhou mocninu všech celých čísel od 1 do 20 (syntaxe zápisů vektorů je uvedena v části 1.4.1). Na první řádek výstupu se v našem případě vešlo pouze 13 prvků a další řádek začíná 14. prvkem, proto je v hranatých závorkách číslo 14.

```
> 2 + 4
[1] 6
> (1:20)^2
[1] 1   4   9  16  25  36  49  64  81 100 121 144 169
[14] 196 225 256 289 324 361 400
```

Pokud pracujeme s velkými, nebo naopak s příliš malými čísly, R defaultně převede výstup do vědeckého formátu. Výsledek je tedy zobrazen v exponenciálním tvaru např. jako 1.0125e-06, kde -06 znamená šestou pozici za desetinnou čárkou, tedy se jedná o číslo 1.0125×10^{-6}. Počet desetinných míst, od kterého se použije vědecký formát, je možné nastavit pomocí funkce `options()` v argumentu `scipen`. Nastavení je platné i pro všechny další výstupy – jedná se o globální úpravu. Chceme-li upravit i počet číslic různých od nuly, které se zobrazí ve výstupu, využijeme argument `digits`. Vstupní hodnota se musí nacházet mezi 0 a 22 včetně. Větší počet číslic není možné nastavit. Obě nastavení jsou platná po dobu spuštění RStudia. Při novém spuštění je potřeba definovat obě úpravy znovu, pokud chceme změnit defaultní nastavení.

Defaultní hodnota argumentu `scipen` je 0 a argumentu `digits` 7. Často se argument `scipen` nastavuje na hodnotu 99 či 999, aby se uživatel zcela vyhnul vědeckému formátu, pokud se s ním v žádném výstupu nechce setkat. Jakmile je funkce použita, platí pro všechny další výstupy a nastavená hodnota není přepsána, ani pokud se použije funkce `options()`, která upravuje hodnotu jiného argumentu. Následující ukázka lépe vysvětlí použití obou argumentů. Práce s funkcemi a jejich argumenty je detailně obecně popsána v sekci 1.3.

```
> 1/987654
[1] 1.0125e-06
> options(scipen = 999)
> 1/987654
[1] 0.0000010125
> options(digits = 3) # pozor, platí zde scipen=999
> 1/987654
[1] 0.00000101
> options(scipen = 0) # pozor, platí zde digits=3
> 1/987654
[1] 1.01e-06
> options(scipen = 0, digits = 7) # defaultní nastavení
```

Poznamenejme, že přestože se u nás jako desetinný oddělovač používá většinou čárka, v R je potřeba používat desetinnou tečku. Čárkou se oddělují jednotlivé prvky vektoru.

Skripty

Jak bylo zmíněno výše, kód lze psát přímo do konzole. Nicméně odtud není možné příkazy trvale uložit. R umožňuje listovat naposledy provedenými příkazy pomocí kurzorové klávesy se šipkou nahoru a znovu je tak provést, potvrdíme-li vybraný příkaz klávesou Enter. Pro práci, kterou chceme opakovaně používat, a tedy uložit, se využívají skripty. Nový skript se otevře nad konzolí jako prázdné okno kliknutím na ikonu „New File" v levém horním rohu a poté na povel „R Script." Skript je možné uložit jako samostatný soubor a příště ho znovu otevřít, modifikovat a také z okna skriptu napsaný kód spouštět pomocí klávesových zkratek Ctrl+Enter pro Wint dows a Command+Enter pro macOS nebo pomocí tlačítka „Run" v pravé části horní lišty okna skriptu. Pokud označíme celý skript, provede se vše, jinak pouze řádek, na kterém je kurzor.

Uložit skript lze kliknutím na ikonu diskety pod názvem skriptu nebo příkazem „Save" v záložce „File". Při ukládání skriptů vyskočí nejprve dialogové okno, které vyžaduje specifikaci kódování. Je vhodné využít kódování UTF-8 nebo WINDOWS, aby byla zachována diakritika českého jazyka. Ve skriptu je možné používat diakritiku, nicméně to není obecně doporučováno kvůli problémům s kódováním při používání skriptu na jiném počítači. Pokud chceme kódování pro ukládaný skript změnit, vybereme v záložce „File" možnost „Save with Encoding…". Opět se zobrazí dialogové okno, ve kterém vybereme nové kódování.

Uložený skript je možné otevřít kliknutím na ikonu složky v levém horním rohu RStudia nebo v záložce „File" a „Open Project". Pokud chceme skript otevřít v jiném než v automaticky zvoleném kódování, nejprve ho otevřeme v defaultním kódování a poté klikneme v záložce „File" na „Reopen with Encoding…". Zobrazí se dialogové okno, v němž vybereme nové kódování, a skript se v něm znovu otevře.

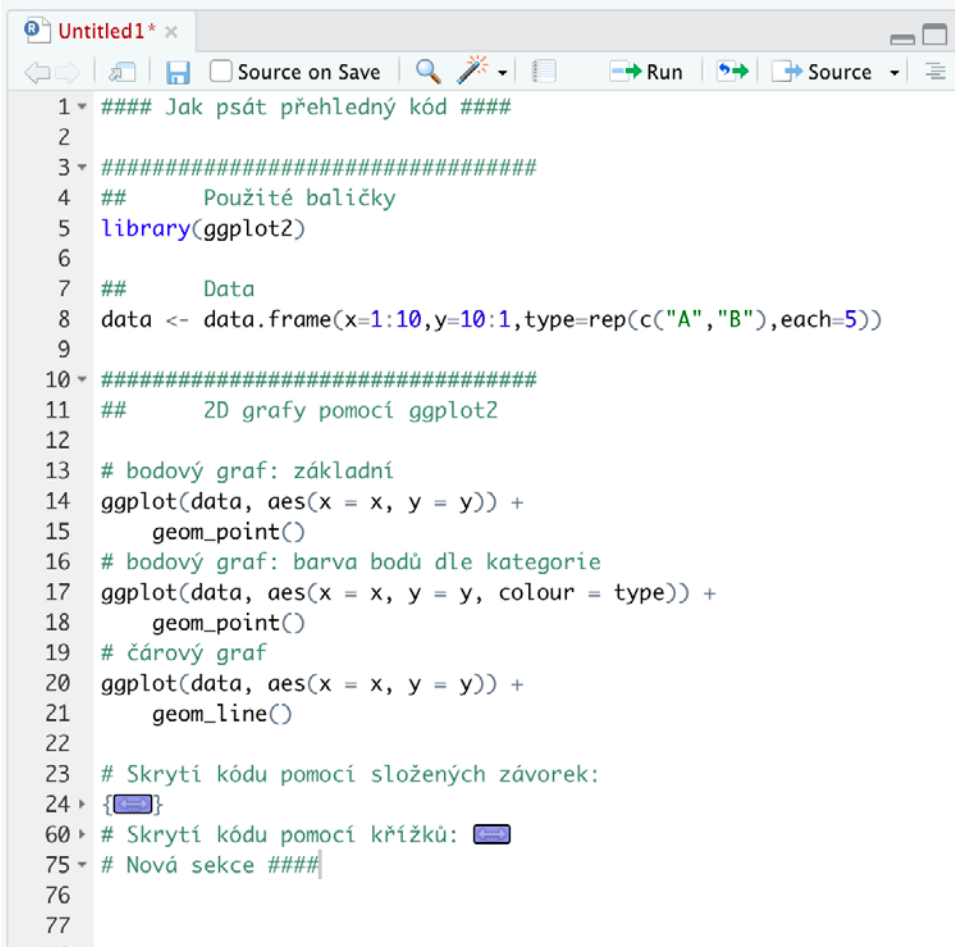
Pro psaní přehledných skriptů je dobré používat komentáře. Pro jejich označení se využívá znaménko křížku # (hashtag). Vše za ním na daném řádku je bráno jako poznámka. Křížků za sebou může být více a rovněž mohou pokračovat za textem poznámky, proto se často využívá více křížků pro identifikaci větší sekce a menší počet křížků pro podsekce. Samostatnými křížky se pak dá označovat poznámka pro konkrétní řádek kódu. Jednotlivé sekce i podsekce je pro přehlednost vhodné oddělovat prázdnými řádky.

U delšího skriptu je možné jeho části skrývat do skrytých sekcí. Končí-li poznámka alespoň čtyřmi křížky, nabídne se u čísla řádku šipka, která po kliknutí zabalí část kódu až po další nabízenou sekci. Druhou možností je uzavřít sekci do složených závorek. U první závorky se pak také u čísla řádku nabídne šipka pro skrytí sekce, které končí u druhé závorky. Výhodou tohoto způsobu je, že takto se vykoná celá sekce, pokud jsme kurzorem u první závorky a požádáme R o přeložení kódu. Skrytí kódu lze poznat podle zvýrazněné oboustranné šipky v místě zabaleného kódu. Podle čísel řádků je možné zjistit, kolik řádků je skrytých.

Základní operátory

Pro násobení se používá znak hvězdičky *, pro dělení lomítko /. Pokud chceme umocňovat, využijeme znaménko stříšky ^. Celočíselné dělení je možné pomocí skupiny znamének %/%, zbytek po dělení zjistíme pomocí dvou znaků procenta %%.

Relační operátory porovnávají dva objekty a výstupem je logická hodnota pravda (TRUE) nebo nepravda (FALSE) – více o logických objektech v sekci 1.4.1. Porovnávat můžeme jak skalární veličiny, tak vektory či matice. Výstupem může být jediná hodnota nebo logický vektor či matice odkazující na porovnávané prvky.



Obrázek 1.2: Ukázka části skriptu, psaní poznámek a skrývání sekcí

Při specifikaci podmínek často využijeme logické operátory. Pro logickou spojku „a" použijeme znak &, pro „nebo" je definována svislá čára |. Pro negaci se v R používá vykřičník !.

```
> 3 != 5
[1] TRUE
> c(1,4) < c(5,2)
[1] TRUE FALSE
```

Na indexování a výběr prvků se využívá hranatá závorka [ ] značící pozici či pozice prvků. Dvojitá hranatá závorka [[ ]] se používá pro indexování složky listu (datový objekt typu list je popsán v sekci 1.4.3). Z datových tabulek je také možné vybírat pomocí znaku dolaru $, po němž nát sleduje název sloupce tabulky.

Tabulka 1.1 obsahuje přehled základních operátorů.

Tabulka 1.1: Přehled základních operátorů v R

Aritmetické operátory:	
+	sčítání
-	odčítání
*	násobení
/	dělení
^	umocnění
%/%	celočíselné dělení
%%	zbytek po dělení
Relační operátory:	
a == b	Je a rovno b?
a != b	Je a nerovno b?
a < b	Je a menší než b?
a > b	Je a větší než b?
a <= b	Je a menší nebo rovno b?
a >= b	Je a větší nebo rovno b?
Logické operátory:	
!	ne, negace
&	a
|	nebo
Indexování:	
[ ]	výběr prvku vektoru, matice, datové tabulky
[[ ]]	výběr složky listu
$	výběr prvku datové tabulky, matice

Přiřazování jména objektům

Chceme-li v R provést příkaz zpracovávající číselné nebo jiné objekty, můžeme příslušné hodnoty napsat do konzole přímo, např. 2+4, anebo se na tyto objekty odvolat jejich jmény analogicky jako u matematických proměnných. Výhodou tohoto přístupu je větší přehlednost, omezení případných chyb a automatické použití nové hodnoty objektu v případě jeho změny. Pokud číslu 2 přiřadíme jméno `hodnota1` a číslu 4 jméno `hodnota2`, pak výpočet 2+4 můžeme napsat také jako `hodnota1+hodnota2` a výsledek bude stejný.

Název objektu přiřadíme pomocí skupiny znamének <-, kdy nalevo je název a napravo je objekt (skalár, vektor, matice a další). Použijeme-li pak název objektu, automaticky se místo něj použije hodnota mu přiřazená. Přestože v některých případech funguje i znaménko =, je obecně zavedeno používat syntaxi <-. Znaménko = se používá při specifikaci hodnot argumentů funkcí.

V ukázce je vytvořen objekt pojmenovaný `koeficient`, který má hodnotu 2, a objekt pojmenovaný `vektor`, který je dvouprvkovým vektorem s hodnotami 3 a 4 (vytváření vektorů je detailně popsáno v sekci 1.4.1). Následuje operace násobení vektoru skalárem za využití jmen obou objektů. Všimněme si, že pokud napíšeme příkaz do konzole bez přiřazení jména, vytiskne se výsledek. Pokud výsledku přiřazujeme jméno, v konzoli se žádný výstup neobjeví, protože cílem operace je vytvořit objekt, nikoliv zobrazit jeho hodnotu. Pro zjištění hodnoty je potřeba jméno objektu napsat do konzole a výsledek se zobrazí.

```
> koeficient <- 2
> vektor <- c(3,4)
> koeficient*vektor
[1] 6 8
> vystup <- koeficient*vektor
> vystup
[1] 6 8
```

Objekty je možné pojmenovávat zcela libovolně pomocí všech dostupných znaků (včetně mezer a speciálních symbolů, jako např. japonských a čínských znaků). Nicméně exotičtěji volené názvy pak vyžadují jejich uvození a zakončení zpětným apostrofem. Zpětný apostrof není potřeba při splnění následujících podmínek:

- jméno začíná písmenem,
- jméno obsahuje pouze písmena, číslice, tečky a podtržítka.

Pokud bude jméno obsahovat jakékoliv jiné znaky jako mezery, čárky či speciální znaky, je potřeba použít zpětné apostrofy, např. `` `1. můj vektor` ``. Tento přístup se obecně nedoporučuje. Častým případem jmen s mezerami a dalšími znaky vyžadujícími zpětné apostrofy jsou názvy sloupců datové tabulky, která je do R importována z jiného datového souboru, např. XLSX, CSV. Pak je vhodné jména sloupců změnit.

Poznamenejme, že R rozlišuje velká a malá písmena. Pokud místo objektu nazvaného `data` napíšeme `Data`, R takový objekt buď nenajde, anebo místo něj použije zcela jiný objekt uložený pod názvem `Data`.

Ukládání pracovního prostředí

Všechny vytvořené a pojmenované objekty RStudio zobrazuje v pravém horním rohu v sekci „Environment", kde lze přímo zjistit jejich rozměry (např. délku vektorů, dimenzi matic a datových tabulek) a další základní informace. S těmito objekty můžeme okamžitě pracovat. Uložením skriptu se však tyto objekty neuloží. Skript obsahuje pouze kód, který slouží k replikaci celého výpočetního procesu. Nicméně je možné uložit i všechny vytvořené objekty, tzn. uložit si pracovní prostředí, které se v RStudiu nazývá Workspace.

V záložce „Session" klikneme na položku „Save Workspace As", pomocí které se prostředí uloží do vybrané složky s koncovkou Rdata. Kliknutím na „Load Workspace" je možné jej opět nahrát. Po doběhnutí kódu je tak možné si vytvořené objekty uložit a pracovat s nimi později bez opětovného pouštění kódu.

Uložit a nahrát pracovní prostředí je možné i přímo z konzole nebo příkazem ve skriptu. Pro uložení se využije funkce `save.image()` ze základního balíčku *base*, která má za vstup název, pod kterým chceme prostředí uložit, s koncovkou RData. Název musí být v uvozovkách. Prostředí se uloží do nastavené pracovní složky. Pokud chceme cílové místo změnit, zadáme do názvu i cestu. Nastavení cesty a pracovní složky je detailně popsáno v části 1.7.1. K nahrání prostředí slouží funkce `load()` ze základního balíčku *base*. Vstupem je název souboru s koncovkou RData v uvozovkách. Pokud se prostředí nenachází v nastavené pracovní složce, bude název opět obsahovat i cestu.

```
> save.image("Data1.RData")
> load("Data1.RData")
```

Chceme-li uložit jen některé objekty, využijeme funkci `save()`, ve které vyjmenujeme jeden či více objektů oddělených čárkami a do argumentu `file` specifikujeme, jak se má soubor pojmenovat. Jméno se píše v uvozovkách a musí mít koncovku RData. Součástí jména výstupního souboru opět může být i cesta, pokud se liší od aktuálně nastaveného adresáře. V ukázce jsou ukládány vybrané tři objekty pojmenované `data`, `vektor` a `x` do souboru pojmenovaného Data2.RData.

```
> save(data, vektor, x, "Data2.RData")
```

1.3 Funkce, konstanty a speciální hodnoty

Pomocí funkcí se provádějí nejen výpočty, importy a transformace dat, ale také globální nastaavení R, nastavení pracovní složky a jiné. Všechny funkce se volají stejným způsobem a stejnou logikou se řídí i práce s jejich argumenty. Obojí je dále detailně popsáno. Následuje seznam předdefinovaných konstant a speciálních hodnot, s nimiž se lze setkat.

1.3.1 Matematické funkce

V základní instalaci R jsou dostupné nejpoužívanější matematické funkce. Funkci voláme jménem, u kterého je potřeba rozlišit velká a malá písmena. V závorce za názvem funkce se definují argumenty, které jsou pro každou funkci určeny specifickými názvy. Názvy argumentů s vysvětlením jejich významu a s možnostmi jejich vstupů a případných defaultních hodnot lze najít v nápovědě či dokumentaci funkcí. Hodnoty argumentů se specifikují pomocí rovnítka = a oddělují se čárkou.

Nápověda k funkcím se nachází v pravé spodní části RStudia v záložce „Help". Pro zobrazení nápovědy napíšeme název funkce za ikonu lupy. Nápovědu pro konkrétní funkci je také možné zobrazit z konzole pomocí funkce `help()`, kde do závorek napíšeme název funkce, pro kterou chceme nápovědu zobrazit. Ekvivalentním zápisem je použít otazník a za něj napsat název funkce. Další informace o nápovědě a dokumentacích funkcí a balíčků jsou uvedeny v sekci 1.8.

Pokud bychom chtěli zobrazit nápovědu pro funkci pojmenovanou `sum`, kód bude vypadat následovně:

```
> help(sum)
> ?sum
```

Princip práce s argumenty funkcí si vysvětlíme na funkci `round()`, která zaokrouhlí číslo na daný počet desetinných míst. Jejími argumenty dle nápovědy jsou `x` a `digits`. Za první argument `x` je možné dosadit pouze numerický vektor (skalár je zde brán jako vektor délky 1). Defaultně žádnou hodnotu nemá specifikovanou, tento argument je tedy potřeba vyplnit vždy, pokud chceme funkci použít. Druhý argument `digits` specifikuje, na kolik desetinných míst se má vstup prvního argumentu zaokrouhlit. Hodnotou tohoto argumentu musí být celé číslo (`integer`), přičemž defaultně je dosazena hodnota 0 (to poznáme tak, že u popisu funkce v nápovědě je tato hodnota uvedena za rovnítkem). Pokud tedy druhý argument nevyužijeme,

funkce vstup automaticky zaokrouhlí na celá čísla (tedy jako bychom uvedli `digits=0`). Pokud argument specifikujeme, aplikuje se nová hodnota.

Pokud dodržíme pořadí argumentů, není třeba jejich jména používat. Funkce dosadí vstupy do příslušných argumentů automaticky. Pokud chceme pořadí argumentů přehodit či některé vynechat, je potřeba jejich názvy použít. Některé funkce mohou využívat i další argumenty, které už nemají specifikované pořadí. Pak je potřeba jména argumentů uvést a tím specifikovat, k jakému argumentu se která vstupní hodnota vztahuje. V nápovědě je vždy detailně popsáno, jaký typ objektu do argumentu vstupuje – zda je to vektor, datová tabulka, matice, textový řetězec apod. Použijeme-li vstup ve špatném formátu, funkce nebude fungovat (na to R upozorní a v některých případech i napíše, jaký vstup je vyžadován) nebo může spočítat špatnou hodnotu.

V následující ukázce je demonstrována práce s argumenty funkce `round()`, jejich prohození a vynechání jejich jmen. V běžné praxi se nejpoužívanější argumenty nepojmenovávají, nicméně ponechání jména ničemu neškodí.

```
> round(x = 237.253)
[1] 237
> round(x = 237.253, digits = 1)
[1] 237.3
> round(237.253, 1)
[1] 237.3
> round(digits = 1, x = 237.253)
[1] 237.3
```

Výstup funkce je možné uložit do nového objektu analogicky jako v ukázce přiřazení názvu pro skalár či vektor v sekci 1.2. Nejprve se napíše jméno, do kterého chceme výstup uložit, poté použijeme syntaxi <- a za ni napíšeme celou funkci. Když pak takto pojmenovaný výstup zavoláme, zobrazí se uložený vypočtený objekt a výpočet se již znovu neprovádí.

```
> promenna <- round(237.253)
> promenna
[1] 237
```

Tabulka 1.2 obsahuje výběr základních matematických funkcí spolu s názvy argumentů a krátkým popisem využití funkce [7]. Vstupem `x` může být i vektor, případně matice. V takových případech se funkce aplikuje zvlášť na všechny prvky. Výjimkou jsou funkce `sum()` a `prod()`, které sečtou a vynásobí všechny prvky vstupního objektu, a dále funkce `cumsum()` a `cumprod()`, které vytvoří vektory částečných součtů a součinů vstupního vektoru.

Tabulka 1.2: Přehled základních funkcí v R

Funkce	Popis
`sqrt(x)`	druhá odmocnina
`abs(x)`	absolutní hodnota
`ceiling(x)`	horní celá část čísla
`floor(x)`	dolní celá část čísla
`trunc(x)`	desetinná část čísla
`exp(x)`	exponenciální funkce
`log(x, base = exp(1))`	logaritmus (defaultně přirozený)
`log10(x)`	desítkový logaritmus
`cos(x), sin(x), tan(x)`	kosinus, sinus, tangens
`acos(x), asin(x), atan(x)`	arkus kosinus, arkus sinus, arkus tangens
`round(x, digits)`	zaokrouhlení na daný počet míst (kladná hodnota značí počet desetinných míst, záporná hodnota řád před desetinnou tečkou, např. `digits = 2` zaokrouhluje na setiny, `digits = -2` na stovky)
`sum(x)`	součet prvků
`cumsum(x)`	kumulativní součet prvků
`prod(x)`	součin prvků
`cumprod(x)`	kumulativní součin prvků

Například exponenciální funkce má první argument nazvaný `x`, jehož vstupem je numerický vektor. Pokud zavoláme funkci `exp(x=4)`, provede se výpočet . Pokud jako vstup vložíme víceprvkový vektor, výstupem bude stejně dlouhý vektor s postupnými mocninami. Název arý gumentu je opět možné vynechat.

```
> exp(1)
[1] 2.718282
> exp(c(1,3,5))
[1] 2.718282  20.085537 148.413159
```

U funkcí `sum()` a `prod()` je možné využít logický argument `na.rm`, ve kterém se specifikuje, jak pracovat s chybějícími hodnotami (chybějící hodnoty jsou detailněji popsané v sekci 1.3.3). Defaultně je hodnota argumentu nastavena jako FALSE. To znamená, že pokud objekt, jehož prvky chceme sečíst, obsahuje chybějící hodnoty, výstupem bude také chybějící hodnota. Nastavíme-li tento argument jako TRUE, chybějící hodnoty se budou ignorovat a výstupem bude součet ostatních prvků objektu. U ostatních funkcí platí, že je-li vstupem chybějící hodnota na některém z prvků objektu, bude i výstupem tohoto prvku. Ostatní prvky vstupního objektu budou funkcemi vyhodnoceny bez problému.