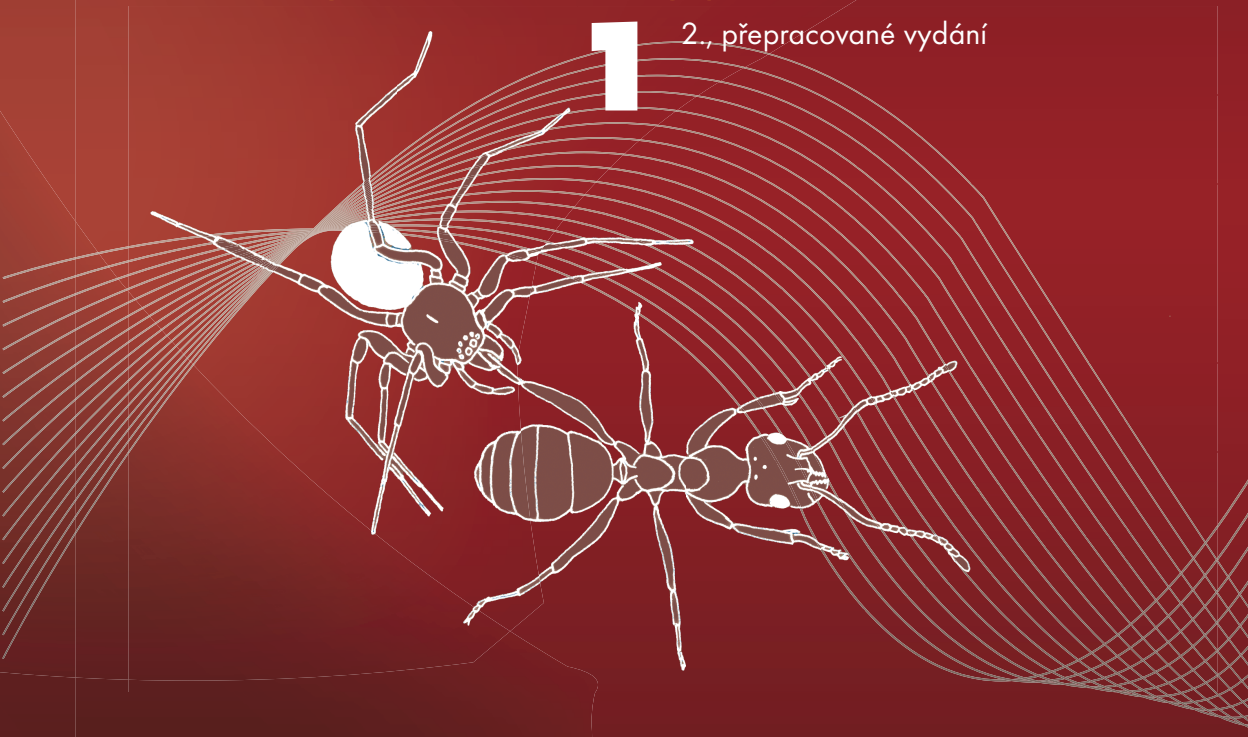
MODERNÍ ANALÝZA BIOLOGICKÝCH DAT

ZOBECNĚNÉ LINEÁRNÍ MODELY V PROSTŘEDÍ R

1

2., přepracované vydání



STANO PEKÁR
MAREK BRABEC

MASARYKOVA UNIVERZITA

MODERNÍ ANALÝZA BIOLOGICKÝCH DAT
ZOBECNĚNÉ LINEÁRNÍ MODELY V PROSTŘEDÍ **R**
1. díl

STANO PEKÁR, MAREK BRABEC

MUNI
PRESS

MODERNÍ ANALÝZA BIOLOGICKÝCH DAT

ZOBECNĚNÉ LINEÁRNÍ MODELY V PROSTŘEDÍ R

1. díl

2., přepracované vydání

STANO PEKÁR
MAREK BRABEC

Masarykova univerzita, Brno 2020

https://www.press.muni.cz/moderni-analyza-1

Pekár S. & Brabec M. 2020. Modern analysis of biological data.
1. Generalized linear models in R. 2nd edition. Masaryk University Press, Brno.

ISBN 978-80-210-9623-3
ISBN 978-80-210-9622-6 (brožováno)
ISBN 978-80-210-9782-7 (váz.)
ISBN 978-80-86960-44-9 (1. vyd.)

OBSAH

Předmluva ke druhému vydání . . . . . **IX**
Předmluva k prvnímu vydání . . . . . **XI**

1 Úvod **1**
1.1 Jak číst tuto knihu 3
1.2 Typy proměnných 5
1.3 Konvence 6

2 Statistický software **7**
2.1 Prostředí **R** 7
2.2 Instalace a ovládání **R** 9
2.3 R Studio 11
2.4 Základní operace 12
2.5 Příprava data frame 20

3 Jednoduché charakteristiky, EDA **23**
3.1 Střední hodnota 23
3.2 Rozptyl 25
3.3 Intervaly spolehlivosti 27
3.4 Tabulky popisných charakteristik 28
3.5 Grafy 29
3.5.1 Grafy rozdělení 32
3.5.2 Bodové grafy 35
3.5.3 Krabicové grafy 36
3.5.4 Lattice grafy 37
3.5.5 Interakční grafy 38
3.5.6 Sloupcové grafy 40
3.5.7 Párové grafy 40
3.5.8 Trojrozměrné grafy 40
3.5.9 Elegantní grafy 41
3.5.10 Grafy odhadů 42
3.5.11 Grafy křivek 42

4 Statistické modelování 45
4.1 Regresní model 45
4.2 Obecný lineární model 47
4.3 Zobecněný lineární model 49
4.4 Hledání „správného“ modelu 53
4.5 Kritika modelu 55

5 První pokus 63
5.1 Popis příkladu 63
5.2 Explorativní analýza (EDA) 63
5.3 Očekávaný model 64
5.4 Statistická analýza 65
5.4.1 Porovnání úrovní pomocí kontrastů 66
5.4.2 Apriorní kontrasty 69
5.4.3 Posteriorní slučování 75
5.4.4 Post hoc testy 78
5.4.5 Diagnostika modelu 80
5.5 Závěr 82

6 Systematická složka 83
6.1 Regrese (a obecnější analog) 84
6.2 ANOVA (a obecnější analog) 87
6.3 ANCOVA (a obecnější analog) 88
6.4 Syntaxe systematické složky 90

7 Náhodná složka 93
7.1 Spojitá měření 94
7.2 Počty a četnosti 97
7.3 Relativní četnosti 99
7.4 Quasi „rozdělení“ 100

8 Gaussovo rozdělení 101
8.1 Charakteristika LM a GLM 101
8.2 Regrese 102
8.3 Vážená regrese 113
8.4 Dvoufaktorová ANOVA 120
8.5 Jednofaktorová ANCOVA 131
8.6 Vícenásobná regrese 137

9 Gama a lognormální rozdělení 151
9.1 Charakteristika gama modelu 151
9.2 Charakteristika lognormálního modelu 152
9.3 Analog regrese 153
9.4 Analog dvoufaktorové ANOVA 161
9.5 Dvoufaktorová ANCOVA 169

10 Poissonovo rozdělení 179
10.1 Charakteristika poissonovského modelu 179
10.2 Analog jednofaktorové ANOVA 180
10.3 Overdispersion a underdispersion 185
10.4 Analog vícenásobné regrese 187
10.5 Analog jednofaktorové ANCOVA 195
10.6 Analog třífaktorové ANOVA (kontingenční tabulka) 202

11 Negativně binomické rozdělení 211
11.1 Charakteristika negativně binomického modelu 211
11.2 Analog jednofaktorové ANOVA 212

12 Binomické rozdělení 221
12.1 Charakteristika binomického modelu 222
12.2 Analog dvoufaktorové ANOVA 224
12.3 Overdispersion a underdispersion 232
12.4 Analog regrese 233
12.5 Analog jednofaktorové ANCOVA 240
12.6 Binární analog jednofaktorové ANCOVA 246

Použitá a doporučená literatura 253

Rejstřík 257
Obecný 257
Příkazy a jejich argumenty 261

PŘEDMLUVA KE DRUHÉMU VYDÁNÍ

Po jedenácti letech jsme se rozhodli připravit druhé vydání prvního dílu *Moderní analýzy biologických dat* (Pekár & Brabec 2009). Tentokrát u jiného nakladatele a v lepší produkční kvalitě.

Jde o mírně přepracované vydání, a to jak koncepčně, tak obsahem. Inovace je spojena především s propracovanějším výkladem látky. Při výuce jsme za těch 11 let zjistili, které věci jsou pro studenty těžce srozumitelné, resp. co v knize chybí. To jsme se snažili lépe vysvětlit, resp. doplnit. Největší změnou je zkrácení kapitoly 5, původně neúměrně dlouhé, přesunem některých jejích částí do jiných kapitol. Věříme, že se tím zlepší i celkové pochopení teorie.

Software **R** se za posledních 11 let značně vyvinul a rozrostl. A to jak do významu, tak i do obsahu. V současné době je to jeden z předních statistických programových nástrojů. Jeho použití pro statistickou analýzu, přípravu dat i podpůrné výpočty se v dnešní době vyučuje na mnoha univerzitách a stalo se tak například v biologii standardem. K analýze dat jej používá celá řada biologů i dalších specialistů (nejen) v oblasti přírodních věd. Počet balíčků implementovaných v **R** se neuvěřitelně znásobil, takže v současné době poskytuje opravdu rozmanité možnosti analýzy dat – od jednoduchých ke komplexním, od tradičních k velmi moderním či nově se vyvíjejícím modelům, metodám a technikám. Mnohé z těchto metod jsou implementovány vícenásobně – lze je nalézt v odlišném zpracování v několika různých balíčcích. Proto jsme reflektovali současné trendy a doplnili nové grafické funkce a ilustrujeme také použití některých užitečných podpůrných funkcí z jiných balíčků, než je stats. V důsledku toho obsah knihy mírně narostl. Věříme, že vás to neodradí, a že se s chutí pustíte do čtení a analýzy vlastních dat.

Září 2020

Stano Pekár
Marek Brabec

PŘEDMLUVA K PRVNÍMU VYDÁNÍ

Tato kniha je určena především studentům a kolegům z biologických oborů, souhrnně tedy biologům, kteří působí na přírodovědeckých, zemědělských, veterinárních, farmaceutických nebo lékařských fakultách či ve výzkumných ústavech podobného zaměření. Je určená lidem, kteří mají pouze základní statistické vzdělání (například jen základní kurz statistiky/biostatistiky), a přitom potřebují korektně analyzovat výsledky svých pozorování či experimentů.

Je znám všeobecný negativní postoj biologů k matematice, a proto jsme se snažili psát text odlehčeně – s minimem matematických požadavků. Na některých místech to šlo snadno, jinde hůře a někdy skoro vůbec. Proto se v textu takřka ve všech kapitolách objevují matematické rovnice (byť ve zjednodušené podobě, tak aby se méně zběhlému čtenáři lépe četly). Přesto je v knize *mnohem* méně matematické a statistické teorie, než bývá ve statistické literatuře zvykem.

Knihu jsme postavili hlavně na příkladech analýz reálných dat. Ty jsou rozpracované od začátku až do konce, tj. od popisu a stanovení cíle až po závěr. Simulují tak (byť poněkud zjednodušeně) postup obvyklý při práci na příspěvku do vědeckého časopisu. Domníváme se totiž, že praktickou zkušenost s analýzou dat nelze ničím jiným nahradit. Pro motivaci jsou samozřejmě nejlepší data z oblastí, které jsou danému čtenáři známé (takže o jejich vlastnostech dokáže kriticky uvažovat). Vzhledem k předpokládanému biologicky orientovanému čtenáři jsme volili příklady z ekologie, etologie, toxikologie, fyziologie, medicíny, zoologie a zemědělské produkce. Jsou to všechno data, jejichž analýzou se S. Pekár zabýval v rámci řešení různých výzkumných problémů. Data byla upravena tak, aby vyhovovala pedagogickým záměrům této knihy. Např. původní dlouhé a složité latinské názvy druhů byly nahrazeny obecným, ale krátkým jménem (specA).

Text knihy je téměř dvojjazyčný. Kniha je psána v češtině, ale pracuje s mnohými anglickými termíny. Důvodů je několik. Především fakt, že software, ve kterém jsou příklady řešeny, nemá (zatím) českou podporu (či lokalizaci). České názvy proměnných a jejich úrovní by musely být bez diakritiky, což by leckdy mohlo způsobit zmatek. Dále jsou anglická slova obecně kratší než jejich české ekvivalenty, a psaní příkazů je tudíž efektivnější. Kromě použití anglických slov v prostředí **R** používáme anglické termíny i v teoretické části. Je to proto, že české statistické názvosloví není vždy zcela ustálené (zejména v oblasti novějších metod, modelů či technik) a bere si často za základ

amatérsky „počeštěné“ anglické termíny. Proto jsme považovali za nejvhodnější používat termíny původní – anglické, s výjimkou tradičních označení, jež jsou v češtině dobře (a bezesporně) zdomácněla.

Na závěr bychom chtěli poděkovat všem kolegům, bez jejichž přispění by tato kniha nevznikla. V první řadě prof. RNDr. V. Jarošíkovi, CSc., který prvního z autorů ještě v průběhu studií na PřF seznámil s GLM a podnítil tak jeho zájem o statistiku vůbec. RNDr. A. Hoňkovi, CSc., za mnohé konzultace, kolegům z VÚRV Praha-Ruzyně, kolegům a studentům na PřF MU Brno za podnětné připomínky k původnímu textu. Konečně bychom chtěli poděkovat několika kolegům, kteří laskavě souhlasili s použitím svých (byť upravených) dat k příkladům uvedeným v této knize. Jsou to jmenovitě: T. Bilde, A. Honěk, J. Hubert, D. Chmelař, J. Lipavský, M. Řezáč, P. Saska a V. Stejskal.

Jakékoliv připomínky k textu a obsahu knihy rádi uvítáme na e-mailových adresách: *pekar@sci.muni.cz* a/nebo *mbrabec@cs.cas.cz.*

Prosinec 2008

Stano Pekár
Marek Brabec

1 ÚVOD

Začněme ukázkou dvou zcela standardních situací. První se odehrává po obhajobě diplomové práce, kdy si student stěžuje jinému studentovi: „Prý jsem použil špatný test." Ta druhá probíhá třeba na chodbě výzkumného ústavu, kde biolog vyčítá jinému kolegovi způsob prezentace výsledků: „Chtělo by to nějakou statistiku." Obě situace mají jedno společné – zoufalý povzdech nad statistikou. Statistické zpracování dat je totiž v mnoha biologických oborech nedílnou součástí bakalářských/diplomových prací i vědeckých publikací, a provází tak biologa celou jeho kariérou. V některých oborech, jako je např. klasická taxonomie, může mít statistická analýza spíše okrajový význam. Pro jiné obory, jakými jsou např. ekologie nebo fyziologie, je často přímo téměř základním metodologickým kamenem. V těchto oborech mohou mít nedostatky ve statistickém zpracování zcela katastrofální dopad. Prezentaci výsledků z rozsáhlého experimentu ve formě vědeckého článku není možné bez kvalitního statistického hodnocení vůbec publikovat. Všechna předchozí snaha autorů tak může přijít vniveč.

Jedinou cestou, jak tomu zabránit, je snažit se porozumět statistice, nebo alespoň vyhledat někoho, kdo jí rozumí (což nemusí být snadné). Věžte ale, že praktická analýza dat je v dnešní době mnohem jednodušší, než tomu bylo kdysi. Zasloužil se o to velice rychlý rozvoj výpočetní techniky. Ten znamenal pro analýzu dat doslova revoluci. Před dobou osobních počítačů i jednoduchá statistická analýza s kalkulačkou a tužkou trvala několik hodin, leckdy i několik dní (přitom nebylo jednoduché se vyvarovat výpočetních chyb apod.). Dnes s použitím počítačů může být i velice složitá analýza otázkou několika mili-sekund. Vytvoření grafu na počítači je často snazší a rychlejší než třeba příprava kávy. Technické usnadnění však vedlo ke zvýšeným nárokům na použití adekvátních statistických metod. Pokud se v minulosti upřednostňovaly nenáročné metody, byť nebyly zcela nejvhodnější, pak dnes se klade důraz na použití metod, jež lépe odpovídají skutečné situaci v datech. Výpočetní náročnost již do značné míry není nepřekročitelnou bariérou. Oproti „univerzálním" jednoduchým postupům se dnes při aplikacích statistiky upřednostňuje použití metod a modelů, které realisticky berou v potaz skutečné vlastnosti dat a studovaného procesu. To znamená, že metoda či statistický model se přizpůsobuje situaci, a nikoli situace modelu (nebo by to tak alespoň být mělo).

Tento samozřejmý požadavek však nelze vždy snadno splnit, neboť je k tomu zapotřebí mít teoretické vědomosti o různých modelech a metodách. Dále je zapotřebí určitých zkušeností s praktickou analýzou dat, s aplikací různých modelů na reálná data,

s tím, jak model sestavit, jak ho odhadnout, provést příslušné testy apod. Je jasné, že zkušenosti člověk jen z knih nezíská. K jejich nabytí musí každý něco „odpracovat" a hlavně „odpřemýšlet". Mohou mu v tom však pomoci rady a vzorové příklady.

Tato kniha se snaží pomoci právě takovým způsobem: ukázkou konkrétních analýz, zpracovaných od pojmenování problému přes vývoj statistického modelu až po formulaci možných závěrů. Není ani nechce být návodem na to, jak vybrat „nejlepší" metodu pro analýzu konkrétních dat (takovýto návod dle našeho názoru ani není možné poskytnout, neboť volba metody souvisí i s povahou problému, který chceme řešit, s tím, na které jeho aspekty klade zadavatel analýzy důraz apod.). Namísto toho se snaží poukázat na to, jak o statistických modelech přemýšlet, jak je používat, ale i to, jak je nepoužívat – tedy upozornit na leckteré problémy a chyby, které se mohou v praxi vyskytnout (a vyskytují). Různé obecnější postupy ilustrujeme na konkrétních (pro biologa snad poutavých) příkladech a podrobně zmiňujeme i přesnou implementaci v jazyce **R**, takže si je může každý vyzkoušet sám, a to jak na datech, jež jsou dodávána s knihou, tak na svých vlastních.

V této knize se budeme věnovat takřka výlučně regresním modelům, protože regrese je v řadě biologických studií velice často používaným nástrojem. Pomáhá řešit důležité otázky typu: jak veličina *y* závisí na veličině *x* či jak predikovat (předpovídat) hodnotu nového pozorování, když známe hodnoty jedné nebo několika tzv. vysvětlujících proměnných. Řeč však bude o regresi v poněkud širším pojetí, než je to, které znáte ze základních kurzů statistiky. Budeme se zabývat tzv. zobecněnými lineárními modely (GLM, Generalized Linear Model). Jejich název explicitně zdůrazňuje, že jde o *zobecnění* lineárních regresních modelů (a modelů analýzy rozptylu), tedy o zobecnění toho, co mnozí znají pod pojmem obecný lineární model. Zobecnění je to velmi užitečné, neboť umožňuje analyzovat data, která ve standardní regresi nelze zpracovávat bez hrubých aproximací a zjednodušení. GLM je poměrně velmi širokou třídou modelů s jednotnou statistickou teorií i velmi univerzální výpočetní implementací, s jejíž pomocí lze provádět rozbor tak rozdílných dat, jako jsou např. měření hmotnosti, koncentrace, počty jedinců nebo relativní četnosti nějakého jevu. Z formálního hlediska se budeme zabývat pouze **jednorozměrnými** GLM modely (resp. statistickými metodami na nich založenými), a to takovými, které jsou vhodné pouze pro nezávislá měření (pozorování). Pro analýzu dat, ve kterých existuje nějaká korelace (např. díky tomu, že jde o opakovaná měření na stejných jedincích), je potřeba použít modelů a metod poněkud jiných, složitějších. Těm se věnujeme ve druhém a třetím dílu (Pekár & Brabec 2012, 2019).

Jak jsme již uvedli, touto knihou se snažíme pomoci praktickému uživateli statistiky při sestavování a používání statistického modelu – a tedy i při výběru metod vhodných pro analýzu různých konkrétních typů dat. Zároveň bychom ho chtěli motivovat k tomu, aby vyhledal pomoc u profesionálního statistika, pokud se dostane do situace, ve které se necítí jistý a kterou by nemusel zvládnout. Věci totiž často nejsou tak snad-

né, jak by se na první pohled mohlo zdát. Stejný soubor dat lze analyzovat pomocí různých metod – záleží třeba na cílech, které sledujeme. V datech se totiž ukrývá velké množství informací, ale nás z nich v danou chvíli zajímají pouze některé. A proto výběr metody závisí na tom, který aspekt reality je pro nás důležitý a od kterých detailů jsme ochotni abstrahovat. Ale i pro analýzu jednoho konkrétního aspektu lze často použít několik postupů (s různými vlastnostmi). Zde budeme v drtivé většině případů prezentovat pouze jeden, aniž bychom tím popírali existenci jiných přístupů. Jinak by se totiž objem textu podstatně zvětšil.

Podobný přístup jsme zvolili i v popisu jazyka **R** a v pojednání o tom, jak zacházet s jeho objekty. Jazyk **R** je nesmírně bohatý. Stejného výsledku lze často dosáhnout několika způsoby. Nemáme zde místo k tomu, abychom je všechny vyjmenovávali (zájemce může konzultovat příslušné manuály). Zvolili jsme ty, které se nám zdály nejjednodušší a pro začínajícího uživatele nejpraktičtější.

1.1 Jak číst tuto knihu

V textu knihy se kombinují ukázky vybraných statistických metod a popis ovládání jazyka **R** jako statistického programového prostředí. Obojí je pak ilustrováno na praktickém řešení konkrétních příkladů.

Jak číst tuto knihu? Záleží na vašich předchozích znalostech a zkušenostech s analýzou dat a s programem **R**. Ti z vás, kteří od absolvování kurzu statistiky/biostatistiky žádnou (nebo skoro žádnou) analýzu dat nedělali a s **R** nikdy nepracovali, by si měli knihu přečíst popořadě, od začátku až do konce. Vy, kteří **R** již trošku znáte a regresi občas používáte, můžete číst trochu na přeskáčku – vybírat si kapitoly, které vás právě zajímají.

V první kapitole je nejdůležitější následující část (kap. 1.2), která pojednává o definici proměnných. Tu rozhodně nevynechávejte ani v případě, že se domníváte, že v tom, jak rozlišovat proměnné, máte jasno. Naše definice se totiž mohou mírně lišit od těch, s kterými jste se mohli setkat v jiných knihách.

Kapitola 2 popisuje instalaci a ovládání softwaru **R**, který budeme v této knize k analýze dat používat. Pokud jste s **R** doposud samostatně nepracovali, pak je tato kapitola určena právě vám. Kromě samotné instalace programového balíku se v ní naučíte (některé) obecné zásady práce v prostředí **R**, ale také důležité konkrétní příkazy používané později v následujícím textu. Dále je zde uvedeno, jakým způsobem se do softwaru vkládají data. To vše je zcela nezbytnou podmínkou dalšího úspěšného používání programu.

Kapitola 3 již nepojednává jen o programu **R**, ale také o tzv. explorativní analýze dat. Tou se myslí jednak výpočet základních statistických charakteristik zkoumaných souborů a jednak jejich neformální srovnání pomocí tabulek grafů a dalších nástrojů. Naleznete zde obecný popis příkazů, jimiž se vytváří (téměř) všechny grafy, se kterými budeme pracovat v pozdějších kapitolách. Tato kapitola uzavírá základní obeznámení s programem **R**.

Následující čtyři kapitoly (kap. 4–7) jsou zaměřené spíše obecněji. Zároveň patří k těm nejdůležitějším. Je tomu tak proto, že jejich obsah využijete k analýze dat, i když používáte jiný software, nebo např. při obecných úvahách o strategii svých analýz. Konkrétně v kap. 4 se zmíníme o práci se statistickými modely toho typu, který se bude v knize dále vyskytovat. Formulace modelu (tradičně v matematickém zápisu) je totiž základním kamenem, na němž je statistická analýza postavena. Řeč bude o regresních modelech a o tom, co je to GLM.

Kapitola 5 je první ukázkou konkrétní analýzy dat. Postup řešení je podrobně rozebrán, okomentován (s odkazy na základní pravidla, kterými by se analýza měla řídit). Kromě toho jsou zde popsány a interpretovány některé **R** výstupy a uvedeny důležité definice (třeba kontrastů).

Kapitola 6 pak již konkrétněji pojednává o různých typech analýz v rámci GLM. Popisuje základní typy modelů v závislosti na charakteru vysvětlujících proměnných. Především ale ukazuje, jak přepsat matematický model do jazyka programu **R**, a naznačuje, jak jej po analýze interpretovat.

Kapitola 7 plní funkci jakéhosi (velmi jednoduchého) „určovacího klíče“, na jehož základě se může tápající čtenář (začátečník) rozhodnout pro konkrétní metodu (uvidíte sami, že s přibývajícími znalostmi a zkušenostmi bude takové rozhodování zahrnovat stále více přemýšlení a méně „předpisových“ šablon). Předpokládáme, že čtenář se bude ke knize vracet jako k pomůcce při pozdějších samostatných analýzách vlastních dat. Pokud si pamatuje obecné znalosti z kapitol 1–6, může začít rovnou kap. 7, která jej odkáže na kapitolu, v níž nalezne řešení příkladu podobného tomu jeho. Nemůžete-li se pro žádnou rozhodnout, čtěte dál v pořadí, v jakém jsou kapitoly v knize uvedené. Avšak napoprvé rozhodně doporučujeme přečíst všechny kapitoly 8–12.

Kapitoly 8–12 převážně sestávají z několika (podrobně okomentovaných) příkladů s vysvětlením některých obecných a teoretických pojmů, na které narazíte v průběhu analýzy. Analýza každého příkladu je vypracována podle obdobného plánu. Tyto kapitoly jsou psány tak, aby byly na sobě nezávislé (daly se číst samostatně). V důsledku toho se s některými stejnými nebo podobnými problémy setkáte opakovaně. Vybranou kapitolu byste si měli přečíst vždy celou, nejen jeden konkrétní příklad.

Na konci uvádíme seznam použité a doporučené literatury. Obsahuje jak tuzemské, tak zahraniční tituly. Vesměs jsou to publikace vztahující se buď přímo k **R**, anebo statistické texty s převahou těch, které se víceméně věnují aplikacím statistiky v biomedicínské oblasti. Na úplný závěr jsou zařazeny rejstříky: a to jak funkcí a argumentů prostředí **R**, tak obecných termínů. Index je záměrně dosti podrobný, aby umožnil snadné vyhledávání požadovaných termínů.

1.2 Typy proměnných

Definice a podrobný rozbor vlastností různých typů proměnných tvoří náplň základního statistického kurzu, který zde samozřejmě nemíníme celý zopakovat. Připomeňme si proto jen pár důležitých skutečností. Existuje několik hledisek, podle kterých lze proměnné klasifikovat (viz Hebák et al. 2004). Pro naši potřebu si vystačíme s následujícím:

Závislá proměnná (response variable) je taková, jejíž variabilitu se snažíme vysvětlit. Jinak řečeno, je to ta proměnná, kterou chceme modelovat v závislosti na jedné nebo vícero proměnných vysvětlujících. V této knize se budeme zabývat jednorozměrnými modely – modelovat tedy budeme vždycky pouze JEDNU náhodnou závislou proměnnou (zobecnění existují, viz např. Yee (2015), ale my se jimi v této knize zabývat nebudeme). Podle konkrétního typu GLM modelu však bude závislá proměnná buď spojitá, nebo diskrétní, ale vždycky numerická (hodnotami jsou čísla). V grafech bude vždycky na svislé ose.

Vysvětlující proměnná (explanatory variable) je ta, kterou vysvětlujeme hodnoty závislé proměnné. V jednorozměrných modelech může být jedna závislá proměnná modelována jednou nebo vícero vysvětlujícími proměnnými. Ty mohou být numerické nebo kategorické (hodnotami jsou znaky či znakové řetězce odpovídající kódovému označení skupin/kategorií). Numerické proměnné mohou být spojité, nebo diskrétní. **Spojitou** vysvětlující proměnnou budeme značit malými písmeny, např. x. Její hodnotu pro i-té pozorování budeme označovat pomocí indexu i, tedy x_i. Spojitou vysvětlující proměnnou budeme označovat jako **kovariátu** v případech, kdy je v modelu zároveň přítomna i nějaká proměnná kategorická. **Kategorickou** proměnnou budeme označovat velkými písmeny, například A. Kategorická proměnná nabývá vždy alespoň dvou různých hodnot, které označujeme jako úrovně (např. samec a samice). j-tou úroveň označujeme indexem j, např. A_j. Kategorickou vysvětlující proměnnou budeme označovat také jako **faktor**, a to ve smyslu, v jakém se zavádí v analýze rozptylu (pozor na správný zápis: v angličtině, a tedy i **R** syntaxi jde o factor). Speciálním typem kategorické proměnné je **uspořádaný** (ordered) faktor, u kterého se úrovně dají smysluplně seřadit (např. malý, menší, nejmenší). To dělá faktor podobný spojité vysvětlující proměnné, akorát u něj nevíme, jak daleko od sebe jsou jednotlivé úrovně.

Předpokládá se, že stejně daleko (což nemusí být vždy pravda). Přesto se této vlastnosti v analýze někdy využívá.

Váhy jsou speciálním případem proměnné. Určují relativní váhu jednotlivých pozorování v analyzovaném souboru. Implicitně analytické funkce v **R** používají stejné váhy pro všechna pozorování. Externě zadané váhy jsou užitečné v případě, kdy takové schéma nevyhovuje a chceme ho změnit – např. na základě známých relativních přesností jednotlivých pozorování. Váhy musí nabývat nezáporných hodnot. Nulové váhy jsou ovšem poněkud kuriózní (vylučují dané měření z analýzy) a nedoporučujeme jejich použití (vybraná pozorování lze z analýzy vyloučit podstatně elegantnějším způsobem).

1.3 Konvence

Text knihy obsahuje několik typů písma. Využíváme je k odlišení základního textu knihy od příkazů (a jiných klíčových slov) jazyka **R**. Pokud uvádíme názvy příkazů a jejich argumentů, v textu používáme font **`Courier New`** tučný, o velikosti 10,5 bodů. Názvy objektů, které uživatelsky vytváříme v průběhu analýzy, jsou psány fontem `Courier New` obyčejný, o velikosti 10 bodů. Ostatní text je psán fontem Minion Pro, velikost 10,5. Názvy proměnných, hodnoty parametrů a matematické formulace jsou psány kurzivou, názvy úrovní faktorů ve strojopisných uvozovkách. Jména balíčků jsou podtržena.

K přepisu všeho, co se odehrává v oknech spuštěného prostředí **R**, používáme font `Courier New`, o velikosti 8 bodů. Pro lepší orientaci přitom rozlišujeme mezi uživatelem zadávanými příkazy, které píšeme tučně (např. **`a <- 1:5`**), a odpovědí programu v normálním stylu (např. `a`). Pro úsporu místa byly některé řádky odpovědí programu nahrazeny tečkami.

Pracovní grafy, tj. ty, které jsou vytvořené na začátku analýzy, byly vytvořeny pokud možno s použitím co nejmenšího počtu příkazů a argumentů, proto jim často chybí popisky, legendy apod. Teprve finální grafy obsahují všechny detaily (za cenu delší syntaxe).

Přirozený logaritmus (se základem *e*) se ve statistice používá velmi často. Budeme ho tedy zapisovat jako log. Jinde toto označení může mít význam jiný (např. v MS Excelu odpovídá logaritmu dekadickému – se základem 10), na to je třeba dát pozor. Navzdory českému prostředí jako oddělovač desetinných míst používáme tečku místo čárky. Konečně, většina čísel je v textu zaokrouhlena na 4 cifry.

2 STATISTICKÝ SOFTWARE

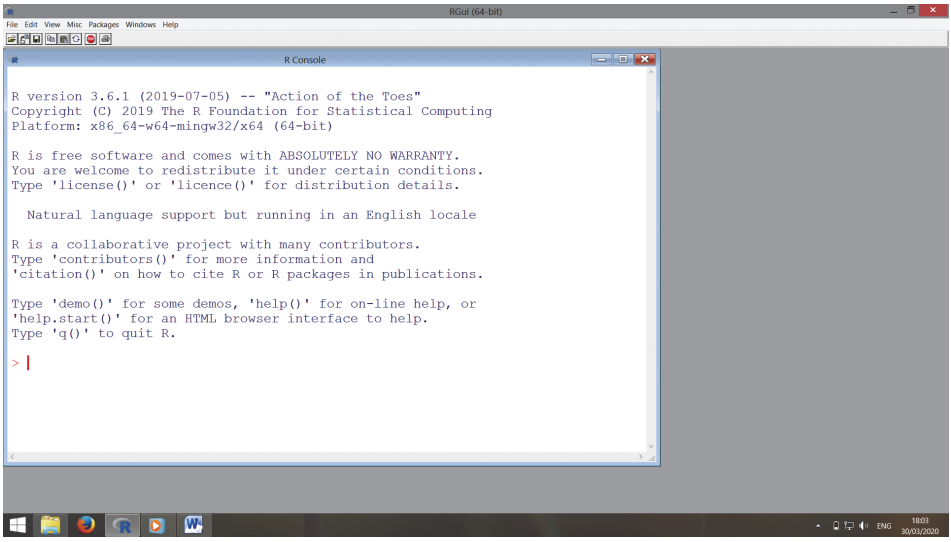
Na trhu je dostupných několik komerčních i nekomerčních programů na analýzu dat, lišících se kvalitou, rozsahem i cenou (viz přehled v knize Meloun & Militký 2004). V našich podmínkách se asi nejčastěji používá základní software, např. MS Excel. Pak jsou hojně využívané specializované programy, jako např. Statistica. A konečně zde existují obrovské (a drahé) balíky jako např. SPSS nebo SAS. Rozdíl mezi uvedenými kategoriemi je nejen v množství implementovaných typů analýz, ale také ve flexibilitě, tj. v možnostech programování uživatelem a jiných úprav „na míru". Jednoduchou analýzu lze samozřejmě provést takřka v kterémkoliv programu. Složitější metody jsou však dostupné pouze ve specializovaném softwaru.

2.1 Prostředí R

Dobrou zprávou je, že nejrozsáhlejší program pro moderní analýzu dat je dostupný zcela zdarma. Jmenuje se **R** a tato kniha je založena na jeho rozsáhlých možnostech. Ty ve skutečnosti pokrývají oblast *mnohem* širší, než prezentuje tato kniha.

Prostředí **R** bylo velice podobné komerčnímu programu S-Plus (© Insightful Corporation), který byl postaven na programovacím jazyku S, jenž byl vyvinut v 80. letech v amerických AT&T Bell Laboratories (a posléze skončil v úpadku). **R** používá dialekt jazyka S v kombinaci s jazykem Scheme, což znamená, že z naprosté většiny bylo ovládání programů **R** a S-Plus stejné. **R** byl vymyšlen novozélandskými autory Ihaka & Gentleman (1996). Dnes je spravován skupinou lidí z celého světa, kteří se sami označují jako R Core Team (R Core Team 2019). Díky tomu je vývoj **R** mnohem dynamičtější než vývoj jiných statistických softwarových balíků.

R není user-friendly program typu MS Excel nebo Statistika. Máme na mysli pěkná barevná okna, kde hlavním ovládacím prvkem je kurzor myši. Připravte se na to, že po otevření **R** vás uvítá prázdné šedé okno (obr. 2-1). Mírně přátelštější vzhled (plus mnoho uživatelsky i programátorsky užitečných prvků) můžeme (taktéž zdarma) získat v rámci integrovaného vývojového prostředí zvaného RStudio (viz kapitola 2.3), které Vám může práci s **R** zefektivnit – ale není to nikterak nutné. **R** se ovládá hlavně příkazy, které zadáváme z klávesnice.



Obr. 2-1 Vzhled hlavního okna programu **R** s příkazovým oknem.

Ptáte se, proč jsme zvolili zrovna takový „nepřátelský" program? Máme k tomu několik důvodů:

- **R** obsahuje všechny zásadní typy moderních analýz, které jsou díky aktivitě mnoha lidí po celém světě neustále doplňovány. Komerční statistický software je zpravidla uzavřený (rozšířit jej lze až nákupem další verze), zatímco **R** je budován jako otevřený systém. To znamená, že nové a další metody lze snadno kdykoliv (zdarma) doplnit.
- Ovládání tohoto programu není (pro nestatistika) tak zcela jednoduché. To však může paradoxně být výhodou. Nutí totiž uživatele osvojit si určité znalosti a o tom, co dělá, (alespoň) trochu přemýšlet. Analýza dat v komerčních programech může být poněkud „nebezpečná". Umožňuje totiž i zcela neznalému jedinci dopracovat se víceméně náhodným klikáním k nějakému výsledku.
- „Přátelské" statistické programy vás doslova zahltí množstvím informací, které na vás po analýze vychrlí. Pak pro vás nemusí být snadné se v nich vyznat. Filozofie **R** je zcela jiná – zobrazit v podstatě pouze to, oč uživatel požádá prostřednictvím vydaných příkazů. Tento přístup vychází z předpokladu, že člověk použije pouze příkazy, o jejichž funkci něco ví. Nebo si dohledá potřebné příkazy, čímž je dále stimulován k prostudování dané problematiky. Implicitně poskytované výstupy **R** jsou typicky docela skromné. A to je dobře. Jak sami uvidíte, lépe se v nich orientuje a detaily je možné si od programu explicitně vyžádat později.

- Silnou stránkou **R** je moderní grafika, zaměřená na přehlednou a efektivní prezentaci dat.

Věřte, že **R** není nepřátelský program. Sami později uvidíte, že jeho ovládání je koneckonců docela snadné a hlavně velice efektivní. K překonání počátečních problémů Vám může pomoci aplikace R Commander (viz kap. 2.3), která se ovládá poněkud lépe (i myší). Abychom vám práci s **R** usnadnili, volání různých procedur (i jejich výstupy) jsme podrobně popsali a vysvětlili.

Jak jsme uvedli výše, **R** je prostředí pro manipulaci s objekty. Objekty jsou např. data a manipulací se rozumí matematické a statistické výpočty, tvorba tabulek a grafů. Ve skutečnosti je **R** mnohem více než jen statistický software: je to výkonný programovací jazyk (pro objektové programování), podobný jazykům C++, Python nebo Java. Nedílnou součástí úspěšného používání **R** je osvojení si základních příkazů používaných k ovládání. Seznámení se se základními příkazy je hlavní náplní této kapitoly. Nejprve si však program **R** musíte nainstalovat.

2.2 Instalace a ovládání **R**

Instalační soubor programu **R** naleznete na internetu na adrese http://www.r-project.org/. Dostanete se k němu přes dialog Download R, kde se vám otevře seznam serverů, ze kterých je možné soubor stáhnout. Zvolíte-li třeba rakouský server (Austria), zobrazí se okno, kde si vyberete platformu, ve které budete s **R** pracovat: Linux, Mac OS X či Windows. Všechny výpočty v této knize byly uskutečněny ve verzi pro Windows (ovládání **R** v alternativních prostředích je velice podobné). Nás teď zajímá instalační soubor. Nové verze softwaru se objevují docela pravidelně v několikaměsíčních intervalech. V době psaní této knihy byla k dispozici jako nejčerstvější verze 3.6.1. Instalační soubor měl tudíž název **R-3.6.1-win.exe**. Tato verze má v sobě zakomponovaných 27 základních balíčků (implementujících různé statistické metody i jiné výpočty). Existuje řada dalších, které obsahují doplňkové metody a další užitečné komponenty (v době psaní knihy jich bylo kolem 15 000). Dostanete se k nim kliknutím na Packages vlevo na liště stránek. Objeví se seznam se stručným popisem funkcí každého balíčku. Máte-li o některý zájem, můžete si ho (samozřejmě zdarma) stáhnout. Jak se to dělá, si řekneme později.

Po instalaci softwaru standardním způsobem se vám na ploše objeví ikona s názvem **R 3.6.1**. Po spuštění programu se otevře (velké) hlavní okno a v něm (menší) okno příkazové (obr. 2-1) čili konzole (R Console). Na horní liště hlavního okna je několik základních příkazů přístupných buď přes rozbalovací menu, nebo tlačítka.

Popíšeme si funkce těch nejdůležitějších. V položce **File** naleznete základní volby jako **Display file(s)...** pro zobrazení souborů v adresáři R-3.6.1, který je umístěn pod

c:\Program Files\R\. To je standardně přednastavený pracovní adresář. Je vcelku pohodlné skladovat v něm data, protože pak při jejich čtení/ukládání nemusíte definovat úplnou cestu. Pokud však chcete změnit pracovní adresář pouze na chvíli, např. po dobu jedné session (tedy jednoho spuštění programu), lze tak učinit pomocí volby **Change dir...** zevnitř spuštěného **R** (v položce **File** hlavního menu).

Volbou **Load Workspace...** (opět z položky **File** hlavního menu) můžete načíst předešlou práci, pokud jste si ji na závěr předchozí session uložili za pomoci volby **Save Workspace...** Tyto možnosti jsou vhodné, především pokud pokračujete v analýze třeba druhý den. Abyste nemuseli všechno dělat od začátku, stačí načíst to, co jste udělali již dříve. Pozor! Uloží, respektive načtou se pouze příkazy a (výsledné) objekty, ne výstupy a grafika. To znamená, že když se chcete podívat na výstup z některého minulého příkazu, musíte si jej znovu vyvolat a spustit (pokud jste si ovšem textový výstup již dříve nezkopírovali např. do textového souboru pomocí Copy a Paste).

Položka **Edit** obsahuje funkce pro kopírování a vkládání, tak jak jsme zvyklí z jiných programů. Dále je tu volba pro vyčistění okna konzole **Clear console**, volba pro otevření tabulkového editoru **Data editor...** a volba pro úpravu vzhledu programu **GUI preferences...** V ní lze změnit velikost oken, typ písma, barvu pozadí a písma atp.

V položce **Misc** naleznete užitečnou volbu **Stop current computation** pro přerušení výpočtu (např. když se výpočet z nějakého důvodu „kousne“ nebo prostě proto, že jste si pozdě uvědomili, že jste zadali vstup chybně apod.). Lze to ale udělat i rychleji – stiskem klávesy ESC nebo kliknutím na tlačítko se značkou STOP. Volba **Remove all objects** se používá pro odstranění všech v současné době definovaných nebo načtených objektů.

Položka **Packages** obsahuje volby pro práci s doplňujícími balíčky, např. těmi, které obsahují doplňující funkce. Balíčky (s výjimkou několika základních) totiž nejsou automaticky načteny do paměti po spuštění programu. To proto, aby operační paměť nebyla zbytečně zahlcena funkcemi, které zrovna nepotřebujeme. Když tedy chceme použít nějakou funkci obsaženou v určitém balíčku, je potřeba konkrétní balíček načíst. Např. volbou **Load package...** Pokud ovšem požadovaný balíček není ve vašem počítači, musíte ji napřed nainstalovat. Procvičíme si to na instalaci balíčku visreg (Breheny & Burchett 2017), kterou budeme používat pro tvorbu některých grafů. Jste-li právě připojeni k internetu, klikněte na volbu **Install package(s)...** a ze seznamu vyberte server, ze kterého budete stahovat, třeba opět Austria (je dobré volit geograficky relativně blízký server, jinak si s volbou není třeba dělat starosti – jednotlivé servery se totiž navzájem „zrcadlí“ – poskytují totéž). Tím se váš počítač připojí k příslušnému serveru a zobrazí vám seznam všech v současnosti dostupných balíčků. Pokud Váš počítač není připojen k internetu, musíte si stáhnout zazipovanou podobu balíčku na výše uvedené adrese zvlášť a nainstalovat balíček ze zipového souboru ručně. Mějte přitom na paměti číslo verze **R**, kterou používáte, protože balíčky mohou být

specifické pro různé verze (může se tak stát, že verze balíčku, který zrovna instalujete, vyžaduje novější verzi R než tu, kterou používáte, a bude tak třeba R upgradovat). Zazipovaný soubor pak nainstalujete pomocí volby **Install package(s) from local files...** Nezapomeňte, že k aktivaci funkcí již nainstalovaného balíčku je potřeba jej načíst znovu při každé session.

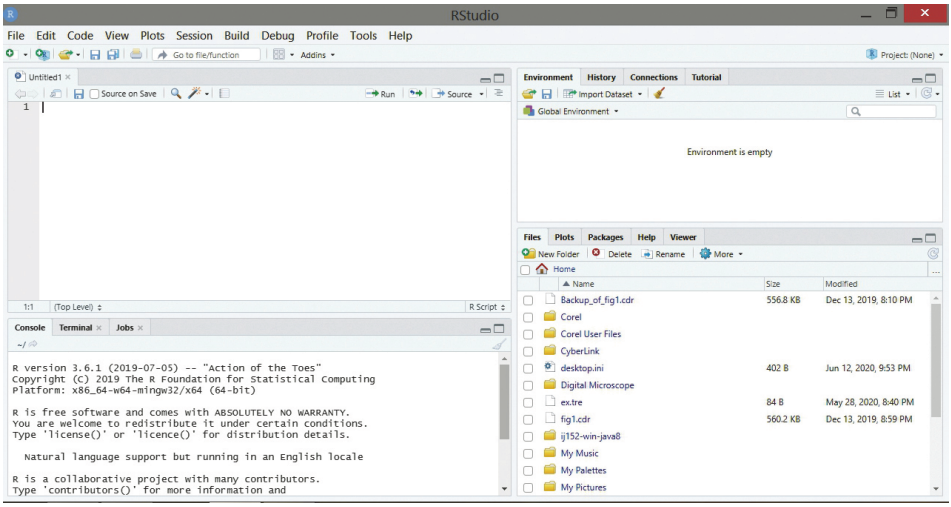
V položce **Windows** jsou volby pro přepínání mezi příkazovým a grafickým oknem. Nejpraktičtější je asi použití **Tile**, tedy dvou oken vedle sebe.

Konečně pod **Help** je mnoho užitečných voleb. Volba **Console** vám vysvětlí, jak se ovládá okno konzole. Příkazy se píšou za zobáček (či prompt) > a jsou standardně rudé, zatímco odpověď programu je psána modrým písmem. Lze je psát buď zvlášť do řádku, nebo ve stejném řádku za sebou, oddělené středníkem. Mezi příkazy oddělené středníkem není nutné (ale je povoleno) vkládat mezery. Předcházející příkazy lze vyvolat jeden po druhém stiskem klávesy s šipkou nahoru a zpět zase stisknutím šipky dolů. Používání šipek velice usnadňuje práci. Když např. uděláte chybu v příkazu, lze si jej zpětně vyvolat šipkou nahoru a přepsat jej. V právě editovaném řádku se pohybujeme šipkami vpravo a vlevo.

Základní informace o programu se dozvíte z **FAQ on R** nebo z **Manuals (in PDF)**. Velice užitečná je volba **R functions (text)...**, která zobrazí detailní popis funkce, jejíž jméno zadáte. Lze tu nalézt popis toho, k čemu daná funkce slouží, seznam všech jejích argumentů, její povolené hodnoty, ale často i odkazy na literaturu (týkající se metody, na které je funkce založena) a pár příkladů. Tento příkaz funguje jenom pro funkce, které jsou aktuálně načteny. Na popis všech funkcí, tedy i těch, které jsou v nenačtených, ale nainstalovaných balíčcích, se lze podívat přes **Html help**. Otevřou se stránky se všemi základními informacemi o R včetně funkcí obsažených v předinstalovaných balíčcích. Pokud nevíte název funkce, ale víte, co dělá, můžete ji zkusit vyhledat volbou **Search help**. Například pokud chcete najít funkci, která počítá Shapiro-Wilkův test, napište klíčové slovo „shapiro“. Program pak prohledá všechny nainstalované balíčky a vypíše ty funkce, které ve svém jménu nebo popisu obsahují dané slovo. Před jménem funkce je uveden název balíčku, v němž je funkce uložena. V našem případě má funkce hledaného testu název `shapiro.test` a je uložena v balíčku stats.

2.3 R Studio

Jak jsme popsali výše, práce v programu R je podobná jiným programovacím jazykům, a proto odlišná od ovládání komerčního softwaru typu Statistica. V roce 2011 se však objevila nadstavba jménem **R Studio** (R Studio Team 2015), která mnoha uživatelům zpříjemňuje používání. Za ní následoval v roce 2016 **R Commander** (Fox 2017). R Studio lze stáhnout z adresy https://rstudio.com/products/rstudio/, R Com-



Obr. 2-2 Vzhled okna programu R Studio.

mander je dostupný jako balíček Rcmdr. Obě nadstavby umožňují uživateli mít lepší kontrolu nad použitými příkazy a získanými výsledky. Práce s **R** se tím více podobá modernímu uživatelsky příjemnému prostředí.

Mezi biology je rozšířenější R Studio. Proto se na něj zaměříme. V R Studio je okno rozdělené na čtyři podokna (obr. 2-2). V prvním je editor příkazů, který obsahuje i kontrolu syntaxe, což je užitečné zvláště tehdy, když jsou příkazy dlouhé. Druhé je pracovní okno (Workspace), které ukazuje seznam objektů, historii a průzkumníka souborů. Třetí je konzola a čtvrté okno je grafické, ale také obsahuje seznam balíčků a Help soubory. Grafické okno poskytuje více formátu pro export grafů než **R**.

My zde R Studio používat nebudeme, protože jeho použití výklad praktických příkladů nijak nevylepší (může ale zvýšit efektivitu Vaší práce poté, co se se základními principy **R** seznámíte).

2.4 Základní operace

Možnosti **R** jsou opravdu rozsáhlé – program obsahuje tisíce příkazů. Samozřejmě je zde nelze všechny uvést a vysvětlit. Soustředíme se pouze na ty nejzákladnější a ty, s nimiž se v této knize potkáte nejčastěji. Některé další budou představeny v kontextu jednotlivých příkladů dále. Stručný seznam těch nejdůležitějších příkazů s názvem **R** Reference Card lze nalézt na http://cran.r-project.org/doc/contrib/Short-refcard.pdf. Je to takový čtyřstránkový tahák.

Přejdeme k jednotlivým operacím. Začneme jednoduchými manipulacemi a pak budeme pokračovat složitějšími postupy. Nejprve program použijeme jako vědeckou kalkulačku. Stačí do příkazového okna napsat `2 + 5`, stisknout klávesu ENTER a program odpoví výsledkem zobrazeným na novém řádku (který pro snazší orientaci ve výstupech označí `[1]`). Základní matematické operátory jsou: sčítání (**+**), odečítaní (**-**), násobení (*****), dělení (**/**), mocnina (**^**). Logické operátory mají následující tvar: méně než (**<**), více než (**>**), rovný (**==**), nerovný (**!=**), méně nebo rovno (**<=**), více nebo rovno (**>=**). Jejich výsledkem není číslo, ale logická hodnota: buď pravda (**TRUE**, ve zkratce **T**), nebo nepravda (**FALSE**, zkráceně **F**).

Jména používaná pro volání matematických funkcí v **R** jsou většinou velmi intuitivní a snadno zapamatovatelná, např. absolutní hodnota (**abs**), logaritmus se základem *e* (**log**), logaritmus se základem 2 (**log2**), logaritmus se základem 10 (**log10**), exponencování (**exp**), sinus (**sin**), kosinus (**cos**), tangens (**tan**), arkus sinus (**asin**), arkus cosinus (**acos**), arkus tangens (**atan**), součet zadané řady čísel (**sum**), součin zadané řady čísel (**prod**). Příkaz pro druhou odmocninu je **sqrt**. Jiné odmocniny je však třeba přepsat jako mocniny (funkce mocniny **^** je obecná a dovoluje i záporné a pro nezáporné argumenty také neceločíselné exponenty). Tyto jednoduché funkce se volají svým názvem, po němž následuje číslo v závorce, jak je vidět v následujících příkladech, které uvádíme pro procvičení:

```
> 3*2
[1] 6
> 3^4
[1] 81
> sqrt(9)
[1] 3
> 8^(1/3)
[1] 2
> 3==4
[1] FALSE
> log(10)
[1] 2.302585
> log10(10)
[1] 1
> exp(2)
[1] 7.389056
> prod(2, 3, 4)
[1] 24
```

Odpovědi na všechny příkazy program pouze zobrazil. Nelze s nimi dále přímo pracovat. Lze je samozřejmě kopírovat a přenést na aktuální příkazový řádek pomocí Copy a Paste. To může být nepohodlné, zejména pokud chceme s výsledkem provést další operace. V takovém případě je potřeba uložit si výsledek do objektu (a jeho název pak použít v závorce při volání nějaké další funkce). Tím nejjednodušším objektem je vektor s jediným prvkem neboli skalár. Vytvořit vektor je snadné. Zvolíme jeho

jméno, třeba `a`, za ním napíšeme šipku **`<-`**, která je vytvořena ze dvou znaků (menší než a pomlčka), a pak napíšeme hodnotu, kterou do něj chceme uložit, třeba `8`. Obsah vektoru zobrazíme napsáním jeho jména do nového řádku, nebo do stejného řádku za středník. Pro vytvoření delších vektorů musíme použít funkci **`c`**. Ta spojí všechny zadané hodnoty (uvedené v závorce a oddělené čárkami) do vektoru:

```
> a <- 8; a
[1] 8
> b <- c(2, 1/2, 95); b
[1] 2.0  0.5 95.0
```

Jméno objektu může být (téměř) libovolné, ale je dobré mít na paměti pár zásad. **R** rozlišuje ve všech jménech (i klíčových slovech apod.) mezi malými a velkými písmeny (takže napsáním `b` a `B` voláme *různé* objekty). Jména nesmí začínat číslicí ani speciálními znaky (jako podtržítko, čárka apod.). Některé znaky a slova je lepší nepoužívat. Jde o standardní **R** příkazy a jiné výrazy, které mají v programu nějaký speciální význam, např. **`break`**, **`c`**, **`C`**, **`D`**, **`diff`**, **`else`**, **`F`**, **`FALSE`**, **`for`**, **`function`**, **`I`**, **`if`**, **`in`**, **`Inf`**, **`mean`**, **`NA`**, **`NaN`**, **`next`**, **`NULL`**, **`pi`**, **`q`**, **`range`**, **`rank`**, **`repeat`**, **`s`**, **`sd`**, **`t`**, **`tree`**, **`TRUE`**, **`T`**, **`var`** a **`while`**. Pokud je použijete, bude jejich původní funkce maskována (tiše nahrazena), což může způsobit řadu později obtížně lokalizovatelných problémů. Proto naše důrazná rada zní: raději se jim vyhýbejte.

Hodnoty vektoru mohou být různých typů. Numerické (numeric), např. `c(1.5,20,-3.1)`, logické (logical), např. `c(TRUE,FALSE,TRUE)`, nebo znakové řetězce (character). Jednotlivé hodnoty znakové proměnné se vždy vkládají do strojopisných uvozovek: např. `c("blue", "red", "green")`. Vektor by měl vždy obsahovat hodnoty stejného typu. Bude-li složen z různých typů, např. čísel a znaků, zkonvertuje se automaticky na obecnější znakový typ. S proměnnou (nebo vektorem) typu charakter pak nelze provádět matematické operace. To je dobré mít na paměti při hledání chyb a důvodů, proč to nefunguje.

Numerický vektor lze vytvořit vložením konkrétních libovolných čísel nebo pomocí dvojtečky, která požaduje vytvořit (celočíselnou) posloupnost od:do.

```
> y <- 1:11; y
[1]  1  2  3  4  5  6  7  8  9 10 11
```

Stejná sekvence se dá vytvořit i jiným postupem – příkazem **`seq`**. Tento příkaz již není tak jednoduchý jako příkazy předešlé. Umí toho totiž více – dokáže vytvořit posloupnost, ve které nejsou jen celá čísla. K tomu, aby fungoval, je potřeba specifikovat jeho **argumenty**. A to ty, které potřebujeme ke zvolenému cíli. Konkrétně počáteční (**`from`**), konečnou hodnotu (**`to`**) a délku kroku (**`by`**). Typické pro funkce v **R** je, že obsahují několik argumentů, které mají svá jména, resp. předdefinovanou pozici (existují i funkce s argumenty, které jména nemají, těmi se teď ale zabývat nebudeme).

To nám dovoluje buď specifikovat argumenty užitím jejich jmen v libovolném pořadí, anebo beze jmen, s dodržením pořadí. Druhá možnost je ekonomičtější s ohledem na psaní, proto ji zde upřednostňujeme. Tato volba však s sebou nese jedno podstatné nebezpečí. Pořadí argumentů není zcela kodifikováno a v principu se může v průběhu vývoje jazyka **R** změnit. Proto je zcela nezbytné při použití jiné verze **R**, než je tato, zkontrolovat pořadí argumentů pro všechny používané funkce. Anebo se naučit psát příkazy se jmény argumentů. My budeme uvádět jména argumentů vždy při prvním použitím v dané funkci.

Vraťme se k příkazu **seq**, abychom vytvořili sekvenci od 1 do 2 s krokem 0.1:

```
> x <- seq(from=1, to=2, by=0.1); x
[1] 1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0
```

Zkuste si pohrát s argumenty tohoto příkazu – vynechte jejich jména a prohoďte jejich pořadí – abyste poznali, k jak dramatické změně dojde.

Z takového vektoru lze vybírat prvky. Nejjednodušeji zadáním pozice prvků v hranaté závorce. Jedno číslo lze napsat přímo jako argument. Několik čísel je potřeba spojit dohromady, buď jako sekvenci pomocí dvojtečky, nebo jako vektor pomocí **c**. Takže chceme-li vybrat nejprve třetí, pak třetí až pátý a nakonec šestý, osmý a devátý prvek z vektoru `x`, uděláme to následovně:

```
> x[3]
[1] 1.2
> x1 < -x[3:5]; x1
[1] 1.2 1.3 1.4
> x2 <- x[c(6,8,9)]; x2
[1] 1.5 1.7 1.8
```

Vybírat můžeme nejen podle pozice, ale i podle jiné (složitější) podmínky. Chceme-li zjistit, které prvky ve vektoru jsou větší než konkrétní hodnota, použijeme příkaz **which**, jenž nám vrátí pořadová čísla prvků, které zadanou podmínku splňují:

```
> which(x>1.5)
[1]  7  8  9 10 11
```

S vektory lze provádět mnohé další operace. Můžeme je spojovat do delšího vektoru (třeba spojit `x1` a `x2` do `x12`), anebo upravovat pomocí matematických funkcí uvedených výše.

```
> x12 <- c(x1, x2); x12
[1] 1.2 1.3 1.4 1.5 1.7 1.8
```

Mnohé matematické funkce jsou v **R** „vektorizovány“, tj. napsány tak, že funkce je automaticky aplikována na všechny prvky zadaného vektoru. Podívejme se například na výpočet třetí mocniny (pro 11 čísel uložených ve vektoru `y` najednou):

```
> y1 <- y^3; y1
[1] 1    8   27   64  125  216  343  512  729 1000 1331
```

Mnohdy nás zajímá délka nějakého vektoru (tedy kolik má prvků). K tomu slouží intuitivně nazvaná funkce **`length`**. Pro představu délku vektoru `y` zjistíme následovně:

```
> length(x=y)
[1] 11
```

Povšimněme si, že tato funkce, stejně jako předešlé, zpracovala vektorový argument **`y`**. Pro někoho možná poněkud matoucí zápis **`x=y`** vzniká tak, že vektor **`y`** zadáváme jako argument funkce **`length`**, který se jmenuje **`x`**. **R** funkce může totiž mít vícero argumentů, a tak je leckdy nutné specifikovat, který že to chceme právě zadat. Různé funkce mohou mít různé typy argumentů. Obecně může tak být argumentem nejen konkrétní proměnná, vektor, skalár, ale i jméno nějaké další funkce, které takový argument připraví (jakoby uvnitř původně volané funkce). Např. místo dvou příkazů

```
> a1 <- y^2; prod(a1)
[1] 1.593351e+15
```

stačí napsat pouze jeden složitější:

```
> prod(y^2)
[1] 1.593351e+15
```

Možnost takovéhoto (i opakovaného) vnoření patří k silným stránkám prostředí **R** – umožňuje rychle psát zadání i pro docela složité výpočty. Nevýhodou vnořování je (zejména pro začátečníka) jistá nepřehlednost, proto ho budeme používat zřídka.

Někdy je např. zapotřebí numerický vektor standardizovat, což znamená odečíst od původních hodnot průměr (tzv. centrování) a pak je vydělit směrodatnou odchylkou (tzv. škálování). Standardizace se používá tehdy, když potřebujeme proměnné (vysvětlující nebo závislé) sjednotit, např. co do jejich rozsahu typicky proto, že byly měřeny na jiné škále. S takovou situací se potkáme v kap. 8-6.

Z definice je průměr standardizované verze původních dat roven nule a rozptyl jedné. Lze to velmi snadno provést pomocí příkazu **`scale`**. Jeho první argument (**`x`**) specifikuje vektor, který chceme zpracovat. Další dva argumenty (**`center`** a **`scale`**) specifikují, zda požadujeme (**`TRUE`**), nebo nepožadujeme (**`FALSE`**) jednotlivě centrování a škálování. Implicitně (tedy pokud argumenty **`center`** a **`scale`** neuvedeme

vůbec) je nastaven požadavek centrování i škálování. Průměr a rozptyl vektoru před a po standardizaci spočteme (pro kontrolu i procvičení) příkazy **mean** a **var**.

```
> mean(y)
[1] 6
> var(y)
[1] 11
> y2 <- scale(y); y2
            [,1]
 [1,] -1.5075567
 [2,] -1.2060454
 [3,] -0.9045340
 [4,] -0.6030227
 [5,] -0.3015113
 [6,]  0.0000000
 [7,]  0.3015113
 [8,]  0.6030227
 [9,]  0.9045340
[10,]  1.2060454
[11,]  1.5075567
attr(,"scaled:center")
[1] 6
attr(,"scaled:scale")
[1] 3.316625
> mean(y2)
[1] 0
> var(y2)
     [,1]
[1,]    1
```

Před standardizací byl průměr roven *6* a rozptyl *11*, po standardizaci je roven *0* a *1* (v souladu s definicí). Je to proto, že od všech hodnot ve vektoru `y` byla nejprve odečtena hodnota *6* a výsledek byl vydělen směrodatnou odchylkou *3.317*.

Dva (nebo více) vektorů stejného typu (např. numeric) lze spojit do matice pomocí dvou příkazů. Pokud chceme vektory skládat „nastojato" (jako sloupcové), použijeme příkaz **cbind**. Pokud „naležato" (jako řádkové vektory), tak **rbind**. Argumenty těchto funkcí jsou názvy spojovaných vektorů (nebo volání funkcí, které vektory vytvoří). Argumentů budeme zadávat tolik, kolik vektorů chceme spojovat. Pozor na správnou dimenzi – **cbind** i **rbind** mohou korektně spojovat jen vektory stejných délek (při zadání vektorů délek různých se ty kratší doplní na délku nejdelšího automaticky (bez varování!) – a to opakováním, což může být poněkud nebezpečné a vést ke špatně odhalitelným chybám)!

```
> a12 <- cbind(x1, x2); a12
      x1  x2
[1,] 1.2 1.5
```

```
[2,] 1.3 1.7
[3,] 1.4 1.8
> a13 <- rbind(x1, x2); a13
   [,1] [,2] [,3]
x1  1.2  1.3  1.4
x2  1.5  1.7  1.8
```

Výsledkem jsou matice, které obsahují i jména vektorů. Obecné matice lze sestavit mnohem efektivněji pomocí příkazu **matrix**. Jeho prvním argumentem (**data**) je vektor čísel, druhým (**nrow**) je počet řádků a třetím počet sloupců (**ncol**). Takže:

```
> b1 <- matrix(c(3,5,1,9,7,2), nrow=2); b1
     [,1] [,2] [,3]
[1,]    3    1    7
[2,]    5    9    2
```

Všimněme si především toho, že se data do matice načítají po sloupcích (to lze změnit pomocí argumentu **byrow=T**).

Když chceme spojit vektory různých typů, třeba numerický a znakový, musíme použít příkazu **data.frame**. Pokud bychom použili např. **cbind**, převedly by se všechny vektory automaticky do nejobecnějšího znakového formátu, čímž bychom ztratili numerické hodnoty. Příkaz **data.frame** se používá k vytváření obecnějších datových struktur. V prostředí **R** budeme nazývat datovou matici **data frame**. Ten má již definované vlastnosti proměnných a také přiřazená čísla pozorování. Zkusme sestavit takový data frame spojením dvou numerických vektorů `x` a `y` s vektorem `size`, který bude typu charakter. Musíme jej ale nejprve vytvořit. Pro vytváření vektorů, v nichž se některé prvky opakují, se používá funkce **rep**. Má dva argumenty: opakovanou hodnotu (**x**) – může jít i o vektor, a počet opakování (**times**). Ve vektoru `size` se bude opakovat 4krát slovo "small", za ním 3krát slovo "medium" a nakonec 4krát slovo "large". Připomínáme, že použití uvozovek je nutné vždy, když specifikujeme hodnoty typu charakter.

```
> size <- rep(c("small","medium","large"), c(4,3,4)); size
[1] "small"  "small"  "small"  "small"  "medium" "medium" "medium"
[8] "large"  "large"  "large"  "large"
> dat <- data.frame(x, y, size); dat
     x  y   size
1  1.0  1  small
2  1.1  2  small
3  1.2  3  small
4  1.3  4  small
5  1.4  5 medium
6  1.5  6 medium
7  1.6  7 medium
8  1.7  8  large
9  1.8  9  large
```

```
10 1.9 10  large
11 2.0 11  large
```

Tento data frame má 3 sloupce a 11 řádků, jak je patrné z výpisu (první sloupec ve výpisu obsahuje čísla řádků, které se generují automaticky a nepočítají se za sloupec). Sloupec jménem `size` obsahuje znakové řetězce. Je to vektor typu charakter. Není to ale ještě faktor připravený pro použití v modelech typu analýzy rozptylu apod., což zjistíme příkazem **`is.factor`** s jediným argumentem názvu zkoumaného vektoru. Skutečný faktor z něj uděláme příkazem **`factor`**, opět s názvem vektoru jako argumentem.

```
> is.factor(size)
[1] FALSE
> size <- factor(size)
```

Úrovně ve faktoru jsou (implicitně) uspořádány abecedně (vzestupně dle ASCII kódů a délek), jak můžeme zjistit příkazem **`levels`**. Pokud chceme, aby byla ve faktoru `size` první úroveň jiná, třeba "medium", použijeme příkaz **`relevel`** a referenční úroveň definujeme argumentem **`ref`**.

```
> levels(size)
[1] "large"  "medium" "small"
> size1 <- relevel(size, ref="medium"); levels(size1)
[1] "medium" "large"  "small"
```

Veškerý text z příkazového okna lze snadno kopírovat přes schránku (stiskem kláves CTRL a C) do jiných aplikací po označení do bloku kurzorem myši. Větší objekty, např. rozsáhlé matice, lze uložit do souboru specifikovaného formátu pomocí příkazu **`write.table`**. Jeho prvním argumentem (**`x`**) je název objektu, pak následuje cesta k místu uložení výsledného souboru (**`file`**) a nakonec oddělovač (**`sep`**). Např. **`\t`** specifikuje, že sloupce budou od sebe odděleny tabulátorem. Po spuštění příkazu

```
> write.table(dat, file="c:\\data.txt", sep="\t")
```

se na harddisku (C:) počítače objeví soubor data.txt obsahující vytvořený data frame.

Všechny objekty, které jsme vytvořili nebo načetli v průběhu jedné session, jsou uloženy do paměti. Jejich seznam zjistíme příkazem **`objects()`**. Když chceme, aby nám tyto objekty nepřekážely, je možné je všechny po ukončení analýzy odstranit příkazem **Remove all objects**. To také nyní udělejte. Ale pozor, neuložené objekty tak budou ztraceny!

```
> rm(list=ls(all=TRUE))
```