EKONOMICKÁ UNIVERZITA V BRATISLAVE
FAKULTA HOSPODÁRSKEJ INFORMATIKY

Evidenčné číslo: 103006/B/2014/3062656893

JAZYK R A JEHO VYUŽITIE PRI AKTUÁRSKYCH ANALÝZACH

Bakalárska práca

2014 **Peter Salinger**

EKONOMICKÁ UNIVERZITA V BRATISLAVE
FAKULTA HOSPODÁRSKEJ INFORMATIKY

JAZYK R A JEHO VYUŽITIE PRI AKTUÁRSKYCH ANALÝZACH

Bakalárska práca

Študijný program: Manažérske rozhodovanie a informačné technológie
Študijný odbor: 6258 Kvantitatívne metódy v ekonómii
Školiace pracovisko: Katedra matematiky a aktuárstva
Vedúci práce: Ing. Michal Páleš, PhD.

Bratislava 2014 **Peter Salinger**

Čestné vyhlásenie

Čestne vyhlasujem, že záverečnú prácu som vypracoval samostatne a že som uviedol všetku použitú literatúru.

Dátum: **.....**

(podpis študenta)

Poďakovanie

Vďaka patrí môjmu vedúcemu práce Ing. Michalovi Pálešovi, PhD., za cenné rady a pripomienky a odbornú pomoc, ktoré mi výrazne pomohli pri vypracovaní bakalárskej práce.

ABSTRAKT

SALINGER, Peter: *Jazyk R a jeho využitie pri aktuárskych analýzach*. – Ekonomická univerzita v Bratislave. Fakulta hospodárskej informatiky; Katedra matematiky a aktuárstva. – Vedúci záverečnej práce: Ing. Michal Páleš, PhD. – Bratislava, FHI EU, 2014, 83s.

Cieľom záverečnej práce je sprístupniť jazyk R používateľom. Práca je písaná jednoduchým spôsobom a používateľ, ktorý bude túto prácu využívať pri práci s jazykom R, nepotrebuje mať skúsenosti s programovacími jazykmi. Záverečná práca zoznamuje používateľa so základmi pre prácu s jazykom R a s ich ďalším využitím v rámci zložitejších operácií na úrovni programovacieho jazyka R. Práca je rozdelená do štyroch kapitol. Prvá kapitola je venovaná opisu programovacieho jazyka R a jeho využívaniu v súčasnosti v zahraničnej ale aj domácej sfére. V ďalšej časti sú charakterizované ciele záverečnej práce. Tretia kapitola opisuje spôsob, akým sa budú napĺňať stanovené ciele. Záverečná kapitola sa zaoberá prezentovaním používateľskej príručky podľa jednotlivých stanovených cieľov v druhej kapitole. Výsledkom riešenia danej problematiky je už spomínaná používateľská príručka, ktorú je možné využiť v praxi pri práci s programovacím jazykom R.

Kľúčové slová:
programovací jazyk R, vektor, matica, tabuľka, štatistika, aktuárstvo

ABSTRACT

SALINGER, Peter: *Language R and its use in actuarial analysis*. – University of Economics in Bratislava. Faculty of Informatics, Department of Mathematics and Actuary. – Supervisor of final thesis: Ing. Michal Páleš, PhD. – Bratislava, FHI EU, 2014, 83p.

The goal of final work is to make the language R more user friendly. The work is written in easy way. The user, who will use this work when working with the language R, does not need to have any experiences with programming languages. The final work is making the user familiar with the basics of the language R and their next usage in more complex levels of operations in programming language R. The first chapter is devoted to the description of programming language and its usage in present on foreign, but also in home field. In the next part, the goals of whole work are characterized. The third is describing the approach, how the set goals will be fulfilled. The final chapter is engaged with presentation of user guidebook according to goals set in second chapter. The conclusion of solving the set problem is above mentioned user guidebook, which can be used in practice when working with programming language R.

Key words:

R language, vector, matrix, table, statistics, actuary

Obsah

Úvod

Jazyk R je voľne dostupný programovací jazyk a aj prostredie. Vznikol spoločne s komerčným jazykom *S–plus* z programovacieho jazyka S. Jazyk R je univerzálny softwarový nástroj a prostredie pre spracovávanie dát a ich analýzu, výpočty a tvorbu grafických výstupov. V jazyku R môžeme riešiť matematické a štatistické úlohy ako sú napríklad operácie s vektormi a maticami, výpočet popisných charakteristík súboru dát, odhad lineárneho a nelineárneho modelu, testy štatistických hypotéz, analýzy časových radov a mnohé iné.

Táto práca je zameraná na sprístupnenie a podporu pre zvládnutie prostredia programovacieho jazyka R. Cieľom je vytvoriť príručku, v ktorej používateľovi opíšeme a názorne na výstupoch predvedieme vybrané činnosti ako sú inštalácia programového prostredia jazyka R, základná orientácia v prostredí jazyka R, použitie základných príkazov a vytváranie objektov rôznych typov. Postupne prejdeme jednotlivými operáciami, až sa dostaneme k aplikácií jazyka R.

V prvej kapitole sa venujeme opisu *open–source* software, medzi ktorý patrí aj jazyk R. Priblížime si súčasný stav využívania jazyka R v zahraničí, ale aj na Slovensku. Druhou kapitolou si spresníme všetky ciele záverečnej práce. Tretia kapitola opisuje spôsob napĺňania jednotlivých cieľov práce t.j. postup tvorby používateľskej príručky pre programovací jazyk R. V poslednej kapitole bude po krokoch popísaný postup od inštalácie prostredia jazyka R až po jeho využitie pri riešení vybraných štatistických a aktuárskych úloh.

1 Súčasný stav riešenej problematiky doma a v zahraničí

Open–source softvér je počítačový softvér, ktorého zdrojový kód je prístupný pod takou licenciou, ktorá umožňuje študovanie resp. vkladanie zmien a vylepšení do zdrojového kódu a umožňuje ďalšiu voľnú redistribúciu v modifikovanej alebo nezmenenej forme.

V dnešnej dobe nie je problém naraziť na software, ktorý využíva licenciu pre *open–source*. Veľká väčšina ľudí po celom svete využíva *open–source* software bez toho, aby si to vôbec uvedomovali. Bežne sa *open–source* vyskytuje vo forme napríklad Linuxových operačných systémov. Veľmi známy *open–source* software *Firefox* slúži ako webový prehliadač, ten je v súčasnosti silnou konkurenciou pre ostatné webové prehliadače.

Pre skúsených programátorov predstavuje veľkú výhodu fakt, že pri práci s *open–source*, rovnako ako pri práci s komerčným programom, môže nastať situácia, kedy program nevie danú problematiku riešiť. Pri *open–source* je však možné nazrieť do zdrojového kódu programu a prezrieť si algoritmus, s ktorým program pracuje. Taktiež je možné tento algoritmus dotvoriť pre potreby problematiky a následne odoslať túto úpravu tvorcom programu, ktorí ju môžu implementovať do programu, aby bola dostupná aj pre iných používateľov.

1.1 Open–source software jazyka R

Na rozdiel od ostatných aplikácii naprogramovaných pre účely štatistického spracovávania dát, ktoré si používateľ nemôže nijako prispôsobiť svojím potrebám, jazyk R nie je len aplikácia, ale aj programovací jazyk. To znamená, že preddefinované funkcie môže používateľ používať v rôznych kombináciách a tým tvoriť vlastné riešenia rôznych problematík. Taktiež je možné pracovať s celým algoritmom vstavanej funkcie a upraviť si ho do vlastnej funkcie.

Túto tému vystihuje citát autora knihy *S Poetry*[1], ktorá je dostupná na internete vo formáte pdf a slúži ako návod k používaniu jazyka S, ktorý je vlastne predchodcom jazyka R. Citát síce hovorí o jazyku S, ale svojím významom má dopad aj na jazyk R. „Podstatou je, že S je jazyk. To robí S oveľa užitočnejším ako keby bol iba balíkom pre štatistiku, grafiku alebo matematiku. Predstavte si, že angličtina by nebola jazykom, ale iba zbierkou slov, ktoré by mohli byť použité iba jednotlivo - balík. Potom to, čo je vyjadrené v tejto vete, je ďaleko komplexnejšie ako akýkoľvek význam, ktorý by mohol byť vyjadrený pomocou balíka angličtiny.“[2]

V súčasnosti má prostredie programovacieho jazyka R veľmi veľa používateľov po celom svete. Programovací jazyk R sa v súčasnosti bežne využíva na akademické účely. Kurzy výučby programovacieho jazyka R prebiehajú napríklad na Fakulte manažmentu Vysokej školy ekonomickej v Prahe, na Prírodovedeckej fakulte a na Matematicko–fyzikálnej fakulte Karlovej Univerzity v Prahe. Tieto kurzy zabezpečuje firma RZJ – STAT s.r.o.[3], táto firma je autorom niektorých doplnkových balíčkov pre jazyk R(*packages*) ako napríklad *mixAK*, *smoothSurv*, *bayesSurv*). Tieto sú zamerané na pokročilé štatistické operácie a sú dostupné na stiahnutie na oficiálnych stránkach projektu.

S výukou jazyka R formou kurzov sa môžeme stretnúť na stránkach *Heriot-Watt university*[4]. Formou bezplatného kurzu je na týchto stránkach predvádzaná práca s jazykom R. Kurz je zameraný pre úplných začiatočníkov. Formou videa sa používatelia oboznamujú s inštaláciou a základným ovládaním jazyka R. Prostredníctvom dokumentov, ktoré slúžia ako vstupné dáta do prostredia jazyka R, sa používatelia naučia pracovať so zložitejšími funkciami a využívať grafické nástroje.

Výhodou jazyka R je, že voľný prístup ku zdrojovému kódu podporuje používateľov k tvorbe vlastných riešení problémov. A keďže používateľov je veľmi veľa, nie je problém získať podporu od ostatných používateľov navštívením špecializovaných fór. Projekt R [5] má k dispozícii hneď dve hlavné fóra (*r–help* alebo *r–dev*). Jedno je zamerané na používateľov a druhé na vývojárov.

[1] http://www.burns-stat.com/documents/books/s-poetry/
[2] http://www.burns-stat.com
[3] http://www.rzj-stat.cz/
[4] http://www.macs.hw.ac.uk/actuarial/R/
[5] r–help http://r.789695.n4.nabble.com/, r–dev http://r-forge.r-project.org/forum/

1.2 Programovací jazyk R na Slovensku

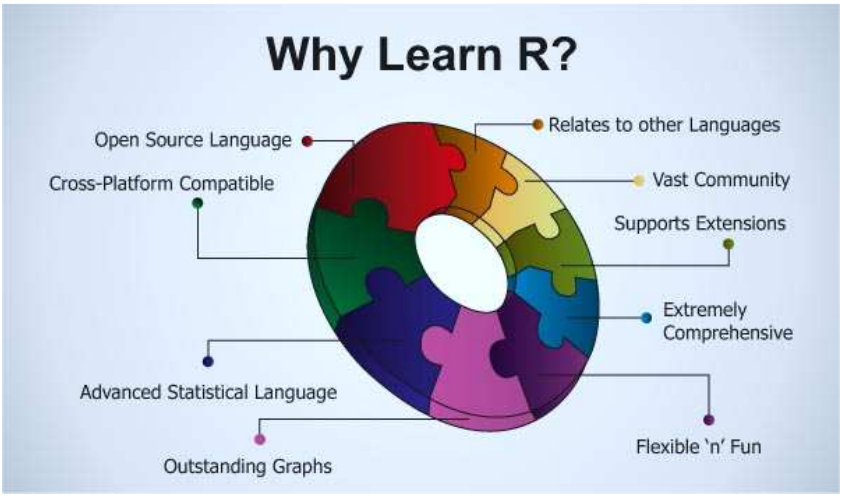
V súčasnosti prebieha výučba jazyka R pre potreby modelovania a prognózovania ekonomických a finančných časových radov. Výučba prebieha na základných príkladoch z predmetu *Ekonometria* a *Prognostika* na Katedre makro a mikroekonomiky na Žilinskej univerzite.

Univerzita Komenského v Bratislave, Fakulta matematiky, fyziky a informatiky využíva programovací jazyk R pri cvičeniach z predmetu *Finančné deriváty*.

Fakulta prírodných vied Univerzity Mateja Bela v Banskej Bystrici, Katedra matematiky rovnako pri výučbe využíva jazyk R.

Na Fakulte Zdravotníctva a Sociálnej Práce Trnavskej univerzity študenti Katedry Verejného Zdravotníctva využívajú programovací jazyk R pri tvorbe grafov k predmetu *Analýza Epidemiologických Dát II*.

Na Ekonomickej Univerzite v Bratislave, Fakulta hospodárskej informatiky sa jazyk R vyučuje v 1. ročníku druhého stupňa v študijnom programe *Aktuárstvo* v predmete *Programovacie techniky pre aktuárov*.



Obr. 1.2.1 Význam štúdia jazyka R[6]

[6] http://www.edureka.in

2 Cieľ práce

Hlavným cieľom bakalárskej práce je priblížiť používateľom programovací jazyk R a na príkladoch predviesť jeho používanie. Práca je zameraná na používateľov, ktorí nemajú programátorské znalosti a zručnosti. Z toho dôvodu sme k napĺňaniu cieľov pristupovali postupne a vysvetľovali sme každý detail spojený s danou problematikou.

Pre začiatok sme potrebovali objasniť, aké výhody pre používateľa predstavuje jazyk R. Vysvetlili sme si, čo znamená *open–source* a povedali sme si príklady využitia tohto typu softwaru ako aj využitia samotného programovacieho jazyka R.

Ďalšiu časť tvorí príručka pre používateľa, ktorá pomôže používateľom zvládnuť prácu v prostredí programovacieho jazyka R od úplných začiatkov. Príručku sme rozdelili do hlavných častí podľa jednotlivých parciálnych cieľov. Tými sú inštalácia programu jazyka R, používateľské prostredie, základné príkazy a aplikácia jazyka R pri riešení štatistických a akuárskych úloh.

1. **Inštalácia programu jazyka R** - v tejto časti sme predviedli, odkiaľ sa dá programové vybavenie jazyka R stiahnuť a podrobný postup pri jeho inštalácii v jednotlivých krokoch.
2. **Používateľské prostredie** - táto časť má používateľovi objasniť možnosti a základnú orientáciu v používateľskom prostredí jazyka R ako aj priebeh prvého spustenia aplikácie.
3. **Základné príkazy** - cieľom tejto časti je naučiť používateľa pracovať so základnými príkazmi programovacieho jazyka R, ktoré sú ďalej obsiahnuté v ostatných častiach používateľskej príručky a vymedziť dátové typy, ktoré môžu hodnoty a objekty v jazyku R nadobúdať.
4. **Podrobný postup pri riešení niektorých štatistických a akuárskych úloh** - poslednou časťou sledujeme aplikáciu jazyka R na riešenie štatistických a aktuárskych úloh, medzi ktoré sme zahrnuli: výpočet vybraných popisných charakteristík, odhad lineárneho regresného modelu, rozdelenia pravdepodobností, testy dobrej zhody.

3 Metodika práce a metódy skúmania

Pri spracovaní cieľov bakalárskej práce sme zvolili postup tvorbou používateľskej príručky, v ktorej sme podrobne jednoduchým a jednoznačným popisom pre používateľa napĺňali všetky stanovené ciele. V priebehu tvorby príručky sme vytvárali pre používateľa výstupy ku každej problematike formou ukážok príkazov, ktoré sme tvorili formou print screenov z prostredia jazyka R.

Naša príručka teda obsahuje podkapitoly:

a) **Inštalácia prostredia** programovacieho jazyka R.
   Časť obsahuje jednoduchú navigáciu po českej verzii stránok[7] projektu R a podrobný postup pri inštalácii jazyka R s *print* screen obrázkami.
b) **Spustenie aplikácie.**
   V časti opisujeme spôsoby, akými sa dá spustiť programové prostredie jazyka R a vysvetľujeme prvotný výpis, ktorý programové prostredie po svojom spustení vypíše do konzoly.
c) **Používateľské prostredi**e.
   Na základe obrázku opisujeme používateľské prostredie, ktoré jazyk R ponúka a základnú orientáciu v tomto prostredí.
d) **Workspace.**
   V časti vysvetľujeme používateľovi význam *workspace* (pracovný priestor), základné príkazy na zmenu adresára pre *workspace* a príkazy, ktorými môžeme prehľadávať a meniť objekty vo vnútri *workspace*.
e) **Podporované dátové tipy.**
   Ide o rozsiahlejšiu časť zameranú na možnosti dátových typov, ktoré môžu hodnoty a objekty v jazyku R nadobúdať. Každý dátový typ sme osobitne popísali a predviedli na obrázku. Okrem hlavných dátových tipov sme si ukázali aj niektoré špeciálne hodnoty, ktoré sa v objektoch môžu nachádzať a vysvetlili sme si ich význam a použitie.
f) **Podporované dátové štruktúry.**
   Dátové štruktúry sú objekty, ktoré môžu obsahovať viacero zložiek. V tejto časti si

[7] http://www.r-project.cz/

bližšie vysvetlíme pojem dátové štruktúry a dané štruktúry rozdelíme do kategórií podľa zložiek, ktoré môžu obsahovať.

g) **Práca s objektmi**.

Ide o najrozsiahlejšiu časť záverečnej práce. Časť je venovaná definíciám dátových štruktúr z predchádzajúcej podkapitoly a operáciám s dátovými štruktúrami. Konkrétne sú to vektory, matice, zoznamy, dátové tabuľky a databázy. Ďalej časť obsahuje prácu s nástrojmi pre tvorbu grafov aj s grafickými výstupmi.

h) **Využitie jazyka R pri niektorých štatistických analýzach.**

S využitím poznatkov z predchádzajúcich podkapitol sme v tejto časti predviedli aplikáciu jazyka R na štatistických operáciách. V časti sme predviedli operácie ako výpočet popisných charakteristík, zostavenie histogramu zo súboru údajov, odhad lineárneho regresného modelu a testy štatistických hypotéz na tomto modeli.

i) **Využitie jazyka R pri vybraných aktuárskych analýzach.**

V tejto podkapitole sme riešili konkrétny príklad z aktuárskej praxe - Analýza portfólia akcií v jazyku R.

4 Výsledky práce

Obsahom tejto kapitoly je používateľská príručka, ktorá je členená do jednotlivých podkapitol podľa stanovených cieľov. Podkapitoly sú zoradené chronologicky za sebou, podľa náročnosti problematiky, ktorej sú venované. Poznatky z jednotlivých kapitol nadväzujú na seba a postupne sa prechádza k zložitejším problematikám.

Najskôr sme sa venovali postupu ako nainštalovať programové vybavenie jazyka R. Potom sme sa zamerali na používateľské prostredie v programovacom jazyku R, objasnili sme si všetky postupy od spustenia aplikácie, základnej orientácie v programe až po pracovné prostredie *workspace*.
Ďalej sme si priblížili jednotlivé dátové typy hodnôt a objektov, ktoré sa v jazyku R môžu vyskytovať a prešli sme k tvorbe dátových štruktúr a k operáciám s nimi. Do tejto časti patria aj databázy a nástroje na tvorbu grafov.

V záverečnej časti príručky sme predviedli aplikáciu predchádzajúcich poznatkov na príkladoch štatistických a aktuárskych analýz.

4.1 Inštalácia prostredia jazyka R

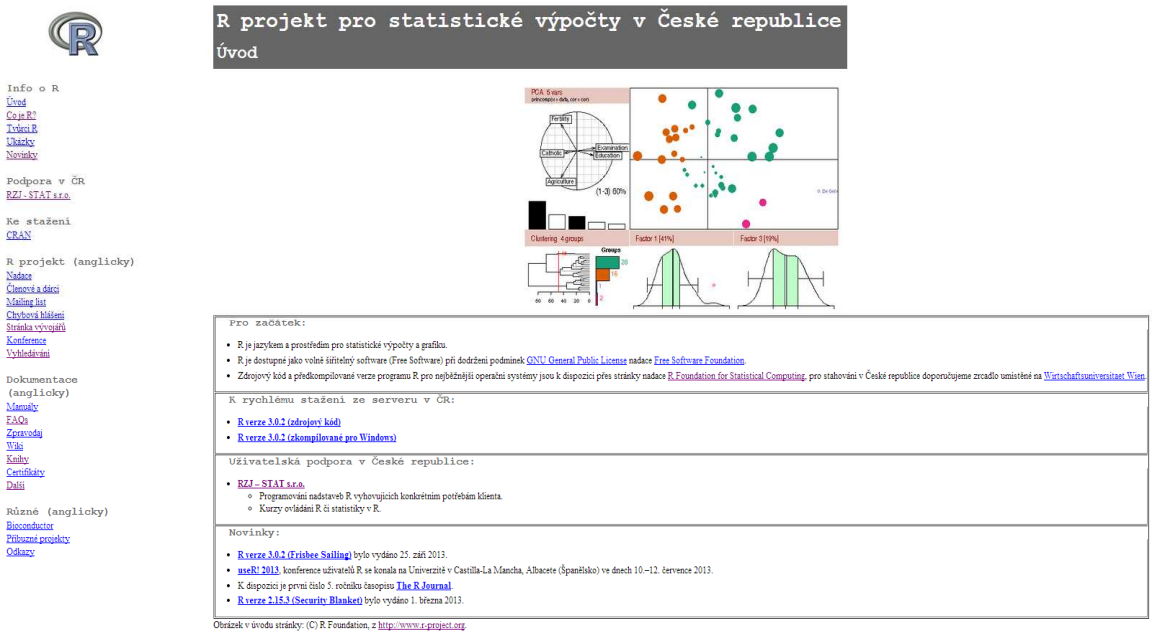
Programové vybavenie jazyka R si môžeme voľne stiahnuť na internete z prostredia[8] projektu R. Tieto sú k dispozícii aj s českým[9] prekladom. Okrem základného inštalačného súboru sú tu k dispozícii aj podporné balíčky (*packages*) na zvládnutie niektorých operácií, ktoré nie sú v základnom balíčku obsiahnuté. Verzie na stiahnutie sú k dispozícii pre operačné systémy Linux, OS X (Mac) a Windows.

1. Stiahnutie inštalačného súboru

K inštalačnému súboru pre operačný systém Windows sa môžeme dostať priamo z titulnej stránky projektu. K ostatným verziám sa dostaneme cez hypertextový odkaz *CRAN* v menu

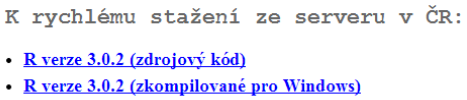
[8] http://www.r-project.org/
[9] http://www.r-project.cz/

stránky. Tam je ešte potrebné vybrať umiestnenie servera (*mirror*). Pre nás je najvýhodnejší z hľadiska rýchlosti pripojenia slovenský[10] server.



Obr. 4.1.1 České prostredie R-project

Pre stiahnutie programu je potrebné kliknúť na odkaz: *R verze 3.0.2 (zkompilované pro Windows)*. Odkaz nás presmeruje priamo k inštalačnému súboru a spustí sa sťahovanie.



Obr. 4.1.2 Linky na stiahnutie

2. Priebeh inštalácie programového vybavenia jazyka R

Spustením inštalačného balíka (dvojklikom alebo v menu spustiť) prejdeme k inštalácii.

- Prvým zobrazeným oknom je ***Select Setup Language***, kde si zvolíme jazyk pri inštalácii a stlačíme tlačidlo ***OK***. Medzi podporovanými jazykmi je aj čeština.
- Zobrazí sa ***Průvodce instalací*** obsahujúci informácie o verzii programu, ktorú sa chystáme nainštalovať, klikneme na tlačidlo ***další***.

[10] http://cran.fyxm.net/

- V ďalšom okne sa nachádza licenčná zmluva na používanie programu R, tá začína názvom licencie (*GNU GENERAL PUBLIC LICENSE*), čo je licencia pre *open–source* software. Klikneme na tlačidlo ***další***.
- V ďalšom okne je umiestnenie, kam sa má program jazyka R nainštalovať. Prednastavené je umiestnenie so súborovou cestou ***C:\Program Files\R\R-3.0.3***.
- Po stlačení tlačidla ***další*** prejdeme do okna, v ktorom vyberáme súčasti, ktoré chceme nainštalovať. Patria sem: *core files, 32–bit files, 64–bit files* a *message translations*. Buď necháme všetky zaškrtnuté alebo si vyberieme len jednu verziu podľa operačného systému. Napríklad ak máme 64–bitový operačný systém, položka *32–bit files* nemusí byť označená.
- Stlačením tlačidla ***další*** sa dostaneme do okna, ktoré sa nás pýta, či si chceme nejakým spôsobom zmeniť nastavenia pri spustení programu. Pre zjednodušenie inštalácie vyberieme možnosť *NO (accept default)* a stlačíme ***další***.
- Ďalšie okno nás informuje, že sa vytvorí zložka s odkazom na program v menu Štart operačného systému. V ľavom dolnom rohu je možné vytvorenie zložky zrušiť.
- Po stlačení tlačidla ***další*** sa zobrazí posledné okno pred priebehom inštalácie. To ponúka možnosti: vytvorenie zástupcu programu jazyka R na pracovnej ploche, možnosť zahrnúť dáta vo formáte .R do inštalácie a možnosť uložiť verziu programu do registrov (užitočné pri budúcich aktualizáciách).
- V nasledujúcej časti prebieha samotná inštalácia programu jazyka R. Po jej dokončení sa zobrazí okno s informáciou, že inštalácia prebehla úspešne. Stlačením tlačidla ***dokončit*** sme úspešne nainštalovali programové vybavenie jazyka R.

4.2 Spustenie aplikácie

Aplikáciu *R-Gui* (*Graphical User Interface*) môžeme spustiť dvomi spôsobmi v závislosti od toho, či pracujeme na počítači s 32–bitovým alebo 64–bitovým operačným systémom.

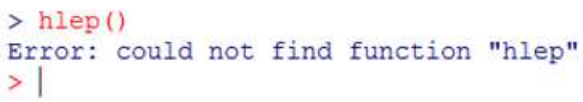
Po spustení aplikácie sa zobrazí konzolové okno, ktoré obsahuje informácie o verzii aplikácie aj programového jazyka a dátume vydania tejto verzie. Verzia platformy, na

ktorej je aplikácia spustená, teda w32-bit alebo w64-bit pre operačný systém Windows s 32 alebo 64 bitovým procesorom.

Používateľ je informovaný o tom, že programovací jazykR je *open–source* software a je dodávaný bez akejkoľvek záruky. Za splnenia určitých podmienok je možné ho ďalej šíriť.

Posledná úvodná informácia obsahuje príkazy `license()` alebo `licence()` pre detaily o distribúcii, `contributors()` zobrazí tvorcov a prispievateľov projektu R, `citation()` zobrazí informácie pre citáciu projektu a prídavných balíkov v publikáciách, `demo()` pre ukážky príkazov, `help()` pre nápoveď alebo `help.start()` pre online nápoveď a `q()` pre ukončenie aplikácie. Posledný riadok konzolového okna obsahuje *prompt* , ktorý oznamuje stav. Znak `>` predstavuje pripravenosť konzoly.

Text nachádzajúci sa v konzolovom okne môžeme farebne rozlíšiť. Vstupné riadky, tzn. tie, do ktorých sme zadali príkazy, sú zobrazené červenou farbou a výstupné riadky, ktoré sa zobrazia v konzole po zadaní príkazu, sú modré. Po zadaní príkazu a stlačení klávesy *Enter* sa zadaný príkaz skompiluje, následne sa hneď spustí a na konzole sa zobrazí výstup tohto príkazu. V prípade, že zadaný príkaz bol chybný, zobrazí sa hlásenie o chybe vo formáte *Error*: stručný popis problému.



Obr. 4.2.1 Hlásenie o chybe

Chybne zadaný príkaz nie je možné opraviť priamo. Musíme ho zadať znovu a opraviť chybu. V histórii príkazov, teda v príkazoch, ktoré sme už do konzoly vložili, môžeme listovať klávesmi ↑ a ↓. Túto históriu príkazov si môžeme uložiť pomocou ***File*** → ***Save history***. Ak chceme uložiť celý obsah konzoly, tzn. aj s výstupmi, použijeme ***File*** → ***Save to File***.

Príkazy môžeme zadávať jednotlivo vždy na nový riadok alebo ich môžeme zadávať do jedného riadku a oddeľovať ich bodkočiarkou. V prípade, že náš príkaz je príliš dlhý, môžeme v písaní príkazu pokračovať na ďalšom riadku, avšak takýto príkaz musí byť ohraničený zátvorkami, aby program vedel, že príkaz ešte neskončil. V prípade použitia tohto spôsobu sa zmení znak *prompt* zo znaku `>` na `+`, čo značí nedokončený príkaz.

Pomocou znaku # môžeme do konzoly vkladať komentáre. Tento znak spôsobí, že program bude ignorovať zvyšok riadku.

```
> #použitie jedného riadku na jeden príkaz
> a<-1
> a
[1] 1
> #použitie jedného riadku na viacero príkazov
> a<-1; a
[1] 1
> #pokračovanie príkazu na ďalšom riadku
> (a<-
+ 2)
[1] 2
>
```

Obr. 4.2.2 Komentáre v konzole

Príkaz sa skladá z názvu príkazu a parametrov, ktoré sa nachádzajú v okrúhlych zátvorkách za názvom príkazu. Niektoré príkazy nevyžadujú zadávanie parametrov, sú vykonateľné aj bez parametrov. Parametre môžeme zadávať dvoma spôsobmi:

1. zadáme ich len pomocou hodnôt parametrov, ale v tom prípade musia nasledovať za sebou podľa toho ako sú definované,
2. môžeme ich zadávať v ľubovoľnom poradí s tým, že jednoznačne určíme, ktorá hodnota patrí akému parametru. Vložený parameter má potom formát `názov_parametra=hodnota_parametra`. Názov parametra môžeme skracovať počtom písmen, nesmie však dôjsť k zameniteľnosti s iným parametrom.

```
> #použitie jedného riadku na jeden príkaz
> a<-1
> a
[1] 1
> #použitie jedného riadku na viacero príkazov
> a<-1; a
[1] 1
> #pokračovanie príkazu na ďalšom riadku
> (a<-
+ 2)
[1] 2
>
```

Obr.4.2.3 Skracovanie názvu parametrov

Výpis hodnôt začína vždy poradovým číslom hodnoty (v hranatých zátvorkách), samotná hodnota nasleduje až za hranatými zátvorkami. Na obr. 4.2.4 je výpis objektu x, ktorým je vektor reálnych čísiel so zaokrúhlením na päť desatinných miest. Nový riadok vždy začína poradovým číslom hodnoty.

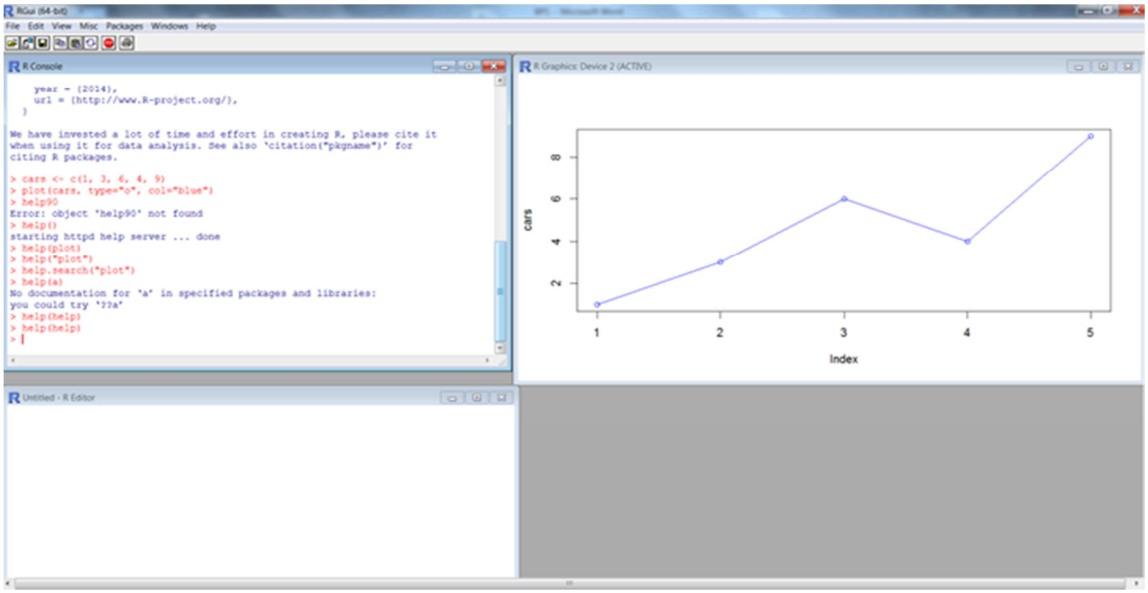
```
> x
 [1] 93.133364 76.219651 75.481061 12.523471  6.376334 46.292967 47.655515
 [8] 93.817766 22.246986  1.486147  5.548814 76.528940 64.502648 24.815738
[15]  7.446224 64.073576 51.362854 43.319775 70.897612 35.187427
> 
```

Obr. 4.2.4 Výpis vektora reálnych čísel

4.3 Používateľské prostredie

Prostredie môžeme rozdeliť do štyroch častí:

1. **Konzolové okno** (*R console*) - slúži na zadávanie príkazov, tvorbu výpočtov, vkladanie objektov a spúšťanie funkcií.
2. **Nápovedné okno** (*help*) - zobrazuje informácie o príkazoch a funkciách a spúšťa sa príkazom *help* (názov príkazu/funkcie). Ak sme v inštalácii programu zaškrtli políčko *HTML help*, nápoveď sa otvorí cez webový prehliadač.
3. **Grafické okno** - ak v konzole použijeme niektorý z grafických nástrojov, zobrazuje sa automaticky.
4. **Okno programového editora** - môžeme ho spustiť cez ponuku ***File*** → ***New Script*** a slúži na tvorbu programov.



Obr. 4.3.1 Používateľské prostredie jazyka R

4.4 Workspace

Workspace je pracovný priestor, v ktorom používateľ pracuje a do ktorého sa ukladajú všetky objekty spolu s históriou príkazov, ktoré používateľ počas svojej činnosti vytvoril. *Workspace* si môžeme uložiť pomocou navigačných okien stlačením ***File → save Workspace***. Jazyk R má niekoľko príkazov na pohyb v adresároch, ktoré sú podobné príkazom bežne používaným v linuxových systémoch. Je to napríklad príkaz `getwd()`, ktorý zobrazí úplnú súborovú cestu k aktuálnemu pracovnému priečinku a pomocou príkazu `setwd()` môžeme súborovú cestu zmeniť. Pomocou `dir()` môžeme zobraziť súbory v adresári.

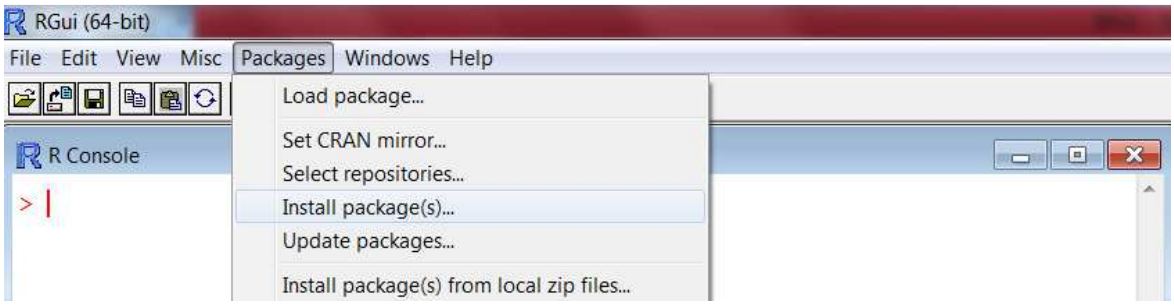
Pre prácu s pracovným priestorom poznáme napríklad príkazy `objects()` a `ls()`. Vypíšu všetky objekty, ktoré sa aktuálne v pracovnom priestore nachádzajú. Príkazom `rm()` môžeme objekty mazať.

4.5 Doplnkové balíčky

Doplnkové balíčky do programovacieho jazyka R sú vlastne časti zdrojového kódu jazyka R, ktoré sú zamerané na riešenie konkrétnych problematík. Tieto balíčky vytvárajú

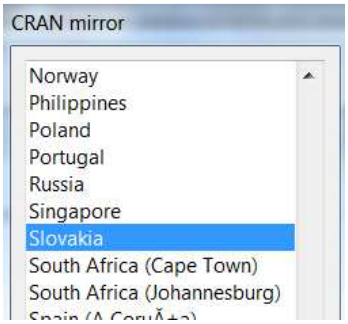
buď samotní vývojári jazyka R alebo aj pokročilejší používatelia jazyka R. Príkladom takýchto používateľov je už spomínaná česká firma RZJ – STAT s.r.o. , ktorá je autorom viacerých balíčkov pre jazyk R. Takto vytvorený balíček sa pošle vývojárom jazyka R spolu s dokumentáciou, tam ho otestujú a keď balíček spĺňa požiadavky a správne rieši problematiku, na ktorú je určený, je sprístupnený aj ostatným používateľom.

Doplnkové balíčky sa sťahujú priamo z prostredia programovacieho jazyka R v menu ***Packages → Install Packages(s)***.



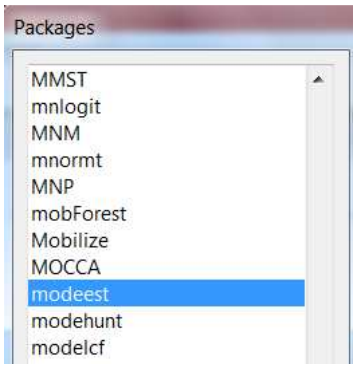
Obr. 4.5.1 Inštalácia doplnkového balíčka

Po kliknutí na položku ***Install package(s)*** sa zobrazí ďalšie programové okno, kde si máme vybrať server(*mirror*), z ktorého chceme doplnkové balíčky sťahovať. Najvýhodnejšie je vybrať možnosť ***Slovakia***, čím sa pri sťahovaní balíčkov budeme pripájať na server v FYXM.net v Bratislave.



Obr. 4.5.2 Výber servera pre sťahovanie balíčkov

Keď si vyberieme server, zobrazí sa okno s ponukou všetkých dostupných balíčkov pre verziu prostredia programovacieho jazyka R, ktorá je nainštalovaná v operačnom systéme.



Obr. 4.5.3 Výber doplnkového balíčka

Po vybraní nami zvoleného balíčka sa tento balík doinštaluje do prostredia jazyka R. Aby sme ho mohli začať používať, je potrebné tento balíček pripojiť k aktívnemu pracovnému priestoru (*workspace*). Spravíme tak príkazom `library()`, do argumentu príkazu dáme názov balíčka.

```
> library(modeest)

This is package 'modeest' written by P. PONCET.
For a complete list of functions, use 'library(help = "modeest")' or 'help.star$
```

Obr. 4.5.4 Pripojenie balíčka

Po pripojení balíčka sa zobrazí uvítacia informácia k balíčku (ak ju autor nastavil). Tá väčšinou obsahuje krátky popis balíčka, meno autora a príkaz na zobrazenie obsiahnutých funkcií. Od tohto momentu môžeme zažať využívať všetky nové funkcie, ktoré sú v balíčku obsiahnuté.

Napr. balíček ***modeest*** budeme využívať pri výpočte modusu súboru dát v podkapitole 4.8. Balíček obsahuje tieto funkcie:

`mlv`	odhad modusu
`plot.mlv`	zobrazenie odhadovaného objektu na graf
`print.mlv`	výpis odhadovaného objektu
`as.numeric.mlv`	zmení typ objektu na číselnú hodnotu (ak ňou nie je)
`mfv`	nájdenie najčastejšie sa vyskytujúcej hodnoty

Okrem týchto piatich funkcií obsahuje ešte ďalších 31 funkcií na výpočet modusu v súbore s rôznymi rozdeleniami pravdepodobností.

4.6 Podporované dátové typy

Pre ukladanie a prácu s hodnotami v programovacom jazyku R využívame dátové objekty. Dátový objekt je jednoduchá premenná, do ktorej priraďovacím príkazom vložíme hodnotu. S objektom potom ďalej narábame akoby sa jednalo o konkrétnu hodnotu. Objekt však nemusí obsahovať len jednu hodnotu, v prípade viacerých hodnôt v objekte hovoríme o dátovej štruktúre.

Do objektov môžeme uložiť rôzne typy dát. Jazyk R pracuje s dátovými typmi:

- číslo,
- komplexné číslo,
- logická hodnota,
- reťazec.

4.6.1 Objekt typu číslo

Číslo vkladáme jednoduchým priraďovacím príkazom <-, za ktorým píšeme číselnú hodnotu, ktorú chceme do daného objektu vložiť. V prípade, že sa jedná o číslo s desatinnými miestami, na ich oddelenie použijeme desatinnú bodku.

```
> x<- 100.23
> x
[1] 100.23
> 
```

Obr. 4.6.1 Objekt typu číslo

4.6.2 Objekt typu komplexné číslo

Komplexné číslo obsahuje komplexný výraz, číslo + imaginárna jednotka. Pri zadávaní treba dodržať syntax jazyka R, teda komplexná jednotka musí byť vždy zadaná ako násobok **i**. V opačnom prípade bude program očakávať, že zadaný výraz je rovnica a **i** reprezentuje objekt.

```
> x<-10+1i
> x
[1] 10+1i
> y<-10+i
Error: object 'i' not found
> i<-5
> y<-10+i
> y
[1] 15
```

Obr. 4.6.2 Objekt typu komplexné číslo

4.6.3 Objekt typu logická hodnota

Logická hodnota môže nadobudnúť hodnoty T/TRUE alebo F/FALSE. Logickú hodnotu môžeme získať ako výsledok vyhodnotenia výrazu alebo si ju do objektu môžeme zadať manuálne. Slová TRUE a FALSE sú kľúčové slová a treba ich zadávať veľkými písmenami.

```
> a<-7+9<4
> a
[1] FALSE
> b<-TRUE
> b
[1] TRUE
> c<-true
Error: object 'true' not found
```

Obr. 4.6.3 Objekt typu logická hodnota

4.6.4 Objekt typu reťazec

Reťazec slúži na zadávanie slov alebo textu. Obsahuje skupinu znakov, ktoré musíme zadávať buď pomocou jednoduchých apostrof alebo pomocou úvodzoviek. Ak by sme tak neurobili, program by predpokladal, že znaky predstavujú objekt.

```
> a<-'ahoj'
> a
[1] "ahoj"
> b<-"ahoj ahoj"
> b
[1] "ahoj ahoj"
> c<-ahoj
Error: object 'ahoj' not found
```

Obr. 4.6.4 Objekt typu reťazec

4.6.5 Špeciálne hodnoty

Jazyk R rozpoznáva špeciálne hodnoty a to sú:

- NA (*not avaible*), predstavuje nezadanú alebo chýbajúcu hodnotu.
- NULL predstavuje prázdnu hodnotu objektu.
- Inf predstavuje hodnotu nekonečna.
- NaN (*Not* a *Number*) je výsledkom výpočtov, ktoré nie sú definované ako napríklad delenie nuly nulou. Ak príde k deleniu reálneho čísla nulou, jazyk R postupuje ako by sa jednalo o limitu.

```
> NA
[1] NA
> NULL
NULL
> NaN
[1] NaN
> Inf
[1] Inf
> 0/0
[1] NaN
> 1/0
[1] Inf
>
```

Obr. 4.6.5 Špeciálne hodnoty v R

4.7 Podporované dátové štruktúry

Dátovou štruktúrou rozumieme objekt, do ktorého môžeme uložiť viacero hodnôt. Môžeme hovoriť o hodnotách rovnakého dátového typu. To sa vyskytuje pri vektoroch, poliach a maticiach. Hodnoty rôzneho dátového typu môžeme vkladať do dátových štruktúr ako sú tabuľky alebo zoznamy.

Tab. 4.7.1 Prehľad dátových štruktúr

Dátová štruktúra	Možnosť uložiť hodnoty rôznych dátových tipov
vektor	Nie
matica	Nie
zoznam	Áno
dátová tabuľka	Áno

Do objektu môžeme vložiť aj databázu, ktorú získame pripojením k inému softwaru, takýto objekt sa po jeho načítaní a uložení stáva dátovou tabuľkou.

4.8 Práca s objektmi

Názvy objektov vyžadujú dodržiavať určité pravidlá, aby sme ich mohli odlíšiť od seba, ale aj od zvyšku príkazu. To znamená, že ich názov musí byť jednoslovný a musí pozostávať zo znakov, ktoré sú povolené pre pomenovávanie objektov, čiže zo znakov malej a veľkej abecedy (R je case sensitive – rozpoznáva veľké a malé písmená). Môžeme používať aj interpunkčné znamienka, číslice od 0 po 9, bodky a podčiarkovník. V prípade, že chceme vytvoriť objekt, ktorého názov má pozostávať z dvoch slov, aby sa dal rozlíšiť, použijeme práve bodku alebo podčiarkovník.

```
> počet_kusov <- 7
> počet_kusov
[1] 7
> Počet_kusov
Error: object 'Počet_kusov' not found
>
```

Obr. 4.8.1 Dlhé názvy objektov

Ako vidno na obr. 4.8.1, k zadávaniu hodnôt do objektov slúži príkaz `<-`. Na zobrazenie hodnoty objektu stačí do konzoly napísať jeho názov. Zobrazený výstup obsahuje v hranatých zátvorkách poradie hodnoty v objekte a za zátvorkami nasleduje hodnota objektu.

Do objektov však môžeme vkladať aj dátové štruktúry. Takto vznikajú zložitejšie operácie s objektmi. Tie si ukážeme postupne v podkapitolách 4.9.1 až 4.9.5.

4.8.1 Vektory a operácie s nimi

Vektor je najjednoduchšia dátová štruktúra v jazyku R. Do vektora môžeme vkladať buď číselné, komplexné, logické hodnoty alebo reťazce. Jeden vektor však môže obsahovať len hodnoty jedného dátového typu. V tejto podkapitole si predvedieme nasledovné operácie s vektormi: definovanie, výber podmnožín, vytváranie postupností a opakovanie zložiek vektora.

1. Definovanie vektora

Pre vytvorenie prázdneho vektoru používame príkaz `vector`, ktorý ma dva parametre: `mode` a `lenght`. Parametrom `mode` udávame dátový typ, ktorý budeme do vektora vkladať a parametrom `lenght` udávame dĺžku daného vektora.

```
> vector(mode="character", length=4)
[1] "" "" "" ""
>
```

Obr. 4.8.2 Definovanie vektora

Oveľa jednoduchší spôsob je tvoriť vektor pomocou príkazu `<-` tak ako sme to používali pri práci s jednoduchými objektmi resp. si ho môžeme vytvoriť aj analogickým príkazom `assign`, ktorého parametre sa rovno vpisujú v presne stanovenom poradí, názov vektora a následne samotný vektor.

```
> numerický_vektor<-c(100, 2, 9, 4.178)
> numerický_vektor
[1] 100.000   2.000   9.000   4.178
> assign("vektor_typu_reťazec", c("text1", "text2"))
> vektor_typu_reťazec
[1] "text1" "text2"
>
```

Obr. 4.8.3 Definovanie vektora priradením a príkazom assign

Ako sme mohli vidieť na obr. 4.8.3, ak pri tvorbe numerického vektora zadávame reálne čísla, formát ich výpisu sa prispôsobí číslu s najväčším počtom desatinných miest.

V prípade, že by sme sa pokúsili vytvoriť vektor s hodnotami rôznych dátových typov, jazyk R vyberie iba jeden typ a prispôsobí mu ostatné hodnoty. Ako sme už spomínali, do vektora môžeme vkladať čísla, komplexné čísla, logické hodnoty a reťazce. Ak sa medzi zadanými hodnotami nachádza reťazec, ostatné hodnoty sa považujú tiež za hodnoty typu reťazec. Ďalším preferovaným typom je komplexné číslo a po komplexnom čísle je číselná hodnota. Na poslednej úrovni sa nachádzajú logické hodnoty.

```
> c(100.1, TRUE, "text", 3+1i)
[1] "100.1" "TRUE"  "text"  "3+1i"
> c(100.1, TRUE, 3+1i)
[1] 100.1+0i   1.0+0i   3.0+1i
> c(100.1, TRUE)
[1] 100.1   1.0
```

Obr. 4.8.4 Preferencie dátových typov v jazyku R

2. Výber podmnožín vektora

Pri práci s vektormi nie vždy potrebujeme narábať s celým vektorom. Niekedy potrebujeme k vektoru pristupovať postupne po jednotlivých hodnotách. To môžeme robiť pomocou hranatých zátvoriek, do ktorých vpisujeme poradie hodnoty vo vektore. Tieto zátvorky umiestnime hneď za názov vektora.

```
> x
[1] 1 7 9 6 4
> x[4]
[1] 6
>
```

Obr. 4.8.5 Výber podmnožiny vektora

Ak pred poradové číslo hodnoty dáme znamienko mínus, táto hodnota sa nezobrazí.

```
> x
[1] 1 7 9 6 4
> x[-3]
[1] 1 7 6 4
> 
```

Obr. 4.8.6 Vynechanie časti vektora pri výpise

V prípade, že chceme skrátiť vektor a používať len časť hodnôt, použijeme príkaz `length`. Má len jeden parameter - názov vektora, s ktorým pracujeme. Takto zadaný príkaz nám len vypíše číslo, koľko hodnôt sa v danom vektore nachádza. Explicitným nastavením však toto číslo môžeme zmeniť.

```
> y<-c(7,8,9,4,5,6,1,2,3,4,5,6,8)
> length(y)
[1] 13
> length(y)=5
> y
[1] 7 8 9 4 5
> 
```

Obr. 4.8.7 Dĺžka a zmena dĺžky vektora

Ďalšie možnosti predstavujú príkazy `head` a `tail`. `Head` zobrazí prvých šesť hodnôt vektora a `tail` posledných šesť hodnôt. Oba príkazy majú rovnaké dva parametre. Prvým je názov vektora a druhým počet hodnôt, ktoré chceme zobraziť a ak ho nenastavíme, jazyk R sa riadi implicitným nastavením, z toho vyplýva šesť hodnôt.

```
> head(y)
[1] 1 2 3 4 5 6
> head(y,3)
[1] 1 2 3
> tail(y,7)
[1] 14 15 16 17 18 19 20
> 
```

Obr. 4.8.8 Príkazy head a tail

Vypisované hodnoty môžeme ovplyvniť aj pomocou logických operátorov, musíme ich však zapísať vektorom rovnakej dĺžky ako vektor, z ktorého vychádzame. Ak by sme

nedodržali tento postup, jazyk R berie prevyšujúce hodnoty ako TRUE a teda ich vypíše. Podmienky výpisu môžeme zadávať aj ako logický výraz.

```
> x
[1] 1 7 9 6 4
> x[c(TRUE,FALSE,FALSE,TRUE,TRUE)]
[1] 1 6 4
> x[x>5]
[1] 7 9 6
>
```

Obr. 4.8.9 Výber podmnožiny pomocou logických hodnôt a logického výrazu

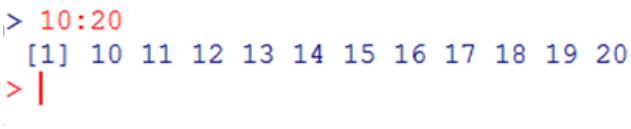
Ďalšou veľmi zaujímavou možnosťou je priradenie názvov jednotlivým hodnotám vektora využitím príkazu names. Má jeden parameter - názov vektora, ktorého hodnoty ideme pomenovávať. Výber hodnôt potom môžeme uskutočniť pomocou poradí hodnôt alebo názvov hodnôt.

```
> x
[1] 1 7 9 6 4
> names(x)<-c("jedna", "sedem", "deväť", "šesť", "štyri")
> x
jedna sedem deväť  šesť štyri
    1     7     9     6     4
> x[c(1,2)]
jedna sedem
    1     7
> x[c("jedna","deväť")]
jedna deväť
    1     9
>
```

Obr. 4.8.10 Názvy hodnôt vo vektore

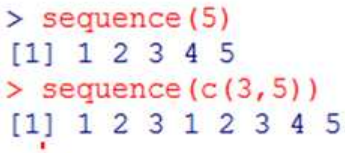
3. Vytváranie postupností

V jazyku R sa nachádza veľa rôznych príkazov na vygenerovanie postupnosti čísiel. Najjednoduchší spôsob ako takúto postupnosť vygenerovať je pomocou operátora :. Číslo pred operátorom vyjadruje hodnotu, od ktorej chceme postupnosť generovať a číslo za operátorom predstavuje hodnotu, po ktorú chceme postupnosť generovať. Pri takejto postupnosti nám budú čísla vždy narastať o jednotku.



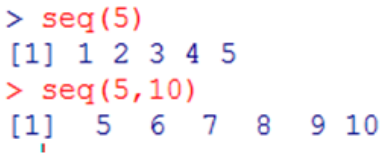
Obr. 4.8.11 Postupnosť pomocou operátora `:`

Podobnú postupnosť môžeme získať príkazom `sequence`, ktorý reprezentuje funkciu vytvárajúcu postupnosť od 1 do číselnej hodnoty v parametri funkcie. Dokonca môžeme túto funkciu spustiť aj hodnotami vektora, čo vyvolá vznik viacerých postupností za sebou.



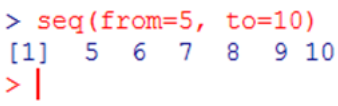
Obr. 4.8.12 Postupnosť príkazom sequence

Veľmi podobný je aj príkaz `seq()`, ktorý v prípade zadania len jedného číselného parametra bude generovať postupnosť od 1 do hodnoty v parametri. V prípade zadania dvoch číselných parametrov sa bude príkaz pri tvorbe postupnosti správať rovnako ako pri tvorbe postupnosti využitím operátora `:`.



Obr. 4.8.13 Nahradenie operátora `:` príkazom `seq`

Ako sme už naznačili, príkaz `seq` má viacero parametrov, vďaka ktorým môžeme získať aj zložitejšie postupnosti. Tento príkaz má celkovo 5 možných parametrov a nie všetky je dovolené kombinovať. Dva, ktoré sme už použili, môžeme zapísať aj tak, že pomenujeme parameter a priradíme mu hodnotu. V tomto prípade to boli parametre `from` a `to`.



Obr. 4.8.14 Postupnosť príkazom seq

Ďalšie dva parametre sú `by` a `length`. Parameter `length` určuje dĺžku výslednej postupnosti, z čoho vyplýva, že v prípade akéhokoľvek nastavenia parametrov bude mať výsledná postupnosť toľko hodnôt, koľko určuje parameter `length`. Pomocou parametra `by` môžeme ovplyvňovať jednotlivé kroky postupnosti, aby sme nemali nárast medzi číslami o jedna ale o toľko, koľko sami určíme. Nemusí pritom ísť výlučne o stúpajúcu postupnosť, parameter `by` môže obsahovať aj záporné hodnoty.

```
> seq(from=7, by=7, length=7)
[1]  7 14 21 28 35 42 49
> seq(from=49, by=-7, length=7)
[1] 49 42 35 28 21 14  7
```

Obr. 4.8.15 Klesajúca postupnosť

Posledný parameter je `along.with`. Nemôžeme ho použiť naraz s inými parametrami. Nezadáva sa doň číselná hodnota, ale vektor. Parameter spočíta, koľko hodnôt sa vo vektore nachádza a prispôsobí tomu výslednú postupnosť. V podstate má rovnaký výsledok ako parameter `length`. Ak by sme doňho vložili len číslo, teda nie vektor, bolo by to brané ako vektor o dĺžke 1 a teda aj výsledná postupnosť by obsahovala len jednu hodnotu.

```
> seq(1,10)
 [1]  1  2  3  4  5  6  7  8  9 10
> x<-seq(1,10)
> y<-10+seq(along.with=x)
> y
 [1] 11 12 13 14 15 16 17 18 19 20
> y<-10+seq(along.with=seq(1,10))
> y
 [1] 11 12 13 14 15 16 17 18 19 20
> y<-10+seq(along.with=100)
> y
[1] 11
>
```

Obr. 4.8.16 Použitie parametra along.with

4. Opakovanie zložiek vektora

V jazyku R môžeme vytvoriť vektor, ktorý sa skladá z hodnôt alebo vektorov a tie sú nejakým spôsobom opakované. Na to slúži príkaz `rep()`. Bez vložených parametrov

vráti hodnotu NULL, keďže nemalo byť čo opakované. Do prvého a najdôležitejšieho parametra sa vkladá hodnota alebo vektor, s ktorým ideme ďalej pracovať. Ďalšími parametrami sú `times`, `each`, `length.out`. Hodnota v parametri `times` určuje počet koľkokrát bude hodnota alebo vektor z prvého parametra opakovaný. Hodnota v parametri `each` určuje koľkokrát sa bude opakovať každá zložka vektoru z prvého parametra. V prípade, že v prvom parametri sa nachádza len jedno číslo, parametre `times` a `each` ovplyvnia výsledný vektor rovnakým spôsobom. Hodnota v parametri `length.out` stanovuje počet hodnôt výstupného vektora. Ak ostatné parametre vyvolali vektor o dĺžke väčšej ako určuje parameter `length.out`, tento vektor sa jednoducho skráti na požadovaný počet hodnôt vynechaním prevyšujúcich hodnôt od konca vektora.

```
> rep()
NULL
> rep(seq(1,3),times=2)
[1] 1 2 3 1 2 3
> rep(7,times=3)
[1] 7 7 7
> rep(seq(1,3),times=2, each=2)
 [1] 1 1 2 2 3 3 1 1 2 2 3 3
> rep(7,times=2, each=2)
[1] 7 7 7 7
> rep(seq(1,3),times=2, each=2, length.out=5)
[1] 1 1 2 2 3
>
```

Obr. 4.8.17 Opakovanie hodnôt vektora príkazom `rep`

4.8.2 Matice a operácie s nimi

Matica predstavuje dvojdimenzionálne pole rozdelené do riadkov a stĺpov. Do matice môžeme vkladať hodnoty dátových typov číslo, komplexné číslo, logickú hodnotu alebo znakový reťazec. Jedna matica však môže uchovávať hodnoty len jedného dátového typu. Rovnako ako pri vektoroch si jednotlivé operácie rozdelíme a budeme sa každej venovať samostatne.

1. Definovanie matice

Maticu vytvoríme jednoduchým príkazom `matrix()`. Má tri základné parametre, kde prvý je názov vektora alebo len hodnota (v tomto prípade sa nastavia všetky hodnoty

matice na hodnotu v parametri) a ďalšie dva slúžia na nastavenie počtu riadkov a stĺpcov matice.

Takto zapísaný príkaz vyvolá výpis vektora, ktorý bude upravený do tvaru matice.

```
> vektor<- 1:80
> matrix(vektor,8,10)
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    1    9   17   25   33   41   49   57   65    73
[2,]    2   10   18   26   34   42   50   58   66    74
[3,]    3   11   19   27   35   43   51   59   67    75
[4,]    4   12   20   28   36   44   52   60   68    76
[5,]    5   13   21   29   37   45   53   61   69    77
[6,]    6   14   22   30   38   46   54   62   70    78
[7,]    7   15   23   31   39   47   55   63   71    79
[8,]    8   16   24   32   40   48   56   64   72    80
>
```

Obr. 4.8.18 Príklad definovania matice

Súčin riadkov a stĺpcov matice sa musí rovnať počtu hodnôt vektora. V opačnom prípade dôjde k hláseniu chyby. Aj napriek chybe vznikne výpis matice s tým, že prevyšujúce hodnoty sú ignorované.

```
> vektor<- 1:80
> matrix(vektor,7,10)
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    1    8   15   22   29   36   43   50   57    64
[2,]    2    9   16   23   30   37   44   51   58    65
[3,]    3   10   17   24   31   38   45   52   59    66
[4,]    4   11   18   25   32   39   46   53   60    67
[5,]    5   12   19   26   33   40   47   54   61    68
[6,]    6   13   20   27   34   41   48   55   62    69
[7,]    7   14   21   28   35   42   49   56   63    70
Warning message:
In matrix(vektor, 7, 10) :
  data length [80] is not a sub-multiple or multiple of the number of rows [7]
>
```

Obr. 4.8.19 Hlásanie chyby pri nedodržaní podmienky o počte hodnôt

Ak by sme si chceli ponechať maticu na ďalšie použitie, jednoducho vytvoríme objekt, do ktorého pomocou príkazu `<-` priradíme príkaz `matrix()`.

V predchádzajúcich prípadoch sa nám zložky vektora ukladali najprv po riadkoch (nasledujúca hodnota na ďalší riadok) a potom po stĺpcoch. Ak by sme to chceli spraviť

opačne, pridáme do príkazu ďalší argument `byrow=TRUE`, čím definujeme jazyku R, že nemá hodnoty ukladať po riadkoch, ale ich má uložiť po stĺpcoch (nasledujúca hodnota do ďalšieho stĺpca).

```
> vektor<- 1:80
> matica<-matrix(vektor,8,10)
> matica
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    1    9   17   25   33   41   49   57   65    73
[2,]    2   10   18   26   34   42   50   58   66    74
[3,]    3   11   19   27   35   43   51   59   67    75
[4,]    4   12   20   28   36   44   52   60   68    76
[5,]    5   13   21   29   37   45   53   61   69    77
[6,]    6   14   22   30   38   46   54   62   70    78
[7,]    7   15   23   31   39   47   55   63   71    79
[8,]    8   16   24   32   40   48   56   64   72    80
>
```

Obr. 4.8.20 Uloženie matice do objektu

```
> matica<-matrix(vektor,8,10,byrow=TRUE)
> matica
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    1    2    3    4    5    6    7    8    9    10
[2,]   11   12   13   14   15   16   17   18   19    20
[3,]   21   22   23   24   25   26   27   28   29    30
[4,]   31   32   33   34   35   36   37   38   39    40
[5,]   41   42   43   44   45   46   47   48   49    50
[6,]   51   52   53   54   55   56   57   58   59    60
[7,]   61   62   63   64   65   66   67   68   69    70
[8,]   71   72   73   74   75   76   77   78   79    80
>
```

Obr. 4.8.21 Príklad definovania matice

Ako sme mali možnosť vidieť na obr. 4.8.21 R pri výpise matice pomenuje riadky a stĺpce ich poradovým číslom v hranatých zátvorkách s dvoma miestami pre poradie, pričom pri riadkoch je číslo na prvom mieste a pri stĺpcoch na druhom. Aj toto však môžeme zmeniť, a to použitím ďalšieho argumentu `dimnames=list`. *List* je samostatná dátová štruktúra (zoznam). Ide vlastne o dvojrozmerné pole, ktoré obsahuje dva na sebe nezávislé vektory. Pre naše momentálne potreby sú to vektory dátového typu znakový reťazec. Predvedieme si to na matici typu 3x3.

```
> vector<-1:9
> matica<-matrix(vector,3,3,dimnames=list(c("1.riadok","2.riadok","3.riadok"
+ ),c("1.stĺpec","2.stĺpec","3.stĺpec")))
> matica
         1.stĺpec 2.stĺpec 3.stĺpec
1.riadok        1        4        7
2.riadok        2        5        8
3.riadok        3        6        9
> 
```

Obr. 4.8.22 Pomenovanie riadkov a stĺpcov matice

2. Výber podmnožín z matíc

Pre prácu s jednotlivými hodnotami matice môžeme postupovať rovnako ako pri operáciách s vektormi. Z uvedeného vyplýva, že do hranatých zátvoriek za názvom objektu, v tomto prípade matice, vpisujeme poradové čísla poprípade výrazy na určenie riadku a stĺpca, s ktorými chceme pracovať. Keďže matica má riadky a stĺpce, do hranatých zátvoriek píšeme dve čísla alebo výrazy. Prvé sa vzťahuje na riadok a druhé na stĺpec.

```
> matica
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> matica[2,2]
[1] 5
> matica[c(1,2),3]
[1] 7 8
> 
```

Obr. 4.8.23 Výber podmnožiny matice

Na predchádzajúcom výpise (obr. 4.8.23) sme najprv chceli, aby program vypísal len konkrétnu hodnotu, ktorá sa nachádzala na druhom riadku a v druhom stĺpci. V druhom výpise sme chceli hodnoty z prvého a druhého riadku z tretieho stĺpca. Očakávali by sme, že výsledkom bude stĺpcový vektor. Jazyk R však vypíše riadkový vektor, pretože sme našim výberom znížili počet dimenzií objektu matica. Jazyk R sa pri každej operácií snaží o zníženie počtu dimenzií objektu, ak je to možné a teda v tomto prípade sa z matice stal vektor. My však môžeme zabrániť tejto zmene a nastaviť výpis v maticovom tvare pomocou argumentu `drop=FALSE`.

```
> matica
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> matica[2,2]
[1] 5
> matica[c(1,2),3]
[1] 7 8
> matica[c(1,2),3,drop=FALSE]
     [,1]
[1,]    7
[2,]    8
>
```

Obr. 4.8.24 Zachovanie maticového tvaru pri výbere hodnôt

3. Operácie s maticami

Ak chceme získať transponovanú maticu z našej matice, použijeme príkaz `t()` a do argumentu dáme názov matice, ktorú chceme transponovať. Determinant matice získame príkazom `det()` a do argumentu dáme názov matice. Hodnosť matice zistíme príkazom `qr()$rank`.

```
> matica
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> t(matica)
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
> det(matica)
[1] 0
> qr(matica)$rank
[1] 2
>
```

Obr. 4.8.25 Transponovaná matica, determinant a hodnosť matice

Násobenie matice číslom môžeme jednoducho zapísať ako rovnicu matica krát číslo.

```
> matica1<-matrix(2:7,2,3)
> matica1 * 17
     [,1] [,2] [,3]
[1,]   34   68  102
[2,]   51   85  119
> 
```

Obr. 4.8.26 Násobenie matice číslom

Násobenie matice vektorom alebo inou maticou môžeme robiť pomocou operátora %*%, ak ich súčin existuje! (obr. 4.8.27).

```
> matica2<-matrix(1:6,3,2)
> matica1 %*% matica2
     [,1] [,2]
[1,]   28   64
[2,]   34   79
> 
```

Obr. 4.8.27 Násobenie matíc

Inverznú maticu získame príkazom `solve()`, do parametra príkazu vložíme maticu.

```
> matica3<-matica1 %*% matica2
> matica3
     [,1] [,2]
[1,]   28   64
[2,]   34   79
> solve(matica3)
           [,1]       [,2]
[1,]  2.1944444 -1.7777778
[2,] -0.9444444  0.7777778
> 
```

Obr. 4.8.28 Výpočet inverznej matice

4.8.3 Zoznamy

Zoznam je najvšeobecnejšia dátová štruktúra v jazyku R. Do zoznamu ukladáme objekty rôznych dátových typov a rôznej dĺžky. Tie nám vytvoria ďalší objekt – tabuľku, kde sa do stĺpcov vkladajú jednotlivé objekty. Výpis zoznamu ale nie je tabuľka, keďže objekty v zozname sú rôznych typov. Výpis zoznamu je vo formáte názov stĺpca (objektu)

a pod ním je obsah objektu. Do zoznamu teda môžeme vkladať vektory, polia, matice, tabuľky, ale aj ďalšie zoznamy. Zoznamy sú z tohto dôvodu najvhodnejšou dátovou štruktúrou na uchovávanie výsledkov rôznych výpočtov.

```
> matica<-matrix(vektor,3,2)
> vektor<-c(4,6,5,4,5,12)
> zoznam<-list(vektor,matica)
> zoznam
[[1]]
[1]  4  6  5  4  5 12

[[2]]
     [,1] [,2]
[1,]    4    4
[2,]    6    5
[3,]    5   12

> |
```

Obr. 4.8.29 Tvorba zoznamu

Pri výpise zoznamu je každý stĺpec označený svojim poradovým číslom v dvojitých hranatých zátvorkách. Respektíve číslo môže byť nahradené názvom stĺpca v prípade, ak je názov nastavený. Ten sa nastavuje už pri tvorbe zoznamu.

```
> list(vektor=vektor,matica=matica)
$vektor
[1]  4  6  5  4  5 12

$matica
     [,1] [,2]
[1,]    4    4
[2,]    6    5
[3,]    5   12

> |
```

Obr. 4.8.30 Tvorba zoznamu s názvami stĺpcov

Názvy stĺpcov zoznamu môžeme nastaviť dodatočne použitím príkazu `names()`, ktorý slúži na výpis názvov stĺpcov. My však môžeme pomocou jednoduchého priraďovacieho príkazu `<-` explicitne tieto názvy nastaviť.

```
> zoznam
[[1]]
[1]  4  6  5  4  5 12

[[2]]
     [,1] [,2]
[1,]    4    4
[2,]    6    5
[3,]    5   12

> names(zoznam)
NULL
> names(zoznam)<-c("vektor","matica")
> zoznam
$vektor
[1]  4  6  5  4  5 12

$matica
     [,1] [,2]
[1,]    4    4
[2,]    6    5
[3,]    5   12
```

Obr. 4.8.31 Doplnenie názvov stĺpcov do existujúceho zoznamu

4.8.4 Dátové tabuľky a operácie s nimi

Dátová tabuľka je špeciály typ zoznamu. Je to dvojrozmerné pole. Vytváranie dátovej tabuľky prebieha úplne rovnako ako v prípade vytvárania zoznamu. Pre zjednodušenie budeme používať tabuľku, kde do stĺpcov budeme vkladať vektory rôznych typov avšak rovnakej dĺžky. Ako sme už načrtli, dátová tabuľka musí mať v rámci jedného stĺpca hodnoty rovnakého dátového typu. Ostatné stĺpce môžu byť aj iných dátových typov. Taktiež v dátovej tabuľke by všetky stĺpce mali mať rovnaký počet riadkov. Operácie s dátovými tabuľkami si rozdelíme do týchto činností: definovanie, výber podmnožín, triedenie údajov, pridávanie a odoberanie stĺpcov a spájanie tabuliek.

1. Definovanie tabuliek

Na obr. 4.8.32 je vidieť príklad tvorby dátovej tabuľky. Jedná sa o približné údaje samosprávnych krajov Slovenskej republiky, tabuľka obsahuje údaje o počte obyvateľov a rozlohe v kilometroch štvorcových.

```
> data.frame(počet_obyvateIov=c(606537,657119,792991,706375,807011,599859,
+ 561525,697502),rozloha=c(2052.7,9454.8,6755,6343.4,8973.7,4501.9
+ ,4146.7,6808.8))
  počet_obyvateIov rozloha
1           606537  2052.7
2           657119  9454.8
3           792991  6755.0
4           706375  6343.4
5           807011  8973.7
6           599859  4501.9
7           561525  4146.7
8           697502  6808.8
>
```

Obr. 4.8.32 Tvorba jednoduchej dátovej tabuľky

Takto zadaný príkaz však vytvorí len výpis a údaje sa neuložia. Preto treba tabuľku vložiť do objektu. Na základe uvedeného prvý riadok na obrázku obsahuje priraďovací príkaz `<-`. Chýba nám aj pomenovanie jednotlivých riadkov, aby bolo jednoznačné, ktoré údaje patria ku ktorému kraju. Jazyk R implicitne nastavuje názvy riadkov tak, že im priradí poradové číslo riadku. Riadky pomenujeme pridaním argumentu `row.names()`.

```
> Tab<-data.frame(počet_obyvateIov=c(606537,657119,792991,706375,807011,599859,
+ 561525,697502),rozloha=c(2052.7,9454.8,6755,6343.4,8973.7,4501.9
+ ,4146.7,6808.8),
+ row.names=c("Bratislavský kraj","Banskobystrický kraj","Košický kraj",
+ "Nitriansky kraj","Prešovský kraj","Trenčiansky kraj","Trnavský kraj",
+ "Žilinský kraj"))
> Tab
                    počet_obyvateIov rozloha
Bratislavský kraj             606537  2052.7
Banskobystrický kraj          657119  9454.8
Košický kraj                  792991  6755.0
Nitriansky kraj               706375  6343.4
Prešovský kraj                807011  8973.7
Trenčiansky kraj              599859  4501.9
Trnavský kraj                 561525  4146.7
Žilinský kraj                 697502  6808.8
>
```

Obr. 4.8.33 Pomenovanie stĺpcov a riadkov tabuľky

2. Výber podmnožín dátových tabuliek

Rovnako ako pri vektoroch a maticiach aj pri dátových tabuľkách je možné robiť výber hodnôt z tabuľky. Slúžia k tomu rovnako ako v ostatných prípadoch hranaté zátvorky a vzhľadom na to, že dátová tabuľka je v podstate svojim formátom podobná matici, aj výber hodnôt sa robí rovnako ako z matice. Stĺpce a riadky môžeme vyberať buď

ako pri maticiach pomocou poradového čísla daného stĺpca alebo riadku alebo pomocou názvu riadku alebo stĺpca v úvodzovkách. Pri výbere stĺpca stačí zadať len jeho názov alebo poradové číslo. Ale ak chceme vyberať riadok, treba zátvorku rozdeliť na dve časti čiarkou ([,]), prvá časť je učená pre riadok a druhá časť pre stĺpec.

```
> Tab[2]
                    rozloha
Bratislavský kraj     2052.7
Banskobystrický kraj  9454.8
Košický kraj          6755.0
Nitriansky kraj       6343.4
Prešovský kraj        8973.7
Trenčiansky kraj      4501.9
Trnavský kraj         4146.7
Žilinský kraj         6808.8
> Tab["rozloha"]
                    rozloha
Bratislavský kraj     2052.7
Banskobystrický kraj  9454.8
Košický kraj          6755.0
Nitriansky kraj       6343.4
Prešovský kraj        8973.7
Trenčiansky kraj      4501.9
Trnavský kraj         4146.7
Žilinský kraj         6808.8
```

Obr. 4.8.34 Výber hodnôt z tabuľky podľa stĺpcov

Ak by sme chceli vyberať hodnoty podľa riadkov, vyzeralo by to ako na obr. 4.8.35 (čiarkou sme jednoznačne určili, že vyberáme podľa riadkov).

```
> Tab[1,]
                  počet_obyvateľov rozloha
Bratislavský kraj           606537  2052.7
> Tab["Bratislavský kraj",]
                  počet_obyvateľov rozloha
Bratislavský kraj           606537  2052.7
>
```

Obr. 4.8.35 Výber hodnôt z tabuľky podľa riadkov

Nie je problém dostať sa ku konkrétnej hodnote zadaním kompletných súradníc do hranatých zátvoriek.

```
> Tab["Bratislavský kraj","rozloha"]
[1] 2052.7
>
```

Obr. 4.8.36 Výber hodnôt z tabuľky kombinovaným výberom

V predchádzajúcich prípadoch boli výpisy vždy vo formáte zoznamu, konkrétne dátovej tabuľky. Pre väčšinu výpočtov však potrebujeme zo zoznamu alebo dátovej tabuľky vytiahnuť hodnoty alebo objekty konkrétneho dátového typu. Ak by sme chceli z našej dátovej tabuľky vybrať vektor, ktorého hodnoty budú počty obyvateľov za jednotlivé samosprávne kraje, použijeme operátor `$`. Takýto zápis ma formát meno tabuľky, operátor `$`, meno objektu(stĺpca). Alternatívou je použitie dvojitých hranatých zátvoriek, do ktorých zadáme poradové číslo objektu(stĺpca).

```
> Tab$počet_obyvateľov
[1] 606537 657119 792991 706375 807011 599859 561525 697502
> Tab[[1]]
[1] 606537 657119 792991 706375 807011 599859 561525 697502
> 
```

Obr. 4.8.37 Výber stĺpca tabuľky operátorom `$` a dvojitými hranatými zátvorkami

3. Triedenie riadkov dátovej tabuľky

Zoradiť riadky môžeme pomocou príkazu `with`. Má dva parametre. Prvý je názov tabuľky a druhý je `order()`. Do argumentu `order` vkladáme názov stĺpca, na základe ktorého má prebehnúť triedenie. Môžeme zadať aj druhý názov stĺpca, ten bude jazyk R brať ako alternatívu, ak by sa v prvom stĺpci nachádzali dve rovnaké hodnoty.

```
> with(Tab,order(rozloha))
[1] 1 7 6 4 3 8 5 2
> 
```

Obr. 4.8.38 Zoradene riadkov tabuľky

Tento výstup nám ukázal, aké by bolo poradie riadkov, ak by sme chceli zoradiť riadky podľa hodnôt v stĺpci rozloha. Nás však skôr zaujíma taký výstup, v ktorom bude celá tabuľka so zoradenými riadkami. Budeme postupovať tak ako keby sme robili výber po riadkoch z tabuľky, ale do hranatých zátvoriek na miesto, kam patrí poradové číslo vyberaného riadku, zadáme náš triediaci príkaz.

```
> Tab[with(Tab,order(rozloha)),]
                    počet_obyvateľov rozloha skratka
Bratislavský kraj             606537  2052.7      BA
Trnavský kraj                 561525  4146.7      TT
Trenčiansky kraj              599859  4501.9      TN
Nitriansky kraj               706375  6343.4      NR
Košický kraj                  792991  6755.0      KE
Žilinský kraj                 697502  6808.8      ZA
Prešovský kraj                807011  8973.7      PO
Banskobystrický kraj          657119  9454.8      BB
> 
```

Obr. 4.8.39 Zoradenie riadkov tabuľky so zachovaním štruktúry tabuľky

4. Pridávanie a odoberanie stĺpcov do dátovej tabuľky

Jazyk R ponúka tri rôzne spôsoby ako pridávať objekty do zoznamu či stĺpce do tabuľky.

Prvý spôsob: použitie príkazu `data.frame()`, teda toho istého príkazu, ktorým sme tabuľku vytvorili. Postup je jednoduchý. Namiesto prvého argumentu vložíme názov našej dátovej tabuľky. Druhým argumentom je potom názov nového stĺpca, znamienko rovnosti a hodnoty.

```
> Tab
                    počet_obyvateľov rozloha
Bratislavský kraj             606537  2052.7
Banskobystrický kraj          657119  9454.8
Košický kraj                  792991  6755.0
Nitriansky kraj               706375  6343.4
Prešovský kraj                807011  8973.7
Trenčiansky kraj              599859  4501.9
Trnavský kraj                 561525  4146.7
Žilinský kraj                 697502  6808.8
> Tab<-data.frame(Tab,skratka=c("BA","BB","KE","NR","PO","TN","TT","ZA"))
> Tab
                    počet_obyvateľov rozloha skratka
Bratislavský kraj             606537  2052.7      BA
Banskobystrický kraj          657119  9454.8      BB
Košický kraj                  792991  6755.0      KE
Nitriansky kraj               706375  6343.4      NR
Prešovský kraj                807011  8973.7      PO
Trenčiansky kraj              599859  4501.9      TN
Trnavský kraj                 561525  4146.7      TT
Žilinský kraj                 697502  6808.8      ZA
> 
```

Obr. 4.8.40 Pridanie stĺpca do tabuľky

Druhý spôsob: podobný prvému spôsobu, len namiesto príkazu `data.frame()` použijeme príkaz `transform()`. Argumenty sú zapísané rovnako.

```
> Tab<-transform(Tab,skratka=c("BA","BB","KE","NR","PO","TN","TT","ZA"))
> Tab
                    počet_obyvateľov rozloha skratka
Bratislavský kraj             606537  2052.7      BA
Banskobystrický kraj          657119  9454.8      BB
Košický kraj                  792991  6755.0      KE
Nitriansky kraj               706375  6343.4      NR
Prešovský kraj                807011  8973.7      PO
Trenčiansky kraj              599859  4501.9      TN
Trnavský kraj                 561525  4146.7      TT
Žilinský kraj                 697502  6808.8      ZA
> |
```

Obr. 4.8.41 Pridanie stĺpca do tabuľky príkazom transform

Tretí spôsob: najjednoduchší - pomocou operátora $ vo formáte názov tabuľky, operátor $, názov nového stĺpca so znamienkom rovnosti, za ktorým zadávame hodnoty.

```
> Tab$skratka=c("BA","BB","KE","NR","PO","TN","TT","ZA")
> Tab
                    počet_obyvateľov rozloha skratka
Bratislavský kraj             606537  2052.7      BA
Banskobystrický kraj          657119  9454.8      BB
Košický kraj                  792991  6755.0      KE
Nitriansky kraj               706375  6343.4      NR
Prešovský kraj                807011  8973.7      PO
Trenčiansky kraj              599859  4501.9      TN
Trnavský kraj                 561525  4146.7      TT
Žilinský kraj                 697502  6808.8      ZA
> |
```

Obr. 4.8.42 Pridanie hodnôt do tabuľky pomocou operátora $

Na odstránenie stĺpca stačí nastaviť hodnotu tohto stĺpca na NULL. Na to poznáme dva spôsoby, ktoré sa líšia iba v tom, ako vyznačiť to, čo ideme mazať. Vyznačiť stĺpec tabuľky môžeme hranatými zátvorkami alebo alternatívne pomocou operátora $.

```
> Tab$skratka=NULL
> Tab["skratka"]=NULL
> Tab
                    počet_obyvateľov rozloha
Bratislavský kraj             606537  2052.7
Banskobystrický kraj          657119  9454.8
Košický kraj                  792991  6755.0
Nitriansky kraj               706375  6343.4
Prešovský kraj                807011  8973.7
Trenčiansky kraj              599859  4501.9
Trnavský kraj                 561525  4146.7
Žilinský kraj                 697502  6808.8
> |
```

Obr. 4.8.43 Odstránenie stĺpca z tabuľky

5. Spájanie dátových tabuliek

Ak potrebujeme spojiť dve tabuľky na základe hodnôt v konkrétnom stĺpci, použijeme príkaz `merge()`. Prvé dva argumenty sú názvy tabuliek. Tretí argument je `by`. Ten slúži na určenie stĺpca, na základe ktorého ideme tabuľky spájať. V prípade, že sa stĺpec nevolá rovnako v oboch tabuľkách, musíme tento argument spresniť. To spravíme tak, že zaňho dáme bodku a názov tabuľky znamienko rovnosti a názov stĺpca. V takom prípade je potrebné tento argument zapísať pre obidve tabuľky.

Posledným argumentom je `all`. Ten je implicitne nastavený na FALSE, čo znamená, že výsledkom spájania je len prienik tabuliek. Ak ho nastavíme na `TRUE`, výsledkom sú všetky hodnoty oboch tabuliek a tam, kde hodnoty neexistujú, je nastavená hodnota NA. Tento argument je možné spresniť aj tak, že určíme hodnoty ktorej tabuľky majú byť úplne zobrazené. Zápis takéhoto argumentu by bol `all.názov_tabuľky`. Spájanie tabuliek si predvedieme na príklade.

Príklad 4.1

Máme tabuľku **Tab** (viď Príloha č. 1). Obsahuje stĺpce rozloha a skratka. V stĺpci rozloha sú približné údaje o rozlohách samosprávnych krajov a v stĺpci skratka sú skratky krajských miest.

Úlohy:

Vytvorte tabuľku **Mestá**, ktorá:

a) bude obsahovať stĺpec názov, v ktorom budú názvy krajských miest a stĺpec značky, v ktorom budú skratky názvov miest,
b) následne spojte tabuľku **Mestá** s tabuľkou **Tab** z obr. 4.8.33.

Riešenie:

a) Príkaz na vytvorenie tabuľky **Mestá** môžeme vidieť na obr. 4.8.44

```
> Mestá<-data.frame(názov=c("Bratislava","Banská Bystrica","Košice"
+ ,"Nitra","Prešov","Trenčín","Trnava","Žilina"),
+ značka=c("BA","BB","KE","NR","PO","TN","TT","ZA"))
> Mestá
           názov značka
1      Bratislava     BA
2 Banská Bystrica     BB
3          Košice     KE
4           Nitra     NR
5          Prešov     PO
6         Trenčín     TN
7          Trnava     TT
8          Žilina     ZA
>
```

Obr. 4.8.44 Príkaz na tvorbu tabuľky Mestá

b) Príkaz na prepojenie tabuľky **Mestá** s tabuľkou **Tab** môžeme vidieť na obr. 4.8.45.

```
> merge(Tab,Mestá, by.x="skratka",by.y="značka",all=TRUE)
  skratka počet_obyvateľov rozloha           názov
1      BA           606537  2052.7      Bratislava
2      BB           657119  9454.8 Banská Bystrica
3      KE           792991  6755.0          Košice
4      NR           706375  6343.4           Nitra
5      PO           807011  8973.7          Prešov
6      TN           599859  4501.9         Trenčín
7      TT           561525  4146.7          Trnava
8      ZA           697502  6808.8          Žilina
>
```

Obr. 4.8.45 Príkaz na zlúčenie tabuliek

4.8.5 Databázy

Pri databázach väčšinou hovoríme o veľkom objeme dát, ktoré sa tvorili za určitým účelom v prostrediach iných ako je prostredie jazyka R. Takéto dáta sú vytvorené v programoch ako napríklad *MS Excell* alebo *MS Access*. Poprípade môže ísť o databázy, ktoré sa nachádzajú na webových lokalitách. Tieto sú spracovávané priamo príkazmi databázového jazyka *SQL*. Najčastejšie sa v tejto oblasti môžeme stretnúť so systémom *MySQL*, ku ktorému sa webové stránky pripájajú pomocou vnorených príkazov *SQL* do jazyka *PHP*. Tieto príkazy sú však univerzálne a dajú sa zakomponovať do kódu jazyka C++, ktorý používa aj jazyk R. V podkapitole sa budeme najprv venovať načítaniu takýchto dát do prostredia programovacieho jazyka R. Pre načítavanie databáz viď Príklad 4.2.

Príklad 4.2

Majme tri databázy, dve z nich budeme mať uložené na pevnom disku ***D*** počítača v zložke ***Databázy*** a jedna bude na webovom serveri v prostredí *MySQL*.

1. Prvá databáza je v programe MS Excel a obsahuje tabuľku Klapky a hovory.
2. Druhá databáza je v prostredí MS Access, obsahuje viaceré tabuľky. My z nej potrebujeme vytiahnuť tabuľku Mená a klapky.
3. Tretia databáza je na webovom serveri.

Úlohy:

a) Nahrajte databázu z prostredia *MS Excel* do prostredia jazyka R.
b) Nahrajte databázu z prostredia *MS Access* do prostredia jazyka R.
c) Nahrajte databázu z prostredia *MySQL* do prostredia jazyka R.

Riešenie:

a) Na prácu s dokumentmi z prostredia MS Excel budeme potrebovať vybraný doplnkový balíček pre jazyk R. Je ním balíček ***gdata*** obsahujúci funkciu na načítavanie súborov z programu *MS Excel*. Okrem tohto balíčka nainštalovaného a načítaného do prostredia jazyka R potrebujeme aj nainštalovať program ***Pearl***. Je to programovací jazyk s licenciou *freeware* (zdarma) voľne dostupný na internete[11]. Po doinštalovaní balíka ***gdata*** a programu ***Pearl*** môžeme využívať funkcie pre prácu s MS Excel.

   Ak má súbor v prostredí MS Excel iba jeden pracovný hárok, údaje z neho načítame jednoducho príkazom: `read.xls()`. Ak má súbor viacero pracovných hárkov, najprv ich všetky uložíme do objektu `wk` príkazom `loadWorkbook()` a potom z nich vyberieme hárok, z ktorého chceme načítať tabuľku príkazom `readWorksheet()`. Pre prácu s týmito príkazmi musíme doinštalovať balík ***XLConnect***.
   Na obr. 4.8.46 je riešenie úlohy a).

[11] http://strawberryperl.com/

```
> library(gdata)
> Db1 = read.xls("D:/Databazy/Klapky a hovory.xlsx")
> library(XLConnect)
> wk = loadWorkbook("D:/Databazy/Klapky a hovory.xlsx")
> db1 = readWorksheet(wk, sheet="klapky_a_hovory")
> db1
    ID Klapka Hovorov             Trvanie Cena.bez.DPH Cena.s.DPH
1  103   2360      57 1899-12-31 03:50:47        14.87          0
2  105   2382     301 1899-12-31 16:50:49        53.68          0
3  106   2468       8 1899-12-31 00:18:55         0.56          0
4  107   2524      62 1899-12-31 01:37:55         8.10          0
5  108   2588       2 1899-12-31 00:02:14         0.07          0
6  109   2871      14 1899-12-31 00:31:09         4.51          0
7  110   2872       4 1899-12-31 00:07:11         0.21          0
8  111   3895       8 1899-12-31 00:10:34         0.60          0
9  112   3995      13 1899-12-31 00:11:58         0.64          0
10 113   3998       9 1899-12-31 01:01:22         4.53          0
11 114   4009       1 1899-12-31 00:00:05         0.00          0
12 115   4019      10 1899-12-31 00:30:50         0.97          0
13 116   4058       1 1899-12-31 00:01:48         0.39          0
14 117   4137       9 1899-12-31 00:13:37         0.43          0
```

Obr. 4.8.46 Súbor príkazov na pripojenie databázy z prostredia MS Excel

b) Pre prácu s prostredím MS Access budeme potrebovať balíček RODBC. Po doinštalovaní balíčka môžeme prejsť k potrebným príkazom. Najprv si musíme uložiť súborovú cestu k databáze. Využijeme príkaz `file.path()` a cestu si uložíme do objektu `cestadb2`. Následne použijeme príkaz na pripojenie do databázy `odbcConnectAccess()`, uložíme si ho do objektu pripojenie. A potom už do objektu `db2` načítame našu tabuľku príkazom `sqlFetch()`.

Riešenie úlohy b) je na obr. 4.8.47.

```
> library(RODBC)
> cestadb2<- file.path("D:/Databazy/Mena a klapky.accdb")
> pripojenie <- odbcConnectAccess2007(cestadb2)
> db2 <- sqlFetch(channel,"Mená a klapky")
> db2
   Klapka                      Priezvisko       Meno Uctovacie  stredisko
1    2382      PETCHEM-ODBYT Fax c.m.2.12       <NA>               223100
2    2468                        KALIVODA      Pavol               237111
3    2524                       SUCHANOVA        FAX               282800
4    2588                        CSALAOVA       Reka               281000
5    2630                            <NA>       <NA>                   NA
6    2871                         MANCZAL      Juraj               282900
7    2872                           PECHA      Peter               282900
8    3895                          VILAGI      Jozef               237300
9    3995                            OLES      Jozef               237112
10   3998                        VACLAVEK   Stanislav               237111
11   4009                   GAJDOSECHOVA    Katarina               281000
12   4019                           PETHO       Zsolt               200000
13   4058                         SZABOVA     Renata               222020
14   4137                       CERNOCHOVA       Lenka               213030
```

Obr. 4.8.47 Súbor príkazov na pripojenie databázy z prostredia MS Access

c) Jazyk R má k dispozícii balíček pre prácu v prostredí *MySQL*, ktorý dokáže jednoducho prepojiť prostredie jazyka R s databázou, ktorá je na serveri *MySQL*. Spomínaným balíkom je ***RMySQL*** ten ale nie je dostupný v kompilovateľnej pre Windowsové prostredie jazyka R. Aj napriek tomu sa tento balík dá v operačnom systéme Windows doinštalovať. Potrebujeme k tomu prídavný program ***Rtools*** pre jazyk R. A zároveň potrebujeme v operačnom systéme nainštalovavý ***MySQL–Server***. Pre náročnosť tohto riešenia nebudeme túto úlohu ďalej rozpracovávať. K riešeniu tohto problému je možné nájsť mnoho článkov[12] na internete.

4.8.6 Grafické nástroje

Ako sme už spomínali, používateľské rozhranie jazyka R má samostatné okno pre prácu s grafikou (*R Graphics*). V tomto okne sa nám zobrazujú grafy, ktoré v konzolovom okne spustíme. Jazyk R ponúka veľký výber rôznych grafov, z toho je veľa z nich obsiahnutých už v základnej verzii programového prostredia jazyka R. K dispozícii sú aj ďalšie podporné balíčky (packages) pre prácu s grafmi v programovom prostredí jazyka R.

V tejto podkapitole budeme zobrazovať grafy:

- graf pomocou základného príkazu na tvorbu grafov `plot()`,
- koláčový graf,
- stĺpcový graf (barplot),
- krabicový graf (boxplot).

Neskôr si v podkapitole venovanej štatistickým analýzam zobrazíme histogram.

Graf príkazom plot

Najzákladnejším príkazom na tvorbu grafov, s ktorým v jazyku R pracujeme, je `plot()`. Má mnoho parametrov, ktoré si ďalej ukážeme. Základným argumentom je zdrojový objekt, z ktorého ideme graf tvoriť. Ak by sme zadali len tento jeden argument, jazyk R nastaví všetky ostatné implicitne a v grafickom okne sa zobrazí výstup. Pri

[12] http://www.ahschulz.de/2013/07/23/installing-rmysql-under-windows/

jednoduchých vstupných objektoch ako napríklad vektor alebo dátová tabuľka o dvoch stĺpcoch je výsledkom bodový graf. V prípade, že v prvom parametri je len jeden vektor, na osi x je poradie hodnoty vo vektore a na osi y sú jednotlivé hodnoty. Ak by sme za prvý parameter zvolili dátovú tabuľku o dvoch stĺpcoch, na osi x sú hodnoty jedného stĺpca a na osi y sú hodnoty druhého stĺpca.

Na nasledujúcom obrázku si ukážeme takto vytvorený graf najprv na jednoduchom vektore a potom na tabuľke **Tab**, ktorú sme vytvorili pri operáciách s dátovými tabuľkami. Pre výber podmnožiny z tabuľky **Tab** použijeme operátor `$`.

```
> plot(c(7,5,6,1,2,4,1,1))
> plot(Tab$rozloha)
>
```

Obr. 4.8.48 Tvorba jednoduchého bodového grafu príkazom plot

Takéto automatické grafy pre nás však nie sú zaujímavé. Preto prejdeme aj k ostatným parametrom, ktoré upravia príkaz `plot` tak, aby náš graf bol zrozumiteľný. Tieto parametre sú použiteľné pre akýkoľvek graf v jazyku R.

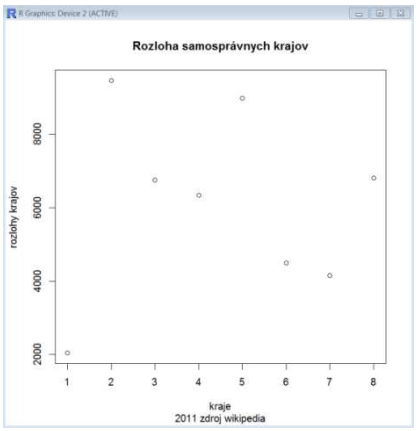
Predvedieme si zmenu grafu zobrazujúceho rozlohu samosprávnych krajov Slovenskej republiky. Údaje do tohto grafu máme z dátovej tabuľky **Tab**, ktorú, ktorú sme vytvorili pri operáciách s dátovými tabuľkami.

Pridáme parametre:

`main`	reprezentuje názov grafu,
`sub`	popis grafu, nachádza sa pod grafom,
`xlab, ylab`	názvy osí grafu,
`xlym, ylym`	zadáva sa dvojprvkovým vektorom a slúži ako ohraničenie grafu (nastavuje maximálne a minimálne hodnoty osí),
`type`	nastaví typ zobrazovaného grafu.

```
> plot(Tab$rozloha,main="Rozloha samosprávnych krajov",xlab="kraje",
+ ylab="rozlohy krajov",sub="2011 zdroj wikipedia")
>
```

Obr. 4.8.49 Príkaz plot na tvorbu grafu s popisnými parametrami



Obr. 4.8.50 Upravený bodový graf príkazom plot

Graf zobrazený príkazom `plot()` má osem rôznych spôsobov ako graficky zobraziť dáta. Nastavujú sa pomocou parametra `type`, do ktorého sa priraďujú písmena podľa typu grafu.

`type="p"` samostatné body(implicitné nastavenie),

`type="l"` čiarový graf,

`type="b"` prerušovaný čiarový graf s bodmi,

`type="c"` prerušovaný čiarový graf bez bodov,

`type="o"` čiarový graf s bodmi,

`type="h"` graf vertikálnych čiar,

`type="s"` schodový graf (horizontálny),

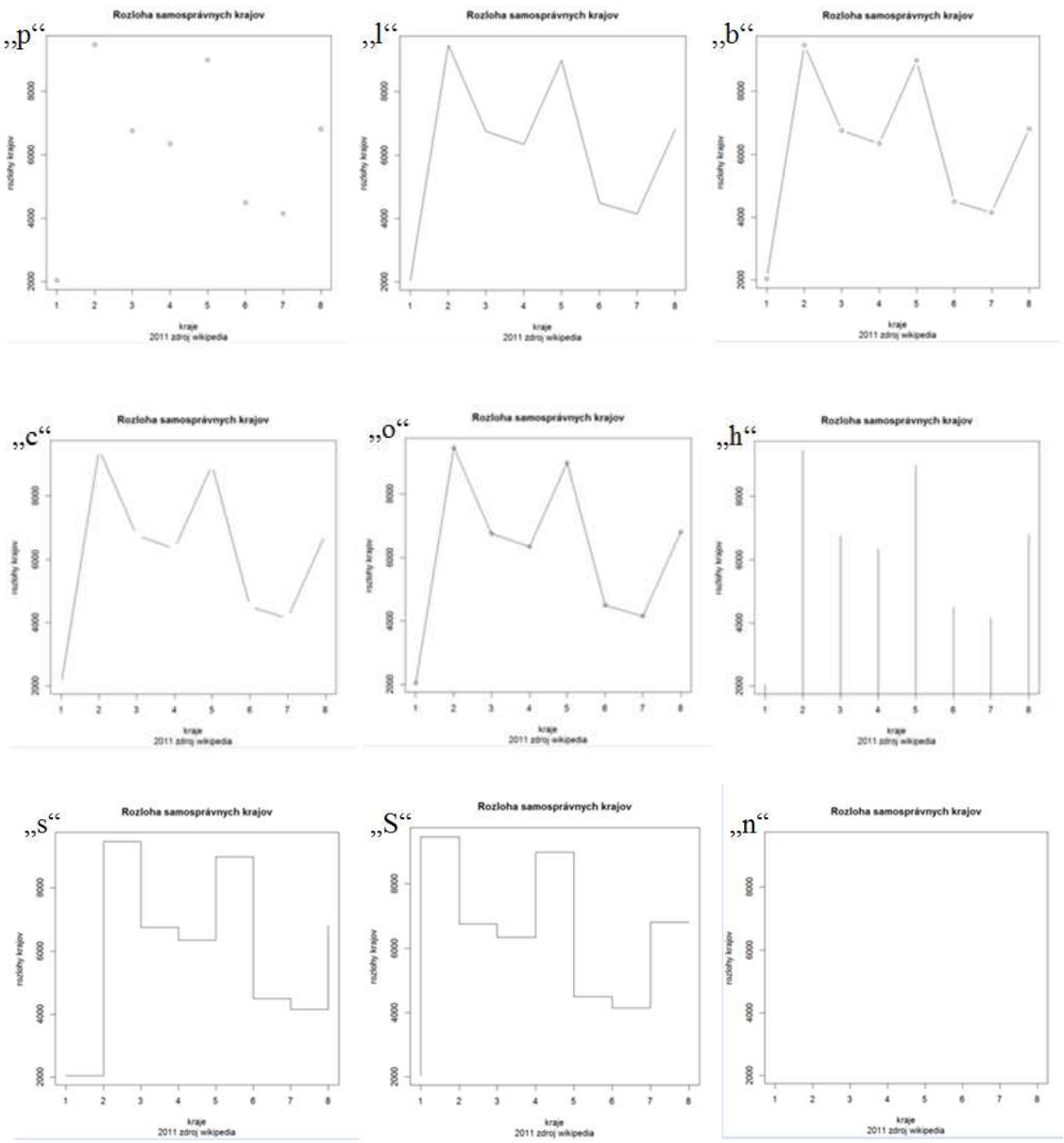
`type="S"` schodový graf začínajúci (vertikálny),

`type="n"` len súradnicový systém,

Predvedieme si jednotlivé modifikácie grafu zobrazeného príkazom plot.

```
> plot(Tab$rozloha,main="Rozloha samosprávnych krajov",xlab="kraje",
+ ylab="rozlohy krajov",sub="2011 zdroj wikipedia",type="p")
> plot(Tab$rozloha,main="Rozloha samosprávnych krajov",xlab="kraje",
+ ylab="rozlohy krajov",sub="2011 zdroj wikipedia",type="l")
> plot(Tab$rozloha,main="Rozloha samosprávnych krajov",xlab="kraje",
+ ylab="rozlohy krajov",sub="2011 zdroj wikipedia",type="b")
> plot(Tab$rozloha,main="Rozloha samosprávnych krajov",xlab="kraje",
+ ylab="rozlohy krajov",sub="2011 zdroj wikipedia",type="c")
> plot(Tab$rozloha,main="Rozloha samosprávnych krajov",xlab="kraje",
+ ylab="rozlohy krajov",sub="2011 zdroj wikipedia",type="o")
> plot(Tab$rozloha,main="Rozloha samosprávnych krajov",xlab="kraje",
+ ylab="rozlohy krajov",sub="2011 zdroj wikipedia",type="h")
> plot(Tab$rozloha,main="Rozloha samosprávnych krajov",xlab="kraje",
+ ylab="rozlohy krajov",sub="2011 zdroj wikipedia",type="s")
> plot(Tab$rozloha,main="Rozloha samosprávnych krajov",xlab="kraje",
+ ylab="rozlohy krajov",sub="2011 zdroj wikipedia",type="S")
> plot(Tab$rozloha,main="Rozloha samosprávnych krajov",xlab="kraje",
+ ylab="rozlohy krajov",sub="2011 zdroj wikipedia",type="n")
>
```

Obr. 4.8.51 Príkaz `plot` s parametrom na modifikáciu typu grafu



Obr. 4.8.52 Modifikácie typu grafu zobrazované príkazom plot

Koláčový graf

Pre zobrazenie tabuľky **Tab** (viď Príloha č. 1), ktorá obsahuje približné údaje o rozlohe samosprávnych krajov Slovenskej republiky, by bol vhodnejší graf, ktorý by znázornil podiel jednotlivých rozlôh na celku. Pre tieto účely je vhodnejšie tvoriť koláčový graf. Zobrazenie tabuľky si ukážeme na príklade 4.3.

Príklad 4.3

Opätovne využime údaje z tabuľky **Tab**, ktorá obsahuje stĺpce rozloha a skratka. V stĺpci rozloha sú približné údaje o rozlohe jednotlivých samosprávnych krajov a stĺpec skratka obsahuje skratku názvu krajského mesta.

Úloha:

Zobrazte údaje z tabuľky **Tab** koláčovým grafom.

Riešenie:

Na tvorbu koláčových grafov v prostredí jazyka R používame príkaz `pie()`.

Príkaz `pie()` na tvorbu koláčového grafu má viacero parametrov, pričom prvý je názov objektu, odkiaľ čerpáme zobrazované dáta.

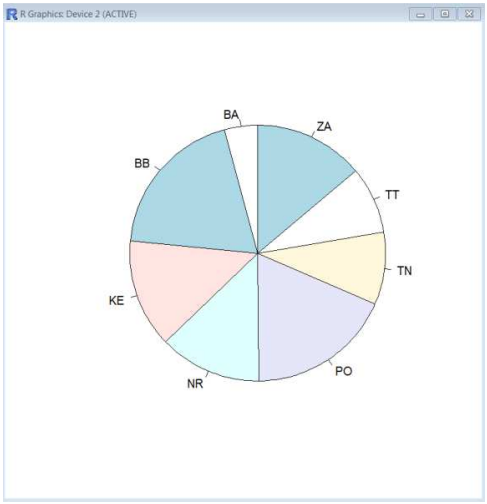
Pri tabuľke **Tab** sú to rozlohy samosprávnych krajov nachádzajúce sa v stĺpci rozloha. Pre výber týchto hodnôt použijeme parameter `$`. Vstupné dáta potom budú v tvare `Tab$rozloha`.

Okrem toho obsahuje ešte parametre:

`labels`	názvy jednotlivých dielov,
`edges`	počet vrcholov grafu (ak je nastavený, má tvar polygónu),
`clockwise`	smer vkladania hodnôt (v alebo proti smeru hodinových ručičiek),
`init.angle`	počiatočný uhol (slúži na otáčanie grafu).

```
> pie(Tab$rozloha,labels=Tab$skratka,init.angle=90)
>
```

Obr. 4.8.53 Príkaz na zobrazenie koláčového grafu



Obr. 4.8.54 Koláčový graf rozlohy samosprávnych krajov

Stĺpcový graf

Stĺpcový graf zobrazuje dáta do obdĺžnikov (vertikálnych alebo horizontálnych v závislosti od nastavenia). Výhodou stĺpcového grafu je, že doňho môžeme zobraziť viacero rovnakých dát. Tie môžu byť členené napríklad podľa rokov. Tým získame prehľadný graf o tom ako sa naše údaje v priebehu rokov vyvinuli.

Príklad 4.4

Zobrazenie stĺpcového grafu si znázorníme na tabuľke **Tab1**. Tabuľka **Tab1** je modifikáciou tabuľky **Tab**, s ktorou sme pracovali v predchádzajúcich príkladoch. Rovnako je súčasťou Prílohy č.1. Tabuľka **Tab1** obsahuje stĺpce počet obyvateľov a počet obyvateľov2. V stĺpci počet obyvateľov2 sú dáta len na ilustračné účely.

Úloha:
Zobrazte údaje z tabuľky **Tab1** do stĺpcového grafu.

Riešenie:

Na zobrazenie tabuľky **Tab1** musíme túto tabuľku najprv upraviť do maticového tvaru, z ktorého už príkazom `barplot()` môžeme zostrojiť stĺpcový graf. Ďalším problémom je, že hodnoty počtu obyvateľov sú v tabuľke **Tab1** zapísané celým číslom, čo predstavuje problém, keďže tieto hodnoty budú zobrazené na osi grafu. Hodnoty o počte

obyvateľov teda upravíme na tvar, ktorý bude vyjadrovať počet obyvateľov v tis. obyvateľov.

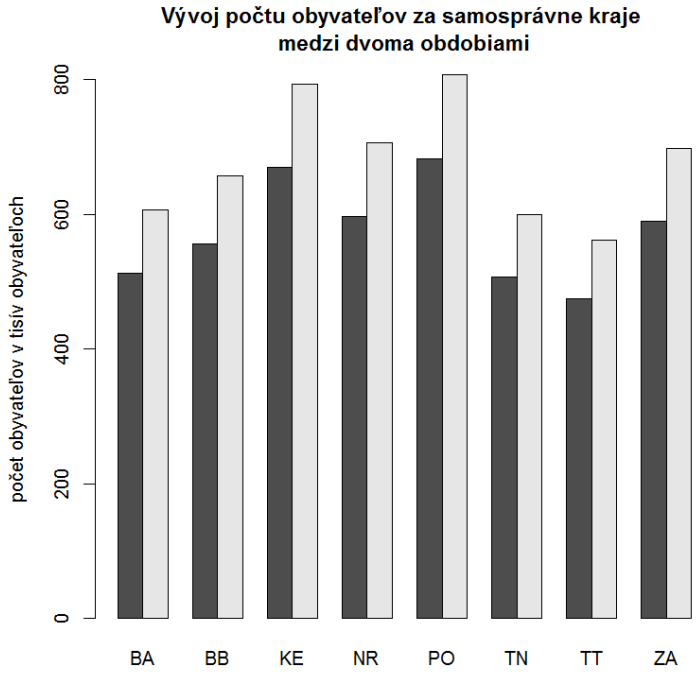
1. Hodnoty upravíme jednoduchým priraďovacím príkazom, kde do aktuálnych hodnôt priradíme hodnoty predelené tisíckou.
2. Maticový tvar tabuľky získame príkazom `rbind()`, ktorý spája vektory do maticového tvaru. Písmeno **r** v názve príkazu hovorí o tom, že v matici budú vektory usporiadané do riadkov[13] (jeden vektor je jeden riadok matice).
3. Argumenty príkazu barplot() sú:
   - `beside` definuje, či údaje majú byť zobrazené nad alebo vedľa seba,
   - `names.arg` definuje názvy stĺpcov zobrazených na grafe.
4. Okrem týchto argumentov môžeme pri tvorbe stĺpcového grafu využiť aj argumenty `main`, `xlab`, `ylab`, `xlim`, `ylim`, `sub`, ktoré sme opísali pri grafoch tvorených príkazom `plot()`.

Postupnosť príkazov na zobrazenie stĺpcového grafu z tabuľky **Tab1** je na nasledujúcom obrázku. Operátor `\n` v argumente `main` znamená, že text pokračuje na nový riadok.

```
> Tab1
                     počet_obyvateľov rozloha skratka počet_obyvateľov2
Bratislavský kraj              606537  2052.7      BA          512523.8
Banskobystrický kraj           657119  9454.8      BB          555265.6
Košický kraj                   792991  6755.0      KE          670077.4
Nitriansky kraj                706375  6343.4      NR          596886.9
Prešovský kraj                 807011  8973.7      PO          681924.3
Trenčiansky kraj               599859  4501.9      TN          506880.9
Trnavský kraj                  561525  4146.7      TT          474488.6
Žilinský kraj                  697502  6808.8      ZA          589389.2
> Tab1$počet_obyvateľov<-Tab1$počet_obyvateľov/1000
> Tab1$počet_obyvateľov2<-Tab1$počet_obyvateľov2/1000
> maticaTab1<-rbind(Tab1$počet_obyvateľov2,Tab1$počet_obyvateľov)
> barplot(maticaTab1,beside=TRUE,names.arg=Tab1$skratka,
+ main="Vývoj počtu obyvateľov za samosprávne kraje \nmedzi dvoma obdobiami",
+ ylab="počet obyvateľov v tis.")
>
```

Obr. 4.8.55 Súbor príkazov na tvorbu stĺpcového grafu

[13] Po stĺpcoch sa spája príkazom cbind()



Obr. 4.8.56 Stĺpcový graf vývoja počtu obyvateľov za samosprávne kraje

Krabicový graf

Krabicový graf (*boxplot*) sa využíva v popisnej štatistike. Z grafu zobrazeného príkazom `boxplot()` môžeme vyčítať informácie o súbore dát, z ktorých je graf tvorený. Hodnoty na konci čiary predeľujúcej graf na polovicu sú minimum a maximum. Hodnoty na okraji „krabice“ sú prvý a tretí kvartil. A čiara, ktorá predeľuje „krabicu“ na polovicu, predstavuje medián.

Graf si zobrazíme na údajoch z tabuľky **Zamestnanci**, konkrétne zo stĺpca mzda_k_31.1.2014. Tabuľka **Zamestnanci** je súčasťou Prílohy č. 1. Pre prácu s vybraným stĺpcom použijeme operátor `$`.

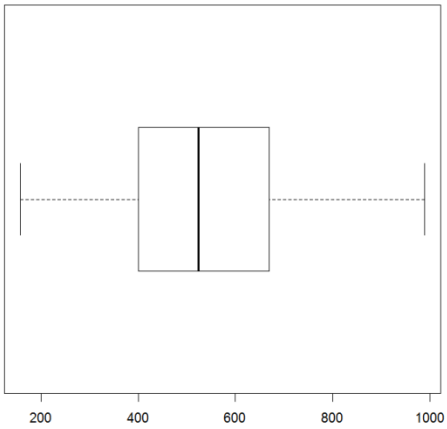
Pri tvorbe krabicového grafu používame argumenty:

`outline` nastavením na hodnotu TRUE zobrazuje aj odľahlé hodnoty,

`horizontal` nastavením na hodnotu TRUE otočí graf do horizontálnej polohy.

```
> boxplot(zamestnanci$mzda_k_31.1.2014,outline=TRUE,horizontal=TRUE)
>
```

Obr. 4.8.57 Príkaz na zobrazenie krabicového grafu miezd z tabuľky **Zamestnanci**



Obr. 4.8.58 Krabicový graf miezd z tabuľky **Zamestnanci**

4.9 Využitie jazyka R pri štatistických analýzach

Ako sme na úvod spomenuli, jazyk R má rozsiahly výber funkcií pre rôzne štatistické operácie a analýzy ako sú napríklad výpočet popisných charakteristík súboru dát, odhad lineárneho a nelineárneho modelu, testy štatistických hypotéz, analýzy časových radov a mnohé iné. Medzi štatistické analýzy, ktoré jazyk R ponúka, patria aj grafické analýzy.

V práci sa budeme bližšie venovať:

- výpočtu popisných charakteristík súboru dát, medzi ktoré zahrnieme výpočty minima a maxima, kvantilov, priemeru, mediánu, modusu, rozptylu a štandardnej odchýlky,
- histogramu,
- odhadu lineárneho regresného modelu a parametrov modelu,
- testom štatistickej významnosti parametrov modelu,
- generovaniu náhodných čísel,
- využitie funkcii na prácu s rozdeleniami pravdepodobností,
- testom dobrej zhody.

Všetky spomínané operácie popisuje príklad 4.5.

Príklad 4.5

Máme tabuľku **Zamestnanci**, obsahuje údaje o mzdách zamestnancov za mesiac január a odpracované dni za január. Tabuľka **Zamestnanci** je súčasťou Prílohy č. 2.

Úlohy:

a) Vypočítajte minimum, maximum, priemer, medián, modus, rozptyl a štandardnú odchýlku z údajov o mzdách zamestnancov a odpracovaných dňoch za mesiac január.
b) Zobrazte histogram údajov o mzdách zamestnancov za mesiac január.
c) Odhadnite lineárny regresný model opisujúci závislosť miezd zamestnancov od odpravovaných dní za mesiac január.
d) Otestujte štatistickú významnosť modelu.

Riešenie:

a) Výpočet popisných charakteristík údajov

Popisné charakteristiky minimum, maximum, prvý a tretí kvartil, priemer a medián súboru údajov môžeme získať príkazom `summary()`. Do parametra vložíme názov objektu, pre ktorý chceme robiť výpočet štatistík. V našom prípade je objektom dátová tabuľka zamestnanci, ktorú vložíme do parametra funkcie `summary()`. Jazyk R vypočíta charakteristiky pre každý stĺpec tabuľky zvlášť.

```
> summary(zamestnanci)
 mzda_k_31.1.2014 odpracované_dni_január
 Min.   :157.7    Min.   : 21.00
 1st Qu.:401.2    1st Qu.: 53.00
 Median :524.1    Median : 68.50
 Mean   :534.2    Mean   : 70.15
 3rd Qu.:665.9    3rd Qu.: 87.50
 Max.   :989.2    Max.   :130.00
> 
```

Obr. 4.9.1 Príkaz na výpočet popisných charakteristík

Modus

Pre výpočet modusu potrebujeme doinštalovať doplnkový balíček pre jazyk R, ktorý sme si spomenuli v podkapitole 4.5 o doplnkových balíčkoch. Podľa postupu

znázorneného v tejto podkapitole nainštalujeme balík ***modeest*** a následne ho pripojíme do aktívneho ***workspace*** príkazom `library()`. Balík modeest obsahuje funkciu `mlv(),` ktorú na výpočet modusu použijeme. Funkcia mlv() má dva parametre, z toho prvý je objekt, z ktorého ideme modus počítať a druhý je `method`. Ten sa mení v závislosti od dátového typu objektu. Pre naše potreby použijeme parameter `method="mfv"`, potom jazyk R berie objekt ako číselný vektor. Preto musíme údaje z tabuľky **Zamestnanci** zadávať ako výber s pomocou operátora `$`.

```
> mlv(zamestnanci$mzda_k_31.1.2014,method="mfv")
Mode (most likely value): 534.1721
Bickel's modal skewness: -0.04
Call: mlv.default(x = zamestnanci$mzda_k_31.1.2014, method = "mfv")
> mlv(zamestnanci$odpracované_dni_január,method="mfv")
Mode (most likely value): 75
Bickel's modal skewness: -0.14
Call: mlv.default(x = zamestnanci$odpracované_dni_január, method = "mfv")
>
```

Obr. 4.9.2 Príkaz na výpočet modusu

Rozptyl

Rozptyl získame príkazom `var()`. Názov var reprezentuje *variance*, čo z prekladu do slovenčiny znamená rozptyl. Na našu tabuľku ho jednoducho použijeme s parametrom, ktorým bude stĺpec našej tabuľky.

```
> var(zamestnanci$mzda_k_31.1.2014)
[1] 30274.3
>
```

Obr. 4.9.3 Príkaz na výpočet rozptylu

Iný spôsob môžeme spraviť vlastným výpočtom.

```
> sse<-(zamestnanci$mzda_k_31.1.2014-mean(zamestnanci$mzda_k_31.1.2014))^2
> sum(sse)/(length(zamestnanci$mzda_k_31.1.2014)-1)
[1] 30274.3
> var(zamestnanci$mzda_k_31.1.2014)
[1] 30274.3
>
```

Obr. 4.9.4 Výpočet rozptylu (tvorba vlastnej funkcie)

Štandardná odchýlka

Štandardná odchýlka je druhou odmocninou rozptylu a pre druhú odmocninu má R príkaz `sqrt()`. Čiže nám stačí do príkazu `sqrt()` vložiť príkaz na výpočet rozptylu `var()`. Ďalším riešením je jednoducho použiť príkaz `sd()`, ktorý odchýlku rovno vypočíta.

```
> sqrt(var(zamestnanci$mzda_k_31.1.2014))
[1] 173.9951
> sd(zamestnanci$mzda_k_31.1.2014)
[1] 173.9951
> sqrt(var(zamestnanci$odpracované_dni_január))
[1] 22.87841
> sd(zamestnanci$odpracované_dni_január)
[1] 22.87841
>
```

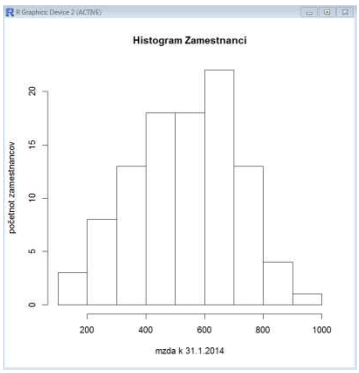
Obr. 4.9.5 Príkazy na výpočet štandardnej odchýlky

b) Histogram údajov

Využijeme grafické prostredie a zobrazíme mzdy z tabuľky **Zamestnanci** na histograme. Histogram vytvárame príkazom `hist()`. Základným parametrom je objekt, z ktorého chceme načítať údaje do histogramu. V našom konkrétnom prípade hovoríme o stĺpci mzda_k_31.1.2014 z našej tabuľky **Zamestnanci**. Pre začiatok nám na vytvorenie histogramu tento jeden parameter úplne stačí. Využijeme však vedomosti, ktoré sme získali pri práci s grafikou v jazyku R a pridáme aj parametre `main`, `xlab`, `ylab`.

```
> hist(zamestnanci$mzda_k_31.1.2014,main="Histogram Zamestnanci",
+ xlab="mzda k 31.1.2014",ylab="početnot zamestnancov")
>
```

Obr. 4.9.6 Príkaz na tvorbu histogramu



Obr. 4.9.7 Histogram

Detailné údaje o našom grafe zistíme jednoducho tak, že si celý príkaz na jeho tvorbu uložíme do objektu. Objekt nazveme **infohistogram**.

```
> infohystogram<-hist(zamestnanci$mzda_k_31.1.2014,main="Histogram Zamestnanci",
+ xlab="mzda k 31.1.2014",ylab="početnot zamestnancov")
> infohystogram
$breaks
 [1]  100  200  300  400  500  600  700  800  900 1000

$counts
[1]  3  8 13 18 18 22 13  4  1

$density
[1] 0.0003 0.0008 0.0013 0.0018 0.0018 0.0022 0.0013 0.0004 0.0001

$mids
[1] 150 250 350 450 550 650 750 850 950

$xname
[1] "zamestnanci$mzda_k_31.1.2014"

$equidist
[1] TRUE

attr(,"class")
[1] "histogram"
>
```

Obr. 4.9.8 Údaje o histograme

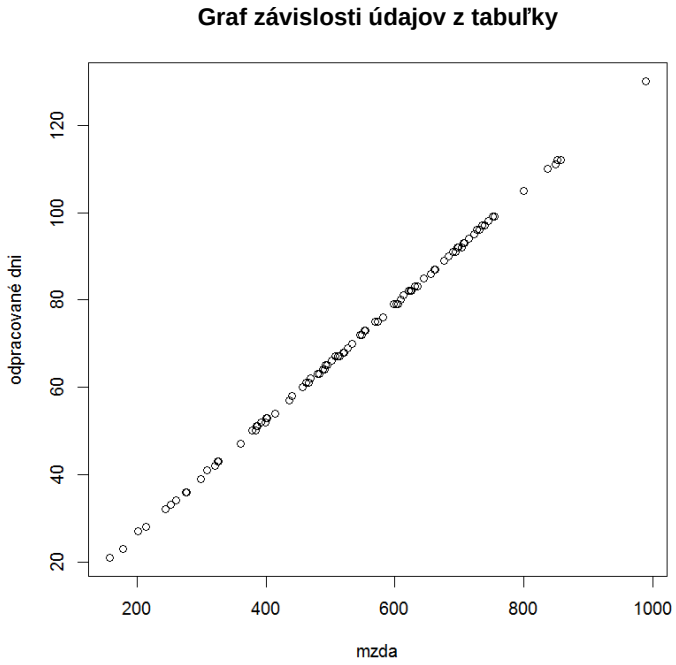
Poďme si tieto informácie rozobrať.

`breaks`	predstavuje hranice intervalov zobrazených na histograme,
`counts`	sú absolútne početnosti zamestnancov v intervaloch,
`destiny`	sú relatívne početnosti zamestnancov,
`mids`	sú stredy intervalov,
`xname`	je názov osi *x*, ten sme pridali parametrom xlab,
`equidist`	hodnota TRUE značí, že stredy intervalov sú rovnako vzdialené.

Histogram môžeme upravovať zadávaním ďalších parametrov, napríklad intervaly môžeme meniť tak, že pridáme parameter `breaks=c().` Do tohto vektora vložíme nami stanovené hranice intervalov. Pridaním parametra `labels=TRUE` si zapneme zobrazovanie početností nad intervalmi grafu.

c) Odhad lineárneho regresného modelu

Aby sme získali lepší prehľad o údajoch a závislostiach z tabuľky **Zamestnanci**, tieto údaje si najprv zobrazímr na jednoduchom grafe závislosti.



Obr. 4.9.9 Bodový graf závislosti v tabuľke **Zamestnanci**

Lineárnym regresným modelom zistíme o koľko by stúpla zamestnancovi mesačná mzda, ak by o jednu hodinu dlhšie pracoval. Odpoveď na túto otázku získame tak, že bodmi zobrazenými na grafe položíme priamku. Smernica priamky potom vysvetľuje o koľko stúpne mzda (y), ak stúpnu odpracované dni o jeden deň (x).

Regresný model má potom tvar:

$$y_i = b_0 + b_1 x_i + \mu$$

kde:

y je vysvetľovaná premenná, mesačná mzda zamestnanca v EUR,

x je vysvetľujúca premenná, odpracované dni za mesiac, v dňoch,

μ je náhodná zložka, vysvetľujúca vplyv všetkých činiteľov, ktoré neboli zahrnuté do modelu,

b_0 úrovňová konštanta, priesečník s *osou y*,

b_1 je parameter modelu, vysvetľujúci zmenu vysvetľovanej premennej y, vyvolanú zmenou vysvetľujúcej premennej x o jednotku.

Parametre modelu potom vieme odhadnúť metódou najmenších štvorcov využitím nasledovných vzťahov:

Vzťah na odhad regresného koeficienta modelu:

$$\hat{b}_1 = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2}$$

Vzťah na odhad lokujúcej konštanty:

$$\hat{b}_0 = \bar{y} - b_1 * \bar{x}$$

Dopočítaním parametrov modelu získame rovnicu vyrovnanej regresnej priamky, ktorá má tvar:

$$\hat{y}_i = \hat{b}_0 + \hat{b}_1 x_i + \mu$$

Na modelovanie budeme využívať príkaz `lm()`. Ten v jazyku R reprezentuje modelovanie lineárnych modelov. Zápis jeho parametrov sa robí tak, že do parametrov napíšeme závislú premennú, znak závislosti (v jazyku R je to operátor ~) a nezávislú premennú.

```
> lm(zamestnanci$mzda_k_31.1.2014 ~ zamestnanci$odpracované_dni_január)

Call:
lm(formula = zamestnanci$mzda_k_31.1.2014 ~ zamestnanci$odpracované_dni_január)

Coefficients:
                    (Intercept)  zamestnanci$odpracované_dni_január
                         0.7131                              7.6045

> |
```

Obr. 4.9.10 Odhad lineárneho regresného modelu

Intercept predstavuje lokujúcu konštantu $\hat{b}_0$=0,7131. Ten v podstate vyjadruje priemernú mzdu, ak by zamestnanec neodpracoval ani jeden deň.
V stĺpci zamestnanci$odpracované_dni_január máme regresný koeficient b_1=7,6045. To znamená, že ak zamestnanec odpracuje o jednu hodinu viac, prinesie mu to zvýšenie mesačnej mzdy o 7,60 eur.

d) Testy štatistickej významnosti

Aby sme mohli s výsledkami modelu z predchádzajúcej úlohy pracovať, je potrebné otestovať štatistickú významnosť parametrov modelu a aj modelu ako celku. Údaje k tomu potrebné získame príkazom `summary()`. Do parametra tohto príkazu dáme celý náš model, ktorý sme použili. Pre lepšiu prácu sme si model odložili do objektu regresia.

```
> regresia<-lm(zamestnanci$mzda_k_31.1.2014 ~ zamestnanci$odpracované_dni_január)
> summary(regresia)

Call:
lm(formula = zamestnanci$mzda_k_31.1.2014 ~ zamestnanci$odpracované_dni_január)

Residuals:
    Min      1Q  Median      3Q     Max
-4.1159 -1.9259 -0.2428  1.8604  4.6876

Coefficients:
                                Estimate Std. Error t value Pr(>|t|)
(Intercept)                      0.71309    0.74746   0.954    0.342
zamestnanci$odpracované_dni_január  7.60455    0.01013 750.333   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.307 on 98 degrees of freedom
Multiple R-squared:  0.9998,    Adjusted R-squared:  0.9998
F-statistic: 5.63e+05 on 1 and 98 DF,  p-value: < 2.2e-16

>
```

Obr. 4.9.11 Popisné charakteristiky odhadnutého modelu

Testy štatistickej významnosti parametrov

Pre parametre testujeme hypotézy:

H_0: $b_i = 0$ Parameter b_i nie je štatisticky významný.

H_1: $b_i \neq 0$ Parameter b_i je štatisticky významný.

Testovacia charakteristika Studentovho rozdelenia:

$$t_i = \frac{\hat{b}_i}{s_{b_i}}$$

Ak je hodnota testovacej charakteristiky t_i vyššia ako tabuľková hodnota pri zvolenej hladine významnosti a pri stupňoch voľnosti $[n - (k + 1)]$, kde n je rozsah súboru a k je počet vysvetľujúcich premenných v modeli, tak potom parameter b_i je významný.

Hodnoty testovacích chrarakteristík získame z výpisu príkazu `summary()` z obr. 4.8.11 a hodnoty charakteristík sa nachádzajú v stĺpci ***t value***. Hodnotu kvantilu Studentovho rozdelenia získame príkazom `qt()`. Ako parametre príkazu použijeme hladinu významnosti a počet stupňov volnosti.

```
> qt(0.975,98)
[1] 1.984467
>
```

Obr. 4.9.12 Príkaz na výpočet kvantilu Studentovho rozdelenia

Zistené hodnoty charakteristík sú:

$$t_{b_0} = 0{,}954$$

$$t_{b_1} = 750{,}333$$

Pri porovnaní hodnoty testovacej charakteristiky s hodnotou kvantilu Studentovho rozdelenia dostaneme 0,954 > 1,984467, z čoho vyplýva, že nemôžeme zamietnuť nulovú hypotézu, že parameter b_0 je štatisticky nevýznamný.

Pre parameter b_1 dostaneme 750,33 > 1,984467, z toho vyplýva, že zamietame nulovú hypotézu a prijímame hypotézu H_1: parameter b_1 je štatisticky významný.

Test významnosti modelu

Významnosť modelu sa testuje pomocou koeficienta determinácie, ten sme získali z výstupu príkazu `summary()` na obr. 4.8.11. Testovanie sa robí prostredníctvom F štatistiky.

Testujeme hypotézy:

H_0: $b_1 = b_2 = b_n = 0$ Parametre modelu sú nevýznamne.

H_1: Aspoň jeden z parametrov je významný.

F štatistika má pre testovanie koeficienta determinancie nasledovný tvar:

$$F_r = \frac{R^2/k}{(1-R^2)/[n-(k+1)]}$$

Takto získanú hodnotu *F* štatistiky porovnáme s tabuľkovou hodnotou kvantilu F štatistiky. Hodnotu kvantilu *F* štatistiky získame príkazom `qf()`. Do parametrov príkazu dáme hladinu významnosti, počet vysvetľujúcich premenných a počet stupňov voľnosti.

```
> qf(0.95,1,98)
[1] 3.938111
>
```

Obr. 4.9.13 Kvantil F štatistiky

Ak zistíme, že hodnota *F* štatistiky pre model je vyššia ako kvantil F štatistiky, potom model ako celok je štatisticky významný. Z výstupu na obr. 4.9.11 vieme, že hodnota *F* štatistiky je oveľa väčšia ako hodnota kvantilu *F* štatistiky, ktorá vyšla 3.938111, Takže zamietame nulovú hypotézu a prijímame hypotézu H_1, že model ako celok je štatisticky významný.

Generátor náhodných čísel

Jazyk R umožňuje generovať náhodné čísla z rozdelení pravdepodobností. Rovnomerné rozdelenie, kde má každé číslo rovnakú pravdepodobnosť výskytu, vyvoláme príkazom `runif()`. Príkaz má tri parametre, kde prvý určuje koľko sa bude generovať čísel a ďalšie dva sú na určenie intervalu, z ktorého sa budú čísla vyberať.

```
> runif(10,1,100)
 [1] 37.793113  9.905361 87.513753  1.386715 55.254794 68.837270 81.434376
 [8] 95.347926 60.805969 33.624234
> round(runif(10,1,100),0)
 [1] 45 18 41 28 41 41  8 72 94 73
>
```

Obr. 4.9.14 Generovanie náhodných hodnôt, ktoré sa riadia rovnomerným rozdelením

Na obr. 4.9.14 sme použili príkaz `round`, ktorý slúži na zaokrúhľovanie a s jeho pomocou sme vygenerovali celé čísla. Príkaz `round` má dva parametre. Prvým je hodnota alebo vektor, ktorý ideme zaokrúhľovať a druhým je počet desatinných miest, ktorý chceme dostať.

Poissonovo rozdelenie `rpois()` má dva parametre. Prvý parameter je (rovnako ako v predchádzajúcom prípade, ale aj v prípade ďalších rozdelení) počet hodnôt, ktoré chceme vygenerovať. Druhý parameter je parameter Poissonovho rozdelenia lambda λ (stredná hodnota a rozptyl).

```
> rpois(10,100)
 [1] 107  99  91 121  96 107 110 103 103 120
>
```

Obr. 4.9.15 Generovanie náhodných čísel Poissonovým rozdelením

Normálne rozdelenie `rnorm()` má tri parametre. Prvý parameter predstavuje počet hodnôt, ktoré ideme generovať. Druhý parameter predstavuje strednú hodnotu μ daného rozdelenia a tretí parameter predstavuje štandardnú odchýlku σ rozdelenia. Normálne rozdelenie sa vygeneruje na päť desatinných miest a je možné použiť príkaz `round` na úpravu počtu desatinných miest.

```
> rnorm(10,50,5)
 [1] 51.97875 42.43351 47.89433 55.45476 52.48174 54.29956 52.66683 52.68782
 [9] 50.40901 44.79636
> round(rnorm(10,50,5),2)
 [1] 46.24 49.43 45.51 48.41 47.55 47.49 57.22 53.29 47.01 55.36
>
```

Obr. 4.9.16 Generovanie náhodných čísel normálnym rozdelením

Rozdelenia pravdepodobností

Jazyk R obsahuje vstavané funkcie pre prácu s rozdeleniami pravdepodobností. Pre prácu s normálnym rozdelením používame príkaz `pnorm()`.Táto má tri parametre, s ktorými budeme pracovať. Prvým parametrom je náhodná premenná x, pre ktorú hľadáme hodnoty rozdelenia. Druhý a tretí parameter sú stredná hodnota a smerodajná odchýlka rozdelenia. Prácu s normálnym rozdelením pravdepodobností si ukážeme na príklade 4.6.

Príklad 4.6

Ako príklad nám poslúži firma, v ktorej pracuje sto zamestnancov. Ich mzda sa pohybuje v intervale od 200 do 1000 EUR. Výšky miezd majú normálne rozdelenie so strednou hodnotou 550 EUR a smerodajnou odchýlkou 175 EUR.

Úloha:

Aká je pravdepodobnosť, že vybraný zamestnanec bude mať mzdu z intervalu 500 až 750 EUR?

Riešenie:

V jazyku R výpočet vykonáme pomocou príkazu `pnorm()`

```
> pnorm(750,550,175)-pnorm(500,550,175)
[1] 0.4859026
>
```

Obr. 4.9.17 Príkaz na výpočet hodnôt pravdepodobností normálneho rozdelenia

Z výpočtu aj z výstupu na obr. 4.9.4 vyplýva, že pravdepodobnosť výberu zamestnanca, ktorý má mzdu z intervalu 500 až 750 EUR, je 48,49 %.

Testy dobrej zhody

Pomocou testov dobrej zhody môžeme zistiť, akým rozdelením pravdepodobností sa riadia naše dáta. Ukážeme si funkciu `fitdistr` z balíka **MASS**, ktorá nastavením parametrov dokáže porovnať dáta s vybranými rozdeleniami. Balík je treba doinštalovať.

Ďalej budeme prezentovať vlastnú funkciu, ktorá bude využívať popisné charakteristiky, ktoré sme o našom súbore zatiaľ zistili a na ich základe využitím Pearsonovho χ^2 testu dobrej zhody porovnáme náš súbor s normálnym rozdelením.

Príklad 4.7

Máme tabuľku **Zamestnanci** z Prílohy č. 2. Budeme pracovať s údajmi zo stĺpca mzda k 31.1.2014.

Úlohy:

a) Zistite, či údaje v stĺpci mzda v tabuľke **Zamestnanci** majú normálne rozdelenie. Riešenie funkciou `fitdistr()`.

b) Zistite, či údaje v stĺpci mzda v tabuľke **Zamestnanci** majú normálne rozdelenie. Riešenie vlastnou funkciou.

Riešenie:

a) Využitie funkcie fitdistr

Používanie tejto funkcie je pomerne jednoduché. Má totiž len dva parametre, z toho prvým je vektor dát, ktoré ideme porovnávať. Do druhého argumentu len zadáme rozdelenie, s ktorým chceme naše dáta porovnávať. Pre náš príklad potrebujeme porovnanie s normálnych rozdelením. Do druhého argumentu potom nastavíme hodnotu "normal". R nám vo výstupe vráti zoznam. Ten si uložíme do objektu.

```
> porovnanie1<-fitdistr(zamestnanci$mzda_k_31.1.2014,"normal")
```

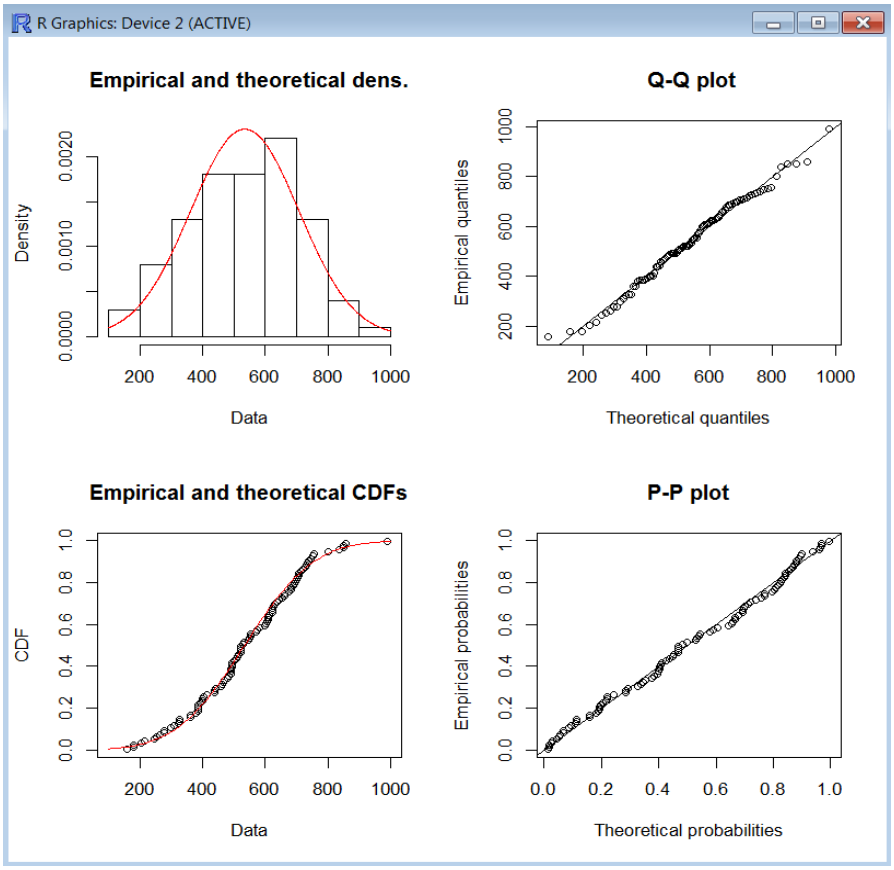
Obr. 4.9.18 Vytvorenie objektu s výsledkami porovnania

Výsledkom porovnávania je pre normálne rozdelenie stredná hodnota μ a smerodajná odchýlka σ. Výsledok porovnania vidíme na obrázku 4.9.6.

```
> porovnanie1
      mean          sd
  534.17210   173.12295
( 17.31229) ( 12.24164)
```

Obr. 4.9.19 Výsledok porovnania

Zostrojíme si grafy funkcií, ktoré nám opíšu naše výsledky. Využijeme pri tom príkaz `plotdist()` z balíka **fitdistrplus**. Balík **fitdistrplus** je treba doinštalovať. Do parametrov príkazu zadáme vstupné dáta, typ rozdelenia a nakoniec zobrazíme funkciu pomocou výsledkov predchádzajúceho porovnania.



Obr. 4.9.20 Grafický výstup príkazu plotdistr

Na výstupe vidíme histogram s dátami a vykreslenou funkciou normálneho rozdelenia. Napravo od neho je graf zobrazujúci skutočné hodnoty na osi *y* a teoretické hodnoty na osi *x*. Vľavo dole sa nachádza zobrazená distribučná funkcia normálneho rozdelenia a na nej zobrazené naše dáta. Posledný graf zobrazuje pravdepodobnosti výskytu hodnôt.

b) Využitie vlastnej funkcie

Napíšeme vlastnú funkciu, ktorá nám Pearsonovým χ^2 testom dobrej zhody porovná náhodné rozdelenie našich dát s Normálnym rozdelením a aj vyhodnotí výsledok testu. Funkciu sme nazvali chi_test a má dva parametre. Do prvého vkladáme vektor pozorovaných dát a druhý je hladina významnosti α.

```
> chi_test(zamestnanci$mzda_k_31.1.2014,0.05)
H0 : dáta majú normálne rozdelenie
H1: dáta nemajú normálne rozdelenie
chi^2 = 3.755707
p-value = 0.7096955
p-value je väčšia ako 5%, teda nemôžeme zamietnúť hypotézu H0,
 že dáta majú normálne rozdelenie.
>
```

Obr. 4.9.21 Priebeh funkcie chi-test

Zdrojový kód funkcie je súčasťou práce a nachádza sa v Prílohe č. 3.

4.10 Využitie jazyka R pri aktuárskych analýzach

Jazyk R obsahuje mnohé balíčky pre sprístupnenie alebo zjednodušenie aktuárskych výpočtov. Avšak mnoho akuárskych problematík sa dá v jazyku R riešiť aj bez nutnosti inštalácie doplnkových balíčkov s využitím iba základných funkcií a grafov, ktoré sú obsiahnuté v základnom inštalačnom balíku.

Využitím jazyka R sa môžeme venovať problematikám z oblastí aktuárstva, ako sú napríklad:

- individuálny model rizika,
- kolektívny model rizika,
- teória krachu,
- kalkulácia poistného,
- bonus-malus systém,
- analýza špecifických rizík,
- teória kredibility
- zovšeobecnené lineárne modely,
- kalkulácia technických rezerv v neživotnom poistení.

V nasledujúcom príklade budeme riešiť problematiku analýzy portfólia akcií v jazyku R. Pri riešení príkladu si vystačíme so základnými funkciami jazyka R. Použijeme funkcie, ktoré pracujú s rozdeleniami pravdepodobností a zároveň využijeme aj grafické nástroje pre vizuálne spracovanie príkladu.

Analýza portfólia akcií v jazyku R

Príklad 4.7

Predpokladajme, že máme záujem kúpiť portfólio skladajúce sa z akcií jednej spoločnosti zaoberajúcej sa výrobou pneumatík a druhej zaoberajúcej sa výrobou liehovín. Modelové údaje (obr. 4.10.1) pozostávajú z výberu týždenných cien akcií.[14] Cieľom analýzy je predvídať budúce správanie jednotlivých portfólií.

Úlohy:

a) Graficky analyzujte vývoj cien akcií a posúďte ich normalitu.

b) Analyzujte pomocou testov dobrej zhody (normality), či sa ceny akcií riadia normálnym rozdelením.

Riešenie:

a) Grafická analýza

Aby sme si predviedli ich vývoj v čase, vytvoríme si jednoduché grafy príkazom `plot()`.

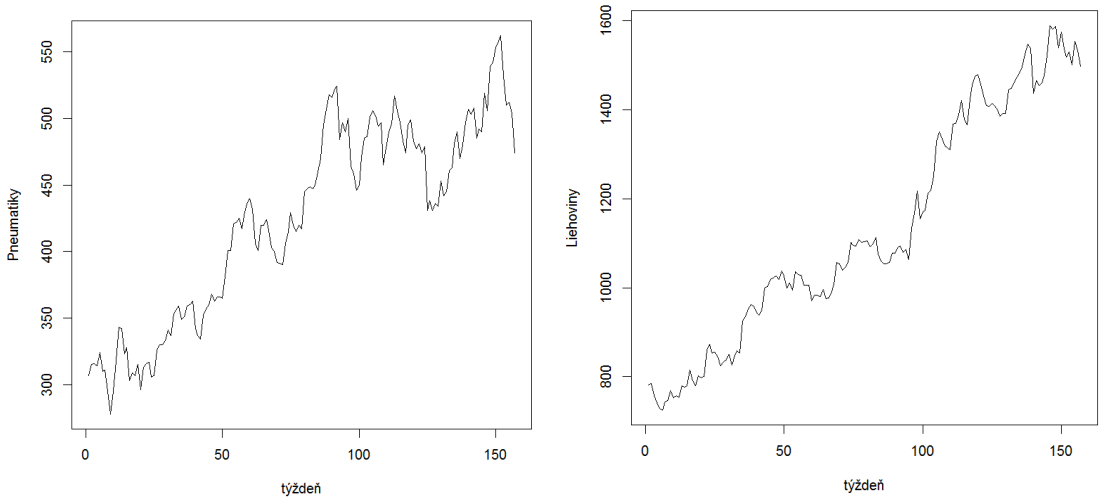
```
> Pneumatiky
  [1] 307 315 316 314 324 310 311 295 278 295 318 343 342 323 328 303 309 307
 [19] 315 296 313 316 317 306 307 326 330 330 333 341 337 353 356 359 349 351
 [37] 359 360 363 342 337 334 352 357 360 368 363 366 366 365 381 401 401 421
 [55] 422 425 417 427 436 440 432 406 401 420 420 424 416 403 400 392 391 390
 [73] 406 415 429 420 415 420 417 445 447 449 447 450 460 470 495 507 518 516
 [91] 522 524 484 497 490 500 464 458 446 450 471 485 486 501 506 502 494 497
[109] 465 478 490 496 517 506 497 483 474 495 499 483 477 481 474 479 431 438
[127] 431 436 434 453 442 445 461 463 481 490 470 480 497 507 503 508 485 492
[145] 490 519 506 539 542 553 558 562 532 510 512 504 474
> Liehoviny
  [1]  781  784  757  741  728  726  743  746  768  752  758  754  779  777  780
 [16]  815  791  779  802  797  800  860  873  854  855  846  824  833  838  851
 [31]  827  847  859  853  926  935  952  962  958  943  938  949 1000 1003 1018
 [46] 1022 1026 1019 1037 1026  999 1011  994 1036 1030 1028 1005 1006 1005  970
 [61]  983  984  980  996  975  976  987 1008 1057 1054 1040 1045 1057 1101 1096
 [76] 1094 1108 1102 1104 1105 1092 1098 1113 1076 1060 1054 1054 1057 1078 1077
 [91] 1091 1093 1079 1086 1064 1134 1167 1217 1155 1171 1174 1213 1218 1250 1329
[106] 1350 1334 1318 1315 1310 1368 1370 1389 1420 1377 1366 1424 1455 1475 1478
[121] 1458 1430 1409 1407 1414 1409 1402 1386 1392 1391 1445 1448 1462 1474 1482
[136] 1495 1525 1547 1538 1436 1465 1454 1460 1476 1521 1588 1581 1587 1539 1574
[151] 1537 1518 1530 1500 1554 1532 1498
>
```

Obr. 4.10.1 Vývoj cien akcií spoločností

[14] Kaas, R. Goodvaerts, M. Dhaene, J. Denuit, M.: *Modern Actuarial Risk Theory*. 2008. s332.

```
> plot(Pneumatiky,xlab="týždeň", type="l")
> plot(Liehoviny,xlab="týždeň",type="l")
>
```

Obr. 4.10.2 Príkaz na tvorbu grafu vývoja cien akcií v čase



Obr. 4.10.3 Graf vývoja cien akcií v čase

Ak $S_t, t = 1,2,\ldots.$, sú ceny akcií, môžeme sledovať jednoduché, celkové alebo priemerné výnosy $S_{t+1} - S_t/S(t)$. My budeme ďalej sledovať logaritmy výnosov (logarithms of the returns, log-returns). Táto úprava je dôležitá pre ich dve vlastnosti – nezávislosť a normalita. Výpočet logaritmov výnosov v jazyku R zapíšeme nasledovne:

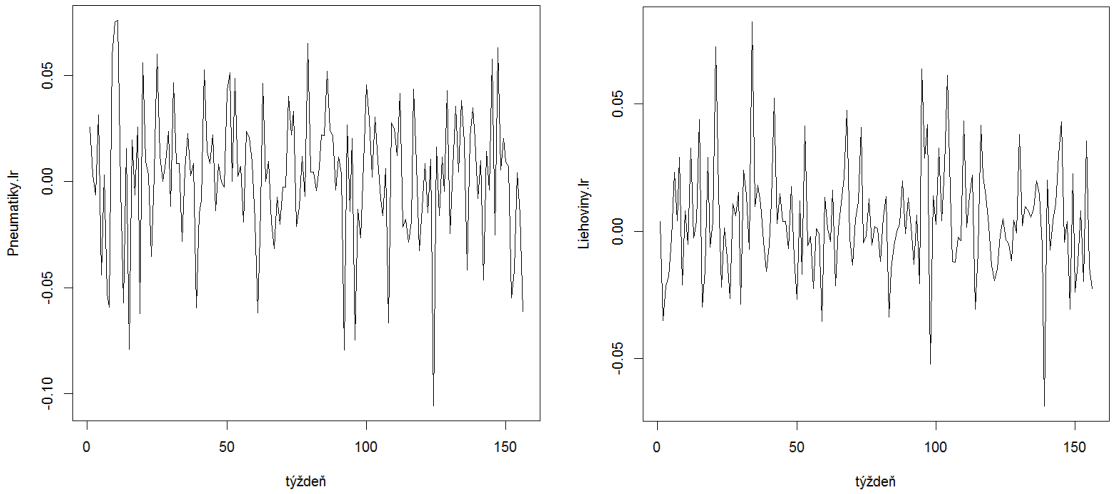
```
> Pneumatiky.lr<-log(Pneumatiky[-1]/Pneumatiky[-length(Pneumatiky)])
> Liehoviny.lr<-log(Liehoviny[-1]/Liehoviny[-length(Liehoviny)])
>
```

Obr. 4.10.4 Príkaz na výpočet logaritmov výnosov akcií

Vytvoríme grafy logaritmov výnosov z akcií.

```
> plot(Pneumatiky.lr,xlab="týždeň",type="l")
> plot(Liehoviny.lr,xlab="týždeň",type="l")
>
```

Obr. 4.10.5 Príkaz na tvorbu grafov logaritmov výnosov

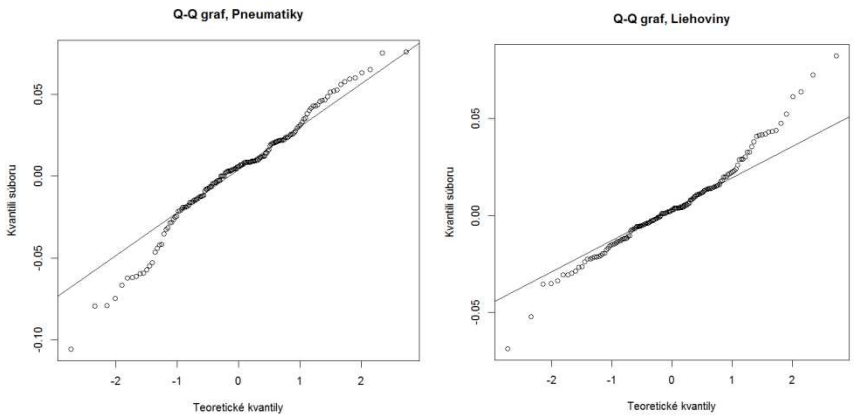


Obr. 4.10.6 Graf vývoja výnosov portfólia

Aby sme zistili, či logaritmy výnosov majú normálne rozdelenie, môžeme ich otestovať pomocou grafov *Q-Q*. V grafe typu *Q-Q normal plot* sú zobrazené kvantity analyzovaného súboru oproti teoretickým kvantilom normálneho rozdelenia. Ak sú hodnoty na grafe *Q-Q* tesne alebo v blízkosti priamky (viď obr. 4.10.8), súbor môže mať normálne rozdelenie. Graf typu *Q-Q* vytvoríme nasledovne:

```
> qqnorm(Pneumatiky.lr, main="Q-Q graf, Pneumatiky",xlab="Teoretické kvantily",
+ ylab="Kvantili súboru")
> qqline(Pneumatiky.lr)
> qqnorm(Liehoviny.lr, main="Q-Q graf, Liehoviny",xlab="Teoretické kvantily",
+ ylab="Kvantili súboru")
> qqline(Liehoviny.lr)
>
```

Obr. 4.10.7 Príkazy na tvorbu *Q-Q* normal plot



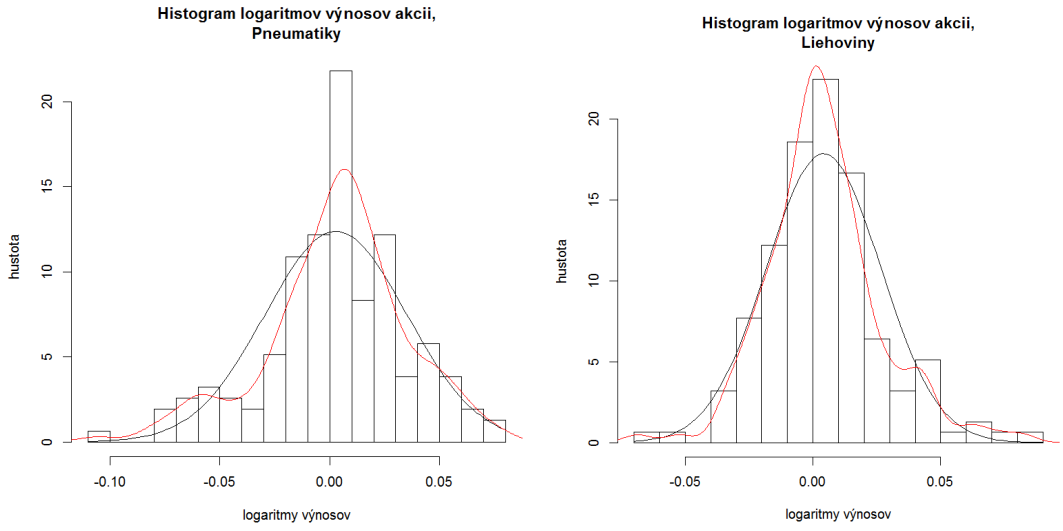
Obr. 4.10.8 *Q-Q* normal plot

V prípade analyzovaných údajov nemôžeme jednoznačne prijať tvrdenie, že ceny akcií sa riadia normálnym rozdelením ani v jednom prípade (v prípade akcií liehovín sa na konci hodnoty viditeľne vzďaľujú od priamky). Pri analýze pomocou grafu *Q-Q* má výrazný vplyv subjektivita posudzovateľa.

Súbory otestujeme aj ďalším spôsobom, a to pomocou histogramu. V histograme ďalej zobrazíme krivku hustoty normálneho rozdelenia (so strednou hodnotou a štandardnou odchýlkou analyzovaných údajov) aj skutočnú hustotu pravdepodobnosti analyzovaných údajov (na grafe je zvýraznená červenou farbou).

```
> hist(Pneumatiky.lr,prob=T, breaks=21,main="Histogram logaritmov výnosov akcii,
+ Pneumatiky",ylab="hustota",xlab="logaritmy výnosov")
> curve(dnorm(x,mean=mean(Pneumatiky.lr),sd=sd(Pneumatiky.lr)),add=T)
> lines(density(Pneumatiky.lr),col="red")
> hist(Liehoviny.lr,prob=T, breaks=21, main="Histogram logaritmov výnosov akcii,
+ Liehoviny",ylab="hustota",xlab="logaritmy výnosov")
> curve(dnorm(x,mean=mean(Liehoviny.lr),sd=sd(Liehoviny.lr)),add=T)
> lines(density(Liehoviny.lr),col="red")
>
```

Obr. 4.10.9 Príkazy na tvorbu histogramu logaritmov výnosov akcií



Obr. 4.10.10 Histogram

b) Analýza pomocou testov dobrej zhody

Na otestovanie toho, či sa logaritmy výnosov z akcii riadia normálnym rozdelením, využijeme *Jarque-Beraov test*. *Jarque-Beraov test* je založený na testovaní šikmosti a

špicatosti. Vychádza sa zo skutočnosti, že tretí centrálny moment normálneho rozdelenia sa rovná 0 a štvrtý centrálny moment sa rovná 3.

JB testovacia charakteristika má tvar:

$$JB = \frac{n}{6}\left(S^2 + \frac{K^2}{4}\right)$$

kde

n je počet pozorovaní (identický s počtom stupňov voľnosti),

S je miera šikmosti (skewness) pozorovaných údajov,

K je miera špicatosti (kurtosis) pozorovaných údajov. [15]

Miery šikmosti a špicatosti pozorovaných údajov vypočítame podľa vzťahu:

$$S = \frac{\widehat{\mu_3}}{\widehat{\sigma^3}} = \frac{\frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})^3}{\left(\frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})^2\right)^{3/2}}$$

$$K = \frac{\widehat{\mu_3}}{\widehat{\sigma^4}} - 3 = \frac{\frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})^4}{\left(\frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})^2\right)^2} - 3$$

kde

$\bar{X} = \frac{1}{n}\sum X_i$ je priemer pozorovaných údajov,

$\widehat{\sigma^2} = \frac{1}{n}\sum(X_i - \bar{X})^2$ je rozptyl pozorovaných údajov. [16]

Jarque-Beraov test má χ^2 rozdelenie. Kritickú hodnotu pre hladinu významnosti α = 0,05 predstavuje hodnota 6,0. Výpočet *JB* testovacej charakteristiky uskutočníme príkazmi, ktoré uvádza obr. 4.10.11.

Testovacia charakteristika *JB* pre logaritmy výnosov z akcii pre spoločnosť zaoberajúcu sa výrobou pneumatík je 9,305 (obr. 4.10.11), čo je hodnota vyššia ako kritická hodnota. Z toho vyplýva, že môžeme zamietnuť hypotézu, že logaritmy výnosov z akcií spoločnosti zaoberajúcej sa výrobou pneumatík majú normálne rozdelenie.

[15] Kaas, R. Goodvaerts, M. Dhaene, J. Denuit, M.: *Modern Actuarial Risk Theory*. 2008. s334
[16] Kaas, R. Goodvaerts, M. Dhaene, J. Denuit, M.: *Modern Actuarial Risk Theory*. 2008. s335

Testovacia charakteristika *JB* pre spoločnosť zameranú na výrobu liehovín vyšla 18,89, teda tiež vyššia ako kritická hodnota. Z toho vyplýva, že rovnako zamietneme hypotézu, že logaritmy výnosov z akcií tejto spoločnosti majú normálne rozdelenie.

```
> x<-Pneumatiky.lr-mean(Pneumatiky.lr)
> m2<-mean(x^2); m3<-mean(x^3); m4<-mean(x^4)
> s2<-m3^2/m2^3; K<-m4/m2^2 -3
> JB<-length(x)/6 * (s2 + K^2/4)
> p.value<- 1-pchisq(JB, df=2)
> JB; p.value
[1] 9.305275
[1] 0.009536414
> x<-Liehoviny.lr-mean(Liehoviny.lr)
> m2<-mean(x^2); m3<-mean(x^3); m4<-mean(x^4)
> s2<-m3^2/m2^3; K<-m4/m2^2 -3
> JB<-length(x)/6 * (s2 + K^2/4)
> p.value<- 1-pchisq(JB, df=2)
> JB; p.value
[1] 18.88978
[1] 7.90928e-05
>
```

Obr. 4.10.11 Príkazy na výpočet *JB* testovacej charakteristiky

Záver

Za výsledok bakalárskej práce môžeme považovať vytvorenie používateľskej príručky pre prácu s programovacím jazykom R. Využitím tohto manuálu by mal byť používateľ schopný zvládnuť základy pre prácu s jazykom R, ako aj jednoduchú aplikáciu jazyka R do praxe.

Používateľ začína od úplných základov, a to nainštalovaním prostredia programovacieho jazyka R do svojho operačného systému. Manuál popisuje a znázorňuje základnú orientáciu v prostredí jazyka R. Pozornosť je venovaná aj syntaxu písania najjednoduchších príkazov (funkcií) v prostredí programovacieho jazyka R. Nemej dôležité je vysvetlenie postupu inštalácie doplnkových balíčkov pre jazyk R, ktoré sú zamerané na riešenie vybraných problematík.

Ďalšia časť je venovaná dátovým typom, s ktorými môžeme na úrovni jazyka R pracovať. Vysvetľuje význam objektov a ich rozčlenenie podľa možnosti obsiahnutých dát. Následne prechádzame k písaniu príkazov pre tvorbu a operácie s dátovými objektmi. Je znázornená práca s každou spomenutou dátovou štruktúrou, s ktorou sa môžeme v jazyku R stretnúť a taktiež vysvetlený postup ako sa z prostredia programovacieho jazyka R pripojiť, alebo do jazyka R nahrať databázy údajov z iných zdrojov.

Dôležitým nástrojom je tvorba grafov, ktoré jazyk R zobrazuje do grafického okna priamo v pracovnom prostredí. Jazyk R poskytuje kvalitné grafické výstupy 2D a 3D grafov pre široké použitie.

Posledná časť príručky je zameraná na využitie všetkých predchádzajúcich poznatkov pri štatistickej analýze údajov, ktorú môže využiť aktuár v praxi.

Zoznam použitej literatúry

[1] Black, Kelly. 2005. *R TUTORIAL* In cyclismo.org [online]. 2011. Dostupné na internete: <http://www.cyclismo.org/tutorial/R/>

[2] Burns, Patrick.: *S Poetry*. Patrick Burns. 1998. ISBN 9781471045523

[3] Henry. 2014. *Estimating the distribution from data* In Cross Validated [online]. 2014. Dostupné na internete: <http://stats.stackexchange.com/questions/25568/estimating- the-distribution from-data>

[4] Chi Yau. 2009. *R TUTORIAL* In r-tutor.com [online]. 2014. Dostupné na internete: <http://www.r-tutor.com/elementary-statistics/probability-distributions/student-t-distribution>

[5] Illinois state university. 2004. *MAT 356 R TUTORIAL* In math.illinoisstate.edu [online]. 2004. Dostupné na internete: <http://math.illinoisstate.edu/dhkim/rstuff/rtutor.html>

[6] Kaas, Rob; Goodvaerts, Marc; Dhaene Jan; Denuit, Michael. 2008. *Modern Actuarial Risk Theory*. 2.vid. Berlin Heidelberg Dordrecht London New York: Springer.2008. s381. ISBN 978-3-642-03407-7.

[7] Kabacoff, Robert I., Ph.D.. 2014. *Quick-R* In statmethods.net [online] 2014. Dostupné na internete: <http://www.statmethods.net/index.html>

[8] Kalina, Martin; Bacigál, Tomáš; Schiesslová, Anna. 2010. *Základy pravdepodobnosti a matematickej štatistiky* In Slovenská technická univerzita v Bratislave [online]. 2010. 215s. ISBN 978-80-227-3273-4. Dostupné na internete: <http://cran.r-project.org/doc/contrib/FundamentalsOfProbAndMStatistics-Slovak.pdf>

[9] Kaspříková, Nikola. 2010 *Návod k R* In data.tulipany.cz [online]. 2010. Dostupné na internete: <http://data.tulipany.cz/navodR.htm>

[10] Konečná, Kateřina. 2010. *Výuka jazyka R* : bakalárska práca.In is.muni.cz [online] Brno : Masarykova univerzita: Přírodovědecká fakulta, 2010. Dostupné na internete: <http://is.muni.cz/th/270073/prif_b/Bakalarska_prace.pdf>

[11] McCown, Frank. 2006. *Producing Simple Graphs with R* In harding.edu [online]. 2007. Dostupné na internete: <http://www.harding.edu/fmccown/r>

[12] Pancikova. 2011. *R manuál* IN fria.fri.uniza.sk [online] 2014. Dostupné na internete: <http://fria.fri.uniza.sk/~kmame/drupal/subory/pancikova/Rmanual/?page_id=23>

[13] Shah, Ajay. 2005. *R by example* In mayin.org [online] 2005. Dostupné na internete: <http://www.mayin.org/ajayshah/KB/R/index.html>

[14] Slawa Rokicki. 2012. *Basics of Histograms* In R-bloggers [online]. 2012. Dostupné na internete: <http://www.r-bloggers.com/basics-of-histograms/>

Prílohy

Príloha č. 1

Tabuľka **Tab**

```
> Tab
                     počet_obyvateIov rozloha skratka
Bratislavský kraj              606537  2052.7      BA
Banskobystrický kraj           657119  9454.8      BB
Košický kraj                   792991  6755.0      KE
Nitriansky kraj                706375  6343.4      NR
Prešovský kraj                 807011  8973.7      PO
Trenčiansky kraj               599859  4501.9      TN
Trnavský kraj                  561525  4146.7      TT
Žilinský kraj                  697502  6808.8      ZA
>
```

Tabuľka **Tab1**

```
> Tab1
                     počet_obyvateIov rozloha skratka počet_obyvateIov2
Bratislavský kraj              606537  2052.7      BA          512523.8
Banskobystrický kraj           657119  9454.8      BB          555265.6
Košický kraj                   792991  6755.0      KE          670077.4
Nitriansky kraj                706375  6343.4      NR          596886.9
Prešovský kraj                 807011  8973.7      PO          681924.3
Trenčiansky kraj               599859  4501.9      TN          506880.9
Trnavský kraj                  561525  4146.7      TT          474488.6
Žilinský kraj                  697502  6808.8      ZA          589389.2
```

Príloha č. 2

Tabuľka **Zamestnanci**

```
> zamestnanci
    mzda_k_31.1.2014 odpracované_dni_január
1             698.97                     92
2             479.54                     63
3             384.86                     51
4             722.40                     95
5             752.21                     99
6             468.73                     62
7             623.90                     82
8             324.98                     43
9             707.64                     93
10            461.59                     61
11            361.17                     47
12            326.81                     43
13            754.19                     99
14            655.75                     86
15            488.99                     64
16            157.68                     21
17            609.14                     80
18            492.68                     65
19            321.51                     42
20            533.07                     70
21            598.22                     79
22            491.40                     64
23            735.93                     97
24            520.78                     68
25            277.04                     36
26            308.61                     41
27            546.47                     72
28            507.31                     67
29            799.93                    105
30            836.38                    110
31            298.31                     39
32            683.14                     90
33            849.25                    111
34            360.66                     47
35            527.27                     69
36            662.31                     87
37            857.11                    112
38            989.17                    130
39            605.53                     79
40            645.31                     85
41            482.90                     63
42            703.78                     92
43            378.13                     50
44            554.04                     73
45            549.17                     72
46            213.11                     28
47            501.42                     66
48            621.75                     82
49            490.74                     64
50            511.55                     67
    mzda_k_31.1.2014 odpracované_dni_január
51            392.82                     52
52            694.31                     91
53            261.02                     34
54            201.92                     27
55            707.06                     93
56            573.96                     75
57            439.68                     58
58            436.18                     57
59            621.30                     82
60            275.18                     36
61            739.00                     97
62            569.48                     75
63            727.66                     96
64            252.17                     33
65            625.91                     82
66            683.47                     90
67            400.30                     53
68            465.67                     61
69            602.57                     79
70            552.60                     73
71            635.26                     83
72            178.07                     23
73            384.17                     50
74            745.45                     98
75            715.37                     94
76            520.89                     68
77            731.92                     96
78            519.99                     68
79            398.45                     52
80            386.62                     51
81            851.73                    112
82            243.49                     32
83            493.08                     65
84            177.53                     23
85            519.80                     68
86            414.26                     54
87            608.58                     80
88            435.94                     57
89            514.14                     67
90            401.54                     53
91            494.53                     65
92            613.68                     81
93            581.26                     76
94            690.53                     91
95            661.72                     87
96            383.38                     50
97            697.29                     92
98            630.42                     83
99            676.57                     89
100           456.76                     60
```

Príloha č. 3

Zdrojový kód funkcie **chi_test**

```
chi_test<- function(x,y){
cat("H0 : dáta majú normálne rozdelenie \nH1: dáta nemajú normálne rozdelenie \n")
alfa<-y
stredna_hodnota<-median(x)
odchylka<-sd(x)
hist_info<-hist(x)
skutoc_poc<-hist_info$counts
ohranic<-hist_info$breaks
m <- length(ohranic) - 1; n <- length(x)
poc_ohranic<-length(ohranic)-1
poc_pozorovani<-length(x)
ohranic[1]<- -Inf
ohranic[m+1]<- Inf
teor_poc<- diff(pnorm(ohranic,stredna_hodnota,odchylka))*poc_pozorovani
chi_stat<-sum((skutoc_poc-teor_poc)^2/teor_poc)
chi_kvantil<-qchisq(alfa,m-2-1)
cat("chi^2 =",chi_stat)
p_hodnota<-1-pchisq(chi_stat,df=m-2-1)
cat("\np-value =",p_hodnota)
if(p_hodnota>0.05)cat(" \np-value je väčšia ako 5%, teda nemôžeme zamietnuť hypotézu H0,\n že dáta majú normálne rozdelenie. \n")
else cat("\np-value je nižšia ako 5%, teda zamietame hypotézu H0, \na prijímame hypotézu h1, že dáta nemajú normálne rozdelenie. \n")
}
```