7th International Conference
APLIMAT 2008
Faculty of Mechanical Engineering - Slovak University of Technology in Bratislava

Section: Open Source Software in Research and Education

SPRACOVANIE DÁT VO FYZIKE OPEN SOURCE PROGRAMOM R

ŠEVČOVIČ, Ladislav, (SK)

Abstrakt. Open source program **R** je komplexný systém na manipuláciu s údajmi, ich spracovanie, analýzu a následné grafické zobrazenie. **R** je implementáciou jazyka **S**. Výsledky štatistickej analýzy sú v grafickej podobe zobrazené na displeji, niektoré medzivýsledky sa môžu ukladať alebo zapisovať do súborov. Pre začiatočníkov sa môže zdať, že program **R** je veľmi zložitý, nie je to však celkom pravda. Hlavná prednosť programu je v jeho flexibilite. Bežné štatistické programy výsledky analýzy zobrazujú priebežne na obrazovke, program **R** ich však uchováva ako „objekty“, ktoré sa nezobrazujú, ale sú počas práce vždy dostupné. Flexibilitu systému **R** predstavíme ukážkami vybraných príkladov použitia štatistických modelov na analýzu dát z fyzikálnych experimentov. V príspevku opíšeme analýzu, fitovanie dát a prezentáciu výsledkov analýzy typografickým systémom LaTeX.

Kľúčové slová. Regresia, spracovanie dát, open source, R.

DATA PROCESSING IN PHYSICS WITH OPEN SOURCE PROGRAM R

Abstract. **R** is an open-source statistical environments modelled under the **S** language. **R** has many function for statistical analyses and graphics. The results from analyses are displayed on the screen, some intermediate results can be saved, written in a file, or used in subsequent analyses. **R** could seem too complex for a non-specialist. This may not be true actually, a main feature of **R** is its flexibility. Classical software displays immediately the results of an analysis, **R** stores these results in an “object”, so that an analysis can be done with no result displayed. We will see some examples of statistical models for analysing the data from experiment illustrating the flexibility of **R**. This contribution covers several topics in data analysis, including curve fitting and presentation of the results of analysis by LaTeX typographical system.

Key words and phrases. Regression, data processing, open source, R.

Mathematics Subject Classification. Primary 62-07, 62P35; Secondary 65Z05.

1 Úvod

Pri spracovaní experimentálnych dát v prírodovednej a technickej praxi obvykle využívame poznatky z prírodných vied, informatiky a taktiež matematickej štatistiky. Dominantnú časť analýzy dát tvoria metódy, pomocou ktorých získavame relevantné informácie z pozorovaní a experimentov. Rozšírenie výkonných osobných počítačov umožnilo zaviesť iteratívnosť a decentralizáciu pri spracovaní experimentálnych údajov a interpretácii výsledkov ich analýzy. Táto situácia však kladie väčšie nároky na užívateľa, ktorý má možnosť výberu z veľkej ponuky počítačovo orientovaného softvéru na štatistické spracovanie dát. Experimentátor je nútený k hlbšej analýze, čo obyčajne vedie aj k radikálnej zmene prístupu na rutinne vykonávanú pracovnú činnosť.

Na trhu existuje veľké množstvo komplexných programov a programových systémov určených na spomenuté účely, či už sú to komerčné (MATLAB, S-Plus, SPSS, Statistica, OriginLab a pod.) alebo open source softvérové prostriedky (**R**, Octave, PyLab, Qtiplot, Labplot, eXtrema, Xmgrace a pod.). Pri ich aplikácii si však musíme uvedomiť, že počítač neprináša nové informácie a tak v konečnom dôsledku sa etapa vyhodnocovania experimentov a pozorovaní stáva najslabším článkom spracovania dát.

Program **R** bol vytvorený Rossom Ihakom a Robertom Gentlemanom v roku 1995 implementáciou jazyka **S**, ktorý bol vyvinutý v Bellovych laboratóriách Rickom Beckerom, Johnom Chambersom a Allanom Willksom v polovici sedemdesiatych rokov, a ktorý slúži ako základ systému S-Plus [10].

Základná distribúcia systému **R** je udržiavaná malou skupinou štatistikov známou pod menom **R** Development Core Team. Obrovské množstvo dodatočnej funkcionality je implementované formou balíkov, vytvorených veľkou skupinou dobrovoľníkov.

R je v podstate integrovaná súprava softvérových nástrojov umožňujúca manipuláciu, numerické spracovanie a grafické zobrazenie dát. Okrem iného poskytuje:

- možnosti efektívne manipulovať s dátami a ukladať ich;
- súbor operátorov na prácu s poliami, napríklad s maticami;
- veľkú prepojenú kolekciu prostriedkov na analýzu dát;
- grafické možnosti na dátovú analýzu a zobrazenie výsledkov priamo na počítači alebo po vytlačení;
- kvalitne vyvinutý, jednoduchý a efektívny programovací jazyk (nazývaný „**S**“) obsahujúci rozhodovacie operátory, cykly, rekurzívne funkcie definované používateľmi, vstupné a výstupné procedúry. (V skutočnosti je väčšina funkcií napísaných v jazyku **S**.)

Označenie „prostredie“ má charakterizovať systém **R** skôr ako plne prepojený systém a nie ako postupne dopĺňanú zbierku špeciálnych nepružných nástrojov, ako sa to často stáva v prípade iných programov určených na analýzu dát.

R je výborný prostriedok na rozvoj nových metód interaktívnej analýzy údajov. Jeho vývoj bol veľmi dynamický a je rozšírený o veľké množstvo balíkov. V prostredí **R** je

možné riešiť aj mnoho klasických i moderných štatistických úloh. Niektoré sú priamo súčasťou základného prostredia **R**, viaceré sú k dispozícii ako balíky. Spolu s programom **R** je dodaných okolo 25 balíkov (nazývaných „štandardné“ alebo „odporúčané“ balíky), mnohé ďalšie sa dajú získať z archívu CRAN (napr. z `http://CRAN.R-project.org` alebo inde).

Pri použití programu **R** sa zobrazí výzva (anglicky prompt), keď **R** očakáva na vstupe príkazy. Prednastavená výzva má tvar ‘`>`’, môžeme ju zmeniť príkazom `options`, napríklad na ‘`R>␣`’ príkazom `options(prompt="R> ")`.

V OS GNU/Linux (ak predpokladáme „prompt“ ‘`$`’) môžete začať pracovať nasledujúcim spôsobom:

1. Vytvorte adresár, napríklad `work`, do ktorého budete ukladať súbory, ktoré použijete na riešenie úlohy programom **R**. Tento bude pracovným adresárom pre túto úlohu, môžete sa do neho dostať príkazom `cd` – *change directory*:

   ```
   $ mkdir work
   $ cd work
   ```

2. Spusťte program **R** príkazom

   ```
   $ R
   ```

3. Teraz už môžete zadávať príkazy programu **R**. Program **R** ukončíme príkazom

   ```
   > q()
   ```

   V tomto momente sa program **R** opýta, či si prajete uložiť údaje. Uložené údaje budú potom k dispozícii pri ďalšom spustení programu **R**.

Pri použití programu **R** pod OS Windows je postup principiálne rovnaký. Po vytvorení pracovného adresára môžete tento nastaviť ako aktuálny pri spustení v položke Štart odkazu na program **R** a program spustiť dvojklikom na ikonku. Adresár môžete zmeniť aj prostredníctvom položky File/Change dir ... po spustení programu **R**.

2 Vytvorenie a načítanie údajov

V programe **R** sa štruktúra dát, premenné ako aj funkcie uchovávajú v podobe objektov. Počas práce s programom sa vytvorené objekty dajú zobraziť príkazom `ls()` alebo `objects()`. Pomocou rôznych funkcií môžeme získať informácie o type a móde týchto objektov, napr. funkciou `mode(x)` zistíme, že mód objektu `x` z nasledujúcej časti je typu `"numeric"` a funkciou `storage.mode(x)` zase, že mód uloženia objektu `x` je typu `"double"`. Viac sa dočítate v príručkách k programu **R** [1, 22]. V nasledujúcich častiach v krátkosti uvedieme základné operácie, ktorými môžeme vytvoriť dátový vektor pomocou klávesnice počítača a ako dáta načítame do prostredia programu **R** z externého súboru.

2.1 Vytvorenie dátového vektora, zadávanie dát klavesnicou

2.1.1 Vektory a priradenie

R pracuje s pomenovanými dátovými štruktúrami. Najjednoduchšou štruktúrou je číselný vektor, ktorý predstavuje usporiadanú postupnosť čísel. Napríklad, na zadanie vektora s názvom `x`, pozostávajúceho z piatich čísel, konkrétne 10.4, 5.6, 3.1, 6.4 a 21.7, použite príkaz

```
> x <- c(10.4, 5.6, 3.1, 6.4, 21.7)
```

Toto je operátor priradenia pomocou funkcie `c()`, ktorá v tomto prípade môže obsahovať ľubovoľný počet *vektorových* argumentov. Jej hodnotou je vektor, ktorý vznikne spojením jej argumentov od začiatku do konca.[9]

Číslo, ktoré sa vyskytuje vo výraze sa chápe ako vektor dĺžky 1.

Poznamenajme, že operátor priradenia (‘`<-`’), pozostáva z dvoch znakov ‘`<`’ („menší ako“) a ‘`-`’ („spojovník“) nasledujúcich bezprostredne za sebou a „ukazuje“ na objekt, ktorému sa priraďuje hodnota výrazu. Vo väčšine prípadov je možné alternatívne použiť operátor ‘`=`’. Priradenie sa môže vykonať tiež použitím funkcie `assign()`. Rovnaký výsledok ako vyššie dosiahneme zadaním:

```
> assign("x", c(10.4, 5.6, 3.1, 6.4, 21.7))
```

Bežný operátor `<-` je možné chápať ako syntaktickú skratku príkazu `assign()`.

Priradenie je možné vykonať aj v opačnom smere použitím zrejmej zmeny v operátore priradenia, teda zadaním

```
> c(10.4, 5.6, 3.1, 6.4, 21.7) -> x
```

Ak zadáme len výraz (bez priradenia), hodnota sa vytlačí a stratí[10]. Teda po použití príkazu

```
> 1/x
```

bude na obrazovke vytlačených 5 prevrátených hodnôt jednotlivých zložiek vektora `x` (hodnota `x` sa, prirodzene, nezmení).

Ďalším priradením

```
> y <- c(x, 0, x)
```

bude vytvorený vektor `y` s 11-imi zložkami, pozostávajúci z dvoch kópií vektora `x`, medzi ktorými je uprostred umiestnená 0.

Základné aritmtické operátory sú bežne používané `+`, `-`, `*`, `/` a `^` sa používa na umocňovanie. Navyše sú k dispozícii všetky elementárne aritmetické funkcie: `log`, `exp`, `sin`, `cos`,

[9] Ak použijeme iný typ argumentov ako vektorový, napríklad argumenty typu zoznam, výsledok použitia funkcie `c()` bude úplne iný.

[10] V skutočnosti, pred použitím ďalšieho príkazu, je ešte stále k dispozícii ako hodnota `.Last.value`.

`tan`, `sqrt` atď., majú obvyklý význam. Funkcie `max` a `min` vyberú najväčšiu resp. najmenšiu zložku vektora. Funkcia `range` vráti vektor dĺžky 2, konkrétne `c(min(x), max(x))`. Funkcia `length(x)` určí počet zložiek vektora `x`, `sum(x)` vráti súčet zložiek vektora `x`, `prod(x)` vypočíta ich súčin.

K dispozícii sú aj dve štatistické funkcie – `mean(x)` vypočíta výberový priemer, ktorý sa rovná `sum(x)/length(x)` a `var(x)` dáva `sum((x-mean(x))^2)/(length(x)-1)`, čiže nevychýlený výberový rozptyl. Ak je argumentom funkcie `var()` matica typu $n \times m$, jej hodnotou bude výberová kovariančná matica typu $m \times m$, pričom riadky vstupnej matice sa chápu ako nezávislé výberové vektory rozmeru m.

Funkcia `sort(x)` vráti vektor rovnakej dĺžky ako vektor `x` so zložkami usporiadanými v narastajúcom poradí; avšak existujú aj flexibilnejšie prostriedky na usporadúvanie (napríklad `order()` alebo `sort.list()`). Všimnime si, že `max` a `min` vyberú najväčšiu a najmenšiu hodnotu svojich argumentov dokonca aj vtedy, ak je zadaných niekoľko vektorov. Paralelné funkcie maximum a minimum – `pmax` a `pmin` – vrátia vektor (jeho dĺžka je rovnaká, ako najväčšia dĺžka vstupného argumentu) obsahujúci na každej pozícii najväčší resp. najmenší prvok na odpovedajúcej pozícii vo všetkých vstupných vektoroch. Vo väčšine prípadov sa používateľ nemusí starať o to, či sú „čísla“ celé, reálne alebo dokonca komplexné. Vnútorne sa operácie vykonávajú v dvojnásobnej presnosti reálnych čísel, alebo v dvojnásobnej presnosti komplexných čísel, ak sú vstupné údaje komplexné.

2.1.2 Generovanie rovnomerných postupností

R má niekoľko prostriedkov na generovanie často používaných číselných postupností. Napríklad `1:30` je vektor `c(1, 2, ..., 29, 30)`. Operátor ‘`:`’ (dvojbodka) má vo výrazoch najvyššiu prioritu, teda, napríklad `2*1:15` je vektor `c(2, 4, ..., 28, 30)`. Zadajte `n <- 10` a porovnajte postupnosti `1:n-1` a `1:(n-1)`. Konštrukcia `30:1` sa dá použiť na vygenerovanie klesajúcej postupnosti.

Funkcia `seq()` je všeobecnejší prostriedok na vytváranie postupností. Má 5 argumentov, pričom pri jednotlivom použití môžu byť použité len niektoré z nich. Prvé dva argumenty, ak sú zadané, určujú začiatok a koniec postupnosti a v prípade, že sú zadané len tieto dva argumenty, výsledok bude rovnaký ako pri použití operátora ‘`:`’. Teda `seq(2,10)` je taký istý vektor ako `2:10`.

Parametre funkcie `seq()` a mnohých ďalších funkcií **R**, môžu byť zadané aj v tvare priradenia hodnôt vnútorným premenným, kedy nezáleží na poradí ich zadania. Prvé dva parametre môžu byť označené `from=value` a `to=value`: `seq(1,30)`, `seq(from=1, to=30)` a `seq(to=30, from=1)` definujú rovnaké vektory ako `1:30`. Nasledujúce dva parametre príkazu `seq()` môžu byť označené `by=value` resp. `length=value`, čím určíme veľkosť kroku resp. celkovú dĺžku postupnosti. Ak nie je daná žiadna z týchto hodnôt, predpokladá sa implicitne nastavená hodnota `by=1`.

Napríklad

```
> seq(-5, 5, by=.2) -> s3
```

priradí premennej `s3` vektor `c(-5.0, -4.8, -4.6, ..., 4.6, 4.8, 5.0)`. Podobne

```
> s4 <- seq(length=51, from=-5, by=.2)
```

priradí rovnaký vektor premennej `s4`.

Piaty parameter sa definuje zadaním `along=vector`, ktorý môže byť použitý len osamotene, pričom sa vytvorí postupnosť 1, 2, ..., `length(vector)` alebo prázdna postupnosť, ak je daný vektor prázdny.

Na zopakovanie zložiek nejakého objektu je možné rôznymi komplikovanými spôsobmi použiť funkciu `rep()`. Najjednoduchší tvar je

```
> s5 <- rep(x, times=5)
```

ktorý priradí 5 kópií vektora `x` od začiatku do konca premennej `s5`. Ďalšia užitočná verzia

```
> s6 <- rep(x, each=5)
```

zopakuje každú zložku vektora `x` 5-krát postupne pre všetky zložky.

2.2 Načítanie dát zo súboru

Väčšie dátové objekty je lepšie zadávať ako hodnoty z externého súboru ako zadávať ich počas práce **R** z klávesnice. Vstupné nástroje **R** sú jednoduché a ich požiadavky sú dosť presné a viacmenej nepružné. Existuje jasný predpoklad tvorcov systému **R**, že budete schopní upravovať svoje vstupné súbory pomocou iných prostriedkov, ako sú napríklad textové editory alebo Perl[11], aby ste ich pripravili podľa požiadaviek **R**. Vo všeobecnosti je to veľmi jednoduché.

Ak majú byť premenné uložené najmä do dátových rámčekov, čo veľmi odporúčame, celá dátova tabuľka (rámček) môže byť načítaná priamo funkciou `read.table()`. Existuje aj jednoduchšia vstupná funkcia `scan()`, ktorá môže byť volaná priamo.

Podrobnejšie informácio o importovaní a exportovaní údajov nájdete v príručke **R** Data Import/Export [22].

2.2.1 Funkcia `read.table()`

Externý súbor, z ktorého sa priamo načítava tabuľka údajov, má obyčajne špecialny tvar:

- Prvý riadok súboru musí obsahovať názvy všetkých premenných tabuľky.
- Každý ďalší riadok obsahuje ako prvú položku riadkové návestie (anglicky label) a hodnoty všetkých premenných.

[11] Pod OS GNU/Linux môžete použiť služby Sed alebo Awk.

Ak má súbor v prvom riadku o jednu položku menej ako v druhom, predpokladá sa, že platia vyššie uvedené predpoklady. Prvých niekoľko riadkov súboru môže teda vyzerať nasledovne:

Vstupný súbor s názvami a označením riadkov:

```
     Price   Floor   Area   Rooms   Age   Cent.heat
01   52.00   111.0    830      5    6.2       no
02   54.75   128.0    710      5    7.5       no
03   57.50   101.0   1000      5    4.2       no
04   57.50   131.0    690      6    8.8       no
05   59.75    93.0    900      5    1.9      yes
...
```

Implicitne sa číselné položky (s výnimkou označení riadkov) načítavajú ako číselné premenné a nečíselné premenné, ako napr. `Cent.heat` v príklade, sa načítavajú ako faktory. V prípade potreby sa to dá zmeniť.

Funkcia `read.table()` sa dá použiť na priame načítanie dátovej tabuľky nasledovne:

```
> HousePrice <- read.table("houses.data")
```

Často chceme vynechať zadanie riadkových návestí a použiť implicitné označenia. V tom prípade je možné vynechať stĺpec riadkových značiek, ako, napríklad, v nasledujúcom príklade:

Vstupný súbor bez označenia riadkov:

```
Price   Floor   Area   Rooms   Age   Cent.heat
52.00   111.0    830      5    6.2       no
54.75   128.0    710      5    7.5       no
57.50   101.0   1000      5    4.2       no
57.50   131.0    690      6    8.8       no
59.75    93.0    900      5    1.9      yes
 ...
```

V tom prípade na načítanie tabuľky údajov použijeme príkaz

```
> HousePrice <- read.table("houses.data", header=TRUE)
```

kde voľba `header=TRUE` určuje, že prvý riadok je riadok hlavičiek (názvov), a teda z tvaru súboru implicitne vyplýva, že nie sú zadané explicitné riadkové návestia.

2.2.2 Funkcia `scan()`

Predpokladajme teraz, že dátové vektory majú rovnakú dĺžku a majú byť načítané súčasne. Predpokladajme navyše, že máme tri vektory, prvý má formát reťazca a ostatné dva sú číselné. Všetky tri vektory tvoria tri stĺpce súboru `input.dat`.

```
a 1 5
b 2 6
c 3 7
d 4 8
```

Prvým krokom je použitie príkazu `scan()` na načítanie troch vektorov ako zoznam (anglicky list) nasledovne:

```
> inp <- scan("input.dat", list("",0,0))
> inp
[[1]]
[1] "a" "b" "c" "d"

[[2]]
[1] 1 2 3 4

[[3]]
[1] 5 6 7 8
```

Druhým argumentom je fiktívny zoznam (a dummy list structure), ktorý určuje formát troch vektorov, ktoré sa majú načítať. Výsledok, uložený v premennej `inp`, je zoznam, ktorého zložkami sú tri načítané vektory. Na oddelenie jednotlivých položiek do troch samostatných vektorov použijeme nasledovné priradenia:

```
> label <- inp[[1]]; x <- inp[[2]]; y <- inp[[3]]
```

V prípade, keď v dátovom súbore používame ako separátor medzi stĺpcami čiarku, príkaz `scan` rozšírime o argument `sep=","`

```
> inp <- scan("input.dat", sep=",", list("",0,0))
```

Pohodlnejšie je zadať fiktívny zoznam v mennom tvare (s označením názvov stĺpcov). V tom prípade sa názvy dajú použiť na prístup k načítaným vektorom. Napríklad

```
> inp <- scan("input.dat", list(id="", x=0, y=0))
> inp
$id
[1] "a" "b" "c" "d"

$x
[1] 1 2 3 4

$y
[1] 5 6 7 8
```

Ak chcete ku premenným pristupovať oddelene, môžu byť buď priradené premenným v pracovnom prostredí:

```
> label <- inp$id; x <- inp$x; y <- inp$y
```

alebo zoznam môže byť pridaný na 2. miesto vyhľadávacej cesty.

Ak je druhý argument jednoduchá hodnota a nie zoznam, načíta sa jediný vektor, ktorého všetky zložky musia mať rovnaký formát ako daná fiktívna hodnota:

```
> X <- matrix(scan("light.dat", 0), ncol=5, byrow=TRUE)
```

Existujú aj prepracovanejšie prostriedky na načítanie údajov, sú podrobnejšie popísané v rôznych príručkách.

2.2.3 Editácia údajov

Po vyvolaní dátovej tabuľky alebo matice, príkaz `edit` otvorí špeciálne prostredie na editovanie tabuliek. Toto prostredie je užitočné na vykonanie drobných zmien po načítaní údajov. Príkazom

```
> xnew <- edit(xold)
```

umožníme editáciu dátového súboru `xold`, pričom pozmenený objekt bude priradený premennej `xnew`. Ak chcete zmeniť pôvodný dátový súbor `xold`, najjednoduchšou cestou je použitie príkazu `fix(xold)`, čo je ekvivalentné s použitím príkazu `xold <- edit(xold)`.

Použitím príkazu

```
> xnew <- edit(data.frame())
```

sa otvorí grafické okno editora na vytvorenie nových údajov prostredníctvom tabuľkového prostredia. Týmto príkazom môžeme upravovať aj už vytvorené dátové objekty:

```
> xnew <- edit(data.frame(xold))
```

kliknutím na položku `Quit` sa pozmenený súbor `xold` uloží pod novým názvom `xnew`.

3 Použitie štatistických modelov programu R

Pri spracovaní a interpretácii empirických dát nás v prvom rade zaujíma ich variabilita. Variabilita experimentálnych údajov má obyčajne viacero príčin. Predovšetkým sa zaujímame o efekty v systéme merania a v systéme, ktorý sledujeme, najmä o systematické vplyvy. Cieľom je porozumieť týmto vplyvom, aby sme mohli na systém merania prípadne pôsobiť a tým nežiadúce vplyvy úplne vylúčiť. Táto zložka variability je obyčajne prekrytá inou, ktorej príčinou je nepresnosť merania a ďalšie náhodné vplyvy. K ním sa ešte pridáva aj výberová chyba, ktorá vzniká tým, že namerané údaje obvykle zastupujú väčší súbor. Údaje

(experimentálne dáta) v niektorých charakteristikách nesúhlasia vždy presne s charakteristikami celého súboru (populácie), ktorých sa naše poznatky majú týkať. Snažíme sa teda identifikovať základné vzťahy variability.

Vo výskume nám úlohu štatistiky najlepšie približuje predstava o *signále* a *šume*. Predstava je taká, že dáta majú tvar

```
DÁTA = SIGNÁL + ŠUM,
```

kde signál reprezentuje informáciu o sledovanom systéme. Táto konštrukcia odzrkadľuje, že signál vyjadrujúci určité „pravidelnosti“ je viac-menej prekrytý šumom. Dáta však môžu byť interpretované len keď disponujeme ich *modelom*, ktorého hlavnou časťou je nejaká rovnica. Modelom sa usilujeme vystihnúť podobu signálu, zjednodušene povedané model odzrkadľuje štruktúru dát. V štatistike teda dáta modelujeme podľa schémy

```
DÁTA = MODEL(FUNKCIA) + REZIDUÁLNA HODNOTA,
```

kde reziduálna hodnota, alebo tiež chyba, vyjadruje rozdiel medzi modelom a meranými dátami, pričom je dôležité, aby tieto hodnoty neobsahovali žiadnu evidentnú štruktúru. Pokiaľ by reziduálne hodnoty mali nejakú štruktúru, je potrebné model upravovať až dovtedy, kým nezahrnie všetky identifikovateľné štruktúry signálu. Modelom sa teda usilujeme čo najvernejšie reprezentovať reálny meraný systém.

3.1 Lineárny model

Podkladom na zavedenie štatistického modelu je lineárny regresný model s nezávislými homoskedastickými chybami e_i, ktorý môžeme zapísať v tvare

$$y_i = \sum_{j=0}^{p} \beta_j x_{ij} + e_i, \quad i = 1, \ldots, n, \tag{1}$$

kde e_i majú normálne rozdelenie NID$(0, \sigma^2)$.[12] V maticovom tvare bude rovnica (1) vyzerať takto

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{e}, \tag{2}$$

kde $\boldsymbol{y}$ je vektor závisle premennej alebo tiež odozva, $\boldsymbol{\beta}$ je vektor neznámych parametrov, $\boldsymbol{X}$ je modelová matica alebo matica funkčných hodnôt[13] so stĺpcami x_0, x_1, …, x_p, ktoré určujú nezávisle premenné. Často je stĺpec x_0 jednotkový a definuje hodnoty „priesečníkov s osou y“ (an *intercept* term, ďalej budeme používať výraz „priesečník“).

Uvedieme niekoľko jednoduchých užitočných príkladov. Predpokladajme, že `y, x, x0, x1, x2, ...` sú numerické premenné a `X` je matica, potom nasledujúcimi vzťahmi môžeme špecifikovať niektoré základné lineárne modely:

[12] NID – Normally and Independently Distributed (statistics). V prípade, keď táto podmienka nie je pre naše dáta splnená, môžeme použiť funkciu pre všeobecný lineárny `glm()`, ktorá disponuje rodinou základných distribučných funkcií (napr. `poisson`, `binomial`, `gaussian`, `inverse`, `Gamma`, `quasi`).

[13] Niekedy ju označujú ako maticu hodnôt pozorovaní nezávisle premenných.

`y ~ x`
`y ~ 1 + x` obidva príklady vyjadrujú jednoduchý lineárny regresný model $y(x)$. V prvom je hodnota „priesečníka“ implicitná, v druhom je explicitne určená hodnota 1.

`y ~ 0 + x`
`y ~ -1 + x`
`y ~ x - 1` toto sú jednoduché modely lineárnej regresie $y(x)$, ktoré prechádzajú počiatkom.

`log(y) ~ x1 + x2` multilineárna regresia s transformovanou premennou $\log(y)$ a implicitnou hodnotou „priesečníka“.

`y ~ poly(x,2)`
`y ~ 1 + x + I(x^2)` polynomiálna regresia $y(x)$ druhého stupňa. Prvý tvar používa ortogonálne polynómy a druhý používa explicitné umocnenie ako základ.

`y ~ X + poly(x,2)` mnohonásobná regresia y s maticovým modelom, ktorý obsahuje maticu X a polynomiálne hodnoty x druhého stupňa.

Operátor `~` je používaný na definovanie *predpisu modelu* v programe **R**. Tvar pre všeobecný lineárny model je (v podstate to platí aj pre nelineárny model)

```
response ~ op_1 term_1 op_2 term_2 op_3 term_3 ...,
```

kde `response` je vektor alebo matica, ktorá definuje rozsah premenných, `op_i` je operátor `+` alebo `-` vyjadrujúci včlenenie alebo vylúčenie výrazu `term_i` do modelu, `term_i` môže byť vektor, matica alebo `1`, ľubovolný činiteľ alebo vyjadrenie, ktoré obsahuje činitele, vektory alebo matice spojené operátorovým vyjadrením. Viac sa dočítate v príručke k programu **R** [22].

Základnou funkciou na fitovanie všeobecných mnohonásobných modelov je funkcia `lm()`, jej volanie v príkazovom riadku je nasledovné:

```
> fitted.model <- lm(formula, data = data.frame)
```

Napríklad príkazom

```
> fitlin <- lm(y ~ 1 + x, data = dataset)
```

budeme fitovať regresný model $y(x) = a + bx$ pre dáta `dataset` s implicitnými hodnotami „priesečníkov“, pričom štatistické výsledky a informácie budú uložené do objektu `fitlin`.

Dôležitý, avšak voliteľný parameter `data = dataset` stanovuje, že premenné potrebné na overenie modelu budú najprv načítané zo súboru dát `dataset`. V tomto prípade bez

ohľadu na to, či je prístup k súboru `dataset` zapísaný do vyhľadavacej cesty (search path) alebo nie.

3.1.1 Funkcie na získanie informácií o modely

Hodnoty funkcie `lm()` sú výsledkom fitovaného modelu; výpis výsledkov je triedou `"lm"`, ktorá môže obsahovať tieto položky:

```
add1    coef      effects  kappa   predict  residuals
alias   deviance  family   labels  print    step
anova   drop1     formula  plot    proj     summary
```

Stručne vysvetlíme význam najpoužívanejších:

`anova(object_1, object_2)` porovnáva submodel s vonkajším modelom a vytvára tabuľku analýzy rozptylu.

`coefficients(object)` vyberá regresné koeficienty (maticu), skrátená forma má tvar `coef(object)`.

`deviance(object)` reziduálna suma štvorcov odchýlok, je váhovaná, ak sú váhy priradené.

`formula(object)` vyberá formulu (rovnicu) modelu.

`plot(object)` vytvára štyri grafy, ktoré zobrazia rezíduá, fitované hodnoty a niektoré diagnostiky.

`print(object)` vytlačí stručnú verziu objektu. Často sa používa implicitne.

`residuals(object)` vyberá maticu rezíduí, je váhovaná, ak sú váhy priradené, skrátená forma má tvar `resid(object)`.

`step(object)` vyberá vhodný model zvyšovaním alebo znižovaním členov a uchováva ich hierarchiu. Model s najmenšou hodnotou AIC (Akaikeho informačné kritérium) sa zobrazuje postupným hľadaním s návratom.

`summary(object)` vytlačí sa obsiahly súhrn výsledkov regresnej analýzy.

`vcov(object)` vráti variančnú a kovariančnú maticu hlavných parametrov fitovaného modelu.

3.1.2 Príklad použitia lineárneho modelu

V krátkosti opíšeme postup lineárnej regresie modelovou funkciou $y = a + bx$, ktorú použijeme na určenie teplotného koeficientu elektrického odporu. Odpor vodiča bol meraný pomocou tzv. Wheatstonovho mostíka, ktorého zapojenie a postup merania je uvedený v knižke *Fyzikálne merania* [21].

Z experimentov vyplýva, že zväčšenie odporu $\mathrm{d}R$, pri vzraste teploty $\mathrm{d}t$ je priamo úmerné veľkosti odporu R a prírastku teploty $\mathrm{d}t$

$$\mathrm{d}R = \alpha R \,\mathrm{d}t, \tag{3}$$

kde koeficient priamej úmernosti α charakterizuje teplotnú závislosť odporu jednotlivých materiálov a nazývame ho *teplotný koeficient odporu*. Zo vzťahu (3) vyplýva definičný vzťah pre α

$$\alpha = \frac{1}{R}\frac{\mathrm{d}R}{\mathrm{d}t} \quad \left(\mathrm{K}^{-1}\right). \tag{4}$$

V prípade kovových vodičov je koeficient α prakticky nezávislý od teploty. Integrovaním vzťahu (3) od referenčnej teploty t_0 po teplotu t (za predpokladu $\alpha(t-t_0) \ll 1$) dostaneme pre odpor pri teplote t

$$R_t = R_0\left[1+\alpha(t-t_0)\right], \tag{5}$$

kde R_0 je odpor pri teplote t_0.

Dáta získane z experimentu boli uložené do súboru `e6.dat` v tvare tabuľky :

```
x y
25 101.61
30 105.33
35 106.48
40 109.64
45 111.86
50 114.13
55 115.98
60 118.81
65 120.26
70 122.22
```

Načítaním dát do prostredia programu **R** vytvoríme objekt `dataset`:

```
dataset <- read.table("/home/laco/aplimat/e6.dat", header=TRUE, sep="",
dec=".")
x <- dataset$x; y <- dataset$y
```

Na určenie optimálnych parametrov a, b a ich neistôt, pričom parameter a reprezentuje hľadanú veličinu R_0 a parameter b zasa veličinu αR_0, čiže $\alpha = b/a$, použijeme funkciu `lm()`. Hľadané parametre môžeme uložiť do objektu `plm` a všetky informácie o fitovaní obsahuje objekt `fitlm`:

```
fitlm <- lm(y ~ 1 + x, data=dataset)
plm <- coef(fitlm)

> plm
(Intercept)           x
 91.1596970   0.4520485

> summary(fitlm)
Call:
```

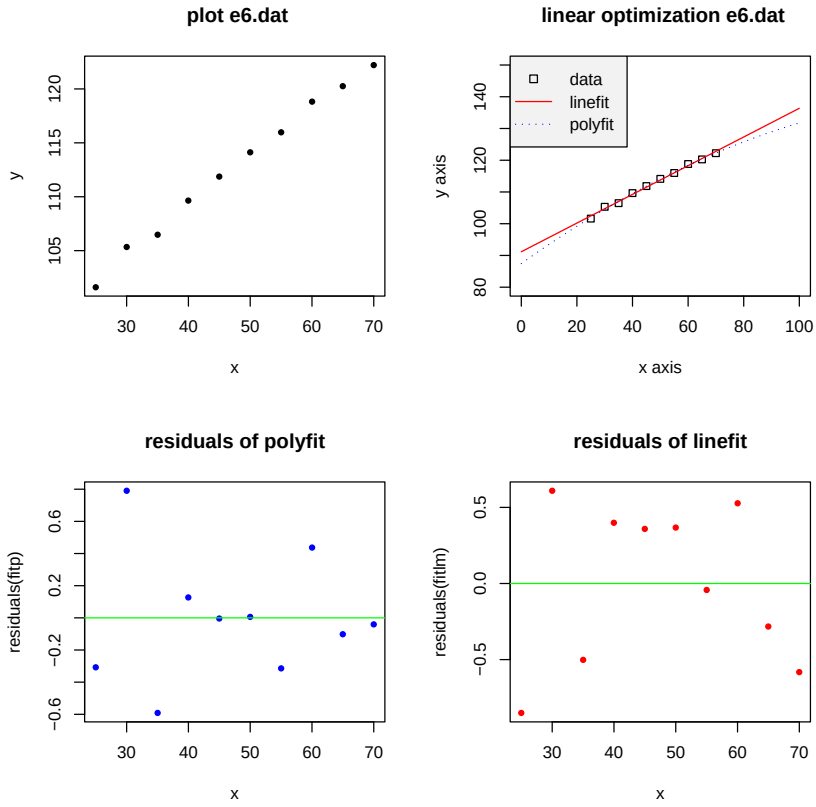
```
lm(formula = y ~ 1 + x, data = dataset)
Residuals:
    Min      1Q  Median      3Q     Max
-0.8509 -0.4468  0.1579  0.3907  0.6088
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 91.15970    0.60695  150.19 4.32e-15 ***
x            0.45205    0.01223   36.96 3.15e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.5555 on 8 degrees of freedom
Multiple R-Squared: 0.9942,     Adjusted R-squared: 0.9934
F-statistic:  1366 on 1 and 8 DF,  p-value: 3.15e-10
```

V ďalších riadkoch sú príkazy, ktorými vygenerujeme dáta na zobrazenie optimálnej krivky $y = a + bx$ a na ukážku porovnania dvoch matematických modelov preložíme ešte dáta aj polynómom druhého stupňa. Takto získaným parametrom však podľa nášho fyzikálneho modelu nemôžeme pririeknuť fyzikálny význam:

```
xfit <- seq(0,100,0.01)
yfitl <- plm[1] + plm[2] * xfit
fitp <- lm(y ~ 1 + x + I(x^2))
ppoly <- coef(fitp)
yfitp <- ppoly[1] + ppoly[2]*xfit + ppoly[3]*xfit^2
```

Nasledujú príkazy na grafické zobrazenie výsledkov fitovania, zápisom `par(mfrow=c(2,2))` rozdelíme grafické okno na štyri časti, znak # používame na komentovanie a vytváranie poznámok do zdrojového kódu. Výsledok je znázornený na obrázku 1:

```
par(mfrow=c(2,2))
# plot data
plot(x, y, pch=20)
title(" plot e6.dat")
# plot optimization
plot(x,y,pch=22,xlab="x axis",ylab="y
axis",xlim=c(0,100),ylim=c(80,150),bty="o")
lines(xfit, yfitl, col=2)
lines(xfit, yfitp, col=4, lty=3)
title(" linear optimization e6.dat")
legend("topleft", c("data","linefit","polyfit"),
lty=c("blank","solid","dotted"), pch=c(0,NA,NA),
col=c("black","red","blue"), bg="gray95")# title="LEGEND"
# plot residulals of polyfit
plot(x, residuals(fitp), pch=20, col=4, main="residuals of polyfit")
abline(h=0, v="", col="green") # line only in x-direction
# plot residuals of linefit
plot(x, residuals(fitlm), pch=20, col=2, main="residuals of linefit")
abline(h=0, v="", col="green") # line only in x-direction
```



Obr. 1: Priebehy výsledkov analýzy lineárnym modelom. V pravom hornom grafe sú znázornené fitované krivky. Dolné piebehy zobrazujú rozptyl rezíduí pre polynomiálnu a lineárnu regresiu. Napriek tomu, že priebeh pre `polyfit` sa „javí prijateľnejší“ (tesnejšie preloženie bodov kvadratickou krivkou), musíme ho na základe už spomenutej fyzikálnej interpretácie odmietnuť

Pre experimentátora je dôležité ukladanie výsledkov analýzy do niektorého kvalitného vektorového grafického formátu vhodného na dalšie publikovanie (napr. PDF, EPS a pod.) a taktiež numerické výsledky do `ASCII` textového súboru. Nasledujúce príkazy slúžia tomuto účelu.

```
# print figure as *.PDF file
dev.print(pdf,file="/home/laco/aplimat/e6-aplimat.pdf")
# zapis vysledkov linefit a polyfit do suborov *.txt
boxl <- summary(fitlm)
parl <- boxl$coef[,]
write.table(parl, file="/home/laco/aplimat/e6-l-par.txt")
boxp <- summary(fitp)
parp <- boxp$coef[,]
write.table(parp, file="/home/laco/aplimat/e6-p-par.txt")
```

Pri konštrukcii regresného modelu je obvykle našou snahou minimalizovať variabilitu vý-

sledku. *Zníženie variability* (numericky vyjadrenej najčastejšie pomocou smerodajnej odchýlky, intervalu spoľahlivosti alebo neistoty) *znamená zvýšenie presnosti*. Môže pritom ísť o presnosť napr. regresných koeficientov (parametrov), predikciu, limity detekcie a pod. Najčastejšie používané veličiny (štatistiky) na vyjadrenie variability sú:

- reziduálny rozptyl,
- smerodajná odchýlka parametrov,
- interval spoľahlivosti.

Vo všetkých veličinách má rozhodujúci vplyv počet stupňov voľnosti $n-p$, kde n je počet experimentálnych dát a p je počet parametrov modelu. *Rastúci počet parametrov regresného modelu prispieva k zvýšeniu variability*, teda k zníženiu „presnosti“ regresného modelu. Keď táto skutočnosť nie je prevážená výrazne lepším (tesnejším) preložením dát, nemá pridávanie ďalších parametrov (napr. zvyšovaním stupňa polynómu) žiadny zmysel. Preto *nemôžeme* na hodnotenie regresie *použiť korelačný koeficient* alebo koeficient viacnásobnej korelácie, ktorý počet parametrov vôbec neberie do úvahy. Na výber optimálneho modelu môžeme použiť napr. *Akaikeho informačné kritérium* – AIC $(\min)$

$$\mathrm{AIC} = n \ln\left(\frac{\mathrm{SSE}}{n}\right) + 2p, \tag{6}$$

kde SSE (sum of squares error, the residual sum of squares, or error sum of squares) je reziduálny súčet štvorcov, $\mathrm{SSE} = \sum_{i=1}^{n} e_i$

```
# compute the Akaike information criterion
> AIC(fitlm)
[1] 20.38844
> AIC(fitp)
[1] 16.62088
```

Ako každy dobrý program zameraný na štatistiku aj **R** poskytuje analýzu rozptylu (ANOVA, *analysis of variance*). Stručne vyjadrené, všeobecne je základná funkcia analýzy rozptylu založená na posúdení hlavných a interakčných efektov nezávisle premenných na závisle premenné. Základnou štatistikou v analýze rozptylu je F-testovacia štatistika rozdielnosti skupinových priemerov, pomocou ktorej sa testuje hypotéza, či sa priemery v skupinách od seba líšia [9]. Príklad použitia funkcie `anova()` na naše dva jednoduché matematické modely ilustrujú nasledujúce riadky:

```
# compare two models...
# compare a submodel with an outer model
# and produce an analysis of variance table
> anova(fitlm,fitp)
Analysis of Variance Table
Model 1: y ~ 1 + x
Model 2: y ~ 1 + x + I(x^2)
  Res.Df    RSS Df Sum of Sq      F  Pr(>F)
```

```
1      8 2.4684
2      7 1.3865  1    1.0818 5.4618 0.05207 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Podrobnejšie o problematike spracovania experimentálnych dát nájde záujemca v špecializovanej literatúre, základné odkazy sú uvedené na konci tohto príspevku v zozname použitej literatúry. Úvod do uvedenej problematiky je autorom spracovaný vo forme slajdov a je voľne prístupný na URL adrese `http://people.tuke.sk/ladislav.sevcovic/esf/esf2-sevcovic-talk.pdf`.

3.2 Nelineárny model

Istou formou nelineárneho modelu vhodného na fitovanie je všeobecný lineárny model `glm()`, ale vo väčšine prípadov volíme niektorý postup nelineárnej optimalizácie. Program **R** ponúka niekoľko modulov na nelineárnu optimalizáciu ako sú `nls()`, `optim()`, `nlm()` a (od verzie **R** 2.2.0) `nlminb()`, ktoré sú funkciami podobné modulom `ms()` a `nlminb()` z programu S-Plus.

V tejto časti sa budeme zaoberať modelom opisujúcim závislosť medzi závislou a nezávislou premennou funkcie nelineárnej vzhľadom na neznáme regresné parametre. Budeme uvažovať regresný model opísaný nelineárnou regresnou funkciou $f(\boldsymbol{x},\boldsymbol{\beta})$ v aditívnom tvare

$$\boldsymbol{y} = f(\boldsymbol{x},\boldsymbol{\beta}) + \boldsymbol{e}, \tag{7}$$

kde $\boldsymbol{x}$ je vektor vysvetľujúcich (nezávisle) premenných a $\boldsymbol{\beta}$ je p členný vektor neznámych regresných parametrov. Na náhodné chyby e_i, $i = 1,2,\ldots,n$, budeme klásť predpoklady klasického regresného modelu, teda musia mať normálne rozdelenie $\mathrm{NID}(0,\sigma^2)$. Nesmieme však zabúdať, že pri aplikácii nelineárnej regresie musíme často uvažovať modely s náhodnými chybami v multiplikatívnom tvare, alebo zmiešané modely, ktoré obsahujú náhodné vplyvy v aditívnom aj multiplikatívnom tvare. Výnimkou nie je ani použitie modelov s predpokladom všeobecnej pravdepodobnostnej funkcie rozdelenia náhodných chýb.

Hodnoty neznámych parametrov sú vyhľadávané iteračne minimalizovaním niektorého ukazovateľa kvality fitovania. Na rozdiel od lineárnej regresie, napríklad, nie je zaručené, že procedúra bude konvergovať k uspokojivým odhadom. Všetky metódy vyžadujú začiatočné odhady hľadaných parametrov a konvergencia závisí od presnosti týchto štartovacích hodnôt.

3.2.1 Príklad použitia nelineárneho modelu

Jednou z možností fitovania nelineárneho modelu je minimalizácia sumy štvorcov odchýlok (SSE – sum of the squared errors) alebo tiež rezíduí. Táto metóda má zmysel vtedy, keď pozorované chyby majú normálne rozdelenie.

Uvedieme príklad z oblasti NMR (nuclear magnetic resonance). Dáta boli získané meraním spinovo-mriežkového relaxačného času $T_{1\rho}$ v rotujúcom súradnicovom systéme (RSS) polymérneho materiálu KELTAN komerčným impulzným spektrometrom (základné informácie o metodike merania sú vo forme PDF dokumentu voľne prístupné na URL adrese `http://people.tuke.sk/ladislav.sevcovic/relaxacnycas_t1r.pdf`). Dáta sú nasledovné:

```
# Keltan T1ro Intensity fit
# Tau(ms) Expt
x y
0.1 1.0
0.4 0.9315
0.8 0.8460
1.0 0.8091
2.0 0.6496
3.0 0.5255
4.0 0.4267
6.0 0.2896
8.0 0.2009
10.0 0.1411
12.0 0.1002
14.0 0.07266
16.0 0.05412
18.0 0.0404
```

Prvý stĺpec `x` reprezentuje čas trvania spin-lock impulzu τ_{SL} v milisekundách a druhý stĺpec `y` amplitúdu NMR signálu reprezentovaného magnetizáciou meranej vzorky.

V takýchto polymérnych systémoch môžu jednotlivé štrukturálne fázy relaxovať nezávisle, čiže každej môžeme priradiť vlastný relaxačný čas $T_{1\rho}$ a amplitúdu signálu voľnej precesie vyjadriť ako superpozíciu niekoľkých (v praxi najviac troch) exponenciálnych zložiek vzťahom

$$M(\tau_{\mathrm{SL}}) = M_0 \sum_i p_i \exp(-\tau_{\mathrm{SL}}/T_{1\rho}{}^i), \tag{8}$$

kde p_i sú relatívne intenzity jednotlivých zložiek magnetizácie, pričom platí $\sum_i p_i = 1$.

Na základe technologickej prípravy a chemického zloženia usudzujeme, že skúmaný materiál ma byť jednozložkový. Táto skutočnosť nám umožňuje z hľadiska fyzikálnej interpretácie vykonať analýzu dát len dvoma matematickými modelmi, s jednoexponenciálnym a dvojexponenciálnym priebehom. Na získanie optimálnych parametrov uvedených matematických modelov použijeme modul `nls()`.

Tak ako v predošlom príklade, analýzu začneme načítaním dát do objektu programu **R**, nazveme ho opäť `dataset`:

```
dataset <-
read.table("/home/laco/aplimat/keltt1rint.dat", header=TRUE, sep="",
dec=".")
```

```
#dataset <- read.table(
#url("http://people.tuke.sk/ladislav.sevcovic/esf/data/keltt1rint.dat")
#, sep="", dec=".")

x <- dataset$x; y <- dataset$y
```

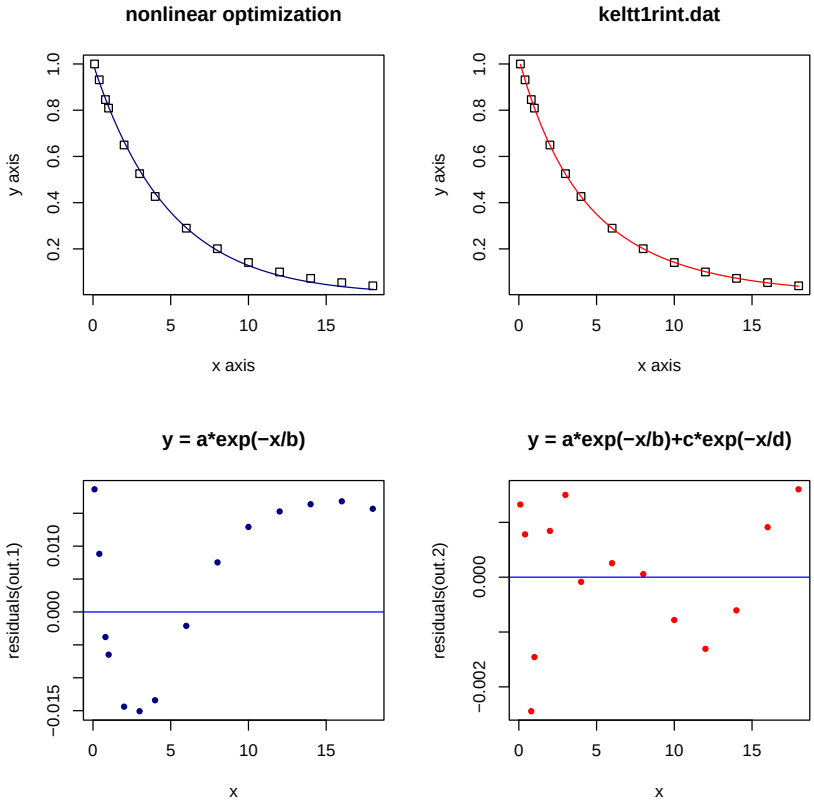
Často sa stáva, že dáta sú z dôvodu spoločného prístupu uložené na nejakej URL adrese. Použitím príkazu, ktorý je na zakomentovaných riadkoch (symbolom #), môžeme dáta načítať zo servera. V nasledujúcich riadkoch definujeme modelové funkcie, položkou `start` zadávame začiatočné odhady hľadaných parametrov, nasleduje výpis hodnôt optimálnych parametrov modelov a napokon vygenerujeme dáta na zobrazenie optimálnych kriviek

```
out.1 <- nls(y~a*exp(-x/b), start=c(a=1, b=5))
out.2 <- nls(y~a*exp(-x/b)+c*exp(-x/d), start=c(a=.4, b=2, c=.6, d=5))

par1 <- coef(out.1)
> par1
       a        b
1.001769 4.863262

par2 <- coef(out.2)
> par2
        a         b         c         d
0.4159660 2.7808638 0.6066518 6.5070024

xfit <- seq(0.1,18,0.01)
yfit1 <- par1[1] * exp(- xfit/par1[2])
yfit2 <- par2[1] * exp(- xfit/par2[2]) + par2[3] * exp(- xfit/par2[4])
```

Tak, ako v prípade lineárnej regresie, nasledujú príkazy na grafické zobrazenie výsledkov fitovania a uloženie obrázku do formátu PDF. Výsledok je znázornený na obrázku 2:

```
par(mfrow=c(2,2))
# (1,1)
plot(x, y, pch=0, xlab="x axis",ylab="y axis", main="nonlinear
optimization")
lines(xfit, yfit1, col="navy")
#title("nonlinear optimization: keltt1rint.dat")
# (1,2)
plot(x, y, pch=0, xlab="x axis",ylab="y axis", main="keltt1rint.dat")
lines(xfit, yfit2, col="2")
# (2,1)
plot(x, residuals(out.1), col="navy",pch=20, main="y = a*exp(-x/b)")
abline(h=0, v="", col="blue") # line only in x-direction
# (2,2)
plot(x, residuals(out.2), col="red", pch=20, main="y =
```

```
a*exp(-x/b)+c*exp(-x/d)")
abline(h=0, v="", col="blue") # line only in x-direction
#
dev.print(pdf, file = "/home/laco/aplimat/kelt-aplimat.pdf")
```



Obr. 2: Priebehy výsledkov analýzy nelineárnym modelom. V ľavom hornom grafe je znázornený priebeh fitovanej krivky pre jednoexponeciálny model, pod ním je priebeh rezíduí. V pravom stĺpci sú výsledky pre dvojexponenciálny model. V úvodnej časti o štatistických modeloch sme spomenuli, že rezíduá nesmú mať žiadnu evidentnú štruktúru. Tejto požiadavke jednozložkový model nevyhovuje, výsledky pre dvojzložkový sú zo štatistického hľadiska prijateľnejšie, nie sú však v súlade s prijatým fyzikálnym modelom (interpretáciou). Východiskom môže byť zväčšenie počtu experimentálnych bodov, zopakovanie experimentu za iných podmienok alebo pripustiť možnosť zložitejšieho fyzikálneho modelu (dvojzložkového) skúmaného polymérneho materiálu

Všetky informácie o hľadaných parametroch s ich neistotami sú uložené v objektoch `out.1` a `out.2`. Výsledky sa nám zobrazia spustením nasledujúcich príkazov:

```
> summary(out.1)
Formula: y ~ a * exp(-x/b)
Parameters:
```

```
  Estimate Std. Error t value Pr(>|t|)
a 1.001769   0.008542  117.28  < 2e-16 ***
b 4.863262   0.098121   49.56 2.98e-15 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
Residual standard error: 0.014 on 12 degrees of freedom
Correlation of Parameter Estimates:
        a
b -0.6083

> summary(out.2)
Formula: y ~ a * exp(-x/b) + c * exp(-x/d)
Parameters:
  Estimate Std. Error t value Pr(>|t|)
a  0.41597    0.04341   9.582 2.35e-06 ***
b  2.78086    0.14548  19.115 3.34e-09 ***
c  0.60665    0.04398  13.794 7.80e-08 ***
d  6.50700    0.19469  33.422 1.36e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
Residual standard error: 0.001399 on 10 degrees of freedom
Correlation of Parameter Estimates:
        a       b       c
b  0.9811
c -0.9997 -0.9843
d  0.9905  0.9540 -0.9896
```

Porovnajme ešte oba modely použitím Akaikeho informačného kritéria a použitím analýzy rozptylu ANOVA. Ako sme to už spomenuli, čím je hodnora AIC nižšia, tým je model presnejší.

```
# compute the Akaike information criterion
> AIC(out.1)
[1] -77.94952

> AIC(out.2)
[1] -140.9884

> anova(out.1,out.2)
Analysis of Variance Table
Model 1: y ~ a * exp(-x/b)
Model 2: y ~ a * exp(-x/b) + c * exp(-x/d)
  Res.Df Res.Sum Sq Df     Sum Sq F value    Pr(>F)
1     12 0.00235231
2     10 0.00001958  2 0.00233273   595.6 3.999e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Venujme ešte letmý pohľad zápisu hľadaných parametrov a ich neistôt do `ASCII` súboru. Sú

obdobné ako v časti o lineárnej regresii, nepotrebujú komentár:

```
box1 <- summary(out.1)
box2 <- summary(out.2)
pars1 <- box1$parameters[,] # short form box1$param[,]
# same as
# pars1 <- box1$coefficients[,] # short form box1$coef[,]
> pars1
  Estimate  Std. Error   t value     Pr(>|t|)
a 1.001769 0.008541948 117.27644 9.902846e-20
b 4.863262 0.098120605  49.56412 2.982630e-15

pars2 <- box2$parameters[,]
> pars2
   Estimate Std. Error   t value     Pr(>|t|)
a 0.4159660 0.04341195  9.581831 2.347991e-06
b 2.7808638 0.14547928 19.115188 3.338581e-09
c 0.6066518 0.04397866 13.794231 7.800836e-08
d 6.5070024 0.19469274 33.421905 1.358461e-11

write.table(pars1, file="/home/laco/aplimat/kelt-par1.txt")
write.table(pars2, file="/home/laco/aplimat/kelt-par2.txt")
```

Na týchto dvoch jednoduchých ukážkach použitia dvoch základných štatistických modelov sme mali možnosť presvedčiť sa, aké silné možnosti poskytuje práca s programom v interaktívnom móde, akým je programový systém **R**. Sme si vedomí, že sme ani zďaleka nevyčerpali všetky možnosti, ktoré použité programové moduly poskytujú. Na podrobnejšie oboznámenie sa s danou problematikou odporúčame čitateľovi špecializovanú literatúru, napr. [2, 4, 5, 7, 11, 12, 15–17] a manuály programu **R** [20].

4 Vytvorenie protokolu v LaTeXu

Často narážame na potrebu uložiť súbor štatistických výsledkov spracovania dát do jedného dokumentu prípadne publikácie. V predchádzajúcich častiach sme videli, že operácie v programovom prostredí **R** sa ukladajú do objektov a môžeme ich tiež ukladať do súborov.

Takto „exportované“ výsledky môžeme vkladať do textových editorov (napr. KOffice, Open Office, MS Office) alebo typografických programov ako napr. LaTeX. Program **R** disponuje priamym výstupom do LaTeXu a tak súčasné použitie oboch prostredí umožňuje vytvárať rozsiahle dokumenty, do ktorých môžeme vkladať počnúc zdrojovými kódmi, výsledkami analýzy aj grafy a samozrejme vlastné komentáre a interpretácie. Pre tých používateľov **R**-ka, ktorí ovládajú prácu s typografickým systémom LaTeX je teda výhodou, že sa nemusia učiť nové syntaktické príkazy a ovládanie nového softvéru.

Tento modul sa volá `Sweave`, umožňuje vytvoriť LaTeXovský zdrojový kód textu dokumentácie a kódu programu **R** zo súboru napísaného pomocou syntaxe programu NOWEB

po jeho prechode interpretérom **R**-ka. Výsledkom kompilácie budú tieto časti:

- text dokumentácie,
- **R** – input a/alebo
- **R** – output (text alebo grafika).

Prácu s modulom `Sweave` predvedieme na nasledujúcej ukážke. Nejakým `ASCII` textovým editorom, napr. v OS GNU/Linux môžeme použiť editor Kate, napíšeme nasledujúci súbor a uložíme ho s príponou `.rnw`, napr. `aplimat-sweave.rnw`

```
\documentclass[a4paper]{article}
\pdfoutput=1
\usepackage{graphicx}
%\usepackage[latin2]{inputenc}
%\usepackage[T1]{fontenc}
\usepackage{slovak}
\usepackage{Sweave}%{/usr/lib/R/share/texmf/Sweave}
\title{Sweave -- príklad použitia}
\author{Ladislav Ševčovič}
\voffset=-60pt
\textwidth = 400pt
\textheight = 700pt
\begin{document}
\maketitle
Tento príklad ukazuje, ako môžeme do prostredia programu \LaTeX{}
vložiť dva kódy programu R. Prvým kódom vykonáme fitovanie za účelom
získania optimálnych parametrov pre dáta {\tt bonate.dat}.
<<>>=
dataset <- read.table("/home/laco/aplimat/bonate.dat", header=TRUE,
sep="", dec=".")
x <- dataset$x; y <- dataset$y
out.2 <- nls(y~a*exp(-x*b)+c*exp(-x*d), start=c(a=200, b=.1, c=50,
d=.01))
par2 <- coef(out.2)
xfit <- seq(0.1,96,0.01)
yfit2 <- par2[1] * exp(- xfit*par2[2]) + par2[3] * exp(- xfit*par2[4])
@

\begin{figure}[!ht]
\begin{center}
<<fig=TRUE, echo=FALSE>>=
plot(x, y, pch=0, main="y = a*exp(-x*b)+c*exp(-x*d)")
lines(xfit, yfit2, col=2)
@
\caption{Obrázok príkladu}
\label{swxplfit}
\end{center}
\end{figure}
\end{document}
```

Úsek, ktorým sa začína zdrojový kód je označený znakmi `<<>>=` a končí znakom `@`. Všetko, čo napíšeme za týmto znakom sa preloží ako LaTeXovský text dokumentácie. V časti, kde je uvedený príkaz na vloženie grafu sú návestia pre dva argumenty. Argument `fig=TRUE` zabezpečí, že Sweave vytvorí jeden EPS a jeden PDF obrázok s názvom `aplimat-sweave-001` a vloží do výstupného súboru `aplimat-sweave.tex` príkaz `\includegraphics{aplimat-sweave-001}`, argumentom `echo=FALSE` sa nastaví, aby

sa R-input nevložil do výsledného dokumentu. Kompiláciu spustíme v prostredí programu **R** príkazom `Sweave('aplimat-sweave.rnw')`:

```
> Sweave('aplimat-sweave.rnw')
Writing to file aplimat-sweave.tex
Processing code chunks ...
 1 : echo term verbatim
 2 : term verbatim eps pdf
You can now run LaTeX on 'aplimat-sweave.tex'
```

Vytvorený súbor `aplimat-sweave.tex` obsahuje nasledujúce riadky:

```
\documentclass[a4paper]{article}
\pdfoutput=1
\usepackage{graphicx}
%\usepackage[latin2]{inputenc}
%\usepackage[T1]{fontenc}
\usepackage{slovak}
\usepackage{Sweave}%{/usr/lib/R/share/texmf/Sweave}
\title{Sweave -- príklad použitia}
\author{Ladislav Ševčovič}
\voffset=-60pt
\textwidth = 400pt
\textheight = 700pt
\begin{document}
\maketitle
Tento príklad ukazuje, ako môžeme do prostredia programu \LaTeX{}
vložiť dva kódy programu R. Prvým kódom vykonáme fitovanie za účelom
získania optimálnych parametrov pre dáta {\tt bonate.dat}.
\begin{Schunk}
\begin{Sinput}
> dataset <-
read.table("/home/laco/aplimat/bonate.dat",
+     header = TRUE, sep = "", dec = ".")
> x <- dataset$x
> y <- dataset$y
> out.2 <- nls(y ~ a * exp(-x * b) + c * exp(-x * d), start = c(a =
200,
+     b = 0.1, c = 50, d = 0.01))
> par2 <- coef(out.2)
> xfit <- seq(0.1, 96, 0.01)
> yfit2 <- par2[1] * exp(-xfit * par2[2]) + par2[3] * exp(-xfit *
+     par2[4])
\end{Sinput}
\end{Schunk}

\begin{figure}[!ht]
\begin{center}
\includegraphics{aplimat-sweave-001}
\caption{Obrázok príkladu}
\label{swxplfit}
\end{center}
\end{figure}
\end{document}
```

Po preLaTeXovaní napr. formátom `pdfcslatex` vznikne takýto dokument:

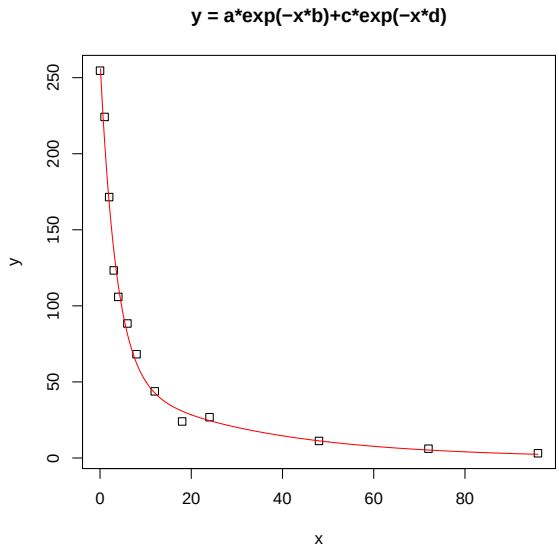
Sweave – príklad použitia

Ladislav Ševčovič

8. januára 2008

Tento príklad ukazuje, ako môžeme do prostredia programu LaTeX vložiť dva kódy programu R. Prvým kódom vykonáme fitovanie za účelom získania optimálnych parametrov pre dáta `bonate.dat`.

```
> dataset <- read.table("/home/laco/rvedok/Vzory/R/aplimat/bonate.dat",
+     header = TRUE, sep = "", dec = ".")
> x <- dataset$x
> y <- dataset$y
> out.2 <- nls(y ~ a * exp(-x * b) + c * exp(-x * d), start = c(a = 200,
+     b = 0.1, c = 50, d = 0.01))
> par2 <- coef(out.2)
> xfit <- seq(0.1, 96, 0.01)
> yfit2 <- par2[1] * exp(-xfit * par2[2]) + par2[3] * exp(-xfit *
+     par2[4])
```



Obr. 1: Obrázok príkladu

5 Záver

Našim zámerom je uľahčiť začínajúcim záujemcom štart pri použití programu **R** v takých oblastiach výučby a výskumu, kde je potrebné spracovať a vyhodnocovať empirické dáta. Príspevkom chceme poskytnúť základné informácie na zvládnutie práce s programom **R**, ako spracovať experimentálne dáta na ďalšiu kvalitatívnu analýzu, prípadne prezentáciu a ako vytvoriť grafické výstupy na profesionálnej úrovni vhodné na publikovanie.

Na komplexnú prácu s programom **R** v prostredí OS GNU/Linux môžeme použiť všestranný textový editor Kate, ktorý poskytuje zvýraznenie zdrojového kódu programu **R** a navyše aj priamu prácu v termináli (môžeme hneď spúšťať aj LaTeXovské formáty). To nám umožňuje editovať kód programu a zároveň ho aj spúšťať. Záverom poznamenajme, že súčasťou programu **R** je modul `Rcmdr`, ktorý je v podstate grafické užívateľské prostredie (GUI) a umožňuje pohodlnú prácu s programom. V prostredí UNIX sa program **R** spúšťa z príkazového riadku odoslaním príkazu `R`, ako sme to spomenuli v úvode. Po štarte programu **R** sa `RCommander` aktivuje načítaním knižnice `Rcmdr` zápisom a odoslaním príkazu

```
> library(Rcmdr)
```

Po načítaní knižnice sa nám otvorí okno programu. Podrobný prehľad a opis modulu nájdete v článku J. Foxa [6].

Literatúra

[1] BUŠA, J. – ŠEVČOVIČ, L.: **R** – Open source systém na spracovanie údajov. Obrazovková verzia na adrese: `http://people.tuke.sk/ladislav.sevcovic/esf/Rs.pdf`, brožúrková verzia na adrese: `http://people.tuke.sk/ladislav.sevcovic/esf/Rtlac.pdf`

[2] BRUNOVSKÁ, A.: *Malá optimalizácia*. Bratislava : Alfa, 1990, ISBN 80-05-00770-1

[3] DÁVID, A.: *Numerické metódy na osobnom počítači*. Bratislava : Alfa, 1988

[4] FARAWAY, J. J.: *Linear Models with R*. Boca Raton : A CRC Press Company, 2005 by Chapman & Hall/CRC, ISBN 1-58488-425-8

[5] FOX, J: *An R and S – Plus Companion to Applied regression*. Thousand Oaks : SAGE Publications, 2002, ISBN 0-7619-2280-6

[6] FOX, J.: *The R Commander: A Basic-Statistics Graphical User Interface to R*. In: Journal of Statistical Software, September 2005, Volume 14, Issue 9, online na `http://www.jstatsoft.org/v14/i09`

[7] GARCIA, A., L. 2000. *Numerical Methods for Physics*. New Jersey : Prentice-Hall, 2000, ISBN 013-906744-2

[8] HEBÁK, P. a kol.: *Vícerozměrné statistické metody 2*. Praha : Informatorium, 2005, ISBN 80-7333-036-9

[9] HENDL, J.: *Přehled statistických metod zpracování dat*. Praha : Portál, 2006, ISBN 80--7367-123-9

[10] IHAKA, J – GENTLEMAN, R.: *R: a language for data analysis and graphics*. Journal of Computational and Graphical Statistics, 1996, vol. 5, p. 299–314

[11] KAUKIČ, M.: *Numerická analýza I. Základné problémy a metódy*. Žilina : MC Energy, s. r. o. 1998

[12] KUDRACIK, F.: *Spracovanie experimentálnych dát*. Bratislava : Univerzita Komenského, 1999, ISBN 80-223-1327-0

[13] MELOUN, M. – MILITKÝ, J.: *Statistická analýza experimentálních dat*. Praha : Academia, 2004, ISBN 80-200-1254-0

[14] NIST *National Institute of Standards and Technology. Statistical reference Datasets*. `http://www.itl.nist.gov/div898/strd/general/dataarchive.html`

[15] PETROVIČ, P. – NADRCHAL, J. – PETROVIČOVÁ, J.: *Programovanie a spracovanie dát I., II.* Košice : Edičné stredisko UPJŠ, 1989

[16] PRESS, W. H. et al.: *Numerical Recipes in C – The Art of Scientific Computing*. New York : Cambridge University Press, 1992, 2nd Ed. Kniha v PDF formáte je dostupná na URL adrese: `http://www.nrbook.com/b/bookcpdf.php`

[17] RIEČANOVÁ, Z. a kol.: *Numerické metódy a matematická štatistika*. Bratislava : Alfa, 1987

[18] ŠEVČOVIČ, L.: *Programy na spracovanie a vizualizáciu experimentálnych dát*. Košice : TU v Košiciach, 2006, ISBN 80-8073-638-3, (obrazovková verzia ISBN 80--8073-639-1). Kniha v PDF formáte je dostupná na URL adrese: `http://people.tuke.sk/jan.busa/kega/`

[19] ŠEVČOVIČ, L.: *Spracovanie a vizualizáciu experimentálnych dát. Úvod do problematiky*. `http://people.tuke.sk/ladislav.sevcovic/esf/esf2-sevcovic-talk.pdf`

[20] *The R Project for Statistical Computing*, `http://www.r-project.org/`

[21] UHRIN, J. – ŠEVČOVIČ, L. – MURÍN, J.: *Fyzikálne merania*. Košice : elfa, 2006, ISBN 80-8086-032-7

[22] VENABLES, W. N. et al.: *An Introduction to R*. Published by Network Theory Limited, 2001, ISBN 0-9541617-4-2, online na adrese: `http://www.r-project.org/`

Kontaktná adresa

Ladislav ŠEVČOVIČ (RNDr.),
Katedra fyziky FEI TU v Košiciach,
Park Komenského 2,
041 20 Košice, SK
`ladislav.sevcovic@tuke.sk`