VÝUKA CHEMIE

POMŮŽE PROGRAM R PŘÍRODOVĚDCŮM?

JIŘÍ MAKOVIČKA

Ústav aplikací matematiky a výpočetní techniky, Přírodovědecká fakulta Univerzity Karlovy, Albertov 6, 128 43 Praha 2
jmakov@natur.cuni.cz

Došlo 27.6.16, přijato 13.7.16.

Klíčová slova: program R, lineární regrese, nelineární regrese, reakční kinetika, objekty v programu R

Úvod

Program R byl původně vytvořen pro statistiky. Patří mezi „free software“ (jak příjemné). Snadno se instaluje a obsluhuje a běží na všech třech hlavních platformách operačních systémů (UNIX, Windows, MacOS). Jeho požadavky na prostředky operačního systému (paměť a čas procesoru) jsou přitom relativně nízké. O historii a vývoji programu R se dozvíme v příslušné literatuře[1].

Velkou výhodou R je, že pro něj bylo vytvořeno velké množství knihoven (package), které pokrývají nejen požadavky soudobé statistiky, ale i dalších odvětví matematiky. V R můžeme pracovat uživatelsky, vykonáváním řádkových příkazů, lze zde však dělat i programy (skripty).

John Chambers, zakladatel jazyka S, z něhož R vychází, charakterizuje R takto[6]:

„To understand computations in R, two slogans are helpful:
– Everything that exists is an object.
– Everything that happens is a function call.“

Programovací jazyk R je objektový, i když trochu jiného pojetí, než klasické objektové programovací jazyky. Jeho základní datovou strukturou, z níž se odvíjejí struktury složitější, je vektor. Velice často používaná je funkce c (), kterou vektor definujeme. Např. příkazem *v<-c(2,5,9)* vytváříme vektor *v* o složkách 2, 5, 9 (přiřazovací příkaz zapisujeme v R symbolem <- nebo =). Cokoliv se v R děje, je realizováno jako volání funkcí. R je ve své podstatě postaven na funkcionálním programování. Vlastní funkce si samozřejmě může psát programátor sám.

Další důležitou charakteristikou R je, že je to jazyk interpretační[2], podobně jako klasický BASIC, či JavaScript. Kvantoví chemici nyní odhazují tento článek do koše a právem, pro jejich složité výpočty se hodí naopak rychlé jazyky kompilační. Kdo by tedy měl číst dále? Zejména ti přírodovědci, kteří chtějí snadno vyhodnotit svá naměřená data a to jak číselně, tak graficky. R obsahuje řadu knihoven a tedy i funkcí vytvořených za tímto účelem. Jeho povaha interpreta mu, jako programovacímu jazyku, zajišťuje velkou pružnost a programátorům dává možnost ji využít. Hotové R funkce, které voláme, pracují relativně rychle (hodně jich bylo naprogramováno, podobně jako většina jazyka R, v jazyce C). Z toho plyne důležitá rada, jak se v R chovat. Dříve, než si něco v tomto interpretu naprogramujeme, podívejme se, zda již v R neexistuje funkce, která nám vyhoví. Určitě ji velice často nalezneme.

Pramenů, z nichž lze čerpat při studiu R, lze na internetu najít mnoho. Počínaje krátkými příspěvky, blogy, tutoriály a konče volně stažitelnými učebnicemi.

Jakým způsobem představíme v tomto článku R přírodovědcům? Uvedeme dva příklady. První demonstruje lineární regresi včetně běžné regresní diagnostiky a druhý řeší soustavu diferenciálních rovnic pro následné reakce, potom generuje experimentální data této úlohy a vyhodnocuje je pomocí nelineární regrese.

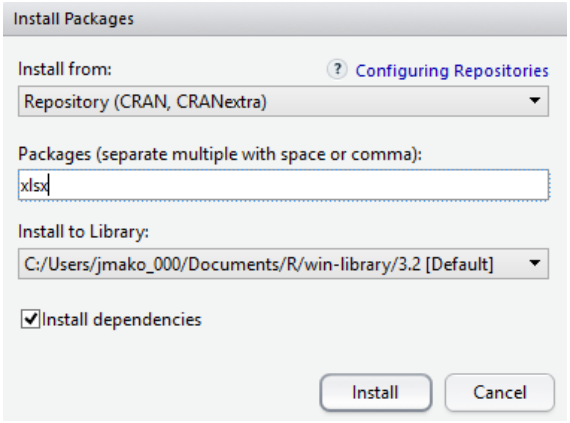
Zájemcům, kteří si chtějí R hned prakticky vyzkoušet, doporučujeme nainstalovat ještě RStudio (též free) a pracovat v něm. Je to kompletní integrované vývojové prostředí pro práci v R. Dává k dispozici řadu karet s užitečnými informacemi a pomůckami.

Název článku je tvořen řečnickou otázkou. Chce-li si však na ni odborný čtenář odpovědět, jistě po přečtení tohoto příspěvku odpoví kladně.

Lineární regrese v R

Příkazem *File/New File/RScript* zobrazíme v RStudiu novou kartu, na kterou budeme psát jednotlivé příkazy a postupně je kliknutím na tlačítko *Run* karty necháme vykonávat. Příkazy bychom mohli zapisovat a provádět přímo v okně *Console,* ale skript na kartě budeme moci uložit a vykonat kdykoli později znovu.

Data, kterými chceme proložit regresní přímku, máme v excelovském sešitu. Načíst je není v R problém. Na kartě *Packages RStudia* klikneme na *Install*. Objeví se okno *Install Packages* (obr. 1). Do pole *Packages* vpíšeme název knihovny *xlsx* a klikneme na *Install.* Po instalaci se knihovna objeví na kartě *Packages.* Aktivací zaškrtávacího okénka před jejím názvem nahrajeme knihovnu k použití. V okně *Console* se objeví vykonaný příkaz *library(xlsx)*, který možno též zapsat do skriptu viz obr. 2, řádek 2 okna se skriptem. Na příkazy tohoto skriptu se budeme v dalším odkazovat jen číslem řádku, číslo obrázku vynecháme.



Obr. 1. **Okno instalace knihoven v RStudiu**

```
1  # regrese primkou
2  library(xlsx)
3  datalk <- read.xlsx("C:/Users/Lenovo/OneDrive/ExcelRegrese/DataXY.xlsx", 4)
4  regprim<-lm(Y~X, data = datalk)
5  summary(regprim)
6  confint(regprim)
7  plot(Y~X,data = datalk, tck=0.03)
8  abline(regprim)
9  par(mfrow=c(2,2))
10 plot(regprim)
11 par(mfrow=c(1,1))
12 shapiro.test(rstandard(regprim))
13 regpar<-lm(Y~X + I(X^2), data = datalk)
14 summary(regpar)
15 confint(regpar)
```

Obr. 2. **Skript s příkazy pro lineární regresi**

Funkcí *read.xlsx()* na řádku 3 načteme z excelovského sešitu naše data. První argument je cesta k sešitu, druhý je číslo listu. Načtená data jsme přiřazovacím příkazem uložili do proměnné *datalk* typu *data.frame*. Programovací jazyk R není typový jazyk. Datový typ proměnné zde nedefinujeme deklarací, ale určí jej program R na základě toho, jaký objekt proměnná označuje. Funkce *read.xlsx()* navrací běžnou tabulku dat, které v R odpovídá datový typ *data.frame*. Naše tabulka obsahuje dvě složky (v Excelu sloupce) nazvané X a Y s hodnotami nezávisle a závisle proměnné. Lze se na ně odvolat zápisem *datalk$X* či *datalk$Y*. Počet pozorování je celkem 43, což zjistíme vykonáním příkazu *length(datalk$X)*. Tabulku *datalk* si lze v RStudiu zobrazit příkazem *View(datalk)*.

Pod pojmem lineární regrese budeme obecně rozumět proklad experimentálních dat regresní funkcí $y = b_0 + b_1 f_1(x) + b_2 f_2(x) + ... + b_m f_m(x)$ lineární v hledaných parametrech b_j, $j = 0, 1, ..., m$. Proklad samozřejmě uskutečníme metodou nejmenších čtverců. V R realizujeme lineární regresi funkcí *lm()*(linear model), která navrací objekt obsahující konkrétní regresní model. Jako první argument *lm()* musíme udat regresní funkci. Syntax tohoto zápisu se v R nazývá formule a řídí se zvláštními pravidly[3]. V případě naší regresní přímky má tvar Y~X (řádek 4). Druhý argument funkce *lm()* specifikuje data – dříve načtená tabulka *datalk*. Regresní model bude uložen v objektu *regprim*. Jeho statistické charakteristiky zobrazíme příkazem *summary(regprim)*, viz řádek 5, výsledek příkazu je na obr. 3.

```
> summary(regprim)

Call:
lm(formula = Y ~ X, data = datalk)

Residuals:
     Min       1Q   Median       3Q      Max
-1.41288 -0.40519  0.01021  0.39097  1.37173

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   4.8821     0.4299   11.36 3.08e-14 ***
X             0.2615     0.0352    7.43 4.10e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7443 on 41 degrees of freedom
Multiple R-squared:  0.5738,    Adjusted R-squared:  0.5634
F-statistic: 55.21 on 1 and 41 DF,  p-value: 4.098e-09
```

Obr. 3. **Statistické charakteristiky regresní přímky**

Z charakteristik modelu je nejdůležitější úsek *Coefficients*, obsahující ve sloupci *Estimate* odhady koeficientů regresní přímky. Hodnotu proměnné Y můžeme tedy predikovat z rovnice Y=4,8821+0,2615*X. Ve sloupci *Pr(>|t|)* jsou příslušné p-hodnoty, které v našem případě svědčí (se zanedbatelným rizikem) ve prospěch zamítnutí hypotézy, která tvrdí, že hodnoty koeficientů regresní funkce jsou nulové. Ze základních charakteristik je dále důležitá standardní chyba reziduí[4] (Residual standard error) a koeficienty determinace[4] (Multiple R-squared, Adjusted R-squared). Celkový F-test modelu s příslušnou p-hodnotou (poslední řádek na obr. 3) svědčí pro zamítnutí hypotézy o tom, že všechny koeficienty regresního modelu, kromě absolutního členu, jsou nulové.

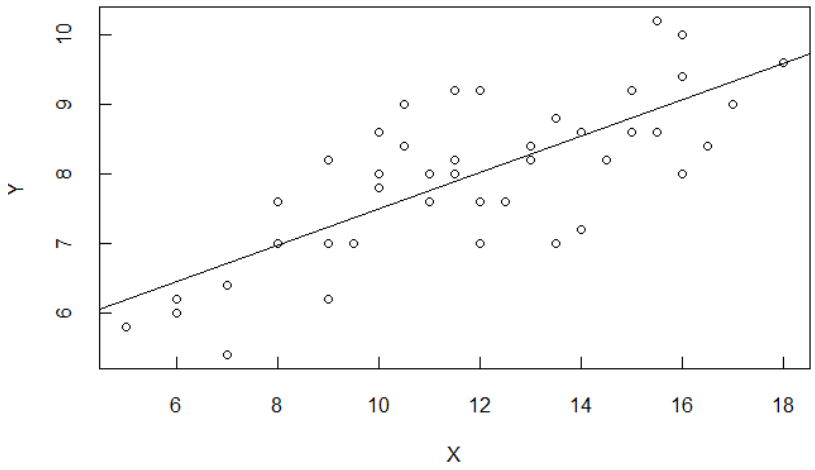
Z údajů uvedených na obr. 3 lze vypočíst intervaly spolehlivosti regresních koeficientů. Není to však třeba. Příkaz *confint(regprim)* (řádek 6) to provede za nás (obr. 4).

```
> confint(regprim)
                2.5 %    97.5 %
(Intercept) 4.0139866 5.7502729
X           0.1904505 0.3326233
```

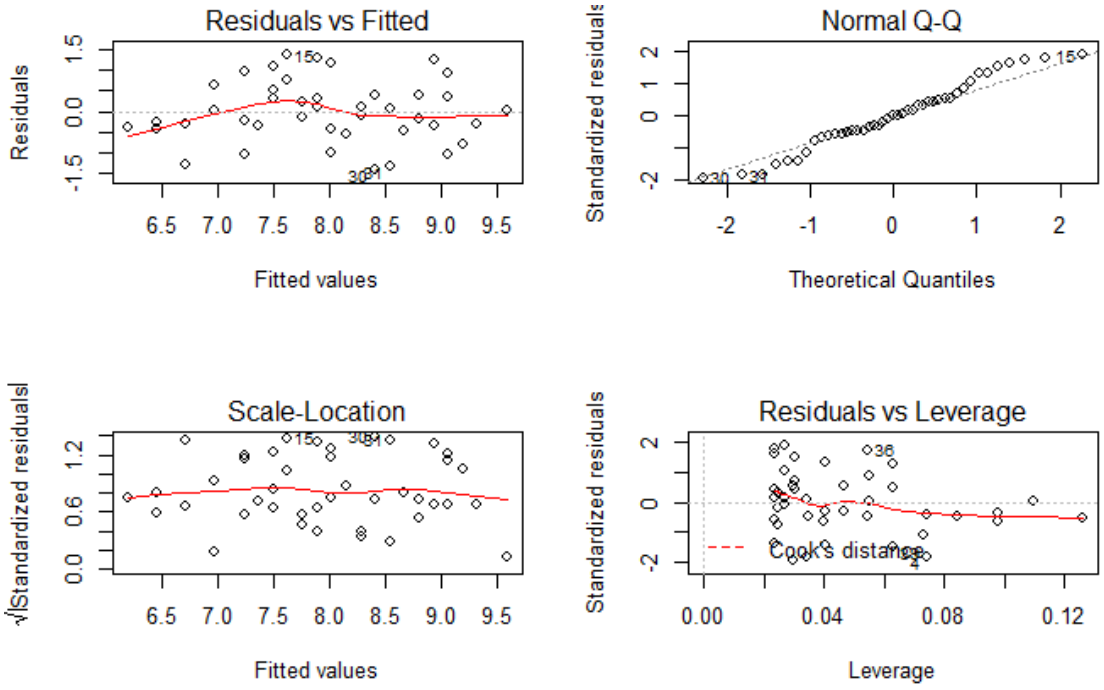
Obr. 4. **Intervaly spolehlivosti parametrů regrese**

Jak zobrazíme graf naměřených bodů a regresní přímky. Zavoláme funkci *plot*() s prvním argumentem určujícím vztah závisle proměnná – nezávisle proměnná, druhý argument specifikuje data (řádek 7). Argument *tck=0,03* určuje délku dílků na osách. Tato funkce vykreslí na kartě *Plots* RStudia graf bodů, které proložíme regresní přímkou příkazem *abline(regprim)* (řádek 8 a obr. 5).

R má četné nástroje diagnostikující regresní model. Jednoduchou grafickou diagnostiku nám poskytuje volání funkce *plot(regprim)*, tedy s hodnotou argumentu rovnou objektu obsahujícímu regresní model (řádek 10). Příkaz vytvoří celkem čtyři grafy. Voláním funkce *par(mfrow=c(2,2))* na řádku 9 zajistíme, že se zobrazí všechny v jednom okně, viz obr. 6 (okno pro tvorbu grafů se nyní skládá ze dvou sloupců a dvou řádků, proto *mfrow=c(2,2)*, zpětné zobrazení jednoho grafu v okně zajistíme příkazem *par (mfrow=c(1,1))* – řádek 11).



Obr. 5. **Regresní graf modelu**



Obr. 6. **Regresní diagnostika provedená funkcí plot()**

První graf na obr. 6 ukazuje závislost reziduí na predikovaných hodnotách Y (predikce provedena pro zadané hodnoty X modelu). Pokud body grafu netvoří „náhodný mrak“ symetricky rozmístěný kolem přímky Residuals=0, pak zvolený regresní model není považován za vhodný. V našem případě však tato situace nenastala. Plnou čarou je na grafu dále zvýrazněn trend reziduí, který také nevypovídá o nevhodnosti modelu.

Graf vpravo nahoře testuje normalitu reziduí. Pokud se body grafu neodchylují významně od přímky na grafu, lze normalitu předpokládat. Normalitu reziduí nepopírá ani Shapirův-Wilkův test normality standardizovaných reziduí – řádek 12, výsledek pak na obr. 7. Za povšimnutí stojí, jak snadno jsme standardizovaná rezidua modelu získali. Stačilo použít funkci *rstandard(regprim)*.

```
> shapiro.test(rstandard(regprim))

        Shapiro-Wilk normality test

data:  rstandard(regprim)
W = 0.97177, p-value = 0.3631
```

Obr. 7. **Shapirův-Wilkův test normality standardizovaných reziduí. S výslednou 36% p-hodnotou lze normalitu předpokládat (ne však prokázat)**

Graf na obr. 6 vlevo dole nám může odhalit nekonstantnost rozptylu reziduí, jejich heteroskedasticitu. Např. v případě růstu rozptylu reziduí s rostoucí hodnotou nezávisle proměnné, by graf měl trychtýřovitý tvar s širší stranou vpravo. V našem případě není tato situace indikována.

Graf vpravo dole na témže obrázku slouží k detekci nestandardních hodnot v našich datech. Jde zejména o tzv. vlivné body. Ani v tomto případě nebudeme zasahovat. Blíže se o této relativně komplikované problematice dočteme v literatuře[4].

Ukažme dále, jak budou vypadat statistické charakteristiky, proložíme-li našimi daty regresní parabolu druhého stupně. Na řádku 13 jsme proto vytvořili regresní model *regpar*. Funkce *I()* ve formuli modelu zajišťuje, že operátory v ní obsažené se interpretují ve svém původním algebraickém významu a ne ve významu formule modelu. V našem případě jde o umocňování. Výpis sumárních charakteristik naší regresní paraboly (obr. 8) ale již neposkytuje příznivou p-hodnotu pro koeficient u kvadratického členu. Totéž samozřejmě potvrzuje i příslušný interval spolehlivosti (viz řádek 15 a obr. 9) na základě kterého nelze na hladině významnosti testu 5 % zamítnout hypotézu, která specifikuje, že hodnota koeficientu u x^2 je nulová.

```
> regpar<-lm(Y~X + I(X^2), data = datalk)
> summary(regpar)

Call:
lm(formula = Y ~ X + I(X^2), data = datalk)

Residuals:
     Min       1Q   Median       3Q      Max
-1.51252 -0.38306 -0.00653  0.44020  1.33751

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.05435    1.26846   2.408   0.0207 *
X            0.60404    0.22675   2.664   0.0111 *
I(X^2)      -0.01480    0.00968  -1.528   0.1343
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7325 on 40 degrees of freedom
Multiple R-squared:  0.5974,    Adjusted R-squared:  0.5772
F-statistic: 29.67 on 2 and 40 DF,  p-value: 1.255e-08
```

Obr. 8. **Proklad regresní parabolou**

```
> confint(regpar)
                2.5 %      97.5 %
(Intercept)  0.49069730 5.618009042
X            0.14576504 1.062320878
I(X^2)      -0.03435889 0.004768608
```

Obr. 9. **Intervaly spolehlivosti pro koeficienty kvadratického modelu.**

Řešení diferenciálních rovnic

Ukažme si nyní, jak se v R mohou řešit diferenciální rovnice reakční kinetiky. Poslouží nám k tomu funkce *integrateODE*() z knihovny *mosaic*. Dají se s ní řešit soustavy obyčejných diferenciálních rovnic prvního řádu rozřešených vzhledem k derivaci. Derivaci závisle proměnné *y* označujeme symbolem *dy*, obor hodnot nezávisle proměnné určujeme argumentem *tdur*. Kinetickou rovnici pro reakci prvního řádu, při které dochází k rozpadu látky A, zapíšeme symbolicky dA~−k*A, kde k je rychlostní kon-

```
Source on Save                                                    Run
 1 # nasledne reakce
 2 library("mosaic", lib.loc="~/R/win-library/3.2")
 3 res=integrateODE(dA ~ -k1*A, dB ~ k1*A-k2*B, dC ~ k2*B,
 4                  k1=1,k2=0.5, A=10,B=0,C=0, tdur = 15)
 5 t<-seq(0,15,0.1)
 6 plot(t,res$A(t),type="l",lwd=2,ylab="koncentrace",xlab="čas",tck=0.03)
 7 lines(t,res$B(t),type="l",lwd=2,tck=0.03)
 8 lines(t,res$C(t),type="l",lwd=2,tck=0.03)
 9 text(0.6,8,"A"); text(5,2.2,"B"); text(10,9.5,"C")
10
11 fnaslr<-function(a0,k1,k2,t){
12   return(a0*k1/(k2-k1)*(exp(-k1*t)-exp(-k2*t)))
13 }
14
15 t<-seq(0,14,0.1)
16 ch<-rnorm(length(t),0,0.1)
17 a0<-10;k1<-1;k2<-0.5
18 B<-fnaslr(a0,k1,k2,t)+ch
19 plot(t,B,type = "p")
20 k1s<-0.8 ; k2s<-0.7
21 naslr<-nls(B ~ fnaslr(a0,k1,k2,t),start = list(k1=k1s,k2=k2s))
22 summary(naslr)
23 cf<-coef(naslr)
24 confint(naslr)
25 yn<-fnaslr(a0,cf[1],cf[2],t)
26 lines(t,yn, lwd=2)
```

Obr. 10. **Následné reakce – skript**

stanta. Sledujeme-li reakci po pět sekund, pak argument *tdur*=5.

Budeme řešit systém dvou následných reakcí A -> B-> C s rychlostními konstantami po řadě k1=1 a k2=0,5. Počáteční koncentrace látek A, B, C jsou A=10,B=0,C=0 (vše v kompatibilních jednotkách), reakce sledujeme 15 sekund, tedy argument *tdur* = 15. Příslušné volání funkce *integrateODE*() je na řádcích 3 a 4 obr. 10. V dalším budeme opět číslo obrázku vynechávat a odkazovat se do skriptu jen číslem řádku. Zápis této soustavy je uveden v prvních třech argumentech funkce *integrateODE*(), dalšími argumenty přiřazujeme hodnoty parametrům soustavy. Výsledkem integrace je objekt pojmenovaný *res*, obsahující tři funkce označené A, B a C, jejichž voláním získáme řešení. Přístup k nim specifikujeme znakem dolar. Tedy chceme-li určit koncentraci látky A v čase *t*, voláme první z nich zápisem *res$A(t)*.

Výsledky zobrazíme v časovém intervalu <0;15> po kroku 0,1. Na řádku 5 tvoříme funkcí *seq()* příslušnou posloupnost hodnot nezávisle proměnné. Na řádku 6 potom zobrazíme funkcí *plot*() úbytek látky A. Hodnota argumentu *type*="l" značí, že body funkčních hodnot spojíme úsečkami, *lwd*=2 nastaví šířku čáry na hodnotu 2, *ylab* a *xlab* udávají popis os. Další dvě funkce *res$B* a *res$C* udávající koncentrace látek B a C chceme zobrazit ve stejném grafu. Opětovným voláním funkce *plot()* bychom ale vytvořili nové okno, tedy nový graf. Proto nyní použijeme funkci *lines(),* která závislost látek B a C na čase vykreslí v původním grafu. Příslušné volání je uvedeno na řádcích 7 a 8. Na řádku 9 potom připojíme ke grafu (viz obr. 11) popis křivek.

Nelineární regrese

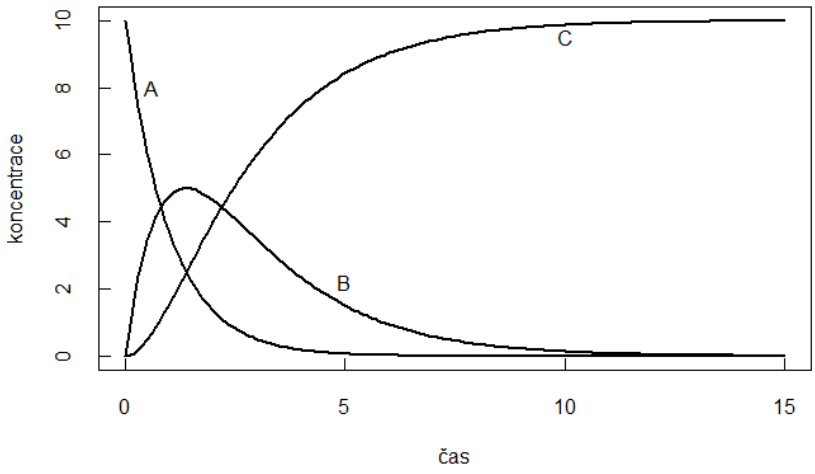
Podívejme se nyní na problematiku následných reakcí z trochu jiného pohledu. Z dat udávajících závislost koncentrace látky B na čase vypočtěme rychlostní konstanty k1 a k2. Tato závislost je dána rovnicí[7]:

$$B = A_0 \cdot \frac{k1}{k2 - k1} \cdot \left(e^{-k1.t} - e^{-k2.t} \right)$$

kde A_0 je počáteční koncentrace látky A, k1 a k2 jsou rychlostní konstanty a *t* je čas. Pro výpočet koncentrace látky B si vytvoříme funkci nazvanou *fnaslr* (řádky 11 až 13), navracející, pro zadané hodnoty vstupních parametrů, hodnotu koncentrace látky B.

Příslušná data si však budeme generovat. Vytvořme vektor *t* uchovávající časové hodnoty, v nichž známe koncentraci látky B. Funkcí *seq()* vytvoříme proto aritmetickou posloupnost jdoucí od nuly do 14 po kroku 0,1 (viz řádek 15). Posloupnost uložíme do vektoru *t*. Experimentální chyby budeme generovat pomocí vektoru *ch* (řádek 16), který naplníme náhodnými daty vzniklými z normálního rozdělení o průměru nula a směrodatné odchylce 0,1. Jejich počet je roven počtu složek vektoru *t*. Ke generaci chyb jsme použili funkci *rnorm()*. Na řádku 17 určíme počáteční koncentraci látky A a hodnoty rychlostních konstant k1 a k2. Koncentrace látky B, jejichž hodnoty jsou zatíženy experimentální chybou *ch,* generujeme na řádku 18 sečtením volání funkce *fnaslr* a vektoru chyb *ch*. Vzniklá data zobrazíme funkcí *plot()* jako experimentální body (argument *type="p")* v grafu (řádek 19). Tento graf je na obr. 13.

Nyní se vžijeme do role experimentátora, který naše data získal měřením a jejich vyhodnocením chce obdržet rychlostní konstanty k1 a k2. Hodnota koncentrace látky B však není lineární funkcí parametrů k1 a k2, proto nutno k vyhodnocení použít techniky nelineární regrese. Regresní funkcí je *fnaslr*. Potřebujeme však znát počáteční odhady hledaných parametrů k1 a k2. Uložili jsme je na řádku 20 po řadě do proměnných k1s a k2s. Vhodným nástrojem pro nelineární regresi je v R funkce *nls*() (nonlinear least squares). Voláme ji na řádku 21 a výsledný objekt, který navrací, pojmenujeme *naslr*. Funkce *nls*() má jako první argument formuli s prokládanou regresní funkcí, jako dru-



Obr. 11. **Následné reakce - graf**

hý argument je seznam (datová struktura list jazyka R) obsahující počáteční hodnoty parametrů. Funkcí *summary (naslr)* (řádek 22) dostaneme souhrnné informace o nelineární regresi (obr. 12). Nyní jen stačí získat funkcí *coef()* z objektu *naslr* do vektoru *cf* hledané hodnoty regresních koeficientů k1 a k2 (řádek 23) a funkcí *confint*()z objektu *naslr* intervaly spolehlivosti (řádek 24) pro tyto koeficienty. Na řádcích 25 a 26 potom proložíme našimi body graf regresní funkce (obr. 13).

```
> summary(naslr)

Formula: B ~ fnaslr(a0, k1, k2, t)

Parameters:
   Estimate Std. Error t value Pr(>|t|)
k1 0.994632   0.009530   104.4   <2e-16 ***
k2 0.498622   0.002391   208.6   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.09977 on 139 degrees of freedom

Number of iterations to convergence: 5
Achieved convergence tolerance: 1.637e-06

> cf<-coef(naslr)
> confint(naslr)
Waiting for profiling to be done...
        2.5%     97.5%
k1 0.9758360 1.0138567
k2 0.4939165 0.5033986
```

Obr. 12. **Výsledné informace o nelineární regresi**

Zdržme se ještě u argumentů funkce *fnaslr* na řádku 25. Posledním argumentem je vektor *t*, který jsme získali na řádku 15. Předchozí tři argumenty ale nejsou skaláry. Tento pojem jazyk R totiž nezná. Pracuje jen s vektory. Proto to jsou vektory o jedné složce. Argumenty funkce *fnaslr* jsou tedy vektory o nestejném počtu složek. Jak to R řeší? Nastupuje tzv. recyklace (recycling), kterou R prodlouží kratší vektory opakováním jejich složek tak, aby jejich výsledná délka (tj. počet složek) byla rovna vektoru nejdelšímu. Potom se provede žádaná operace. V našem případě se počet složek každého z prvních tří argumentů zvětší (opakováním první složky) na počet složek vektoru *t*. Další příklad recyklace je na obr. 14. K vektoru a1 o pěti složkách přičítáme vektor a2 o dvou složkách. Recyklací vznikne z a2 vektor také o pěti složkách (4,5,4,5,4) a pak dojde k sečtení tohoto vektoru a vektoru a1. Použití recyklace dává R najevo varovnou zprávou.

```
> a1<-c(1,2,3,4,5)
> a2<-c(4,5)
> a1
[1] 1 2 3 4 5
> a2
[1] 4 5
> a3<-a1+a2
Warning message:
In a1 + a2 :
  longer object length is not a multiple of shorter object length
> a3
[1] 5 7 7 9 9
```

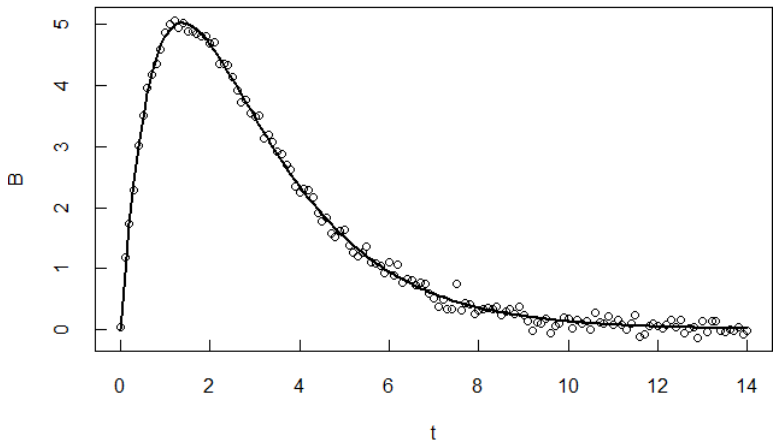
Obr. 14. **Příklad recyklace**

Objektový přístup jazyka R

Objektově orientované funkce jazyka R (generické funkce) pracují tak, že zacházejí s objekty určitých typů jim příslušným způsobem. Jinými slovy, generická funkce akceptuje celou řadu argumentů různých typů a každý tento argument zpracovává jiným, jeho typu odpovídajícím způsobem. Ukažme si to na příkladu funkce *plot()*. Zadáme-li jí jako první dva argumenty vektory x (nezávisle proměnná) a y (závisle proměnná) obsahující příslušné funkční hodnoty (*plot(x,y)*), nebo jako první argument formuli y~x (*plot(y~x)*), nakreslí nám graf funkce. Voláme-li ji s argumentem typu *lm* (získaným funkcí *lm()* pro lineární regresi), poskytuje nám standardně čtyři grafy regresní diagnostiky, je-li její argument typu nls (získaný funkcí *nls()* pro nelineární regresi), poskytne jeden reziduální graf.

Funkce *summary()* navrací souhrnné informace o datech v objektu zadaném jako argument. Tyto informace se opět liší v závislosti na typu tohoto objektu.

Podobně funkce coef() či confint() poskytují koeficienty či intervaly spolehlivosti, které extrahují z objektů



Obr. 13. **Graf regresní funkce – koncentrace látky B v závislosti na čase *t***

definovaných typů zadaných jako argumenty těchto funkcí.

Přesněji řečeno, voláme-li generickou funkci R s argumentem určitého typu (např. funkci *plot()* s argumentem typu *lm*) funkce zjistí, zda tento typ obsluhuje speciální činností, zakódovanou v jiné funkci. Pokud ano, zavolá tuto funkci, která obsluhu volání provede (funkce *plot()* volá *plot.lm()*). Pokud ne, probíhá obsluha standardním způsobem (např. voláním *plot.default()*).

Objektové vlastnosti jazyka R jsme popsali velice zjednodušeně, což je ale pro naše účely, seznámení s jazykem, dostačující. Přesnější informace lze nalézt v citované litratuře[5].

LITERATURA

1. http://www.unt.edu/rss/R_Programming_Notes.pdf, staženo 10.7.2016
2. https://en.wikipedia.org/wiki/Interpreted_language, staženo 10.7.2016
3. http://www.montefiore.ulg.ac.be/~kvansteen/GBIO0009-1/ac20092010/Class8/Using%20R%20for%20linear%20regression.pdf, staženo 10.7.2016
4. Meloun Milan, Militký Jiří: Interaktivní statistická analýza dat. Karolinum 2012.
5. http://www.cyclismo.org/tutorial/R/objectOriented.html, staženo 10.7.2016
6. http://adv-r.had.co.nz, staženo 10.7.2016
7. https://cs.wikipedia.org/wiki/Rychlostn%C3%AD_rovnice, staženo 12.7.2016.

J. Makovička *(Institute of Applied Mathematics and Information Technologies, Faculty of Science, Charles University:* **Will the Program R Help the Natural Scientists?**

Examples from chemical practice (linear and nonlinear regression, solving of differential equations for reaction kinetics) were chosen so as to show, how can the program R help natural scientists.