Základy programovacieho jazyka EViews a ich aplikácia na analýzy, prognózy a rýchle odhady vývoja makroekonomických ukazovateľov
1. časť

Ing. Miroslav Kľúčik

infostat® Inštitút informatiky a štatistiky
DÁVAME INFORMÁCIÁM ZMYSEL

INFOSTAT – Inštitút informatiky a štatistiky

Základy programovacieho jazyka EViews a ich aplikácia na analýzy, prognózy a rýchle odhady vývoja makroekonomických ukazovateľov *1. časť*

Ing. Miroslav Kľúčik

Edícia: Dokumenty
Bratislava január 2010

Príručka *Základy programovacieho jazyka EViews a ich aplikácia na analýzy, prognózy a rýchle odhady vývoja makroekonomických ukazovateľov (1. časť)* vznikla ako súčasť riešenia výskumného projektu *"Modelový aparát na rýchle odhady vývoja makroekonomických ukazovateľov slovenskej ekonomiky"*, ktorý INFOSTAT začal riešiť v roku 2008 s pokračovaním v rokoch 2009 a 2010 s podporou Agentúry na podporu výskumu a vývoja. Príručka je určená pre užívateľov, ktorí využívajú ekonometrické nástroje na analýzu a prognózu dát. Prvá časť príručky pomôže výskumníkom získať základé znalosti spojené s prácou so súbormi, objektami pracovného hárku, spúšťaním programu a odhadom jednoduchým regresných rovníc a zároveň využijú praktické znalosti pri dokumentácii vlastnej výskumnej práce.

Práca neprešla jazykovou úpravou.

Obsah

Úvod

Príručka *Základy programovacieho jazyka EViews a ich aplikácia na analýzy, prognózy a rýchle odhady vývoja makroekonomických ukazovateľov (1. časť)* vznikla ako súčasť riešenia výskumného projektu *"Modelový aparát na rýchle odhady vývoja makroekonomických ukazovateľov slovenskej ekonomiky"*, ktorú INFOSTAT začal riešiť v roku 2008 s pokračovaním v rokoch 2009 a 2010 s podporou Agentúry na podporu výskumu a vývoja[1].

EViews je softvérová aplikácia orientovaná na ekonometrickú analýzu. EViews patrí na svete medzi najvyužívanejší ekonometrický softvér. Jeho najznámejšími konkurentmi sú SAS, GAUSS, Stata a iné. Medzi známymi užívateľmi EViews-u na Slovensku je okrem INFOSTATu napríklad i Ministerstvo Financií SR a Národná Banka Slovenska.

EViews je určený predovšetkým pre ekonómov zaoberajúcich sa napríklad analýzou dát, finančnou analýzou, makroekonomickými prognózami, simuláciami či nákladovou analýzou. Slúžiť môže však výskumníkom vo všetkých sférach výskumu, kde využívajú sofistikovanú analýzu dát, regresie a prognózy. Okrem štandardného prostredia softvéru pôsobiaceho v operačnom systéme Microsoft Windows ponúka EViews balík programovacieho jazyka (command language), ktorý pomôže výskumníkom pri dokumentácii ich výskumnej práce. Výhody programovacieho jazyka všeobecne súvisia hlavne s automatizáciou analytických procesov, následným šetrením voľného času a zároveň financií. Tieto výhody sa dajú využiť hlavne pri makroekonomických analýzach, ktoré sú spojené s častou aktualizáciou údajov. Softvér takto umožňuje spätné spustenie programu na zmenených (aktualizovaných) dátach. Automatizáciou práce s databázou má výskumník k dispozícii viac času na odbornú analýzu, hlavne ak pracuje s veľkým počtom časových radov, resp. s väčším počtom výpočtových úkonov. Programovací jazyk EViews-u je veľmi jednoduchý a nemal by robiť vážnejšie problémy nikomu, kto pracuje v oblasti výskumu. Veľkou výhodou je kompatibilita EViews-u s databázami Microsoft Excelu. Softvér umožňuje analýzy v tabuľkovej i grafickej forme na dostatočnej úrovni (porovnateľné s Microsoft Excel). V príručke sú vysvetlené niektoré príkazy programovacieho jazyka verzie EViews-u 5.1, avšak tieto sú kompatibilné s predošlými i budúcimi verziami EViews-u. Momentálne najnovšou verziou EViews-u je verzia 6.0 vyvinutá firmou Quantitative Micro Software.

Príručka je určená pre výskumníkov a záujemcov, ktorí využívajú ekonometrické nástroje na analýzu a prognózu dát. Slúži ako pomôcka pri výučbe programovacieho jazyka EViews. Používateľovi stačí základné vzdelanie zo štatistiky, avšak potrebná je angličtina, vzhľadom na anglické menu. Je určená najmä začiatočníkom užívania softvéru EViews. Výučba programovacieho jazyka si nevyžaduje hlbokú znalosť samotného programu. Používatelia môžu počas výučby prostredníctvom tejto príručky získavať základy samotného softvéru a zároveň sa širšie oboznamovať so štatistickými nástrojmi, hlavne prostredníctvom pomoci (Help), v ktorej sú jednoduchým spôsobom vysvetlené všetky používané príkazy a analytické nástroje.

Štýl popisu jednotlivých programovacích prvkov je orientovaný na jednoduchosť, aby užívateľ pochopil hlavný zmysel. Ďalej aj na detailnosť, aby nenarážal na zbytočné prekážky, s ktorými sa často programátor v praxi stretáva. Používateľom sa odporúča vyskúšať všetky príklady uvedené v tejto príručke, pričom platí zásada, že je vždy lepšie vyskúšať navrhnuté príklady na vlastnej problematike riešenej používateľom. Prístupom pokus-omyl sa používateľ ľahšie naučí i ľahšie pochopí písaný text príručky.

[1] **Táto práca bola podporovaná Agentúrou na podporu výskumu a vývoja na základe zmluvy č. APVV-0032-07.**

Úvodná kapitola tejto príručky je zameraná na základnú prácu so súbormi a dátami v EViews-e, teda so základným súborom, databázami a programom. V 2. kapitole sú vysvetlené hlavné prvky práce s časovým rozpätím súborov spolu s predvolenými nastaveniami adresárov. S databázami súvisí aj vysvetlenie preberania dát z Excel-u. Následne sú ozrejmené základné nástroje na analýzu a transformáciu časových radov. To zahŕňa prácu s časovými radmi, skupinami časových radov, prácu s grafmi a ďalšími prvkami často využívanými pri kódovaní, akými sú programovacie premenné, komentáre atď. Tretia kapitola obsahuje informácie o kontrole spúšťania programu prostredníctvom cyklov. Posledná kapitola obsahuje opis tvorby jednoduchých regresných rovníc, používanie umelých vysvetľujúcich premenných a popis ďalších vlastností regresných odhadov (koeficienty, metóda najmenších štvorcov, rezíduá).

Užívateľ programovacieho jazyka v prostredí EViews tak postupne získa dostatok znalostí z oblasti ekonometrickej analýzy orientovanej na makroekonómiu. Používatelia EViews-u, ktorí využijú možnosti programovacieho jazyka hneď pri spoznávaní EViews-u sa už tejto vymoženosti nevzdávajú, naopak, ich záujem sa určite prehĺbi. Na druhej strane, ak sú užívatelia už zvyknutí pracovať len s Menu, prechod na programovací jazyk je ťažký, ale v prípade širšieho používania tohto softvéru nutný. Ak dochádza užívateľ do styku s týmto softvérom len zriedkavo, učiť sa programovací jazyk nie je potrebné.

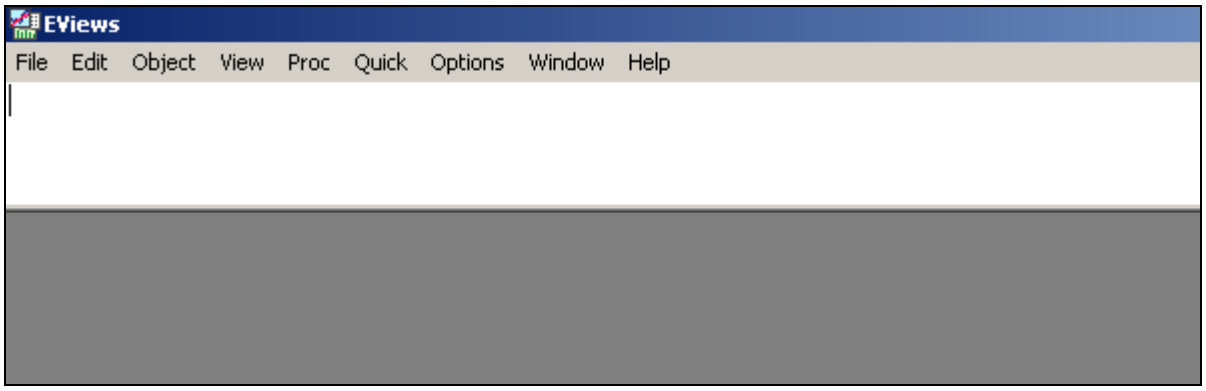
Výhodou tejto príručky je, že sa zaoberá skutočnými údajmi a postupmi, ktoré sa využívajú na analýzy, prognózy a rýchle odhady makroekonomických ukazovateľov slovenskej ekonomiky. Takto si ľahšie nájde vzťah k tejto príručke používateľ, ktorý pracuje v oblasti makroekonomického výskumu. Užívatelia, ku ktorým sa dostane táto príručka či už v tlačenej či elektronickej forme, môžu využiť možnosť ovplyvniť autora konštruktívnymi pripomienkami a návrhmi pre prípadné ďalšie verzie či časti príručky. Za takúto formu pomoci autor vopred ďakuje.

1. Základná práca s pracovnými hárkami a databázami

Základnou informáciou po inštalácii a spustení EViews-u je, s akými pracovnými súbormi pracuje. S aplikáciou sú asociované súbory s koncovkou *.wf1 (štandardný pracovný hárok), *.prg (program na spustenie) a databázové súbory (*.edb, *.e0, *.e1a, *.e1b). Po dvojitom kliknutí v ľubovoľnom prehliadači sa otvorí v aplikácii EViews iba súbor s koncovkou *.wf1 a databázový súbor s koncovkou *.edb. Avšak pri ostatných databázových súboroch (*.e0, *.e1a, *.e1b) sa vypíše chybové hlásenie. Pri spustení programového súboru (*.prg) sa priamo spustí program, čo často vedie tiež k chybovému hláseniu. Preto odporúčame na začiatku otvárať aplikáciu EViews priamo prostredníctvom menu, teda prázdnu bez pracovného objektu.

Po otvorení aplikácie EViews prostredníctvom menu v operačnom systéme, či priamo z Desktopu, sa objaví úvodné okno, kde v hornej časti pod Menu je biela oblasť – viď kurzor (obr. 1.1), ktorá slúži na vpisovanie príkazov pomocou programovacieho jazyka. Ostatná oblasť je prednastavená v šedej farbe a neplní žiadny účel. Pred tým, než si vytvoríme nový súbor (cez menu to je prostredníctvom File/New/Workfile...) je dôležité poznať prednastavený adresár (default – aktuálny, s ktorým pracujeme), v ktorom chceme daný súbor vytvoriť. Túto informáciu nájdeme v pravom dolnom rohu (Path = c:\...) – obr. 1.2. Lišta ďalej obsahuje predvolenú (aktuálnu) databázu (DB =...) a meno súboru, s ktorým práve pracujeme (WF =...), táto informácia je dôležitá ak máme otvorených viacero súborov v okne naraz.

Obr. 1.1 Pracovná plocha EViews-u – vrchná časť



Obr. 1.2 Pracovná plocha EViews-u – dolná časť

Biela oblasť sa teda využíva výhradne na vpisovanie príkazov pomocou príkazového jazyka EViews-u. Pri písaní príkazov do bieleho príkazového panelu (lišty) nerozlišujeme veľké a malé písmená. Rovnako nezáleží na medzerách medzi príkazom a špecifikáciou príkazu (musí byť minimálne jedna), rovnako ako na medzere medzi operátormi vo výpočtoch: 5+2 alebo 5 + 2 (nemusí byť žiadna medzera). Vykonanie príkazu sa uskutočňuje stlačením klávesy Enter. Môžete si to overiť samostatne napísaním akéhokoľvek textu a stlačením Enter-u. Objaví sa chybové hlásenie, lebo po prvé, nemáme ešte otvorený žiadny súbor, a po druhé, je nepravdepodobné, aby sme so šťastím narazili na ešte neznámy príkaz. Ak píšeme príkazy do daného panelu, po zavretí EViews-ovského okna sa uložia do súboru command.log (v predvolenom adresári). Pri väčších úlohách je lepšie používať pri písaní príkazov osobitný súbor – tzv. Program (File/New/Program...), ktorý je možné spustiť stlačením Run, a na rozdiel od príkazového riadku spustí všetky príkazy v danom okne naraz, a nemusíte spúšťať každý riadok osobitne prostredníctvom klávesy Enter.

Ak chceme zmeniť prednastavený adresár, kam sa nám budú ukladať súbory s ktorými pracujeme, použijeme príkaz (všeobecný zápis príkazu) [2]:

```
cd cesta
```

Príkaz, ktorým by sme nastavili adresár ktorý je vidieť na obr. 1.1, by vyzeral nasledovne:

```
cd "c:\documents and settings\klucik.klucik\my documents\flash_estim\flash_estim\
maj09"
```

Cesta daného adresára sa môže zadávať bez úvodzoviek len v tom prípade, ak neobsahuje medzeru – čo v našom prípade obsahuje: teda „*documents and settings*". Nastavený aktuálny adresár treba vždy sledovať a je obsahom každého väčšieho programu. Napríklad ak vyberáme údaje z Excel-ovského súboru, program EViews-u hľadá daný Excel-ovský súbor v adresári, ktorý je nastavený ako aktuálny. Rovnako, ak ukladáme súbory, ktoré sú otvorené v jednom okne, všetky sú uložené do prednastaveného adresára. Často sa stáva, že sme nepozorní a potom nemôžeme nájsť súbor, s ktorým sme naposledy pracovali, prípadne máme súbor s rovnakým názvom vo viacerých adresároch.

Existujú štyri hlavné druhy súborov, s ktorými budeme v EViews-e pracovať:

wf1 – Workfile – základný súbor – pracovný súbor

edb – Database – databáza – súbor, do ktorého ukladáme rôzne dáta, ktoré sa dajú spätne vyvolať v rovnakej forme

prg – Program – na písanie dlhších programov pre ucelené úlohy pri práci s databázami, ktorý sa dá spätne editovať a spúšťať

txt – Text File – textové súbory – na uľahčovanie niektorých procesov pri písaní programov a na ukladanie niektorých príkazov

Nasledujúce príkazy pracujúce s otváraním, ukladaním a zatváraním vyššie uvedených druhov súborov je ľahšie vykonávať prostredníctvom menu, než vypisovaním príkazov. Nutné je to však pri písaní kompaktného programu pre špecifickú úlohu do súboru typu Program. Preto si vypíšeme aj tieto základné príkazy:

[2] všeobecné príkazy v tejto príručke budú uvádzané v tvare: príkaz *príkaz*, t.j., písmo v tvare „obyčajné" sa píše ako je napísané a písmo v tvare „*kurzíva*" obsahovo odkazuje na to, čo nasleduje za príkazom „obyčajné" obyčajné *kurzíva*

Vytvorenie nového pracovného súboru:

```
workfile názov_súboru frekvencia začiatok koniec
```

Uloženie súboru:

```
save názov_súboru
```

Otvorenie súboru:

```
open názov_súboru
```

Zatvorenie súboru:

```
close názov_súboru
```

frekvencia – ide o frekvenciu časových radov (pozri v menu File/New/Workfile...) – identifikuje sa prostredníctvom jediného znaku – napr. ročná (annual) = a, polročná (semiannual) = s, štvrťročná (quarterly) = q, mesačná (monthly) = m, týždenná (weekly) = w, denná s piatimi dňami za týždeň (daily – 5-day-week) = d, denná so siedmimi dňami v týždni (daily – 7-day-week) = 7, nedatovaná (undated) = u.

začiatok a koniec – začiatočný dátum súboru a konečný dátum súboru – táto informácia je uvedená v základnom okne súboru – **Range**; časový interval, v ktorom v danej chvíli v súbore pracujeme sa nazýva **Sample**.

Alternatívnym príkazom na vytvorenie súboru je príkaz WFCREATE (EViews verzia 5.0), ktorý má rovnaké funkcie a možnosti ako WORKFILE (verzia 4.1). Ak príkaz WORKFILE spúšťame z hornej lišty, bude sa nás EViews pýtať na cestu, ak ho spúšťame z *.prg programu, tak ho automaticky otvorí bez dotazovania.

Ak chceme vytvoriť súbor s menom NOVY, v ktorom budeme pracovať s mesačnými dátami a následne uložiť a zavrieť, príkazy budú vyzerať nasledovne:

```
workfile novy m 1993 2009
save novy
close novy
```

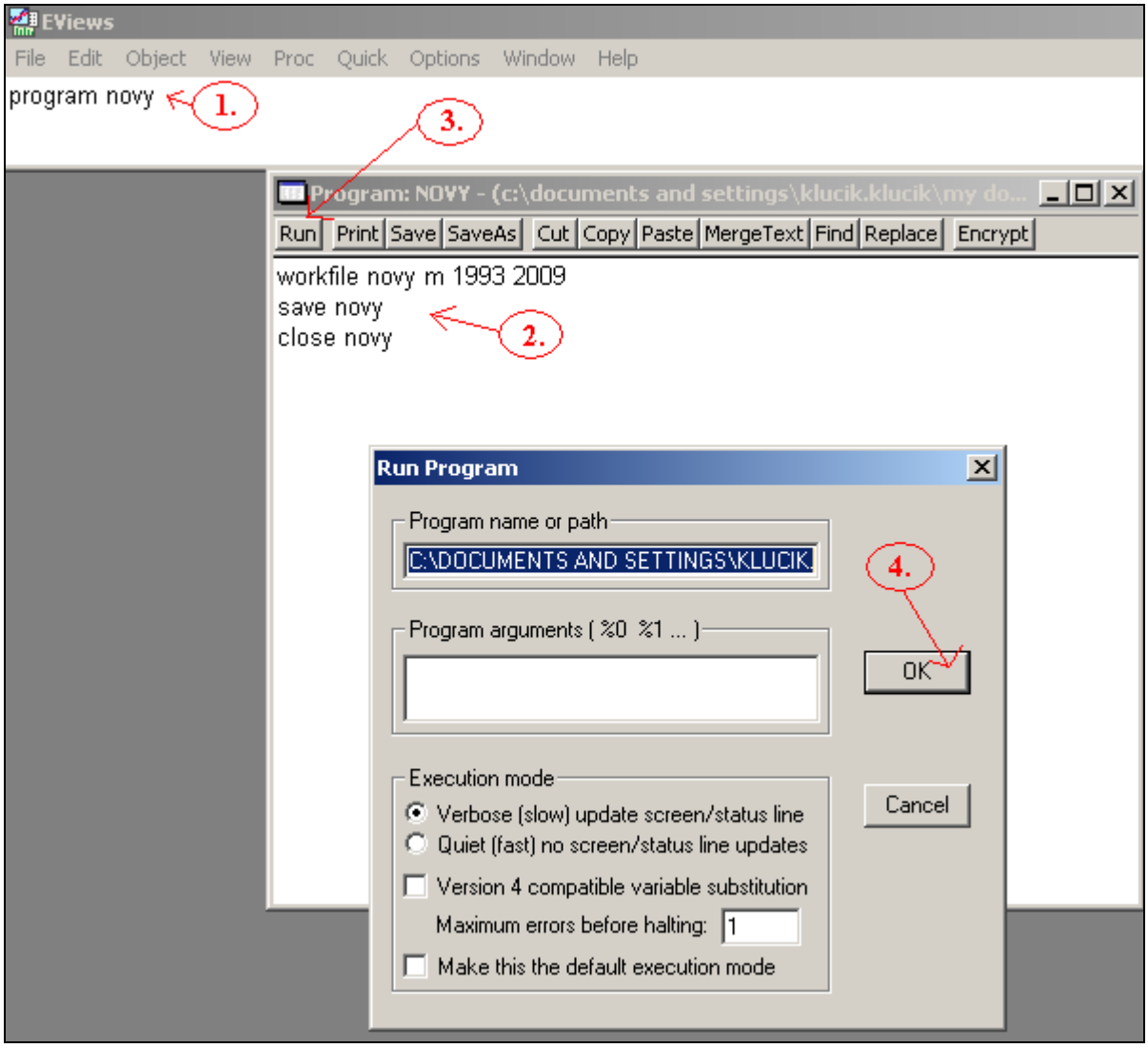
V pomoci programu EViews sa dajú nájsť rôzne možnosti (Options) pre prácu s uvedenými príkazmi, pri ktorých sa dá uviesť aj adresár, do ktorého chceme uložiť súbor a ďalšie náležitosti. Ak pri príkazoch neuvádzame cestu (adresár), bude uložený do adresára, ktorý je nastavený ako aktuálny (default Path =... v pravom dolnom rohu obrazovky).

Na obr. 1.3 (nasledujúca strana) je zobrazený program NOVY (názvy programov a súborov môžu byť rovnaké, lebo majú inú koncovku - *.wf1 a *.prg) a postup spúšťania programu:

1. napíšeme do hornej lišty príkaz: program NOVY a stlačíme ENTER
2. do vytvoreného okna Programu NOVY napíšeme tri príkazové riadky uvedené na obr. č. 1.3 (nasledujúca strana)
3. stlačíme RUN v okne Programu NOVY
4. po stlačení RUN sa ukáže okno s dotazmi na charakter spustenia programu, v ktorom ak nechceme meniť prednastavené náležitosti stlačíme OK a program sa spustí. Vytvorí nový súbor s menom NOVY, uloží ho do predvoleného adresára a zatvorí ho. Overiť si, či sa tak stalo môžete, ak nájdete vo svojom prehliadači (Total Commander alebo Microsoft Explorer) v aktuálnom adresári súbor novy.wf1.

Okno, ktoré sa ukáže po stlačení RUN v programe NOVY (obr. 1.3) ponúka niektoré zaujímavé náležitosti. „Execution mode“ môžete meniť medzi Verbose (slow) – pomalý a Quiet (fast) – rýchly. Pri pomalom móde spúšťania programu je možné v ľavom dolnom rohu (dolnej lište) EViews-u počas spúšťania programu pozorovať, ktoré príkazy sú vykonávané. Pri rýchlom móde tomu tak nie je, a spustenie programu je o niečo rýchlejšia. Ak máme príliš dlhý program a zvolíme pomalý mód, tak môžeme vidieť v ktorej fáze spúšťania sa program nachádza, rýchly mód na druhej strane šetrí čas. Okrem spomínaných sa tam nachádza i možnosť Maximum errors before halting: 1 (Maximálny počet chýb pred zastavením: 1). Toto znamená, že po nájdení jedinej chyby sa spúšťanie programu preruší a chybu nám vypíše. Pri dlhších programoch je výhodné písať vysoké číslo maximálneho počtu chýb, nakoľko môžeme potom všetky nájdené chyby opraviť naraz. Po nájdení jednej chyby pokračuje program spúšťaním ďalších príkazov, ktoré nasledujú po príkaze chybnom. Poslednou možnosťou, ktorá nás teraz zaujíma je „Make this the default execution mode“, teda „Toto bude prednastavená metóda spúšťania“. Tú vyberieme, ak chceme spúšťať program vždy pri rovnakých voľbách (maximálny počet chýb, pomalý alebo rýchly mód).

Obr. 1.3 Vytvorenie pracovného hárku



V ďalšom kroku sa naučíme importovať dáta do EViews-ovského súboru. Ukážeme si to na dátach z Microsoft Excel-u. Na importovanie dát z Excel-u sa používa príkaz READ. Za príkazom READ nasledujú v zátvorke (a za ňou) niektoré parametre oddelené čiarkou (môže ale nemusí byť medzera).

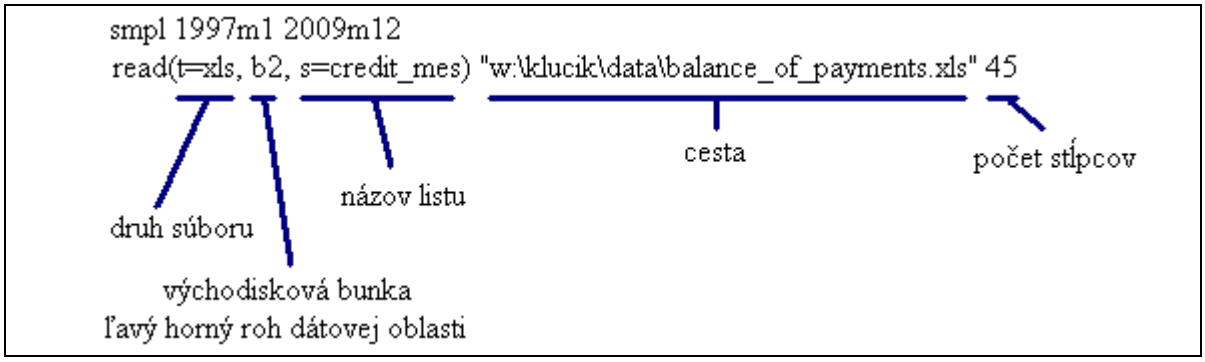
read(t=*typ_súboru*, *horizontálne\vertikálne_dáta*, *ľavá_ horná_bunka*, s=*názov_listu*) *cesta počet_stĺpcov\riadkov resp. názvy_časových_radov*

V tomto príklade sa snažíme vložiť do súboru mesačné dáta platobnej bilancie z Excel-ovského súboru BALANCE_OF_PAYMENTS.XLS (obr. 1.4).

read(t=xls, b2, s=credit_mes) "w:/klucik/data/balance_of_payments.xls" 45

typ súboru – vždy píšeme „t=" a následne typ súboru. Okrem *.xls (teda Excel-ovských súborov), je možnosť importovať súbory Lotus a ASCII textu (s týmito typmi súborov sa nebudeme v príručke zaoberať).

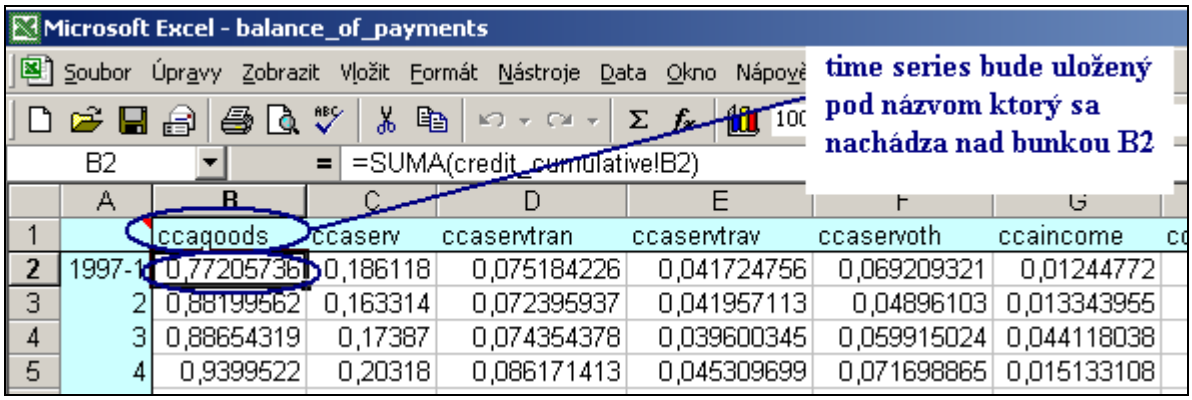
Obr. 1.4 Importovanie dát z Excel-u 1



horizontálne\vertikálne dáta – ak sú zoradené časové rady v Excel-i ako na obr. 1.5 – t.j. v stĺpcoch, tak nepíšeme nič, ak sú časové rady v riadkoch, píšeme – „t", napr.:

read(t=xls, t, b2, s=credit_mes) "w:/klucik/data/balance_of_payments.xls" 45

Obr. 1.5 Importovanie dát z Excel-u 2



	A	B	C	D	E	F	G
1		ccagoods	ccaserv	ccaservtran	ccaservtrav	ccaservoth	ccaincome
2	1997-1	0,77205736	0,186118	0,075184226	0,041724756	0,069209321	0,01244772
3	2	0,88199562	0,163314	0,072395937	0,041957113	0,04896103	0,013343955
4	3	0,88654319	0,17387	0,074354378	0,039600345	0,059915024	0,044118038
5	4	0,9399522	0,20318	0,086171413	0,045309699	0,071698865	0,015133108

ľavá horná bunka – (obr. 1.5), bunka B2 je prednastavenou bunkou, t.j. ak je prvý údaj v tabuľke B2, tak tento údaj nemusíme písať (dobrovoľne). EViews automaticky prenesie všetky dáta smerom nadol od tejto bunky. Ak je prvá bunka bázy dát v inej bunke (z príkladu na obr. 1.7 neskôr), tak tento údaj musíme písať, inak by očakával, že sú údaje uložené od bunky B2, teda napr.:

```
read(t=xls, c4, s=all) "w:/klucik/data/zo_bec.xls" 25
```

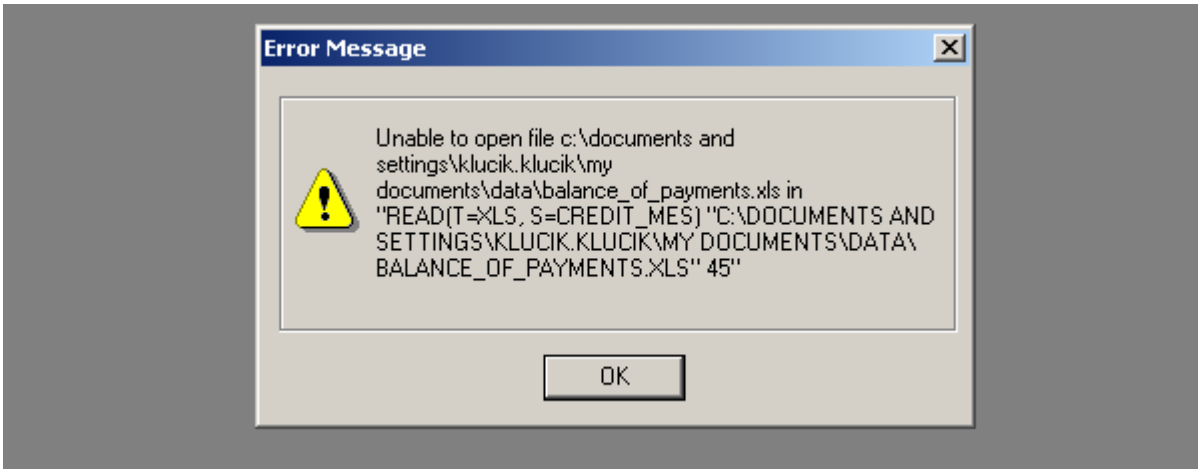
cesta – píšeme adresu konkrétneho súboru typu Excel spolu s jeho názvom, rovnako ako pri príkaze CD, t. j. ak sa v ceste nachádza medzera, píšeme s úvodzovkami. Ak nemáme za príkazom READ napísanú cestu, bude hľadať EViews Excel-ovský súbor v prednastavenom adresári.

Počet stĺpcov či riadkov – uvedieme počet časových radov, ktoré chceme preniesť do EViews-u. V prípade, ak sú údaje usporiadané v stĺpcoch ako na obr. 1.5, tak píšeme počet stĺpcov. V prípadoch, ak sú dáta v riadku, píšeme počet riadkov. Ak tento parameter vynecháme, EViews sa snaží automaticky nájsť všetky údaje v danom liste (podľa stĺpcov) a tie prenesie. Preto náš údaj o počte stĺpcov resp. riadkov je zbytočný, ak prenášame celý list a máme časové rady zoradené.

Názov časových radov (objektov) – V prípade, ak chceme priradiť názvy časovým radom, ktoré importujeme, môžeme namiesto počtu prenášaných časových radov uviesť hneď mená daných objektov. V tomto prípade však priradí uvedené názvy časových radov automaticky podľa poradia v Excel-ovských bunkách. Pôvodné názvy uvedené v Excel-i tak prepíše.

Dôležité je pripomenúť, že ak importujeme dáta z Excel-u, musí byť v tom momente súbor *.xls, z ktorého dáta prenášame zatvorený, inak vypíše EViews chybovú výstrahu, že na danej adrese nemohol súbor nájsť (obr. 1.6).

Obr. 1.6 Chybové hlásenie pri prenášaní dát z Excel-u



Pri vkladaní údajov do EViews-ovského súboru je dôležitý „Sample“, t. j. začiatok a koniec časového rozpätia, v ktorom práve v súbore pracujeme, ako bolo spomenuté vyššie. Dáta platobnej bilancie uložené v Excel-i existujú od roku 1997. Preto pred príkaz READ musíme použiť príkaz SMPL – Sample.

```
smpl začiatok koniec
```

V našom prípade teda potrebujeme, aby nám v súbore NOVY, ktorého Range (celkové rozpätie) je 1993 až 2009, uložil dáta správne od roku 1997. Preto nastavíme Sample do tohto začiatku. Dáta platobnej bilancie nám končia momentálne v decembri 2008, ale koniec SMPL **nemusíme** zadávať presne, pretože prázdne bunky v Excel-i berie EViews ako NA, t. j. nevyplní ich. Preto môže vyzerať celkový príkaz nasledovne:

```
smpl 1997m1 2009m12
read(t=xls, b2, s=credit_mes) “w:/klucik/data/balance_of_payments.xls“ 45
```

V prípade iných ako mesačných údajov píšeme smpl 1997q1 2009q4 a podobne. Za rokom nasleduje vždy znak – m (mesačné), s (polročné), q (štvrťročné) alebo w (týždenné), d (denné)... Za týmto znakom nasleduje poradie (polroku – 1-2, štvrťroku – 1-4, mesiaca – 1-12, týždňa – 1-53, dňa 1-365 atď.). Ak je pracovný hárok bez kalendárnej frekvencie, stačí na stanovenie Sample dvojbodka za rokom a číslo pozorovania: napr. 1994:25.

V našom druhom príklade – obr. 1.7 (ďalšia strana) – vidíme, že začiatok dát je v januári 2004, tak:

```
smpl 2004m1 2009m12
read(t=xls, c4, s=all) “w:/klucik/data/zo_bec.xls“ 25
```

Správne nastavenie momentálneho časového rozpätia v súbore vidíme na hlavnom okne súboru ako bolo spomenuté skôr. Ak by sme nesprávne nastavili smpl – napríklad:

```
smpl 1993m1 2009m12
```

- tak by nám uložil EViews dáta do súboru už od roku 1993. Dôležitým je teda zadanie začiatočnej bunky v príkaze READ rovnako ako začiatočného dátumu SMPL v EViewse.

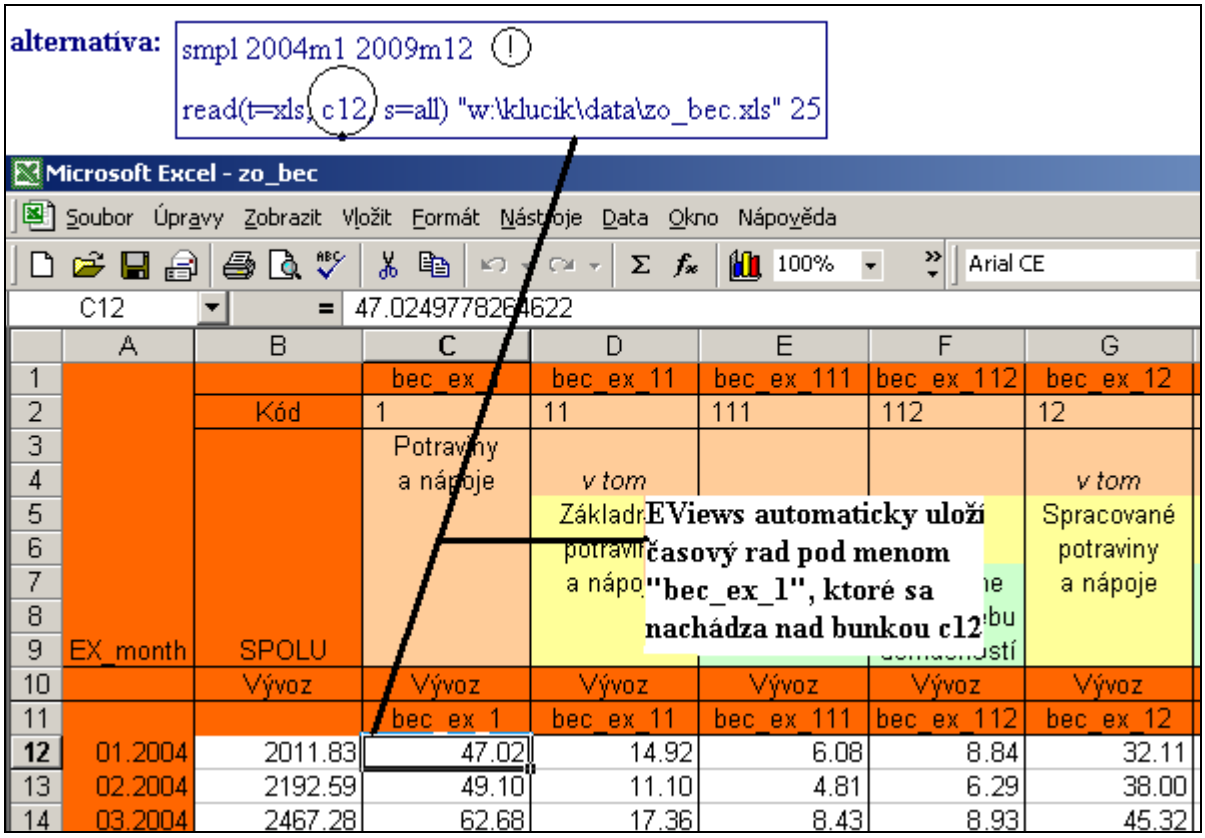
Ak máme údaje z Excel-u importované, môžeme ich vložiť do databázy. Databázy slúžia na uchovávanie objektov z pracovných hárkov a sú praktické z hľadiska opakovaného využívania určitých dát. Pri analýze rôznych údajov máme aj druhú možnosť, že vždy importujeme dáta z Excel-ovského súboru, čím máme zabezpečenú istotu, že údaje sú najčerstvejšie (ak nové údaje vpisujeme do Excel-u a nie rovno do EViews-u). Často však využívame v EViews-e už spracované dáta, potom je lepšie využívať súbory EViews-u s koncovkou *.edb – teda databázy.

Základné príkazy spojené s prácou s databázami sú nasledovné:

```
dbcreate názov_databázy
store(d=názov_databázy) názvy_súborov
fetch(d=názov_databázy) názvy_súborov
```

Okrem týchto príkazov ešte môžeme pracovať s príkazmi ako OPEN, CLOSE, ale všeobecný zápis týchto príkazov je pre všetky pracovné súbory rovnaký ako bol uvedený vyššie pri WORKFILE.

Obr. 1.7 Importovanie dát z Excel-u 3



Ak chceme uložiť do databázy dáta platobnej bilancie a zahraničného obchodu, ktoré sme importovali do nášho súboru NOVY, použijeme jednoduchý príkaz:

```
dbcreate nova
```

Týmto si vytvoríme databázu s názvom NOVA. Táto databáza bude uložená automaticky (pri databázach netreba využívať príkaz SAVE) do nastaveného adresára, rovnakého, v akom pracujeme so súborom NOVY a programom NOVY. Ak spúšťame príkaz prostredníctvom klávesy Enter z horného panelu, tak sa nás bude program pýtať na adresár, v ktorom má databáza byť uložená. Ak ho spúšťame z programu *.prg, tak databázu automaticky uloží do nastaveného adresára. Pri ukážkach budeme odteraz uvažovať iba s príkazmi spúšťanými z programu typu *.prg. Následne vložíme dáta do databázy príkazom STORE:

```
store(d=nova) CCA CCACT CCAGOODS CCAINCINV
```

Do databázy s názvom NOVA ukladáme časové rady s názvami CCA, CCACT, CCAGOODS, CCAINCINV. Ak chceme do databázy uložiť všetky súbory uložené momentálne v našom súbore NOVY.WF1, tak označíme myšou všetky časové rady, ktoré chceme preniesť, stlačíme pravé tlačítko myši, zvolíme COPY a jednoducho napravo od príkazu store(d=nova) stlačíme PASTE. Jednoduché postupy pomocou COPY a PASTE budeme často využívať hlavne pri väčšom množstve analyzovaných časových radov.

Ak by sme chceli využiť časové rady uložené v databáze NOVA, využijeme príkaz FETCH.

```
fetch(d=nova) *
```

Znak * [3] znamená, že importujeme všetky objekty, ktoré sa nachádzajú v danej databáze. Znak * nefunguje pri predošlom príkaze STORE. Celkový program s vytvorením nového súboru, importovaním dát z Excel-u, vytvorením databázy, vytvorením druhého súboru WF1 a importovaní všetkých súborov z databázy by vyzeral nasledovne:

```
cd "w:\klucik\data"
workfile prenos1 m 1993 2009
smpl 1997m1 2009m12
read(t=xls, b2, s=credit_mes) "w:/klucik/data/balance_of_payments.xls" 45
smpl 2004m1 2009m12
read(t=xls, c4, s=all) "w:/klucik/data/zo_bec.xls" 25
dbcreate nova
store(d=nova) CCA CCACT CCAGOODS CCAINCINV
workfile prenos2 m 1993 2009
fetch(d=nova) *
```

Treba ešte dodať, že po importe dát z Excel-ovských súborov by bolo vhodné zmeniť späť rozpätie súboru (SAMPLE) na 1993m1 2009m12, ale pri príkazoch spojených s databázami to **nie je nutné**, nakoľko STORE aj FETCH pracujú s časovým radmi celými, bez ohľadu na zvolený SAMPLE.

[3] znak „*" patrí medzi takzvané „wildcards" – „žolíky", resp. „náhradné znaky", druhým takýmto znakom je aj „?" - podrobnejšie v 2. kapitole

Čo sa týka aktualizácie dát v databáze, v prípade príkazu STORE budú v už existujúcej databáze časové rady s rovnakým názvom automaticky nahradené, čo je z praktického hľadiska značnou výhodou.

2. Základné príkazy pre prácu s objektami

V tejto etape práce už vieme pracovať s jednotlivými súbormi EViews-u a rovnako už máme k dispozícii dáta, ktoré môžeme importovať z Excel-u. Na to, aby sme mohli pristúpiť k samotnej analýze dát, potrebujeme mať ďalšie znalosti o príkazoch týkajúcich sa časového rozpätia (Sample), tvorby objektov v pracovnom hárku – časové rady (Series), skupiny (Groups), grafy, skaláre (Scalars) a ďalších príkazoch.

2.1 Časový interval pracovného hárku

Príkaz SMPL je vari najčastejšie používaným príkazom v dlhých programoch. Medzi jednotlivými procedúrami a etapami vždy meníme časové rozpätie (Sample). Kým pri vytvorení súboru zadávame základné časové rozpätie (Range), ktoré sa už nemusí meniť (iba v nutných prípadoch, napr. ak daný súbor už používame dlhšie než sme plánovali), časové rozpätie (Sample), s ktorým práve pracujeme, sa podľa potrieb mení často. V predchádzajúcej kapitole bol predstavený základný tvar príkazu SAMPLE:

```
smpl začiatok koniec
```

alebo

```
smpl názov_Sample_objektu
```

V prípade, keď chceme určitú časť vnútri časového rozpätia medzi začiatkom a koncom vynechať, existuje zložitejšia forma zápisu:

```
smpl začiatok1 koniec1 začiatok2 koniec2
```

Tento príkaz vynechá časť rozpätia medzi koniec1 a začiatok2. Napríklad:

```
smpl 1993m1 2004m1 2004m3 2006m12
```

Príkaz teda vynechá jedno pozorovanie – z februára 2004.

Samozrejme by sme mohli vzor *začiatok koniec* používať vždy, ale sú prípady, keď to je maximálne nepraktické. Napr. ak chceme stanoviť maximálnu dĺžku intervalu, stačí zadať:

```
smpl @all
```

Tento príkaz stanoví najskorší začiatok a najneskorší koniec časového intervalu v pracovnom súbore, teda vlastne Range – celkový rozsah súboru. Medzi ďalšie skrátené formy stanovenia rozpätia súboru patria:

@first – prvé pozorovanie v pracovnom hárku
@last – posledné pozorovanie v pracovnom hárku

napríklad:

```
smpl @first @last
```

- stanoví rozpätie pracovného hárku na maximum, t.j. na Range pracovného hárku.

Ak opakujeme príliš často to isté rozpätie, ktoré je navyše zložité, môžeme použiť príkaz SAMPLE. Tento príkaz vytvorí objekt Sample – časového rozpätie, jeho znak v pracovnom hárku je  .

```
Sample sampll @first 1994m12 1996m1 2008m12
```

Uvedený príkaz vytvorí objekt časového rozpätia SAMPLE s názvom SAMPLL, ktorý obsahuje časové rozpätie od začiatku pracovného hárku do konca roka 2008, okrem pozorovaní z roku 1995. Objekt Sample po otvorení (dvojnásobným kliknutím myšou) ukáže tabuľku na obr. 2.1 (ďalšia strana). Pre aktiváciu daného Sample napíšeme:

```
smpl sampll
```

Tento príkaz je totožný výberom "*Set workfile Sample equal to this*" a stlačením OK v tabuľke na obr. 2.1 objektu Sample (nasledujúca strana).

Ak pracujeme s veľkým množstvom časových radov, a je potrebné pristupovať ku každému časovému radu osobitne, bolo by časovo náročné až nemožné pred každou operáciou s ďalším časovým radom používať opäť ten istý príkaz so SMPL. Preto môžeme využiť výrok IF statement (AK...). Tento výrok využijeme hlavne pri sezónnom očisťovaní, keď potrebujeme pre každý časový rad stanoviť taký časový interval, aký má samotný časový rad (teda každé pozorovanie musí byť prítomné – bez NA (non available)). Príkaz potom vyzerá veľmi jednoducho:

```
smpl IF názov_časového_radu <>na
```

V tomto prípade sa Sample nastaví na také rozpätie, v ktorom daný časový rad nemá žiadne NA pozorovania. Znaky menší a väčší spolu znamenajú podmienku NEROVNÁ SA. Pri časových radov bez zlomov by to mal byť teda začiatok a koniec časového radu. Sezónne očistenie (napr. pomocou nástroja Tramo/Seats) v prípade nastavenia časového rozpätia aj mimo známych pozorovaní časového radu, by bolo deformované, lebo uvedený nástroj vypočíta pri NA pozorovaniach prognózovanú hodnotu.

Pomocou príkazu IF je možných mnoho variácií okrem menší/väčší. Môžeme si stanoviť Sample na pozorovania s rôznymi podmienkami, napr. >0, =<1, prípadne môžeme do podmienky zaradiť okrem reálnych čísiel i pomenovania ostatných časových radov, napr.:

```
smpl IF názov_časového_radu1 > názov_časového_radu2
```

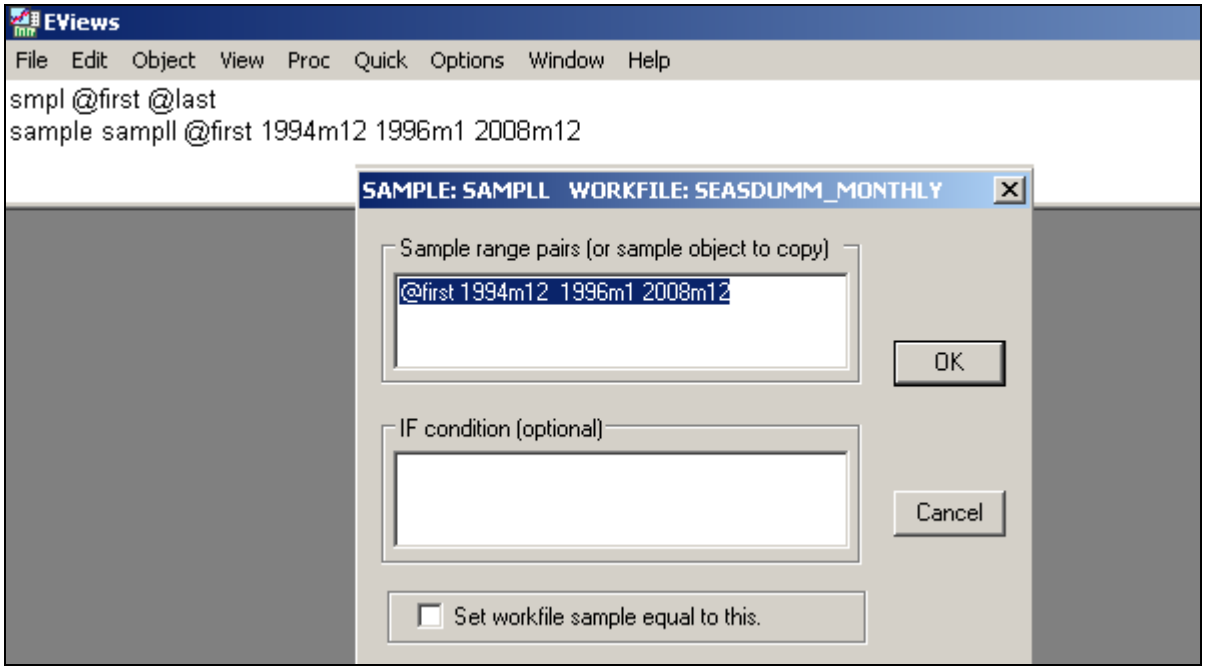
Kombinovať príkaz IF môžeme aj spolu s príkazom AND. Ak chceme aby bol Sample vytvorený na pozorovania, v ktorých je časový rad väčší ako 1 a súčasne druhý časový rad väčší ako nula, príkaz vyzerá nasledovne:

```
smpl IF názov_časového_radu1 >1 and názov_časového_radu2 >0
```

Dôležité je pripomenúť si, že pri príkazoch sa nemusí nachádzať medzera medzi znamienkom > a jednotkou. Rovnako sa vykoná príkaz aj v prípade, ak sa medzi príkazom a špecifikáciou nachádza viac medzier (musí byť však minimálne jedna), napr.:

```
smpl    1993m1    1996m1
```

Obr. 2.1 Stanovenie rozpätia pracovného hárku

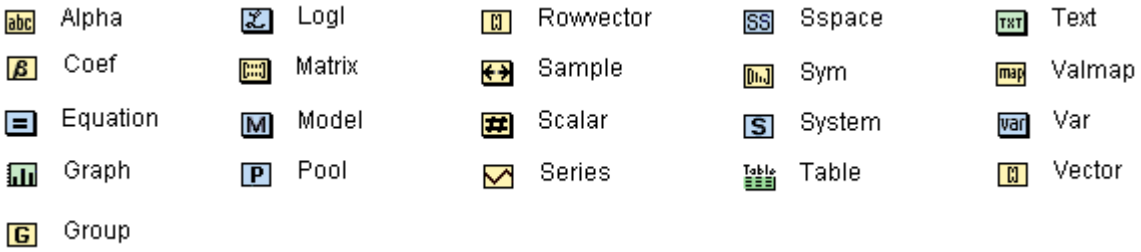


K ďalším príkladom so SMPL sa dostaneme neskôr, keď budeme rozoberať tzv. cykly (loops) spojené s príkazmi FOR..NEXT.

2.2 Základné objekty pracovného hárku

Pracovný hárok pracuje s veľkým množstvom druhov objektov. Objektom je napríklad časový rad. Ďalej skupina časových radov, skalárna veličina, graf, tabuľka atď. Na nasledujúcom obr. 2.2 sú zobrazené základné objekty EViews-u spolu s grafickým označením daného objektového typu.

Obr. 2.2 Grafické označenie objektových typov



Z uvedených objektov budeme používať nasledujúce:

Coef – vektor koeficientov
Equation – rovnica
Graph – graf
Group – skupina (časových radov)
Matrix – matica
Sample – objekt čas. rozpätia
Scalar – skalárna veličina
Series – časový rad
Table - tabuľka

Krátke vysvetlenie funkcií jednotlivých objektov:

Coef – vektor koeficientov vyjadruje najčastejšie parametre funkcií alebo systémov, v našom prípade sa s vektorom koeficientov stretneme pri ukladaní parametrov odhadnutých regresných rovníc
Equation – objekt, ktorý slúži na odhad rovníc, testy odhadnutých rovníc a prognózovanie
Graph – grafické znázornenie objektov
Group – skupina časových radov, slúži na vykonávanie spoločných operácií pre viacero časových radov naraz
Matrix – maticový objekt
Sample – objekt obsahujúci určitý časový interval, ktorý je uložený pod určitým menom, jeho význam spočíva hlavne vo využití často používaných subintervalov pri práci s časovými radmi
Scalar – skalárny objekt, obsahujúci jeden skalár, teda jednu hodnotu, jedno číslo uložené pod určeným menom objektu (často používaná numerická hodnota, napr. 3,14...)
Series – časový rad, v čase usporiadané hodnoty určitých pozorovaní
Table – dvojdimenzionálna tabuľka prezentujúca určitý výstup

Pri začiatku práce musíme daný objekt vytvoriť, resp. definovať. Všetky objekty majú rovnaký základný všeobecný syntax pre ich vytvorenie:

```
typ_objektu názov_objektu
```

napr.

```
series novy
scalar novsi
```

Tieto dva príkazy nám vytvoria časový rad s názvom NOVY a skalárnu veličinu s názvom NOVSI. Ku konkrétnym špecifickým príkladom pre každý druh objektu sa dostaneme neskôr.

K názvom objektov treba dodať, že maximálny počet znakov pre jeden názov je 16. Ak uvedieme v programovom kóde (v príkazoch) názov dlhší, program vypíše chybové hlásenie. Obsahom názvov môžu byť číslice (nie však na prvom mieste), písmená (nerozlišuje sa veľkosť písma), a niektoré znaky napr. dolný podčiarkovník „_“ . Na druhej strane niektoré znaky sú zakázané – napr. pomlčka „-“. O správnosti použitých znakov v názve sa presvedčíte pri vypísaní chybového hlásenia „.... is an illegal or reserved name“ – „.... je zakázaný alebo rezervovaný názov“.

Zakázané sú názvy objektov, ktoré sa podobajú resp. sú identické s niektorými príkazmi programovacieho jazyka. Sú to tieto:

ABS, ACOS, AND, AR, ASIN, C, CON, CNORM, COEF, COS, D, DLOG, DNORM, ELSE, ENDIF, EXP, LOG, LOGIT, LPT1, LPT2, MA, NA, NOT, NRND, OR, PDL, RESID, RND, SAR, SIN, SMA, SQR, THEN

Nepomenované objekty v pracovnom hárku budú nazvané UNTITLED. V Menu EViews-u si môžete v rámci Možností (Options) odškrtnúť možnosť – „Allow only one Untitled“ (dovoliť iba jeden objekt s názvom UNTITLED), hlavne z dôvodu častého využívania objektov UNTITLED. Ak pracujete s väčším počtom objektov, neustále dotazy pri otváraní ďalšieho

UNTITLED objektu zo strany EViews-u, či nechcete predošlý objekt uložiť, dokáže spomaliť a znechutiť prácu.

Okrem samotných názvov objektov budeme využívať i zobrazenie popisu daného objektu. Ide o príkaz displayname:

názov_objektu.displayname *názov_objektu_ktorý_je_zobrazený*

Niekedy pri dlhších programoch spájame rôzne názvy časových radov, či máme taký veľký počet časových radov rovnakého druhu, že môžu vzniknúť názvy ako napr. W00_IPI_SA_TC00 a podobne. V takých prípadoch môže byť praktické, ak sa naučíme priraďovať k jednotlivým časovým radom názvy resp. popisy, ktoré skutočne popisujú o aké časové rady ide. Ak vytvoríme následne graf, už nebude v legende napísané W00_IPI_SA_TC00 ale zrozumiteľný popis daného časového radu. Ušetríme tým čas a nebudeme musieť po každom vytvorení grafu ručne dopisovať popisy časových radov v grafe.

w_ipi_sa_tc00.displayname Industrial production index – trend cycle

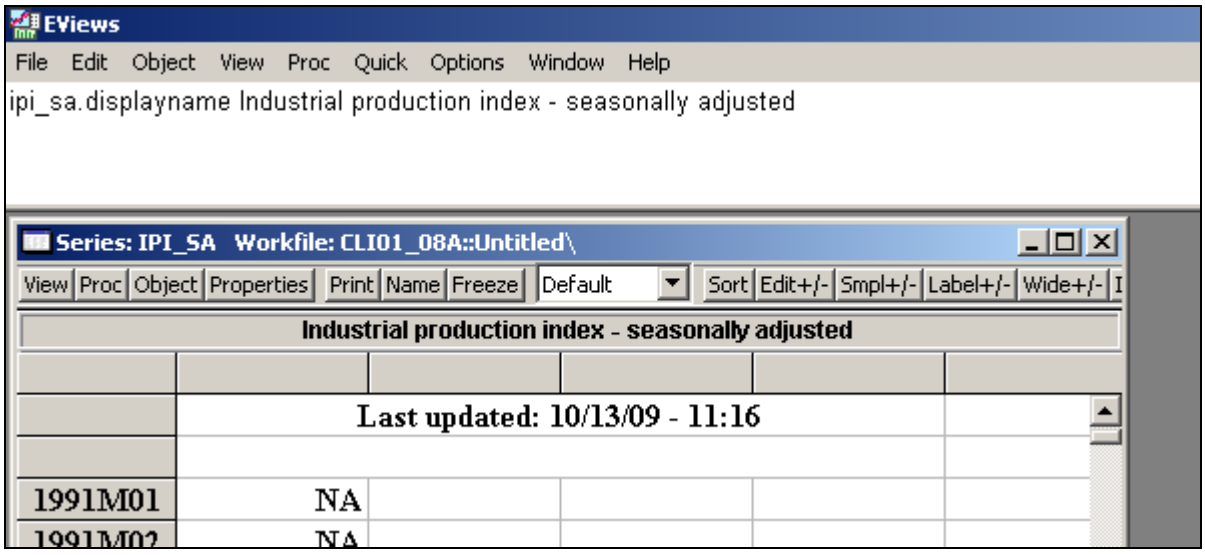
V tomto prípade bude v grafe zobrazený popis časového radu s názvom W_IPI_SA_TC00 ako Industrial production index – trend cycle. V prípade popisov objektov zostávajú veľké písmená zachované, to znamená, že program už rozlišuje malé a veľké písmená.

Pri časovom rade a ostatných objektoch sa rovnako využíva príkaz DISPLAYNAME ako pri grafoch:

ipi_sa.displayname Industrial production index – seasonally adjusted

Výstup tohto posledného príkazu môžeme vidieť na obr. 2.3. Popis sa nachádza teda vždy v šedom poli pod ikonami objektu. Bez priradenia popisu objektu pomocou príkazu DISPLAYNAME, by sa nachádzal na rovnakom mieste názov časového radu, v našom prípade IPI_SA.

Obr. 2.3 Uloženie názvu objektov



Programovací jazyk umožňuje meniť názvy objektov v pracovnom hárku alebo v databáze. Tvar príkazu je rovnaký pre premenovanie každého z objektov:

```
rename staré_meno nové_meno
```

Pri príkaze RENAME sa môžu využívať aj skôr spomenuté tzv. žolíky. Prácu si môžeme uľahčiť pri premenovávaní viacerých objektov:

```
rename x_* y_*
```

Týmto premenujeme všetky objekty, ktoré sa začínajú na X_ na objekty so začiatkom názvu Y_. Ďalším príkazom často využívaným a aplikovateľným pre všetky objekty je príkaz DELETE, teda vymazať.

```
delete zoznam_objektov
```

V tomto prípade môžeme opäť využívať žolíky - ? – pre jeden znak a - * – pre viacero znakov. Príkaz vymaže len objekty v otvorenom pracovnom hárku (resp. databáze), v ostatných pracovných súboroch ich ponechá.

```
delete *_a tseries m?series a??ual
```

Tento príkaz nám vymaže všetky objekty (bez výzvy na potvrdenie), ktoré sa končia znakmi _A, ďalej vymaže objekt s názvom TSERIES, objekty s názvom začínajúcim sa na M a končiace sa na SERIES a súčasne medzi nimi je jeden akýkoľvek znak. Poslednými objektami na vymazanie sú objekty s názvom A??UAL, kde namiesto otáznikov môžeme dosadiť akékoľvek znaky.

2.3 Časové rady – series

Slovo Series znamená v preklade rad. V prostredí EViews-u je chápané pod samotným pojmom Series automaticky pojem časový rad. V prípade, ak pracujeme v pracovných hárkoch s datovanými pozorovaniami hovoríme o časových radoch. Časové rady sú teda chronologicky usporiadané pozorovania určitej veličiny v čase. V tejto príručke sa budeme zaoberať väčšinou s časovými radmi, na úkor ostatných štatistických objektov. Pri tvorbe nového pracovného hárku je však možné si vybrať okrem radov s pravidelnou frekvenciou pozorovaní (regular frequency – dated data) i nedatované – neštruktúrované (undated – unstructured data) dáta alebo tzv. panelové dáta – prierezové dáta (balanced data – cross/section data).

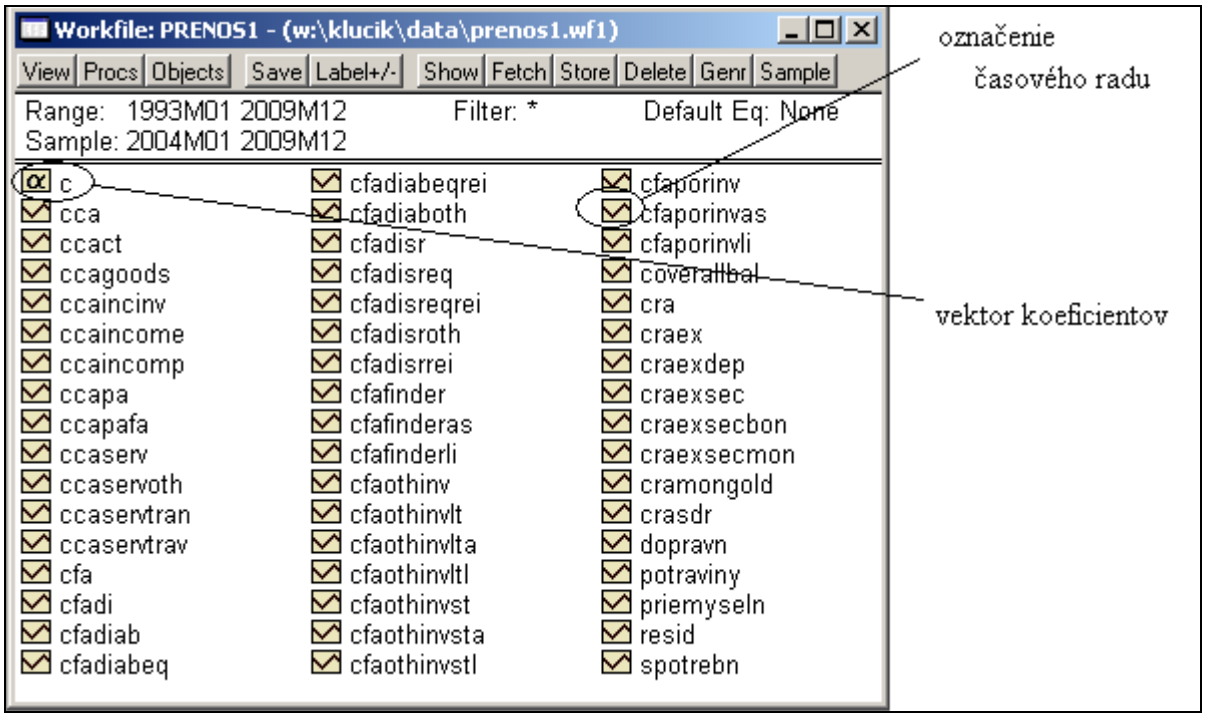
Podľa krátkeho programu na strane 15 by momentálne vyzeral náš pracovný súbor – PRENOS1, do ktorého sme vložili časové rady z Excel-u ako je zobrazené na obrázku na nasledujúcej strane (obr. 2.4). V našom súbore z obrázku 2.4 sa nachádzajú dva druhy objektov. Prvým je „c“, čo je preddefinovaný vektor koeficientov. Ostatné objekty v súbore sú časové rady. Ak chceme vytvoriť nový časový rad, použijeme príkaz SERIES.

```
series názov_časového_radu
```

prípadne

```
series názov_časového_radu = výraz
```

Obr. 2.4 Časové rady – series



Prvý z uvedených príkazov vytvorí časový rad vyplnený znakmi NA, teda prázdny časový rad. Ak vytvoríme prázdny časový rad, môžeme daný objekt otvoriť a vkladať údaje ručne pomocou stlačenia ikony Edit. Ak však chceme zadať hodnoty ihneď, môžeme využiť druhý príkaz, keď za názov časového radu vložíme výraz, ktorému sa má nový rad rovnať. Napr.

```
series novy = 0
```

Takýto príkaz vytvorí časový rad s názvom NOVY so všetkými pozorovaniami rovnými nule. Pozor však na Sample, v ktorom práve pracujeme. Ak pracujeme iba v časovom rozpätí ako je vidieť na obr. 2.4 – január 2004 až december 2009, tak daný príkaz dosadí nuly iba v tomto časovom období. Od začiatku základného časového rozpätia (Range) až po december 2003 zostanú NA hodnoty. Do pôvodného stavu vrátime v časovom rade NOVY hodnoty na prázdne bunky (NA pozorovania) pomocou príkazu

```
series novy = na
```

Za znak ROVNÁ SA môžeme vložiť akýkoľvek výraz, ktorý môže obsahovať i názvy ostatných časových radov (existujúcich v súbore).

```
series novy = cca
```

V tomto prípade v danom Sample sa vytvorí nový súbor NOVY, ktorý bude identický (od 2004-2009) s časovým radom CCA. V prípade, ak už existuje časový rad, ktorému ideme priradiť výraz, nemusí sa príkaz SERIES už písať. napr.

```
cca = 0
```

Výrazy môžu obsahovať okrem základných operátorov (+, -, *, /, ^) aj základné matematické a štatistické funkcie, ktoré sa začínajú na znak @, kde X je názov časového radu:
@abs(x) (absolútna hodnota), @exp(x) (znamená e^x), @sqrt(x) (odmocnina), @mean(x) (priemer).

Existujúci časový rad môžeme napríklad pretransformovať na absolútne hodnoty samého seba:

```
series cca = @abs(cca)
```

resp.

```
cca = @abs(cca).
```

Dôležitým aspektom pri analýze časových radov je časová dimenzia. V regresnej a korelačnej analýze často používame časovo posunuté premenné. V praxi stačí uviesť za meno časového radu v zátvorke hodnotu (tentoraz bez medzery!), o ktorú chceme časový rad posunúť. [4]

```
series novy1 = novy(-1)
```

Časový rad NOVY bude obsahovať časovo posunuté hodnoty pôvodného časového radu o jedno pozorovanie, t.j. napríklad v riadku pre máj 2004 sa bude nachádzať údaj z apríla 2004 – pozri obr. č. 2.5 na nasledujúcej strane – červené šípky. Príklad na obrázku je pre prípad mesačných údajov. Rovnako to však platí i pre údaje s inou frekvenciou. Výraz (-1) znamená pri štvrťročných dátach predošlý štvrťrok, pri ročných predošlý rok. V zátvorke môže byť samozrejme uvedené aj vyššie číslo ako jedna. Časové rady môžu byť posunuté aj smerom dopredu. Ak bude v zátvorke výraz (3), znamená to posunutie časového radu o 3 pozorovania dopredu.

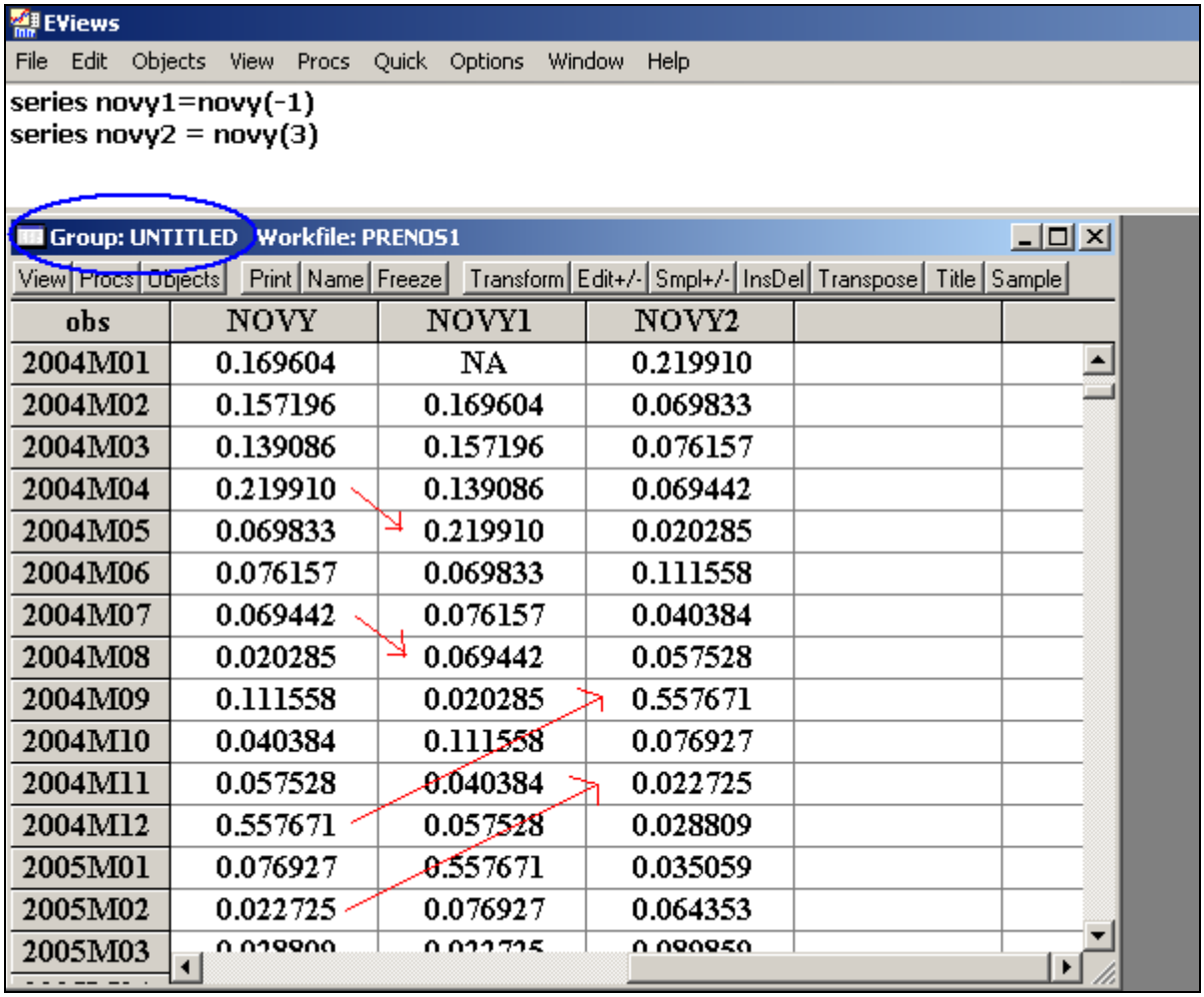
Príklad 1: Jednoduché operácie s časovými radmi

V nasledujúcom komplexnom príklade sú uvedené niektoré najjednoduchšie operácie s časovými radmi. V prvom kroku si opäť definujeme aktuálny adresár, kam sa nám uloží nový pracovný hárok a z ktorého budeme importovať údaje. Vytvoríme si pracovný hárok s názvom CASOVERADY s mesačnou periodicitou od roku 1993 do 2009. Pomocou príkazu READ importujeme dva časové rady z Excel-ovského súboru DATA_MONTHLY.XLS. Prvý sa nachádza v stĺpci BK a druhý v stĺpci CN, pričom v danom liste sú údaje od januára 1993, teda nebudeme musieť meniť časové rozpätie (Sample) pri importovaní, lebo sa zhoduje s obdobím nášho nového pracovného hárku. Aby sme nemuseli pre každý časový rad napísať osobný príkaz READ, použijeme importovanie prostredníctvom vymenovania daných časových radov. Následne priradíme každému časovému radu popis pomocou príkazu DISPLAYNAME. Ďalej chceme vypočítať priemernú hodnotu časového radu priemyselnej produkcie – IPI, použijeme na to príkaz @MEAN. Predtým si môžeme stanoviť obdobie, pre ktoré má byť priemer vypočítaný. Ako sme si ukazovali skôr, príkaz SMPL sa môže použiť s podmienkou IF, v tomto prípade bude obdobie stanovené na čas, keď je index priemyselnej produkcie nenulový, tak si zabezpečíme, aby bol priemer vyrátaný za správne obdobie. Následne vrátime časové rozpätie na celý rozsah súboru (1993-2009). Ako sa pracuje s posunutými premennými môžeme vidieť na vytvorení indexov medzimesačného a medziročného rastu pre tržby v priemysle – TURNIND.

[4] Treba rozlišovať nasledujúce dva prípady: `series novy1 = novy(-1)` a `series novy1 = novy*(-1)`. Prvý prípad je vytvorenie časového radu, ktorým bude časový rad NOVY posunutý o jedno pozorovanie späť. V druhom prípade bude nový časový rad NOVY1 časovým radom NOVY vynásobený mínus jednotkou.

Príliš dlhý názov priemerného rastu na konci programu v príklade môžeme premenovať pomocou príkazu RENAME.

Obr. 2.5 Časovo posunuté hodnoty časového radu



obs	NOVY	NOVY1	NOVY2
2004M01	0.169604	NA	0.219910
2004M02	0.157196	0.169604	0.069833
2004M03	0.139086	0.157196	0.076157
2004M04	0.219910	0.139086	0.069442
2004M05	0.069833	0.219910	0.020285
2004M06	0.076157	0.069833	0.111558
2004M07	0.069442	0.076157	0.040384
2004M08	0.020285	0.069442	0.057528
2004M09	0.111558	0.020285	0.557671
2004M10	0.040384	0.111558	0.076927
2004M11	0.057528	0.040384	0.022725
2004M12	0.557671	0.057528	0.028809
2005M01	0.076927	0.557671	0.035059
2005M02	0.022725	0.076927	0.064353
2005M03	0.028809	0.022725	0.089859

Program pre Príklad 1:

```
cd "w:\klucik\data"
workfile casoverady m 1993 2009

read(t=xls, s=production) "w:/klucik/data/data_monthly.xls"  turnind ipi

ipi.displayname Industrial production index (2005=100)
turnind.displayname Turnover in Industry in current prices

smpl IF ipi <> na
series ipi_mean = @mean(ipi)

smpl @all

series turnind_mom = turnind/turnind(-1)*100
```

```
series turnind_yoy = turnind/turnind(-12)*100
turnind_yoy.displayname Trzby v priemysle – medzirocny rast v %

series turnindyoymean = @mean(turnind_yoy)
rename turnindyoymean rastIND
```

2.4 Tvorenie skupín (Groups)

Pre uskutočnenie ekonometrickej analýzy v EViews-e musíme nevyhnutne spoznať tvorenie tzv. Groups teda skupín. Skupiny sa vytvárajú z viacerých časový radov. Vytváranie skupín umožňuje ľahšiu manipuláciu s časovými radmi, na ktorých chceme previesť rovnaké operácie. Vytvorenie skupiny umožňuje rovnako tvorbu spoločných grafov z členov danej skupiny. Všeobecný príkaz na vytvorenie skupiny je:

```
group názov_skupiny názov_časového_radu1 názov_časového_radu2... atď.
group skupina1 cca ccapa ccaserv
```

V uvedenom konkrétnom praktickom príklade sme vytvorili skupinuz z troch časových radov – CCA, CCAPA a CCASERV. Tie sú označované ako group members, t.j. členovia skupiny. Skupina má v pracovnom hárku znak G. Do skupiny môžeme pridávať dodatočne ďalšie časové rady pomocou príkazu ADD a odoberať pomocou príkazu DROP. Skupinu s názvom SKUPINA1 rozšírime o ďalšie dva časové rady CFA a CFADI.

```
skupina1.add cfa cfadi
```

všeobecne:

```
názov_skupiny.add časový_rad1 časový_rad2...
```

pre prípad odobratia časového radu zo skupiny – drop:

```
názov_skupiny.drop časový_rad1 časový_rad2...
```

Skupinu je možné rozšíriť i o ďalšiu skupinu časových radov naraz. Vytvoríme si SKUPINU2, ktorá bude tvorená 3 novými časovými radmi a časovými radmi v SKUPINE1:

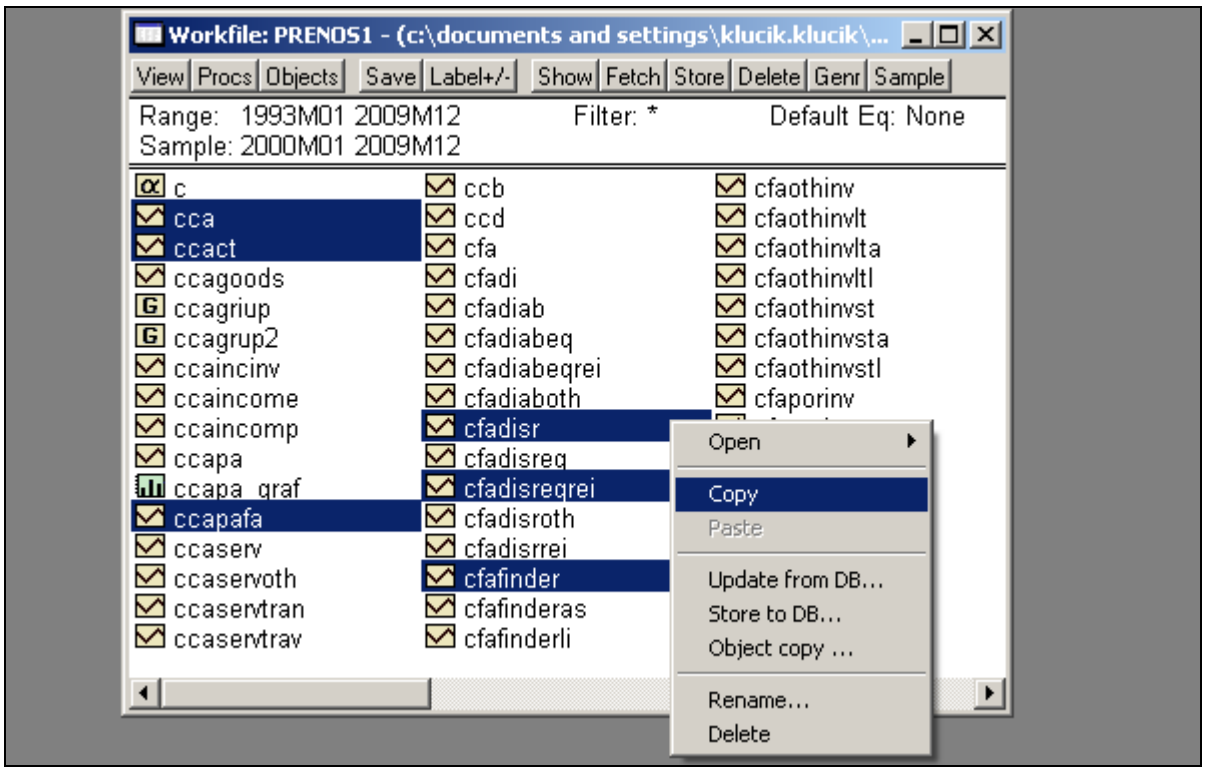
```
group skupina2 CFADISRREI CFAFINDER CFAFINDERAS skupina1
```

Po tomto príkaze obsahuje SKUPINA1 pôvodné časové rady (nič sa nezmenilo) a SKUPINA2 obsahuje časové rady CFADISRREI, CFAFINDER, CFAFINDERAS, CFA, CFADI, CCA, CCAPA a CCASERV. Posledné tri sú pôvodné časové rady SKUPINY1 a ďalšie dve predtým sa stali súčasťou SKUPINY1 po použití príkazu ADD. Pri príkaze GROUP nezáleží na zvolenom časovom rozpätí, jednotlivé časové rady prechádzajú do skupiny s kompletnou informáciou.

Určite ste si všimli, že je časovo náročné vypisovať názvy časových radov pri vytváraní každej novej skupiny. Aj v prípade písania programu v programovacom jazyku sa tento čas dá jednoducho skrátiť pomocou použitia myši. Pomocou myši si označíme časové rady v pracovnom hárku, ktoré chceme aby sa stali členmi novej skupiny časových radov a po stlačení pravého tlačítka myši zvolíme Copy. Prejdeme do nášho programu na miesto *názov_časového_radu* po príkaze GROUP a jednoducho stlačíme Paste. Tým sme ušetrili veľa času a zároveň sme sa vyhli preklepom a potenciálnym chybovým hláseniam EViews-u pri spúšťaní programu. Pri

označovaní časových radov v pracovnom hárku funguje zároveň aj tlačítko CTRL, teda je možné označiť časové rady, ktoré nenasledujú za sebou (rovnako ako pri označovaní buniek v Excel-i). Výstup je zobrazený na obr. 2.6.

Obr. 2.6 Vytváranie skupín časových radov (groups)



2.5 Tvorba grafov

Vytvoriť graf je často praktickejšie prostredníctvom Menu. Ak však pracujeme s väčším množstvom časových radov, tak je výhodnejšie použiť programovací jazyk. Na tvorbu grafov existuje viacero príkazov. Z nich uvedieme iba zopár. Podrobnosti o príkazoch sú uvedené v EViews-e v *Users Guide a Command and Programming Reference*. Príkazový jazyk EViews-u pozná dva ekvivalentné príkazy pre vytvorenie grafu. Prvým je príkaz GRAPH a druhým príkaz FREEZE. Príkaz GRAPH vytvára graf zo zadaných časových radov, prípadne skupín a príkaz FREEZE „zmrazí“ náhľad na zadané objekty. Pre GRAPH je všeobecný zápis nasledovný:

graph *názov_grafu.typ_grafu*(možnosti) *časový_rad1 časový_rad2...*
line(možnosti) *názov_objektu*

Existuje viacero typov grafov. Vo všeobecnom zápise vyššie je uvedený najčastejšie používaný LINE. Okrem čiarových grafov existujú ďalšie typy grafov: AREA (plošný), BAR (stĺpcový), ERRBAR (stĺpcový graf chýb), PIE (koláčový), SCAT (scatterplot – bodový graf), SPIKE (klinový) a ďalšie grafy. Každý z týchto typov má vlastný rad možností, rovnako ako príkaz LINE vyššie.

Okrem typu grafu sú spojené s vlastnosťami grafu ďalšie príkazy týkajúce sa zadávania textu do grafu, úpravy x-ovej a y-ovej osi, legendy, zlučovania grafov atď. Pri každom grafe sú však ciele väčšinou rozdielne, preto sa tejto veľmi širokej téme nebudeme v tejto príručke venovať. Podrobnosti sa dajú nájsť v už spomínanom *Users Guide a Command and Programming Reference* pre EViews.

Môžeme si teraz uviesť jednoduché príklady použitia príkazu GRAPH. Ak chceme spraviť čiarový graf z jedného časového radu (ktorý má názov CCAPA), príkaz bude vyzerať nasledovne:

```
graph graf1.line ccapa
```

resp.

```
line ccapa
```

V prvom prípade program vytvorí grafový objekt s názvom GRAF1 (objekt grafu má v pracovnom hárku grafické označenie). V druhom prípade vytvorí z rovnakého časového radu opäť čiarový graf, avšak bez názvu (UNTITLED), t. j. nevytvorí nový objekt, iba poskytne grafový náhľad na časový rad CCAPA (používa teda objekt časového radu -).

Druhým spôsobom okrem príkazu GRAPH je vytvorenie grafového objektu pomocou príkazu FREEZE (zmraziť), ktorý nám daný náhľad transformuje do grafového objektu, avšak môžeme ho použiť iba v spojení s príkazom *typ_grafu* (LINE, SCAT atď.). Príkaz FREEZE má rovnakú funkciu ako tlačítko Freeze na paneli časového radu (obr. 2.7, strana 30). Teda príkaz bude vyzerať nasledovne:

```
freeze(ccapa_graf) ccapa.line
```

všeobecný zápis príkazu je nasledovný:

```
freeze(názov) názov_objektu.príkaz_náhľadu
```

Príkaz nám vytvorí grafový objekt s názvom CCAPA. Ak by sme použili príkaz nasledovne, t.j. dvojkrokovo:

```
line ccapa
freeze ccapa
```

EViews by vytvoril najprv náhľad na graf časového radu CCAPA a následne by „zmrazil“ časový rad CCAPA, nie však jeho grafový náhľad ale údaje, ktoré sú v tabuľke. To znamená, že výsledkom by bol tabuľkový objekt. Preto používame v dlhých programoch väčšinou príkaz FREEZE, v ktorom je zakomponovaný aj typ grafu, ktorý chceme vytvoriť. Na rozdiel od práce bez programovacieho jazyka, keď po stlačení tlačítka Freeze, zobrazí EViews graf na monitore, v prípade príkazu program grafový objekt iba vytvorí, avšak neotvorí. V prípade, ak chceme aby po skončení programu boli dané grafy otvorené na monitore použijeme príkaz SHOW.

```
show názov_objektu
```

```
show ccapa_graf
```

Posledné dva príkazy FREEZE a SHOW sa dajú použiť aj na ostatné objekty pracovného hárku, teda napr. skupiny, tabuľky atď. K príkladom sa dostaneme neskôr.

Pri tvorbe grafov netreba tiež zabudnúť na zvolený Sample pracovného hárku. Všimnite si na obr. č. 2.7 zvolený Sample a grafy. Grafy sú vytvorené iba pre to časového obdobie, aké bolo zvolené v Sample v tom čase! To znamená, že objekty grafov ostanú po zmenenom Sample

nezmenené. Naopak, ak si vyskúšate zmenu Sample pri otvorenom grafe na monitore, len ako náhľad na graf (cez menu, alebo príkazom – line ...), zmení sa Sample grafu rovnako ako budete meniť Sample pracovného hárku.

Ako ste si mohli všimnúť v predošlej stati o tvorbe grafov, vo všeobecnom vzore príkazu GRAPH je naznačená rovnaká možnosť, na konci riadku príkazu GRAPH, pridávať viacero časových radov. V danom príklade je graf (grafový objekt) vytvorený priamo z daných časových radov bez vytvorenia skupiny. Výsledok je rovnaký, ako keď vytvoríme najprv skupinu z troch časových radov a následne vytvoríme graf pomocou príkazu LINE:

```
group skupina3 CCAPAFA CCASERV CCASERVOTH
line skupina3
```

resp.

```
group skupina3 CCAPAFA CCASERV CCASERVOTH
graph graf3.line skupina3
```

resp.

```
freeze(skupina3_graf) skupina3.line(n)
```

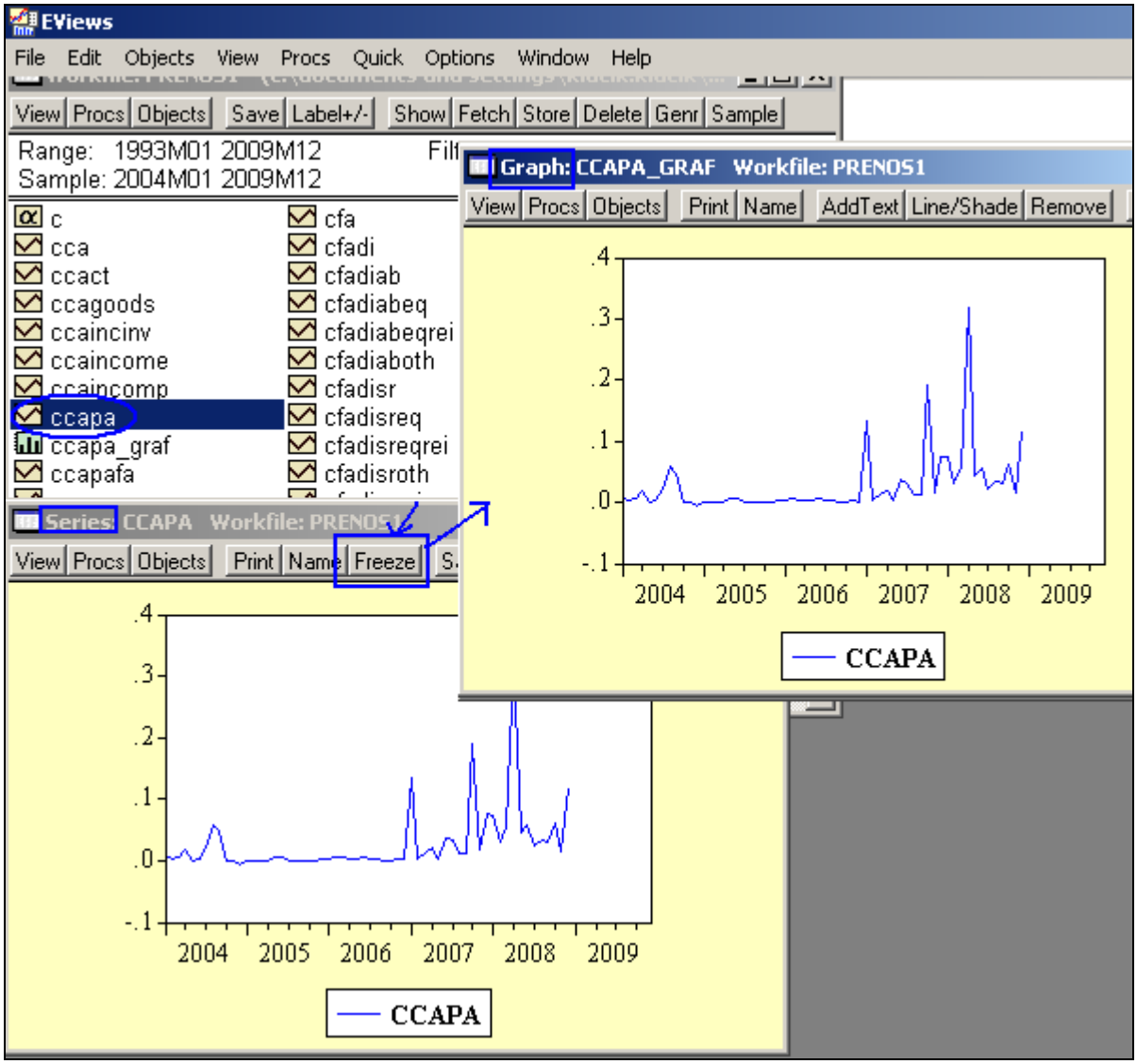
Prvý spôsob poskytuje iba grafový náhľad na SKUPINU3 bez toho aby vytvoril nový objekt grafu, to znamená, že sa nám graf stratí, ak zmeníme náhľad, rovnako ako to bolo v prípade vytvorenia grafu v samotnom okne časového radu. Ak chceme vytvoriť grafový objekt, môžeme teda použiť príkaz GRAPH, rovnako ako v predošlej stati, alebo príkaz FREEZE. V poslednom príklade je využitá pri príkaze LINE aj možnosť ´n´ (patrí medzi Options príkazu LINE). V tomto prípade vytvorí príkaz líniový graf z troch časových radov, s tým, že všetky časové rady sú normalizované – t.j. majú normalizovanú mierku – jednotkovú štandardnú odchýlku a nulový priemer. Na ďalšom obrázku môžeme vidieť rozdiel medzi pôvodným grafom a normalizovaným grafom z rovnakých časových radov (obr. 2.8, strana 31).

Keďže typ grafu LINE je najčastejšie používaným pri grafickej analýze časových radov, uvedieme si okrem spomenutej možnosti „n“ aj ďalšie. Pripomenieme si najprv všeobecný zápis príkazu LINE spomenutý vyššie, doplnený o ďalšie dve podoby zápisu:

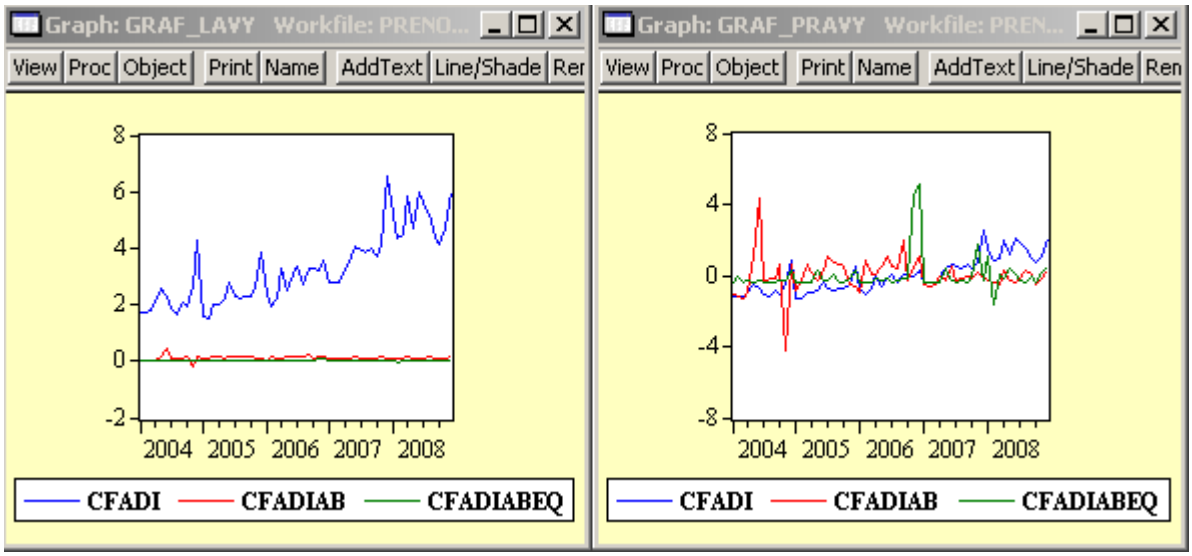
```
line(možnosti) názov_objektu(objektov)
názov_objektu.line(možnosti)
názov_grafu.line(možnosti)
```

Medzi možnosti príkazu LINE patria hlavne tzv. Scale options (mierky), medzi ktoré patrí i nami využitá možnosť „n“. Okrem nej poznáme tieto najdôležitejšie možnosti mierky:
d – duálna mierka bez pretínania (čiary sa nemôžu pretínať, graf má dve mierky, jednu vľavo a druhú vpravo)
x – duálna mierka s možnosťou pretínania
s – kumulovaný čiarový graf. Každá oblasť predstavuje kumulatívny počet daného časového radu.

Obr. 2.7 Vytvorenie grafu pomocou príkazu FREEZE



Obr. 2.8 Zobrazenie normalizovaných časových radov



2.6 Programovacie premenné

Programovacie premenné výrazne obohacujú programovací jazyk EViews-u a umožňujú rýchlejšie spracovanie tradičných operácií pri spúšťaní programov. Existujú dva druhy programovacích premenných – *kontrolné premenné* a *reťazcové premenné*. Tieto dva druhy premenných sa nazývajú programovacie, pretože sa nedajú použiť priamo z príkazového riadku (hornej lišty). Využiť sa dajú len v súbore typu Program.

2.6.1 Kontrolné (skalárne) premenné

Kontrolné premenné môžu nahradzovať číselné hodnoty v programe. Preto sa môžu nazývať tieto premenné aj skaláre (skalárne veličiny). Všeobecný zápis kontrolnej premennej je nasledujúci:

```
!názov_premennej
```

Názov kontrolnej premennej vždy začína znakom ´!´, keďže maximálny počet znakov v názve objektu EViews-u je 16, samotný názov premennej za výkričníkom môže byť dlhý maximálne 15 znakov. Názvom skalárnej premennej môže byť rovnako číslo, ako aj písmeno, či ich kombinácia.

Kontrolnú premennú treba pred jej použitím definovať – vytvoriť. Teda napríklad:

```
!x = 10
!pi = 3.14159
```

Ak tieto príkazy vypíšeme do príkazovej lišty priamo v programe, pri otvorenom pracovnom hárku, nevytvorí sa žiadny objekt. Kontrolné premenné nie sú objektom, môže sa definovať iba ich hodnota a tá následne potom môže prejsť do rôznych objektov, ku ktorým ju priradíme.

Napríklad:

```
series pi = !pi
```

Táto dvojica príkazov:

```
!pi = 3.14159
series pi = !pi
```

nebude fungovať, ak ju budeme zadávať postupne priamo do hornej príkazovej lišty pracovného prostredia EViews-u. Pri druhom príkaze (series pi) vypíše chybové hlásenie že „*!pi is not defined*“, teda nie je definovaný. Avšak, ak oba riadky skopírujeme do nového súboru typu Program a spustíme Run, EViews vytvorí časový rad s názvom PI a priradí mu všade hodnotu našej skalárnej premennej !pi, teda 3.14159.

Zmysel tejto veličiny pozostáva v tom, že nebudeme musieť priraďovať jednotlivo vždy určitú hodnotu k viacerým objektom. Ak sa opakuje často určitý príkaz priraďujúci objektom rovnakú hodnotu, stačí nám nahradiť túto hodnotu na začiatku programu (pred prvým použitím) skalárnou veličinou.

Skalárne veličiny môžu vystupovať aj vo vnútri rôznych expresných výrazov:

```
series skalarna = 12+!pi*10
```

Ich najrozšírenejšie využitie je však pri cykloch FOR...NEXT a pri stanovovaní Sample. K používaniu cyklov sa dostaneme neskôr. Pri stanovovaní aktuálneho časového rozpätia pracovného hárku pomocou príkazu SMPL môžeme využívať skalárne veličiny nasledovne:

```
!rast = 10
smpl 1991m1 1991m1+!rast
```

Dané výrazy zmenia aktuálny Sample hárku na 1991m1 1991m11. V prípade „smpl 1991m1 1991m1“ by sme mali v aktuálnom Sample jedno pozorovanie (teda januárové). Ak pridáme (+10), teda číslo desať k januáru, znamená to, že rozširujeme vlastne Sample o 10 pozorovaní. Výsledné časové rozpätie bude obsahovať teda 11 pozorovaní – 1991m1 1991m11.

Ďalším príkladom je použitie tzv. prírastkových (úbytkových) skalárnych veličín (incremental resp. decremental). Táto problematika je rozpísaná neskôr, v kapitole 3.5.

2.6.2 Reťazcové premenné

Názov reťazcová premenná je odvodená od slova reťazec – teda String, String variable. Reťazec je text, ktorý sa uvádza v úvodzovkách. Reťazcová premenná je teda premenná, ktorej hodnota je text uvedený v úvodzovkách. Všeobecný syntax reťazcovej premennej je nasledujúci:

```
%názov_premennej = “text“
```

Každá reťazcová premenná sa začína znakom ´%´, za ktorým nasleduje maximálne 15 znakový názov danej premennej. Príklad:

```
%d = "Hrubý domáci produkt v stálych cenách roka 2000"
```

Celý výraz v predošlom riadku znamená priradenie reťazca (výrazu v úvodzovkách) reťazcovej premennej s názvom D. Reťazcové premenné podobne ako kontrolné premenné nie sú objektom, môže sa teda definovať iba ich hodnota a tá následne môže prejsť do rôznych objektov, ku ktorým ju priradíme. Obe rovnako využívame výlučne v programovom súbore *.prg EViews-u.

Reťazec, ak jeho obsahom je číselný údaj, je možné previesť do formy skaláru, teda skalárnej veličiny. Toto je možné pomocou príkazu @val:

```
!názov_skalárnej_veličiny = @val(%názov_reťazcovej_veličiny)
```

```
%retazec1  = "0.3"
!skalar1 = @val(%retazec1)
```

Z textu v úvodzovkách sme dostali číselnú hodnotu – 0.3 uloženú v skalárnej veličine.

Kontrolné i reťazcové veličiny môžu byť obe využité ako substitučné veličiny – Replacement variables. V rôznych príkazoch môžeme nahradiť želaný text, resp. hodnotu reťazcovou či skalárnou veličinou. Príklad pre substitučnú reťazcovú premennú:

```
%x = "novy"
series {%x} = 110
```

V prvom riadku sme definovali reťazcovú premennú X ako premennú, ktorej obsahom resp. hodnotou je text v úvodzovkách – NOVY. V druhom riadku sme vytvorili časový rad s názvom NOVY, ktorého hodnotou je vo všetkých pozorovaniach číslo 110. Všimnite si, že uvedenú reťazcovú premennú uvádzame pri substitúcii v zloženej zátvorke. Ak by sme zloženú zátvorku vynechali, tak by EViews bral ako vstup príkaz: series "novy" = 110, čo by vyvolalo chybové hlásenie.

Príklad pre skalárnu premennú:

```
!one = 1
series novy{!one} = 1
```

Skalárna veličina s názvom ONE je definovaná v prvom riadku, v druhom riadku je vytvorený časový rad s názvom NOVY1. Pri skalárnej veličine je v tomto prípade označenie substitučnej veličiny do zloženej zátvorky voliteľné (môže i nemusí byť).

2.7 Ďalšie často využívané príkazy

V tejto podkapitole si vysvetlíme ďalšie veľmi jednoduché avšak praktické príkazy, aby mohli naše znalosti následne vyústiť do praktického využitia pri makroekonomickej analýze.

2.7.1 Komentáre

Každý bežný programovací jazyk disponuje aj možnosťou vpisovať do programu komentáre. Hlavne pri dlhších programoch strácame prehľad o prebiehajúcich procesoch, preto môžeme prispievať do programu komentármi, popismi atď. Komentáre sa začínajú na znak „ ‘ ” – na anglickej klávesnici sa nachádza naľavo od klávesy ENTER, jeho ASCII znak je ALT+39.

Príklad na komentár:

```
!i = 1
series anovy = novy2/100                    ‘ delenie časového radu 100
series astary = stary!i
group avsetky a*
‘ ***********KONIEC prvej časti programu ***************
```

Všetko, čo nasleduje za znakom „ ‘ ”, program ignoruje, až po ďalší nový riadok po stlačení klávesy ENTER. Do poznámok je vhodné písať zvýrazňovacie znaky, tak sa budeme lepšie orientovať v texte a ľahšie nájdeme príslušne state, ktoré by sme chceli prípadne upravovať. Poznámky resp. komentáre možno písať priamo do riadku, kde už je príkaz, alebo do nového riadku. Príkaz pred týmto znakom bude vykonaný bez vypísania chybového hlásenia a program bude pokračovať vykonaním príkazu v ďalšom riadku.

2.7.2 @-funkcie

Teraz si vysvetlíme niektoré veľmi praktické a často využívané príkazy, ktoré sa začínajú na znak zavináča @. Doteraz sme využívali @-funkcie pri stanovovaní časového obdobia pracovného hárku - @all, @first, @last, ďalej sme spomenuli niektoré základné matematické funkcie @abs, @exp, @sqrt a štatistickú funkciu @mean. Zo špeciálnych funkcií to bola funkcia @val pre transformáciu reťazca na skalár. Paleta @-funkcií je v EViews-e veľmi široká a dali by sa voľne rozdeliť do nasledujúcich skupín, pričom ich členenie nie je jednoznačné, nakoľko niektoré sú veľmi špecifické a je ich ťažké zaradiť jednoznačne do určitej skupiny:

a) funkcie pre stanovenie časového obdobia – @all, @first, @last
b) matematické funkcie – okrem spomenutých tam zaraďujeme i faktoriál @fact, logaritmus @log, funkciu zaokrúhľovania @round a iné.
c) štatistické funkcie – okrem priemeru tam môžme zaradiť @var (rozptyl), @cor (korelácia), @sum (suma), @median (medián), @min (minimum), @max (maximum), @obs (počet pozorovaní)
d) rekódovacie funkcie – rekódery slúžia na zmenu určitej hodnoty na inú, ide o dve základné funkcie, ktoré budeme pri analýze časových radov často využívať – @recode a @nan

e) funkcie spojené s regresnými odhadmi – k týmto funkciám sa dostaneme v 4. kapitole (@coefs, @aic...)
f) funkcie pre časové rady – prvá diferencia (@d), logaritmická diferencia (@dlog), kĺzavý priemer (@movav) atď.
g) špeciálne funkcie – funkcie pre integrál, derivácie a logistickú transformáciu (@beta(a,b), @logit(x)...)
h) štatistické distribučné funkcie – sú asociované s rôznymi rozdeleniami časových radov (normálne, exponenciálne, binomické a ďalšie rozdelenia) – @c, @d...
i) štatistické funkcie pre skupiny – sú výrazy na výpočet štatistík z rôznych skupín na základe spoločných vlastností (@obsby(arg1, arg2[, s])...)
j) ostatné špecifické funkcie – napr. spomenutá funkcia @val

Z uvedených budeme v tejto príručke využívať hlavne príkazy z prvých 5 skupín. V nasledujúcej časti si vysvetlíme rekódovacie funkcie.

Ide o takzvané rekódery. Prvým z nich je @recode. Príkaz @recode rekóduje resp. prepíše na základe danej podmienky jednu hodnotu na inú hodnotu. Príkaz @recode má všeobecný syntax:

```
@recode(podmienka, x, y)
```

Príkaz vráti hodnotu x, ak je splnená žiadaná podmienka. Ak nie je splnená podmienka, vráti hodnotu y. Tento príkaz využívame hlavne pri extrapoláciách, t.j. predlžovaní časového radu do minulosti resp. pri zmene frekvencie časového radu (napríklad zo štvrťročnej na mesačnú). Zoberme si nasledujúci konkrétny príklad. Chceme sa vyhnúť napríklad situácii delenia nulou. Ak máme v nejakom časovom rade nuly, môžeme ich nahradiť veľmi malým číslom – napr. 0.000001:

```
series beznul1 = @recode(beznul1=0, 0.000001, beznul1)
```

Tento príkaz nahradí všetky nulové hodnoty časového radu BEZNUL1 hodnotou 0.000001, ostatné hodnoty ponechá – znak Y je teda pri nesplnení podmienky BEZNUL1=1 pôvodná hodnota pozorovania časového radu.

Druhým rekódovacím príkazom je príkaz @nan. Slúži opäť na nahradenie jedného znaku druhým. Tento príkaz je však orientovaný iba na znak NA – t.j. prázdne pozorovanie – NON AVAILABLE. Keď vykonávame operácie automaticky s viacerými časovými radmi v skupinách, pričom časové rady majú niekde vynechané pozorovanie – NA, daná operácia sa nevykoná. Napríklad chceme sčítať všetky časové rady priemyselného odvetvia, výsledkom čoho sú celkové tržby v priemysle. Výsledný časový rad bude mať v niektorých pozorovaniach NA hodnotu, a to preto, lebo ju obsahoval iba jeden časový rad. V tomto prípade je teda vhodné nahradiť NA pozorovania v časových radoch nulami. Všeobecný syntax príkazu je nasledujúci:

```
@nan(x, y)
```

Hodnota X je pôvodný časový rad, hodnota Y je hodnota, ktorú chceme dosadiť namiesto NA hodnoty v časovom rade X. To znamená, že príkaz vráti hodnotu X (pôvodnú) ak X<>NA a vráti Y ak X=NA. Potom:

```
series bezna = @nan(bezna, 0)
```

Časový rad s názvom BEZNA bude obsahovať všetky pôvodné hodnoty BEZNA (X) a hodnoty NA budú nahradené hodnotou 0 (Y).

2.7.3 Skaláre

Často využívaným príkazom je príkaz SCALAR. Scalar je označenie pre skalárnu veličinu, teda samostatný číselný údaj (jedno číslo). Skalárne veličiny využívame hlavne pri výpočte tzv. bázických indexov, t.j. indexov s bázickým základom – môže byť ním napríklad priemer roka 2000, vtedy hovoríme o bázickom indexe priemerného roku 2000. Pri tržbách v stavebníctve je priemerom roka 2000 suma všetkých tržieb jednotlivých mesiacov roku 2000 vydelená počtom 12. Ako to však spravíme? Doteraz vieme robiť operácie len s celými časovými radmi, vieme ich zrátať, odčítať, vynásobiť atď. To znamená, že by sme teoreticky mohli vytvoriť nových 12 časových radov a do každého dať hodnotu tržieb v stavebníctve za jednotlivý mesiac a potom sčítať a vydeliť 12timi. To je však veľmi nepraktické, preto použijeme príkaz SCALAR.

Všeobecný popis príkazu SCALAR je nasledujúci:

```
scalar názov [=výraz]
```

Ak chceme nejaký časový rad vydeliť číslom 2,1654 tak napíšeme:

```
scalar skalar1 = 2,1654
series casrad1 = casrad1/skalar1
```

Skalár má v pracovnom hárku označenie . Skaláru musíme priradiť hodnotu súčtu všetkých pozorovaní v roku 2000 a deliť číslom 12. Na to musíme použiť jeden z dvoch nasledujúcich príkazov. Jednoduchší je príkaz resp. funkcia @sum (teda suma), druhou je funkcia @elem.

Funkcia @sum vypočíta sumu všetkých hodnôt časového radu v danom Sample pracovného hárku. Argument pre Sample však môžeme použiť iba vtedy, ak je priradený časovému radu, t.j. ak zisťujeme vo vybranom časovom období sumu určitého časového radu:

```
@sum(názov_časového_radu[,Sample])
```

Funkcia @elem priradí (vráti) hodnotu časového radu v určenom pozorovaní (dátume):

```
@elem(názov_časového_radu, argument)
```

Ak je pracovný hárok datovaný (t.j. pracujeme s časovými radmi), tak sa ako argument uvádza pozorovanie v úvodzovkách, napr. január 2005 – "2005m1". Argument Sample môžeme využiť aj v prípadoch, ak ho priraďujeme skaláru, na rozdiel od predošlej funkcie @sum.

Ak chceme zistiť hodnotu tržieb v stavebníctve za celý rok 2000, musíme zrátať hodnoty v jednotlivých mesiacoch. Použijeme príkaz SCALAR a priradíme mu hodnotu pomocou príkazu @sum. Snažíme sa priradiť hodnotu skaláru, to znamená, že v zátvorke Sample (*názov časového radu*[,Sample]), nebudeme môcť použiť. Preto musíme nastaviť najprv Sample pracovného hárku pomocou príkazu SMPL na rok 2000 a následne použiť príkaz SCALAR:

```
smpl 2000m1 2000m12
scalar baza2000 = @sum(turncon)
```

Pre štvrťročné údaje v pracovnom hárku so štvrťročnou frekvenciou:

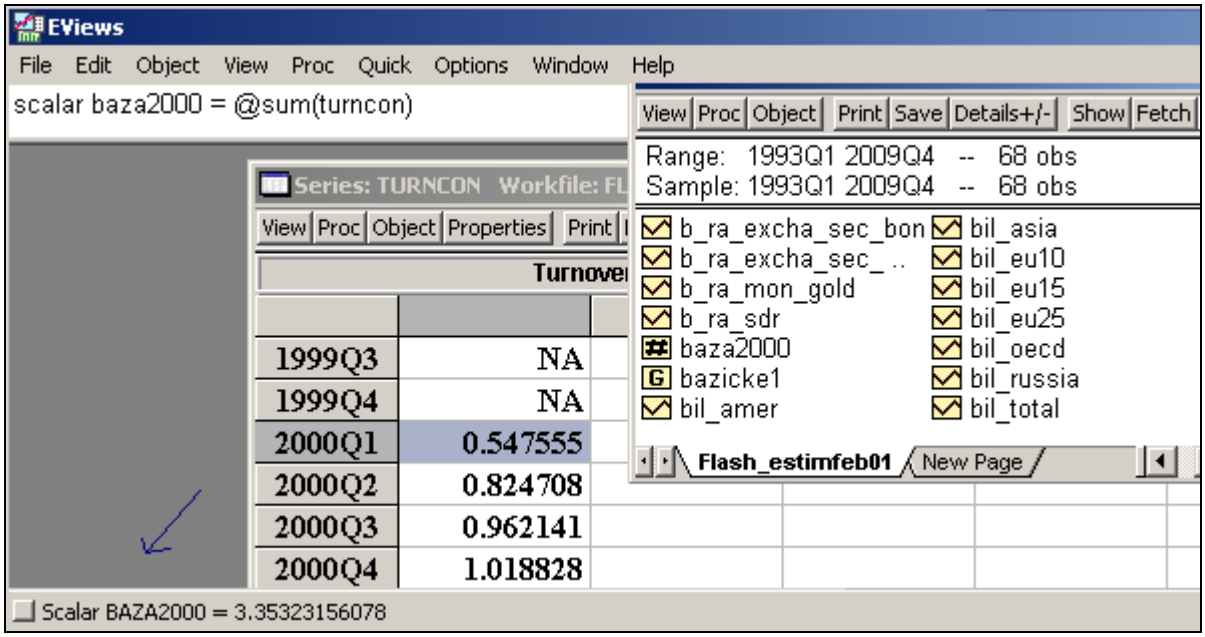
```
smpl 2000q1 2000q4
scalar baza2000 = @sum(turncon)
```

Sčítanie hodnôt tržieb v stavebníctve za rok 2000 sa dá aj pomocou príkazu @elem. Pri mesačných údajoch je to však zložitejšie, pretože musíme vypisovať každý mesiac osobitne. Tento príkaz si však predstavíme, nakoľko sa určite zíde pri písaní dlhších programov. Na zvolenom Sample pracovného hárku nezáleží, nakoľko si príkaz vyhľadá vždy presnú hodnotu časového radu v danom pozorovaní. Pre štvrťročné údaje by príkaz na zistenie skaláru 2000 vyzeral nasledovne:

```
scalar baza2000 = (@elem(turncon, "2005m1") + @elem(turncon, "2005m1") +
@elem(turncon, "2005m1")+@elem(turncon, "2005m1"))
```

Pre kontrolu si môžeme výpočet overiť pomocou kalkulačky. Výslednú hodnotu skaláru v EViews-e zistíme, ak klikneme dvakrát myšou na objekt skaláru. Jeho hodnota sa objaví v ľavom dolnom rohu obrazovky – v stavovom riadku – obr. č. 2.9.

Obr. 2.9 Skalárny objekt



Príklad 2: Výpočet stálych cien štvrťročných údajov z odvetví

Nasledujúci príklad bude obsahovať praktické riešenie úlohy pri makroekonomickom prognózovaní. Prepočet tržieb z bežných do stálych cien si vyžaduje okrem pôvodného časového radu aj index stálych cien, respektíve časový rad spotrebiteľských či výrobných cien na prepočet časového radu do stálych cien. Ide teda o očistenie časového radu o vplyv rastu cien, prípadne o vplyv ich poklesu. Vypovedacou hodnotou časového radu v stálych cenách je informácia o skutočnom náraste resp. poklese nakúpeného, prípadne vyrobeného množstva určitej jednotky v danom odvetví. V príklade sa budeme zaoberať prepočtom do stálych cien pre časové rady v odvetviach stavebníctva (produkcia), priemyslu, veľkoobchodu, maloobchodu a tržieb v doprave. Následne časové rady prepočítame do bázických indexov roku 2000, to znamená do indexov s priemerom roka 2000=100. Bázické indexy slúžia ako normalizačný nástroj. Normalizované časové rady sa ľahko interpretujú pri regresných odhadoch.

V príklade využijeme štvrťročné údaje. V prvom kroku definujeme aktuálny adresár, ďalej vytvoríme pracovný hárok a postupne importujeme dáta z Excel-u. Pri importovaní musíme dávať pozor na zvolený Sample súboru. Uvedené dáta nebudeme vkladať do databáz. Následne môžeme pristúpiť k samotnému výpočtu.

Najprv vytvoríme nový program prostredníctvom menu alebo napísaním do hornej lišty – „program staleceny“. Ďalej opíšeme nasledovné:

```
cd “w:\klucik\data”
workfile staleceny q 1993 2009
smpl 1993q1 2009q4
read(t=xls, s=all3) “w:/klucik/data/data_quarterly.xls“ 148
read(t=xls, s=all4) “w:/klucik/data/data_quarterly.xls“ 90

‘**** priradíme k základným časovým radom popis – bez diakritiky ****
conprod.displayname Stavebna produkcia v beznych cenach
turnind.displayname Trzby v priemysle v beznych cenach
turnret.displayname Trzby v maloobchode v beznych cenach
turnwho.displayname  Trzby vo velkoobchode v beznych cenach
turntrans.displayname Trzby v doprave a v skladovani v beznych cenach

series conprod_sc = conprod/conprod_isc
series turnind_sc = turnind/ppi
series turnret_sc = turnret/turnret_isc
series turnwho_sc = turnwho/ppi
series turntrans_sc = turntrans/cpitrans

smpl 2000q1 2000q4
scalar conprod2000 = @sum(conprod_sc)/4
scalar turnind2000 = @sum(turnind_sc)/4
scalar turnret2000 = @sum(turnret_sc)/4
scalar turnwho2000 = @sum(turnwho_sc)/4
scalar turntrans2000 = @sum(turntrans_sc)/4

smpl @all
series conprod_00 = conprod_sc/conprod2000
series turnind_00 = turnind_sc/turnind2000
series turnret_00 = turnret_sc/turnret2000
series turnwho_00 = turnwho_sc/turnwho2000
series turntrans_00 = turntrans_sc/turntrans2000

group turntran turntrans_00 turntrans_sc turntrans
freeze(turntrans_graf) turntran.line(d)
freeze(turntrans_graf2) turntran.line(x)
show turntrans_graf
show turntrans_graf2
```

Do pracovného hárku importujeme všetky údaje, čo máme v štvrťročnej databáze Excel-ovského súboru, nakoľko vyberať správny stĺpec pre každý časový rad je zbytočne zdĺhavé. Stále ceny vypočítame jednoduchým vydelením pôvodného časového radu cenovým indexom. Výslednému časovému radu dáme príponu _sc (stále ceny). K novovytvorenému časovému radu by sme nemuseli dávať žiadnu príponu, avšak pri vzorci typu x = x + 1 by sa nám následne stratili hodnoty pôvodného časového radu. Pri chybách by sme museli opäť importovať pôvodné dáta z Excel-ovského súboru.

Uvedené skratky znamenajú:
conprod – stavebná produkcia v bežných cenách
turnind – tržby v priemysle v bežných cenách
turnret – tržby v maloobchode v bežných cenách
turnwho – tržby vo veľkoobchode v bežných cenách
turntrans – tržby v doprave a v skladovaní v bežných cenách
ppi – cenový index výrobcov
cpitrans – index spotrebiteľských cien v doprave

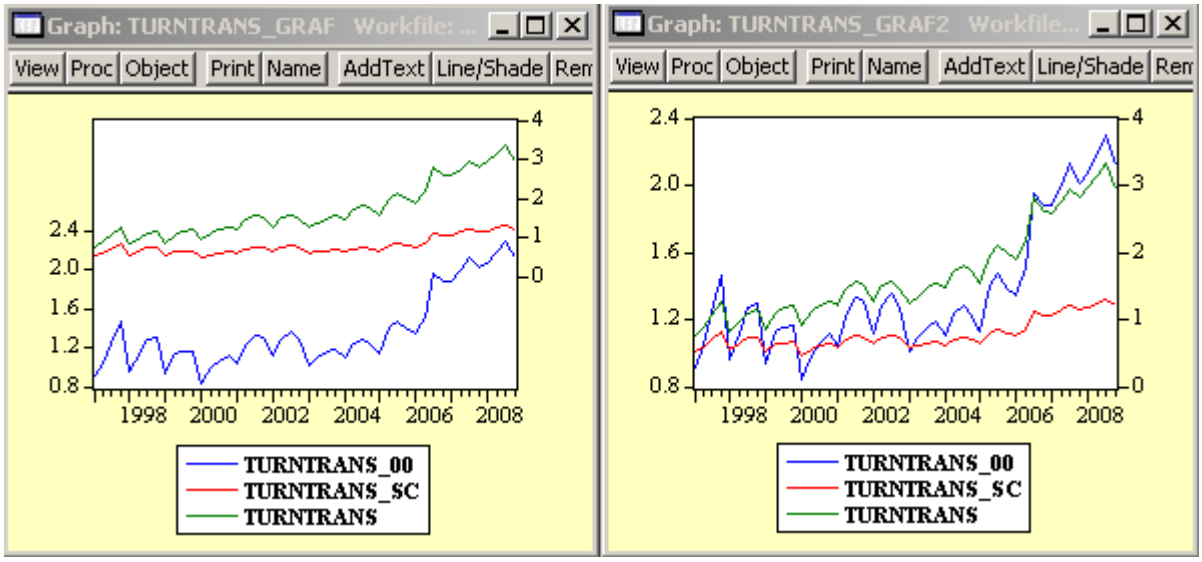
Pomocou príkazu *displayname* môžeme priradiť tieto popisy k jednotlivým časovým radom.

*_isc – znamená príslušný index v stálych cenách pre jednotlivé odvetvia,
*_00 – je finálny bázický index v stálych cenách priemerného roka 2000

Po vydelení jednotlivých časových radov v bežných cenách danými cenovými indexami máme k dispozícii časové rady v stálych cenách. Tieto chceme teraz transformovať na bázické indexy. V prvom rade musíme pre všetky časové rady vypočítať priemer roku 2000, teda bázický rok. K tomu využijeme príkaz SCALAR a postupujeme presne ako v opise tohto príkazu vyššie. Posledným krokom výpočtu je vydelenie časových radov v stálych cenách (_sc) vypočítanými skalárnymi veličinami. Netreba zabudnúť rozšíriť späť Sample na @all, aby výsledné bázické indexy boli prepočítané na celý časový úsek a nie iba pre obdobie 2000m1 až 2000m4, čo bol náš posledný stanovený Sample. Výsledným bázickým indexom v stálych cenách priradíme príponu 00. Výsledok jedného časového radu si pozrieme v grafe – obr. 2.10 (vľavo). Príkazu LINE bol priradený argument (options) ´n´ v zátvorke, ktorý znamená zobrazenie duálnej mierky – bez pretínania čiar. Na ľavej strane sú zobrazené hodnoty bázického indexu v stálych cenách – turntran_00 (všimnite si, že hodnota 1 sa nachádza približne v roku 2000 – bázický index). Na pravej strane sa nachádzajú hodnoty zostávajúcich časových radov – turntran_sc a pôvodný turntran. Ak by sme chceli zobraziť naraz všetky grafy a súčasne ponechať duálnu mierku, použili by sme v zátvorke príkazu line argument ´x´- obr. 2.10 (napravo).

Na to, aké jednoduché operácie sme na uvedených piatich časových radoch uskutočňovali, je program až príliš dlhý a náročný na písanie. V nasledujúcej kapitole skrátime tento program na 10 riadkov pomocou tzv. cyklov – FOR..NEXT.

Obr. 2.10 Tvorba grafov – duálna mierka



3. Kontrola spúšťania programov

Kontrola spúšťania programu znamená kontrolovať spúšťanie programu a príkazov tak, aby sa mohli určité sekvencie opakovať a rovnako, aby sa mohli zadávať podmienky vykonania určitých príkazov. Medzi príkazy kontroly spúšťania patria štandardné príkazy, ktoré nechýbajú v žiadnom programovacom jazyku: podmienka IF (..THEN..ELSE..ENDIF), cykly FOR...NEXT a podmienka WHILE..WEND.

3.1 Výraz IF

Slovo IF znamená AK (keď). Výraz používame, ak chceme, aby sa určité procesy, operácie, vykonali len vtedy, ak bude splnená určitá podmienka. To je prvá forma príkazu IF, ktorú sme si už popísali v stati o Sample súboru. Ďalšími formami príkazu IF sú zložené: IF...ENDIF, IF...ELSE...ENDIF. Podmienky využívajú takzvané relačné operátory, t.j. okrem ROVNÁ SA = aj nasledujúce: >, >=, <, <=, <>. Posledný znak znamená NEROVNÁ SA. Príkazy IF využívajú aj výrazy OR (alebo), THEN (potom) a AND (a).

IF...ENDIF – Príklad:

```
if !cislo1 > 1 then delete cislo2
endif
```

Za výrazom IF nasleduje podmienka – výraz, za výrazom nasleduje výrok THEN (potom) a nakoniec príkazy, ktoré sa majú vykonať, ak je podmienka správna (true). Celý príkaz IF musí byť ukončený slovom ENDIF, aby program vedel, kde sa končia príkazy, ktoré sú spojené s podmienkou. Ak je výraz za IF pravdivý (true) všetky príkazy po ENDIF sa vykonajú, ak je nepravdivý (false), všetky príkazy až po ENDIF budú ignorované a nič sa neudeje. Program by v tom prípade pokračoval ďalšími príkazmi po ENDIF výroku.

IF...ELSE... ENDIF – Príklad:

V prípade ELSE (inak) môžeme rozšíriť predošlý príklad nasledovne:

```
if !cislo1 > 1 then delete cislo2
else
        series cislo = cislo1*100 + cislo2
endif
```

Ak bude podmienka za IF pravdivá, program vykoná výraz nasledujúci za THEN, teda DELETE – vymaže časový rad CISLO2. Ak by podmienka nebola pravdivá, program odignoruje požiadavku. Avšak ak dáme do príkazu výraz ELSE, teda v preklade INAK, program vykoná všetky príkazy, ktoré sú za ELSE príkazom. V našom prípade sa tak stane vtedy, ak je daný skalár menší než jedna.

Teda všeobecne môžeme napísať:

```
AK....podmienka_správna.....potom...príkaz
else(podmienka_nesprávna)....príkaz
koniec
```

Do uvedenej podmienky vyššie (>1) nemôžeme dať jednoducho časový rad (kombinovať ho s relačnými operátormi), lebo program chápe časový rad ako maticu. V tomto prípade nie je jasné, ktoré pozorovanie daného časového radu sa má brať do úvahy, preto by každé pozorovanie v aktuálnom Sample nemalo byť väčšie než jedna (z príkladu vyššie). Preto keď Sample daného časového radu nie je stanovené na celý Range pracovného hárku a použijeme uvedenú podmienku spolu s časovým radom, program vypíše chybové hlásenie, že matica obsahuje NA pozorovanie, pritom ide o časový rad. V danom prípade si teda budeme musieť dávať pozor, keď budeme používať relačné operátory v kombinácii s časovými radmi, aby každé pozorovanie malo určitú hodnotu. Ak bude podmienka znieť IF casovyrad = 1, všetky hodnoty budú musieť byť rovné jednej (každé pozorovanie). Používanie časových radov a relačných operátorov je veľmi mätúce, preto odporúčame zatiaľ ho nepoužívať.

Ešte si ukážeme príklad použitia výrokov OR a AND:

```
if !cislo1 > 1 or < 0 then delete cislo2
else
        rename cislo2 cislo5
endif
```

V tomto prípade podmienka bude splnená vtedy, ak skalár bude mať hodnotu menšiu ako nula alebo väčšiu ako jedna. Ak bude jeho hodnota medzi 0 a 1, tak bude časový rad s názvom CISLO2 premenovaný.

```
if !cislo1 > 1 and !cislo3 < 0 then delete cislo2
else
        rename cislo2 cislo5
endif
```

V tomto prípade je podmienka splnená, ak platia obe podmienky súčasne! Teda skalár CISLO1 musí byť väčší ako jedna a zároveň skalár CISLO3 musí byť menší ako nula.

Príklad 3: Jednoduchá podmienka IF

Ukážeme si praktický príklad aplikácie príkazu IF a to v dvoch jeho formách. Prvú už poznáme, ide o stanovovanie časového rozpätia pracovného hárku a druhú formu použijeme IF..THEN. Pred nami stojí problém: máme k dispozícii časové rady s určitou dĺžkou a chceme použiť ich logaritmy. Niektoré však obsahujú mínusové hodnoty, preto ich nebude možné logaritmovať. Vzhľadom na počet a dĺžku časových radov by bolo veľmi náročné prezerať samotné časové rady jeden po druhom, preto tento problém vyriešime pomocou krátkeho programu.

K dispozícii máme časové rady ESI, DFA a TRANSPORT. Dané časové rady chceme logaritmovať. Jedná sa o bázické indexy. Ak sa nachádza v ich pozorovaniach záporná hodnota, kladné hodnoty dosiahneme tým, že prirátame k časovým radom hodnotu 100.

```
smpl if ESI <>na
if @min(ESI) <= 0 then
series ESI = ESI + 100
endif

smpl if DFA <>na
if @min(DFA) <= 0 then
series DFA = DFA + 100
endif

smpl if TRANSPORT <>na
if @min(TRANSPORT) <= 0 then
series TRANSPORT = TRANSPORT + 100
endif
```

V prvom kroku stanovíme časové obdobie pracovného hárku tak, aby sa v hodnotách nenachádzali NA pozorovania – prázdne pozorovania. V tomto prípade použijeme kombináciu príkazu SMPL a IF, ktorú sme sa už naučili používať. Ďalej pomocou príkazu @min zistíme minimálnu hodnotu daného časového radu. Ak bude táto kladná, tak nebude treba prirátavať k hodnotám časového radu 100. Ak bude minimálna hodnota záporná, prirátame k celému časovému radu 100. Príkaz začínajúci sa podmienkou IF a pokračujúci prevedením príkazu za THEN, v prípade ak je podmienka pravdivá, t.j. časový rad má záporné hodnoty, musíme ukončiť výrazom ENDIF. Ak neobsahuje časový rad záporné hodnoty, program nevykoná žiadny príkaz. Ak by sme chceli vykonať príkaz po nesplnení podmienky, museli by sme zadať príkaz ELSE medzi THEN a ENDIF. V príklade vyššie je vidieť, že sme použili rovnaké príkazy pre všetky časové rady. Tento príklad sa dá však napísať jednoduchšie, bez toho, aby sme museli vypisovať pre každý časový rad tie isté príkazy. Ako, to si ukážeme pri cykloch FOR..NEXT.

3.2 FOR..NEXT cyklus

Obsahom každého programovacieho jazyka sú aj tzv. cykly – loops. Umožňujú opakované vykonanie určitého príkazu. V nasledujúcom texte si vysvetlíme cyklus FOR..NEXT, ktorý nám umožní výrazne skrátiť program uvedený v predošlej podkapitole.

Cyklus začína výrazom FOR a končí výrazom NEXT, všetky príkazy medzi nimi sa opakujú podľa tzv. FOR counter (počítadlo), ktoré nasleduje hneď za výrazom FOR. Existujú dva druhy FOR...NEXT cyklov. Prvý využíva tzv. String variables – reťazcové premenné a druhý Control variables (scalars) – kontrolné premenné (skalárne).

Všeobecný výraz pre FOR...NEXT cyklus je:

```
for počítadlo=štart to koniec [veľkosť_kroku]
..
..
next
```

Counterom resp. počítadlom je práve reťazcová alebo kontrolná premenná.

3.2.1 For...NEXT cyklus a reťazcová (string) premenná

V prípade reťazcovej premennej sa nepoužíva forma cyklu FOR...TO...NEXT ale iba FOR...NEXT. Výraz TO sa používa iba pri kontrolnej premennej. Všeobecný výraz pre FOR...NEXT cyklus je teda nasledujúci:

```
for %reťazcová_premenná1  %reťazcová_premenná2 ... objekt1 objekt2...
...
...
next
```

Reťazcová premenná začína znakom % a za ním nasleduje názov reťazcovej premennej –

```
%názov
```

Názvom môže byť text i číslo. Za príkazom FOR napíšeme názov reťazca (napr. %1). Všetko čo bude nasledovať za názvom reťazca bude jeho obsahom. Napríklad:

```
for %1 conprod turnind
series {%1}
next
```

Takýto príkaz vytvorí dva časové rady – CONPROD a TURNIND, ktorý bude mať v tomto prípade všetky hodnoty NA. CONPROD a TURNIND sú obsahom reťazcovej premennej.

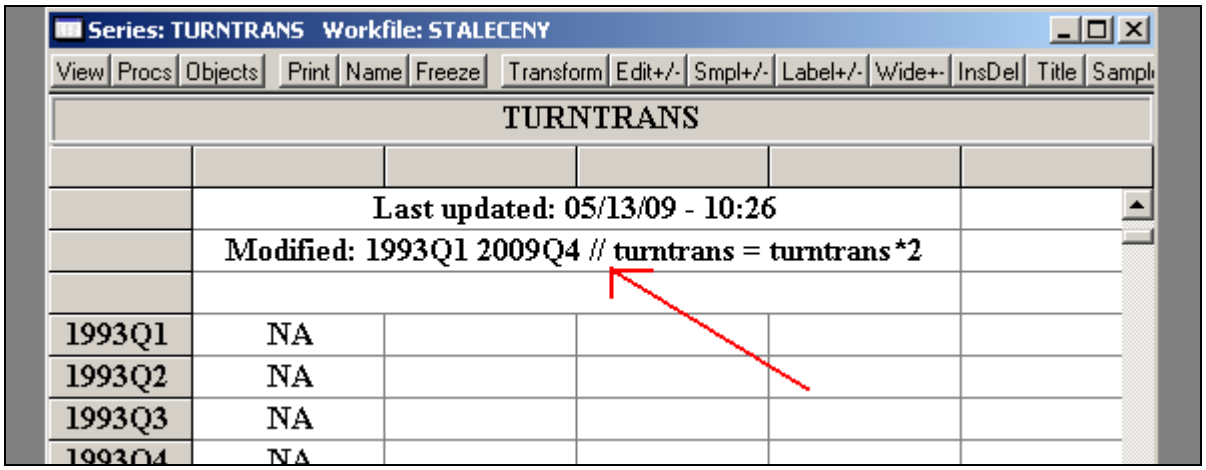
Ak sme používali v stati o reťazcoch výraz %retazec = „text“, tak v tomto prípade nepoužívame úvodzovky, lebo nejde o reťazec, ale o reťazcové premenné, ktoré v našom prípade budú predstavovať názvy objektov (časových radov). Teda v uvedenom príkaze píšeme FOR %1 conprod turnind, a nie FOR %1 “conprod“ a “turnind“.

K reťazcovej premennej, ktorú sme nazvali %1 je priradená v každom cykle iba jedna premenná vymenovaná za ním. Tú program dosadí všade na miesto, kde nájde po výroku FOR v programe znak %1. V príkazoch medzi výrokmi FOR a NEXT sa uvádza reťazcová premenná v zloženej zátvorke {}. Uvedieme si úplne jednoduchý program. Chceme vynásobiť časové rady CONPROD, TURIND, TURNRET, TURNWHO a TURNTRANS hodnotou 2. Keby sme nepoznali príkaz FOR..NEXT, vypisovali by sme každý príkaz pre každý časový rad samostatne –

```
series conprod = conprod*2,
series turnind = turnind*2
atď.
```

Chceme zdvojnásobiť hodnoty v každom časovom rade, pričom pôvodný časový rad sa stratí, lebo nový časový rad bude pomenovaný rovnako ako predošlý. V prípadoch dlhších programov bude lepšie, ak nebudeme nazývať nový časový rad rovnako ako pôvodný, lebo môžeme spraviť výpočtovú chybu pri analýzach. Či ide o pôvodný časový rad alebo rad po operáciách ľahko zistíme, ak si otvoríme daný časový rad – obr. 3.1. V oblasti nad údajmi sa nachádza informácia, akými poslednými operáciami daný časový rad prešiel (dátum, čas, výraz). Problémom sa vyhneme, ak jednoducho časový rad nazveme iným menom. Nižšie sú vysvetlené oba prípady.

Obr. 3.1 Okno s operáciami časového radu



Zdvojnásobenie daných časových radov si zjednodušíme tým, že použijeme reťazcovú premennú, ktorú si nazveme napríklad %retazec – v budúcnosti je lepšie použiť kratšie a jednoduchšie názvy reťazcov – napr. označovanie číslami. Potom:

Prvý prípad pre rovnaký názov časového radu:

```
for %retazec conprod turnind turnret turnwho turntrans
        series {%retazec} = {%retazec}*2
next
```

V druhom prípade však využijeme jednoduchý trik. Nový časový rad vytvoríme z pôvodného tým, že pridáme na koniec názvu číslo 1. V predošlom príklade vykonanie prvého člena reťazca CONPROD prebiehalo v prvom cykle nasledovne:

```
series conprod = conprod*2
```

Našim cieľom však je

```
series conprod1 = conprod*2
```

To znamená, že jednoducho pridáme príponu hneď za koniec zloženej zátvorky reťazca:

```
series {%retazec}1 = {%retazec}*2
```

Reťazcová premenná pracuje s reťazcami – teda vlastne s reťazcami textov (ako bolo vysvetlené v kapitole o reťazcových premenných). Nášmu programu je jedno kam daný text dosadíme. V tomto prípade sa stáva text reťazca súčasťou tradičného príkazu

```
series názov_časového_radu = výraz
```

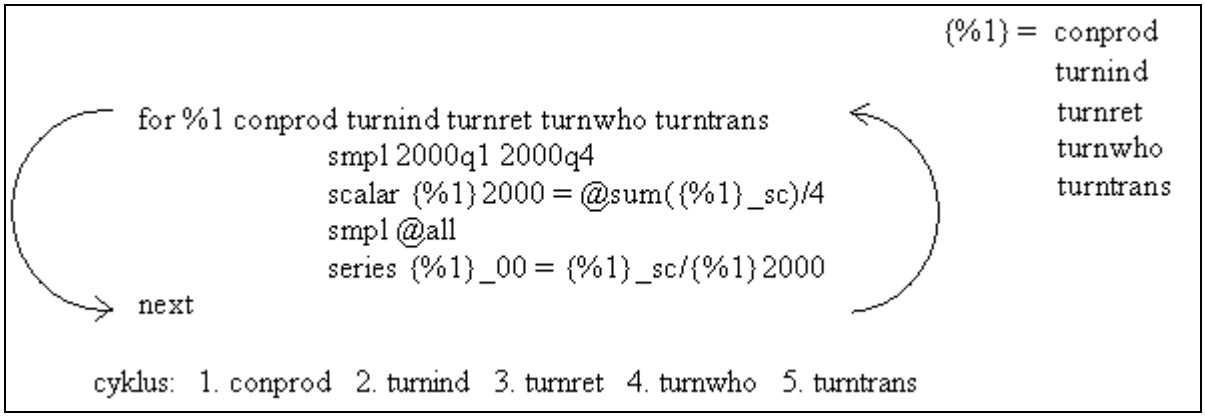
Druhý prípad pre nový názov časového radu bude vyzerať teda nasledovne:

```
for %retazec conprod turnind turnret turnwho turntrans
        series {%retazec}1 = {%retazec}*2
next
```

V príklade o stálych cenách musíme uvedených päť časových radov prepočítať do stálych cien, teda sa jedná o delenie príslušnými cenovými indexami. Tento krok necháme v pôvodnom stave. Ďalšie dva kroky – t.j. výpočet priemerného bázického obdobia a výpočet bázického indexu sa dajú skrátiť nasledovne:

```
for %1 conprod turnind turnret turnwho turntrans
        smpl 2000q1 2000q4
        scalar {%1}2000 = @sum({%1}_sc)/4
        smpl @all
        series {%1}_00 = {%1}_sc/{%1}2000
next
```

Obr. 3.2 Grafické zobrazenie cyklu FOR..NEXT



Program sa spustí v nasledujúcich 5 krokoch – cykloch. Každý sa začína výrokom FOR a končí výrokom NEXT:

```
1.                                                          for
        smpl 2000q1 2000q4
        scalar conprod2000 = @sum(conprod_sc)/4
        smpl @all
        series conprod_00 =  conprod_sc/conprod2000         next

2.                                                          for
        smpl 2000q1 2000q4
        scalar turnind2000 = @sum(turnind_sc)/4
        smpl @all
        series turnind_00 =  turnind_sc/turnind2000         next

3.                                                          for
        smpl 2000q1 2000q4
        scalar turnret2000 = @sum(turnret_sc)/4
        smpl @all
        series turnret_00 =  turnret_sc/turnret2000         next
```

```
4.                                                                  for
        smpl 2000q1 2000q4
        scalar turnwho2000 = @sum(turnwho_sc)/4
        smpl @all
        series turnwho_00 =  turnwho_sc/turnwho2000                 next

5.                                                                  for
        smpl 2000q1 2000q4
        scalar turntrans2000 = @sum(turntrans_sc)/4
        smpl @all
        series turntrans_00 =  turntrans_sc/turntrans2000           next
```

Celý príklad (Príklad 2) zo strany 38 bude vyzerať nasledovne. Pridali sme aj vytvorenie grafov pre porovnanie stálych cien a bázických indexov pre všetky časové rady:

```
cd "w:\klucik\data"
workfile staleceny q 1993 2009
smpl 1993q1 2009q4
read(t=xls, s=all3) "w:/klucik/data/data_quarterly.xls" 148
read(t=xls, s=all4) "w:/klucik/data/data_quarterly.xls" 90

series conprod_sc = conprod/conprod_isc
series turnind_sc = turnind/ppi
series turnret_sc = turnret/turnret_isc
series turnwho_sc = turnwho/ppi
series turntrans_sc = turntrans/cpitrans

for %1 conprod turnind turnret turnwho turntrans
        smpl 2000q1 2000q4
        scalar {%1}2000 = @sum({%1}_sc)/4
        smpl @all
        series {%1}_00 = {%1}_sc/{%1}2000
        group g{%1} {%1} {%1}_00 {%1}_sc
        freeze({%1}graf) g{%1}.line(x)
        show {%1}graf
next
```

Ukážeme si aj príklad riešený v predošlej podkapitole (Príklad 3, str. 35). Vtedy sme tri časové rady testovali na záporné hodnoty pozorovaní. Teraz rovnaký program dokážeme napísať pomocou FOR..NEXT cyklu a reťazcovej premennej nasledovne:

```
for %1 ESI DFA TRANSPORT
smpl if {%1} <>na
if @min({%1}) <= 0 then
series {%1} = {%1} + 100
endif
next
```

Jediná zmena bola, že sme vymenili názvy časových radov za názov reťazca v zloženej zátvorke – {%1} a v prvom riadku za výrazom FOR sme definovali, ktoré časové rady sa majú vystriedať v názve %1. Na konci sme cyklus ukončili výrazom NEXT.

3.2.2 For...NEXT cyklus a kontrolná premenná (skalárna)

Všeobecný výraz pre cyklus FOR..TO..NEXT pre kontrolnú premennú vyzerá nasledovne:

```
for !názovskaláru = štart to koniec [veľkosť_kroku]
..
..
next
```

Na rozdiel od reťazcovej premennej sa skalárna (kontrolná) premenná začína výkričníkom ´!´. Po výkričníku nasleduje názov skalárnej premennej. Často sa používa iba jedno písmeno napr. FOR !a. Po názve skalárnej premennej nasleduje výraz OD...DO, teda napr. 1 TO 10 (od jedna do desať). Napríklad, ak chceme vytvoriť 10 časových radov s názvom CONPROD1, CONPROD2..až CONPROD10, príkaz bude vyzerať nasledovne:

```
for !x = 1 to 10
series conprod{!x}
next
```

Daný príkaz vytvorí 10 časových radov, ktoré budú obsahovať iba NA pozorovania. V každom cykle opäť priradí premennej !x iba jednu hodnotu. V prvom cykle to bude 1, v druhom 2 atď. V tomto prípade chceme priradiť k názvu CONPROD číslo, tak uvádzame skalárnu premennú v zloženej zátvorke, rovnako ako predtým reťazcovú premennú. Rovnako ste si všimli, že všeobecný syntax príkazu FOR..TO.. vyššie obsahuje v hranatej zátvorke aj výraz STEP (krok). V predošlom príklade bola krokom jednotka, táto veľkosť kroku je prednastavená (default). Cyklus sa rátal po jednom kroku, to znamená 1, 2, 3 atď. až 10. Ak by sme mali zápis FOR !x = 10 to 1, a očakávali by sme, že cyklus pôjde od 10 do 1 po jednom kroku – 10, 9, 8, 7...1. museli by sme dopísať výraz STEP –1:

```
for !x = 10 to 1 step –1
series conprod{!x}
next
```

Výsledok – názvy časových radov – by však bol úplne rovnaký ako v predošlom príklade.

Teraz si však uvedieme zložitejší prípad, keď použijeme najčastejšie využívaný program pri práci so skupinami. Použijeme rovnaký príklad ako v príkladoch na strane 38 a 48 (Príklad 2). Výpočet bázického indexu môžeme spraviť aj nasledovne: vytvoríme si skupinu časových radov – Group s názvom STALECENY tvorenú zo všetkých piatich časových radov ako v predošlom príklade. Následne použijeme príkaz FOR..NEXT, avšak tentoraz budeme kombinovať reťazcovú premennú a kontrolnú premennú:

```
group staleceny conprod turnind turnret turnwho turntrans
for !y = 1 to staleceny.@count
%1 = staleceny.@seriesname(!y)
        smpl 2000q1 2000q4
        scalar {%1}2000 = @sum({%1}_sc)/4
        smpl @all
        series {%1}_00 = {%1}_sc/{%1}2000
       group g{%1} {%1} {%1}_00 {%1}_sc
        freeze({%1}graf) g{%1}.line(x)
        show {%1}graf
next
```

V príklade sú použité dva nové príkazy: .@count a .@seriesname. Oba príkazy slúžia ako prípona k názvu skupiny:

```
názov_skupiny.@count
názov_skupiny.@seriesname(i)
```

Príkaz .@count získa počet časových radov danej skupiny. Príkaz .@seriesname(i) dokáže extrahovať názov i-teho časového radu v danej skupine.

Výraz

```
for !y = 1 to staleceny.@count
```

znamená

```
for !y = 1 to počet_časových_radov_v_skupine_STALECENY,
```

v našom prípade teda

```
for !y = 1 to 5
```

Výraz

```
%1 = staleceny.@seriesname(!y)
```

vytvorí reťazec (teda text), ktorý bude tvoriť i-ty časový rad skupiny STALECENY. V našom prípade má skupina päť členov. To znamená, že v prvom cykle bude výraz %1 znamenať meno prvého časového radu a v poslednom cykle piateho časového radu (1 to 5). Výraz %1 sme si vysvetlili v predošlej kapitole a jednotka znamená názov reťazca, ktorý sme zvolili sami, a je bez významu. Spustenie programu bude vyzerať úplne rovnako ako spustenie programu na strane 38 a 40. Jednotlivé cykly však budú vyzerať v príkaze FOR nasledovne:

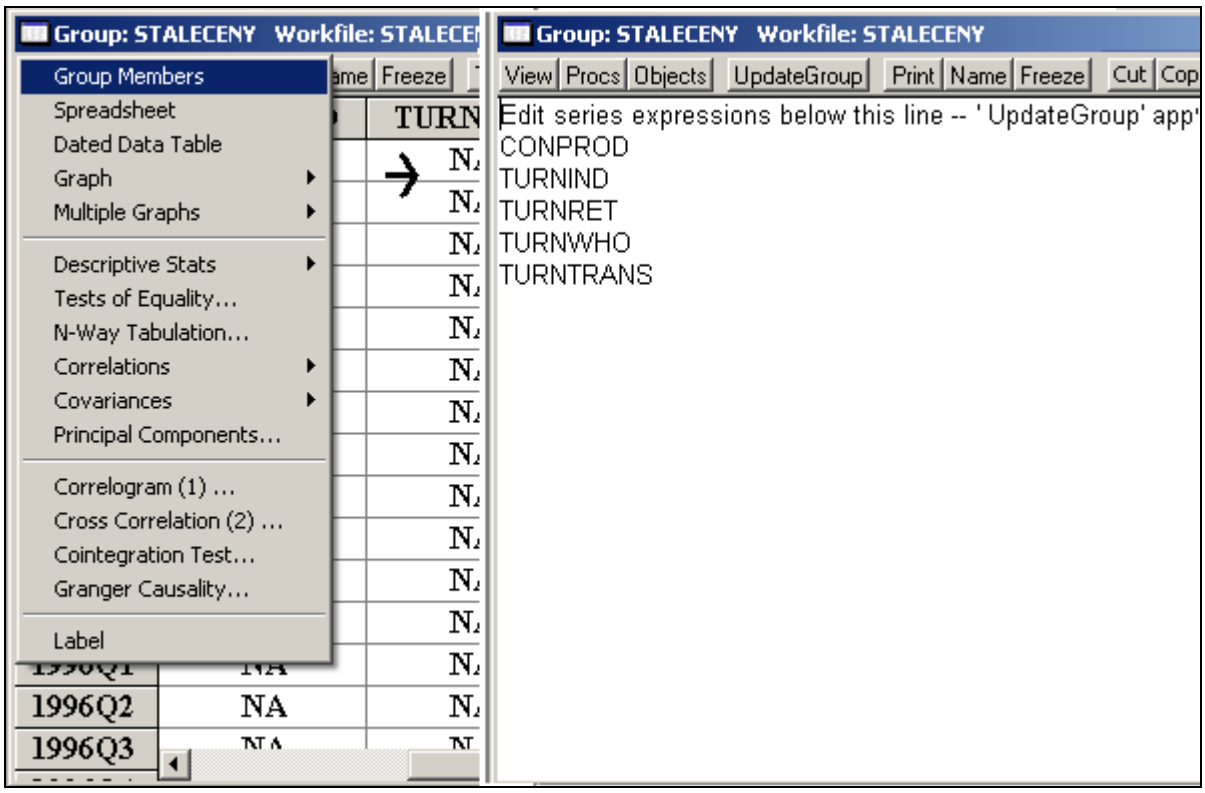
```
for !y = 1 to 5
%1 = staleceny.@seriesname(1)   - t.j. conprod
%1 = staleceny.@seriesname(2)   - t.j. turnind
.
.
%1 = staleceny.@seriesname(5)   - t.j. turntrans
next
```

Postupnosť je daná poradím jednotlivých časových radov v skupine, tú si môžeme zistiť cez menu danej skupiny – obr. 3.3 na nasledujúcej strane. Ako vidíme z anglického popisu na pravej strane obrázku - ´Edit series expressions...´, časové rady môžeme editovať, t.j. aj meniť ich poradie. Prednastavené (pôvodné, originálne) poradie je také, aké sme napísali pri tvorbe danej skupiny časových radov.

Rovnako ako pri cykle FOR..NEXT a reťazcovej premennej si teraz na skalárnej premennej a prostredníctvom práce so skupinou zopakujeme rovnaký príklad ako Príklad 3 na strane 43 – výsledný program vyzerá nasledovne:

```
group rady ESI DFA TRANSPORT
for !x = 1 to rady.@count
%1 = rady.@seriesname(!x)
smpl if {%1} <>na
if @min({%1}) <= 0 then
series {%1} = {%1} + 100
endif
next
```

Obr. 3.3 Členovia skupiny



Rozdiel oproti predošlému príkladu na strane 50 s reťazcovou premennou je, že sme do cyklu FOR..NEXT nezaradili definíciu reťazca hneď za výrazom FOR, ale reťazec %1 sme definovali pomocou vytvorenia skupiny a príkazov @count (počet časových radov v skupine) a @seriesname (názov časového radu v skupine). S pomocou skalárnej veličiny !x sme zabezpečili, aby sa v cykle FOR..NEXT vystriedali všetky časové rady skupiny RADY od prvého (for 1 to...) až po posledný (@count).

Príklad 4: Viacnásobná definícia reťazcov v cykle FOR..NEXT

V ďalšom príklade si uvedieme ukážku použitia viacnásobnej definície reťazcov pri jednom FOR výraze a zároveň prvú jednoduchú ukážku vnoreného cyklu FOR..NEXT v inom FOR..NEXT cykle. Príklad sa týka opäť výpočtu stálych cien a zároveň aj bázických indexov na báze roku 2005. Postup je jednoduchý, najprv vydelíme časové rady v bežných cenách ich príslušným indexom stálych cien, vypočítame priemernú hodnotu časového radu za rok 2005 (bázu) a vydelením všetkých pozorovaní touto bázou dostaneme bázický index. Problémom je však index stálych cien. Tento index je zverejňovaný ako medziročný index v stálych cenách a zároveň v cenách roku 2005, preto nemôžeme vykonať jednoduchú operáciu delenia bežných cien indexom stálych cien, ale stále ceny musíme rátať postupne za každý rok dozadu a dopredu od roku 2005. Prvým krokom je výpočet hodnoty stálych cien za rok 2005. Tento môžeme dostať rôznymi voliteľnými spôsobmi. Nakoniec sme si zvolili ponechať hodnoty v bežných cenách. Následne postupne za každý rok spätne od 2005 vydelíme hodnoty bežných cien indexom v stálych cenách z predošlého roku. Stále ceny smerom od roku 2006 do 2010 nemusíme robiť postupne, lebo už máme k dispozícii hodnoty z predošlého roku, a teda všetky hodnoty v bežných cenách vynásobíme indexom stálych cien *_isc. V záverečnej fáze pomocou príkazu @elem vyrátame priemerný rok 2005 v stálych cenách a touto hodnotou budú vydelené všetky pozorovania daného časového radu. Výsledok bude bázický index v stálych cenách roku 2005.

```
for %1 2004 2003 2002 2001 2000 1999 1998 1997 1996 1995 1994 1993
    for %2 %3 conprod conprod_isc conabr conabr_isc connew connew_isc coninl
    coninl_isc conrep conrep_isc turnveh turnveh_isc turnvehA turnvehA_isc turnret
    turnret_isc turnretA turnretA_isc turnretB turnretB_isc turnretC turnretC_isc
    turnretD turnretD_isc turnretE turnretE_isc turnretF turnretF_isc turnretG
    turnretG_isc turnretH turnretH_isc turnretI turnretI_isc

    smpl 2005m1 2005m12
    series {%2}_sc = {%2}
    smpl {%1}m1 {%1}m12
    series {%2}_sc = {%2}_sc(12)/{%3}(12)
    smpl 2006m1 2010m12
    series {%2}_sc = {%2}_sc(-12)*{%3}

    smpl @all
    scalar {%2}_sc_00=(@elem({%2}_sc, "2005m1")+@elem({%2}_sc,
    "2005m2")+@elem({%2}_sc, "2005m3")+@elem({%2}_sc,
    "2005m4")+@elem({%2}_sc, "2005m5")+@elem({%2}_sc,
    "2005m6")+@elem({%2}_sc, "2005m7")+@elem({%2}_sc,
    "2005m8")+@elem({%2}_sc, "2005m9")+@elem({%2}_sc,
    "2005m10")+@elem({%2}_sc, "2005m11")+@elem({%2}_sc, "2005m12"))/12
    series {%2}_00 = {%2}_sc/{%2}_sc_00
    next
next
```

Dvojitá definícia FOR..NEXT reťazca v našom príklade, teda – for %2 %3 – prebieha rovnako ako cyklus FOR s jedným reťazcom FOR %1, s tým rozdielom, že sa striedajú v každom cykle naraz aj %2 aj %3. Názorne si ukážeme cykly z vnútorného cyklu príkladu:

```
        for %2 %3 conprod conprod_isc conabr conabr_isc connew connew_isc coninl
        coninl_isc conrep conrep_isc turnveh turnveh_isc turnvehA turnvehA_isc turnret
        turnret_isc turnretA turnretA_isc turnretB turnretB_isc turnretC turnretC_isc
        turnretD turnretD_isc turnretE turnretE_isc turnretF turnretF_isc turnretG
        turnretG_isc turnretH turnretH_isc turnretI turnretI_isc
```

1. cyklus: %2=conprod %3=conprod_isc
2. cyklus: %2=conabr %3=conabr_isc
3. cyklus: %2=connew %3=connew_isc
4. cyklus: %2=coninl %3=coninl_isc

atď.

Ďalej si načrtneme ako fungujú vnorené FOR..NEXT cykly. V programe máme dva cykly FOR..NEXT:

```
for %1 2004 2003 2002 2001 2000 1999 1998 1997 1996 1995 1994 1993
    for %2 %3 conprod conprod_isc conabr conabr_isc connew connew_isc coninl
    coninl_isc conrep conrep_isc turnveh turnveh_isc turnvehA turnvehA_isc turnret
    turnret_isc turnretA turnretA_isc turnretB turnretB_isc turnretC turnretC_isc
    turnretD turnretD_isc turnretE turnretE_isc turnretF turnretF_isc turnretG
    turnretG_isc turnretH turnretH_isc turnretI turnretI_isc
    ...
    next
next
```

Vždy „vyšší“ cyklus má prednosť, t.j. najprv sa začne cyklus rokom 2004, spúšťa sa všetko, čo je vnútri cyklu, t.j. celý FOR cyklus vnútri (%2, %3 pre všetky časové rady) a vracia sa späť do nadradenejšieho cyklu, ktorý pokračuje rokom 2003 – obr. 3.4 (ďalšia strana). Pri nadradenejšom druhom cykle sa v našom príklade opakuje príkaz vytvorenia priemerného roku 2005 viackrát pre ten istý časový rad, avšak to nemá vplyv na výsledok. Vnorením FOR..NEXT funkcie sme si tak či tak uľahčili prácu, ako keby sme mali osobitne vypisovať viacero cyklov.

Reťazcový cyklus FOR..NEXT môže mať okrem dvoch definícií (for %2 %3) i viacero definícií, napr.

```
for %1 %2 %3 %4 %5 %6 conprod1 conprod2 conprod3 conprod4 conprod5 conprod6
turnind1 turnind2 turnind3 turnind4 turnind5 turnind6
```

V tomto prípade spustenie cyklu prebieha rovnako. V prvom cykle sa dosadia vždy všetky *conprod*, v druhom cykle všetky *turnind* objekty.

Obr. 3.4 Priebeh vnorených cyklov FOR..NEXT

1. 2.

```
for %1 2004 2003 2002 2001 2000 1999 1998 1997 1996 1995 1994 1993
```

1a. 1a. 1b. 1b. 1c. 1c. . . .

```
    for %2 %3 conprod conprod_isc conabr conabr_isc connew connew_isc coninl
    coninl_isc conrep conrep_isc turnveh turnveh_isc turnvehA turnvehA_isc turnret
    turnret_isc turnretA turnretA_isc turnretB turnretB_isc turnretC turnretC_isc
    turnretD turnretD_isc turnretE turnretE_isc turnretF turnretF_isc turnretG
    turnretG_isc turnretH turnretH_isc turnretI turnretI_isc
    ...
    next
next
```

1a. cyklus

```
smpl 2005m1 2005m12
series conprod_sc = conprod
smpl 2004m1 2004m12
series conprod_sc = conprod_sc(12)/conprod_isc(12)
smpl 2006m1 2010m12
series conprod_sc = conprod_sc(-12)*conprod_isc

smpl @all
scalar conprod_sc_00=(@elem(conprod_sc, "2005m1")+@elem(conprod_sc,
"2005m2")+@elem(conprod_sc, "2005m3")+@elem(conprod_sc,
"2005m4")+@elem(conprod_sc, "2005m5")+@elem(conprod_sc,
"2005m6")+@elem(conprod_sc, "2005m7")+@elem(conprod_sc,
"2005m8")+@elem(conprod_sc, "2005m9")+@elem(conprod_sc,
"2005m10")+@elem(conprod_sc, "2005m11")+@elem(conprod_sc, "2005m12"))/12
series conprod_00 = conprod_sc/conprod_sc_00
```

↓

1b. cyklus, 1c. cyklus, 1d. cyklus...

↓

2a. cyklus

```
smpl 2005m1 2005m12
series conprod_sc = conprod
smpl 2003m1 2003m12
series conprod_sc = conprod_sc(12)/conprod_isc(12)
smpl 2006m1 2010m12
series conprod_sc = conprod_sc(-12)*conprod_isc

smpl @all
scalar conprod_sc_00=(@elem(conprod_sc, "2005m1")+@elem(conprod_sc,
"2005m2")+@elem(conprod_sc, "2005m3")+@elem(conprod_sc,
"2005m4")+@elem(conprod_sc, "2005m5")+@elem(conprod_sc,
"2005m6")+@elem(conprod_sc, "2005m7")+@elem(conprod_sc,
"2005m8")+@elem(conprod_sc, "2005m9")+@elem(conprod_sc,
"2005m10")+@elem(conprod_sc, "2005m11")+@elem(conprod_sc, "2005m12"))/12
series conprod_00 = conprod_sc/conprod_sc_00
```

...

3.2.3 Aplikácia cyklov pri nastavovaní rozpätia súboru

Cykly môžeme použiť aj v prípade nastavovaní Sample v pracovnom hárku. Tento nástroj sa zíde najmä ak potrebujeme častokrát meniť Sample alebo potrebujeme vykonať výpočty pri rôzne nastavených časových intervaloch. Pri Sample môžeme rovnako ako pri práci s časovými radmi využiť reťazcové i skalárne premenné.

V prvom príklade využijeme skalárnu premennú, ktorú nazveme !horizon. Cyklus bude prebiehať od čísla 10 po číslo 72 samozrejme po kroku jedna, keďže nedefinujeme krok inej hodnoty. V ďalšom riadku nasleduje príkaz SMPL, podobný, ako sme si už uvádzali (str. 26 – „smpl 1991m1 1991m1+!rast"), teda začiatok i koniec časového rozpätia sú rovnaké, avšak po konci časového rozpätia máme napísaný výraz plus a názov našej skalárnej premennej. To znamená, že v každom cykle sa bude posúvať Sample pracovného hárku o jedno pozorovanie, pričom začne posunom o 10 pozorovaní oproti začiatku Sample a skončí pri 72 pozorovaní po začiatku. Nasleduje potom príkaz, ktorý si vysvetlíme až v ďalšej kapitole – EQUATION, teda rovnica. Názov rovnice bude obsahovať vždy číslo z daného cyklu, preto budeme ľahšie môcť identifikovať odhadnutú rovnicu a Sample, na ktorom bola odhadnutá.

```
for !horizon=10 to 72

smpl 1970:1 1970:1+!horizon

equation eq{!horizon}.ls sales c orders

next
```

Nasledujúci príklad je podobný, pričom využíva reťazcovú premennú a cyklus FOR...NEXT. Obsahom reťazca sú až tri definície reťazca: %1, %2 a %3. Pričom ak sa pozornejšie pozrieme na definíciu v prvom riadku za výrazom FOR, vidíme, že %1 je vždy začiatok časového rozpätia, %2 koniec časového rozpätia a reťazec %3 je slovom (early, mid, late), ktoré vchádza do názvu objektu rovnice neskôr. Prvé reťazce %1 a %2 sa nachádzajú iba v príkaze SMPL na stanovenie časového rozpätia pracovného súboru, pri ktorom sa majú odhadovať rovnice. Na rozdiel od predošlého príkladu nám nevzniká 63 rovníc ale iba 3, keďže nasledujúci príklad má iba tri cykly.

```
for %1 %2 %3 1955:1 1960:4 early 1970:2 1980:3 mid 1975:4 1995:1 late

smpl %1 %2

equation {%3}eq.ls sales c orders

next
```

3.3 Cyklus WHILE..WEND

Cyklus IF vykoná príkazy, ak sú splnené určité podmienky. Cyklus FOR zasa umožňuje opakovať príkazy pri určených veľkostiach reťazcovej alebo kontrolnej premennej. Posledný z cyklov softvéru EViews WHILE..WEND umožňuje opakovať určité príkazy v prípade, ak je splnená podmienka. Teda na rozdiel od IF cyklu umožňuje opakovanie príkazov, a na rozdiel od FOR cyklu je flexibilný s ohľadom na stanovené podmienky vykonania príkazov.

Všeobecný tvar cyklu WHILE..WEND začína výrokom WHILE a končí sa výrokom WEND. Všetky príkazy vnútri tohto cyklu sú vykonané, pokiaľ platí podmienka za výrokom WHILE na začiatku cyklu. Podmienka sa stanovuje pomocou kontrolnej premennej a logických podmienok. Cyklus WHILE obsahuje najčastejšie prírastkové a úbytkové premenné, ktoré budú podrobnejšie vysvetlené v kapitole 3.5. Cyklus WHILE má 4 časti, prvou časťou je definovanie kontrolnej premennej, ktoré sa musí nachádzať pred začiatkom cyklu. Kontrolná premenná používaná na stanovenie podmienky v cykle musí byť predtým definovaná. Druhou časťou cyklu je výraz WHILE a podmienka. Treťou časťou sú výrazy vo vnútri cyklu a poslednou časťou je koniec cyklu označený ako WEND.

```
definovanie kontrolnej premennej
while (kontrolná premenná)podmienka_správna.....potom..
.....príkazy....
wend
```

Príklad použijeme opäť rovnaký ako pri reťazcovej premennej v cykle FOR...NEXT. Cieľom je vypočítať bázické indexy zo stálych cien odvetvových štatistík:

```
group staleceny conprod turnind turnret turnwho turntrans
!y = 0
while !y < 5
        !y = !y + 1
        %1 = staleceny.@seriesname(!y)
        smpl 2000q1 2000q4
          scalar {%1}2000 = @sum({%1}_sc)/4
          smpl @all
          series {%1}_00 = {%1}_sc/{%1}2000
                group g{%1} {%1} {%1}_00 {%1}_sc
          freeze({%1}graf) g{%1}.line(x)
          show {%1}graf
wend
```

Nahradené oproti programu pre cyklus FOR...NEXT boli iba tieto dva riadky:

```
for !y = 1 to staleceny.@count
...
next
```

namiesto toho boli použité tieto príkazy:

```
!y = 0
while !y < 5
        !y = !y + 1
.....
wend
```

Presne podľa definície cyklu WHILE...WEND najprv bola definovaná kontrolná premenná !Y. Potom bola definovaná podmienka, dokedy má program bežať: V našom prípade do toho bodu v čase, keď y bude stále menšie ako 5. Podľa príkladu pri cykle FOR..NEXT, sme potrebovali dostať do zátvorky namiesto y! - @seriesname(!y), poradie resp. číslo časových radov vo vytvorenej skupine STALECENY. Teda potrebujeme čísla 1, 2, 3, 4 a 5, lebo máme v skupine 5 časových radov. Máme dve možnosti, aby sa !Y rovnalo týmto piatim hodnotám. Necháme Y! prírastkovo stúpať o jednu jednotku od 1, alebo úbytkami klesať o jednu jednotku od hodnoty 5. To znamená použijeme prírastkovú, resp. úbytkovú premennú:

```
!y = !y +1    resp.   !y = !y – 1
```

V prvom prípade, ak chceme aby prvá hodnota premennej !Y bola 1, musíme definovať v prvom riadku !y = 0, tak aby v prvom cykle:

```
!y = 0 + 1.
```

V druhom prípade (úbytková premenná), musíme stanoviť hodnotu !Y na 6, tak aby v prvom cykle:

```
!y = 6 – 1
```

– bola hodnota 5. Takto v oboch prípadoch bude v každom cykle (ktorých bude spolu 5) hodnota 1 až 5. Spustenie programu vyzerá teda nasledovne:

```
!y = 0
!y = 1                                                    (!y = 0 + 1)
%1 = staleceny.@seriesname(1)   - conprod
!y = 2                                                    (!y = 1 + 1)
%1 = staleceny.@seriesname(2)   - turnind
!y = 3                                                    (!y = 2 + 1)
%1 = staleceny.@seriesname(3)   - turnret
!y = 4                                                    (!y = 3 + 1)
%1 = staleceny.@seriesname(4)   - turnwho
!y = 5                                                    (!y = 4 + 1)
%1 = staleceny.@seriesname(5)   - turntrans
                                    (!y = 5 teda je menšie ako 5 – koniec cyklu)
wend
```

To, že v poslednom piatom cykle je premenná !y už rovná piatim, je síce porušenie podmienky WHILE, ale program kontroluje plnenie podmienky vždy na začiatku cyklu – a na začiatku cyklu bola hodnota !Y rovná ešte štyrom.

Pomocou cyklu WHILE sme prišli k identickým výsledkom ako v predošlom prípade pri cykle FOR...NEXT. Pri tomto type príkladov používame častejšie cyklus FOR, nakoľko je kratší. Na vysvetlenie princípov cyklu WHILE...END však bol tento príklad postačujúci.

3.4 Predčasné ukončenie cyklu

V praxi sa stretneme určite so situáciou, že pri použití cyklov nebudeme chcieť čakať na skončenie celého cyklu, ale budeme potrebovať tento cyklus prerušiť ak nastane nami požadovaná situácia. V tomto prípade existuje k dispozícii príkaz EXITLOOP, ktorý je použiteľný pri cykloch FOR...NEXT a WHILE..WEND. V prípade aplikácie tohto príkazu vnútri cyklu, sa cyklus preruší pred skončením a program pokračuje ďalšími príkazmi. Všeobecný syntax príkazu EXITLOOP:

```
začiatok cyklu
exitloop
koniec cyklu
```

Uvedieme si špecifický príklad z oblasti praxe. Príklad sa bude týkať rozloženia časových radov na jeho zložky (trendová, cyklická, sezónna a iregulárna zložka). Máme k dispozícii veľké množstvo časových radov, ktoré máme rozložené do trendovej, iregulárnej a sezónnej zložky. Potrebujeme rozložiť ďalej trendovú zložku, ktorá môže obsahovať ešte cyklickú zložku časového radu. Každý časový rad chceme očistiť však takým spôsobom, aby sme neporušili cyklickú zložku. Použijeme metódu Months FOR Cyclical Dominance (MCD), ktorá nám určí minimálny počet mesiacov, pri ktorých priemerná zmena trendovo-cyklickej zložky dominuje priemernej zmene náhodnej zložky, t.j. pri ktorom je zmena náhodnej zložky menšia ako trendovej zložky[5]. Vzorec MCD (1):

$$\frac{\sum abs(I_t - I_{t-m}) / I_{t-m}}{\sum abs(TC_t - TC_{t-m}) / TC_{t-m}} < 1 \qquad (1)$$

I_t – *iregulárna zložka*
TC_t – *trendovo-cyklická zložka*
t, m – *čas*

[5] [2] *Kľúčik M., Haluška J.:* Construction of Composite Leading Indicator for the Slovak Economy. Scientific Annals of the "Alexandru Ioan Cuza", University of Iasi, Romania, 2008 (p. 363 – 370).

Pre každý časový rad teda potrebujeme zistiť jedno číslo, ktoré potom neskôr chceme využiť pri vyrovnaní (vyhladení) časových radov. Program bude vyzerať nasledovne:

```
for %1 %2 adIPI_IR adIPI_trd adIPI_MAN_IR adIPI_MAN_trd adIPI_MIN_IR
adIPI_MIN_trd adIPIMANTRA_IR adIPIMANTRA_trd adIRECZK_IR
adIRECZK_trd adIREEUR_IR adIREEUR_trd adIREUSD_IR adIREUSD_trd
adLBOET1_IR adLBOET1_trd adLBOET2_IR adLBOET2_trd adLDCONST_IR
adLDCONST_trd adLDIND_IR adLDIND_trd adLDINDELE_IR
adLDINDELE_trd adLDINDMAN_IR adLDINDMAN_trd adLDINDMIN_IR
adLDINDMIN_trd adLDRETAIL_IR adLDRETAIL_trd adLDTRANSP_IR
adLDTRANSP_trd adLDWHOLE_IR adLDWHOLE_trd adLOANHO1_IR
adLOANHO1_trd adLOANHO2_IR adLOANHO2_trd adLOANNON1_IR
adLOANNON1_trd adLOANNON2_IR adLOANNON2_trd adM0_00_IR
adM0_00_trd adM0SC_00_IR adM0SC_00_trd adM1P_00_IR adM1P_00_trd
adM1SC_00_IR adM1SC_00_trd adM2P_00_IR
        for !c = 1 to 24
        %3 = @str(!c)
                series {%1}_ic{%3} = @sum(@abs({%1}-{%1}(-{%3}))/{%1}(-
                        {%3}))/@sum(@abs({%2}-{%2}(-{%3}))/{%2}(-{%3}))
         if {%1}_ic{%3} > 1 then delete {%1}_ic{%3}
         else exitloop
         endif
         next
next
```

Pre každý časový rad vyskúša program 24 mesiacov, preto používame okrem cyklu FOR...NEXT na výmenu (opakovanie) všetkých časových radov aj cyklus na výmenu 24 mesiacov (od 1. mesiaca po 24. mesiac). Táto výmena je zabezpečená aj pomocou príkazu @str, ktorý premení číslo z cyklu FOR..NEXT na reťazec, ktorý sa využije o riadky nižšie pri výpočte vzorca pre MCD a po príkaze IF.

Príkaz @str je opačný k @val, ktorý sme si už v tejto príručke vysvetlili. Na rozdiel od @val, príkaz @str vracia reťazec, ktorý predstavuje uvedenú hodnotu, skalár v zátvorke. Všeobecný zápis príkazu je nasledovný:

```
@str(skalár)
```

Dôležitým je však samotný vzorec na výpočet MCD – v prípade ak daná hodnotu vypočítaná vzorcom prekročí jednotku (minimálny počet mesiacov, keď trendovo-cyklická zložka dominuje iregulárnej _TRD a _IR) – táto podmienka je uvedená v podmienke príkazu IF..ELSE, program ukončí cyklus pomocou príkazu EXITLOOP a pokračuje v ďalšom časovom rade, to znamená skončí sa vnorený cyklus (pre počet mesiacov), ale hlavný cyklus pokračuje ďalším časovým radom.

3.5 Prírastkové a úbytkové premenné

S využitím cyklov pri výpočtoch náročných na nadmerné množstvo dát súvisí aj používanie tzv. prírastkových (incrementing) a úbytkových (decrementing) premenných. Prírastková veličina prirátava k pôvodnej vlastnej hodnote určitú hodnotu:

```
i = i + 1
```

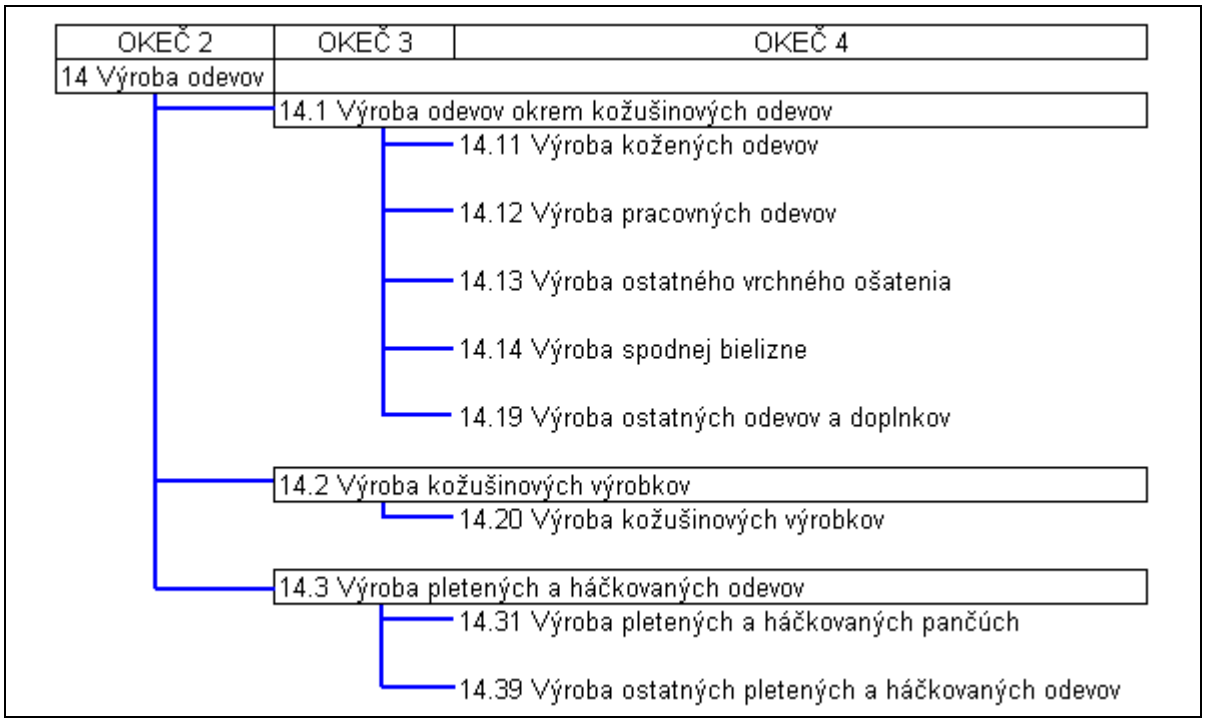
V tomto prípade sa premenná bude zvyšovať vždy o jednu jednotku pri každom použití, t.j. ak sa vyskytuje daný výraz v kóde viackrát. Často sa vyskytuje práve vnútri cyklov. Okrem prírastkových veličín existujú tzv. úbytkové veličiny, ktoré sa naopak po každom volaní znižujú o určitú hodnotu:

```
i = i – 1
```

Praktické využitie prírastkovej veličiny bolo uvedené v predošlej kapitole pri cykle WHILE, ale môžeme si uviesť pre zaujímavosť i nasledujúci príklad spojený s makroekonomickými časovými radmi:

Časové rady priemyselnej výroby na mesačnej báze sú k dispozícii na rôznych úrovniach agregácie v závislosti od odvetvovej klasifikácie ekonomických činností (OKEČ, angl. NACE). V príklade sa snažíme agregovať časové rady priemyslu z trojokečovej úrovne (nižšia úroveň) na dvojokečovú úroveň (vyššia úroveň). Máme k dispozícii časové rady OKEČov na trojčíselnej úrovni: 141 142 143 s názvami tr141, tr142, tr143. Teraz všetky tieto časové rady potrebujeme agregovať na úroveň dvojokečov t.j. 14. OKEČ klasifikácia funguje na princípe, že prvá číslica je označením najvyššej úrovne, druhá nižšej až po najnižšiu úroveň poslednej číslice. Pritom platí že 141 + 142 + 143 = 14. To znamená, že úroveň 14 je agregáciou podúrovní 141 až 149 (v našom zjednodušenom prípade 141 až 143). Prehľad názvov jednotlivých OKEČov je na nasledujúcom obrázku, kde je znázornená aj agregácia zo 4-OKEČového delenia na 3-OKEČové:

Obrázok 3.5 Klasifikácia ekonomických činností – príklad



Pri veľkom množstve časových radov je zadávanie vyššie uvedeného výrazu sčítania príliš zdĺhavé, preto použijeme program, ktorý nám výrazne skráti čas riešenia tohto problému. Program by vyzeral nasledovne:

```
series ntr14
group gr14 tr141 tr142 tr143
        for !a = 1 to gr14.@count
        %1 =gr14.@seriesname(!a)
        series ntr14 = ntr14 + {%1}
next
```

Skupina s názvom GR14 (OKEČ 14) obsahuje tri časové rady tržieb s názvami TR141 TR142 TR143. Cyklus vymení postupne všetkých členov GR14 vo výraze ntr14 = ntr14 + {%1}. Práve výraz ntr14 = ntr14 + {%1} je prírastkovou premennou. Aby nám program nevypísal chybové hlásenie, že neexistuje časový rad s názvom NTR14 v prvom cykle, musíme ho definovať hneď v prvom riadku programu. Prírastková veličina začína teda hodnotami NA. V prvom cykle budú teda hodnoty časového radu NTR14 vytvorené z prvého člena skupiny GR14 – NTR141. V druhom cykle sa priráta k tejto hodnote druhý časový rad NTR142 a v poslednom časový rad NTR143. Takto sme pomocou cyklu dokázali nahradiť jednoduchý výraz series NTR14 = NTR141 + NTR142 + NTR143. Tento program je preto viac-menej zbytočný, lebo ho môžeme napísať do jedného riadku:

```
series NTR14 = NTR141 + NTR142 + NTR143
```

Avšak využiť sa dá v prípade, ak zrátavame viacero OKEČov s väčším počtom časových radov. V nasledujúcom príklade to nebude len OKEČ č. 14 ale všetky OKEČe priemyslu. V tomto prípade predošlý cyklus FOR..NEXT bude vnorený do druhého cyklu FOR..NEXT, pomocou ktorého vymeníme všetky OKEČe na dvojčíselnej úrovni (OKEČ č. 34 nie je v SR zaradený):

```
for %1 05 06 07 08 09 10 11 12 13 14 15
16 17 18 19 20 21 22 23 24 25 26 27
28 29 30 31 32 33 35 36 37 38 39
    series ntr{%1} = 0
    group gr{%1} tr{%1}*
    for !a = 1 to gr{%1}.@count
        %2 =gr{%1}.@seriesname(!a)
        series ntr{%1} = ntr{%1} + {%2}
    next
next
```

V tomto prípade teda zrátame všetky trojOKEČe do príslušného dvojOKEČu. Predchádzajúci príklad znamenal približne to, že v prvom riadku by nebolo

```
for %1 05 06 07 08 09 10 11 12 13 14 15
atď.
```

ale iba jeden náš dvojOKEČ a to s číslom 14:

```
for %1 14
```

V tomto prípade agregujeme všetky trojOKEČe do dvojOKEČu, teda napríklad do OKEČu 07 zrátavame 071 a 072 atď. V tomto prípade si dokonca zľahčíme prácu a nebudeme musieť definovať členov každej skupiny, namiesto toho použijeme žolík *. Prvý cyklus, v ktorom sa vymenia všetky dvojOKEČe zabezpečí vytvorenie každej skupiny GR05, GR06 až GR39 a priradí ku každému názvu správne časové rady. Toto sa udeje v riadku

```
group gr{%1} tr{%1}*
```

V prípade prvého cyklu do skupiny GR05 zaradí časové rady s názvom TR05*, teda všetky časové rady, ktoré sa začínajú na písmená TR, čísla 05 a ďalšie hodnoty (TR051, TR052 atď.). Jediné čo budeme musieť zabezpečiť je, aby pracovný hárok neobsahoval objekty s rovnakými názvami, ktoré by sa nám mohli zapliesť neplánovane do výpočtu. Rovnako sme zmenili definovanie časového radu pred začiatkom cyklu, aby nám pri prvom prebiehajúcom cykle vnútorného FOR..NEXT nepísalo chybové hlásenie:

```
series ntr{%1} = 0
```

Daný časový rad sme definovali v prvom cykle (vnorenom) ako rovný nule a nie ako NA. Toto však nie je nutné. Táto definícia iba zabezpečí pri viacerých spusteniach programov (napr. po nájdenej chybe a opakovanom spustení v stále otvorenom súbore), aby príslušný časový rad na dvojOKEČovej úrovni napr. NTR14 bol nulový. Pri druhom spustení by totiž ešte obsahoval hodnoty z predošlého spustenia programu.

4. Regresia

4.1 Jednorovnicové odhady

V tejto kapitole sa pozrieme na základy tvorby jednorovnicových odhadov a ich testovacích štatistík.

Regresná analýza je štatistická technika na modelovanie a analýzu dát, ktorá sa snaží odhaliť vzájomnú závislosť dvoch alebo viacerých premenných. Závisle premenná je funkciou nezávisle premenných. Sila tejto závislosti je premietnutá v jednotlivých parametroch regresnej rovnice a v chybe odhadu. Parametre regresnej rovnice sú odhadované najčastejšie metódou najmenších štvorcov (najrozšírenejšia metóda). Okrem tejto metódy existujú aj ďalšie pokročilejšie a špecifické techniky, ako napr. vážená metóda najmenších štvorcov, dvojstupňová metóda najmenších štvorcov, nelineárna metóda najmenších štvorcov, generalizovaná metóda momentov, ARCH atď. Všetky tieto metódy sú k dispozícii v softvérovej palete EViews-u. My sa v tejto kapitole sústredíme len na základnú regresiu a hľadanie regresnej závislosti pomocou metódy najmenších štvorcov a jednoduchých rovníc a vysvetlíme si aj základné pojmy ako parametre, rezíduá a ďalšie charakteristiky regresnej analýzy.

Jednorovnicová regresia je najširšie používanou technikou v regresnej analýze. Obsahom tejto analýzy je špecifikácia regresného modelu, odhad regresného modelu a diagnostická analýza.

Špecifikácia regresného modelu si vyžaduje nadefinovať závisle, nezávisle premenné a neznáme parametre rovnice – tieto pojmy môžeme zhrnúť do pojmu reprezentácia rovnice. Následne je potrebné zvoliť metódu regresie a posledným krokom v našom prípade je zvoliť správny Sample súboru.

4.1.1 Reprezentácia rovnice

Rovnice sú v programovej aplikácii EViews reprezentované viacerými spôsobmi. Prvou reprezentáciou je vymenovanie závisle premennej, nezávisle premennej a koeficientov – „by list“ Ak chceme vysvetliť závislosť Hrubého domáceho produktu (HDP) od stavebnej produkcie (CONPROD), prvá reprezentácia bude vyzerať nasledovne:

```
LS HDP C CONPROD
```

LS znamená „LEAST SQUARES“ – teda metóda najmenších štvorcov a je to metóda, ktorou sa odhadujú parametre lineárnej a nelineárnej regresie. V tomto prípade, keď sa v reprezentácii príkazu rovnice nenachádzajú žiadne operátory (+,-...), je regresia automaticky chápaná ako lineárna. HDP a CONPROD sú názvy časových radov. HDP je závisle premenná a CONPROD nezávisle premenná – závisle premenná sa uvádza vždy na prvom mieste. Takto zisťujeme vlastne závislosť vývoja HDP od vývoja stavebnej produkcie (CONPROD). Posledný znak v prvej reprezentácii je C – je to názov koeficientu. Názov C je v tomto prípade povinný – vychádza z objektu  . Tento objekt je prednastaveným objektom – je vždy obsahom pracovného hárku a do tohto objektu sa zapisujú koeficienty z rovníc. Je to vektor koeficientov a dané koeficienty reprezentujú parametre regresných rovníc.

Druhý prípad reprezentácie rovnice je pomocou rovnice odhadu – „by Formula“:

```
HDP = C(1) + C(2) * CONPROD
```

V prípade, že používame logaritmickú transformáciu časových radov:

```
LOG(HDP) = C(1) + C(2) * LOG(CONPROD)
```

V tomto prípade neplatí pravidlo, že naľavo sa nachádza závisle premenná a na pravej strane nezávisle premenné. Znak rovná sa môže byť umiestnený kdekoľvek v rovnici, pričom program sám rozpozná závisle premennú (lebo je bez parametra): teda napr. rovnica vyššie sa dá napísať aj týmto spôsobom, pričom sú v skutočnosti i pre program identické:

```
LOG(HDP) – C(1) = C(2)* LOG(CONPROD)
```

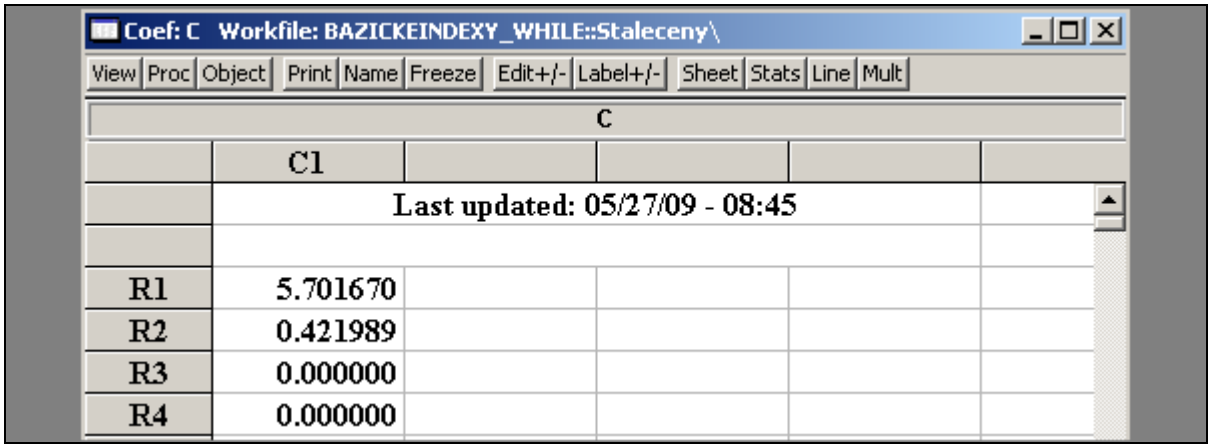
Výraz môže byť akokoľvek zložitý. Napríklad za koeficientom môže byť výraz zložený z viacerých časových radov a konštánt:

```
log(hdp) = c(1) + c(2) * (log(conprod/cpi2000*2)+2*log(ppi/ppiind*2))
```

atď.

Veľkosť písma sa pri písaní rovníc do príkazového riadku alebo programu nerozlišuje. Rovnako nezáleží na tom, či je medzi operátormi a jednotlivými názvami časových radov či parametrov (koeficientov) medzera alebo nie. Koeficienty môžeme označovať v zátvorke číslom. Podľa číselného označenia sa takto ukladajú v poradí do objektu C, t. j. do vektora. Jednotlivé parametre rovnice sú uložené nasledovne (obr. 4.1):

Obr. 4.1 Objekt koeficientov



Objekt vektor koeficientov obsahuje v stĺpci parametre odhadnutej funkcie. Riadok 1 (na obrázku 4.1) obsahuje koeficient s označením C(1), riadok 2 (R2) koeficient C(2) atď. Prednastavený objekt vektor koeficientov C obsahuje vždy koeficienty naposledy odhadovanej rovnice. V uvedenom príklade na obrázku je koeficient C(1) konštanta a koeficient C(2) je parameter premennej produkcie v stavebníctve.

Ak dosadíme do druhej reprezentácie rovnice odhadnuté parametre, výsledkom je rovnica:

```
LOG(HDP)= 5.701670252 + 0.4219890233 * LOG(CONPROD)
```

4.1.2 Odhad objektu rovnice

Po tom, čo sme si vysvetlili ako bývajú špecifikované vzťahy medzi závisle a nezávisle premennými, môžeme si predstaviť základné príkazy, ktoré nám tieto vzťahy dokážu prijať a následne vytvoriť output. Hlavným a základným príkazom je príkaz na vytvorenie jednoduchej regresnej rovnice – EQUATION, už spomínaný v predošlej kapitole.

Všeobecný syntax vytvorenia objektu rovnice – EQUATION:

```
equation názov_rovnice.metóda(možnosti) výraz
```

Príkaz na vytvorenie rovnice, s ktorou sme pracovali je nasledovný:

```
equation equ1.ls log(hdp) = c(1) + c(2) * log(conprod)
```

EQU1 je názov rovnice. Výraz je rovnica, ktorej reprezentácie sme si vysvetlili vyššie. Výraz tvorí v našom prípade rovnica log(hdp) = c(1) + c(2) * log(conprod). Metóda, ktorú budeme používať na odhad regresného vzťahu sa nazýva metóda najmenších štvorcov a má skratku *ls*. Medzi názvom rovnice a metódou je vždy bodka. Možnosti sme neuviedli žiadne, to znamená, že metóda najmenších štvorcov sa prevedie na automaticky nastavené parametre (default). Teraz máme splnené dve podmienky špecifikácie rovnice pre regresiu – reprezentáciu rovnice a stanovenú techniku výpočtu odhadu regresnej analýzy. Posledným krokom je stanovenie veľkosti časového rozpätia (Sample). Vzťah medzi závisle a nezávisle premennými bude vyrátaný v tom Sample, ktorý máme práve nastavený! Často dochádza k chybám pri odhade rovnice, lebo si nevšimneme, že sa nám medzičasom zmenil Sample pracovného hárku. Ak by sme mali napr. nastavený Sample na 1993q1 1993q1 v pracovnom hárku so štvrťročnou frekvenciou, a použili by sme príkaz equation vyššie, program vydá chybové hlásenie: „Insufficient number of observations“ – „Nedostatočný počet pozorovaní“. Pritom dobre vieme, že časové rady v rovnici majú dostatok pozorovaní na kvantifikáciu odhadu. Preto je výhodnejšie písať pred príkaz rovnice priamo príkaz na vytvorenie Sample:

```
smpl 1993q1 2008q4
equation equ1.ls log(hdp) = c(1) + c(2) * log(conprod)
```

V prípade ak máme vo výraze časovo posunuté premenné napr. log(conprod(-1)), berie program do výpočtu odhadu aj údaj conprod z 1992q4, teda údaj mimo stanoveného časového rozpätia. Toto je výhodné z praktického hľadiska, nakoľko nebudeme musieť počítať počty pozorovaní a meniť následne Sample pre rovnicu odhadu. Zadaný Sample pre rovnicu odhadu predpokladá teda nulové časové posunutie.

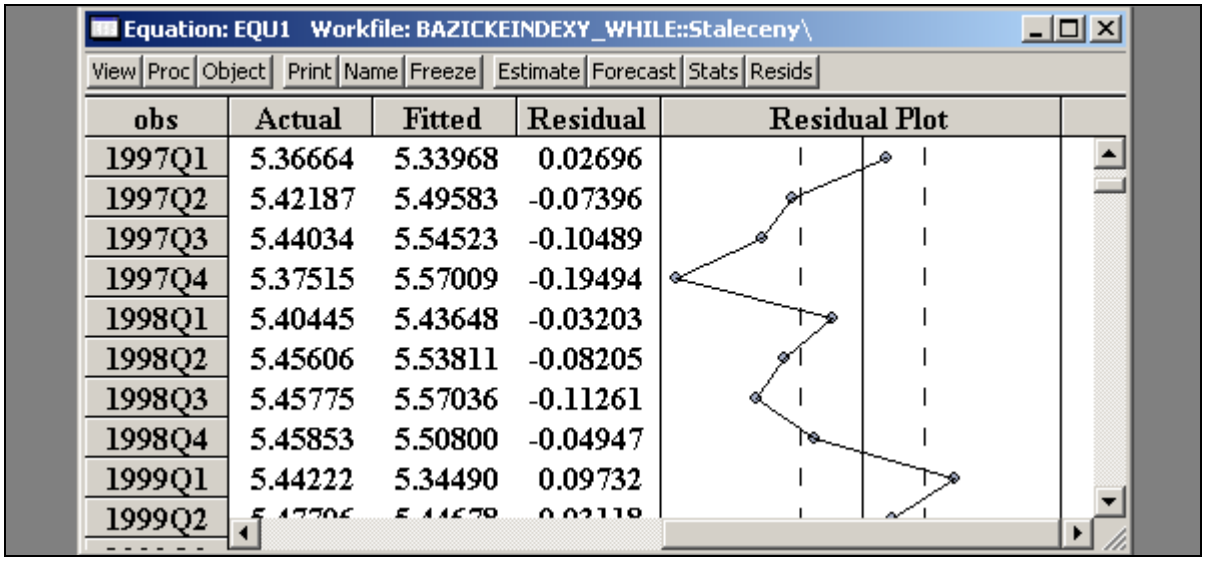
Vyššie uvedený príkaz v nám teda vytvorí nový objekt s názvom EQU1. Ide o objekt rovnice, ktorá má v pracovnom hárku označenie . Príkaz uložil prvé dva koeficienty do objektu vektorového koeficientu C, ako je vidieť z predošlého obr. 4.1. Posledným objektom, ktorý sa vzťahuje na výstup danej rovnice sú rezíduá. Rezíduá znamenajú pri regresii odchýlku hľadanej funkcie od tzv. fitted function – odhadnutej funkcie. Všetky tri zložky môžeme vidieť v tabuľke

výstupu rovnice – obr. 4.2 (prístup cez Menu objektu Equation\View\Actual, Fitted, Residual\Actual, Fitted, Residual Table).

Hodnoty Actual – skutočné sú v našom prípade hodnoty závisle premennej – log(HDP), hodnoty Fitted sú hodnoty pravej strany rovnice – teda 5.701670252 + 0.4219890233 * LOG(CONPROD). Rozdiel medzi oboma stranami rovnice predstavuje rezíduá.

Tak ako vkladá EViews po odhade rovnice koeficienty do vektoru koeficientov C, ktorý je automatickým objektom v každom pracovnom hárku, druhým takýmto prednastaveným objektom je RESID. RESID je objekt typu časového radu (SERIES), do ktorého sa ukladajú po každom odhade rovnice jej rezíduá.

Obr. 4.2 Zložky odhadnutej regresnej rovnice



Zatiaľ vieme pri regresii odhadnúť vzťah premenných pomocou vytvorenia objektu rovnice. Ak by sme si chceli uchovávať rezíduá z každej rovnice, ktoré sú k dispozícii v menu daného objektu, môžeme ich jednoducho skopírovať z tabuľky z obr. 4.2 do novovytvoreného časového radu, alebo môžeme použiť príkaz ihneď po odhade rovnice, pokým sú ešte uchované rezíduá v aktuálnom časovom rade RESID:

```
series reziduaequ1 = resid
```

Prípadne pomocou príkazu MAKERESID priamo po vytvorení odhadu rovnice:

```
equation equ1.ls log(hdp) = c(1) + c(2) * log(conprod)
equ1.makeresid reziduaequ1
```

Všeobecný syntax príkazu MAKERESID:

názov_rovnice.makeresid *názov_rezíduí(časového_radu)*

Medzi názvom rovnice a príkazom je opäť bodka.

Rovnako ako rezíduá si môžeme uložiť aj koeficienty (parametre) premenných regresie, aby neboli uchované len nakrátko v objekte C. To je možné pomocou príkazu COEF. Tento príkaz vytvorí objekt typu koeficientový vektor (stĺpcový). Všeobecný príkaz na vytvorenie tohto objektu:

```
coef(počet_koeficientov) názov_vektoru
```

Počet koeficientov sa rovná počtu riadkov v stĺpci. Závisí to od toho, koľko parametrov chceme mať v regresnej funkcii. My máme dva parametre – konštantu a jeden parameter premennej stavebná produkcia. Preto nám stačí uviesť v zátvorke číslo 2. Názov vektoru je názov jednotlivých parametrov i názov samotného objektu v pracovnom hárku, ktorý má označenie rovnaké ako automatický objekt C – teda .

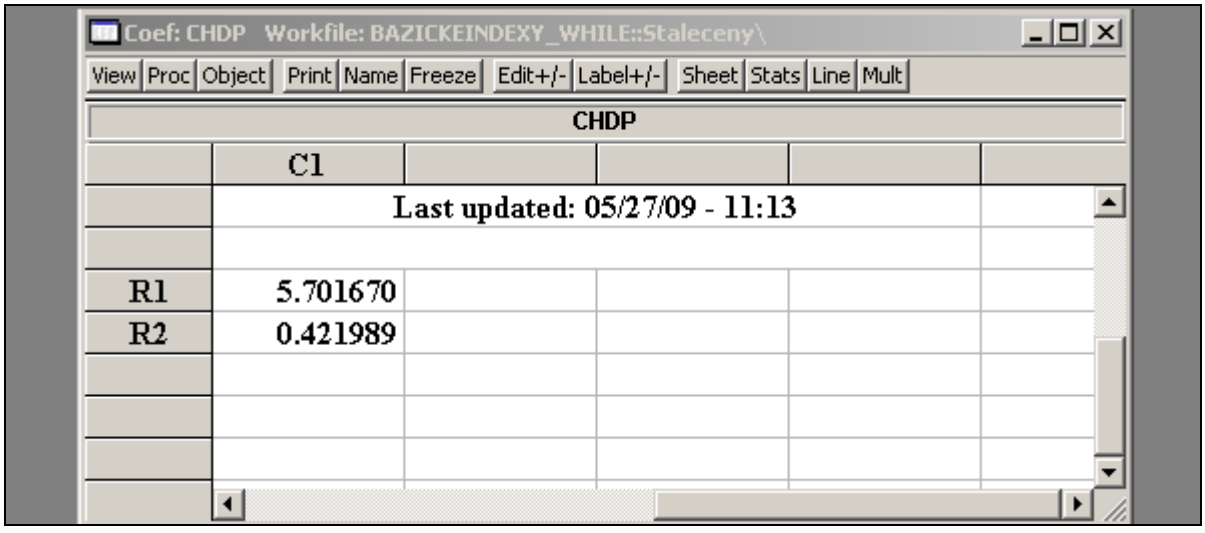
Postup uloženia koeficientov do objektu je mierne zložitejší, nakoľko sa musia zhodovať aj názvy koeficientov s vektorom, ktorý vytvoríme pomocou príkazu COEF. Najprv musíme definovať vektor a následne doňho môžeme uložiť koeficienty z rovnice. V tomto prípade sa koeficienty rovnice nemusia nazývať C ale ľubovoľne, nakoľko už musia byť predtým definované pomocou COEF príkazu ako koeficienty. Koeficienty našej regresie nazveme chdp(1) a chdp(2):

```
coef (2) chdp
equation equ1.ls log(hdp) = chdp(1) + chdp(2) * log(conprod)
equ1.makeresid reziduaequ1
```

Najprv sme definovali objekt vektoru s názvom CHDP a s názvami vektorov CHDP(1) a CHDP(2) – teda iba dvojriadkový vektor (stĺpcový). Následne sme vytvorili rovnicu pomocou príkazu EQUATION a v treťom kroku uchovali rezíduá.

Objekt koeficientu aj samotné koeficienty sa zhodujú s výsledkami, ktoré sme mali v predošlých príkladoch (obr. 4.3):

Obr. 4.3 Výsledky regresných koeficientov



Koeficienty môžu byť extrahované z odhadnutej rovnice i druhým spôsobom a to priamo pomocou príkazu. Jeho forma je nasledovná:

názov_rovnice.názov_koeficientu

Tento výraz môžeme používať kdekoľvek, i pri definovaní nového časového radu, napr.:

```
series koefrovn = equ1.chdp(2) + 2
```

Tento príkaz nám vytvorí nový časový rad s názvom KOEFROVN, ktorého hodnoty budú vyrátané ako súčet druhého koeficientu odhadnutého v rovnici s názvom EQU1. Túto rovnicu sme odhadli vyššie v texte. Časový radu bude mať v aktuálnom Sample pracovného hárku hodnotu 2.421989 (z obrázku 4.3).

Vysvetlíme si teraz všetky štatistické výstupy odhadu regresnej rovnice. Doteraz sme poznali iba Actual, Fitted, Residual Table prístupom cez Menu daného objektu rovnice. Výstup sa nachádza v menu objektu rovnice na adrese Equation\View\Estimation Output. Prostredníctvom programovacieho jazyka ho generujeme nasledovne:

názov_rovnice.output

teda

```
equ1.output
```

Tabuľka výstupu odhadu regresnej rovnice pomocou metódy najmenších štvorcov vyzerá nasledovne (obr. 4.4):

Obr. 4.4 Výstup odhadu regresnej rovnice

Equation: EQU1 Workfile: BAZICKEINDEXY_WHILE::Staleceny\

View | Proc | Object | Print | Name | Freeze | Estimate | Forecast | Stats | Resids

Dependent Variable: LOG(HDP)
Method: Least Squares
Date: 05/27/09 Time: 11:13
Sample (adjusted): 1997Q1 2008Q4
Included observations: 48 after adjustments
LOG(HDP) = CHDP(1) + CHDP(2) * LOG(CONPROD)

	Coefficient	Std. Error	t-Statistic	Prob.
CHDP(1)	5.701670	0.010841	525.9537	0.0000
CHDP(2)	0.421989	0.023723	17.78784	0.0000

R-squared	0.873071	Mean dependent var	5.606677
Adjusted R-squared	0.870312	S.D. dependent var	0.181495
S.E. of regression	0.065361	Akaike info criterion	-2.577023
Sum squared resid	0.196512	Schwarz criterion	-2.499056
Log likelihood	63.84855	Durbin-Watson stat	1.453975

Údaje zobrazené v tabuľke na obr. 4.4 môžeme rozdeliť do štyroch skupín. Prvou sú popisné údaje rovnice, teda použitá metóda na odhad regresie, čas a dátum odhadu, informácia o použitom časovom rozpätí a počte použitých pozorovaní pri odhade. Druhou informáciou v tabuľke outputu je výraz – teda samotná rovnica. Tretím blokom sú takzvané Coefficient results, teda výsledky koeficientov – teda výška jednotlivých koeficientov, ich štandardná chyba, údaje t-štatistiky a na základe nej vyrátaná hladina významnosti. V poslednom štvrtom bloku sú takzvané sumárne štatistiky (Summary statistics) odhadu, tých je značne viac – dva stĺpce po 5 štatistík. V tejto knižke sa nebudeme zaoberať teóriou štatistiky. Vzťahy, na základe ktorých boli vypočítané štatistiky vo výstupoch sú uvedené v pomoci (Help) programu. Pre nás je dôležitá otázka, či môžeme údaje z tejto tabuľky extrahovať a použiť pri analýze ďalej. Z menovaných štyroch skupín máme doteraz programovo (pomocou príkazov) pod kontrolou z prvej skupiny stanovenie závisle premennej (log(HDP)), metódy odhadu (ls), Sample (1997q1 2008q4), a počet pozorovaní (vyplýva zo Sample a z údajov v časových radoch). Informácia o reálnom čase a dátume odhadu rovnice pre nás nie je zaujímavá. Z druhej skupiny vieme napísať dvoma spôsobmi reprezentáciu rovnice pre odhad. Ak dlhšie pracujeme s odhadmi, zistíme, že údaje, ktoré sú vlastne záverečnými výstupmi z odhadu môžeme využiť i ďalej v analýzach. Preto nás zaujíma, ktoré hodnoty a akým spôsobom sa dajú extrahovať z tretej a štvrtej skupiny z tabuľky výstupu odhadu rovnice.

Tieto informácie sú dostupné pomocou tzv. @-funkcií, s ktorými sme sa v texte príručky stretli už viackrát. Tieto funkcie dokážu vrátiť len skalárnu veličinu alebo maticu (či vektor).

Z tretej skupiny – zo skupiny koeficientových výsledkov, dokážu extrahovať skalárnu veličinu nasledujúce funkcie:

@coefs(i) – vráti hodnotu i-teho koeficientu. Okrem extrakcie tejto hodnoty z výstupu rovnice týmto spôsobom, môžeme koeficienty vyňať i z vektoru koeficientov pomocou príkazu COEF, ktorý sme rozoberali vyššie

@stderrs(i) – vráti štandardnú chybu i-teho koeficientu
@tstats(i) – vráti hodnotu t-štatistiky pre i-ty koeficient
@coefs – vráti vektor hodnôt koeficientov
@stderrs – vráti vektor hodnôt štandardných chýb koeficientov
@tstats – vráti vektor hodnôt t-štatistík koeficientov

Poslednú štatistiku, t.j. probability nedokážeme pomocou príkazov EViews-u extrahovať. Môžeme si ju však vypočítať samostatne a to pomocou t-štatistiky, ktorú si dokážeme vyňať z tabuľky. Niektoré pojmy z výstupu rovnice si môžeme krátko vysvetliť:

Coefficients – odhadnuté koeficienty – sú vyrátané podľa metódy regresie, v našom prípade pomocou štandardnej rovnice metódy najmenších štvorcov. Jednotlivé koeficienty znamenajú príspevok nezávisle premenných k závisle premennej (ak sú ostatné konštanty), teda mieru závislosti nezávisle premennej a závisle premennej pri nemennosti ostatných premenných. Ak je prítomný samotný koeficient – konštanta (ako v našom príklade vyššie), ten sa rovná hodnote, pri ktorej sú ostatné premenné nulové (pretína y-novú os). Použitie bázických indexov súvisí s ľahšou interpretáciou a s normalizáciou časových radov s rôznymi jednotkami.
Standard Error – štandardná chyba – štandardná chyba vypočítaných koeficientov, vyrátajú sa ako druhá mocnina diagonál kovariačnej matice koeficientov.
t-statistics – t-štatistika – podiel odhadnutých koeficientov k ich štandardnej chybe, interpretácia t-štatistiky je zobrazená v ďalšom stĺpci výstupu rovnice – probability – pomocou pravdepodobnosti.

Probability – pravdepodobnosť – je hraničná hladina signifikancie (významnosti). Ak si stanovíme 5% hladinu významnosti, všetky koeficienty s pravdepodobnosťou (aj p-value) nad 95% sú v našej regresnej rovnici štatisticky významné (*p* menšie než 0.05).

Z tretej skupiny tabuľky výstupov rovnice sme mohli teda extrahovať okrem pravdepodobnosti všetky skalárne veličiny, resp. vektory. Zo štvrtej skupiny – tzv. sumárnej štatistiky môžeme vrátiť vo forme skalárnej veličiny alebo matice resp. vektoru nasledujúce štatistiky:

@aic – vráti hodnotu Akaikeho informačného kritéria
@dw – vráti hodnotu Durbin-Watsonovej štatistiky
@logl – hodnota log-likelihoodovej funkcie
@meandep – priemer závisle premennej
@r2 - R^2
@rbar2 – adjustované R^2
@schwarz – Schwarzovo informačné kritérium
@sddep – štandardná odchýlka závisle premennej
@ssr – suma štvorcov rezíduí
@coefcov – vráti maticu obsahujúcu kovariačnú maticu koeficientov
Opäť si krátko vysvetlíme niektoré z týchto pojmov:

Akaike Information Criterion, Schwarz Criterion – informačné kritéria slúžiace ako kritérium pre výber správneho modelu
R^2 – meria úspešnosť regresie pri prognózovaní hodnôt závisle premennej v zvolenom Sample. Je to miera rozptylu vysvetlená nezávisle premennými. Ak by bola regresia dokonalá, rovnala by sa táto štatistika jednej.
Adjusted R^2 (regulované R^2) – jednou z negatívnych vlastností R^2 je, že nikdy neklesne po pridaní ďalších premenných do vzorca (regresorov). Upravované R^2 penalizuje pridávanie ďalších regresorov.
Štandardná chyba regresie (S. E. of Regression) – suma odhadnutého rozptylu rezíduí.
Suma druhej mocniny rezíduí – Sum-of-Squared Residuals – suma štvorcov rezíduí
Log likelihood – štatistický test spojený s distribúciou hodnôt
Durbin-Watson Statistics – meria autokoreláciu rezíduí
Mean and Standard Deviation (S.D.) of the Dependent Variable – Priemer a štandardná odchýlka závisle premennej

Ďalšie informácie z tabuľky, ktoré majú nižšiu vypovedaciu schopnosť, ale ktoré môžeme v budúcnosti využiť:
@regobs – počet pozorovaní v regresii
@ncoef – počet odhadnutých koeficientov

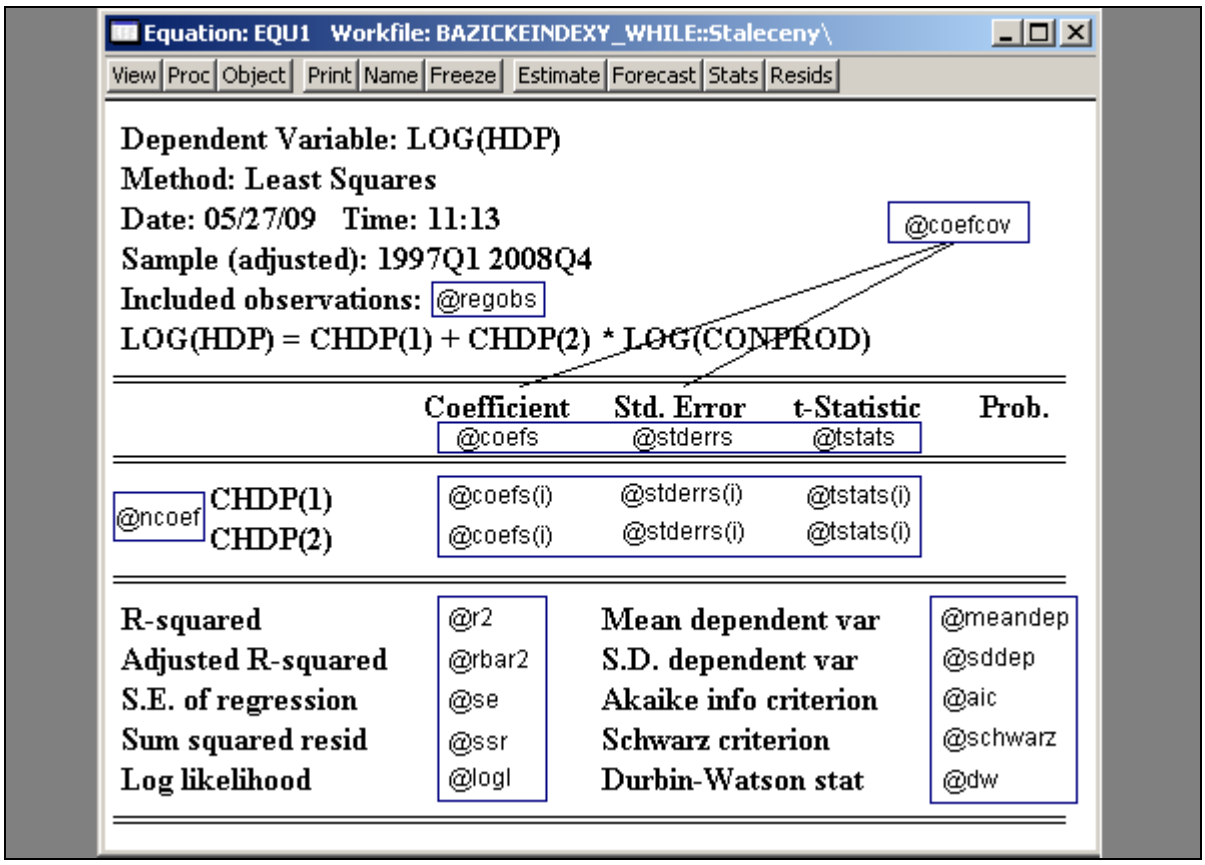
Prehľad extrahovaných hodnôt z výstupu rovnice je znázornený na obr. 4.5 na nasledujúcej strane.

4.1.3 Práca s objektom rovnice

Môžeme si zhrnúť základné charakteristiky a príkazy spojené s objektom rovnice:

a) metóda – tú sme už spomínali, v našom prípade budeme využívať iba LS – teda obyčajnú metódu najmenších štvorcov. V EViews-e existujú ďalšie metódy (v zátvorke so skratkami):
   - autoregresívne podmienená heteroskedasticita – ARCH a GARCH modely (arch), binárne závislé modely premenných – probit, logit, gompit modely (binary), cenzorované a skrátené regresné modely – tobit (censored), modelovanie počítacích dát – negatívne bionomiálne a kvázi-maximálne pravdepodobnostné počítacie modely (count), generalizovaná metóda momentov (gmm), logit – binárne odhady (logit), lineárna a nelineárna regresia najmenších štvorcov (ls), modely radových závisle premenných (ordered), probit – binárne odhady (probit) a lineárne a nelineárne modely dvojstupňových najmenších štvorcov – TSLS, ARMA (tsls).

Obr. 4.5 Popis výstupu odhadu regresnej rovnice



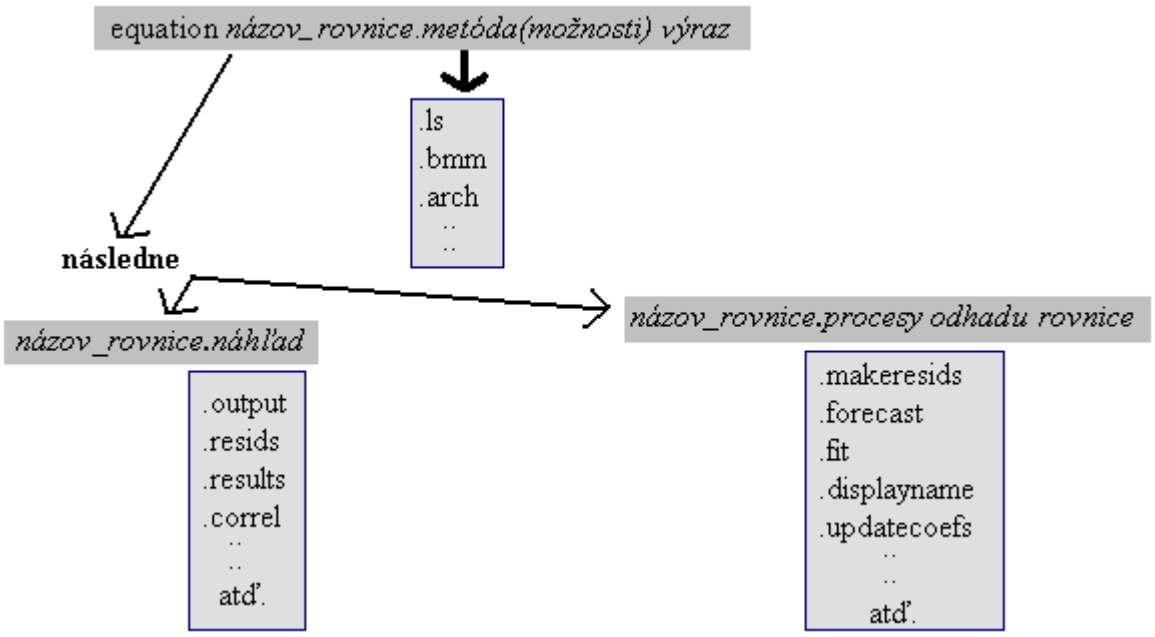
b) druh náhľadu na rovnice (Equation views):
   - zahŕňa vyššie spomenutý náhľad vo forme tabuľky – output. Okrem tohto pohľadu je známych viacero príkazov na rôzne druhy náhľadu. Z tých, čo sa vzťahujú na metódu najmenších štvorcov môžeme uviesť nasledujúce:
   - kovariačná matica koeficientov (coefcov), korelogram rezíduí (correl), korelogram rezíduí umocnených na druhú (correlsq), histogram a deskriptívna štatistika rezíduí (hist), návestie, označenie rovnice (label), tabuľka odhadov rovnice (output), tabuľkové zobrazenie skutočných a odhadnutých (actual and fitted) hodnôt závisle premennej spolu s rezíduami (resids), tabuľka výsledkov odhadu (results) a Whiteov test heteroskedasticity (white).

c) procesy odhadu rovnice (vrátane prognózovania) – sú ďalšie príkazy spojené s uchovaním dát z odhadu rovnice a zároveň príkazy na prognózovanie závisle premenných na základe regresného odhadu. Z tých čo budeme používať môžeme vymenovať nasledovné:
   - DISPLAYNAME – stanoví meno, ktoré bude zobrazované pri danej rovnici
   - FIT – statická prognóza
   - FORECAST – dynamická prognóza
   - MAKERESIDS – vytvorenie časového radu obsahujúceho rezíduá z rovnice
   - UPDATECOEFS – aktualizácia koeficientového vektoru z rovnice

Prehľad všetkých možností príkazov spojených s objektom rovnice je zobrazený na obr. 4.6 na nasledujúcej strane.

Objekt rovnice, rovnako ako aj ostatné objekty pracovného hárku EViews-u, môžu byť premiestnené do databázy, rovnako ako časové rady. Následne môžu byť spätne skopírované z databázy do pracovného hárku, ak budeme opäť chcieť s nimi pracovať. Rovnicové objekty sa premiestňujú vo forme posledne odhadnutých charakteristík. To znamená, že môžu byť skopírované i do pracovného hárku, ktorý neobsahuje časové rady obsiahnuté v regresnej rovnici objektu. Rovnicový objekt tohto typu však nemôžeme potom opäť odhadnúť ani meniť jednotlivé charakteristiky, iba pozorovať. Ak chceme odhadnúť daný vzťah nanovo, musia byť k dispozícii v danom pracovnom hárku i časové rady, prípadne koeficienty obsiahnuté vnútri tohto objektu.

Obr. 4.6 Objekt rovnice a jeho syntax



4.2 Používanie špeciálnych funkcií v rovniciach

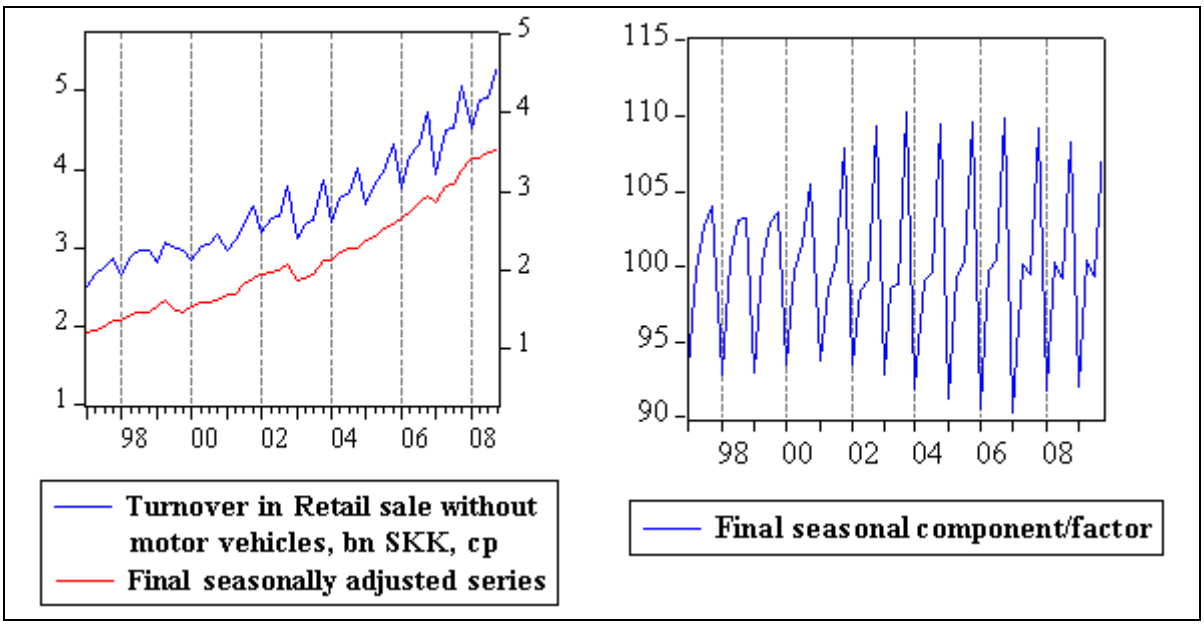
Závisle premenná, ktorú sa snažíme v regresných rovniciach vysvetliť pomocou ostatných premenných (teda nezávisle premenných), môže obsahovať sezónnu a trendovú zložku. Veľkú časť rozptylu závisle premennej môžeme preto často vysvetliť len pomocou sezónnej a trendovej zložky. Sezónne a trendové premenné patria medzi takzvané umelé premenné (dummy variables), lebo sa pomocou nich snažíme umelo napodobniť sezónne či trendové priebehy časových radov.

4.2.1 Sezónne umelé premenné

Sezónne umelé premenné reprezentujú sezónnosť časových radov. V prípade štvrťročných časových radov existujú štyri sezónne premenné, v prípade mesačných časových radov 12. Sezónnosť vychádza z reálnych dôkazov o priebehu niektorých časových radov. Väčšina ekonomických premenných vykazuje sezónnosť. Napríklad v čase Vianoc – v decembri výrazne narastá spotreba, naopak cez leto výrazne ubúda priemyselná výroba (závodné prázdniny), v letných mesiacov stúpa i tzv. sezónna zamestnanosť. Aby sme tieto výkyvy mohli napodobiť, musíme priradiť tomu-ktorému pozorovaniu pridanú hodnotu, akú nevykazujú ostatné sezóny. Na nasledujúcom obrázku (obr. 4.7) vidíme sezónne očistený časový rad maloobchodných tržieb (štvrťročné pozorovania), t.j. taký časový rad, ktorý neobsahuje sezónnosť – obr. 4.7 vľavo. Červená línia je teda časový rad v prípade, keby nevykazoval žiadne sezónne výkyvy. Na pravej strane môžeme vidieť sezónnu zložku časového radu. Práve tento časový rad môžeme napodobniť pomocou sezónnych umelých premenných. Ako vidíme zo sezónnej zložky na obrázku, každý kvartál sa chová v rámci jedného roka inak. Najvyššie tržby bývajú v štvrtom kvartáli a najmenšie v prvom kvartáli.

V EViews-e existujú dva spôsoby, ako môžeme používať sezónne umelé premenné. Prvým je automatický spôsob predstavený pomocou funkcie @seas a druhý spôsob je vytvorenie vlastných premenných.

Obr. 4.7 Sezónna zložka časových radov



Funkcia @seas patrí v EViews-e medzi základné časové premenné (Basic Date Functions), všeobecný syntax funkcie je nasledovný:

@SEAS(*číslo_sezónny*)

V prípade štvrťročných časových radov je číslo sezóny 1 až 4, v prípade mesačných 1 až 12. Funkcia vráti umelú premennú podľa periódy pozorovania. Pre lepšie pochopenie vytvoríme regresnú rovnicu, kde závisle premennou budú tržby v maloobchode a vysvetľujúcimi premennými sezónne umelé premenné.

```
equation retail1.ls turn_retail = c(1) * @seas(1) + c(2) * @seas(2) + c(3) * @seas(3)+ c(4) * @seas(4)
```

Všetky štyri sezónne premenné boli významné (obr. 4.8 na nasledujúcej strane), pričom najväčšiu váhu mala štvrtá sezónna premenná a najmenšiu prvá (podľa výšky koeficientov). Ako vidíme z grafu rezíduí, skutočných a fitted hodnôt, regresná rovnica dokázala vysvetliť iba sezónnu variáciu časového radu, pričom je jasne viditeľný rastúci trend časového radu rezíduí. Všimnite si, že reziduálna línia sa podobá na sezónne očistený rad z obr. 4.7 vyššie. Zelená línia predstavuje fitted hodnoty, teda vlastne jednoduchú sezónnu zložku – každá sezóna má pevne stanovený koeficient.

Obr. 4.8 Sezónne premenné ako vysvetľujúce premenné

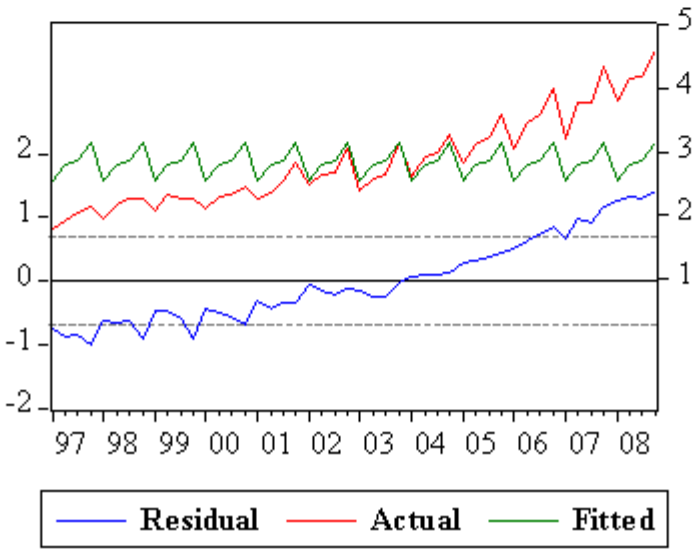
Equation: RETAIL1 Workfile: SEASDUMM::Untitled\

View | Proc | Object | Print | Name | Freeze | Estimate | Forecast | Stats | Resids

Dependent Variable: TURN_RETAIL
Method: Least Squares
Date: 06/10/09 Time: 13:36
Sample: 1997Q1 2008Q4
Included observations: 48
TURN_RETAIL = C(1) * @SEAS(1) + C(2) * @SEAS(2) + C(3) * @SEAS(3) + C(4) * @SEAS(4)

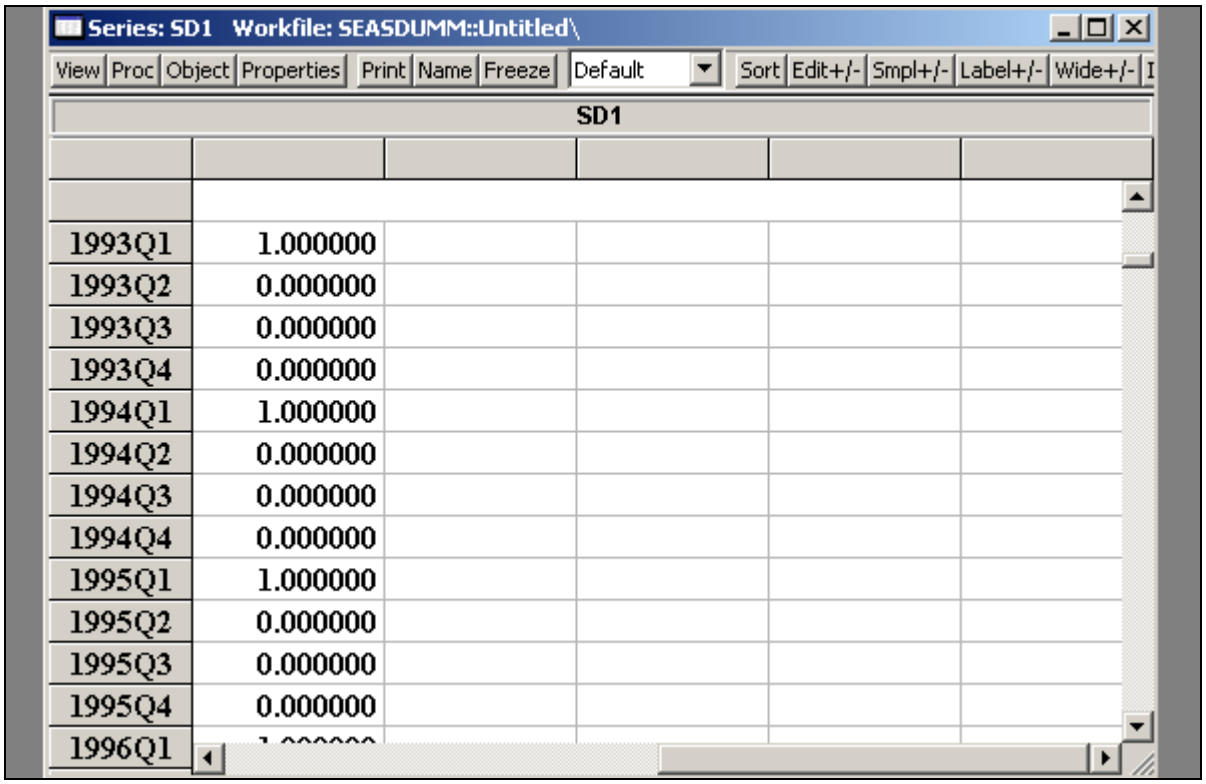
	Coefficient	Std. Error	t-Statistic	Prob.
C(1)	2.552166	0.203450	12.54443	0.0000
C(2)	2.816545	0.203450	13.84391	0.0000
C(3)	2.882017	0.203450	14.16571	0.0000
C(4)	3.165770	0.203450	15.56042	0.0000

R-squared	0.094690	Mean dependent var	2.854125
Adjusted R-squared	0.032964	S.D. dependent var	0.716683
S.E. of regression	0.704772	Akaike info criterion	2.217770
Sum squared resid	21.85496	Schwarz criterion	2.373704
Log likelihood	-49.22649	Durbin-Watson stat	0.069504



Druhou možnosťou vytvorenia sezónnych umelých premenných je vytvorenie časových radov pre jednotlivé sezóny. Ak chceme zvýrazniť dôležitosť určitej sezónny, tak jej oproti ostatným sezónam priradíme pridanú hodnotu (obr. 4.9):

Obr. 4.9 Časový rad umelej sezónnej premennej



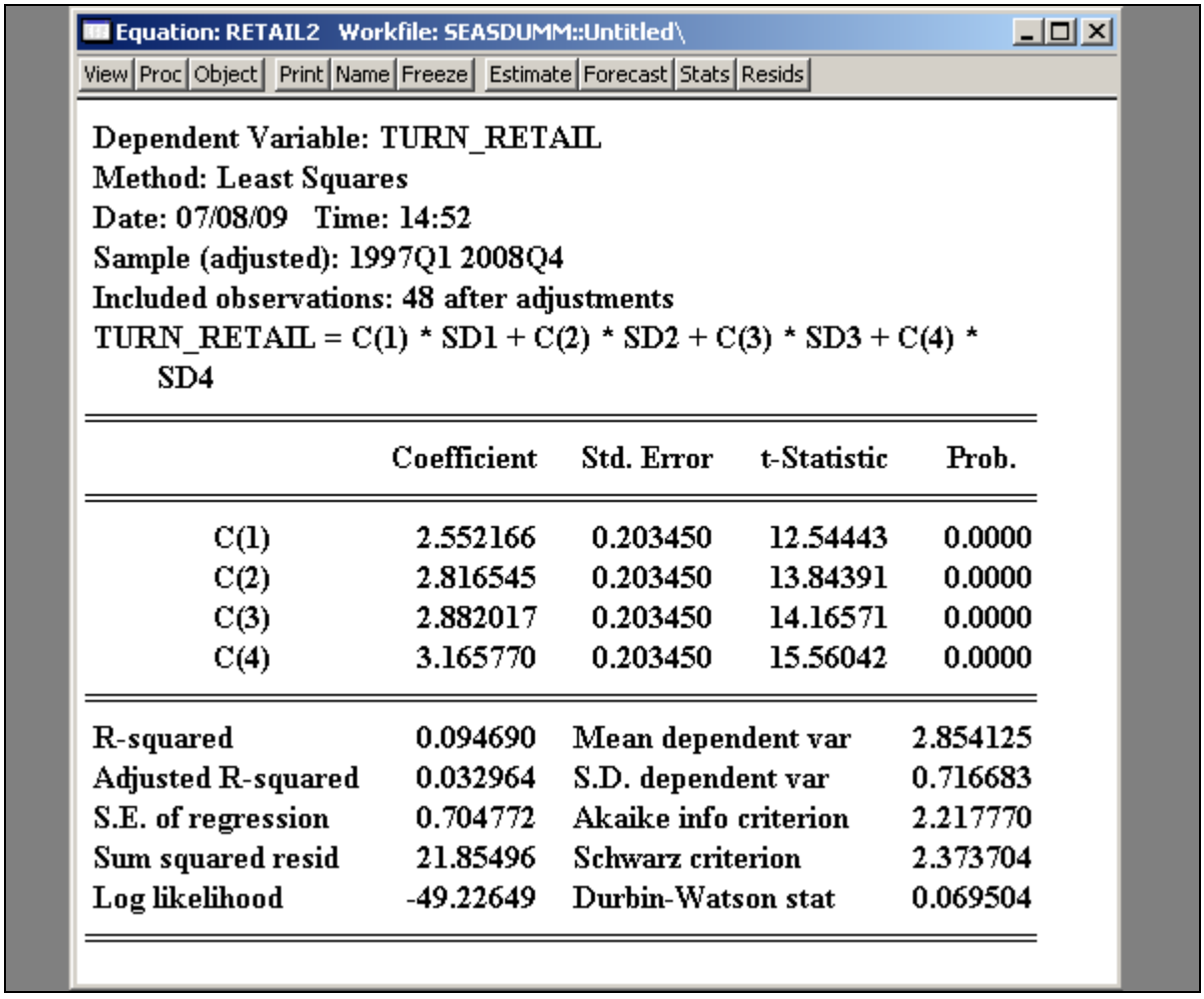
Časový rad pre sezónnu zložku sme nazvali SD1, ktorá je umelou premennou pre prvý kvartál. V prvom kvartáli je jednotka, v ďalších ostala nula. Rovnako vytvoríme aj ďalšie tri časové rady pre umelé premenné ostatných sezón. Teraz skúsime porovnať výsledky automatických sezónnych premenných EViews-u a našich premenných v regresii (obr. 4.10 nasledujúca strana).

Štatistiky výsledkov oboch regresných rovníc sú identické. Výhodou našej formy sezónnych umelých premenných je ľahšia matematická operácia. Napríklad môžeme používať diferencie, ktoré nemôžeme používať pri funkcii @SEAS(1), lebo funkcia nemá vlastnosti časového radu.

Pri tvorbe sezónnych umelých premenných môžeme využiť trocha praktických skúseností získaných v predošlých kapitolách. Potrebujeme vytvoriť štyri časové rady, pri ktorých sa vždy každé štvrté pozorovanie rovná jednej a ostatné tri sa rovnajú nule. Postup je jednoduchý. V prvom kroku vytvoríme všetky štyri časové rady a priradíme im hodnotu 0.

```
smpl @all
genr sd1=0
genr sd2=0
genr sd3=0
genr sd4=0
```

Obr. 4.10 Výstup odhadu regresnej rovnice pri umelých sezónnych premenných 2



Equation: RETAIL2 Workfile: SEASDUMM::Untitled\

View | Proc | Object | Print | Name | Freeze | Estimate | Forecast | Stats | Resids

Dependent Variable: TURN_RETAIL
Method: Least Squares
Date: 07/08/09 Time: 14:52
Sample (adjusted): 1997Q1 2008Q4
Included observations: 48 after adjustments
TURN_RETAIL = C(1) * SD1 + C(2) * SD2 + C(3) * SD3 + C(4) * SD4

	Coefficient	Std. Error	t-Statistic	Prob.
C(1)	2.552166	0.203450	12.54443	0.0000
C(2)	2.816545	0.203450	13.84391	0.0000
C(3)	2.882017	0.203450	14.16571	0.0000
C(4)	3.165770	0.203450	15.56042	0.0000

R-squared	0.094690	Mean dependent var	2.854125
Adjusted R-squared	0.032964	S.D. dependent var	0.716683
S.E. of regression	0.704772	Akaike info criterion	2.217770
Sum squared resid	21.85496	Schwarz criterion	2.373704
Log likelihood	-49.22649	Durbin-Watson stat	0.069504

Funkcia príkazu GENR je v tomto prípade identická s príkazom SERIES. Na vytvorenie časových radov teda použijeme pre zmenu príkaz GENR. Teraz musíme priradiť do správneho pozorovania jednotku, najprv iba do prvého roku:

```
smpl 1993q1 1993q1
genr sd1=1
smpl 1993q2 1993q2
genr sd2=1
smpl 1993q3 1993q3
genr sd3=1
smpl 1993q4 1993q4
genr sd4=1
```

Do ďalších rokov stačí, ak skopírujeme štruktúru jednotiek a núl konkrétneho časového radu. Napríklad pre sezónnu premennú SD1 : 1 0 0 0 stačí prekopírovať do ďalších rokov tiež 1 0 0 0. Postupne na každý rok by sme museli použiť príkaz:

```
smpl 1994q1 1994q4
genr sd1=sd1(-4)
```

Avšak stačí rovno napísať na celý Sample:

```
smpl 1994q1 2009q4
genr sd1=sd1(-4)
genr sd2=sd2(-4)
genr sd3=sd3(-4)
genr sd4=sd4(-4)
```

Tým pádom sú časové rady umelých sezónnych premenných v konečnej forme.

Ak sa nám zdá, že je to príliš veľa vypisovania pomocou klávesnice, program sa dá skrátiť. Využijeme to pri vytvorení umelých premenných pre mesačné časové rady. V tomto prípade rovnako potrebujeme pre každý mesiac priradiť umelú premennú SD1 až SD12, pričom v každom roku bude iba jedna jednotka a jedenásť núl – teda štruktúra premennej pre január bude 1-0-0-0-0-0-0-0-0-0-0-0 pre každý rok. Každý časový rad má názov SD za ktorým nasleduje číslo 1 až 12. Využijeme cyklus FOR..NEXT a skalárnu premennú:

```
smpl @all
  for !a = 1 to 12
      genr sd{!a}=0
  next
```

Spustenie programu vyzerá nasledovne:

```
for ...
!a = 1 ..... genr sd1 = 0   .....next
!a = 2 ..... genr sd2 = 0   .....next
atď.
!a = 12 ..... genr sd12 = 0 .....next
koniec
```

Následne priradíme jednotku konkrétnemu mesiacu všetkým premenným naraz. Opäť použijeme rovnaký cyklus:

```
for !b = 1 to 12
      smpl 1991m{!b} 1991m{!b}
      genr sd{!b}=1
next
```

V každom cykle priradí program namiesto skalárnej veličiny {!b} príslušné číslo od 1 do 12. V prvom cykle priradí namiesto každej zátvorky jednotku, v druhom dvojku atď.

```
Prvý cyklus:
    smpl 1991m1 1991m1
    genr sd1 = 1
Druhý cyklus:
    smpl 1991m2 1991m2
    genr sd2 = 2
atď.
```

V každom cykle program vráti iba jednu hodnotu. Často NESPRÁVNE očakávame nasledovný postup programu:

```
Prvý cyklus:
    smpl 1991m1 1991m2
    genr sd1 = 3
Druhý cyklus:
    smpl 1991m4 1991m5
    genr sd2 = 6
atď.
```

Posledným krokom je okopírovanie štruktúry z prvého roka do ďalších rokov, rovnako ako v prípade štvrťročných údajov:

```
smpl 1992m1 2010m12
  for !c = 1 to 12
      genr sd{!c}=sd{!c}(-12)
  next
```

Spolu sú teda tri kroky nasledovné:

```
smpl @all
  for !a = 1 to 12
      genr sd{!a}=0
  next
for !b = 1 to 12
      smpl 1991m{!b} 1991m{!b}
      genr sd{!b}=1
next
smpl 1992m1 2010m12
  for !c = 1 to 12
      genr sd{!c}=sd{!c}(-12)
  next
```

4.2.2 Trendové umelé premenné

Rovnako ako pri sezónnych „dummies“, aj v tomto prípade môžeme použiť dva postupy – vlastný a automatický pomocou funkcií EViews-u. Funkcia @trend je zaradená v EViews-e medzi osobitnými trendovými funkciami. Funkcia vytvorí premennú, ktorá má predstavovať trend. Sezónna funkcia @seas(x) vytvárala sezónnosť. Trendovú zložku sme videli pri obrázkoch tržieb z maloobchodu pri vysvetľovaní sezónnych premenných. Trend časového radu tržieb v maloobchode bol rastúci (bez sezónnych zložiek). Okrem sezónnych zložiek a trendovej zložky má každý časový rad variabilnú (iregulárnu) zložku a prípadne i cyklickú zložku. Niekedy sa trendová a cyklická uvádzajú spoločne ako trendová zložka. Pre nás má na začiatku význam stanoviť jednoduchý lineárny trend.

Funkcia @trend má dve verzie.

```
@trend("bázický _dátum")
```

V prvej verzii s rovnakým názvom vráti program trend, ktorý rastie pri každom pozorovaní o jednu jednotku. Bázický dátum znamená pozorovanie, kedy sa začne rátať začiatok trendu. Ak máme časový rad od roku 1993 a trend v ňom rozpoznávame až od 1995, môžeme zadať @trend(1995).

Druhou verziou je

```
@trendc("bázický_dátum")
```

Tento príkaz vráti trend, ktorý vzrastá podľa kalendárnych dní (períód) medzi jednotlivými pozorovaniami. Opäť môže byť zadaný aj bázický dátum, od kedy chceme, aby sa nám začal trend rátať. Začiatok trendu znamená, že má časový rad hodnotu nula. Oba trendy v pracovnom hárku s regulárnou periódou vrátia rovnaký trend. Oba trendy sa zvyšujú o jednu jednotku pri ďalšom pozorovaní. Rozdiel je pri neregulárnej periodicite, napr. pri dennej periodicite s vynechaným víkendom, kde dáva druhý typ trendu (trendc) iné hodnoty, nakoľko pripočíta po každom víkende o 2 jednotky viac. Oba trendy sa teda rozlišujú tým, či berieme do úvahy aktivitu, ktorú merajú, ako nepretržite rastúcu (klesajúcu), alebo rastúcu (klesajúcu) iba počas výskytu danej aktivity – napr. či produkcia automobilov rastie celý týždeň nepretržite, alebo počas víkendu je zastavená.

Pri ekonomických časových radoch by mal trend odrážať skôr počet pracovných dní alebo počet dní v mesiaci. Trend v pracovnom hárku s regulárnou periodicitou, a s ohľadom na rozdielny počet kalendárnych dní, ľahko získame vytvorením trendu resp. časového radu s trendom pomocou ďalších príkazov časových funkcií (doteraz sme poznali len @seas):

@datediff – vráti počet časových jednotiek medzi zadanými dátumami pri zohľadnení určenej časovej jednotky

```
@datediff(dátum1, dátum2[, časová_jednotka])
```

Napr. pri mesačnej periodicite pracovného hárku medzi januárom 2000 a januárom 2001 je počet dní:

```
@datediff(730000, 730365, "dd")
```

String "dd" je počet dní, "mm" by bol počet mesiacov a "ww" počet týždňov. Počet 730000 je počet dní od roku 0 – 365 x 2000 = 730000, t.j. náš odhadnutý počet dní. Tento počet nemusíme vždy odhadovať z hlavy, prípadne pomocou kalkulačky. Na to nám slúži ďalšia časová funkcia:

@dateval – premení označenie dátumu (pozorovania) v pracovnom hárku z reťazcovej formy na číselnú.

```
@dateval("dátum", "formát_dátumu")
```

príklad:

```
scalar sdaysdif3 = @dateval("2000m1")
```

Formát dátumu nemusíme udávať, predvolené nastavenie je mm/dd/yy. Rovnako príkazy berú aj formát dátumu ako ho uvádzame pri stanovovaní Sample – 1993m1. Tento príkaz vráti hodnotu 730019, teda náš predošlý odhad 730000 nebol presný. Následne môžeme spojiť predošlé dva príkazy:

```
series dda = @datediff(@dateval("1993m1"), @dateval("2009m12"), "dd")
```

Hodnota časového radu bude –6178 (to jest počet dní medzi decembrom 2009 a januárom 1993, preto tá mínusová hodnota), avšak hodnota bude rovnaká pre všetky pozorovania v časovom rade DDA. To nebolo našim cieľom. Preto použijeme ešte ďalšiu časovú funkciu.

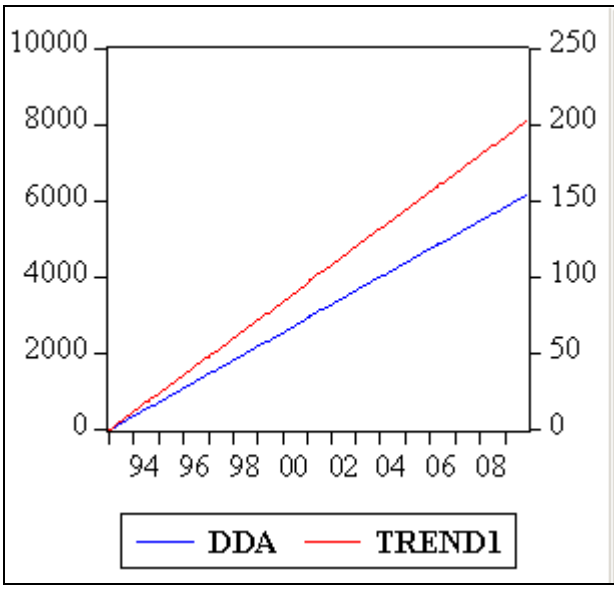
```
@date
```

Táto funkcia vráti reťazec vo formáte mesiac, deň, rok: mm/dd/yy. Výhodou je, že vráti daný reťazec pre každé pozorovanie v aktuálnom Sample pracovného hárku, t.j. dokážeme obsadiť každé pozorovanie správnou hodnotou @dateval. Druhou chybou v našom programe je mínusová hodnota diferencie medzi dátumami. To znamená, že druhý argument bude začiatkom nášho trendu a nie koncom:

```
series dda = @datediff(@date, @dateval("1993m1"), "dd")
```
alebo rovnako, s iným formátom dátumu
```
series dda = @datediff(@date, @dateval("1/1/1993"), "d")
```

Takýto výraz nám vytvorí časový rad DDA s trendom. Môžeme trendy vytvorené pomocou jednoduchého príkazu @trend a DDA môžeme porovnať:

Obr. 4.11 Trendové časové rady



Group: DDATREND Workfile: SEASDUMM_MONT

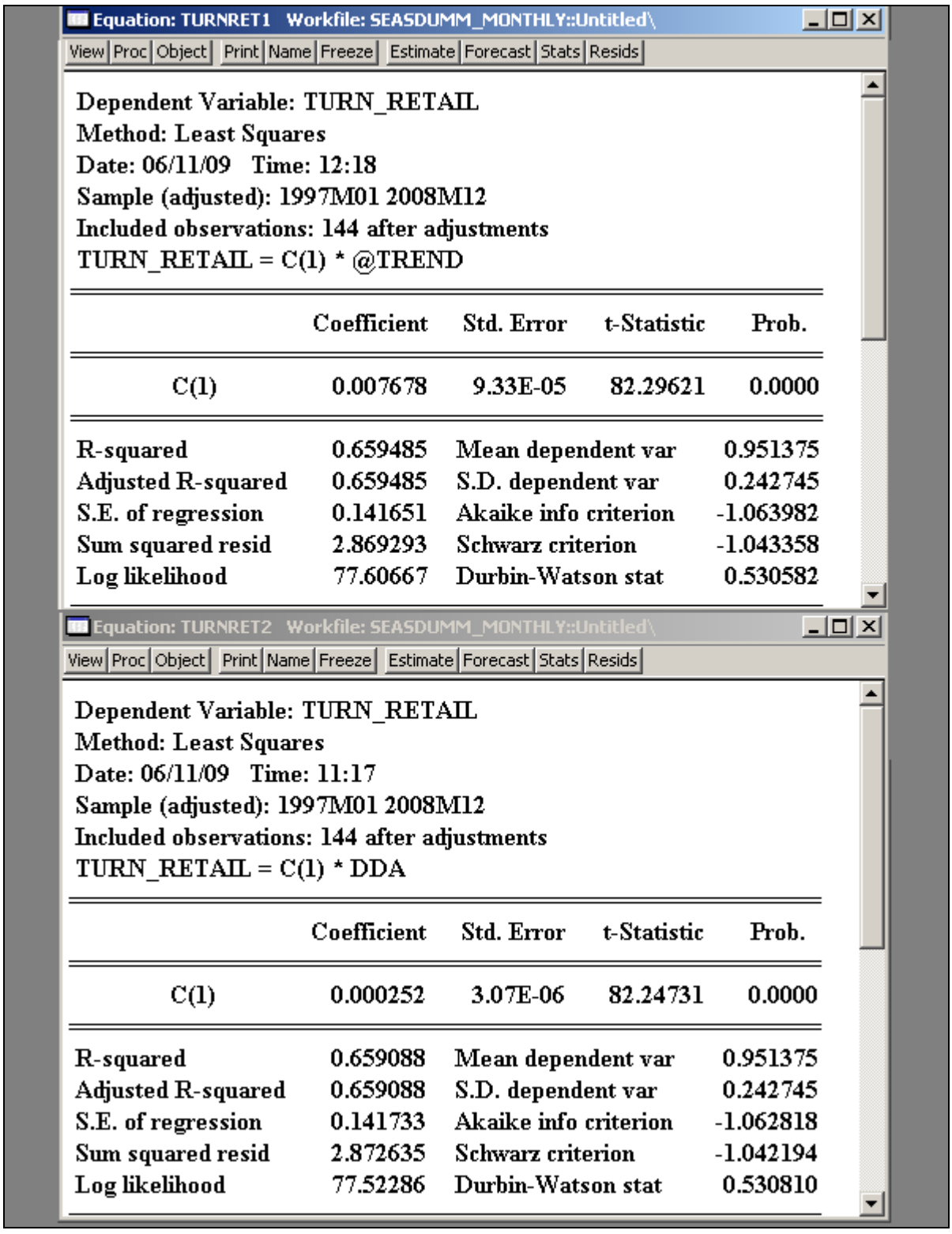
View | Proc | Object | Print | Name | Freeze | Default

obs	DDA	TREND1
1993M01	0.000000	0.000000
1993M02	31.00000	1.000000
1993M03	59.00000	2.000000
1993M04	90.00000	3.000000
1993M05	120.0000	4.000000
1993M06	151.0000	5.000000
1993M07	181.0000	6.000000
1993M08	212.0000	7.000000
1993M09	243.0000	8.000000
1993M10	273.0000	9.000000

Napravo v obr. 4.11 je vidieť, že každé pozorovanie obsahuje v prípade denného trendu kumulatívne súčty dní jednotlivých mesiacov podľa kalendára a trend podľa funkcie @trend pripočítava v každom pozorovaní jednu jednotku. Krivky jednotlivých trendov majú iný sklon

(graf je normalizovaný), čo je vidieť aj na výške koeficientov regresnej rovnice pri závisle premennej tržieb v maloobchode a nezávisle premennej @trendu na obr. 4.12.

Obr. 4.12 Trendové časové rady ako vysvetľujúce premenné



Equation: TURNRET1 Workfile: SEASDUMM_MONTHLY::Untitled\

View | Proc | Object | Print | Name | Freeze | Estimate | Forecast | Stats | Resids

Dependent Variable: TURN_RETAIL
Method: Least Squares
Date: 06/11/09 Time: 12:18
Sample (adjusted): 1997M01 2008M12
Included observations: 144 after adjustments
TURN_RETAIL = C(1) * @TREND

	Coefficient	Std. Error	t-Statistic	Prob.
C(1)	0.007678	9.33E-05	82.29621	0.0000

R-squared	0.659485	Mean dependent var	0.951375
Adjusted R-squared	0.659485	S.D. dependent var	0.242745
S.E. of regression	0.141651	Akaike info criterion	-1.063982
Sum squared resid	2.869293	Schwarz criterion	-1.043358
Log likelihood	77.60667	Durbin-Watson stat	0.530582

Equation: TURNRET2 Workfile: SEASDUMM_MONTHLY::Untitled\

View | Proc | Object | Print | Name | Freeze | Estimate | Forecast | Stats | Resids

Dependent Variable: TURN_RETAIL
Method: Least Squares
Date: 06/11/09 Time: 11:17
Sample (adjusted): 1997M01 2008M12
Included observations: 144 after adjustments
TURN_RETAIL = C(1) * DDA

	Coefficient	Std. Error	t-Statistic	Prob.
C(1)	0.000252	3.07E-06	82.24731	0.0000

R-squared	0.659088	Mean dependent var	0.951375
Adjusted R-squared	0.659088	S.D. dependent var	0.242745
S.E. of regression	0.141733	Akaike info criterion	-1.062818
Sum squared resid	2.872635	Schwarz criterion	-1.042194
Log likelihood	77.52286	Durbin-Watson stat	0.530810

Ak v prípade sezónnych dummy premenných boli tieto identické v oboch postupoch, i vlastnom pomocou príkazov i pomocou domácich funkcií, pri trendových časových radov je to podobné. Vlastnú @trend funkciu dokážeme nakódovať veľmi jednoducho.

Najprv určíme začiatok trendu:

```
smpl 1993m1 1993m1
  genr t = 0
```

A pre ďalšie roky:

```
smpl 1993m2 2009m12
  genr t = t(-1) + 1
```

V tomto prípade je náš trendový časový rad T identický s TREND1 z predošlého príkladu. Druhý typ časového radu (DDA) a jeho vytvorenie sme si predstavili vyššie. Takto máme k dispozícii oba trendy v požadovanej forme, t.j. takej, akú môžeme využívať i pri diferenciách atď. v zložitejších výrazoch v rovniciach.

4.3 Ďalšie vlastnosti regresných odhadov

4.3.1 Metóda najmenších štvorcov

Na regresné odhady rovníc v tejto príručke využívame iba metódu najmenších štvorcov. Anglické pomenovanie je LEAST SQUARES a teda skratka pri kódovaní je LS. Ako bolo uvedené na začiatku kapitoly o jednorovnicových odhadoch, reprezentácia funkcie je možná dvomi spôsobmi – vymenovaním členov a úplnou špecifikáciou rovnice. Prvú reprezentáciu využívame hlavne ak vpisujeme priamo do okna pri vytváraní funkcie konkrétnu špecifikáciu (napr. cez OBJECT/NEW OBJECT../EQUATION...). Pri vymenovaní členov funkcie EViews automaticky predpokladá lineárny vzťah medzi vymenovanými premennými a priradí každej premennej automaticky koeficient C(1), C(2) atď. V nasledujúcom príklade je medzi dvoma premennými i písmeno C, čo znamená, že chceme aby EViews priradil do rovnice aj konštantu (intercept), ktorú automaticky nepredpokladá.

```
LS log(HDP) C log(CONPROD)
```

Pri kódovaní budeme radšej využívať druhý typ špecifikácie regresnej rovnice, pri ktorom vypisujeme celý tvar rovnice:

```
LOG(HDP) = C(1) + C(2) * LOG(CONPROD)
```

Tento typ zadania rovnice je explicitný, nakoľko samostatne rozhodneme už pri samotnej špecifikácii o lineárnosti resp. nelineárnosti vzťahu, ktorý chceme interpretovať pomocou regresie.

Pripomenieme si všeobecný syntax vytvorenia objektu rovnice – equation:

```
equation názov_rovnice.metóda(možnosti) výraz
```

Príkaz na vytvorenie rovnice, s ktorou sme pracovali je nasledovný:

```
equation equ1.ls log(hdp) = c(1) + c(2) * log(conprod)
```

Treba si pripomenúť, že názvy koeficientov (parametrov) jednotlivých premenných v regresii sme zvolili rovnaký ako je nastavený v prostredí EViews. Ak chceme mať názvy koeficientov vlastné, vytvoríme si predtým objekty koeficientov podľa návodu uvedenom v úvode kapitoly 4.

Metódu najmenších štvorcov je možné využiť okrem regresie v rovniciach i pri regresii prierezových dát (panel equations), pri poolových rovniciach, VAR metóde atď. Všeobecný syntax príkazu LS pre jednoduchú regresiu je nasledujúci:

názov_rovnice.ls(*možnosti*) *y x1 x2 x3 ...*
alebo
názov_rovnice.ls(*možnosti*) *špecifikácia*

V prvom prípade ide o reprezentáciu formou vymenovania, kde y je závisle premenná a x1, x2 atď. sú nezávisle premenné. V druhom prípade ide o vytvorenie rovnice pomocou úplnej špecifikácie.

Doteraz sme využili metódu najmenších štvorcov iba vo svojej predvolenej forme, teda bez využitia dodatočných možností, ktoré sa dajú špecifikovať v zátvorke ihneď za skratkou LS. Existuje viacero detailnejších špecifikácií, ktoré sú uvedené v *EViews Command and Programming Reference*. Tieto sa zriedka využívajú, preto uvedieme iba popis vybraných:

ls(m) – maximálny počet iterácií
ls(w=názov_časového_radu) – vážená metóda najmenších štvorcov, pri ktorej sa každé pozorovanie vynásobí uvedeným časovým radom
ls(h) – štandardné chyby budú konzistentné s Whiteovou heteroskedasticitou
ls(n) – štandardné chyby budú konzistentné s Newey-Westovou heteroskedasticitou a autokoreláciou

Uvedené možnosti je možné zadať i bez programovacieho jazyka. Po odhadnutí rovnice sa nachádzajú v objekte rovnice v okne Equation a následne Options, vedľa Specification, ktoré je prednastaveným náhľadom na špecifikáciu objektu rovnice.

4.3.2 Testy pre regresné odhady

Na záver 1. časti príručky uvedieme základné informácie o testoch spojených s regresnými odhadmi. V prvom prípade pôjde o testy špecifikácie rovnice a stability rovnice, ďalej testy odhadnutých koeficientov a nakoniec testy rezíduí z rovníc.

Testy špecifikácie a stability (Specification and Stability Tests)

V krátkosti si môžeme predstaviť niekoľko testov, ktoré overia stabilitu parametrov odhadnutých pomocou regresie. Medzi tieto testy patria*:*

1. Chow´s test bodov zlomu (prerušenia) – Chow´s Breakpoint Test, Chow´s Forecast Test – pomocou týchto testov sa snažíme nájsť bod zlomu (štrukturálnu zmenu), pri ktorom sa menia koeficienty (parametre) odhadnutej funkcie (napr. ropný šok).

Všeobecný zápis príkazu:

chow(*možnosti*) *pozorovanie1 pozorovanie2 ....*

Prednastaveným testom je prvý z týchto dvoch testov – Chow´s Breakpoint test. Ak chceme vykonať Chowov Forecast Test, musíme uviesť do zátvorky v možnostiach písmeno f. V tom prípade musíme uviesť na rozdiel od Breakpoint testu jediný bod obratu – teda jedno časové obdobie za príkazom:

```
chow(f) 1992m1
```

2. Ramseyho regresný test chybnej špecifikácie (RESET) – RESET znamená skratku Regression Specification Error Test. RESET test identifikuje chyby špecifikácie odhadu ako napríklad: vynechaná premenná, nesprávna funkcionálna forma, korelácia medzi x a náhodnou zložkou atď. Všetky tieto chybné špecifikácie majú spoločné to, že produkujú nenulový priemer vektora náhodnej zložky. Preto tento test testuje práve priemer náhodnej zložky.

```
reset(n, možnosti)
```

kde n je číslo exponentu, ktoré sa použije v regresii v teste. Možnosti (Options) obsahujú iba jednu voľbu a to je p – Print pre tlač výsledkov.

Príklad:

```
equation rovnica1.ls hdp c turnind turncon
rovnica1.reset(2)
```

3. Rekurzívne rezíduá – sú rezíduá odhadnuté ako chyba odhadu prognózy o jeden krok vpred (založená na odhade t-1). Rekurzívne rezíduá sú testované na základe nasledujúcich testov:

3a. Cusum test – kumulatívna suma rekurzívnych rezíduí. Tento test zobrazí túto sumu s 5% hraničnou oblasťou. Ak sa kumulatívna suma nachádza mimo tejto oblasti, naznačuje to nestabilitu parametrov rovnice.

3b. Cusum of Squares Test – naznačuje na rozdiel od Cusum testu nestabilitu rozptylu parametrov mimo kritických 5% línií.

3c. One-Step Forecast Test – test prognózy o jeden krok vpred (jednostupňová prognóza) – teda samotné rezíduum odhadnuté ako odchýlka na t-1 období sa porovná so štandardnou odchýlkou celého obdobia. Na grafe sa porovnávajú rezíduá a štandardné chyby a v dolnej časti grafu sú naznačené pravdepodobnosti jednotlivých odhadov. Ak sú nižšie ako 0.05 (p-hodnoty) korešponduje to s tým, že rekurzívne rezíduá sa dostávajú mimo dvoch hraníc štandardných odchýlok (teda naznačuje to nestabilitu)

3d. N-Step Forecast Test – n-krokový test prognóz – využíva rekurzívne kalkulácie na Chowov test, ale na rozdiel od Chowovho Forecast testu automaticky vypočíta všetky kombinácie období pre prognózy (pridávaním jedného pozorovania).

3e. Rekurzívne odhady koeficientov – umožňuje sledovať vývoj koeficientov, ak je odhad spravený na stále väčšom množstve pozorovaní. Ak krivka koeficientu vykazuje veľkú variáciu, je to znakom nestability. Popri koeficientoch sú zobrazené i štandardné chyby okolo odhadnutých koeficientov.

Rekurzívne rezíduá možno aplikovať všeobecným zápisom:

názov_rovnice.rls(*možnosti*) c(1) c(2)...

Následne pomocou možností možno aplikovať konkrétne dané testy:

r – zobrazí rekurzívne rezíduá spolu so štandardnou chybou

r,s – zobrazí rekurzívne rezíduá a štandardné chyby a zároveň ich uloží ako časový rad (pod menom R_RES a R_RESSE...)

c – zobrazí rekurzívne koeficienty s ohraničeniami štandardných chýb

c, s – zobrazí rekurzívne koeficienty spolu so štandardnými chybami a zároveň ich uloží pod menami R_C1, R_C2.. pre koeficienty a R_C1SE, R_C2SE atď. pre štandardné chyby

o – zobrazí p-hondoty rekurzívneho jednokrokovej prognózy Chowovho prognostického testu

n – zobrazí p-hodnoty rekurzívneho n-krokovej prognózy Chowovho testu.

q – zobrazí CUSUM rezíduá a 5%-né kritické línie

v – zobrazí CUSUMQ štatistiku a 5%-né kritické línie

Testy koeficientov

Medzi základné testy odhadnutých parametrov (Coefficients Tests) patria:

1a. Confidence Ellipses (konfidenčné elipsy) – zobrazuje vzájomné konfidenčné intervaly dvoch funkcií z odhadnutých parametrov.

1b. Wald Test – Waldov test vyráta štatistiky na neobmedzenej regresii, ak sú reštrikcie správne, tak odhady bez reštrikcií by sa mali približovať k uspokojivým reštrikciám

1c. Ommited Variables – vynechané premenné – vykoná testy, či premenné pridané k existujúcej rovnici prispejú k vysvetleniu rozptylu závisle premennej.

1d. Redundant Variables Test – testuje štatistickú významnosť podvýberu zo zahrnutých premenných v rovnici.

Všeobecné zápisy jednotlivých testov koeficientov:

názov_rovnice.cellipse(*možnosti*) *reštrikcie*

napr.

rovnica1.cellipse c(1), c(2)

zobrazí graf (elipsu) s 95 % spoľahlivosťou pre C(1) a C(2)

názov_rovnice.wald *reštrikcie*

napr.

rovnica1.wald c(1) =0, c(2)=0

názov_rovnice.testdrop *parametre*

napr.

```
rovnica1.testdrop turnind
```

```
názov_rovnice.testadd parametre
```

napr.

```
rovnica1.testadd turntrans
```

Testy rezíduí

Rezíduá odhadnuté v rovniciach sa testujú v nasledujúcich testoch:

- *Correlograms and Q-statistics (korelogram a q-štatistika)* – zobrazuje autokoreláciu a parciálnu autokoreláciu rezíduí
- *Correlogram of Squared Residuals (korelogram štvorcových rezíduí)* – tento test sa využíva na hľadanie autoregresívne podmienenej heteroskedasticity v rezíduách (ARCH – autoregresive conditional heteroskedasticity). Ak nie je ARCH v rezíduách, tak by mali byť autokorelácie nulové v každom oneskorení (lag) a Q-štatistika by mala byť nevýznamná
- *Histogram a test normality* – tento test zobrazuje základné deskriptívne štatistiky rezíduí; ak sú rezíduá normálne rozdelené, mal by byť histogram v tvare zvonu a Jarque-Bera štatistika nevýznamná.
- *LM test seriálovej korelácie* – je alternatívou Q-štatistiky. Pomocou tohto testu sa hľadajú závisle premenné posunuté v čase (oneskorené).
- *ARCH LM Test* – hľadá autoregresívne podmienenú heteroskedasticitu ARCH v rezíduách.
- *Whietov test Heteroskedasticity* – testuje heteroskedasticitu v rezíduách.

Všeobecné zápisy uvedených testov. Treba si pripomenúť, že nasledujúce príkazy patria medzi už spomenuté náhľady na objekt rovnice (obr. 4.6, str. 63).

```
názov_rovnice.correl(n, možnosti)
názov_rovnice.correlsq(n, možnosti)        ‘ n je počet oneskorení (lags)
názov_rovnice.auto(stupeň, možnosti)       ‘ stupeň (order), ktorý chceme testovať
názov_rovnice.hist(možnosti)
názov_rovnice.archtest(možnosti)
názov_rovnice.white(možnosti)
```

Register príkazov

Zdroje

1. EViews-Help - Copyright © 1994-2004 Quantitative Micro Software, LLC. All rights reserved. Help system build: April 16, 2004. http://www.eviews.com.

2. *Haluška J., Olexa M., Juriová J., Kľúčik M.:* Modelový aparát na rýchle odhady vývoja makro-ekonomických ukazovateľov slovenskej ekonomiky (Využitie konjunkturálnych a spotrebiteľských prieskumov), INFOSTAT, december 2008

3. *Kľúčik M., Haluška J.:* Construction of Composite Leading Indicator for the Slovak Economy. Scientific Annals of the "Alexandru Ioan Cuza", University of Iasi, Romania, 2008 (p. 363 – 370)

Vydal: INFOSTAT
Inštitút informatiky a štatistiky,
Dúbravská cesta 3
845 24 Bratislava 45

V edícii: Dokumenty

Pod číslom: 1

Vedúci redaktor: Ing. Michal Olexa, PhD.

Výkonný redaktor: Ing. Miroslav Kľúčik

Počet strán: 92

Počet výtlačkov: 9

ISBN: 978-80-89398-15-7

D1-1095/2010

ISBN 978-80-89398-15-7