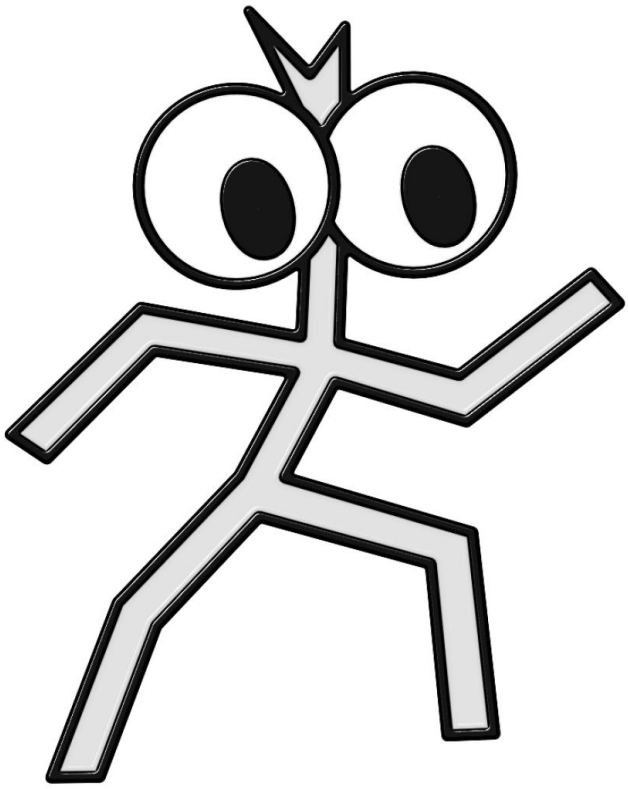
Programovací jazyk DARWin

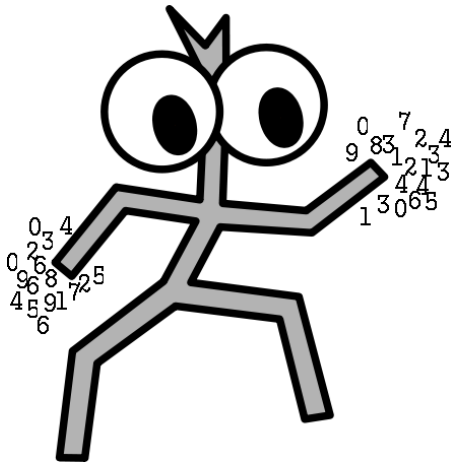
DATA **A**NALYSIS **R**OBOT for **Win**dows



Definice a popis

Programovací jazyk DARWin

Definice a popis



1. Obsah

2. Obecný úvod

Jazyk DARWin (Data Analysis Robot for Windows) je úplný interpretovaný algoritmický aritmetický jazyk, umožňující práci s daty v QCExpertu, maticemi a vektory, vytváření řídících programových struktur (jako for, while, if), definici funkcí s možností rekurze a interakci s okolím pomocí vstupních a výstupních procedur. Je součástí interaktivního prostředí QCExpert a běží pouze v tomto prostředí. Součástí jazyka je interaktivní programové vývojové prostředí, knihovna standardních funkcí, knihovna matematických funkcí, knihovna základních statistických funkcí, grafických funkcí, datových a I/O funkcí, funkcí lineární algebry a knihovna statistických metod.

2.1. Terminologická a syntaktická pravidla jazyka a manuálu

V tomto manuálu se používají některá typografická pravidla pro usnadnění orientace v textu. Skripty v jazyku DARWin jsou tištěny fontem `Courier`, obvykle bez tučného zvýraznění klíčivých slov. Názvy položek menu, oken, názvy objektů uživatelského prostředí a názvy kláves jsou tištěny *kurzívou*. V definici příkazů a funkcí, především v kapitole 7, se používá standardní notace pro programovací jazyky. Velikost písmen při zápisu příkazů, funkcí, jmen proměnných nehraje roli, například zápisy PRINT a PrinT jsou ekvivalentní. Rovněž jsou ignorovány mezery, pokud se nevyskytují ve jménu objektu, nebo v textovém řetězci. Obecně jsou dodržována následující pravidla:

Povinný text je psán základním fontem, například `DELETEVARS`.

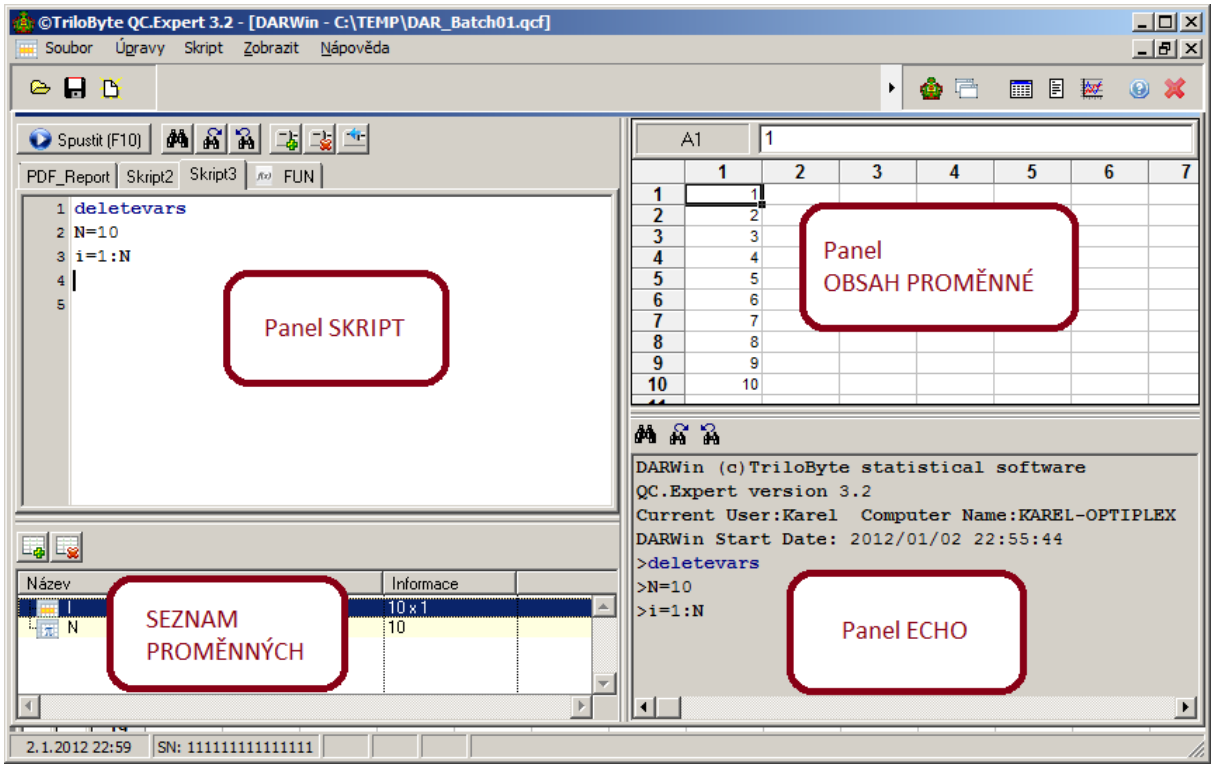
Nepovinný text (například nepovinné argumenty funkcí) jsou v hranatých závorkách., například `NORMALR(`*`N`* `[, MEAN=`*`X`*`] [, SDEV=`*`S`*`])` znamená povinné zadání argumentu N a nepovinné zadání MEAN=X, případně SDEV=S.

Symboly a jména, které se musí nahradit aktuálními jsou tištěna kurzívou, například argument X v `ABS(`*`X`*`)`.

Alternativní texty, kdy je třeba zvolit jednu z několika možností jsou odděleny kolmicí, například `LINEADD(H=Y|V=X|A=I,B=S)` znamená, že je nutné zadat jednu ze tří možností buď H=Y, nebo V=X, nebo současně A=I a B=S

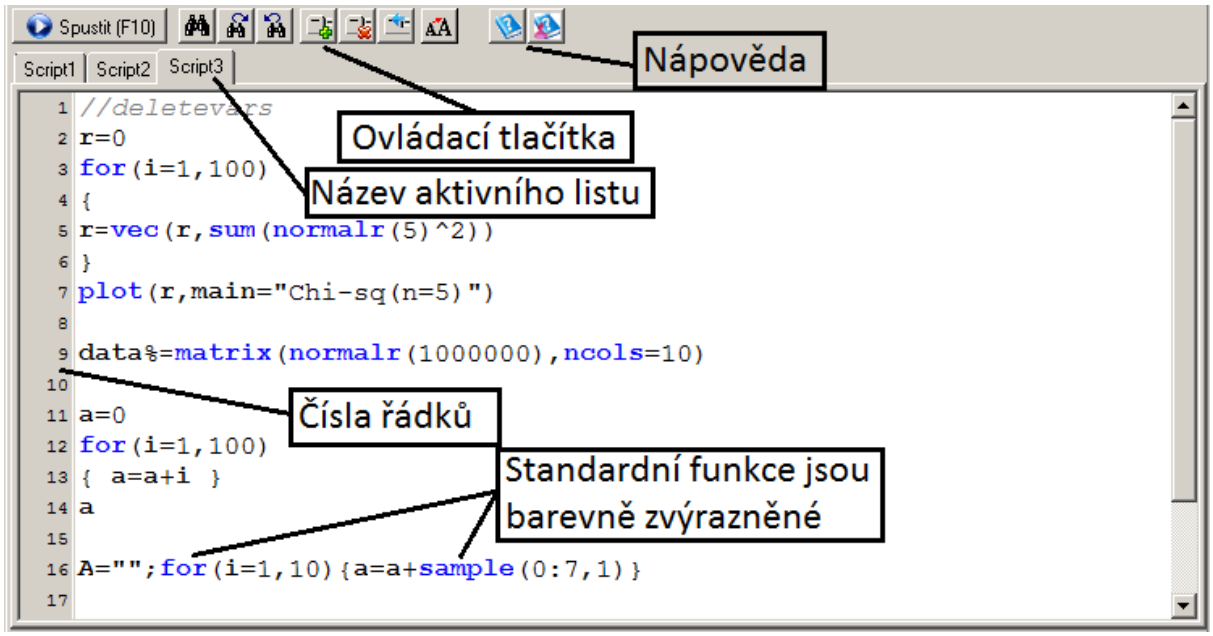
Pokračování řady je znázorněno tečkami, například `BIND(`*`M1`* `[,` *`M2`*`,` *`M3`*`, ...]).`

3. Interaktivní prostředí jazyka DARWin



Interaktivní prostředí DARWinu má 4 panely. Panel Skript, panel Seznam proměnných, panel Obsah proměnné a panel Echo, jak se znázorněno na obrázku.

3.1. Panel Skript



Panel Skript slouží jako textový editor pro zápis skriptu (programu) v jazyce DARWin. Skript se zapisuje do jednotlivých listů, které lze v panelu přidat, odebrat a přejmenovat. Při prvním spuštění je v panelu jeden list s názvem *Skript1*. Požadovaná část skriptu se spustí tak,

že se text označí (vybere) a tato označená část se spustí klávesou *F10*, nebo tlačítkem *Spustit (F10)*. Pokud není vybrán žádný text, spustí se celý skript zapsaný v daném listu. Tímto způsobem lze spouštět libovolné souvislé fragmenty skriptu. Spuštěné příkazy, výrazy a hodnoty výrazů se zapisují do panelu Echo, který je tak zápisem historie práce. Jednotlivé řádky jsou uvozeny znakem „>“ (tzv. prompt). Je-li spuštěn pouze výraz, zapíše se do panelu Echo kromě samotného výrazu i jeho hodnota. Před hodnotami se prompt „>“ nezobrazuje. Obsah panelu Echo se po zavření okna DAR nebo při ukončení programu QCExpert vymaže. Termínem „spuštění“ nebo „spuštění skriptu“ budeme označovat stav mezi okamžikem zahájení výpočtu stiskem klávesy *F10* (popř. tlačítkem *Spustit (F10)* 

) a okamžikem ukončení přirozeným skončením skriptu, chybou, nebo dalším stiskem klávesy *F10*, popř. tlačítkem *Stop (F10)* 

.

Příklad

V následující tabulce uvádíme v levém sloupci obsah skriptového listu s vyznačením vybrané části textu, která byla spuštěna klávesou *F10*, v pravém sloupci je uveden výpis v panelu Echo. Označením textu a spuštěním klávesou *F10* získáme následující výsledky v panelu Echo.

Panel Skript (označení textu a stisk *F10*)	Panel Echo

`A=normalr(5)/5+15`
 	
/* Spuštění výrazu, jehož hodnota není nikam přiřazena, proto se vypíše v panelu Echo, včetně samotného výrazu. Výsledkem výrazu je sloupcový vektor s pěti prvky. */
	`>normalr(5)/5+15`
	`14.8158476122516`
	`15.1939199213969`
	`15.1542583361369`
	`14.7639928023267`
	`14.9646688732564`

`A=normalr(5)/5+15`
 	
/* Spuštění části předchozího výrazu. */
	`>normalr(5)`
	`-1.12242321838977`
	`2.18020313462297`
	`-0.401253593830604`
	`0.00964743490463493`
	`-1.54085859868898`

`A=normalr(5)/5+15` /* Spuštění jiné části téhož výrazu */	`>5+1` `6`
`A=normalr(5)/5+15` /* Spuštění celého přiřazovacího příkazu, hodnota je přiřazena do proměnné A, proto se v panelu Echo nevypisuje obsah proměnné, pouze se zkopíruje provedený výraz. */	`>A=normalr(5)/5+15`

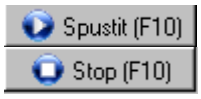
`...`
`N=1000`
`print(sqrt(r))`
 	
`a=0`
	`>a=0`

```
for(i=1,100)
{ a=a+i }
a

A=vec(2,0,4,0,2,1,0,0,0,2)
...

/* Spuštění požadované části programu a výpis
proměnné a. */
```

```
>for(i=1,100)
>{
>a=a+i
>}
>a
5050
```

Rozpoznané příkazy a funkce jsou ve skriptu a v panelu Echo zobrazeny modře a tučně, názvy proměnných černě a tučně, operátory, čísla a textové řetězce černě. Komentáře jsou zobrazeny šedě a kurzívou (italic). Komentáře se do panelu Echo nepřepisují. Řádky listu se skriptem jsou pro lepší orientaci číslovány. Pro práci s textem v panelu Skript platí obvyklá pravidla pro editaci textu, operace „ZPĚT“ („undo“) lze provést stiskem *Ctrl-Z*. Hledání v textu je možné pomocí *Ctrl-F*.

Ovládací tlačítka panelu Skript



Spuštění skriptu (toto tlačítko je ekvivalentní klávese *F10*). Při běžícím skriptu se tlačítko *Spustit* změní na *Stop* a slouží k přerušení běžícího skriptu.

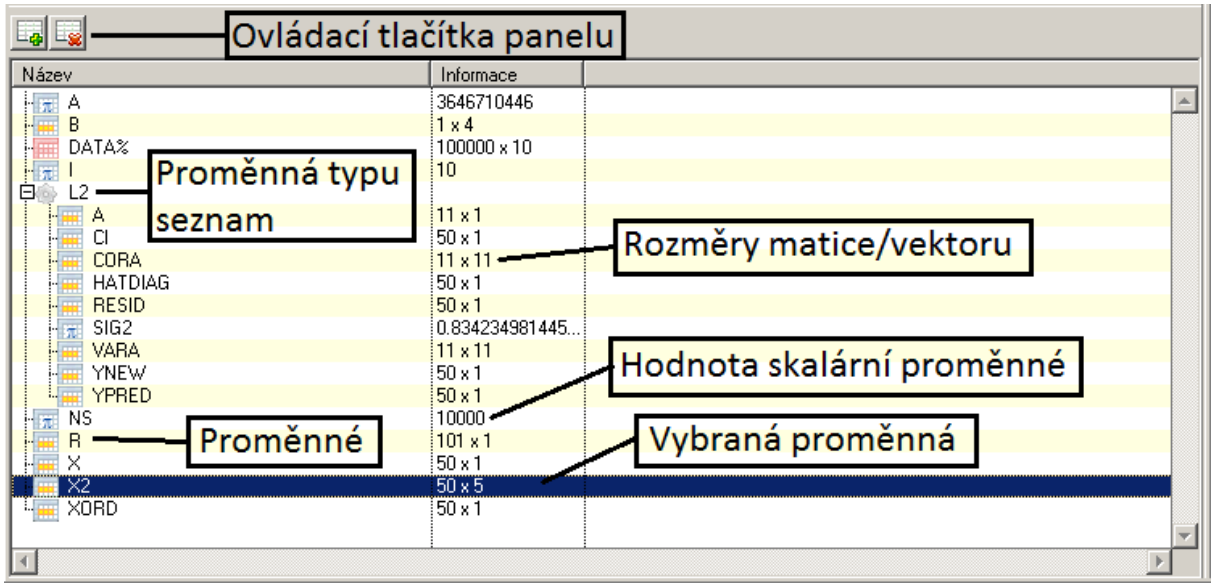
Vyhledávání v textu, ekvivalent je stisknutí kláves *Ctrl-F*.

Správa listů v panelu Skript. Tlačítka pro přidání nového skriptového listu, smazání aktivního skriptového listu a přejmenování aktivního skriptového listu.

Výběr velikosti písma.

Standardní nápověda DARWin a dynamická nápověda k aktivním uživatelským funkcím

3.2. Panel Seznam proměnných



V tomto panelu se vytváří seznam vytvořených proměnných. Ve sloupci *Název* jsou názvy proměnných, u proměnných typu „seznam“ i struktura proměnné. Ve sloupci *Informace* je hodnota proměnné, v případě vektoru nebo matice je zde uveden rozměr v pořadí [počet řádků] x [počet sloupců]. Kliknutím na řádek v panelu Seznam proměnných se zobrazí obsah příslušné proměnné v panelu *Obsah proměnné*, kde jej lze editovat. Dvojité kliknutí na proměnnou zapíše její název na pozici kurzoru do skriptového listu.

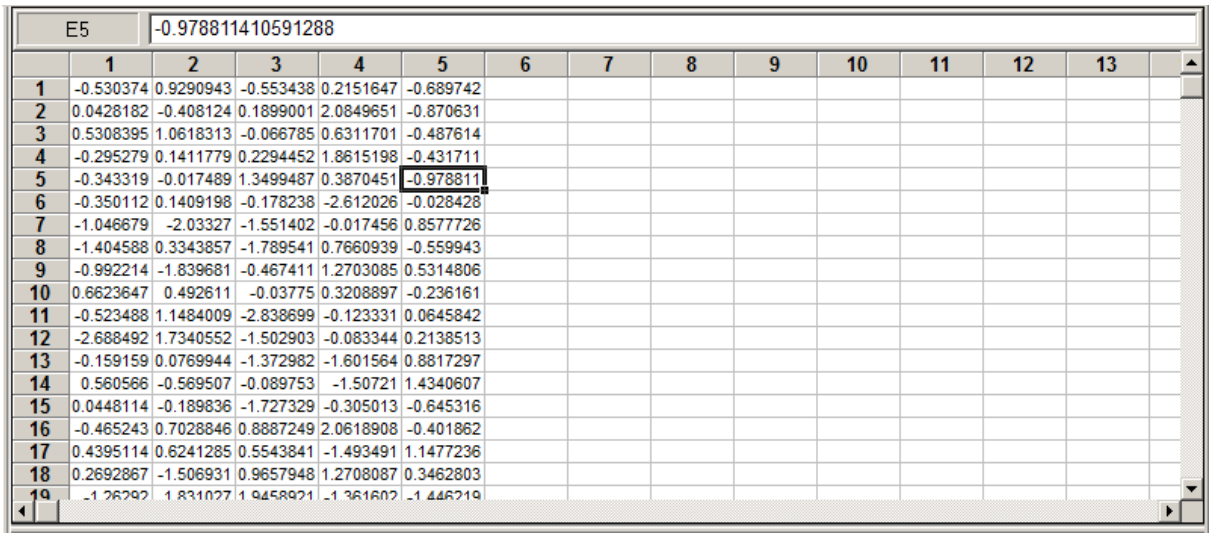
Ikony pro různé typy proměnných

- Skalár (jediná hodnota) proměnná, číslo nebo textový řetězec
- Matice / Vektor
- Proměnná typu seznam, s viditelnými jednotlivými prvky seznamu
- Proměnná Big Data (skalár, vektor, nebo matice, název proměnné končí znakem %)
- Proměnná s nedefinovanou hodnotou

Ovládací tlačítka panelu Seznam proměnných

Interaktivní přidání a smazání proměnné. Nově založenou proměnnou lze naplnit přímo v panelu Obsah proměnné.

3.3. Panel Obsah proměnné

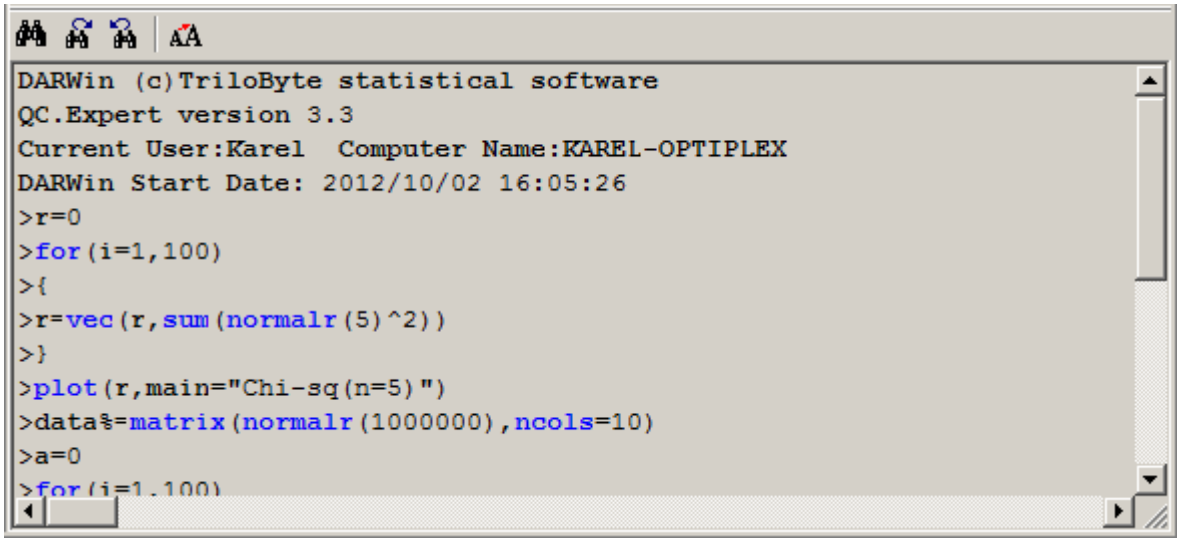


E5 -0.978811410591288

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	-0.530374	0.9290943	-0.553438	0.2151647	-0.689742								
2	0.0428182	-0.408124	0.1899001	2.0849651	-0.870631								
3	0.5308395	1.0618313	-0.066785	0.6311701	-0.487614								
4	-0.295279	0.1411779	0.2294452	1.8615198	-0.431711								
5	-0.343319	-0.017489	1.3499487	0.3870451	-0.978811								
6	-0.350112	0.1409198	-0.178238	-2.612026	-0.028428								
7	-1.046679	-2.03327	-1.551402	-0.017456	0.8577726								
8	-1.404588	0.3343857	-1.789541	0.7660939	-0.559943								
9	-0.992214	-1.839681	-0.467411	1.2703085	0.5314806								
10	0.6623647	0.492611	-0.03775	0.3208897	-0.236161								
11	-0.523488	1.1484009	-2.838699	-0.123331	0.0645842								
12	-2.688492	1.7340552	-1.502903	-0.083344	0.2138513								
13	-0.159159	0.0769944	-1.372982	-1.601564	0.8817297								
14	0.560566	-0.569507	-0.089753	-1.50721	1.4340607								
15	0.0448114	-0.189836	-1.727329	-0.305013	-0.645316								
16	-0.465243	0.7028846	0.8887249	2.0618908	-0.401862								
17	0.4395114	0.6241285	0.5543841	-1.493491	1.1477236								
18	0.2692867	-1.506931	0.9657948	1.2708087	0.3462803								
19	-1.26292	1.831027	1.9458921	-1.361602	-1.446219								

Tento panel obsahuje datovou tabulku, která zobrazuje obsah aktuálně vybrané proměnné (u proměnných typu „seznam“ se zobrazuje obsah jen u jednotlivých položek seznamu). V tabulce lze proměnnou upravovat a měnit její obsah, po označení lze kopírovat obsah tabulky do jiných aplikací (Excel, apod.).

3.4. Panel Echo



Do panelu Echo se kopírují všechny příkazy spuštěné ve skriptovém listu a zobrazují hodnoty všech spuštěných výrazů, pokud nejsou přiřazeny do proměnné. Kopírované příkazy a výrazy jsou uvozeny znakem „>“ na začátku řádku, výsledky se vypisují bez tohoto znaku. Obsah panelu Echo nelze upravovat, lze z něj kopírovat (Ctrl-C) a lze v něm hledat (Ctrl-F, případně ovládací tlačítka panelu). Po spuštění DARWinu se na začátek zapíše hlavička ve tvaru:

```
DARWin (c)TriloByte statistical software
QC.Expert version X.X
Current User:Jméno_uživatele  Computer Name:Jméno_počítače
DARWin Start Date: 2011/12/24 18:00:00
```

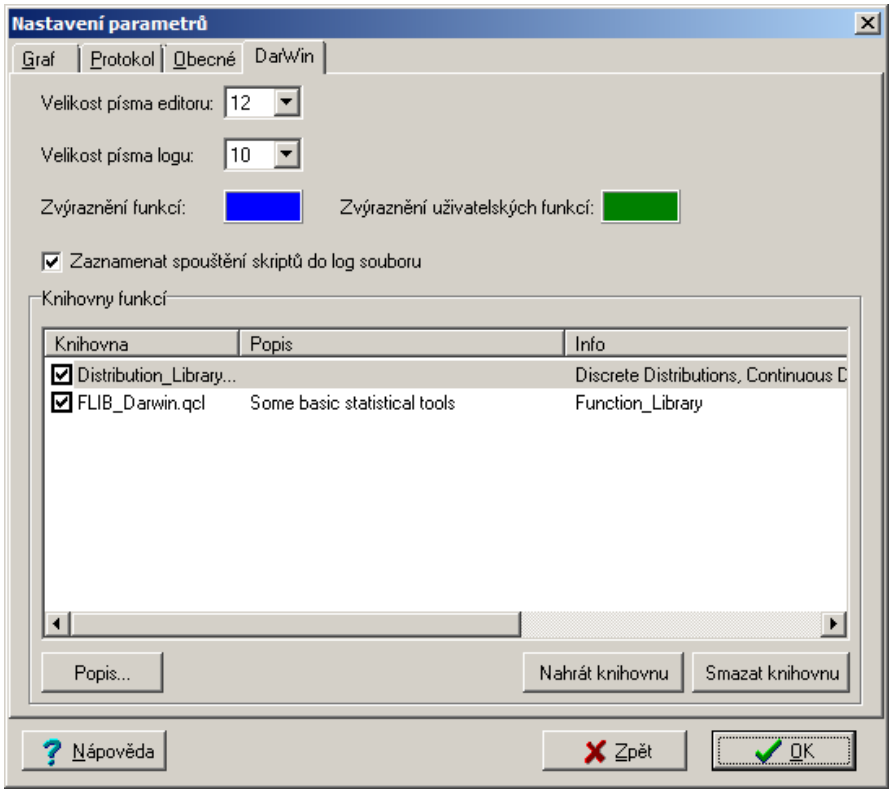
Obsah panelu lze archivovat do LOG-souborů a tím dokumentovat a částečně zálohovat práci, viz následující odstavec.

Ovládací tlačítka panelu Echo

	Tlačítka pro vyhledávání textu (Najdi, Najdi další, Najdi předchozí)

3.5. Log-soubory

Průběh práce se systémem DARWin lze podrobně dokumentovat a zaznamenávat do textových log-souborů. Je-li zvolena možnost *Zaznamenávat spouštění skriptů do log souboru* v nabídce hlavního menu (*Soubor – Nastavení – DARWin*), viz následující obrázek, vytvoří se při každém spuštění DARWinu soubor s názvem ve tvaru `YYYYMMDD_HHMMSS_TTT.log` v podadresáři DARWINLOG pracovního adresáře (Menu: *Soubor – Nastavení – Obecné*). Název souboru představuje datum, čas a tisíciny sekundy okamžiku spuštění DARWinu. Do tohoto souboru se kopíruje obsah panelu Echo popsaného v předchozím odstavci. Po ukončení práce s DARWinem zůstává tento soubor zachován na disku a při dalším spuštění DARWinu se založí nový soubor s jiným jménem. Log-soubory mohou sloužit pro případnou pozdější referenci, rekonstrukci a dokumentaci činnosti a vývojových prací v systému DARWin.



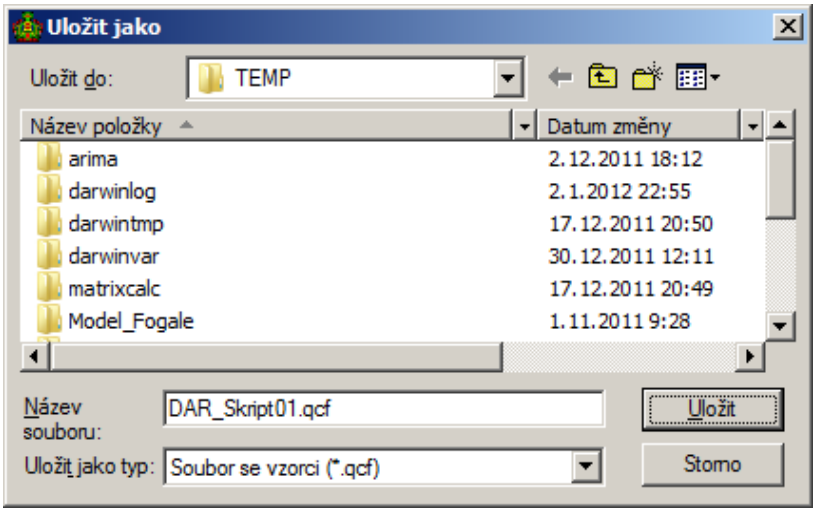
Vzhledem k tomu, že se log-soubory automaticky nemažou, je vhodné sledovat obsah adresáře DARWINLOG, neboť někdy může velikost těchto souborů rychle narůstat, například při častém vypisování rozsáhlých matic do panelu Echo. Mazat nepotřebné log-soubory je možné pouze ručně (je ovšem možné také napsat pro automatické mazání souborů i program v samotném DARWinu).

Log-soubor vždy začíná hlavičkou (viz odst. 3.4) a po zavření DARWinu se na konec souboru zapíše informace s datem a časem ukončení práce ve tvaru `DARWin End Date: YYYY/MM/DD HH:MM:SS`, například:

```
DARWin End Date: 2011/12/31 23:59:59
```

3.6. Soubory systému DARWin

Je-li aktivní okno DARWin, jsou v hlavním menu v položce *Soubor* (kromě obvyklých standardních položek) dostupné položky pro otevření souboru, uložení souboru, otevření nedávných souborů a založení prázdného skriptu. Položka *Uložit soubor Darwin (Ctrl-S)* otevře okno (viz obrázek) s možností *Uložit jako typ*, kde lze vybrat z možností *Soubor se skripty*, kdy se uloží pouze všechny záložky se skripty bez proměnných do souboru s příponou QCF, nebo *Soubor se skripty a proměnnými*, kdy se uloží záložky se skripty včetně obsahu všech proměnných v panelu *Seznam proměnných* do souboru s příponou QCD. V závislosti na obsahu proměnných mohou být soubory QCD značně velké. Třetí možností je uložení souboru jako *Knihovna funkcí DARWin*, kdy se soubor uloží s příponou QCL a funkce v něm definované je možno využívat po aktivaci knihovny již bez otevírání tohoto souboru, jako by byly součástí jazyka DARWin, viz odstavec 6.5, str. 38.



Při prvním spuštění vytvoří DARWin pomocné soubory v pracovním adresáři (implicitně C:\TEMP) složky DARWINLOG, v níž jsou ukládány log-soubory (viz odstavec 3.5), složku DARWINTMP, která slouží pro ukládání dočasných systémových a pomocných souborů a složku DARWINVAR, ve které jsou uloženy proměnné BigData (viz odst. 6.2.9, str. 28). Tento adresář je vhodné občas kontrolovat, případně vymazat soubory s nepotřebnými proměnnými, které mohou zabírat místo na disku.

Přípona	*Implicitní Složka*	*Typ souboru*	*Popis*
QCF	C:\TEMP	DAR Skripty	Obsahuje skripty z jednotlivých záložek v textovém formátu.
QCD	C:\TEMP	DAR Skripty a data	Obsahuje skripty z jednotlivých záložek včetně obsahu všech proměnných v binárním formátu.
QCL	C:\TEMP	DAT Knihovna funkcí	Obsahuje skripty z jednotlivých záložek v textovém formátu. Definice funkcí je možno připojit jako knihovnu, viz 6.5.
LOG	C:\TEMP\darwinlog	Log-soubor	Zaznamenává se jen je-li zapnutá funkce Log, viz odst. 3.5. název souboru se skládá z datumu a času na tisíciny sekundy, kdy byl soubor založen, např. 20111226_232124_787
	C:\TEMP\darwinvar	proměnné Bigdata	Binární soubory bez přípony s obsahem proměnných big data (suffix %). Název souboru odpovídá názvu proměnné. Smazáním souboru se ztratí obsah proměnné v DARWinu, proměnná se označí jako nedefinovaná.

4. Základní struktura a syntaxe jazyka DARWin

4.1. Proměnná

Datová struktura typu vektor, matice, nebo seznam (list) se dá přiřadit do proměnné (neboli: proměnné je přiřazena datová struktura). Může obsahovat buď číselné hodnoty, nebo textové hodnoty. Textové hodnoty se zapisují do uvozovek: "TEXT". Název proměnné musí

začínat písmenem a může obsahovat číslice. V názvu proměnné se nerozlišují velká a malá písmena, XYZ, Xyz, XyZ, xyz je stále tatáž proměnná. Délka názvu proměnné není omezena. Proměnné se přiřadí hodnota pomocí operátoru "=", nebo v interaktivním prostředí jazyka DARWin v panelu *Obsah proměnné*, viz dále. Hodnota proměnné je přístupná buď v datové tabulce v panelu *Obsah proměnné*, nebo zapsáním a spuštěním názvu proměnné. Jednotlivé prvky vektoru, nebo matice jsou přístupné pomocí indexových závorek [ ] a [[ ]], prvky listu jsou přístupné pomocí znaku $ (dolar) a názvu prvku listu. U proměnných rozlišujeme typ a strukturu. Standardní proměnná může obsahovat maximálně matici s 65535 řádky a 256 sloupci. Proměnné BigData, jejichž název končí znakem % (procento) mají neomezenou velikost, viz odst. 6.2.9, strana 28.

Standardní proměnné systému DARWin se interně ukládají do souboru matrixdata.vts v pracovním adresáři (typicky C:\QCEWork). Tento soubor má formát datového sešitu QCExpert a lze také v QCExpertu otevřít. Ukládá se vždy při ukládání skriptu, proměnné odpovídají jednotlivým listům. Této skutečnosti lze využít k obnově standardní proměnné, pokud je omylem vymazána. Maximální počet standardních proměnných je tedy 256, počet proměnných BigData není omezen.

4.2. Datové typy (hodnoty)

4.2.1. Numerická hodnota

Numerická hodnota: Reálné číslo, celá čísla nejsou zvláštním typem. Numerický rozsah číselné hodnoty je $-(2^{1024})$ až $-(2^{-1024})$, 0, 2^{-1024} až 2^{1024}. Vyjádřeno dekadickým zápisem v absolutní hodnotě asi od 5.56268E-309 do 1.797693E308 a nula. Přesnost výpočtů je 15-16 desetinných míst. Dekadický exponent se vkládá standardně, pomocí E nebo e. Ve skriptu se vždy používá desetinná tečka, i když výstupy v panelu Echo, v okně Protokol, Data, v exportu a podobně jsou závislá na nastavení systému a mohou používat i desetinnou čárku.

Příklady možného zápisu čísel

```
>1234.56789
1234.56789
>-5e-10
-5E-10
>.0000000000000000001
1E-19
>100e2
10000
>12345678901234567890
1.23456789012346E19
```

4.2.2. Textové řetězce

Textový řetězec je jednotlivý znak nebo text v uvozovkách. Maximální délka řetězce je 16383 znaků. Operace s řetězci se provádějí pomocí textových funkcí. Spojování řetězců je možné pomocí operátoru sčítání „+“. Řetězce je možné přiřazovat proměnným. V datových strukturách jako vektor nebo matice zabírá libovolně dlouhý řetězec vždy jednu buňku. Zvláštní hodnotou je prázdný znak "". Při použití operátoru „+“ platí, že je-li alespoň jeden z operátorů řetězec, je výsledkem řetězec. Například:

```
>4+5              >"4"+"5"          >"4"+5            >a=normalr(1)
9                 "45"              "45"              >"R="+round(a,3)
                                                      "R=1.409"
```

Při více „sčítancích“ se výraz vyhodnocuje zprava, takže například:

```
>"2"+3+4                    >2+3+"4"
"27"                        "234"
```

Příklad zápisu textového řetězce:

```
>"ABCDEFGH"
"ABCDEFGH"
>"2.718281828"
"2.718281828"
>"Výsledek: "
"Výsledek: "
>"A"+""+"B"
"AB"
//Prázdný znak jako startovací prázdná hodnota (placeholder)
>A="";for(i=1,10){a=a+sample(0:7,1)}
"6723065274"
```

4.2.3. Logická hodnota

Logická hodnota není v DARWinu zvláštní typ. Logická „nepravda“ (FALSE, NO) je reprezentována numerickou hodnotou 0. Logická hodnota „pravda“ (TRUE, YES) je reprezentována hodnotou 1. Logické hodnoty jsou typicky používány v logických operacích (jako AND, OR, NOT), jsou výsledkem relačních funkcí (jako GT, EQ, ISTEXT). Používají se v logickém indexování [[ ]] a v podmíněných příkazech IF a WHILE. Výsledkem logických a relačních funkcí je tedy vždy buď 0 nebo 1:

```
>gt(3,5)          >and(1,0)          >gt(5,vec(3,6,4,-2))
0                 0                  1
>gt(5,3)          >or(1,0)           0
1                 1                  1
                                     1
```

Jakákoliv nenulová hodnota použitá jako logická hodnota je interpretována jako 1:

```
>or(0,777)
1
```

4.2.4. Nedefinovaná hodnota (funkce INI)

Nedefinovaná (inicializační) hodnota se jedné nebo více proměnným přiřadí příkazem `INI`. Tato hodnota znamená pouze existenci proměnné, neurčuje její typ a její délka (počet prvků, pokud ji chápeme jako vektor) je nula. Prázdnou hodnotu nelze použít v aritmetických operacích. Je však možno ji spojit do vektoru s numerickou nebo textovou hodnotou (vektorem, maticí) pomocí funkcí `VEC`, `BIND`, `BINDV`. Přitom se nedefinovaná proměnná přizpůsobí rozměru této matice. Tato hodnota je proto velmi výhodná v cyklech, když se skládají vektory, nebo matice, tabulky, atd. po jednotlivých prvcích, sloupcích apod. V panelu proměnných se ve sloupci *Informace* objevuje text „*Nedefinováno*“.

Příklad definice a použití funkce INI a prázdné hodnoty:

```
INI(A1,A2,A3)     >a2=vec(a2,1:3)   ini(a3)
>count(a1)        >a2               for(i=1,5)
0                 1                 {
```

```
>a1+1                | 2                | A3=bind(a3,i+sample(1:9,3))
Error : "Invalid    | 3                | }
variant             |                  | >a3
operation"          |                  | 3    6    12   12   13
>vec(a1,100)        |                  | 8    11   9    11   14
100                 |                  | 7    10   11   5    12
```

4.2.5. Datum a čas

Datum a čas nejsou zvláštními typy dat. Pro jejich reprezentaci se využívá typu numerická hodnota a textový řetězec. Pomocí datových/časových funkcí lze převádět datum z numerické do textové reprezentace a naopak. Textová reprezentace času je textový řetězec obsahující hodiny (0-23), minuty, sekundy oddělené dvojtečkou a tisíciny sekund oddělené desetinnou tečkou, například "5:36:53.430", nebo "18:58:56.951". Je možné vypouštět začáteční nuly, tedy například zápis "05:05:05" i zápis "5:5:5" jsou platné a totožné. Datum má textový formát ve tvaru den, měsíc, rok oddělené tečkou, například "23.8.2012", nebo "2.12.2012". Dále je definován formát obsahující jak čas tak datum oddělené mezerou, např.

"23.8.2012 10:48:22.483".

Numerická reprezentace data představuje počet dní od 0.1.1900. Desetinná místa představují uběhnuvší část dne od půlnoci, tedy 0.25 je 6 hodin ráno, apod. Počet desetinných míst reálných čísel umožňuje reprezentovat tímto způsobem čas s přesností tisíciny sekundy. Pomocí funkce DATETIMEN (date-time to numeric) a DATETIMES (date-time to string) lze obě reprezentace vzájemně převádět, například:

```
>datetimes(41123.476663)          | >datetimen("20.8.2012 23:15:59")
"2.8.2012 11:26:23.683"           | 41141.9694328704
```

Numerická hodnota data může být záporná, lze zapsat libovolné datum a čas od 1.1.0000 do 31.12.65535.

Časové rozdíly – diference, intervaly, tedy vzdálenost mezi dvěma časy, případně daty a časy se vyjadřují opět numericky v desetinném počtu dní, nebo textově pomocí počtu hodin, minut, sekund, případně tisícin sekundy. Například v neděli 7.6.2054 uplyne 750 000 dní od začátku letopočtu:

```
>datetimedifn("7.6.2054 0:0:0","1.1.0001 0:00:00")
750000
```

Vygenerování 5 milionů náhodných čísel a výpočet součtu jejich logaritmů trvá 1.9 sekund:

```
>t1=strdatetime(0)
>x%=normalr(5000000)
>s=sum(ln(x%^2))
>t2=strdatetime(0)
>td=datetimedifn(t2,t1)*86400
>"Comp time = "+round(td,3)+" seconds"
"Comp time = 1.935 seconds"
```

Další funkce a příklady viz jednotlivé datové a časové funkce v kapitole 7.2.

4.3. Datové struktury

4.3.1. Skalár

Skalár je jediná numerická hodnota, nebo textový řetězec. Místo termínu skalár se v tomto manuálu používá podle kontextu i termín číslo, hodnota, nebo textový řetězec, neboť se v podstatě jedná o jednoprvkový vektor, či jednoprvkovou matici a většina funkcí mezi skalárem, vektorem, případně maticí nečiní rozdíl. Přiřadí-li se `A=5`, je možné tuto hodnotu chápat formálně jako skalár, vektor i matici, jak uvádí příklad s pomocí žádného, jednoho nebo dvou indexů.

Příklad

```
>55+66
121
>A=5
>a
5
>a[1]
5
>a[1,1]
5

>b="QWERTY"
>b
"QWERTY"
```

4.3.2. Vektor

Vektor je sloupcová matice (s pouze jedním sloupcem), přístup k jednotlivým prvkům je možný pomocí jednoho indexu v indexových hranatých závorkách [ ], případně [[ ]]. Délka vektoru je počet jeho prvků a lze zjistit funkcí `COUNT`. Další příklady práce s vektorem viz např. 6.2.11 nebo popis funkce `vec`.

Příklad

```
>a=vec(1,3,5,7)
>a
1
3
5
7
>count(a)
4
>a[3]
5
>a[3,1]
5
```

```
>b=vec("A","E","I","O","U")
>b
"A"
"E"
"I"
"O"
"U"
b+" = "+(ascii(b)-64)
"A = 1"
"E = 5"
"I = 9"
"O = 15"
"U = 21"
```

4.3.3. Matice

Další datovou strukturou je obdélníková, nebo čtvercová matice *n* x *m*. Přístup k jednotlivým prvkům je možný pomocí dvou indexů v indexových hranatých závorkách [ ], případně [[ ]]. První index je vždy řádkový, druhý sloupcový. Schematicky je matice znázorněná na následujícím obrázku s naznačenými hodnotami indexů.

11	12	.	1m
21	22	.	2m
31	32	.	.
41	42	.	.
.	.	ij	.
n1	n2	.	nm

Příklad

```
//matice náhodných písmen:
> A=matrix(sample("A":"Z",25,repl=1),ncols=5)
>A
"M"   "B"   "P"   "V"   "T"
"J"   "O"   "D"   "D"   "O"
"T"   "P"   "W"   "V"   "P"
"X"   "J"   "M"   "V"   "W"
"R"   "Y"   "G"   "P"   "U"
>A[3,3]
"W"
```

4.3.4. Seznam

Seznam (angl. „*list*“) je jedna nebo více datových struktur skalár, vektor, matice, nebo seznam, přičemž každá z těchto struktur je prvkem seznamu. Prvek seznamu má jméno a pořadové číslo (pořadí). Jméno jednotlivých prvků jsou buď implicitní, nebo explicitně definovaná v příkazu `LIST`. Přístup k jednotlivým prvkům seznamu je možný pomocí znaku „`$`“ (dolar) umístěného mezi jméno seznamu a jméno prvku seznamu. Obsah proměnné typu seznam nelze zobrazit, nebo vytisknout jako celek.

Příklad

```
// Následující příkaz vytvoří list s prvky X, Y a Z
// a uloží jej do proměnné A:
>A=list(X=matrix(1:9,ncols=3),Y=ln(2),Z=list(T="ABC",U=11:13))
// K prvku X seznamu A se dostaneme pomocí selektoru $: A$X
>A$X
1     4     7
2     5     8
3     6     9

//Prvek seznamu A$X je matice, lze tedy použít indexy:
>A$X[2,2]
5
// Prvek seznamu A$Z je opět seznam s prvky T a U, které
```

```
// jsou přístupné opět selektorem $:
>A$Z$T
"ABC"
>A$Z$U[2]
12
```

Další příklady zápisu s různými typy proměnných:

```
a1=5.67
b1=seq(1,2,count=20)
c1="Computational Statistics"
d1=list(A=1,B=1:10,C="Sample List")
d1$b[3]
b=sample(0:2,20,repl=1)
B[[ not(zero(B)) ]]
```

4.3.5. Výraz

Výrazem nazýváme část skriptu, jehož výsledkem je datová struktura. Výsledem výrazu lze přiřadit (uložit) do proměnné.

Příklady výrazů a jejich výsledků v panelu Echo

```
>128
128
>33*44-55
1397
>matrix(11:16,ncols=2)
11    14
12    15
13    16
>round(transp(normalr(10)),2)
0.34  0.25  -1.61 -0.23 0.4   0.76  0.45  -0.15 0.11  -0.13
>"A"+(1:3)
"A1"
"A2"
"A3"
>A[2,3]
36
```

Ve výrazech lze neomezeně používat kulaté závorky obvyklým způsobem k určení priorit operací. Povinně se kulaté závorky používají k vymezení argumentů funkce. Základní priorita operací (operátorů) je následující:

standardní i uživatelské funkce >
„^“ (umocňování) >
„–“ (unární mínus) >
„*“, „/“, „#“ (násobení, dělení a maticové násobení) >
„+“, „–“ (sčítání, odčítání) >
„:“ (dvojtečka, operátor sekvence) >
„=“ (přiřazení),

takže například

– 2 ^ 2 jsou – 4, ale (–2) ^ 2 jsou 4. Nebo 1 : 3 + 1 je 1 2 3 4, ale (1 : 3) + 1 je 2 3 4.

Výrazy se v jazyce DARWin vyhodnocují podle priorit, a pak zprava doleva, což nemá na výsledek vliv s možnou výjimkou kombinace maticového a skalárního násobení matic, viz odst. 6.2.6, str. 27, případně sčítání numerických a textových hodnot.

4.3.6. Příkaz

Příkazem nazýváme část skriptu, jehož výsledkem je nějaká akce, činnost, například přiřazení hodnoty do proměnné, vytvoření grafu, tisk, export do souboru, příkaz a tělo cyklu FOR nebo WHILE, smazání proměnné, nastavení trasování. Příkaz musí být zapsán v jediném řádku, nebo pomocí operátoru @ rozdělen na více řádků a ukončen středníkem, viz odstavec 6.2.10, str. 29.

Příklady příkazů

```
>print(ln(10),\t,log(exp(1)))
>export(normalr(100),"random_data.txt")
>A=5
>delete(B)
>plot(normalr(1000))
>x=x+1
>for(i=1,10){print(rnd(1),\n)}
```

4.3.7. Skript

Skript je zápis posloupnosti příkazů, případně výrazů, které je možné spustit. Skript se spouští ve skriptovém listu v panelu Skript buď jako celek, nebo pouze označená část, dále viz odstavec 3.1.

5. Komunikace s okolím, vstupy a výstupy

Vstupní a výstupní operace poskytují široké možnosti použití programů v DARWinu k automatické generaci reportů, tabulek a diagnostik a integraci do datového toku kteréhokoliv procesu nebo organizace. Všechny funkce zmíněné v této kapitole jsou podrobně popsány v části 7.2 Příkazy a funkce. Přesto je zde shrnujeme jako orientační přehled funkcí, které jsou k dispozici pro komunikaci systému DARWin s okolím.

5.1. Komunikace v rámci QCExpertu

5.1.1. Tabulka proměnných

Tabulku proměnných lze použít k jednorázovému ručnímu zadání hodnot, vektorů, řetězců, buď z klávesnice, nebo ze schránky (Ctrl-C) z jiné aplikace. Rovněž je možno kopírovat obsah proměnné do schránky a použít jej v jiné aplikaci.

5.1.2. Výstup do protokolu

Výsledky výpočtů je možné tisknout pomocí příkazu PRINT do okna Protokol systému QCExpert. Tento výstup umožňuje mnohem flexibilnější tvar výstupů, než obsah proměnné. List protokolu je ovšem omezen 65535 řádky a 255 sloupci. Rozsáhlejší výstupy lze tisknout příkazem PRINT ve stejném formátu do textového souboru, kde je velikost neomezená. Takto vytvořené tabulky je pak možné exportovat a použít v jiných aplikacích.

5.1.3. Výstup do datového listu

Obsah proměnné (typicky vektor, nebo matice) lze zkopírovat do datového listu v okně DATA systému QCExpert, kde jej lze použít jako k další interaktivní analýze pomocí funkcí statistického systému QCExpert.

5.1.4. Grafické výstupy

Příkaz PLOT, GRAPHSHEET, a řada dalších funkcí, které jsou uvedeny u tohoto příkazu slouží ke tvorbě grafů v okně GRAFY systému QCExpert. Grafy lze interaktivně uložit do souboru, tisknout, nebo přenášet pomocí schránky do jiných aplikací. K automatickému ukládání grafů vytvořených v okně GRAFY do souboru slouží příkaz EXPORTGRAPH, případně PDFPLOT. Grafy vytvoření v okně GRAFY lze otevřít jednotlivě dvojitým kliknutím. Graf se otevře v interaktivním okně, kde jej lze zvětšovat a modifikovat. Podrobný popis interaktivního grafu je uveden v uživatelském manuálu QCExpert®.

5.1.5. Interaktivní dialogová okna

Funkce DIALOG poskytuje mnohostranný nástroj pro vytvoření interaktivního dialogového okna, v němž lze umístit prvky pro uživatelsky snadné zadání, nebo výběr vstupních parametrů.

5.1.6. Příkaz Message

Zobrazení okna se zprávou slouží například jako orientační zobrazení mezivýsledků, průběhu výpočtu, pro odlaďování programu, apod. Do okna message lze zapsat libovolný víceřádkový text.

5.2. Komunikace pomocí souborů

5.2.1. Vstup z textového souboru

Příkazem IMPORT lze načíst obsah textového souboru do proměnné jazyka DARWin. Může se jednat o textový řetězec, i o rozsáhlou matici dat se sloupci oddělenými libovolným oddělovačem. Tím je možné načítat data z externích aplikací, které mohou uložit data v tomto formátu.

5.2.2. Výstup do textového souboru

Výše zmíněný příkaz PRINT (odst. 5.1.2) lze využít k zápisu textu a tabulek do textového souboru. Sloupce tabulek lze oddělit tabulátory, středníky, apod. Tento soubor lze pak otevřít v jiných aplikacích (textových editorech, tabulkových procesorech, databázových systémech, atd.).

5.2.3. Výstup do grafického souboru

Grafy sestrojené v grafickém listu (odst. 5.1.4) lze příkazem EXPORTGRAPH exportovat do grafických souborů ve formátu BMP, JPG, GIF, WMF. Grafy lze před zápisem do souboru přeformátovat do libovolného rozlišení a tím docílit požadované kvality grafů.

5.3. Komunikace s databází QCEDataCenter

Inteligentní analytická databáze QCEDataCenter je výkonný nástroj pro on-line integraci dat z různých zdrojů a následnou diagnostiku a analýzu a distribuci informací. Tuto databázi lze připojit do systému QCExpert a pomocí prostředků DARWin provádět libovolné analýzy a výsledky ukládat zpět do databáze, publikovat v jiných formátech, či vytvářet

finální tisknutelné reporty ve formátu PDF. Příkazy a funkce týkající se databáze QCEDataCenter začínají písmeny DB….

5.4. Výstup PDF

Velmi silným nástrojem jazyka DARWin jsou příkazy pro publikování PDF-souborů, které umožňují reprezentativní prezentaci výsledků v jednoduchých nebo i velmi komplexních a rozsáhlých zpráv. Příkazy začínající PDF… umožňují dynamické stavění stránek A4 obsahujících texty, tabulky, grafy a bitové obrázky (např. logo firmy) s možností formátování, volby fontu, velikosti, barvy, záhlaví a zápatí, okrajů, atd. Příkazy pro tvorbu PDF mohou být automatizovaným finálním reprezentativním a snadno realizovatelným výstupem Vaší práce.

5.5. Odeslání e-mailu

DARWin umožňuje zaslání emailu pomocí protokolu SMTP. Má-li uživatel přístup na SMTP bránu, je možné příkazem **`sendmail`** zasílat na jednu nebo více adres textovou zprávu a soubory jako přílohu. Lze tím například zajistit automatické on-line zasílání informací o sledovaných procesech a poruchách, periodicky generovaných zpráv PDF a podobně. Při používání příkazu je ovšem dbát opatrnosti, aby při nějaké chybě v algoritmu nedošlo k přeplnění schránek, zahlcení serveru, apod. Je snadné díky nepozornosti zaslat omylem během pár minut tisíce mailů! Příkaz **`sendmail`** *NENÍ* určen k hromadnému zasílání e-mailů!

5.6. Dávkové spouštění skriptů

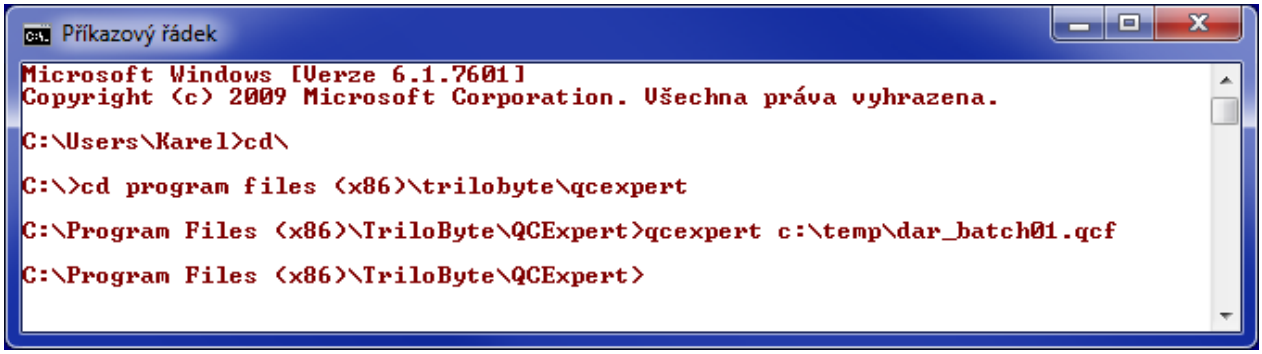
Dávkové spouštění skriptů umožňuje spuštění skriptů z příkazového řádku, z jiných aplikací či aplikačního prostředí. Je možné naplánovat automatické opakované spouštění skriptů v zadaném čase, nebo při určitých událostech pomocí Plánovače úloh, který je součástí Windows®. Spuštění skriptu se provede spuštěním aplikace QCEXPERT.EXE s názvem skriptu včetně cesty jako parametrem. Název skriptu musí být uveden včetně přípony .QCF. Skript je prováděn na pozadí, bez otevření okna aplikace QCEXPERT. Provedou se postupně všechny listy se skripty kromě listů s funkcemi (viz odst. 6.3). Pokud je nastaven zápis do log-souboru (viz odst. 3.5), vytvoří se log-soubor s informací o spuštění a o případných chybách, ke kterým mohlo dojít během provádění skriptu. Chybová hlášení se rovněž ukládají do chybového souboru ScriptErr.log v pracovním adresáři (implicitně c:\temp), například:

```
28.11.2011 17:44:46 c:\temp\dar_batch01.qcf  Floating point division by zero :: A=5/0
```

Skripty pro dávkové zpracování mohou obsahovat interaktivní prvky, jako dialogová okna a okna se zprávou, Tato okna se zobrazí na displeji a mohou sloužit k případné interakci s uživatelem. Je-li například skriptový soubor uložen pod názvem dar_batch01.qcf v adresáři C:\TEMP, bude volání dávkového zpracování vypadat asi takto:

```
qcexpert c:\temp\dar_batch01.qcf
```

Následující obrázek ilustruje volání dávky z příkazového řádku.



6. Definice a syntaxe jazyka

6.1. Poznámky k syntaxi a zápisu

V následujících odstavcích uvádíme úplný popis funkcí jazyka DARWin. Kód a výpisy z panelu Echo jsou pro lepší přehlednost uváděny fontem `Courier New`. Pokud se ve výpisech vyskytují spouštěné výrazy a příkazy zároveň s výsledky a výpisy proměnných, jsou u spouštěného kódu uváděny na začátku řádků i prompty („>“) pro lepší rozlišitelnost textu psaného uživatelem a výstupu (odpovědi) DARWinu. Pokud chce uživatel kód použít, je nutné použít jen řádky začínající promptem > a po zkopírování do skriptu tyto prompty před spuštěním skriptu smazat.

6.2. Operátory a zvláštní znaky

6.2.1. Komentáře //, /*

Komentář je část skriptu, která se při spuštění ignoruje. Využívá se především k popisu skriptu, vysvětlivkám. Je vhodné doprovázet skripty komentářem pro větší přehlednost. Po čase si obvykle ani autor skriptu nepamatuje, co chtěl „říci“ a nekomentovaný program je často nutné napsat celý znova. Je možné používat dva typy komentáře. Řádkový komentář začíná dvěma lomítky „//“ a končí koncem řádku (nemá koncový znak). Za tímto kódem je jakýkoliv text do konce řádku ignorován. Blokový komentář začíná kombinací lomítko a hvězdička (bez mezery) /* (začátek komentáře) a končí opačnou kombinací hvězdička a lomítko */ (konec komentáře). Těmito znaky lze ohraničit libovolné množství textu ve skriptu. Může být využit i pro vyřazení části kódu v programu. Pozor na nechtěné kombinace komentářů, které se mohou vzájemně rušit, například „//“ a „/*“ na témže řádku.

Příklad

```
//---------- Začátek programu ------------
R=0 // Spodní mez
S=10 // Horní mez
/*  // Neaktivní "zablokované" alternativní zadání R a S:
R=10
S=30
*/
x=R:S // naplnění vektoru x
// ---------- Konec programu -----------
```

6.2.2. Příkazové složené závorky { }

Příkazové (složené) závorky { } slouží pro ohraničení posloupnosti příkazů ve struktuře příkazů FOR, IF, WHILE a v definici funkcí Function. Příkazové závorky je však možné použít i mimo tyto případy.

Příklad

```
for (i=1:10){print(i,\t,log(i),\n)}

x=normalr(1);if (LT(x, 0))
{
  x= -x
}
a=sqrt(x)
```

6.2.3. Přiřazení =

Přiřazovací znak rovnítko = přiřadí výraz na pravé straně do proměnné na levé straně od "=". Tento znak se nepoužívá jako relační operátor. Tam je nutno použít funkci EQ.

Příklad

```
>A=4+5
>a
9
```

6.2.4. Aritmetické binární operátory + - * /^

+, -, *, /, ^

Sčítání, násobení a umocňování s obvyklou prioritou operací. Operátor „+“ slouží navíc pro spojování řetězců. Je-li alespoň jeden z operandů řetězec, je výsledkem také řetězec. Je-li jeden z operandů matice nebo vektor a druhý (na pořadí nezáleží) skalár, provede se operace se skalárem pro všechny prvky matice nebo vektoru. Pokud jsou oba operandy matice nebo vektory, musí mít buď *a)* stejný rozměr a operace se provede po jednotlivých prvcích, nebo *b)* jeden operand je matice typu (NxM) a druhý operand je sloupcový vektor (Nx1), nebo řádkový vektor (1xM). V případě *b)* se operace provede po jednotlivých sloupcích, resp. řádcích matice. Toho lze využít například při centrování a normování sloupců a podobně.

Příklad

```
>"ABC"+"-4-"+8+123+1
"ABC-4-132"

>"ABC"+"-4-"+8+123+"1"
"ABC-4-81231"

>"A-"+(1:5)
"A-1"
"A-2"
"A-3"
"A-4"
"A-5"
```

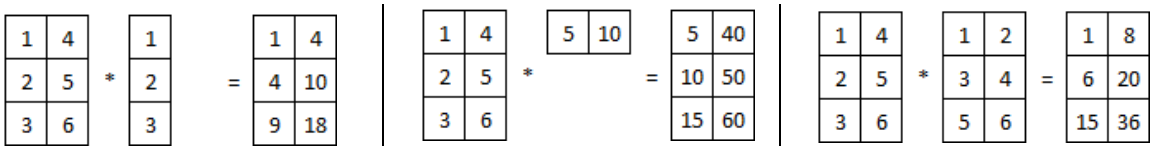
```
>1+2*3^4
163

>vec("A","B","C")+(1:3)
"A1"
"B2"
"C3"

>(1:4)+vec(10,20,30,40)
11
22
33
44
```

```
>(1:4)+10
11                          >matrix(1:9,ncols=3)*vec(10,100,1000)
12                          10    40    70
13                          200   500   800
14                          3000  6000  9000
```

6.2.5. Skalární aritmetické operace s maticemi a vektory

Maticové násobení pomocí operátoru „#“ je popsáno níže v odstavci 6.2.6. Dále lze matice násobit skalární hodnotou *k*; tím se získá *k*-násobek matice. Je-li například `A` matice s prvky $a_{i,j}$, pak výsledkem příkazu `B=3*A` bude matice `B` s prvky $b_{i,j} = 3\ a_{i,j}$. Budou-li `A` a `B` matice stejného rozměru *N*x*M*, lze použít standardních binárních operátorů +, -, *, /, ^ pro operaci mezi prvky $a_{i,j}$ a $b_{i,j}$, takže matice `C = A/B` bude mít také rozměr *N*x*M* a prvky $c_{i,j} = a_{i,j} / b_{i,j}$.

S výhodou lze využít skalárního násobení matice a vektoru. Bude-li `A` matice rozměru *N*x*M* a `R` bude sloupcový vektor *N*x1, bude výsledkem příkazu `C=A*R` matice `C` rozměru *N*x*M* s prvky $c_{i,j} = c_{i,j} \,.\, r_i$. Podobně, bude-li `A` matice rozměru *N*x*M* a `R` bude řádkový vektor 1x*M*, bude výsledkem příkazu `C=A*R` matice `C` rozměru *N*x*M* s prvky $c_{i,j} = c_{i,j} \,.\, r_j$. Stejně to platí pro další binární aritmetické operátory a rovněž pro binární logické funkce `GT`, `GE`, `LT`, `LE`, `NE`, `EQ`. Tři příklady popsaných operací jsou na následující ilustraci.



Podobně lze na matice a vektory aplikovat i skalární funkce jako třeba `sqrt`, `exp`, `sin`, atd. Uvedených skalárních operací lze rovněž využít ve spojitosti s funkcí APPLY třeba k centrování a normování vektorů v matici.

Příklad

```
>x=matrix(2*normalr(15)+3,ncols=3) //matice 5x3
>xmean=apply(x,"average",dir=2)
>xmean // Střední hodnoty sloupců X (řádkový vektor 1x3)
3.14195057706205  4.50713304355102  3.67232750529602
>round(x-xmean,6)
// Centrované sloupce X, "odečtení" vektoru od matice
1.959322    -1.043658   -2.590902
3.74365     -0.7692     -0.913976
-0.087922   0.821855    -2.212984
-2.09504    2.198043    1.817004
-3.520009   -1.20704    3.900858
```

6.2.6. Maticové násobení

Znak „#“ (dvojitý kříž) slouží pro maticové násobení dvou matic nebo vektorů. Počet sloupců první matice musí být stejný jako počet řádků druhé matice. Použije-li se standardní znak pro násobení „*“ (hvězdička), vynásobí se jen jednotlivé prvky matice nebo vektoru, viz předchozí odstavec. Pozor, maticové násobení „#“ má stejnou prioritu jako standardní násobení „*“, avšak vzhledem k tomu, že maticové násobení není komutativní a výrazy se vyhodnocují zprava doleva, velmi se doporučuje ve výrazech obsahující obě operace

s maticemi používat kulaté závorky, neboť například A*A#A není totéž jako A#A*A, je-li A matice.

Příklad

```
>X=bind(rep(1,10),1:10)
>XX=transp(X)#X
>XX
10    55
55    385
```

6.2.7. Selektor prvku seznamu $

Znak „$“ (dolar) umístěný bez mezery mezi jméno seznamu a jméno jeho prvku vybere příslušný prvek seznamu.

Příklad

```
>s=list(a=5,b="ABCDEF",c=6:10)
>s$b
"ABCDEF"
>s$c[2]
7
```

6.2.8. Operátor sekvence, dvojtečka :

Operátor sekvence dvojtečka „:“ je binární operátor. Vytvoří vektor s celočíselnou sekvencí s krokem 1 s použitím prvního a druhého operandu jako začátku a konce sekvence. Operátor sekvence má nižší prioritu, než sčítání. Je-li první operand větší, než druhý, je sekvence klesající. Funkce transpozice `transp` je zde použita jen pro úsporu místa.

Příklad

```
>transp(1:5)
1     2     3     4     5
>transp(3:-2)
3     2     1     0     -1    -2
>transp((1:10)^2)
1     4     9     16    25    36    49    64    81    100
```

Podobně lze použít dvojtečkový operátor i pro textové znaky. Vytvoří se sekvence podle kódu ASCII, v řetězcích se ignorují všechny znaky kromě prvního.

Příklad

```
>transp("ZA":"abc") // (totéž jako "Z":"a")
"Z"   "["   "\"   "]"   "^"   "_"   "`"   "a"
```

6.2.9. BigData suffix %

Znak „%“ (procento) jako přípona za názvem proměnné bez mezery identifikuje „velkou“ tabulku dat. Končí-li proměnná znakem %, neukládá se ve formátu tabulky. Hodnoty této proměnné se ukládají jako textový soubor do pracovního adresáře a nemají omezení velikosti (velikost je omezena pouze dostupnou pamětí). Práce s těmito proměnnými „BigData“ je stejná jako s obyčejnými proměnnými.

Příklad

```
>x=1:100000 // Data se nevejdou do spreadsheetu
Error : "Chybný odkaz na buňku"

>x%=1:100000 // Do "BigData"-matice se vejdou miliony řádků.
>x%[99990:100000]
99990
99991
99992
99993
99994
99995
99996
99997
99998
99999
100000
```

6.2.10. Víceřádkový příkaz @ … ;

Znak @ na začátku řádku nebo před příkazem uvozuje začátek víceřádkového příkazu. Takový příkaz může být zapsán na více řádků a musí být ukončen středníkem. Víceřádkové příkazy jsou výhodné především v případě dlouhých příkazů PLOT, PRINT, NNLEARN, DIALOG, a podobně, jejichž zápis do jediného řádku by byl nepřehledný. Chceme-li zapsat tímto způsobem více příkazů, musí být každý z nich uvozen znakem @ a ukončen středníkem.

Příklad

```
//Standardní příkaz
v jednom řádku
>x=1+2+3
>x
6
```

```
//Příkazy rozdělené do více řádků
@x=1
+2+
3;
@ D2=vec(0,1.128,1.693,2.059,
2.326,2.534,2.704,2.847,2.970,
3.078,3.173,3.258,3.336,3.407,
3.472,3.532,3.588,3.640,3.689,3.735);
```

6.2.11. Indexové závorky [ ]

Indexové hranaté závorky [ ] slouží k adresování prvků vektorů a matic. První index je řádkový (číslo řádku), druhý index je sloupcový (číslo sloupce). Vektor je přitom jen zvláštní typ matice s jedním sloupcem a *n* řádky, u něhož je povoleno používat jen jeden index, který pak odkazuje na daný řádek (prvek) vektoru. Syntaxe jazyka dovoluje různé způsoby použití indexových závorek, které je popsáno v následujících 6 odstavcích.

1. Odkaz na řádek a sloupec matice nebo vektoru.

Příklad

```
x[3,5] // prvek matice x ve 3. řádku a 5. sloupci
```

2. Intervalové indexování, indexování výčtem a prázdným indexem

Je-li indexem matice nebo vektoru vektor, vrací se všechny sloupce, resp. řádky, které odpovídají hodnotě vektoru indexů. Není-li uveden index, vrací se všechny sloupce, resp. řádky.

Příklad

`a[10:20,2:6]` submatice tvořená druhým až šestým sloupcem a desátým až dvacátým řádkem matice a.
`a[,vec(1,3,5)]` // submatice tvořená 1., 3. a 5. sloupcem a všemi řádky matice a.

`b[vec(1,3,5)]` // vektor tvořený 1., 3. a 5. prvkem vektoru b.

3. Logická indexace [[ ]]

Index ve dvojitých hranatých závorkách musí být vektor nul a jedniček stejné délky nebo rozměru jako je příslušná dimenze matice, případně délka vektoru. Vrací se řádky, sloupce, případně prvky, kterým odpovídá jednička v logickém indexu. Jako index lze s výhodou použít logická funkce na zdrojovém vektoru (matici). Matice či vektor logických indexů zde tedy funguje jako mřížka (overlay) *K*(*n*x*m*), kterou položíme na matici *A*(*n*x*m*) a vybereme ty prvky *A*[*i,j*], kde jsou příslušné indexy *K*[*i,j*] jedničky: `A[[K]]`.

Příklad

```
>x=1:10
>ii=vec(1,0,0,1,1,1,0,1,0,1)
>x[[ii]]
1
4
5
6
8
10
```

```
>x=normalr(10)
y=1:10
>x[[lt(x,0)]]
-1.95551799804119
-0.507056775394412
-2.07615542267181
>y[[lt(x,0)]]
1
5
9
```

Další možností logické indexace je výběr sloupců, případně řádků matice. Zde je logickým indexem vektor *K* hodnot 0 a 1 a čárka, která určuje zda jde o sloupcový nebo řádkový index. Tak lze výhodně vybrat datové řádky splňující nějakou podmínku.

Příklad

```
>a=matrix(vec(1,2,3,10,20,30,100,200,300),ncols=3)
>a[[,vec(1,0,1)]]
1     100
2     200
3     300
>a[[vec(1,0,1),]]
1     10    100
3     30    300
```

4. Definice a naplnění vektoru nebo matice konstantou.

Proměnná musí být nedefinovaná, čehož lze dosáhnout například příkazem `delete`. Pak se přiřadí požadovaná konstanta do pravé spodní rohové buňky matice, nebo vektoru

(tedy buňky s nejvyšším řádkovým a sloupcovým indexem). Touto konstantou se pak naplní celá matice.

Příklad

```
>delete(a)
>a[3,4]=1
>a
1     1     1     1
1     1     1     1
1     1     1     1
```

5. Zvětšení dimenze matice

Přiřazení hodnoty do neexistujícího prvku matice nebo vektoru zvětší tuto matici nebo vektor na příslušnou velikost, ostatní hodnoty nad původní dimenzi matice se doplní nulou.

Příklad

```
>a=bind(vec(1,2),vec(4,7))
>a
1     4
2     7
```

```
>a[3,5]=99
>a
1     4     0     0     0
2     7     0     0     0
0     0     0     0     99
```

6. Záporné indexy, vypouštění řádků a sloupců

Záporná hodnota indexu způsobí vypuštění příslušného řádku, či sloupce z matice. Tímto silným nástrojem je umožněna snažší manipulace s vektory a maticemi, vytváření submatic, odstraňování části dat a podobně. Není možno kombinovat kladné a záporné indexy u jediného směru (je možné zapsat: A[vec(-2, -4), 1], ale nikoli A[vec(1, 2, -2, -4), 1]).

Příklad

```
// Rohové prvky matice A(4x4)
a=matrix(1:16,ncols=4)
a[-(2:3),-(2:3)]

// Vynechání prvků
1,3,5,6,11,16 z vektoru x
>x=normalr(20)
>x1=x[-vec(1,3,5,6,11,16)]
>count(x1)
14
// Vynechání každého pátého
řádku
>x=normalr(25)
>xi=1:25
>i=-5*(1:5)
>bind(xi[i],x[i])
1     0.0710872138379552
2     -0.180564729065626
3     1.58436236861215
4     -0.106992352401548
6     0.937575477540491
16    0.627633323249614
17    -1.56902175656619
18    1.58550544608754
19    0.770823204730091
21    -0.795454358251783
22    -1.56151447065492
23    0.98218317810612
24    -1.02783119371402

// Výběr jedinečných hodnot
z vektoru x
// (odpovídá SQL select
distinct):
>x=sample(1:10,50,repl=1)
>x1=sort(1,x)
>x1[[ne(vec(x1[-1],999),x1)]]
1
2
3
4
5
```

```
7     -1.02310933239758
8     0.436291733151174
9     -0.234901786098745
11    2.55209876658992
12    -0.905511308448221
13    -1.04532565890302
14    0.659992912539357
```

```
6
7
8
9
10
```

7. Indexování výrazu

Je-li výsledkem výrazu vektor nebo matice, lze tento výraz indexovat pouze tehdy, je-li celý výraz uzavřen v kulatých závorkách. Tento způsob lze použít i s logickou indexací [[ ]].

Příklad

```
>(10:19)[2:3]
11
12

>(ln(1:10))[8:10]
2.07944154167984
2.19722457733622
2.30258509299405
```

```
>(1:10)[[lt(random(1:10),0.75)]]
1
3
4
6
8
9
>(transp(a)#a)[1,1]
```

8. Přiřazení do indexované proměnné (index „na levé straně“)

Do jednotlivých prvků indexované proměnné (vektoru nebo matice) lze přiřazovat hodnoty pomocí indexových nebo logických závorek u proměnné vlevo od přiřazovacího rovnítka. Lze použít skalární indexy, např. A[3,3], výčtové indexy, např. A[vec(1,3,5,9)] i intervalové indexy např. A[2:4,1:5], případně logický vektor nebo matici, např. A[[lt(A,0)]]. Přiřazovaná hodnota vpravo od rovnítka musí být buď skalární, nebo vektor s odpovídajícím počtem prvků, Je-li na pravé straně skalární (jediná) hodnota, dosadí se tato hodnota do všech specifikovaných prvků, je-li na pravé straně vektor, nebo matice, musí mít stejné rozměry jako specifikované prvky vlevo od rovnítka.

Příklad

```
>a=1:3
>a[2]=99
>a
1
99
3

>a=matrix(1:12,ncols=4)
>a
1    4    7    10
2    5    8    11
3    6    9    12
>a[1,]=99
>a
99   99   99   99
2    5    8    11
3    6    9    12
```

```
//Nahrazení záporných čísel nulami
>a=round(matrix(normalr(12),ncols=4),2)
>a
-0.16     1.37      -0.18     -0.74
0.6       -0.33     0.91      -1.48
1.12      0.54      2.19      0.5
>ii=lt(a,0)
>ii
1    0    1    1
0    1    0    1
0    0    0    0
>a[[ii]]=0
>a
0    1.37 0    0
0.6  0    0.91 0
1.12 0.54 2.19 0.5
```

6.2.12. Oddělovač příkazů ;

Oddělovač příkazů „;“ (středník) odděluje více příkazů na jednom řádku. Povinné je použití středníku na konci víceřádkového příkazu, viz odstavec 6.2.10, str. 29. Na konci posledního příkazu na řádku středník být nemusí.

Příklad

```
a=0; b=2; for(i=1,10) {a=a+b^(-i)}; print(a)
0.9990234375
```

6.2.13. Řídicí kódy pro formátování tisku \n, \t

V příkazech `PRINT`, `PDFTEXT`, `MESSAGE` je pro znak tabulátor používá kód `\t` a pro znak nový řádek kód `\n`. Tyto kódy nelze uložit do proměnné, nejedná se o textové řetězce. V příkaze `PRINT` při tisku do tabulky *Protokol* je nutné pro přechod do sousední buňky použít kód `\t`.

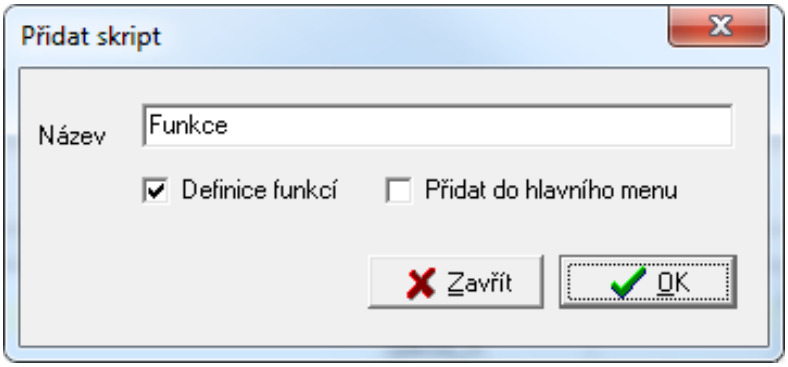
Příklad

```
x=normalr(100)
print("Min",\t,"Max",\t, "Průměr",\t, "Sm. Odch.",\n)
print(min(x),\t,max(x),\t,average(x),\t,sqrt(var(x)),\n,\n,\n)
print(\t, \t,"Datum:",\t,strdate)
```

Min	Max	Průměr	Sm. Odch.
-1.869971	2.171144	-0.041539	0.86654
		Datum:	4.9.2012

6.3. Uživatelské funkce

Uživatelská funkce rozšiřuje možnosti jazyka o neomezený počet nových funkcí definovaných uživatelem a použitelných stejně jako standardní funkce jazyka. Pro definici uživatelských funkcí slouží zvláštní funkční skriptový list, který se získá zaškrtnutím políčka Definice funkcí při vytváření, nebo přejmenování skriptového listu.



V následující tabulce jsou komentovány definice a použití uživatelské funkce na jednoduchém příkladu.

Script_1	Function	Panel Echo po spuštění skriptu

/*
Hlavní program (skript)
Volá se funkce „myfun“ s aktuálními argumenty a,b, které zde mají hodnotu 5 a 3. Výsledek je uložen do proměnné c a zobrazen v panelu Echo.
*/
 	
a=5
b=3
c=myfun(a,b)
c
 
	/*
	Definice funkce myfun
	Ve funkčním skriptovém listu je funkce definována s formálními argumenty x, y, výsledek je uložen do dočasné lokální proměnné z a předána jako výsledek pomocí příkazu return.
	Definice funkcí se nespouští.
	*/
	
	function myfun(x,y)
	{
	z = (x+y) * (x-y)
	return (z+1)
	}
		Po spuštění skriptu se zavolá funkce myfun a výsledek 17 je zobrazen v panelu Echo.
		
		>a=5
		>b=3
		>c=myfun(a,b)
		>c
		17

Uživatelské funkce lze volat v kterémkoliv skriptovém listu v rámci otevřeného souboru QCF. Funkci lze volat i v rámci jiné funkce v témže nebo jiném funkčním listu. Nezáleží přitom na pořadí definice funkce. Ve funkci se může použít jiná funkce, která je definovaná níže, například:

```
function fun2(x)    // Výpočet normy vektoru x s pomocí fun1
{
return(sqrt(fun1(x)))
}
function fun1(x)
{
return(transp(x)#x)
}
```

Definice uživatelských funkcí se zapisují do zvláštního skriptového listu, kdy se v dialogu *Přidat skript* označí políčko *Definice funkcí*. V záložce tohoto listu je pak znak funkce *f(x)* graficky odlišující obyčejné skriptové listy od listů s definicemi funkcí. Těchto listů s funkcemi může být ve skriptovém panelu více, definice různých funkcí v různých listech jsou rovnocenné. Při volání funkce v obyčejném skriptovém listu se prohledají hlavičky definic funkcí ve všech funkčních skriptových listech, a pokud je volaná funkce nalezena, provede se s aktuálními argumenty.

Skript1	Záložka obyčejného skriptového listu
Funkce	Záložka skriptového listu s definicemi funkcí
Skript2	Záložka skriptového listu spustitelná z menu

Samotná definice funkce se skládá z klíčového slova „FUNCTION“, názvu uživatelské funkce a seznamu formálních argumentů oddělených čárkami v kulatých závorkách. To je hlavička uživatelské funkce, která musí být zapsána v jednom řádku. Následuje tělo funkce, tedy sekvence příkazů v příkazových závorkách. Uzavřením odpovídající příkazové závorky formální definice funkce končí. Tělo funkce musí obsahovat příkaz RETURN následovaný kulatými závorkami, které obsahují jediný objekt, který je

funkcí vrácen (výsledek funkce). Tímto objektem může být číslo, textový řetězec, vektor, matice, nebo seznam. Provede-li se při volání funkce příkaz return, provádění funkce končí a případné další příkazy funkce jsou ignorovány, je tedy příkaz return typicky (ale ne nutně vždy) posledním příkazem těla uživatelské funkce. Je-li použit příkaz RETURN s prázdnými závorkami, je výsledkem funkce vždy číselná hodnota 0. Má-li být výsledkem funkce více hodnot různého typu, nebo různé logické povahy (vektor X, číslo A1, řetězec STR5, matice Z, další vektor XX, apod.), je možno předat výsledek ve tvaru seznamu, například:

```
return(list(X=X, cislo=A1, nazev=STR1, MAT=Z, vektor2=XX))
```

Formální syntaxe uživatelské funkce	Příklad definice uživatelské funkce:	Příklad volání uživatelské funkce:

`FUNCTION fname([X1 [, X2,...] ] )`
`{`
`...`
`<<příkazy těla funkce>>`
`RETURN ( [ A ])`
`...`
`}`
	`function Fn1(x)`
	`{`
	`t=x*x`
	`return(t+1)`
	`}`
		`y=fn1(4) + fn1(5)`

Seznam formálních argumentů funkce jsou názvy proměnných, které budou funkci předány při jejím volání, oddělené čárkami. Tyto proměnné se pak použijí v těle funkce jako vstupní hodnoty funkce. Seznam argumentů může obsahovat jednak povinné argumenty, jednak nepovinné argumenty, jejichž implicitní hodnota je definována pomocí „=“ v seznamu argumentů. Není-li pak nepovinný argument uveden při volání funkce, použije se hodnota přiřazená v hlavičce. Chceme-li použít jinou hodnotu, je nutné argument uvést ve volání funkce. Tam musí být uveden stejný název argumentu, jako v hlavičce a jemu přiřazená hodnota. V seznamu formálních parametrů musí být vždy uvedeny nejdříve všechny povinné argumenty a teprve pak případné argumenty nepovinné s přiřazenými implicitními hodnotami. Implicitní hodnotou může být číslo, vektor, jiný objekt, nebo i libovolný výraz, jehož hodnota je známa v okamžiku volání funkce. Je tedy možno používat jako implicitní hodnotu nepovinného argumentu i funkce jiného formálního argumentu.

Příklady přípustných hlaviček funkcí s nepovinnými argumenty a jejich volání

Definice funkce (hlavička)	Možné volání funkce

`function F1(x,y,pocet=100)`	`r=F1(x,y)`
	`r = F1(x, x^2, pocet=200)`
	`r = F1(x, x^2, 200) // Chyba: nepovinný argument musí mít při volání uvedeno jméno`

`function F2(x,N=count(x),z=normalr(N))`	`F2(normalr(100))` `F2(2*x, z=x/10)`

`function F3(N=500, x,y)`
`// Chybné pořadí argumentů, implicitní musejí být na konci, správně:`
 	
`function F3(x,y,N=500)`
	`F3(4,7)`
	
	`F3(a,b,N=10)`

Do uživatelské funkce lze vložit nápovědu, která se pak zobrazí v okně uživatelské nápovědy. Doporučujeme nápovědu do hotových funkcí vkládat, aby bylo možné funkci použít i později, případně jiným uživatelem. Nápověda má tvar textového blokového komentáře ohraničeného kódy `/*` a `*/` vloženého mezi hlavičku a tělo funkce. Podrobnější popis tvorby, doporučenou strukturu nápovědy a příklad je uveden v odst. 6.6.2 na straně 39.

6.3.1. Instance jazyka

Při zavolání funkce se vytvoří nová skrytá instance (úroveň, frame) v níž se vytvářejí všechny objekty vytvářené volanou funkcí. Proměnné v těle volané funkce (lokální proměnné) jsou tedy platné jen po dobu provádění této funkce, po skončení funkce jsou nedostupné a také neovlivňují proměnné definované v základní instanci, které jsou v panelu Seznam proměnných a které jsou přístupné v základním skriptu. Používá-li se tedy například v těle funkce proměnná A, a v hlavním skriptu rovněž proměnná A, jedná se o různé proměnné, které se vzájemně neovlivňují, proměnná A ze základní instance není dostupná ve volané funkci a proměnná A v těle funkce není dostupná z hlavního skriptu. Jediný způsob předávání proměnných ze základní instance do funkce je tedy pomocí seznamu argumentů funkce. Jediný způsob předání proměnných z funkce do základní instance je prostřednictvím příkazu return. V jazyce neexistují „globální“ proměnné. Funkce však může používat I/O příkazy jako PRINT, PLOT, EXPORT, IMPORT a podobně, které jsou vždy globální.

6.3.2. Rekurzivní volání funkce

Funkce může ve svém těle volat kteroukoliv jinou definovanou uživatelskou funkci. Funkce může ve svém těle rovněž volat sebe sama (rekurzivní volání). Při každém rekurzivním volání funkce se vždy vytvoří nová instance a volaná funkce si vytváří v této instanci nové proměnné. Maximální hloubka rekurze (počet úrovní rekurzivního volání) je určena velikostí paměti a paměťovou náročností funkce. Následující příklad má jen ilustrativní význam, daleko rychleji lze samozřejmě vypočítat faktoriál třeba pomocí funkce `FACT(N)`, nebo `PROD(1:N)`.

Příklad

```
// Definice rekurzivní funkce pro výpočet faktoriálu N:
function fak(N)
{
if(zero(N)){return(1)}
if(not(zero(N))){return(N*fak(N-1))}
}

// Volání funkce:
>fak(5)
120
>fak(70)
1.19785716699699E100
```

6.3.3. Maskování a konflikty funkcí a proměnných

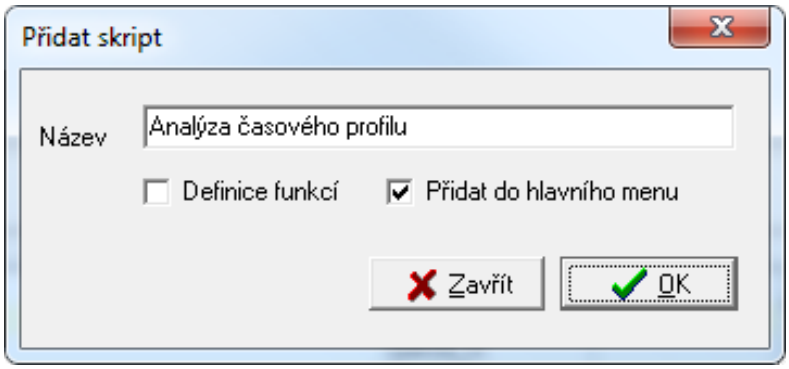
Jsou-li definovány dvě, nebo více funkcí stejného jména, bude se při volání provádět ta, která je definována na prvním místě. Jsou-li funkce se stejným jménem definovány ve dvou různých funkčních listech, bude se provádět ta, která je definována ve dříve založeném listu. Má-li uživatelská funkce stejný název, jako některá standardní funkce jazyka, má přednost uživatelská funkce.

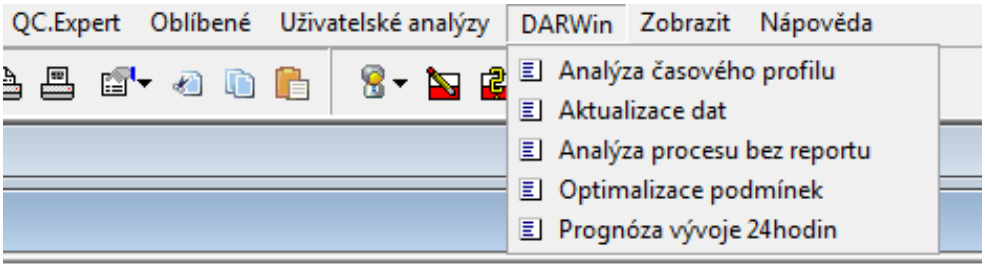
Proměnné se stejným jménem jako standardní, nebo uživatelská funkce jsou povoleny a nejsou formálně v konfliktu. Výjimkou jsou funkce a příkazy bez argumentu jako `Deletevars`, `Pi`, `Stop`, `Traceon`, `Traceoff`, jejichž jména nelze použít jako jména proměnné. Při použití proměnné a funkce se stejným jménem rozhoduje o výsledku syntaxe zápisu. Samozřejmě se z praktických důvodů nedoporučuje používání proměnných se jmény funkcí. Ilustrace je v následujícím příkladu.

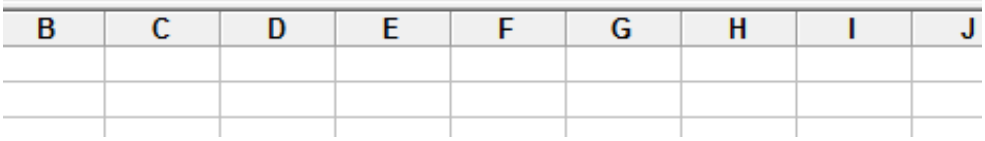
```
>sin=5
>sin(1)
0.841470984807896
>sin
5
>delete(sin)
>sin
Error : "Proměnná "SIN" není definovaná"
```

6.4. Spouštění skriptu z menu QCExpert

Odladěné skripty lze snadno spouštět z hlavního menu systému QCExpert. Nastaví-li se při definici, nebo přejmenování skriptu (blíže viz odst. 3.1, str. 9) volba *Přidat do hlavního menu*, vytvoří se v hlavním menu položka DARWin, v níž budou všechny takto označené skripty aktuálně otevřeného souboru skriptů QCF, viz následující obrázky.







Názvy položek menu DARWin jsou shodné s názvy skriptů. Volbou příslušné položky menu se spustí vždy celý příslušný skript. S použitím vstupních a výstupních operací (např. DBIMPORT, IMPORT, EXPORT, EXPORTGRAPH, PRINTPDF, atd.), čtení a zápis dat (např. GETSHEET, PUTSHEET), uživatelských dialogových oken DIALOG, MESSAGE,

uživatelských funkcí, statistické knihovny QCExpert, matematické knihovny a dalších nástrojů lze tak vytvářet novou funkcionalitu systému QCExpert pro řešení specifických úloh.

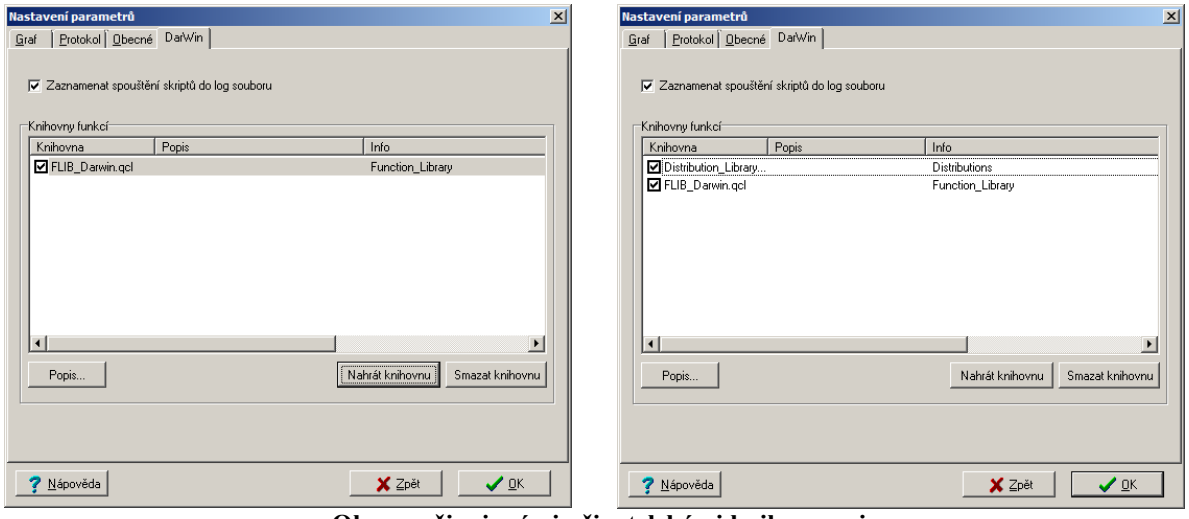
6.5. Knihovna funkcí DARWin

6.5.1. Vytvoření knihovny funkcí

Knihovna uživatelských funkcí umožňuje jednoduše rozšiřovat funkcionalitu DARWinu podle specifických potřeb uživatele a rovněž sdílet nové funkce jazyka mezi uživateli. Uživatelem vytvořené funkce se stávají součástí jazyka bez nutnosti mít tyto funkce vždy ve funkčním listu otevřeného skriptu. Při vytváření knihovny funkcí se postupuje jednoduše. Nejdříve se vytvoří a odladí ve funkčním listu (nebo více listech) potřebné funkce, viz odst. 6.3. Soubor se skripty funkcí se uloží ve formátu QCL (jako *Knihovna funkcí DARWin*). Listy, které nejsou označeny jako definice funkcí, jsou v tomto formátu ignorovány, ale nejsou smazány. Takto uložený soubor je připraven k připojení a aktivaci knihovny funkcí. Soubor QCL lze kdykoliv otevřít a editovat, přidávat další funkce, a opět uložit. Případně provedené změny se projeví ihned, knihovnu není třeba odpojit a znovu připojit.

6.5.2. Připojení a aktivace knihovny funkcí

Uložené funkční skripty lze nahrát (připojit) v okně nastavení parametrů (*Soubor – Nastavení* – Záložka *DARWin*). Tlačítkem *Nahrát knihovnu* otevřeme požadovanou knihovnu uloženou jako soubor QCL a aktivujeme zaškrtnutím políčka u jejího názvu. Po zavření okna *Nastavení* jsou k dispozici všechny funkce definované v aktivovaných knihovnách. Připojenou knihovnu lze odpojit tlačítkem *Smazat*. Tím je knihovna smazána pouze ze seznamu připojených knihoven, soubor QCL zůstává zachován a lze jej kdykoliv opět připojit. Připojené knihovny se načítají vždy při spuštění programu QCExpert, po přesunutí nebo smazání příslušného souboru QCL bude knihovna a její funkce nedostupné a bude nutné knihovnu odpojit a znovu připojit. Okno s připojenými uživatelskými knihovnami je znázorněno na následujícím obrázku.

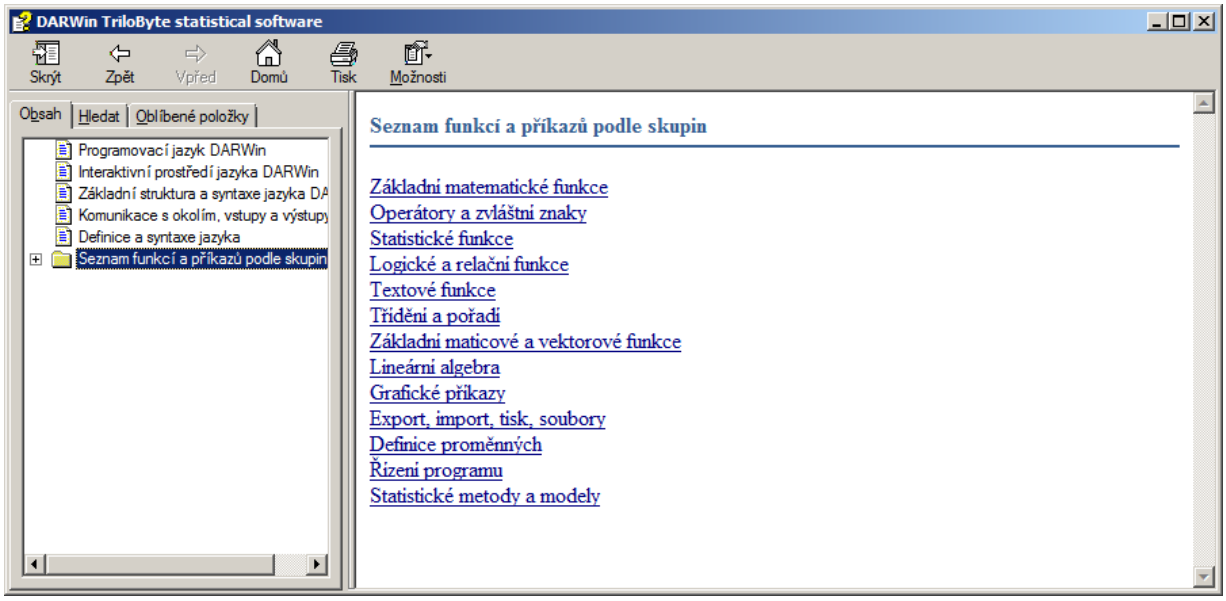


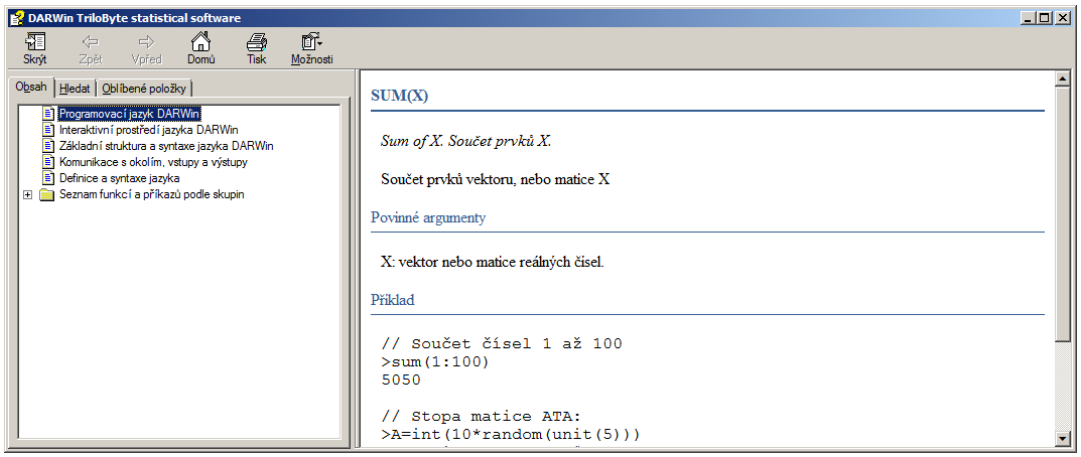
Okno s připojenými uživatelskými knihovnami

6.6. Help – systém nápovědy DARWin

6.6.1. Standardní nápověda DARWin

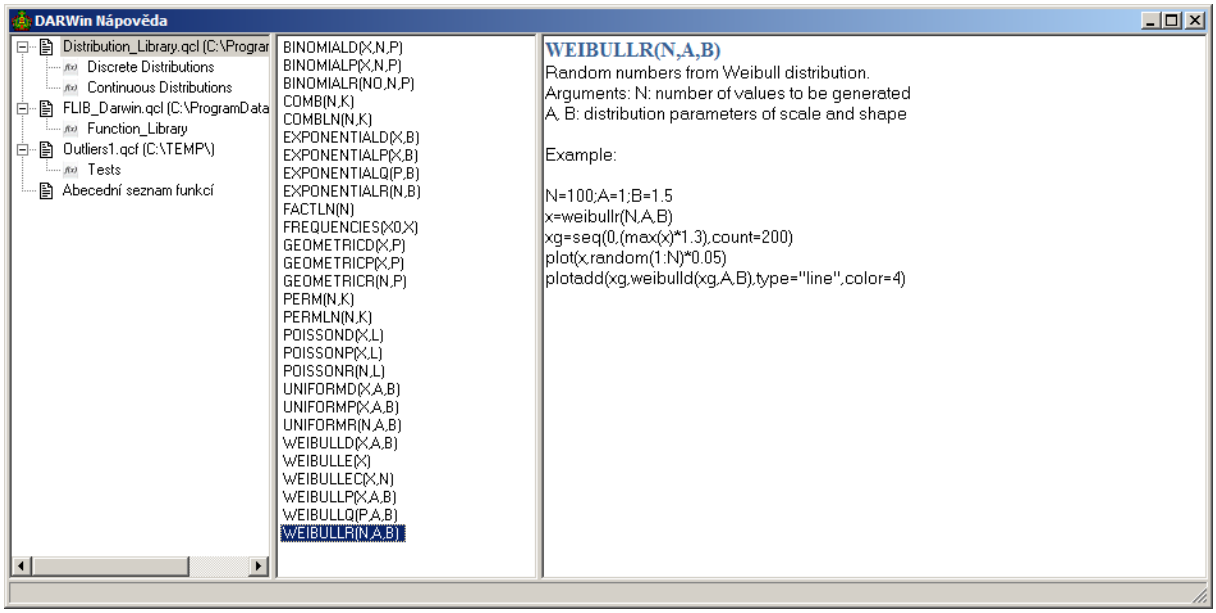
Nápověda k prostření DARWin je k dispozici po stisku klávesy *F1*. Je-li kurzor ve skriptovém okně na názvu některé funkce, otevře se nápověda k této funkci s definicí, popisem a případnými příklady. Jinak se zobrazí úvodní stránka nápovědy. Příklady je možno zkopírovat (Ctrl-C, Ctrl-V) a spouštět ve skriptovém okně.





6.6.2. Nápověda pro uživatelské funkce a knihovnu funkcí

Nápověda pro uživatelem vytvořené funkce, viz odst. 6.3 a 6.5 se vkládá mezi hlavičku funkce a otevírací příkazovou závorku definice funkce do blokových komentářových závorek. Text uvedený v těchto závorkách se zpracovává do nápovědy pro uživatelské funkce, která je k dispozici po kliknutí na tlačítko *Nápověda DARWin – uživatelské funkce* v liště okna DARWin, viz obrázek.



Text nápovědy v tomto okně se generuje dynamicky a obsahuje všechny funkce v připojených a aktivních uživatelských knihovnách a funkce z funkčních listů v otevřeném souboru. Nemá-li funkce nápovědu, objeví se v nápovědě pouze její název a seznam argumentů z hlavičky funkce. Doporučujeme však nápovědy do funkcí vkládat. Formát nápovědy není striktně závazný, jakýkoliv text uvedený mezi /* a */ se zobrazí v dynamické nápovědě, doporučuje se však toto členění:

‹ Popis funkce ›
‹ Povinné argumenty ›
‹ Nepovinné argumenty, jsou-li nějaké ›
‹ Popis struktury výsledku, zvláště, je-li to seznam ›
‹ Spustitelný příklad, který pokud možno není závislý na externích datech ›

Příklad hlavičky uživatelské funkce s nápovědou:

```
function weibullq(P,A,B)
/*
Quantile function of Weibull distribution A*(ln(A/(A-p)))^(1/B).

Arguments:
p: real probability, 0 <= p < 1
A, B: distribution parameters, A: Scale parameter, B > 0: Shape parameter

Example:
A=2;B=2 // Weibull parameter values
p=seq(0,0.999,count=200)
plot(p,weibullq(p,A,B),type="line",main="Weibull quantile; A=2, B=2" )
*/
{
                                    …
                          <<TĚLO FUNKCE>>
                                    …
} // End function weibullq
```

Pokud jsou součástí příkladu komentáře, nelze použít blokový komentář, ale pouze řádkové komentáře začínající kódem „//“, neboť celá nápověda je již formálně blokovým komentářem.

7. Standardní příkazy a funkce jazyka

7.1. Úvodní poznámka

Jeden příkaz nebo jedno volání funkce musí být vždy na jediném řádku. Argumenty příkazů a funkcí jsou buď povinné, nebo nepovinné. Nepovinné argumenty jsou v definici uvedeny v hranatých závorkách. Pojmenované argumenty jsou uvedeny ve tvaru JMENO = HODNOTA, například MEAN = X. Jméno pojmenovaného argumentu se musí vždy uvést v tomto tvaru, u pojmenovaných argumentů není nutno dodržet pořadí. Nepojmenované argumenty je nutné zadávat v uvedeném pořadí. Například funkci pro generování náhodných čísel:

```
NORMALR(N, MEAN=X, SDEV=S)
```

je možné volat těmito způsoby (nechť `k=10`):

```
normalr(k)
normalr(k, mean=4)
normalr(k, sdev=0.1)
normalr(k, sdev=0.1, mean=5)
normalr(k, mean=5, sdev=0.1)
```

Chybné volání je například:

```
normalr(k, 4)
normalr(mean=5,k)
```

Podrobnosti o syntaxi a způsobu zápisu jsou uvedeny rovněž v odstavci 2.1 na straně 8.

7.2. Příkazy a funkce

ABS (X)

Absolute value, Absolutní hodnota

Absolutní hodnota čísla

Povinné argumenty

X: číslo, numerický vektor, nebo matice

Příklad

```
>abs(-5)
5
```

```
>abs(vec(3,-4,-5,2))
3
4
5
2
```

Viz také

`SIGN, ZERO, HEAV, INT, CEIL, FRAC`

ACOS (X)

Arc Cosine, Arcus kosinus

Povinné argumenty

X: číslo, numerický vektor, nebo matice

Viz také

`COS, ASIN, ATAN`

ACOSH(X)

Hyperbolic arc cosine. Hyperbolický arkus kosinus

Hyperbolický arkus kosinus, inverzní funkce ke kladné větvi COSH(X).

Povinné argumenty

X: číslo, numerický vektor, nebo matice, funkce je definovaná pouze pro $X \geq 1$.

Příklad

```
>acosh(1:5)
0
1.31695789692482
1.76274717403909
2.06343706889556
2.29243166956118
```

Viz také

`COSH, ASINH, ATANH`

AND(X1, X2)

And, Logical product, Logický součin.

Logický součin numerických pravdivostních hodnot X1 a X2. Číslo 0 se považuje za nepravdu (false), číslo různé od 0 (typicky 1) se považuje za pravdu (true). Hodnoty výsledku uvádí pravdivostní tabulka:

X1	X2	AND(X1,X2)
0	0	0
0	1	0
1	0	0
1	1	1

Povinné argumenty

X1, X2: číslo, numerický vektor, nebo matice obsahující pravdivostní hodnoty.

Příklad

```
>and(ge(5,3),ge(1,-1))
1

// Výběr prvků Y splňující zadané podmínky
>x=sample(0:1,10,repl=1)
>y=normalr(10)
```

```
>y[[and(not(zero(x)),ge(y,0))]]
0.99874719943179
0.810622183581444
```

Viz také

OR, NOT, XOR, GT, LT, GE, LE, EQ, NE

APPLY(X, F, DIR=1)

Apply F on columns or rows of a matrix, Aplikuje funkci F na sloupce, nebo řádky matice

Vytvoří řádkový, nebo sloupcový vektor hodnot získaných aplikací funkce F na sloupce, nebo řádky matice X. Tak lze snadno získat sloupcové, nebo řádkové průměry, součty, maxima. Funkce F může být i libovolná uživatelská funkce, jejímž jediným argumentem je vektor a výsledkem jediná hodnota.

Povinné argumenty

X: matice o rozměru (N x M)

F: Textový řetězec obsahující platný název funkce, jejímž argumentem je vektor a výsledkem jediná hodnota, například SUM, AVERAGE, MEDIAN, VAR, MIN, MAX.

DIR: numerická hodnota 1 nebo 2, určuje směr použití funkce F. Je-li DIR=1 (implicitní hodnota), vypočítají se hodnoty F po řádcích a výsledkem je sloupcový vektor délky N, je-li DIR=2, vypočítají se hodnoty F po sloupcích a výsledkem je řádkový vektor délky M.

Příklad

```
// Maxima sloupců náhodné matice A (10 x 5):
>a=matrix(normalr(50),ncols=5)
>round(apply(a,"max",dir=2),5)
1.7966    1.05819     1.37638     1.86831     2.03436

// Od sloupců matice B se odečtou
// příslušné sloupcové průměry (centrování sloupců):
>B=bind(1:4,2:5)
>B1=B-apply(B,"average",dir=2)
>B1
-1.5      -1.5
-0.5      -0.5
0.5       0.5
1.5       1.5
```

Viz také

ROWS, COLS, SPLIT, BIND, MATRIX, FUNCTION

ARG(X,Y)

Argument function. Funkce argument.

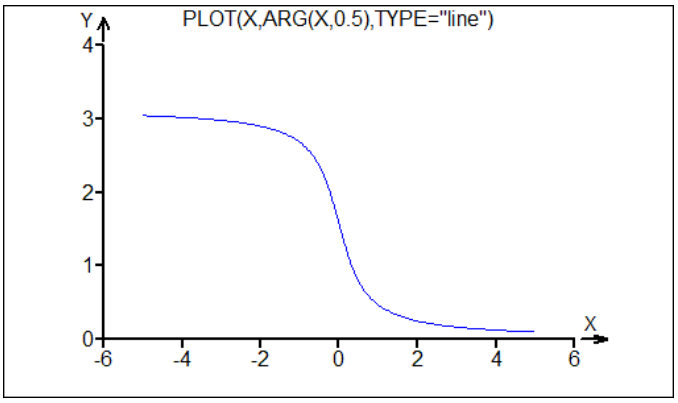
Arg-funkce je definována jako $\mathrm{Arg}(x,y) = 2\arctan\left(\frac{y}{x+\sqrt{x^2+y^2}}\right)$ pro x > 0 a y ≠ 0 a $\mathrm{Arg}(x,y) = \pi - 2\arctan\left(\frac{y}{x+\sqrt{x^2+y^2}}\right)$ pro x < 0 a y ≠ 0 a používá se v komplexní analýze, kde *x* je reálná a *y* imaginární složka komplexního čísla *z* = *x* + *iy*. Pro komplexní *z* platí, že $z = |z|e^{i\arg(z)}$.

Povinné argumenty

X, Y: reálné číslo, je-li jeden z dvojice argumentů numerický vektor, musí být druhý argument číslo (skalár).

Příklad

```
x=seq(-5,5,count=200)
plot(x,arg(x,0.5),type="line")
```



Viz také

ATAN, TANH, ATANH

ASCII(S)

ASCII code, ASCII kód

ASCII kód prvního znaku řetězce S. Poznámka: ASCII je standardní číselné kódování znaků v počítači (American Standard Code for Information Interchange).

Povinné argumenty

S: textový řetězec

Příklad

```
>ascii("ABC")
65
// Tento program vygeneruje ASCII tabulku znaků do 128:

print(transp(vec("*",astext(0:9))),\n)
for(i=0:12)
```

```
{
  print(astext(i),\t)
  for(j=0,9)
  {
    c=10*i+j
    print(chr(c),\t)
  }
  print(\n)
}
```

*	0	1	2	3	4	5	6	7	8	9
0		□	□	□	□	□	□		□	
1		□	□	□	□	□				
2	□	□	□	□	□	□	□	□	□	□
3	-			!	"	#	$	%	&	'
4	(	)	*	+	,	-	.	/	0	1
5	2	3	4	5	6	7	8	9	:	;
6	<	=	>	?	@	A	B	C	D	E
7	F	G	H	I	J	K	L	M	N	O
8	P	Q	R	S	T	U	V	W	X	Y
9	Z	[	\	]	^	_	`	a	b	c
10	d	e	f	g	h	i	j	k	l	m
11	n	o	p	q	r	s	t	u	v	w
12	x	y	z	{	|	}	~	□	€	□

Viz také

CHR, ASNUMERIC

ASIN(X)

Arc Sine, Arcus sinus.

Povinné argumenty

X: číslo, numerický vektor, nebo matice,

Viz také

SIN, ACOS, ATAN

ASINH(X)

Hyperbolic arc sine. Hyperbolický arkus sinus.

Hyperbolický arkus sinus, inverzní funkce ke kladné větvi COSH(X).

Povinné argumenty

X: Reálné číslo, numerický vektor, nebo matice.

Příklad

```
>bind((-3:3),asinh(-3:3))
```

```
-3        -1.81844645923207
-2        -1.44363547517881
-1        -0.881373587019543
 0         0
 1         0.881373587019543
 2         1.44363547517881
 3         1.81844645923207
```

Viz také

SINH, ACOSH, ATANH

ASNUMERIC(S)

As numeric, Převod na číslo

Převede řetězec na číselnou hodnotu. Pokud není řetězec interpretovatelný jako číslo, vrátí nulu.

Povinné argumenty:

S: textový řetězec

Příklad

```
>asnumeric("453"+"."+"25")+3.53
456.78
```

Viz také

ASTEXT

ASTEXT(X)

As text; Převod na textový řetězec

Převede číselnou hodnotu X na řetězec.

Povinné argumenty

X: číslo, numerický vektor, nebo matice

Poznámka: U některých řetězcových operací se konverze čísla na text se provede automaticky bez nutnosti použití funkce astext, viz příklady.

Příklady

```
>astext(3^3)+" litrů"
"27 litrů"

>3^3+" litrů"
"27 litrů"

>a=1
>b=25
>astext(a)+astext(b)
"125"
```

```
//Automatická konverze bez použití astext(1:4):
>"A"+(1:4)
"A1"
"A2"
"A3"
"A4"
```

Viz také

ASNUMERIC

ATAN (X)

Arc Tangent, Arcus tangens

Povinné argumenty

X: číslo, numerický vektor, nebo matice

Viz také

ACOS, ASIN

ATANH (X)

Hyperbolic Arc Tangent. Hyperbolický arcus tangens.

Povinné argumenty

X: reálné číslo, numerický vektor, nebo matice, -1 < X < +1.

Viz také

ACOSH, ASINH

AVERAGE (X)

AVG (X)

Arithmetic average, Aritmetický průměr

Aritmetický průměr vektoru, nebo matice

Povinné argumenty
X: číslo, numerický vektor, nebo matice

Příklad

```
>average(vec(3,4,4,5,3,4,3,4))
3.75

// Průměr milionu náhodných čísel:
>average(normalr(1000000))
0.00173227950478392
```

Viz také

MEDIAN, VAR, MEAN, APPLY

BIND(M1[,M2,M3,...])

Bind vectors or matrices, Spojení vektorů nebo matic

Spojení dvou nebo více vektorů nebo matic vedle sebe, argumenty musí mít stejný počet řádků.

Povinné argumenty

M1, M2, …: matice nebo vektory

Příklad

```
>bind(1:3, 11:13, 21:23)
1   11    21
2   12    22
3   13    23
```

Viz také

BINDV, SPLIT, SPLITV

BINDV(M1[, M2, M3, ...])

Bind vectors or matrices vertically, Spojení vektorů nebo matic pod sebe

Spojení dvou vektorů nebo matic pod sebe, argumenty musí mít stejný počet sloupců

Povinné argumenty

M1, M2, …: matice nebo vektory

Příklad

```
>bindv(vec(1,2,3),vec(4,5,6))
1
2
3
4
5
6
>bindv(transp(vec(1,2,3)),transp(vec(4,5,6)))
1   2     3
4   5     6
```

Viz také

BIND, SPLIT, SPLITV

CAPS(S)

Capitals, Velká písmena

Převod řetězce na velká písmena. Obsahuje-li řetězec S znaky „a“ až „z“, jsou převedeny na „A“ až „Z“.

Povinné argumenty

S: Textový řetězec

Příklad

```
>caps("aBcDeFgH")
"ABCDEFGH"
```

Viz také

LOWCASE, ASNUMERIC, ASTEXT, LETTERS

CEIL(X)

Ceiling, Celočíselný strop

Nejmenší celé číslo větší nebo rovno X

Povinné argumenty

X: číslo, numerický vektor, nebo matice

Příklad

```
>ceil(3.4)
4
>ceil(-3.4)
-3
```

Viz také

INT, FRAC, ROUND, FLOOR

COL(X, N)

N-th column. N-tý sloupec matice X

Vrací N-tý sloupec matice nebo vektoru X. Je-li N vektor, vrací všechny sloupce odpovídající hodnotám N ve tvaru matice, nebo vektoru.

Povinné argumenty

X: Matice nebo vektor
N: Kladné celé číslo, nebo vektor celých kladných čísel

Příklad

```
>A=bind(vec(1,2,3),vec(11,12,13))
>col(A,2)
11
12
13
```

```
>col(transp(A),1:2)
1   2
11  12
```

Viz také

ROW, SPLIT, [], VEC

COR (X)

Correlation matrix of X. Korelační matice X.

Je-li X matice s *m* sloupci a *n* řádky, vrací funkce COR čtvercovou výběrovou korelační matici C(*m* x *m*) matice X. Prvky c[i, j] matice C jsou párové korelační koeficienty mezi sloupci *i* a *j* matice X. Na diagonále C jsou jedničky.

$$c_{ij} = \frac{\hat{\sigma}^2\left(x_i, x_j\right)}{\hat{\sigma}\left(x_i\right)\hat{\sigma}\left(x_j\right)} = \frac{\sum_{k=1}^{n}\left(x_{ki} - \bar{x}_i\right)\left(x_{kj} - \bar{x}_j\right)}{\sqrt{\sum_{k=1}^{n}\left(x_{ki} - \bar{x}_i\right)^2 \sum_{k=1}^{n}\left(x_{kj} - \bar{x}_j\right)^2}}$$

Povinné argumenty

X: Numerický vektor, nebo matice. Je-li X vektor, vrací cor(X) hodnotu 1.

Příklad

```
>x=bind(vec(3,4,6,7,9),vec(3,3,6,7,7),vec(8,7,5,4,1))
>x
    3      3      8
    4      3      7
    6      6      5
    7      7      4
    9      7      1
>cc=cor(x)
>round(cc,4)
    1           0.9299      -0.9941
    0.9299      1           -0.8909
    -0.9941     -0.8909     1
```

Viz také

VAR, AVERAGE, MEAN

COS (X)

Cosine of X. Kosinus X.

Povinné argumenty

X: Numerická hodnota, vektor, nebo matice

Viz také

`SIN, TAN, COTAN`

COSH (X)

Hyperbolic cosine of X. Hyperbolický kosinus X.

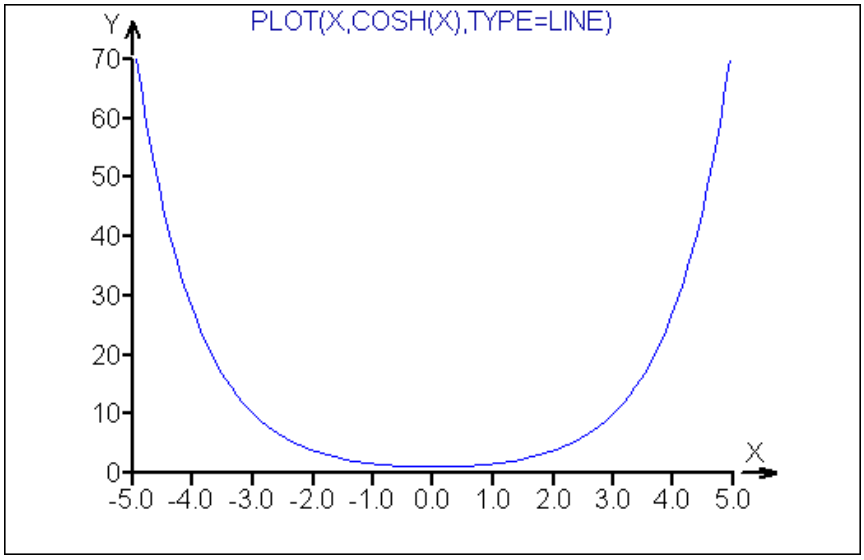
Funkce $\cosh(x) = \frac{e^x + e^{-x}}{2}$

Povinné argumenty

X: Numerická hodnota, vektor, nebo matice.

Příklad

```
x=seq(-5,5,count=200)
plot(x,cosh(x),type=line)
```



Viz také

`SINH, TANH`

COTAN (X)

Cotangent of X. Kotangens X.

Povinné argumenty

X: číslo, numerický vektor, nebo matice

Viz také

`SIN, COS, TAN`

COUNT (X)

No of elements. Počet prvků vektoru, nebo matice.

Vrací počet prvků vektoru, nebo matice X.

Povinné argumenty

X: Vektor nebo matice.

Příklad

```
>count(unit(12))
144
```

Viz také

`DIM, NCOLS, NROWS`

CUSUM(X)

Cumulative sums. Kumulativní součty.

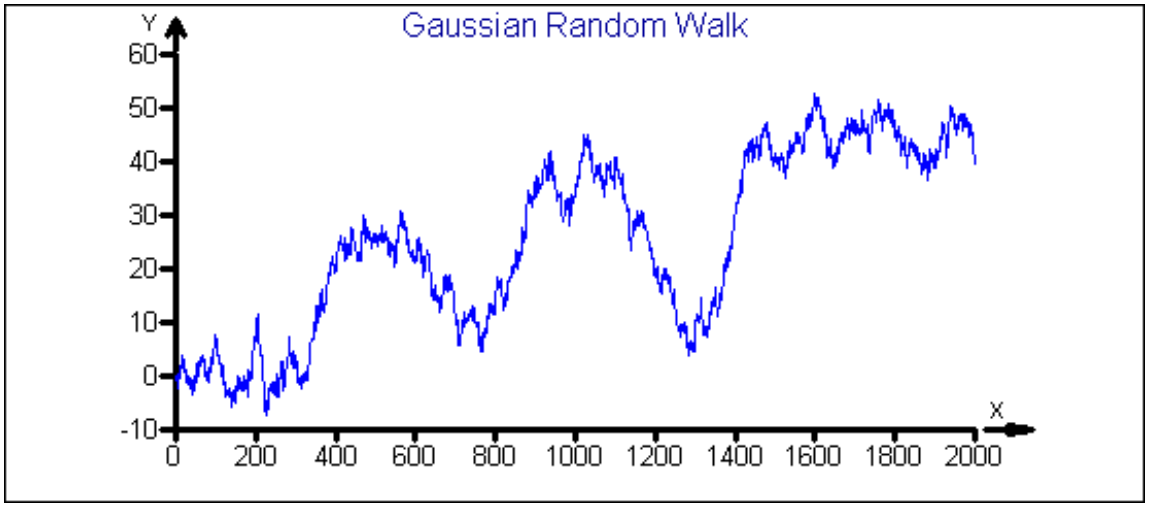
Kumulativní (částečné) součty vektoru X.

Povinné argumenty

X: numerický vektor.

Příklad

```
>cusum(vec(1,2,3))
1
3
6
>cusum(normalr(5)) // Gaussovská náhodná procházka
0.553998671372093
-0.154006915823345
-0.768427579848724
-1.44818904337812
-2.79528101487966
x=cusum(normalr(2000))
plot(x,type="line",main="Gaussian Random Walk")
```



Viz také

DIFF, SUM

DATETIMEDIFN(D1,D2)

Date – Time difference in numerical format. Rozdíl dvou časových okamžiků v numerickém formátu

Vrací počet dnů mezi časovým okamžikem D1 a D2 v desetinném numerickém tvaru. Je-li D1 > D2, je výsledek kladný.

Povinné argumenty

D1, D2: Datum a čas v textovém formátu, např. "25.3.1984 14:29:52.9". Přípustná jsou všechna platná data jsou od roku 0000 do roku 65535.

Příklad

```
//Počet dnů, které uplynuly od 1.1.2010 0:00:00
>datetimedifn(strdatetime(0),"1.1.2010")
965.518054664353
//Počet sekund, které uplynuly od 1.1.2010 0:00:00
>datetimedifn(strdatetime(0),"1.1.2010 0:0:0")*24*60*60
83420890.9790001

// Doba výpočtu v sekundách:
>t1=strdatetime(0)
>a=0
>for(i=1,10000)
>{
>a=a+i
>}
>t2=strdatetime(0)
>td=datetimedifn(t2,t1)*86400
>"Comp time = "+round(td,3)+" seconds"
"Comp time = 0.874 seconds"
```

Viz také

TIMEDIFS, TIMEDIFN, STRDATETIME, DATETIMEN, DATETIMES

DATETIMEN(D)

Convert DateTime from string to numerical format. Převod data a času z textového do numerického formátu.

Převede datum ve formátu "D.M.YYYY H:M:S.ttt" na počet dnů od 30.12.1899.

Povinné argumenty

D: Datum a čas v textovém formátu, např. "25.3.1984 14:29:52.9".

Příklad

```
>DATETIMEN("31.12.1999")
```

```
36525
>DATETIMEN(strdatetime(0))
41144.5578932523
```

Viz také

STRDATETIME, DATETIMEDIFN, DATETIMES, TIMEDIFS, TIMEDIFN

DATETIMES(X)

Convert DateTime from numerical to string format. Převod data a času z numerického do textového formátu.

Převede datum v numerickém formátu (počet dnů od 30.12.1899) do textového formátu "D.M.YYYY H:M:S.ttt". Numerický formát data může být záporný, to umožňuje zapsat data od počátku letopočtu (1.1.0000) do konce roku 65535.

Povinné argumenty

X: Numerická hodnota, nebo vektor, představující počet dnů od 30.12.1899. Je-li hodnota celočíselná, vrací se v textovém řetězci pouze datum, jinak se vrací i časový údaj.

Příklad

```
>datetimes(datetimen(strdatetime(0)))
"23.8.2012 13:49:36.996"

>datetimes(int(datetimen(strdatetime(0))))
"23.8.2012"

// 8-minutové intervaly od současného okamžiku
>dstart=datetimen(strdatetime(0))
>dd=dstart+8*(0:10)/24/60
>datetimes(dd)
"23.8.2012 13:50:13.219"
"23.8.2012 13:58:13.219"
"23.8.2012 14:06:13.219"
"23.8.2012 14:14:13.219"
"23.8.2012 14:22:13.219"
"23.8.2012 14:30:13.219"
"23.8.2012 14:38:13.219"
"23.8.2012 14:46:13.219"
"23.8.2012 14:54:13.219"
"23.8.2012 15:02:13.219"
"23.8.2012 15:10:13.219"

// 8-minutové intervaly v celých minutách
>dstart=int(datetimen(strdatetime(0))*24*60)/24/60
>dd=dstart+8*(0:5)/24/60
>datetimes(dd)
```

```
"23.8.2012 13:52:00.000"
"23.8.2012 14:00:00.000"
"23.8.2012 14:08:00.000"
"23.8.2012 14:16:00.000"
"23.8.2012 14:24:00.000"
```

Viz také

DATETIMEN, STRDATETIME, DATETIMEDIFN, TIMEDIFS, TIMEDIFN

DAYINWEEK(D)

Number of the day in week. Číslo dne v týdnu

Převede datum ve formátu "D.M.YYYY H:M:S.ttt" na pořadové číslo dne v týdnu (pondělí = 1, neděle = 7).

Povinné argumenty

D: Datum a čas v textovém formátu, např. "25.3.1984 14:29:52.9".

Příklad

```
>dayinweek("1.1.2001")
1

>dayname=vec("Pondělí", "Úterý", "Středa", "Čtvrtek",
"Pátek", "Sobota", "Neděle")
>ii=dayinweek("1.1.2001")
>dayname[ii]
"Pondělí"
```

Viz také

DAYINYEAR, STRDATETIME, DATETIMEDIFN, DATETIMES

DAYINYEAR(D)

Number of the day in year. Číslo dne v roce

Převede datum ve formátu "D.M.YYYY H:M:S.ttt" na pořadové číslo dne v roce (1.ledna = 1.den).

Povinné argumenty

D: Datum a čas v textovém formátu, např. "25.3.1984 14:29:52.9".

Příklad

```
//Přestupný rok
>dayinyear("31.12.2000")
366
```

Viz také

DAYINWEEK, STRDATETIME, DATETIMEDIFN, DATETIMES

DBCONNECT(USER=S1, PSWD=S2 [SERVER=S3, DB=S4, ROLE=S5, LOCALE=S6])

Database Connect. Připojení k databázi QCE DataCenter.

Připojí se k existující databázi QCE DataCenter. Odpojení databáze lze provést buď příkazem DBDISCONNECT, nebo připojením jiné databáze. Současně může být připojena nejvýše jedna databáze.

Povinné argumenty

USER: Textový řetězec, platné jméno uživatele databáze.
PSWD: Textový řetězec, platné heslo příslušného uživatele databáze.

Nepovinné argumenty

SERVER, DB, ROLE, LOCALE: Textové řetězce. Jméno serveru, databáze včetně úplné cesty a role.

Příklad

```
dbConnect(user="John", pswd="masterkey", SERVER="localhost",
DB="C:\trilobyte\database.fdb")
```

Viz také

DBGETFIELDS, DBGETTABLES, DBIMPORT, DBIMPORTTABLE, IMPORT, EXPORT

DBCREATE(USER=,PSWD=,SERVER=,DB=[,LOCALE=WIN1250])

Create database. Vytvořit databázi.

Vytvoří prázdnou databázi FireBird a definuje uživatele a heslo. Případné definice tabulek se provede příkazem DBCREATETABLE.

Povinné argumenty

USER: Textový řetězec, jméno administrátora nové databáze. Další jména uživatelů lze vytvářet prostředky Firebird, nebo v programu QCExpert Data Center.
PSWD: Textový řetězec, heslo pro přístup do databáze. Další hesla a uživatele lze vytvářet prostředky Firebird, nebo v programu QCExpert Data Center.
SERVER: Název nebo IP adresa serveru.
DB: Cesta a název souboru databáze s příponou .FDB.

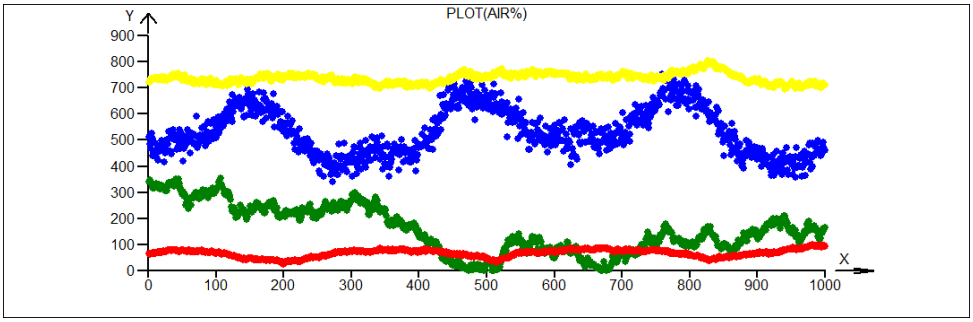
Nepovinné argumenty

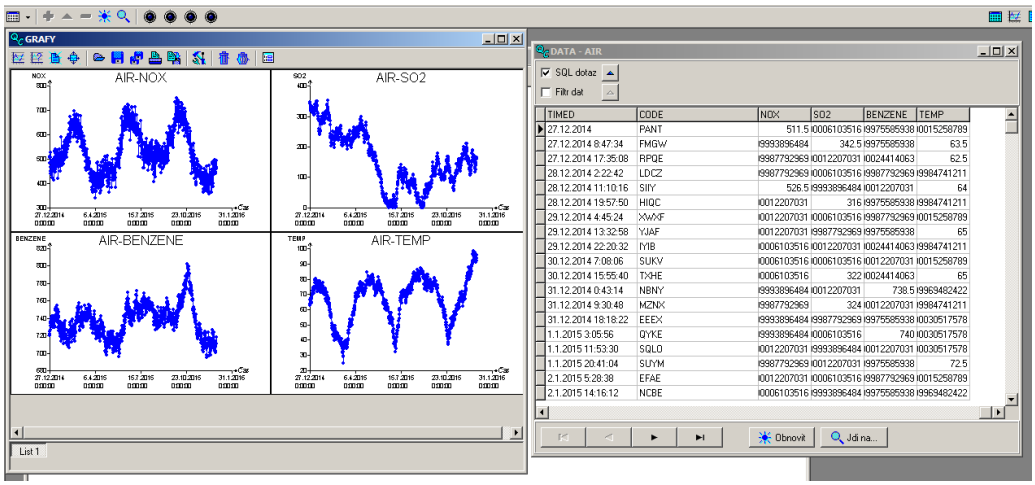
ROLE, LOCALE: Textové řetězce.

Příklad

```
// Simulace syntetických dat v DARWinu
// a jejich export do tabulky databáze:

// Vytvoření prázdné databáze
DBCREATE(USER="sysdba",PSWD="masterkey",SERVER="localhost",
DB="C:\temp\database.fdb")
```

```
// Definice typu sloupců databázové tabulky, fmt:
fmt1=vec("Timed","Code","NOX","SO2","Benzene","Temp")
fmt2=vec("DATETIME","STRING","FLOAT","FLOAT","FLOAT","FLOAT")
fmt3=vec(0,24,0,0,0,0)
fmt=bind(fmt1,fmt2,fmt3,"")

// Simulace "environmentálních" dat pro tabulku:
N=1000
ibase=1:N
// Použití BigData (%) pro případ, že by N bylo větší než 65000
dates%=datetimes(seq(42000,42366,count=N))
codes%=rep("",N)
for(i=1,N)
{ codes%[i]=letters("A",sample(1:26,4,repl=1)) }

// Syntéza "realistických" dat:
air%=3*cusum(normalr(N))+normalr(N)*10+500+200*sin(ibase/100)^4
air%=bind(air%,10*cusum(normalr(N))+normalr(N)*2+140)
air%=bind(air%,3*cusum(normalr(N))+normalr(N)*5+720)
shift=980+200*abs(sin(ibase/100-2))^0.5
air%=bind(air%,4*cusum(normalr(N))+normalr(N)*10+shift)
air%=round(air%,1)
// Grafické znázornění dat (pro kontrolu):
plot(air%)
//Připojení k databázi a vytvoření tabulky "AIR":
DBCONNECT(USER="sysdba",PSWD="masterkey",SERVER="localhost",
DB="C:\temp\database.fdb")
xx=bind(dates%,codes%,air%)
DBCREATETABLE(NAME="Air",FORMAT=fmt,MODE="DROP",DATA=xx)
```





Výsledná databázová tabulka v aplikaci TriloByte: QCExpert DataCenter®

Viz také

DBCREATETABLE, DBCONNECT, DBGETFIELDS, DBGETTABLES, DBIMPORT, DBIMPORTTABLE, IMPORT, EXPORT

DBCREATETABLE(NAME=[,FORMAT=,MODE="APPEND"|"ERASE"|"DROP",DATA=,SECURITY=1|0])

Create table in the connected database. Vytvořit tabulku v připojené databázi.

Vytvoří prázdnou tabulku v připojené databázi, případně ji naplní daty.

Povinné argumenty

NAME: Textový řetězec, název tabulky.

Nepovinné argumenty

FORMAT: Názvy polí nové tabulky ve formátu, který vrací funkce DBGETFIELDS.

MODE: Textový řetězec, "APPEND", "ERASE", "DROP" definující způsob přidání polí a dat. MODE="APPEND" přidá řádky k existující tabulce. Struktura přidávaných dat musí být shodná se strukturou existující tabulky. Pokud tabulka v databázi neexistuje, vytvoří se. MODE="ERASE" smaže případně existující tabulku v databázi a vytvoří novou ze zadaných dat. Ostatní tabulky databáze nejsou ovlivněny.

DATA: Matice s daty s odpovídajícím počtem a typem sloupců podle argumentu FORMAT.

SECURITY: Číselná hodnota 0, nebo 1. Je-li SECURITY=1, nevytvoří se nová tabulka, pokud databáze vytvořena jinak, než příkazem DBCREATE. Tím je zajištěno, aby se omylem nepoškodily tabulky v jiných databázích.

Viz také

DBCREATE, DBCONNECT, DBGETFIELDS, DBGETTABLES, DBIMPORT, DBIMPORTTABLE, IMPORT, EXPORT

DBDISCONNECT()

Disconnect database. Odpojení databáze.

Odpojí aktuálně připojenou databázi, která byla naposled připojena príkazem DBCONNECT.

Povinné argumenty

Žádné

Příklad

```
dbdisconnect()
```

Viz také

DBCONNECT

DBGETFIELDS(S1)

Get database fields. Získání názvů polí.

Získání názvů polí (sloupců) tabulky S1, Názvy jsou vráceny jako vektor textových řetězců. Databáze musí být předtím připojena pomocí příkazu DBCONNECT.

Povinné argumenty

S1 textový řetězec s názvem existující tabulky.

Příklad

```
>dbConnect(user="sysdba",pswd="masterkey")
>tab1=dbGetTables()
>dbGetFields(tab1)
"Field1"
"Field2"
"Field3"
```

Viz také

DBCONNECT, DBGETTABLES, DBIMPORT, DBIMPORTTABLE, IMPORT, EXPORT

DBGETTABLES()

Get database table. Získat tabulky databáze.

Vrací seznam názvů tabulek nalezených v připojené databázi jako vektor textových řetězců.

Příklad

```
>dbconnect(user="sysdba",pswd="masterkey")
>dbgettables()
"QC_SHEET1"
```

Viz také

DBCONNECT, DBGETFIELDS, DBIMPORT, DBIMPORTTABLE, IMPORT, EXPORT

DBIMPORT(QUERY)

Import SQR query. Import SQL dotazu.

Importuje datovou tabulku vzniklou aplikací SQL dotazu na databázi připojenou příkazem DBCONNECT. SQL dotaz je v podobě textového řetězce argumentem funkce. Syntaxe SQL je popsána v manuálu k systému QCE DataCenter.

Povinné argumenty

QUERY: Textový řetězec obsahující platný dotaz SQL.

Příklad

```
>dbconnect(USER="sysdba",PSWD="masterkey")
>dbgettables()
```

```
"TABLE1"
"TABLE2"

>tab1=dbimport("select * from TABLE1")
>dim(tab1)
7212
51
```

Viz také

DBCONNECT, DBGETFIELDS, DBGETTABLES, DBIMPORTTABLE, IMPORT, EXPORT

DBIMPORTTABLE(TABLE [, STARTDATE=, ENDDATE=, FIELDS=, VALIDONLY=1|0, SYSFIELDS=0|1])

Import table from QCE DataCenter database. Import tabulky z databáze QCE DataCeter.

Importuje datovou tabulku z databáze připojené příkazem DBCONNECT. Název tabulky je dán textovým argumentem TABLE. Z tabulky lze vybrat pouze časový interval (podle standardních sloupců QCE DataCenter Datum a Čas) a vybrané sloupce (pole), pouze platné záznamy, případně s nebo bez systémových sloupců. Tento příkaz je speciální případ funkce DBIMPORT bez použití SQL query.

Povinné argumenty

TABLE: Textový řetězec, název tabulky.

Nepovinné argumenty

STARTDATE: Textový řetězec. Počáteční datum ve formátu "DD.MM.YY".

ENDDATE: Textový řetězec. Konečné datum ve formátu "DD.MM.YY".

FIELDS: Vektor textových řetězců (získaný např. funkcí DBGETFIELDS), které se mají importovat.

VALIDONLY: Číslo 0, nebo 1. Pokud je VALIDONLY=0, importují se všechny záznamy. Pokud je VALIDONLY=1, importují se pouze platné záznamy.

SYSFIELDS: Číslo 0, nebo 1. Pokud je SYSFIELDS=0, neimportují se systémová pole. Pokud je SYSFIELDS=1, importují se všechna pole, není-li v argumentu FIELDS určeno jinak. Implicitní hodnota je 0.

Příklad

```
>dbconnect(USER="sysdba",PSWD="masterkey")
>dbgettables()
"TABLEA"
"TABLEB"
>tab1=dbimporttable("TABLEB")
>dim(tab1)
7212
40
```

Viz také

DBCONNECT, DBGETFIELDS, DBGETTABLES, DBIMPORT, IMPORT, EXPORT

DELETE(V1, [V2,...])

DEL(V1, [V2,...])

Delete variables. Vymazání proměnných.

Vymaže jednu nebo více proměnných v seznamu argumentů.
Poznámka: Vymazání proměnné je nutné například před definicí vektoru nebo matice pomocí indexové adresy. Vymazanou proměnnou nelze obnovit.

Povinné argumenty

V1, V2, ...: Názvy proměnných (nikoli textové řetězce)

Příklad

```
>a=5
>b=10
>a
5
>b
10
>delete(B)
>b
Error : "Proměnná "B" není definovaná"

>a=5
>delete(a)
>a[3,3]=0    // Naplnění matice nulami
>a
0   0     0
0   0     0
0   0     0
```

Viz také

DELETEVARS

DELETESHEET(P1)

Delete data sheet in QCExpert Data window. Smaže určený list z okna DATA QCExpertu.

Smaže datový list s názvem P1 v datovém okně QCExpertu.

Povinné argumenty

P1: Textový řetězec, název datového listu, který se má smazat. Nerozlišují se malá a velká písmena. Pokud zadaný list neexistuje, neprovede se žádná akce a nedojde k chybě. Při pokusu o smazání posledního listu se vymaže pouze jeho obsah, list zůstane zachován. Příkaz je ekvivalentní ručnímu smazání listu v prostředí QCExpertu(Menu: *Formát – List – Smazat*).

Příklad

```
deletevars
sname="DATA_X"
b=1:20
putsheet(b,sname)
putsheet(b*20,"DATA_Y")
deletesheet(sname)

// Smazání všech existujících listů v okně DATA:
// (zůstane pouze základní list "Sheet1")

snames=getsheetnames()
for(s=snames)
{
deletesheet(s)
}
```

Viz také

GETSHEET, GETSHEETNAMES, PUTSHEET, DELETEVARS

DELETEVARS

Delete all variables. Smazání všech proměnných.

Smaže BEZ VAROVÁNÍ všechny proměnné v pracovním prostoru. Do pracovního souboru se případná změna proměnných dostane až po ukončení práce, nebo po uložení pracovního prostoru. Vymazané proměnné nelze obnovit.

Příklad

```
deletevars
```

Viz také

DELETE, DEL

DET(X)

Determinant. Determinant matice.

Vrací determinant čtvercové matice X1

Povinné argumenty

X: čtvercová numerická matice

Příklad

```
>a=matrix(round(normalr(9),2),ncols=3)
>a
-0.31     -0.73       -0.82
0.8       1.87        0.68
1.43      -0.86       0.14
>det(a)
```

```
1.866384
```

Viz také

```
INV, PINV, SVD, NORM
```

DIAG(X)

Diagonal. Diagonála/Diagonální matice.

Je-li X vektor délky N, vrací čtvercovou diagonální matici (NxN) s vektorem X na diagonále. Je-li X matice (NxM), vrací vektor délky min(N,M) s diagonálními prvky matice X.

Povinné argumenty

X: vektor nebo matice

Příklad

```
>a=diag(vec(1,2,3))
>a=inv(a)
>diag(a)
1
0.5
0.333333333333333
```

Viz také

```
TRANSP, UNIT, MATRIX
```

DIALOG(DLGPARS=dp, Xi=list(TYPE= dlgtype,[VAL=value, LABEL=itemlabel, [ROWS=rows, FUN=command, CLOSE=closewin, LABELS=rblabels]])

Create and display user dialogbox. Vytvoří a zobrazí uživatelské dialogové okno

Tato funkce vytvoří interaktivní dialogové okno podle zadaných parametrů a zobrazí ho. Každé dialogové okno má vždy dvě systémová tlačítka: *OK* a *Zavřít*. Po stisku tlačítka *Zavřít* (Cancel) se okno zavře a neprovede se žádná akce. Po stisku tlačítka *OK* se aktuální hodnoty všech položek okna uloží do seznamu a tento seznam se vrátí jako výsledek funkce. Prvky seznamu mají jména shodná se jmény Xi (např X1, X2, X3, ...) použité v argumentu funkce. Počet prvků dialogového okna není omezen. Položky v okně se vytvářejí v pořadí, v němž byly zapsány jako argumenty funkce DIALOG. Globální vlastnosti dialogového okna se definují v argumentu DLGPARS, který musí mít strukturu seznamu, viz povinné argumenty. Aby byl možný přístup k hodnotám zadaným v dialogovém okně, je nutné vždy přiřadit výsledek funkce DIALOG do proměnné. Hodnoty položek jsou pak přístupné jako prvky seznamu.

Argumenty

DLGPARS: Seznam s povinnými prvky:

COLWIDTH: Celé číslo. Šířka položek dialogového okna v bodech, např. 100, nebo 200 apod.

NCOLS: Celé číslo. Počet sloupců položek v dialogovém okně, typicky 1, 2, nebo 3.

NAME: Textový řetězec. Název okna, který se objeví v jeho záhlaví.

Je výhodné pro lepší přehlednost definovat tento argument před vlastním voláním funkce DIALOG, viz příklad.

Xi: Definice položky dialogového okna. Jméno tohoto argumentu určuje uživatel. Toto jméno je pak použito jako název prvku výsledného seznamu. Tento argument je seznam, jehož struktura závisí na typu položky. Povinný prvek seznamu je

TYPE: Textový řetězec určující typ položky. Další prvky jsou formálně nepovinné: VAL, LABEL, LABELS, ROWS, FUN, CLOSE, jejich význam je uveden u každého typu. Přípustné hodnoty prvku TYPE jsou:

TYPE= "editnum": Jednořádkové editační okénko pro numerickou hodnotu.
Vrací: Obsah editačního řádku v okamžiku stisku tlačítka *OK*.
Další prvky:

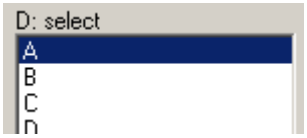
VAL: Numerická hodnota. Počáteční hodnota v editačním řádku.
LABEL: Textový řetězec. Název položky.

TYPE= "editstr": Jednořádkové editační okénko pro textový řetězec.
Vrací: Obsah editačního řádku v okamžiku stisku tlačítka *OK*.
Další prvky:

VAL: Textový řetězec. Počáteční hodnota v editačním řádku.
LABEL: Textový řetězec. Název položky.

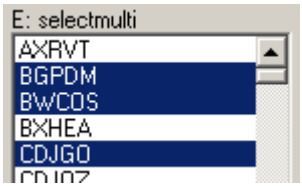
TYPE= "drop": Vyklápěcí seznam.
Vrací: Vybranou hodnotu v okamžiku stisku tlačítka *OK*.
Další prvky:

VAL: Textový nebo numerický vektor. Prvky vyklápěcího seznamu.
LABEL: Textový řetězec. Název položky.
ROWS: Počet řádků, které se objeví po vyklopení seznamu.



TYPE= "select": Seznam s možností výběru jedné hodnoty.
Vrací: Vybranou hodnotu v okamžiku stisku tlačítka *OK*.
Další prvky:

VAL: Textový nebo numerický vektor. Prvky seznamu.
LABEL: Textový řetězec. Název položky.
ROWS: Počet zobrazených řádků seznamu.



TYPE= "selectmulti": Seznam s možností výběru více hodnot.

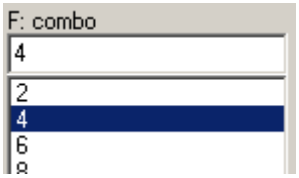
Vrací: Vybranou hodnotu, případně vektor vybraných hodnot v okamžiku stisku tlačítka *OK*.

Další prvky:

VAL: Textový nebo numerický vektor. Prvky seznamu.

LABEL: Textový řetězec. Název položky.

ROWS: Počet zobrazených řádků seznamu.



TYPE= "combo": Seznam s možností výběru jedné hodnoty a editačním řádkem.

Vrací: Obsah editačního řádku v okamžiku stisku tlačítka *OK*.

Další prvky:

VAL: Textový nebo numerický vektor. Prvky seznamu.

LABEL: Textový řetězec. Název položky.

ROWS: Počet zobrazených řádků seznamu.

TYPE= "slider": Posuvka pro „spojité“ nastavení číselné hodnoty s editačním řádkem. Nastavenou hodnotu lze v editačním řádku libovolně změnit.

Vrací: Obsah editačního řádku v okamžiku stisku tlačítka *OK*.

Další prvky:

VAL: Dvouprvkový numerický vektor s počáteční a koncovou hodnotou.

LABEL: Textový řetězec. Název položky.

TYPE= "spinner": Výběr ze seznamu hodnot pomocí šipek. Vybranou hodnotu lze v editačním řádku libovolně změnit.

Vrací: Obsah editačního řádku v okamžiku stisku tlačítka *OK*.

Další prvky:

VAL: Textový nebo numerický vektor. Prvky seznamu.

LABEL: Textový řetězec. Název položky.

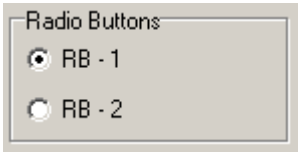
TYPE= "check": Zaškrtávací okénko pro zadání logické hodnoty.

Vrací: Numerickou hodnotu 0, je-li okénko nezaškrtnuté, numerickou hodnotu 1, je-li okénko zaškrtnuté v okamžiku stisku tlačítka *OK*.

Další prvky:

VAL: Numerická hodnota 0, nebo 1. Určuje počáteční stav okénka.

LABEL: Textový řetězec. Název položky.



TYPE= "radiobuttons": Přepínač pro výběr právě jedné z několika hodnot.

Vrací: Celočíselnou hodnotu 1 až *n* (*n* je počet políček přepínače) danou pořadovým číslem zaškrtnutého políčka v okamžiku stisku tlačítka *OK*.

Další prvky:

VAL: Celočíselná hodnota od 1 do *n*. Určuje počáteční stav přepínače.

LABEL: Textový řetězec. Název skupiny přepínačů.

LABELS: Vektor textových řetězců. Názvy jednotlivých přepínačů.

TYPE= "button": Tlačítko pro spuštění uživatelského příkazu. Při stisku tlačítka se provede funkce DIALOG vždy dosazení aktuálních hodnot do výsledné proměnné.

Vrací: Hodnotu výrazu FUN, pokud existuje.

Další prvky:

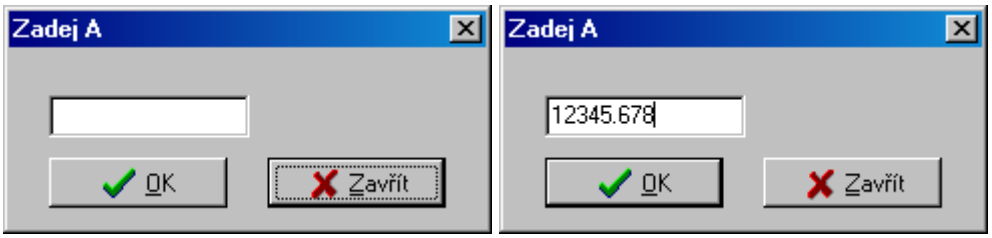
FUN: Textový řetězec obsahující platný výraz nebo příkaz. Ve výrazu je možné použít i aktuální hodnoty jednotlivých položek dialogového okna. Výhodným použitím tlačítka je volání uživatelské funkce.

LABEL: Textový řetězec. Jednořádkový název tlačítka.

CLOSE: Numerická (logická) hodnota 0, nebo 1. Je-li CLOSE=0, okno se po stisku tlačítka a po vyhodnocení, nebo provedení FUN obnoví s nezměněnými hodnotami. Je-li CLOSE=1, provede se FUN, stávající hodnoty všech položek se uloží a okno se zavře.

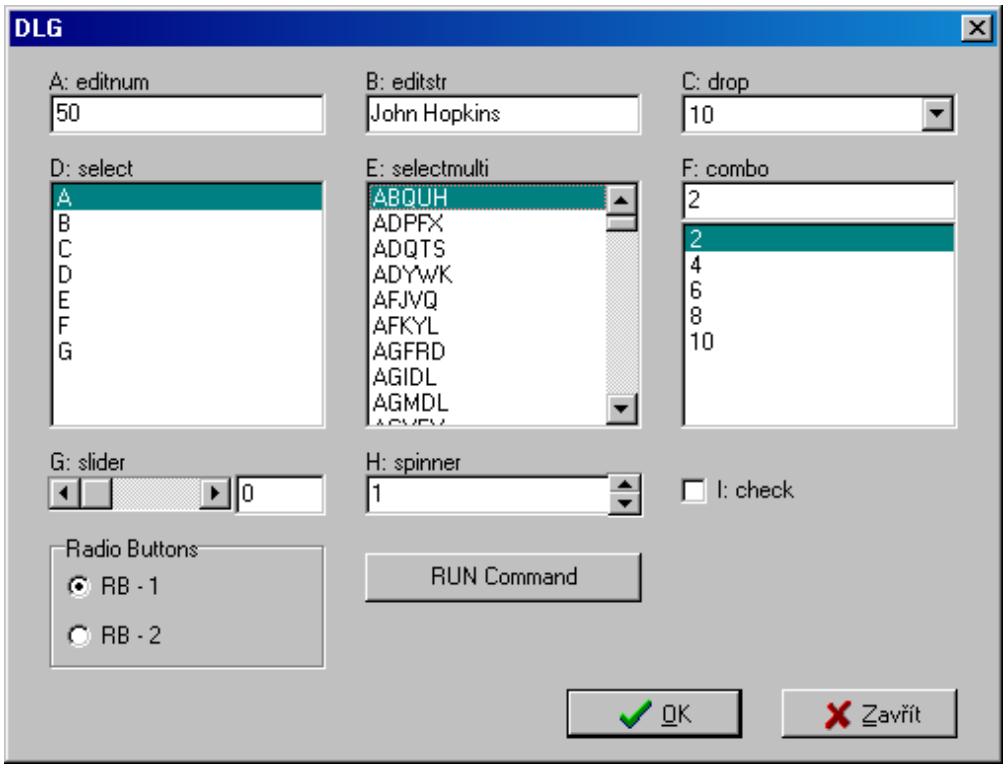
Příklad

```
// Nejjednodušší dialogové okno, 1 sloupec
>dp=list(colwidth=100, ncols=1, name="Zadej A")
>dd=dialog(dlgpars=dp, A=list(type="editnum"))
>dd$a
12345.678
```




```
// Dialogové okno se všemi interaktivními položkami,
// 3 sloupce:
s=rep("",100)
for(i=1,100){s[i]=chr(sample(65:90,5))}
s=sort(1,s)
x=0
dp=list(colwidth=140, ncols=3, name="DLG")
```

```
@dd=dialog(dlgpars=dp,
A=list(type="editnum", val=50, label="A: editnum"),
B=list(type="editstr", val="John Hopkins",
label="B: editstr"),
C=list(type="drop", val=10*(1:5), rows=3, label="C: drop"),
D=list(type="select", val=vec("A","B","C","D","E","F","G"),
rows=6, label="D: select"),
E=list(type="selectmulti", val=s, rows=6,
label="E: selectmulti"),
F=list(type="combo", val=2*(1:5), rows=5, label="F: combo"),
G=list(type="slider", val=vec(0,100), label="G: slider"),
H=list(type="spinner", val=1:10, label="H: spinner"),
I=list(type="check", val=0, label="I: check"),
J=list(type="radiobuttons", val=1,label="Radio Buttons",
labels=vec("RB - 1", "RB - 2")),
K=list(type="button", fun="x=x+1", label="RUN Command",
close=0)
);
```



Viz také

`FUNCTION, PARSE, MESSAGE`

DIFF(X [, ORD=1])

Difference. Diference vektoru.

Není-li zadán argument ord, nebo je-li ord=1, pak pro vektor X délky N vrací vektor diferencí X[2]-X[1], X[3]-X[2], … délky N-1. Je-li ord >1, vrací vektor diferencí řádu ord délky N-ORD. Diference řádu ord se počítají opakováním DIFF ord-krát.

Povinné argumenty

X: numerický vektor
ORD: celočíselná hodnota, řád diference (odpovídá řádu derivace)

Příklad

```
>a=vec(1,4,9,16,25,36,49)
>diff(a)
3
5
7
9
11
13
```

```
>a=vec(1,4,9,16,25,36,49)
>diff(a,ord=2)
2
2
2
2
2
```

Viz také

CUSUM, SUM

DIM(X)

Dimension. Dimenze matice nebo vektoru.

Vrací dvouprvkový vektor s počtem řádků a počtem sloupců matice, nebo vektoru X

Povinné argumenty

X: numerický vektor nebo matice

Příklad

```
>a=vec(1,4,9,16,25,36,49)
>dim(a)
7
1

>a=diag(vec(1,2,3,4))
>dim(a)
4
4
```

Viz také

NCOLS, NROWS, MATRIX, COUNT

EIGENVAL(X)

Real Eigenvalues. Reálná vlastní čísla (hodnoty) symetrické čtvercové matice.

Pro čtvercovou a symetrickou matici X rozměru NxN vrací vektor délky N obsahující reálné vlastní hodnoty matice X. Funkce nekontroluje symetrii X, pro asymetrickou matici X nevypočítá správné hodnoty, neboť vlastní čísla nesymetrické matice jsou obecně komplexní.

Povinné argumenty

X: symetrická čtvercová numerická matice

Příklad

```
>a=matrix(normalr(15),ncols=3) // Náhodná matice [5x3]
>b=transp(a)#a // Symetrická matice [3x3]
>eigenval(b) // Vlastní čísla
0,468742773405623
2,60870891107615
3,42363802627921
```

Viz také

EIGENVEC, SVDD, SVDU, SVDV

EIGENVEC(X)

Eigenvectors. Reálné vlastní vektory symetrické čtvercové matice.

Pro čtvercovou a symetrickou matici X rozměru NxN vrací matici rozměru NxN obsahující reálné vlastní vektory matice X. Funkce nekontroluje symetrii X, pro asymetrickou matici X nevypočítá správné hodnoty, neboť vlastní vektory nesymetrické matice jsou obecně komplexní.

Povinné argumenty

X: symetrická čtvercová numerická matice

Příklad

```
>a=matrix(normalr(15),ncols=3)
>b=transp(a)#a
>round(eigenvec(b),3)
-0.704    -0.307      0.64
-0.534    -0.366     -0.763
0.468     -0.879     0.094
```

Viz také

EIGENVAL, SVDD, SVDU, SVDV

EQ(X1, X2)

Equal. Rovnost.

Rovnost čísel, nebo řetězců X1, X2, nebo jednotlivých prvků vektoru, nebo matice. Vrací jedničku (v případě rovnosti X1 = X2), nebo nulu (v případě nerovnosti X1 ≠ X2). Výsledek má stejné dimenze jako argumenty X1, X2.

Povinné argumenty:

X1, X2 jsou čísla, textové řetězce, numerické, nebo řetězcové vektory nebo matice stejné dimenze. Pro řetězce se porovnává celá délka řetězce.

Příklad

```
>eq(3,3)
1
>eq(vec(1,2,3,4),vec(1,1,3,4))
1
0
1
1
```

Viz také

NE, LT, GT, LE, GE, ZERO

EXEC(filename [,PARAMS=,DIR=,WAIT=1|0,HIDE=0|1])

Run external application or execute DOS command. Spuštění externí aplikace nebo DOSovského příkazu.

Spustí zadaný příkaz jako kdyby byl zadán do příkazového řádku. Tento příkaz slouží ke spuštění externích příkazů a programů. Při jeho použití se doporučuje opatrnost, nesprávné použití (např. WAIT=1, HIDE=1) může zablokovat QCExpert.

Povinné argumenty:

filename: Textový řetězec, název aplikace nebo příkazu včetně cesty. Má-li se otevřít soubor známého typu (například TXT, JPG) stačí zadat jako „filename“ název tohoto souboru, který se pak otevře ve standardní aplikaci.

Nepovinné argumenty:

PARAMS: Textový řetězec, který bude předán aplikaci jako seznam argumentů.

DIR: Textový řetězec, cesta ke spouštěné aplikaci (může být uvedena jako součást názvu v povinném argumentu „filename“).

WAIT: Logická hodnota 0 nebo 1. Určuje, zda má DARWin počkat na ukončení spuštěné aplikace.

HIDE: Logická hodnota 0 nebo 1. Určuje, zda se má aplikace spustit na pozadí.

Příklad

```
tx=letters("A",sample(1:26,1000,repl=1))
export(tx,"C:\temp\mytext.txt")
exec("C:\windows\notepad.exe",params="C:\temp\mytext.txt")
```

Viz také

EXPORT, EXPORTGRAPH

EXP(X)

Exponential. Exponenciální funkce.

Povinné argumenty:

X: číslo, numerický vektor, nebo matice

Příklad

```
>exp(1)
```

```
2.71828182845905
```

Viz také

```
LN, LOG, TANH
```

EXPORT(A,FILENAME,[DELIMITER="\t",DECIMALSEPAR="."])

Export variable. Export proměnné do souboru.

Exportuje proměnnou A do textového souboru.

Povinné argumenty

A je jméno exportované proměnné bez uvozovek.
FILENAME je řetězec, nebo řetězcová proměnná s názvem souboru včetně cesty. Použitá cesta musí existovat, nové neexistující adresáře se nevytvoří.

Nepovinné argumenty

DELIMITER: je řetězec, nebo řetězcová proměnná s oddělovačem hodnot na řádku. Není-li uveden, použije se tabulátor \t.
DECIMALSEPAR: je řetězec, nebo řetězcová proměnná s desetinným oddělovačem. Není-li uveden, použije se desetinná tečka ".".

Příklad

```
>R=transp((1:10)/2)
>export(r,"c:\0\data_R.txt",delimiter=";")

// V adresáři c:\0 bude soubor "data_R.txt" s obsahem:
0.5;1;1.5;2;2.5;3;3.5;4;4.5;5
```

Viz také

```
PRINT, COPY, DBIMPORT, DBIMPORTTABLE, IMPORT, FILECOPY
```

EXPORTGRAPH(filename [, RESIZE=vec(width,height), SHEETNAME=])

Export graphics. Export grafického okna do souboru.

Exportuje všechny grafy v aktivním listu do souboru. Formát souboru je určen použitou příponou v názvu souboru. Bez zadaného názvu grafického listu se graf exportuje pouze je-li příkaz EXPORTGRAPH použit současně s některým příkazem pro tvorbu grafu (PLOT, plotadd, atd.

Povinné argumenty

FILENAME: Řetězec, nebo řetězcová proměnná s názvem souboru včetně cesty.

Nepovinné argumenty

RESIZE je dvouprvkový vektor, který určuje šířku a výšku exportovaného grafu v pixelech. Graf se před exportem transformuje na požadovanou velikost. Pokud není argument RESIZE zadán, použije se rozlišení v grafickém listu.

SHEETNAME je řetězec s názvem grafického listu, který se má exportovat.

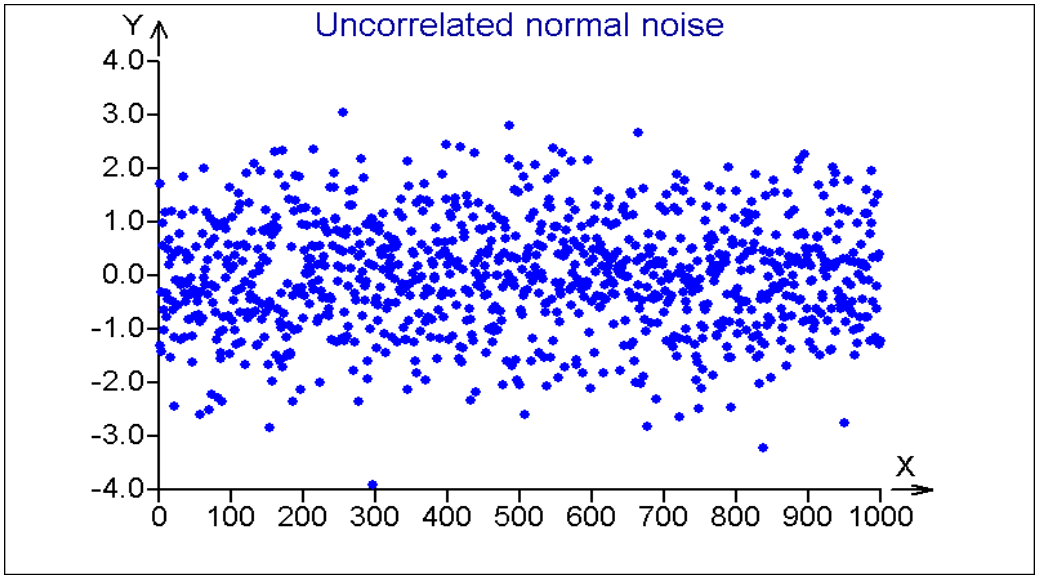
Příklad

```
x=normalr(1000)
plot(x,main="Uncorrelated normal noise")
EXPORTGRAPH("C:\0\FIG_1.jpg",resize=vec(800,480))
EXPORTGRAPH("C:\0\FIG_1.gif",resize=vec(800,480))
EXPORTGRAPH("C:\0\FIG_1.wmf",resize=vec(800,480))
EXPORTGRAPH("C:\0\FIG_1.bmp",resize=vec(800,480))
```

Poznámka:

Pokud grafika nepoužívá velké množství barev, nebo plynulé barevné přechody, je nejvýhodnější ukládat grafy ve formátu GIF, který zachovává kvalitu grafiky, ukládá však maximálně 256 barev. Formát BMP (Bitmap) zachovává 100% informace do barevné hloubky 32 bitů, nepoužívá kompresi, proto jsou soubory BMP větší a hodí se, pokud je požadována záruka 100% kvality a kompatibility (přenositelnosti). Soubor WMF (Windows Metafile) zachovává všechny informace, v některých grafických programech je dá objektově editovat (texty, symboly, barvy), při velkém množství objektů může být otevření v externím grafickém programu pomalé, některé programy nemusí objekty zobrazit správně. Soubory JPG jsou obvykle mnohem menší než, než BMP, zachovávají libovolné množství barev do hloubky 32 bitů. Používají však kompresi, která snižuje kvalitu grafiky. Orientační velikosti výsledných souborů v bajtech:

```
1 536 054 FIG_1.bmp
   14 775 FIG_1.gif
   97 772 FIG_1.jpg
  314 740 FIG_1.wmf
```



Viz také

EXPORT, PRINT, COPY

FACT (N)

Factorial of N. Faktoriál N.

Faktoriál celého nezáporného čísla N.

Povinné argumenty

N: Celé nezáporné číslo, vektor, nebo matice.

Příklad

```
>fact(5)
120
```

Viz také

GAMMA, LNGAMMA, PROD

FILECOPY (SRCDIR, DESTDIR, FNAME)

Copy a file or files. Kopíruje soubory na disku.

Zkopíruje soubory určené v argumentu FNAME ze zdrojového adresáře (složky) SRCDIR do cílového adresáře DESTDIR.

Povinné argumenty

SRCDIR: Textový řetězec obsahující cestu do zdrojového adresáře včetně názvu diskové jednotky.
DESTDIR: Textový řetězec obsahující cestu do cílového adresáře včetně názvu diskové jednotky.
FNAME: Textový řetězec nebo vektor textových řetězců obsahující názvy souborů včetně přípony, které se mají zkopírovat.

Příklad

```
FILECOPY("C:\temp","C:\temp\darwinlog",file)
```

Viz také

FILEMOVE, FILEEXISTS, FILEDELETE, FILEFIND, EXPORT, IMPORT

FILEDELETE (SRCDIR, FNAME)

Delete a file or files. Smaže soubory na disku.

Smaže soubory určené v argumentu FNAME ze zdrojového adresáře (složky) SRCDIR.

Povinné argumenty

SRCDIR: Textový řetězec obsahující cestu do zdrojového adresáře včetně názvu diskové jednotky.
FNAME: Textový řetězec nebo vektor textových řetězců obsahující názvy souborů včetně přípony, které se mají vymazat.

Příklad

```
bigdata%=1:1000000
bigdata2%=1:100000
```

```
ff=FILEFIND("C:\TEMP\darwinvar","*.*")
bigfiles=ff$name[[gt(ff$size,1000000)]]
FILEDELETE("C:\TEMP\darwinvar",bigfiles)
```

Viz také

FILEMOVE, FILEEXISTS, FILECOPY, FILEFIND, EXPORT, IMPORT

FILEEXISTS (DIR, FNAME)

Checks if a file or files exist. Zjistí, zda v daném adresáři existují hledané soubory.

Vrací vektor stejné délky jako FNAME obsahující jedničku, pokud existuje odpovídající soubor ve vektoru FNAME v adresáři DIR, jinak nulu.

Povinné argumenty

SRCDIR: Textový řetězec obsahující cestu do zdrojového adresáře včetně názvu diskové jednotky.

DESTDIR: Textový řetězec obsahující cestu do cílového adresáře včetně názvu diskové jednotky.

FNAME: Textový řetězec nebo vektor textových řetězců obsahující názvy souborů včetně přípony, které se mají zkopírovat.

Příklad

```
>fnames=vec("DBSTAT.QCE","LRMODEL.QCE","TEXT.TXT")
>file="QCEXPERT.INI"
>FILEEXISTS("C:\temp",fnames)
1
1
0
>FILEEXISTS("C:\temp","*.qce")
1
1
1
```

Viz také

FILEMOVE, FILECOPY, FILEDELETE, FILEFIND, EXPORT, IMPORT

FILEFIND (DIR, FNAME)

Lists files in a directory. Najde soubory v adresáři.

Vrací seznam (list), který obsahuje názvy všech souborů specifikovaných ve FNAME v adresáři DIR. Seznam obsahuje také velikosti a datum vytvoření souborů.

Povinné argumenty

DIR: Textový řetězec obsahující cestu včetně názvu diskové jednotky, v níž se mají hledat soubory.

FNAME: Textový řetězec obsahující masku hledaných souborů. Maska má obvyklá pravidla DOSu, například `"*.*"`, případně `"*"` hledá všechny soubory, `"*.TXT"` hledá textové soubory s příponou TXT, apod.

Struktura výsledného seznamu

Date: Vektor textových řetězců obsahující datum vytvoření souboru ve tvaru `"DD.MM.RRRR HH.MM.SS"`, například:

"12.1.2012 15:05:08"
"12.2.2012 23:26:12"
"17.5.2012 21:56:28"

Found: Logická hodnota 0 nebo 1 indikující, zda byl nalezen alespoň jeden soubor odpovídající masce FNAME.

Name: Vektor textových řetězců obsahující jména nalezených souborů včetně přípony.

Size: Vektor numerických hodnot obsahující velikosti příslušných souborů v bytech.

Příklad

```
ff=FILEFIND("C:\TEMP\","*")
nfiles=count(ff$name)
totsize=sum(ff$size)/1024
@
message("Directory = C:\TEMP\",\n,"Number of files =
"+nfiles,\n,
"Total size = "+round(totsize,3)+" kbytes")
;
```

Viz také

`FILEMOVE, FILECOPY, FILEDELETE, FILEFIND, EXPORT, IMPORT`

FILEMOVE (SRCDIR, DESTDIR, FNAME)

Moves a file or files. Přesune soubory na disku.

Přesune soubory určené v argumentu FNAME ze zdrojového adresáře (složky) SRCDIR do cílového adresáře DESTDIR.

Povinné argumenty

SRCDIR: Textový řetězec obsahující cestu do zdrojového adresáře včetně názvu diskové jednotky.
DESTDIR: Textový řetězec obsahující cestu do cílového adresáře včetně názvu diskové jednotky.
FNAME: Textový řetězec nebo vektor textových řetězců obsahující názvy souborů včetně přípony, které se mají přesunout.

Příklad

```
file="QCEXPERT.INI"
```

```
FILECOPY("C:\temp","C:\temp\darwinlog",file)
```

Viz také

FILECOPY, FILEEXISTS, FILEDELETE, FILEFIND, EXPORT, IMPORT

FISHERP(X, N1, N2)

Fisher cumulative probability. Distribuční funkce Fisherova rozdělení.

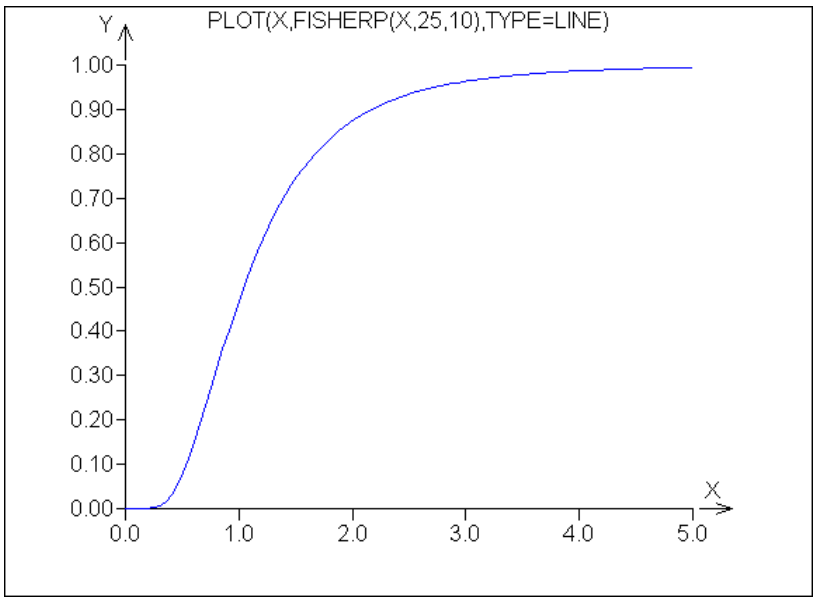
Distribuční funkce Fisherova rozdělení s N1 a N2 stupně volnosti, X je kvantil, funkce vrací pravděpodobnost mezi 0 a 1. Je-li argument X vektor s více prvky, vrací funkce vektor všech pravděpodobností.

Povinné argumenty

X: vektor nezáporných reálných hodnot.
N1, N2: Celá kladná čísla, počet stupňů volnosti v čitateli a jmenovateli.

Příklad

```
// p-hodnota statistiky STA:
>STA=3
>1-fisherp(STA,5,10)
0.0655575620938438

// Graf distribuční funkce:
x=seq(0,5,count=100)
plot(x,fisherp(x,25,10),type="line")
```



Viz také

FISHERQ, NORMALP, CHISQP, STUDENTP

FISHERQ(P,N1,N2)

Fisher quantile. Kvantil Fisherova rozdělení.

Kvantilová funkce Fisherova rozdělení s N1 a N2 stupně volnosti, P je pravděpodobnost, funkce vrací F-kvantil. Je-li argument P vektor s více prvky, vrací funkce vektor všech kvantilů.

Povinné argumenty

P je vektor nezáporných reálných hodnot mezi 0 a 1, $x \geq 0$, $x < 1$.
N1, N2: Celá kladná čísla, počet stupňů volnosti v čitateli a jmenovateli.

Příklad

```
>fisherq(0.99,5,10)
5,63632618766905
```

Viz také

FISHERP, NORMALQ, CHISQQ, STUDENTQ

FLOOR(X)

Floor value. Spodní celé číslo.

Největší celé číslo menší nebo rovno X.

Povinné argumenty

X: vektor reálných hodnot.

Příklad

```
>floor(vec(2.2,2.6,3.9,-1.1))
2
2
3
-2
```

Viz také

TRUNC, ROUND, INT, FRAC

FOR(I=N1, N2) {}

FOR(I=X) {}

For cycle. Příkaz cyklu.

Řídicí cyklus programu „FOR“ s řídicí proměnnou pro iterace a opakované výpočty.

Povinné argumenty

I je řídicí proměnná cyklu, N1 je počáteční číselná hodnota cyklu, N2 je koncová číselná hodnota cyklu. Alternativně lze definovat cyklus pomocí vektoru; X je číselný nebo textový vektor.

Tělo cyklu musí být vždy ohraničeno příkazovými závorkami. Tělo cyklu se opakuje s hodnotou proměnné I rostoucí po jedné od round(N1) do round(N2). Je-li round(N1) = round(N2), tělo cyklu se provede jednou. Je-li round(N1) > round(N2), tělo cyklu se přeskočí.

Pokud je cyklus definován pomocí vektoru X, provede se tělo cyklu count(X)-krát, přičemž I bude nabývat postupně všech hodnot vektoru X.

Příklad

```
>for(i=1,10){print(i,"-")}
1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 10 -

//Aproximace pí
a=0
for(R=(1:300)^4)
{
a=a+1/R
}

>sqrt(sqrt(a*90))
3.14159264467573
```

Viz také

WHILE, IF, :, SEQ

FRAC(X)

Fractional part of X. Desetinná část čísla X.

Vrací desetinnou část kladného nebo záporného čísla, X – INT(X).

Povinné argumenty

X: reálné číslo, vektor, nebo matice.

Příklad:

```
>frac(vec(-1.23,-0.1111,0.999,12.5))
-0.23
-0.1111
0.999
0.5
```

Viz také

TRUNC, ROUND, INT, FLOOR

FUNCTION name (ARG)

User function definition. Definice uživatelské funkce

Definice uživatelské funkce je přípustná pouze ve funkčním skriptovém listu označeném na záložce grafickým symbolem *f(x)*. Za příkazem FUNCTION následuje v témže řádku hlavička funkce, která musí obsahovat jméno funkce a v kulatých závorkách seznam formálních argumentů funkce. Jméno funkce se řídí stejnými pravidly jako jména proměnných, musí začínat písmenem a obsahovat jen písmena bez diakritiky a číslice. Počet formálních argumentů není shora omezen, ale musí být nejméně jeden. Na dalších řádcích následuje tělo funkce v příkazových závorkách { }. V těle lze používat

formální argumenty jako proměnné. Tělo funkce by mělo obsahovat příkaz RETURN, který funkci ukončí a vrátí hodnotu funkce volající instanci. Blíže také viz odst. 6.3 na straně 33.

Povinné argumenty

ARG: Seznam jmen formálních proměnných, jejichž hodnoty budou funkci předány z volající instance při volání této funkce.

Příklad:

```
//Definice funkce
//ve funkčním skriptovém listu:

function ctverecplusjedna(x)
{
a=x*x
return(a+1)
}

//Proměnná a má platnost pouze v instanci
//volané funkce a neovlivní případnou
//proměnnou a ve volající instanci.
```

```
// Volání funkce
// z volající instance:

>T=3
>ctverecplusjedna(T)
10

// Nebo:
>ctverecplusjedna(3)
10
```

Viz také

RETURN, odst. 6.3.

GAMMA(X)

Gamma function. Gamma funkce.

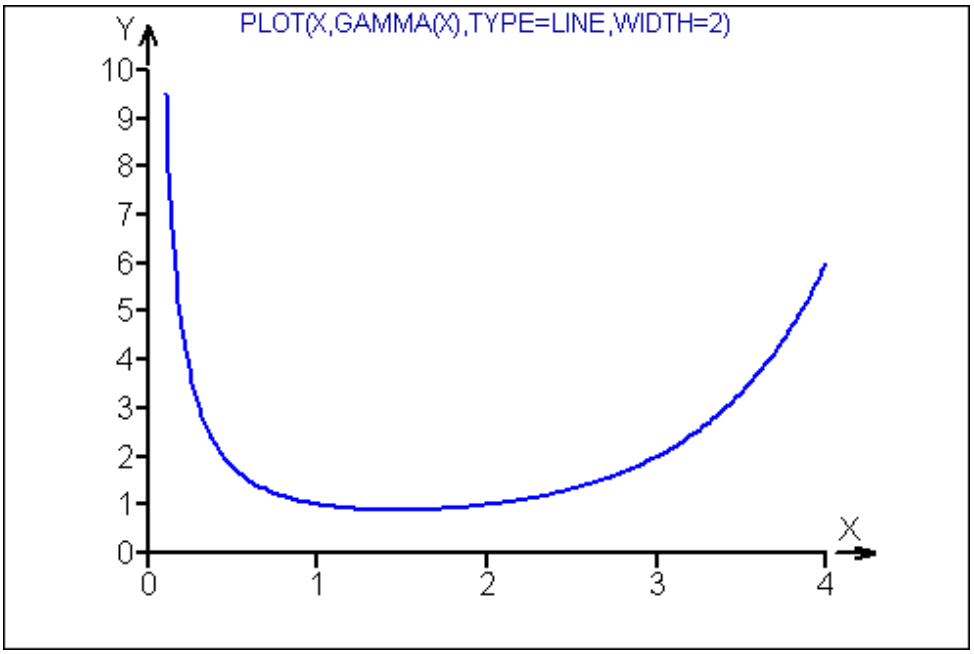
Vrací hodnotu gamma funkce $\Gamma(x) = \int_0^\infty z^{x-1} e^{-z}\, dt$. Pro celočíselné kladné hodnoty je $\Gamma(n) = (n-1)!$

Povinné argumenty

X: Reálné číslo, vektor, nebo matice.

Příklad

```
>x=(4:10)/2
>bind(x,gamma(x))
2       1
2.5     1.32934038817914
3       2
3.5     3.32335097044784
4       6
4.5     11.6317283965674
5       24

x=seq(0.1,4,count=200)
plot(x,gamma(x),type="line",width=2)
```



Viz také

GAMMALN, FACT

GAMMALN(X)

Log Gamma function. Logaritmus gamma funkce.

Vrací hodnotu logaritmu gamma funkce $\ln\Gamma(x)$, pro celočíselné kladné hodnoty je $\ln\Gamma(n) = \ln((n-1)!)$

Povinné argumenty

X: Reálné číslo, vektor, nebo matice.

Příklad

```
>gammaln(25)
54.7847293981123
>ln(fact(24))
54.7847293981123
```

Viz také

GAMMA, FACT

GE(X1, X2)

Greater or Equal. Větší nebo rovno.

X1 je větší, nebo rovno X2, pro čísla, nebo řetězce X1, X2, nebo jednotlivé prvků vektoru, nebo matice. Vrací jedničku (v případě, že X1 ≥ X2), nebo nulu (v případě, že X1 < X2). Výsledek má stejné dimenze jako argumenty X1, X2.

Povinné argumenty

X1, X2: čísla, textové řetězce, numerické, nebo řetězcové vektory nebo matice stejné dimenze. Pro řetězce se porovnává celá délka řetězce.

Příklad

```
>ge(2,3)
0

>ge(vec(1,2,3,4),vec(1,4,2,3))
1
0
1
1

>GE("Se","Sa")
1
```

Viz také

EQ, NE, LT, GT, LE, ZERO

GETIMAGE(filename)

Get numerical data form a BMP image file. Načte data z grafického souboru BMP jako numerickou matici.

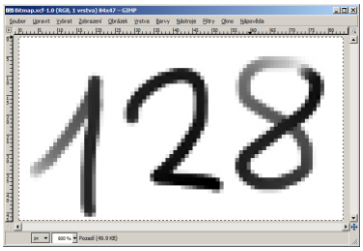
Tato funkce načte obsah zadaného grafického souboru BMP (Windows Bitmap) a převede do numerické matice, kde každému bodu (pixelu) grafiky odpovídá jedna buňka numerické matice. Číselné hodnoty v matici odpovídají RGB-barvě pixelu v použitém nastavení. Např. ve 24-bitové grafice: 0=černá, 16777215 (= FFFFFF = $2^{24} - 1$) = bílá. Je třeba mít na paměti, že výsledná matice má stejný rozměr jako bitová mapa, případně bude nutno použít proměnnou *BigData*. Možné použití – analýza obrazu z technologické kamery, apod.

Povinné argumenty

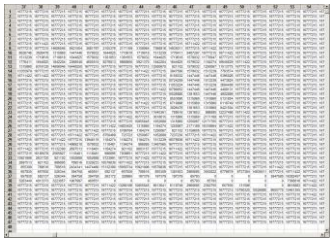
filename: Textový řetězec s názvem grafického souboru. Soubor musí mít formát BMP.

Příklad

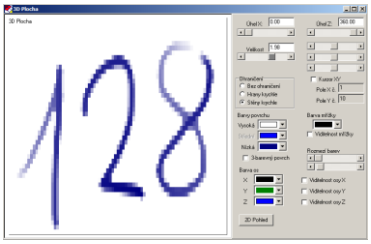
```
img=getimage("C:\TEMP\Bitmap.bmp")
plot3Dsurface(img)
```



Zdrojový obrázek v grafickém editoru



Datová matice načtená funkcí getimage



Možné zobrazení datové matice příkazem plot3Dsurface

Viz také

PUTIMAGE, EXPORTGRAPH, EXPORT, IMPORT

GETSHEET(P1)

Get data form QCExpert data sheet. Načte data z datového listu QCExpertu do proměnné.

Načte obsah datového listu P1 v okně *Data* programu *QCExpert* do proměnné. Pokud určený list neexistuje, dojde k chybě.

Povinné argumenty

P1: Textový řetězec s názvem existujícího datového listu. Velikost písmen se nerozlišuje.

Příklad

```
>x=getsheet("sheet1")

///*****************
jmeno="list1"
x=normalr(20)
putsheet(x,jmeno,header="Var1")
a=getsheet(jmeno)
varname=getsheetheader(jmeno)
```

Viz také

PRINT, PRINTSHEET, EXPORT, PUTSHEET, GETSHEETHEADER, GETSHEETNAMES

GETSHEETHEADER(P1[, ALL=0|1])

Get header form QCExpert data sheet P1. Načte názvy sloupců z datového listu P1 QCExpertu do proměnné.

Načte názvy neprázdných (případně všech) sloupců datového listu P1 v okně *Data* programu *QCExpert* do proměnné typu textového vektoru. Pokud určený list neexistuje, dojde k chybě.

Povinné argumenty

P1: Textový řetězec s názvem existujícího datového listu. Velikost písmen se nerozlišuje.

Nepovinné argumenty

ALL: Číselná hodnota 0, nebo 1. Implicitní hodnota je 0. Určuje, zda se se mají načíst názvy všech 256 (i prázdných) sloupců datového listu.

Příklad

```
>jmeno="list1"
>x=matrix(normalr(20),ncols=5)
```

```
>putsheet(x,jmeno,header="Var"+(1:5))
>a=getsheet(jmeno)
>varnames=getsheetheader(jmeno)
>varnames
"Var1"
"Var2"
"Var3"
"Var4"
"Var5"
```

Viz také

PRINT, PRINTSHEET, EXPORT, PUTSHEET, GETSHEET, GETSHEETNAMES

GETSHEETNAMES()

Get names of all existing data sheets in QCExpert „Data“ window. Načte názvy všech datových listů v okně „Data“ QCExpertu.

Načte názvy existujících listů v okně *Data* programu *QCExpert* do proměnné typu textového vektoru.

Povinné argumenty

žádné

Příklad

```
>getsheetnames()
"Sheet1"
```

GRAPHSHEET([COLS=N, NAME= , ERASE=1|0])

Graphsheet columns. Nastavení počtu sloupců v grafickém listu.

Vymaže obsah aktuálního grafického listu a nastaví počet sloupců pro grafy. Grafy se umisťují na list po řádcích. Nastavení musí být provedeno během spuštěného skriptu a má platnost jen do ukončení skriptu. Bez tohoto příkazu se grafy umisťují do jednoho sloupce.

Nepovinně parametry

COLS=N: Počet grafů v listu vedle sebe. Grafy se vkládají do listu po řádcích.

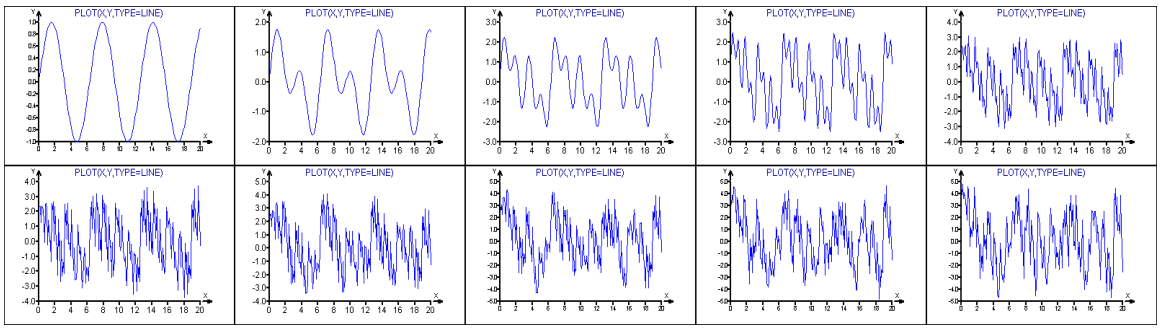
NAME: Textový řetězec, jméno listu, do něhož se bude kreslit. Pokud list neexistuje, vytvoří se.

ERASE: Je-li hodnota 1, vymaže se před vytvořením následujícího grafu obsah grafického listu. Je-li hodnota 0, existující grafy na listu jsou zachovány.

Příklad

```
graphsheet(cols=5)
x=(1:200)/10; y=sin(x)
for(i=1,10)
{
 plot(x,y,type="line")
```

```
 y=y+sin(2^i*x)
}
```



Viz také

PRINTSHEET, PLOT, EXPORTGRAPH

GT(X1,X2)

Greater Than. Větší než.

X1 je větší, než X2, pro čísla, nebo řetězce X1, X2, nebo jednotlivé prvků vektoru, nebo matice. Vrací jedničku (v případě, že X1 > X2), nebo nulu (v případě, že X1 ≤ X2). Výsledek má stejné dimenze jako argumenty X1, X2.

Povinné argumenty

X1, X2: čísla, textové řetězce, numerické, nebo řetězcové vektory nebo matice stejné dimenze. Pro řetězce se porovnává celá délka řetězce.

Příklad

```
>gt(2,3)
0
>gt(vec(1,2,3,4),vec(1,4,2,3))
0
0
1
1
```

Viz také

EQ, NE, LT, LE, GE, ZERO

HEAV(X)

Heaviside step function. Znaménková skoková funkce.

Vrací 0, je-li X záporné, jinak vrací 1.

Povinné argumenty

X: číslo, nebo číselný vektor, nebo matice.

Příklad

```
print(transp(bind(-5:5,heav(-5:5))))
```

-5	-4	-3	-2	-1	0	1	2	3	4	5
0	0	0	0	0	1	1	1	1	1	1

Viz také

SIGN, ZERO

CHISQP(X, N)

Chi-square distribution function. Distribuční funkce rozdělení chí-kvadrát (χ^2).

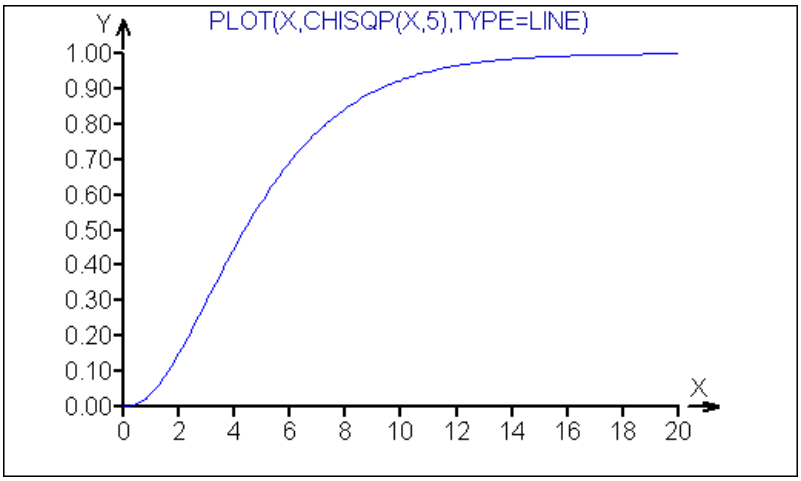
Funkce vrací pravděpodobnost odpovídající kvantilu X rozdělení χ^2 s N stupni volnosti. Je-li argument X vektor s více prvky, vrací funkce vektor všech pravděpodobností.

Povinné argumenty

X: číslo, nebo reálný vektor, N je kladné celé číslo.

Příklad

```
>chisqp(5,5)
0.584119813004183

x=seq(0.001,20,count=200)
plot(x,CHISQp(x,5),type="line")
```



Viz také

CHISQQ, NORMALP, STUDENTP, FISHERP

CHISQQ(p, N)

Chi-square quantile. Kvantilová funkce rozdělení chí-kvadrát (χ^2).

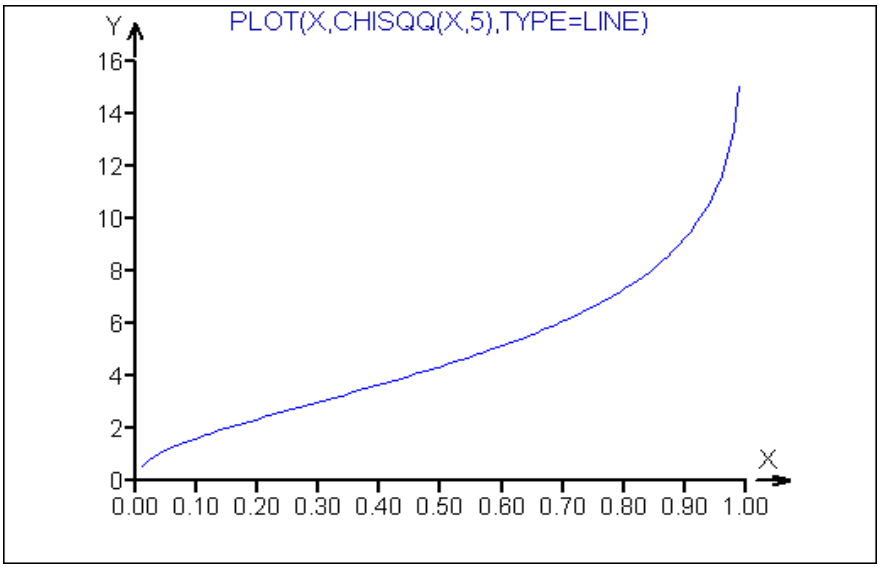
Funkce vrací pravděpodobnost odpovídající kvantilu x rozdělení χ^2 s N stupni volnosti. Je-li argument p vektor s více prvky, vrací funkce vektor všech kvantilů.

Povinné argumenty

P: reálný vektor
N: kladné celé číslo

Příklad

```
>chisqq(0.99,5)
15.0862724699129

x=seq(0.01,0.99,count=100)
plot(x,CHISQq(x,5),type="line")
```



Viz také

CHISQP, NORMALQ, STUDENTQ, FISHERQ

CHR (N)

Character ASCII. Znak ASCII.

ASCII znak odpovídající rozšířenému kódu N ($0 \leq N < 255$)

Funkce vrací řetězec obsahující znaky podle hodnoty N. Je-li N vektor, vrací jediný řetězec délky count(N). Pro ascii-kódy N<32 vrací funkce CHR mezeru, Pro hodnoty N > 255 se použije N mod 256. Funkci CHR je možné použít pro vložení znaku uvozovky CHR(34), nebo jiných speciálních znaků do řetězce, viz příklady.

Povinné argumenty

N: kladné celé číslo nebo vektor celých kladných čísel.

Příklad

```
>chr(97)
"a"
>chr(97:100)
"abcd"
>chr(sample(65:90,4)) // 4 náhodná písmena
"KNUG"

print(15+chr(176)+27+"'"+13.92+chr(34))
15°27'13.92"
```

Orientační tabulka ASCII znaků (kód znaku je 16*řádkový index + sloupcový index), viz rovněž dekadickou tabulku u funkce ASCII. Znaky s kódem nad 128 se mohou lišit podle nastavení jazyka a kódové stránky.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0																
1																
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	□
8	€	□	‚	□	„	…	†	‡	□	‰	Š	‹	Ś	Ť	Ž	Ź
9	□	‘	’	“	”	•	–	—	□	™	š	›	ś	ť	ž	ź
10		ˇ	˘	Ł	¤	Ą	¦	§	¨	©	Ş	«	¬	-	®	Ż
11	°	±	˛	ł	´	µ		•	¸	ą	ş	»	Ľ	˝	ľ	ż
12	Ŕ	Á	Â	Ă	Ä	Ĺ	Ć	Ç	Č	É	Ę	Ë	Ě	Í	Î	Ď
13	Đ	Ń	Ň	Ó	Ô	Ő	Ö	×	Ř	Ů	Ú	Ű	Ü	Ý	Ţ	ß
14	ŕ	á	â	ă	ä	ĺ	ć	ç	č	é	ę	ë	ě	í	î	ď
15	đ	ń	ň	ó	ô	ő	ö	÷	ř	ů	ú	ű	ü	ý	ţ	˙

Viz také

`ASCII, ASTEXT, LETTERS`

IF(EXPR) {}

IF control structure. Řídicí příkaz podmíněného větvení.

Je-li výraz EXPR pravdivý, nebo různý od 0, provedou se příkazy ve složených příkazových závorkách, jinak se obsah této závorky ignoruje.

Povinné argumenty

EXPR reálná hodnota (logický výraz)

Příklad

```
// Počítání odmocniny jen pro nezáporná čísla
x=normalr(5)
for(i=1,5)
{
print(x[i]);if (ge(x[i],0)){print(\t,sqrt(x[i]))}
print(\n)
}
```

1.548342089	1.244323948
0.3986913991	0.6314201447
-0.2721562541	
-0.4820386738	
0.06227708705	0.2495537759

Viz také

WHILE, FOR

IMPORT(A, FILENAME, [DELIMITER="\t", DECIMALSEPAR=".", STARTROW=, ENDROW=])

Import from file. Import proměnné ze souboru.

Do proměnné A se načte obsah souboru FILENAME. Pokud proměnná A existovala, přepíše se novým obsahem. Typ proměnné je určen strukturou dat v souboru. Soubor musí mít formát odpovídající požadovanému formátu proměnné. Hodnoty v řádku jsou odděleny tabulátorem. Je-li použit jiný oddělovač, je nutné jej zadat v argumentu DELIMITER.

Povinné argumenty

A: název proměnné do které se mají data ze souboru importovat.
FILENAME: název souboru s daty včetně cesty.

Nepovinné argumenty

DELIMITER: znak použitý pro oddělení hodnot v řádku, (implicitně tabulátor)
DECIMALSEPAR: desetinný oddělovač (implicitně tečka)
STARTROW: první řádek, který má být importován (implicitně 1)
ENDROW: poslední řádek, který má být importován (implicitně poslední řádek souboru)

Příklad

```
delete(a)
a=matrix(normalr(10000),ncols=10)
export(A,"C:\0\data.txt")
import(b,"C:\0\data.txt",startrow=2,endrow=11)

// Import textového souboru C:\0\data1.txt:
// Obsah souboru: 0.5;1;1.5;2;2.5;3;3.5;4;4.5;5
>import(b,"C:\0\data1.txt",delimiter=";")
>b
0.5 1     1.5   2     2.5   3     3.5   4     4.5   5
```

Viz také

EXPORT, DBIMPORT, DBIMPORTTABLE

INI(A1[, A2, A3,...])

Initialize variables. Inicializace proměnné.

Vytvoří proměnné uvedené jako argumenty a přiřadí jim prázdnou (nedefinovanou) hodnotu, viz 4.2.3, str. 17. Pokud proměnné již existovaly, vymaže nejdříve jejich obsah. Prázdné hodnoty jsou užitečné při plnění matic v cyklu FOR.

Povinné argumenty

A1, …: jména proměnných.

Příklad

```
x=1:5
powers=0:6
ini(x1)
for(i=powers)
{
x1=bind(x1,x^i)
}
table=bindv(transp("i^"+powers),x1)
>table

x^0 x^1   x^2   x^3   x^4   x^5   x^6
1   1     1     1     1     1     1
1   2     4     8     16    32    64
1   3     9     27    81    243   729
1   4     16    64    256   1024  4096
1   5     25    125   625   3125  15625
```

Viz také

DELETE, DELETEVARS

INT(X)

Integer part. Celá část čísla.

Funkce vrací celé číslo N, s největší absolutní hodnotou tak, že $|N| \leq |X|$. V případě kladného X prostě uřízne desetinnou část.

Povinné argumenty

X: reálné číslo, vektor, nebo matice.

Příklad

```
>int(vec(-2.9,2.9))
-2
2
```

Viz také

TRUNC, ROUND, FRAC, FLOOR, CEIL

INTPOWER(X,N)

Integer power. Celočíselná mocnina.

Funkce počítá N-tou mocninu čísla X pomocí postupného násobení, nikoli pomocí logaritmů. Poskytuje přesnější výsledky, zabraňuje možnosti, že celá mocnina celého čísla není celé číslo.

Povinné argumenty

X: reálný vektor, nebo číslo
N: celé číslo, nebo celočíselný vektor, je-li N necelé číslo, použije se INT(N).

Příklad

```
>intpower(vec(2,10),vec(10,3))
1024
1000
>transp(intpower(2,1:10))
2   4     8     16    32    64    128   256   512   1024
>intpower(2,-7)
0.0078125
```

Viz také

POWER, EXP, INT

INV(A)

Matrix inversion. Inverze matice.

Funkce vrací inverzní matici k matici A, takže inv(A)#A = A#inv(A) je jednotková matice.

Povinné argumenty

A: Nesingulární reálná čtvercová matice

Příklad

```
>x=matrix(int(20*normalr(9)),ncols=3)
>x
2   0     0
0   -1    -1
0   0     -4
>inv(x)
0.5 0     0
0   -1    0.25
0   0     -0.25
>x#inv(x)
1   0     0
0   1     0
0   0     1

>inv(rep(transp(1:4),4))
Error : "Nelze spočítat inverzní matici - vstupní matice je
singulární"
```

Viz také

PINV, DET, EIGENVEC, EIGENVAL

ISNUM(X)

Is numeric? Je numerická hodnota?

Funkce vrací 1, je-li prvek X číselná hodnota, jinak vrací 0.

Povinné argumenty

X: vektor nebo matice

Příklad

```
>isnum(vec(1,2,"ABC",4))
1
1
0
1

>a=vec(1,2,"ABC",4)
>ln(a[[isnum(a)]])
0
0.693147180559945
1.38629436111989
```

Viz také

ISTEXT, ZERO, ASNUMERIC, ASTEXT

ISTEXT(X)

Is text? Text?

Funkce vrací 1, je-li prvek X textová hodnota, jinak vrací 0.

Povinné argumenty

X: vektor nebo matice

Příklad

```
>istext("123+5")
1
```

Viz také

ISNUM, ASTEXT, ASNUMERIC, LENGTH

ISUNDEF(X)

Is variable undefined? Nedefinovaná proměnná?

Funkce vrací 1, je-li X nedefinovaná hodnota (vytvořená příkazem INI, nebo interaktivně tlačítkem Přidat proměnnou), jinak vrací 0.

Povinné argumenty

X: Jméno proměnné

Příklad

```
>ini(A)
>isundef(A)
```

```
1
```

Viz také

INI, DELETE, DELETEVARS, BIND, BINDV, VEC

LE(X1,X2)

Less or equal. Menší nebo rovno.

X1 je menší nebo rovno X2, pro čísla, nebo řetězce X1, X2, nebo jednotlivé prvků vektoru, nebo matice. Vrací jedničku (v případě, že X1 ≤ X2), nebo nulu (v případě, že X1 > X2). Výsledek má stejné dimenze jako argumenty X1, X2.

Povinné argumenty

X1, X2: čísla, textové řetězce, numerické, nebo řetězcové vektory nebo matice stejné dimenze. Pro řetězce se porovnává celá délka řetězce.

Příklad

```
>le(5,3)
0

>le(1:4,vec(1,4,2,3))
1
1
0
0
```

Viz také

EQ, NE, LT, GT, GE, ZERO

LENGTH(S)

Length of string. Délka řetězce.

Funkce vrací délku řetězce. Je-li S vektor řetězců, je výsledkem vektor délek jednotlivých prvků.

Povinné argumenty

S: řetězec, nebo vektor řetězců.

Příklad

```
>length("Bill Doyle")
10
>s1=vec("Anna","Carol","Dave","Elisabeth","Jim")
>length(s1)
4
5
4
9
3
```

Viz také

POS, ASNUMERIC, ISTEXT

LETTERS(S,N)

Letters or charcters starting from S. Písmena a znaky ASCII od S.

Je-li N jediné číslo, funkce vrací znak s ASCII kódem o N-1 větším než první znak řetězce S. Je-li N vektor, je výsledkem řetězec znaků s ASCII kódy N[1], N[2], atd. Druhá a další znaky řetězce S se ignorují.

Povinné argumenty

S: znak nebo textový řetězec, N je celé číslo, nebo vektor celých čísel

Příklad

```
>letters("A",10) // desáté písmeno abecedy
"J"

>letters("A",1:26) // Prvních 26 znaků abecedy
"ABCDEFGHIJKLMNOPQRSTUVWXYZ"

>letters("0",1:10)
"0123456789"

>letters("a",sample(1:26,5)) // 5 náhodných znaků malé
abecedy
"ltvfc"
```

Viz také

LENGTH, ASTEXT, SAMPLE

LINEADD(H=Y|V=X|A=Intercept,B=Slope, [COLOR=C], [SHADE=Sh], [WIDTH=W])

Add line to plot. Přidání přímky do grafu.

Příkaz zobrazí přímku zvolených vlastností do již vytvořeného grafu. Musí být proveden ve jednom spuštění s příkazem vytvářejícím nový graf. Samostatně nelze LINEADD provést. Poloha přímky je daná prvním argumentem: H=Y vytvoří horizontální přímku y=Y, V=X vytvoří vertikální přímku x=X a A=Intercept, B=Slope vytvoří obecnou přímku y=a+bx. Jsou-li Y, X vektory, případně Intercept a Slope vektory shodné délky, vytvoří se příslušný počet přímek s těmito argumenty. Vždy musí být použita jedna z možností H=, nebo V=, nebo obě hodnoty A=, B=. Ostatní argumenty příkazu jsou nepovinné.

Povinné argumenty

(buď H, nebo V, nebo A a B):
H je hodnota na ose X pro svislou přímku.
V je hodnota na ose Y pro vodorovnou přímku.

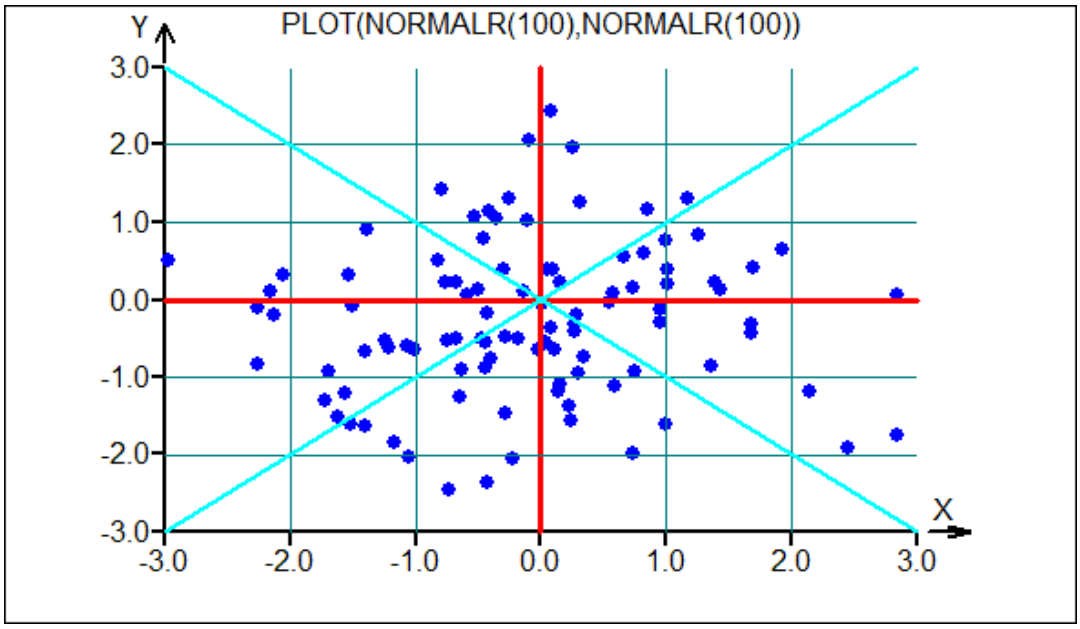
Intercept je hodnota úseku na ose Y (absolutní člen) obecné přímky.
Slope je hodnota směrnice obecné přímky.

Nepovinné argumenty

C je celé číslo odpovídající barevné škále.
Sh je sytost barvy v intervalu 1 až 100.
W je celé číslo odpovídající požadované šířce přímky v grafu (1 je nejtenčí).

Příklad

```
plot(normalr(100),normalr(100))
lineadd(h=-2:2,color=5)
lineadd(v=-2:2,color=5)
lineadd(h=0,color=3,width=3)
lineadd(v=0,color=3,width=3)
lineadd(a=0,b=1,color=6,width=2)
lineadd(a=0,b=-1,color=6,width=2)
```



Viz také

PLOT, PLOTADD, TEXT

LINREG(X,Y[,W=rep(1,nrows(X))][,xnew=X][,absolute=1][,alpha=0.05])

Linear regression. Lineární regrese.

Lineární regrese metodou nejmenších čtverců. Vstupem je matice nezávisle proměnné **X** a vektor závisle proměnné **y**. Funkce vypočítá odhady **a** vektoru parametrů **α** modelu $y = \mathbf{x}^T\boldsymbol{\alpha} + \varepsilon$, neboli $y = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \ldots \alpha_m x_m + \varepsilon$. Výsledkem jsou kromě odhadu parametrů také jejich kovarianční a korelační matice, predikované hodnoty a položšířky jejich intervalu spolehlivosti a diagonální prvky projekční matice. Vektor predikovaných hodnot pro zadané X je dán jako $\hat{y} = \mathbf{Xa}$, kde odhady vektoru parametrů $\mathbf{a} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$, případně,

jsou-li zadány váhy v argumentu W, $\mathbf{a} = \left(\mathbf{X}^T\mathbf{W}\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{W}\mathbf{y}$, kde W je diagonální čtvercová matice se zadaným vektorem W na diagonále. Regresní rezidua e_i jsou definována jako odchylky naměřených hodnot od odhadnutého modelu (tedy predikce) y, $\mathbf{e} = \mathbf{y} - \hat{\mathbf{y}} = \mathbf{y} - \mathbf{A}\mathbf{x}$. Jejich rozptyl je $s_e^2 = \frac{1}{n-m}\mathbf{e}^T\mathbf{e}$. Odhad kovarianční matice **C** regresních parametrů **a** je dána jako $\mathbf{C} = s_e^2\left(\mathbf{X}^T\mathbf{W}\mathbf{X}\right)^{-1}$, přičemž její diagonální prvky Cii jsou rozptyly jednotlivých regresních parametrů a_i. Projekční matice ("Hat" matrix) je definována jako $\mathbf{H} = \mathbf{X}\left(\mathbf{X}^T\mathbf{W}\mathbf{X}\right)^{-1}\mathbf{X}^T$ a její diagonální prvky H_{ii} odpovídají vlivu jednotlivých řádků dat na regresní model. Data odpovídající velkým hodnotám H_{ii} je vhodné prověřit a v případě pochybnosti o jejich správnosti odstranit. Interval spolehlivosti predikované hodnoty v libovolné hodnotě **x** (1 x *m*) na dané hladině významnosti α lze konstruovat pomocí vztahu $\mathbf{x}\mathbf{a} \mp s_e\sqrt{mF_{1-\alpha}(m, n-n)}\sqrt{\mathbf{x}\left(\mathbf{X}^T\mathbf{W}\mathbf{X}\right)^{-1}\mathbf{x}^T} = \mathbf{x}\mathbf{a} \mp CI_\alpha$.

Povinné argumenty

X: Vektor nebo matice hodnot nezávisle (vysvětlující) proměnné (prediktoru) s *N* řádky. Je-li X matice s *M* sloupci, musí být tyto sloupce lineárně nezávislé.

Y: Vektor hodnot závisle (vysvětlované) proměnné (odezvy) délky *N*.

Nepovinné argumenty

W: Vektor délky *N* obsahující relativní váhy jednotlivých měření. Implicitní hodnota je vektor jedniček, rep(1,N).

XNEW: reálný vektor nebo matice (N_1 x *M*) nových hodnot X, pro které má model predikovat hodnoty závisle proměnné. Implicitní hodnotou je argument X.

ABSOLUTE: Numerická hodnota 0 nebo 1, určije, zda se má do modelu zahrnout absolutní člen. Pokud je ABSOLUTE=0, model bude bez absolutního členu a bude procházet počátkem souřadnic.

ALPHA: Kladná numerická hodnota menší než 0.5, udává zvolenou hladinu významnosti α pro intervaly spolehlivosti predikce.

Struktura výsledného seznamu

Seznam s následujícími prvky:

$A: Bodové odhady regresních koeficientů, numerický vektor délky M nebo M+1, podle hodnoty argumentu ABSOLUTE.

$YPRED: Predikované hodnoty pro dané X.

$VARA: Odhad kovarianční matice regresních koeficientů, na diagonále jsou rozptyly regresních koeficientů.

$CORA: Odhad korelační matice regresních koeficientů.

$YNEW: Predikované hodnoty pro dané XNEW. Pokud není argument XNEW zadán, je tento vektor shodný s $YPRED.

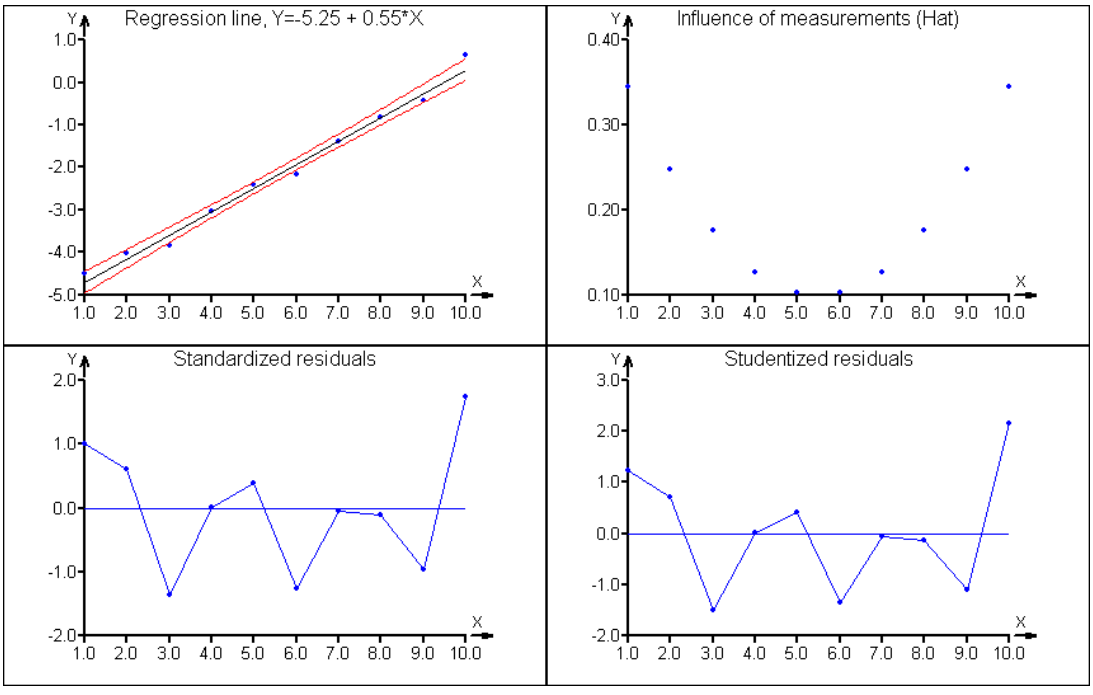
$CI: Pološirka intervalu spolehlivosti CI_α predikovaných hodnot $YNEW na zadané hladině významnosti ALPHA. Interval spolehlivosti predikce pro YNEW[i] je tedy YNEW[i]-CINEW[i] až YNEW[i]+CINEW[i].

$HATDIAG: Numerický vektor délky N. Diagonální prvky projekční matice ("Hat" matrix) $\mathbf{H}=\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$. Používá se např. k posouzení relativního vlivu jednotlivých bodů a konstrukci studentizovaných reziduí.

$SIG2: Odhad reziduálního rozptylu
$RESID: Jednotlivá rezidua (Y - YPRED), numerický vektor délky N.

Příklad

```
n=10
// Generate X and Y
x=1:n
y=0.5*x-5+normalr(n)*0.3
// Generate XNEW for plotting
xg=seq(min(x),max(x),count=200)
// Perform regression
lr=linreg(x,y,rep(1,n),xg,1,0.05)
// Set two columns in graph sheet
graphsheet(cols=2)
// Plot data
equation= "Y="+round(lr$a[1],2)+" + "+round(lr$a[2],2)+"*X"
plot(x,y,main="Regression line, "+equation)
// Add regression line Y=a+bX
lineadd(a=lr$a[1],b=lr$a[2],color=4)
// Add confidence intervals or predicted line
plotadd(xg,lr$ynew+lr$ci,type="line",color=3)
plotadd(xg,lr$ynew-lr$ci,type="line",color=3)
// Plot Hat matrix diagonal to visualize influence of the data
plot(LR$HATDIAG,main="Influence of measurements
(Hat)", ,type="pointline")
// Plot classical standardized residuals
plot(LR$RESID/sqrt(LR$SIG2),main="Standardized
residuals",type="pointline")
lineadd(h=0)
// Calculate and plot Studentized residuals (corrected for influence)
estu=LR$RESID/sqrt(LR$SIG2*(1-LR$HATDIAG))
plot(estu,main="Studentized residuals",type="pointline")
lineadd(h=0)
```



Viz také

LOCALREG, POLYREG, SPLINE1, SPLINE2, NNLEARN

LIST([name1=]Z1[...,[nameN=]Zn])

Make list. Vytvořit seznam.

Vytvoří seznam prvků. Seznam lze uložit do proměnné, prvky seznamu mohou být čísla, řetězce, vektory, matice a seznamy. Přístup k prvkům seznamu je možný pomocí „$“. Není-li použito jméno prvku, vytvoří se implicitní jméno prvků seznamu tvořené názvem proměnné a pořadovým číslem prvku 001, 002, … Seznamy se v QCExpertu používají typicky jako rozsáhlejší struktury výsledků statistických metod. Je také výhodnou strukturou pro uložení různých nesourodých výsledků (řetězec, číslo, vektory, matice) uživatelských funkcí, viz odst. 6.3, strana 33.

Povinné argumenty

Z1, Z2, …, Zn: prvky seznamu (alespoň jeden prvek).

Nepovinné argumenty

name1, name2, …, nameN: jména prvků seznamu.

Příklad

```
>R=list(P1="John",P2=vec(22,156,20110,15),P3=28)
>R$P2
22
156
20110
15
>R$P2[2]
156

// Seznam obsahující další seznam:
>r=list(A=1:5,B=list(X=1,Y="EEE",Z=(1:10)/10),C="ABCDEFGH")
>r$B$Z[4:6]
0.4
0.5
0.6
>r$C
"ABCDEFGH"

function qeq(a,b,c) // Příklad seznamu jako výsledku funkce
{ // Quadratic equation a*x^2 + b*x + c = 0
//a=1;b=1;c=-5
D=b*b-4*a*c
if(lt(D,0)){typ="Komplexní kořeny"}
if(eq(D,0)){typ="Dvojný kořen"}
if(gt(D,0)){typ="Reálné kořeny"}
xr=-b/(2*a)
x1i=sqrt(abs(D))/(2*a); x2i=-sqrt(abs(D))/(2*a)
x1r=xr+x1i; x2r=xr+x2i
if(lt(D,0)){id=-1;
x=bindv(transp(vec(xr,x1i)),transp(vec(xr,x2i)))}
if(eq(D,0)){id=0; x=vec(xr,xr)}
if(gt(D,0)){id=1; x=vec(x1r,x2r)}
```

```
return(list(typ=typ,id=id,koreny=x))
}

>Q=qeq(1,3,-5)
>Q$typ
"Reálné kořeny"
>q$koreny
1.19258240356725
-4.19258240356725

>Q=qeq(1,3,5)
>Q$typ
"Komplexní kořeny"
>q$koreny
-1.5      1.6583123951777
-1.5      -1.6583123951777
```

Viz také

$, VEC, MATRIX

LN(X)

Natural logarithm. Přirozený logaritmus.

Povinné argumenty

X: kladné reálné číslo, vektor, nebo matice

Příklad

```
>ln(10)
2.30258509299405
>ln(1:10)
0
0.693147180559945
1.09861228866811
1.38629436111989
1.6094379124341
1.79175946922805
1.94591014905531
2.07944154167984
2.19722457733622
2.30258509299405
```

Viz také

LOG, LOG2, LOGN, EXP

LOCALREG(X,Y,[POLDEG=1][,KERNEL="quad"][,KWIDTH=0.3][,XNEW=SEQ(MIN(x),MAX(x),COUNT=100)][,ALPHA=0.05])

Univariate local regression

Pro daný vektor nezávisle proměnné X a závisle proměnné Y stejné délky *N* provede lokální regresi a vrací predikované hodnoty a jejich interval spolehlivosti. Jako lokální lineární regresní model je k dispozici úplný polynom *r*-tého stupně, *r* = 0, 1, 2, 3, .., který zahrnuje konstantu při *r*=0, model odpovídá jádrovému vyhlazení klouzavým váženým průměrem, přímkový model při *r*=1, což odpovídá klasické lokální lineární regresi, kvadratický a kubický model při *r*=2, resp. *r*=3, což odpovádá polynomické lokální regresi. Vyšší stupně polynomu se obecně nedoporučují. Jsou k dispozici dvě jádra: kvadratické („quad“) a exponenciální Gaussovo („gauss“)

$$k(x) = \frac{1}{\left[5\left(x_r/R\right)^2 - 1\right]}, \text{ resp. } k(x) = \exp\left(-3\left(x_r/R\right)^2\right),$$

kde $x_r = \frac{(x_i - x)}{\max x - \min x}$, které určuje váhu jednotlivých hodnot nezávisle proměnné x_i při výpočtu regresní funkce v bodě *x*.

Povinné argumenty

X: Vektor reálných čísel délky *N* obsahující hodnoty nezávisle proměnné.

Y: Vektor reálných čísel délky *N* obsahující hodnoty závisle proměnné.

Nepovinné argumenty

POLDEG: Celé nezáporné číslo, typicky 0, 1, nebo 2 udávající stupeň lokálního regresního polynomu. Implicitní hodnota je 2.

KERNEL: Textový řetězec `"quad"` nebo `"gauss"` určující který typ jádra se použije při výpočtu vah lokální regrese. Na typu jádra obvykle zásadně nezáleží. Implicirní hodnota argumentu je `"quad"`.

KWIDTH: Numerická kladná hodnota. Relativní šířka jádra (kernel width). Čím menší hodnota. tím „lokálnější“, tedy podrobnější proložení a větší podíl šumu ve vysvětleném součtu čtverců. Tento argument je hlavním ladicím parametrem a jeho empirická volba je obvykle výsledkem několika pokusů. Implicitní hodnota je 1.

XNEW: Vektor reálných čísel délky N_1 obsahující nové hodnoty nezávisle proměnné, pro něž má být počítána predikce a interval spolehlivosti predikce, implicitní hodnota je shodná s argumentem X.

ALPHA: kladná numerická hodnota menší než 0.5, zvolená hladina významnosti pro výpočet intervalu spolehlivosti, implicitní hodnota ALPHA=0.05.

Struktura výsledného seznamu

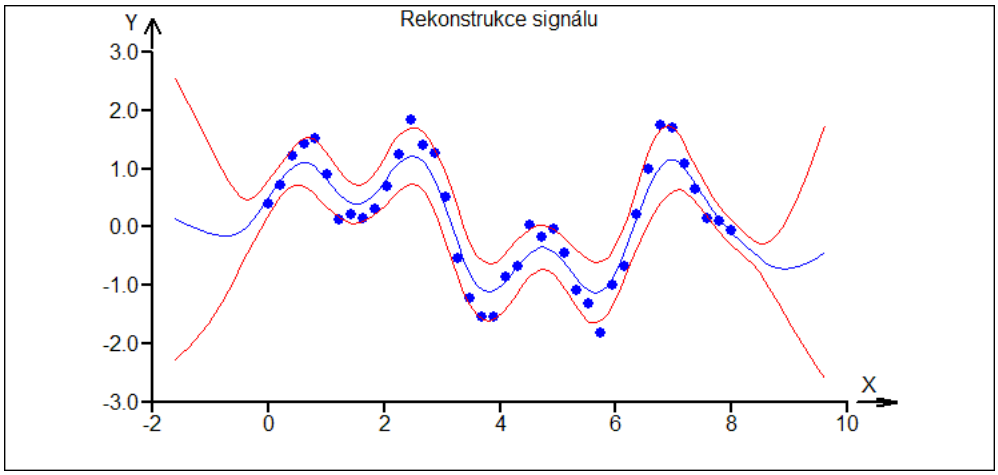
$CI: Vektor reálných čísel délky N_1 obsahující poloviční šířku konfidenčního intervalu pro každý řádek XNEW.

$YPRED: Vektor reálných čísel délky N_1 obsahující predikované hodnoty závisle proměnné pro každý řádek XNEW.

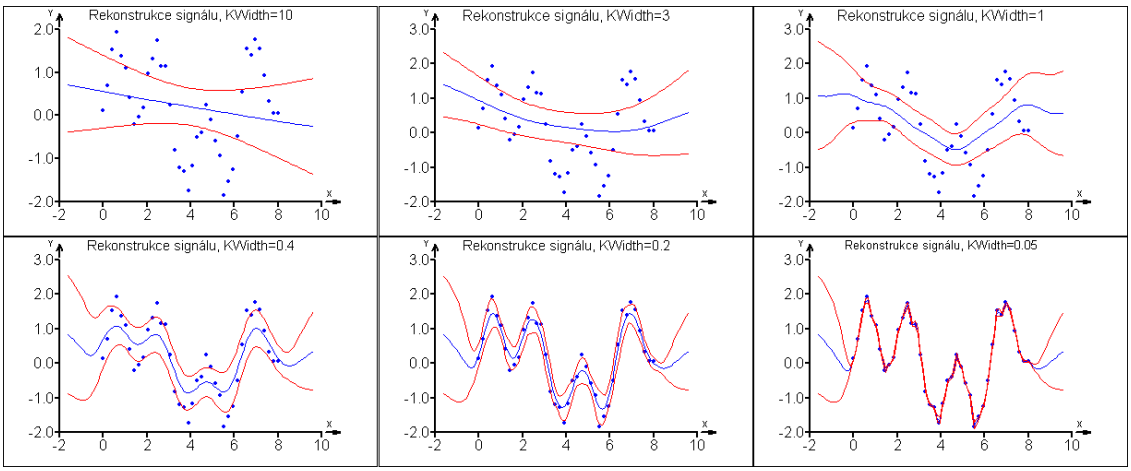
Příklad

```
// Simulace dat - součet sinů se šumem:
n=40
s=0.2
x=seq(0,8,count=n)
y=sin(x)+sin(3*x)+normalr(n)*s
```

```
// Parametry pro LOCALREG:
deg=3
ker="quad"
wd=0.3
// Data pro predikci a graf s přesahem 20%:
xmin=min(x)
xmax=max(x)
xrange=xmax-xmin
xg=seq(xmin-0.2*xrange,xmax+0.2*xrange,count=100)
// Výpočet křivky lokální regrese:
lor=localreg(x,y,poldeg=deg,kernel=ker,kwidth=wd,xnew=xg)
// Graf modelu s intervaly spolehlivosti predikce
plot(x,y,main="Rekonstrukce signálu")
plotadd(xg,lor$ypred,type="line")
plotadd(xg,lor$ypred+lor$ci,type="line",color=3)
plotadd(xg,lor$ypred-lor$ci,type="line",color=3)
//*********************************
```



Vliv šířky jádra (od 10 do 0.05) na tvar regrese, POLDEG = 1. První model se blíží regresní přímce, poslední model se blíží křivce procházející všemi body.



Viz také

LINREG, POLYREG, SPLINE1, SPLINE2

LOG (X)

Decadic logarithm. Dekadický logaritmus.

Povinné argumenty

X: kladné reálné číslo, vektor, nebo matice

Příklad

```
>log(10)
1
>log(1:10)
0
0.301029995663981
0.477121254719662
0.602059991327962
0.698970004336019
0.778151250383644
0.845098040014257
0.903089986991944
0.954242509439325
1
```

Viz také

LN, LOG2, LOGN, EXP

LOG2 (X)

Binary logarithm. Binární (dvojkový) logaritmus.

Povinné argumenty

X: kladné reálné číslo, vektor, nebo matice.

Příklad

```
>bind(1:10,log2(1:10))
1   0
2   1
3   1.58496250072116
4   2
5   2.32192809488736
6   2.58496250072116
7   2.8073549220576
8   3
9   3.16992500144231
10  3.32192809488736
```

Viz také

LN, LOG, LOGN, EXP

LOGN(Z,X)

General logarithm of the base Z. Logaritmus se základem Z.

Povinné argumenty

X a Z: kladné reálné číslo, vektor, nebo matice, Z nesmí být 1

Příklad

```
>logn(2:10,10)
3.32192809488736
2.09590327428938
1.66096404744368
1.43067655807339
1.28509720893847
1.18329466245494
1.10730936496245
1.04795163714469
1
```

Viz také

LN, LOG, LOG2, EXP

LT(X1,X2)

Less than. Menší než.

X1 je menší, než X2, pro čísla, nebo řetězce X1, X2, nebo jednotlivé prvků vektoru, nebo matice. Vrací jedničku (v případě, že X1 < X2), nebo nulu (v případě, že X1 ≥ X2). Výsledek má stejné dimenze jako argumenty X1, X2.

Povinné argumenty

X1, X2: čísla, textové řetězce, numerické, nebo řetězcové vektory nebo matice stejné dimenze. Pro řetězce se porovnává celá délka řetězce.

Příklad

```
>lt(5,3)
0
>lt(1:4,vec(1,4,2,3))
0
1
0
0
```

Viz také

EQ, NE, GT, LE, GE, ZERO

MAD(X)

Median Absolute Deviation. Mediánová absolutní odchylka

Pro reálný vektor **x** vrací hodnotu mediánu absolutních odchylek od mediánu, $MAD = \text{med}\left|x_i - \text{med}(\mathbf{x})\right|$ Tato statistika slouží jako robustní odhad variability **x**. Její modifikace (viz funkce MADS) je robustním odhadem směrodatné odchylky.

Povinné argumenty

X: numerický vektor nebo matice.

Příklad

```
>mad(vec(2,5,7,3,4,7,3,4,3,6,5))
1
```

Viz také

MADS, MEDIAN, VAR

MADS(X)

Median Absolute Standard Deviation. Mediánová absolutní směrodatná odchylka

Pro reálný vektor **x** vrací robustní odhad směrodatné odchylky σ na základě mediánu absolutních odchylek (MAD), $\hat{\sigma} = \frac{MAD}{\Phi^{-1}\left(\frac{3}{4}\right)}$.

Povinné argumenty

X: numerický vektor nebo matice.

Příklad

```
>mads(vec(2,5,7,3,4,7,3,4,3,6,5.00))
1.48260222029421
>sqrt(var(vec(2,5,7,3,4,7,3,4,3,6,5.00)))
1.69491217257039
>mads(vec(2,5,7,3,4,7,3,4,3,6,500))
1.48260222029421
>sqrt(var(vec(2,5,7,3,4,7,3,4,3,6,500)))
149.438524909987
```

Viz také

MAD, MEDIAN, VAR, APPLY

MATRIX(X,NCOLS=|NROWS= [,BYROWS=0|1])

Create matrix from vector. Vytvoření matice z vektoru.

Funkce vrací numerickou, nebo textovou matici vytvořenou z vektoru X s daným počtem sloupců nebo řádků. Matice se vytváří z prvků vektoru X buď po řádcích, nebo po sloupcích podle hodnoty argumentu BYROWS. Není-li délka vektoru X celočíselným násobkem požadovaného počtu sloupců, nebo řádků, doplní se chybějící hodnoty v matici nulami.

Povinné argumenty

X: číselný nebo řetězcový vektor, sloupcový, nebo řádkový. Prvky tohoto vektoru se použijí k naplnění matice.

NCOLS, NROWS: Musí být použit právě jeden z těchto argumentů. Celé číslo, které určuje počet sloupců, nebo počet řádků matice.

Nepovinné argumenty

BYROWS: číselná hodnota 0, nebo 1, která určuje směr naplnění matice. Implicitní hodnota je 0. Je-li hodnota argumentu 0, plní se matice po sloupcích, je-li hodnota argumentu 1, plní se matice po řádcích.

Příklad

```
// Plnění matice po sloupcích a po řádcích:
>matrix(1:16,ncols=4)
1   5     9     13
2   6     10    14
3   7     11    15
4   8     12    16

>matrix(1:16,ncols=4,byrows=1)
1   2     3     4
5   6     7     8
9   10    11    12
13  14    15    16

>matrix(rep(1:4,4),ncols=4)
1   1     1     1
2   2     2     2
3   3     3     3
4   4     4     4
```

Viz také

VEC, UNIT, BIND, BINDV

MAX(X)

Maximum. Maximum.

Funkce vrací maximální hodnotu vektoru nebo matice.

Povinné argumenty

X: číselný nebo řetězcový vektor, nebo matice

Příklad

```
>max(vec(2,5,7,3,8,4,6,8,1))
8
```

```
// Pořadová čísla (pozice) maximálních prvků vektoru r
>r=vec(2,5,7,3,8,4,6,8,1,6,0,8,7)
>maxr=max(r)
>ii=1:nrows(r)
>ii[[eq(r,maxr)]]
5
8
12

>max(vec("A","B","C"))
"C"
```

Viz také

MIN, SORT, ORDER, APPLY

MEDIAN(X)

Median of X. Medián X.

Vrací výběrový medián výběru (dat) ve vektoru, nebo matici.

Povinné argumenty

X: Reálný vektor, nebo matice.

Příklad

```
>median(1:4)
2.5
>median(1:5)
3

>x=normalr(100)
// Mediánová absolutní odchylka (MAD):
// a robustní směrodatná odchylka:

>median(abs(x-median(x)))
0.728045059231947
>median(abs(x-median(x)))/normalq(0.75)
1.07940122129151
```

Viz také

AVERAGE, VAR, MEAN, MAD, APPLY

MESSAGE([LABEL=S][, S1[, ..., Sn]])

Display text message. Zobrazí okno se zprávou.

Zobrazí informační okno se zprávou, program se nezastaví. Okno nelze při běhu programu zavřít tlačítkem. Dalším voláním funkce MESSAGE se předchozí okno zavře, lze tedy mít vždy zobrazeno nejvýše jedno okno. Voláním MESSAGE( ) bez argumentu

se okno zavře a žádné další se nevytvoří. Toto okno lze použít ke sledování průběhu programu, zobrazení mezivýsledků, odlaďování, apod.

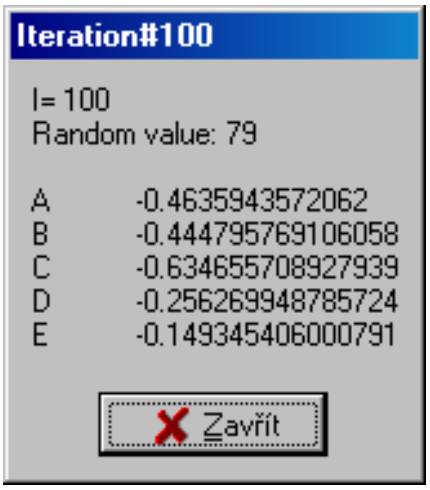
Nepovinné argumenty

Label=S: Textový řetězec, Text do záhlaví okna.

S1 – Sn: Textové řetězce, číslo, vektor, nebo matice, které se zobrazí v okně. Chceme-li zobrazit víceřádkový text, použijeme symbol konce řádku \n podobně jako např. v příkazu PRINT.

Příklad

```
for(i=1,100)
{
@message(Label="Iteration#"+i,"I= "+i,\n,
"Random value: " +int(rnd(100)),
\n,\n,bind("A":"E",normalr(5)));
pause(0.1)
}
```




```
dp=list(colwidth=150, ncols=1, name="Zadání")
@dd=dialog(dlgpars=dp,
area=list(type="editnum", val=0,
label="Zadej plochu čtverce")
);
area=dd$area
if(lt(dd$area,0)){message(label="ERROR","Plocha nemůže být
záporná");stop}
if(ge(dd$area,0)){message(label="Strana čtverce","Strana =
"+sqrt(area))}
```

Viz také

DIALOG, PAUSE, STOP, PRINT

MIN(X)

Minimum. Minimum.

Funkce vrací maximální hodnotu vektoru nebo matice.

Povinné argumenty

X je číselný, nebo řetězcový vektor, nebo matice

Příklad

```
>min(normalr(100000))
-4.39331024998306

>r=vec(2,1,7,5,8,4,6,8,1,6,1,8,7,1)
>minr=min(r)
>ii=1:nrows(r)
>ii[[eq(r,minr)]]
2
9
11
14
```

Viz také

MAX, SORT, ORDER, APPLY

MULTIVAR(X [, CORREL=0|1, BIPLOT=0|1, LOADINGS=0|1, VARIANCES=0|1, COMPO=0|1, NORMAL=0|1, ANDREWS=0|1, MAHALA=0|1])

Multivariate analysis of a matrix. Vícerozměrná analýza matice.

Statistická metoda odpovídající vícerozměrné analýze a PCA (principal component analysis – analýza hlavních komponent) programu QCExpert. X je vstupní matice dat [N x M] s N řádky a M sloupci. Výsledkem této funkce jsou grafy v grafickém listu a seznam se strukturou podle nastavení argumentů.

Povinné argumenty

X: matice reálných čísel [N x M].

Nepovinné argumenty

CORREL: 1 má-li se k výpočtu použít korelační matice (normovaná data, nezáleží na velikosti střední hodnoty a rozptylu ve sloupcích), 0 mají-li se použít původní data a kovarianční matice. Implicitní hodnota CORREL=1.

BIPLOT: 1 má-li se vytvořit graf Biplot. 0 nemá-li se vytvořit graf Biplot.

LOADINGS: 1 mají-li se vytvořit sloupcové grafy s hodnotami zátěží.

VARIANCES: 1 má-li se vytvořit sloupcový graf vysvětleného rozptylu

COMPO: 1 mají-li se vytvořit grafy dat v souřadnicích hlavních komponent

NORMAL: 1 má-li se vytvořit Q-Q graf vícerozměrné normality dat a graf symetrie.

ANDREWS: 1 má-li se vytvořit Andrewsův graf Fourierových reprezentací dat

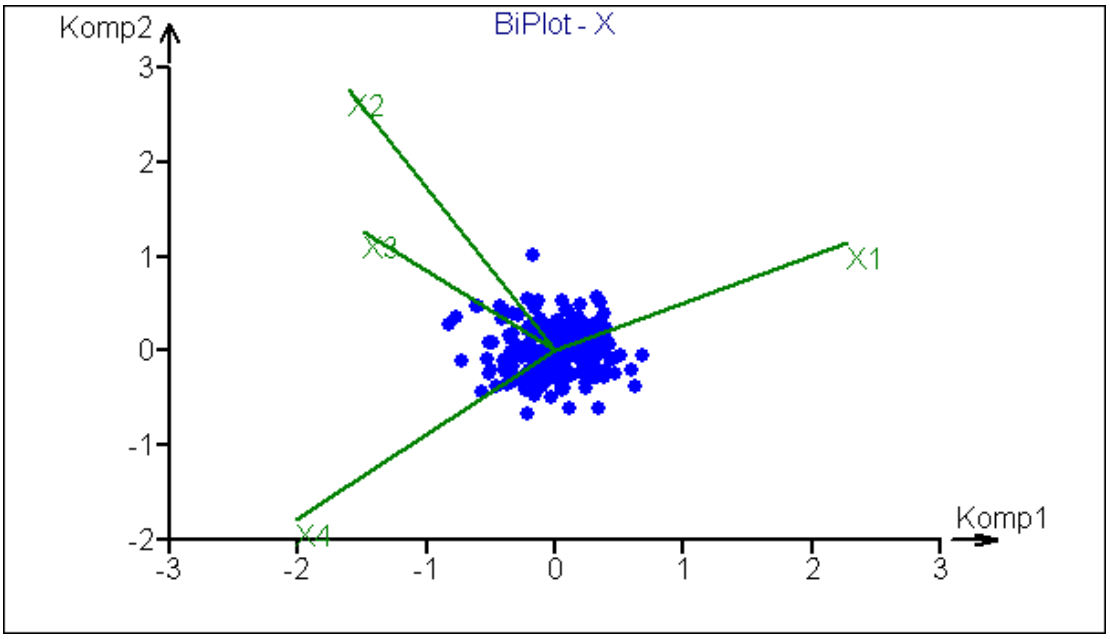
MAHALA: 1 má-li se vytvořit graf klasických a robustních Mahalanobisových vzdáleností a diagnostický graf odlehlých dat

Struktura výsledného seznamu:

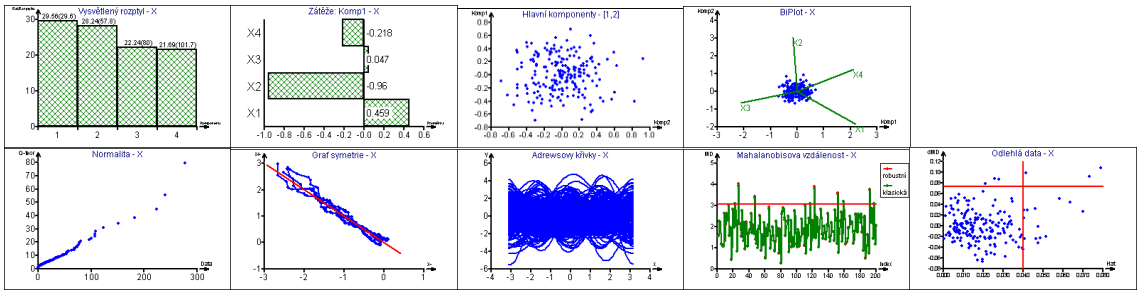
Corr: matice M x M: korelační matice dat
Cov: matice M x M: kovarianční matice dat
EigenVectors: matice M x M: vlastní vektory
ExpVar: vektor M x 1: Vysvětlený rozptyl jednotlivými komponentami
Loadings: matice M x M: Zátěže
Maha: vektor N x 1: Vektor Mahalanobisových vzdáleností
Mean: vektor M x 1: Průměry sloupců
MEstimates: vektor M x 1: Robustní průměry sloupců metodou M-odhadu
RobMaha: vektor N x 1: Robustní Mahalanobisovy vzdálenosti
Scores: matice N x M: Matice skórů
Transformed: matice N x M: matice transformovaných dat
Variance: vektor M x 1: Rozptyly sloupců

Příklad

```
>x=matrix(normalr(800),ncols=4)
>MV=multivar(x,biplot=1)
```




```
>MV=multivar(x,biplot=1,loadings=1,compo=1,normal=1,andrews=
1,mahala=1)
// (Zobrazeny jsou jen vybrané grafy)
```



```
>print(mv$corr) // Korelační matice dat
```

1	-0.1010051776	-0.06354876422	0.06975904985
-0.1010051776	1	-0.02203884906	0.06543935692
-0.06354876422	-0.02203884906	1	-0.08290587053
0.06975904985	0.06543935692	-0.08290587053	1

```
// rozptyly sloupců x z diagonály kovarianční matice:
>diag(mv$cov)
0.999201522701996
1.11477545448409
0.948125320527254
1.00695676623743
```

NCOLS (X)

Number of columns of a vector or matrix. Počet sloupců vektoru nebo matice.

Funkce vrací počet sloupců datové struktury.

Povinné argumenty

X: vektor nebo matice

Příklad

```
>ncols(1:4)
1
>ncols(transp(1:4))
4
>ncols(unit(5))
5
```

Viz také

NROWS, COUNT, DIM, MATRIX, VEC

NE (X1 ,X2)

Not equal. Není rovno.

Nerovnost čísel, nebo řetězců X1, X2, nebo jednotlivých prvků vektoru, nebo matice. Vrací jedničku (v případě nerovnosti X1 ≠ X2), nebo nulu (v případě rovnosti X1 = X2). Výsledek má stejné dimenze jako argumenty X1, X2.

Povinné argumenty

X1, X2: čísla, textové řetězce, numerické, nebo řetězcové vektory nebo matice stejné dimenze. Pro řetězce se porovnává celá délka řetězce.

Příklad

```
>ne(5,3)
1
>ne(1:4,vec(1,4,2,3))
0
1
1
```

1

Viz také

EQ, LT, GT, LE, GE, ZERO

NNLEARN(X, Y[, XNAMES=, YNAMES=, LAYERS=, MODELFILE=, ITERATIONS=, USEFORTEACH=, EXPONENT=, ALPHA=, MOMENTUM=, LEARNRATE=, IDENTERROR=, MEANERR=0|1, RESIDUALS=1|0, GRNET=0|1, GRPREDICT=0|1], BESTMODEL=0|1])

Learn Neural Network. Natrénování neuronové sítě.

Natrénuje se neuronová síť na trénovacích datech **X** (nezávisle proměnná, prediktor, dimenze n x p) a **Y** (závisle proměnná, odezva, dimenze n x q) pro predikci odezvy na základě daných hodnot prediktoru. Predikce se pak provede pomocí funkce NNPREDICT. Úkolem neuronové sítě je nalézt analytický vztah $\mathbf{y} = G(\mathbf{x}) + \varepsilon$ mezi odpovídajícími řádky matic (dat) **X** a **Y** tak, aby minimalizoval součet kvadratických norem $\| \mathbf{e}_i \|$, $\mathbf{e}_i = \mathbf{y}_i - G(\mathbf{x}_i)$, kde $\mathbf{x}_i$ je i-tý řádek matice dat **X** a $\mathbf{y}_i$ je i-tý řádek matice dat **Y**. $G()$ je model (funkce, neuronová síť), který zobrazuje body z $\mathbf{R}^p$ do $\mathbf{R}^q$. Tedy pro zadaný vektor **x** s p prvky je $\mathbf{y}=G(\mathbf{x})$ predikcí vektoru odezvy **y** s q prvky při daných hodnotách prediktoru **x**. Natrénovaný model (neuronová síť) má strukturu seznamu, kterou lze přiřadit do proměnné a použít jako argument funkce NNPREDICT. Optimalizace (trénování) parametrů neuronové sítě začíná z náhodných startovacích hodnot, proto se mohou natrénované sítě i pro stejná trénovací data lišit, i když predikované hodnoty budou velmi podobné, nebo stejné. Samotná optimalizace modelu $G()$ probíhá iterativně s počtem iterací typicky v řádu stovek až tisíců, podle složitosti modelu. Model má strukturu m vrstev s k_1, … k_m neurony. Počet vrstev a počet neuronů v jednotlivých vrstvách je dán argumentem LAYERS. Doporučený počet vrstev je 1, 2 nebo 3, doporučený počet neuronů je 1 až 20. Tyto hodnoty jsou ovšem orientační, závislé na počtu a struktuře dat a na zkušenosti, požadavcích a potřebách uživatele. Další podrobnosti lze nalézt v uživatelském manuálu QCExpert.

Povinné argumenty

X: Numerický vektor, nebo matice, hodnoty prediktoru. Počet sloupců odpovídá počtu nezávisle proměnných.

Y: Numerický vektor, nebo matice, hodnoty odezvy. Počet sloupců odpovídá počtu závisle proměnných.

Nepovinné argumenty

XNAMES: Vektor řetězců stejné délky jako počet sloupců X. Názvy sloupců X, tyto řetězce se použijí na označení proměnných v grafech. Např. xnames= "A":"D"

YNAMES: Vektor řetězců stejné délky jako počet sloupců Y. Názvy sloupců Y, tyto řetězce se použijí na označení proměnných v grafech.

LAYERS: Celočíselný kladný numerický vektor udávající počet neuronů v jednotlivých vrstvách neuronové sítě. Počet prvků tedy definuje počet vrstev sítě. Implicitní hodnota je dvouprvkový vektor vec(2,3), tedy dvouvrstvá síť s 2 neurony v první vrstvě a 3 neurony ve druhé vrstvě.

MODELFILE: Řetězec, platný název souboru včetně cesty, do kterého se má uložit natrénovaný model. Tento soubor lze pak využít k predikci funkcí NNPREDICT. Není-li tento argument zadán, žádný soubor se neukládá, pro predikci lze použít výsledek funkce NNLEARN.

ITERATIONS: Kladné celé číslo. Počet iterací optimalizace. Implicitně je ITERATIONS=1000. Obvyklý vhodný počet iterací bývá mezi 1000 až 20000 (podle složitosti úlohy a časových možností.

USEFORLEARN: Reálné číslo menší nebo rovno 100. Podíl řádků v procentech, které se použijí pro trénování. Musí být voleno tak, aby pro trénování sítě zbylo dost dat. Obvykle se volí hodnoty mezi 70 a 90. Chceme-li použít všechna data, tento argument nemusíme zadávat, implicitní hodnota je 100.

EXPONENT: Reálné číslo větší než 1. Exponent Lp-normy. Pro standardní metodu nejmenších čtverců je roven 2 (implicitní hodnota). Je-li menší než 2, např. 1.5, je výsledný model robustnější vůči vybočujícím hodnotám a hrubým chybám.

ALPHA: Reálné číslo mezi 0 a 1. Hladina významnosti, implicitní hodnota je 0.05

MOMENTUM: Reálné číslo. Parametr optimalizačního algoritmu související s tlumením kroku, implicitně 0.9.

LEARNRATE: Reálné číslo. Parametr optimalizačního algoritmu související s velikostí kroku, implicitně 0.1.

IDENTERROR: Reálné číslo. Relativní maximální chyba pro ukončení trénování.

MEANERR: Celé číslo 0, nebo 1. Logická hodnota. Mají se do výsledkového seznamu ukládat hodnoty středních chyb během optimalizace?

RESIDUALS: Celé číslo 0, nebo 1. Logická hodnota. Mají se do výsledkového seznamu ukládat hodnoty reziduí?

GRNET: Celé číslo 0, nebo 1. Logická hodnota. Má se vytvořit graf struktury neuronové sítě?

GRPREDICT: Celé číslo 0, nebo 1. Logická hodnota. Má se do výsledkového seznamu zahrnout matice predikovaných hodnot pro zadané hodnoty prediktoru X?

BESTMODEL: Má se použít model s minimální střední chybou nalezený během trénování (1), nebo poslední nalezený model po skončení trénování (0)?

Struktura výsledného seznamu

FStat: Číslo. Hodnota F-statistiky použitelná pro porovnání modelů.

Layers: Číselný vektor, počty neuronů v jednotlivých vrstvách, včetně vstupní a výstupní vrstvy prediktorů a odezvy. Tedy první hodnota je počet sloupců X a poslední hodnota je počet sloupců Y.

ModelType: Textový řetězec, typ neuronové sítě (NN: statická neuronový síť)

Prediction: Numerický vektor nebo matice, predikované hodnoty pro prediktor X.

PValue: Číslo. p-hodnota pro F-statistiku FStat. Je-li menší, než 0.05, lze považovat model za statisticky významný.

RSS: Číslo. Reziduální součet čtverců.

UsedForLearn : 20 x 1: Logický vektor nul a jedniček udávající které řádky byly vybrány pro trénování sítě. Je-li vynechán argument USEFORLEARN, nebo je-li zadáno USEFORLEARN=100, jsou v tomto vektoru samé jedničky. Nula znamená, že příslušný řádek dat X, Y byl při učení vynechán. Tento vektor může pak být použit jako logický index [[ ]].

ValueMax: 2 x 1

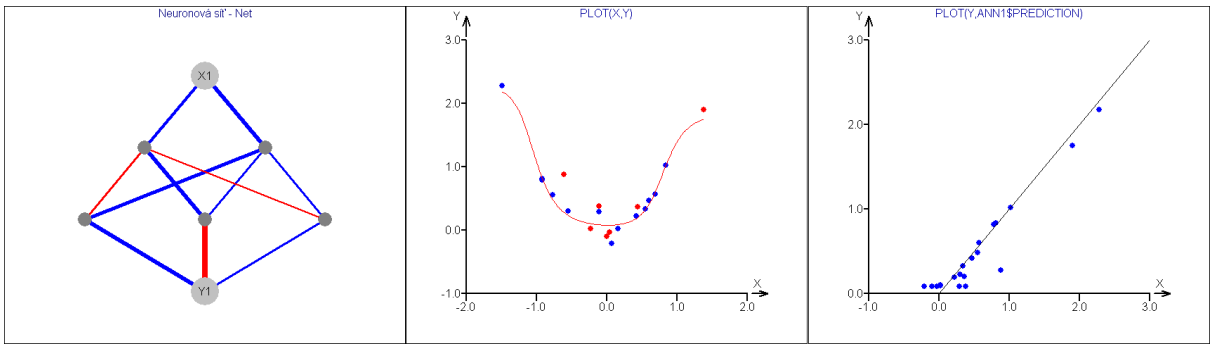
ValueMin: 2 x 1

Weights: Reálná matice vah neuronové sítě. První řádek odpovídá spojnicím vycházejícím nahoru z prvního neuronu první skryté vrstvy, druhý řádek odpovídá spojnicím vycházejícím nahoru z druhého neuronu první skryté vrstvy, atd.

WeightsSV: Reálný vektor, hodnoty vah neuronové sítě uspořádané do jednoho sloupce.

Příklad

```
x=normalr(20)
y=x^2+normalr(20)/5
graphsheet(cols=3)
@ann1=NNlearn(x,y,layers=vec(2,3),iterations=5000,
grnet=1,useforlearn=70)
x1=seq(min(x),max(x),count=200)
y1=NNPredict(x1,model=ann1)
plot(x,y)
@plotadd(x[[not(ANN1$UsedForLearn)]],
y[[not(ANN1$UsedForLearn)]], color=3);
plotadd(x1,Y1$Prediction,type="line",color=3)
plot(y,ANN1$Prediction)
lineadd(a=0,b=1,color=4)
```



Viz také

NNPREDICT, NNTIMELEARN

NNPREDICT(X, MODELFILE=|Model= [, FORECAST=, ALPHA=])

Predict with Neural Network. Predikce pomocí neuronové sítě.

Predikce hodnot odezvy pro zadaný vektor nebo matici prediktoru podle modelu vytvořeného funkcí NNLEARN, nebo NNTIMELEARN. Model může být uložen v proměnné, nebo v souboru na disku. V případě časové řady může NNPREDICT vypočítat i časovou předpověď (forecast). Tato funkce pozná, který typ modelu byl zadán a podle toho vrací výslednou strukturu seznamu s proměnnou „Forecast“ (pro model typu AR, nebo DIFF z funkce „nntimelearn“), nebo bez ní (pro model typu NN z funkce „nnlearn“).

Povinné argumenty

X: Číselný vektor, prediktor se stejným počtem sloupců jako měl prediktor při tvorbě modelu. V případě, kdy ModelType je AR, nebo DIFF (časová řada), je X vektor časové řady délky alespoň MODELDEPTH+1, viz funkci NNTIMELEARN.

MODELFILE | MODEL: Nutno zadat právě jeden z těchto dvou argumentů. MODELFILE je textový řetězec, platný název souboru včetně cesty, ze kterého se má přečíst natrénovaný model. MODEL je textový řetězec obsahující název proměnné, v níž je model uložen. Obsah této proměnné je výsledkem funkce NNLEARN, nebo NNTIMELEARN.

Nepovinné argumenty

FORECAST: Celé kladné číslo. V případě, kdy typ modelu je AR nebo DIFF (časová řada), určuje počet předpovědí do budoucnosti.

ALPHA: Hladina významnosti.

Struktura výsledného seznamu

Forecast : Pouze pro model časové řady typu AR, nebo DIFF. Vektor předpověděných hodnot pro časovou řadu. Délka vektoru je rovna argumentu FORECAST.

Prediction : Pro model NN (z funkce NNLEARN) vektor nebo matice s počtem sloupců shodným s počtem sloupců odezvy Y, na níž byl model natrénován a počtem řádků shodným s argumentem X. Pro model časové řady (z funkce NNTIMELEARN) vektor délky N – R + F, kde N je délka vektoru X, R je hodnota argumentu MODELDEPTH při volání funkce NNTIMELEARN a F je délka předpovědi, tedy hodnota argumentu FORECAST. Tento vektor pak obsahuje predikci pro naměřená data a pro předpověď.

Příklad

viz NNLEARN, NNTIMELEARN.

Viz také

NNLEARN, NNTIMELEARN

NNTIMELEARN(X [,IDENT=, MODELTYPE=AR|DIFF, MODELDEPTH=, LAYERS=, MODELFILE=, ITERATIONS=, EXPONENT=, ALPHA=, MOMENTUM=, LEARNRATE=, IDENTERROR=, MEANERR=0|1, RESIDUALS=1|0, GRNET=0|1, GRPREDICT=0|1], BESTMODEL=0|1])

Learn Time Series Neural Network. Natrénování časové neuronové sítě

Natrénuje se neuronová síť modelující jednorozměrnou časovou řadu. Časová řada je dána hodnotami sledované veličiny v pravidelných (stejně dlouhých) intervalech. Tyto hodnoty jsou zadány jako vektor X. Jsou k dispozici dva modely: AR: autoregresní model, který modeluje hodnoty x_i a DIFF: diferenční model, který modeluje diference $\Delta x_i = x_i - x_{i-1}$. Pomocí natrénovaného modelu lze pak predikovat již naměřené hodnoty (prediction) a také (pokud neuronová síť nalezla model, kterým se proces reprezentovaný časovou řadou řídí) předpovídat budoucí hodnoty, které ještě nebyly naměřeny. Obecně bývá vhodnější použít pro stacionární řady typ AR a pro nestacionární řady s trendem apod. typ DIFF. Natrénovaný model (neuronová síť) má strukturu seznamu, kterou lze přiřadit do proměnné a použít jako argument funkce NNPREDICT. Optimalizace (trénování) parametrů neuronové sítě začíná z náhodných startovacích hodnot, proto se mohou

natrénované sítě i pro stejná trénovací data lišit, i když predikované hodnoty budou velmi podobné, nebo stejné. Samotná optimalizace modelu $G()$ probíhá iterativně s počtem iterací typicky v řádu stovek až tisíců, podle složitosti modelu. Model má strukturu m vrstev s k_1, … k_m neurony. Počet vrstev a počet neuronů v jednotlivých vrstvách je dán argumentem LAYERS. Doporučený počet vrstev je 1, 2 nebo 3, doporučený počet neuronů je 1 až 20. Tyto hodnoty jsou ovšem orientační, závislé na počtu a struktuře dat a na zkušenosti, požadavcích a potřebách uživatele. Další podrobnosti lze nalézt v uživatelském manuálu QCExpert.

Povinné argumenty

X: Numerický vektor, hodnoty pravidelné časové (či jiné) řady. Předpokládají se stejné časové intervaly mezi jednotlivými hodnotami a samotná časová osa se nebere v úvahu.

Nepovinné argumenty

IDENT: Dvouprvkový vektor obsahující dva textové řetězce. Pokud se zpracovává databáze systému QCEDataCenter, ukládá do souboru s modelem informaci o tabulce a sloupci, který se má použít při automatické předpovědi v databázi QCEDataCenter. První prvek je název tabulky databáze FDB ve tvaru "TABLE=*název_tabulky*", druhý prvek je název sloupce této tabulky ve tvaru "FIELD=*název sloupce*".

MODELTYPE: AR, nebo DIFF bez uvozovek. Typ modelu. AR: autoregresní model $x_i = G(x_{i-1}, x_{i-2}, .. x_{i-R})$, kde R je hloubka modelu daná argumentem MODELDEPTH. DIFF: diferenční model $\Delta x_i = G(\Delta x_{i-1}, \Delta x_{i-2}, .. \Delta x_{i-R})$, kde R je hloubka modelu daná argumentem MODELDEPTH a Δx_i je diference $x_i - x_{i-1}$.

MODELDEPTH: Celé kladné číslo, hloubka modelu (řád modelu) R, počet po sobe jdoucích hodnot řady X, z nichž se počítá následující hodnota, obvykle se volí podle povahy procesu a dat hodnoty (velmi orientačně) mezi 3 – 20.

LAYERS: Celočíselný kladný numerický vektor udávající počet neuronů v jednotlivých vrstvách neuronové sítě. Počet prvků tedy definuje počet vrstev sítě. Implicitní hodnota je dvouprvkový vektor vec(2,3), tedy dvouvrstvá síť s 2 neurony v první vrstvě a 3 neurony ve druhé vrstvě nepočítaje proměnné.

MODELFILE: Řetězec, platný název souboru včetně cesty, do kterého se má uložit natrénovaný model. Tento soubor lze pak využít k predikci funkcí NNPREDICT. Není-li tento argument zadán, žádný soubor se neukládá, pro predikci lze použít výsledek funkce NNLEARN.

ITERATIONS: Kladné celé číslo. Počet iterací optimalizace. Implicitně je ITERATIONS=1000. Obvyklý vhodný počet iterací bývá mezi 1000 až 20000 (podle složitosti úlohy a časových možností.

USEFORTEACH: Reálné číslo menší nebo rovno 100. Podíl řádků v procentech, které se použijí pro trénování. Musí být voleno tak, aby pro trénování sítě zbylo dost dat. Obvykle se volí hodnoty mezi 70 a 90. Chceme-li použít všechna data, tento argument nemusíme zadávat, implicitní hodnota je 100.

EXPONENT: Reálné číslo větší než 1. Exponent Lp-normy. Pro standardní metodu nejmenších čtverců je roven 2 (implicitní hodnota). Je-li menší než 2, např. 1.5, je výsledný model robustnější vůči vybočujícím hodnotám a hrubým chybám.

ALPHA: Reálné číslo mezi 0 a 1. Hladina významnosti, implicitní hodnota je 0.05

MOMENTUM: Reálné číslo. Parametr optimalizačního algoritmu související s tlumením kroku, implicitně 0.9.

LEARNRATE: Reálné číslo. Parametr optimalizačního algoritmu související s velikostí kroku, implicitně 0.1.

IDENTERROR: Reálné číslo. Relativní maximální chyba pro ukončení trénování.

MEANERR: Celé číslo 0, nebo 1. Logická hodnota. Mají se do výsledkového seznamu ukládat hodnoty středních chyb během optimalizace?

RESIDUALS: Celé číslo 0, nebo 1. Logická hodnota. Mají se do výsledkového seznamu ukládat hodnoty reziduí?

GRNET: Celé číslo 0, nebo 1. Logická hodnota. Má se vytvořit graf struktury neuronové sítě?

GRPREDICT: Celé číslo 0, nebo 1. Logická hodnota. Má se do výsledkového seznamu zahrnout matice predikovaných hodnot pro zadané hodnoty prediktoru X?

BESTMODEL: Má se použít model s minimální střední chybou nalezený během trénování (1), nebo poslední nalezený model po skončení trénování (0)?

Struktura výsledného seznamu

FStat: Číslo. Hodnota F-statistiky použitelná pro porovnání modelů.

Layers: Číselný vektor, počty neuronů v jednotlivých vrstvách, včetně vstupní a výstupní vrstvy prediktorů a odezvy. Tedy první hodnota je počet sloupců X a poslední hodnota je počet sloupců Y.

ModelType: Textový řetězec, typ neuronové sítě (AR: Autoregresní časová řada, DIFF: Diferenční časová řada)

Prediction: Numerický vektor nebo matice, predikované hodnoty pro prediktor X.

PValue: Číslo. p-hodnota pro F-statistiku FStat. Je-li menší, než 0.05, lze považovat model za statisticky významný.

RSS: Číslo. Reziduální součet čtverců.

ValueMax: 2 x 1

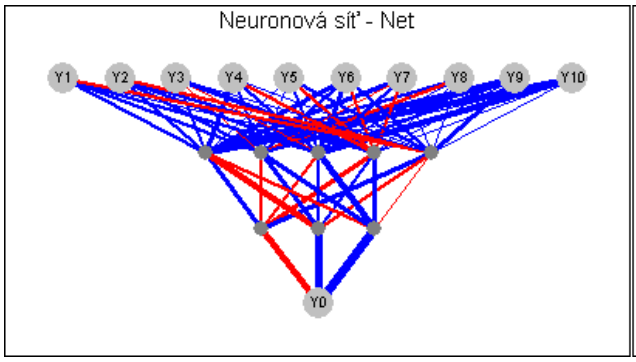
ValueMin: 2 x 1

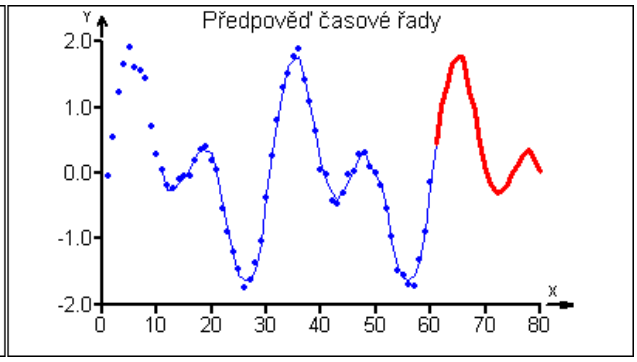
Weights: Reálná matice vah neuronové sítě. První řádek odpovídá spojnicím vycházejícím nahoru z prvního neuronu první skryté vrstvy, druhý řádek odpovídá spojnicím vycházejícím nahoru z druhého neuronu první skryté vrstvy, atd.

WeightsSV: Reálný vektor, hodnoty vah neuronové sítě uspořádané do jednoho sloupce.

Příklad

```
//Simulace a předpověď časové řady:
N=60
NF=20
dpt=10
x=seq(0,100,count=N)
y=sin(x/8)+sin(x/4)+normalr(N)/10
graphsheet(cols=2)

//Natrénování neuronové sítě:
ann2=nntimelearn(y,modeldepth=dpt,layers=vec(5,3),iterations
=5000,grnet=1)

//Predikce, předpověď dvaceti hodnot (NF):
y1=NNPredict(y,model=ann2,forecast=NF)
plot(1:N,y,main="Předpověď časové řady")
plotadd((1+dpt):(N+NF),vec(ann2$Prediction,Y1$Forecast),type
="line")
plotadd((N+1):(N+NF),Y1$Forecast,type="line",color=3,width=3)
```





Viz také

`NNLEARN, NNPREDICT`

NORM(X)

Eucleidian norm of a vector or matrix. Eukleidovská norma vektoru nebo matice.

Eukleidovská norma ||X||, tedy odmocnina ze součtu čtverců všech prvků vektoru nebo matice.

Norma vektoru: $\|\mathbf{x}\| = \sum_{i=1}^{n} x_i^2$; norma matice: $\|\mathbf{X}\| = \sum_{i=1}^{n}\sum_{j=1}^{m} x_{ij}^2$.

Povinné argumenty

X: číselná matice nebo vektor

Příklad

```
>norm(vec(1,1))
1.4142135623731

>norm(unit(9))
3
```

Viz také

`DET, MATRIX, VECTOR`

NORMALD(X [, MEAN=0] [, SDEV=1])

Normal density function. Hustota pravděpodobnosti normálního rozdělení.

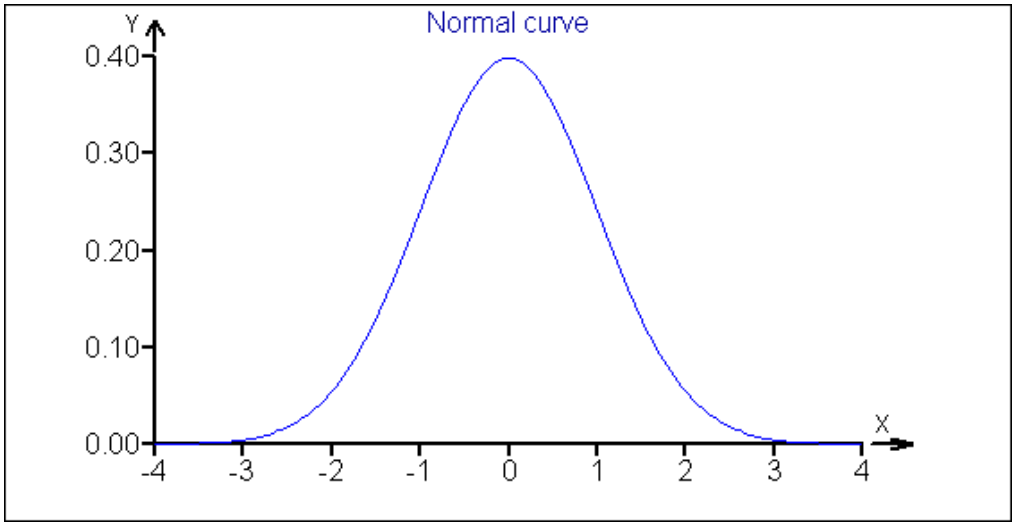
Funkce vrací hodnotu hustoty pravděpodobnosti normálního rozdělení pro kvantil X.

Povinné argumenty

X: kvantil rozdělení, reálné číslo, nebo vektor.
Nepovinné argumenty:
MEAN: střední hodnota, reálné číslo, implicitně MEAN=0.
SDEV: směrodatná odchylka, kladné reálné číslo, implicitně SDEV=1.

Příklad

```
x=seq(-4,4,count=200)
plot(x,normald(x),type="line" ,main="Normal curve")
```



Viz také

NORMALP, NORMALQ, NORMALR

NORMALP(X [, MEAN=0] [, SDEV=1])

Normal distribution function (cumulative probability function). Distribuční funkce normálního rozdělení.

Funkce vrací hodnotu distribuční funkce normálního rozdělení pro kvantil X.

Povinné argumenty

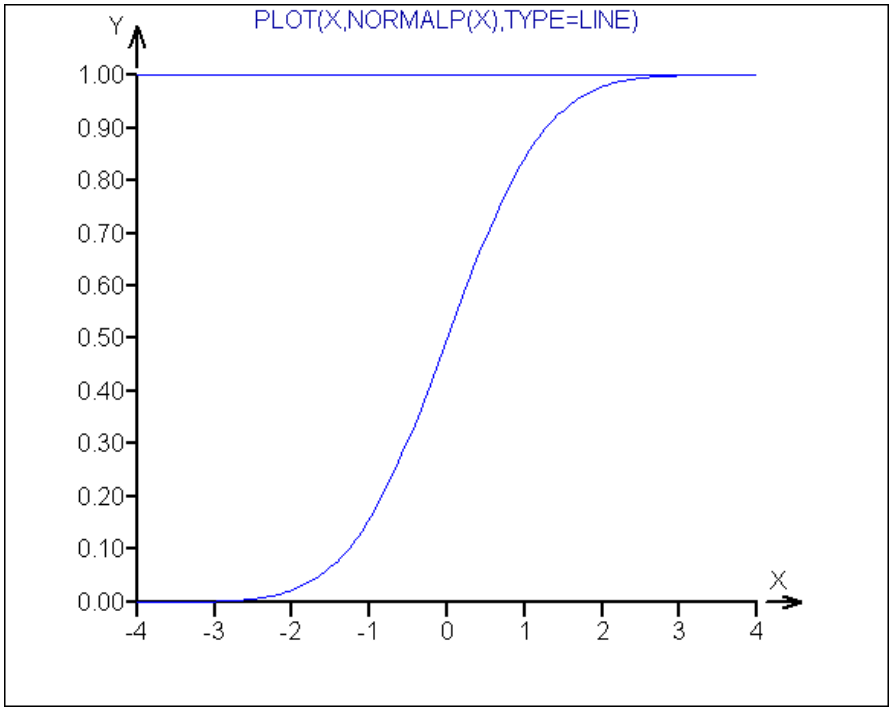
X: kvantil rozdělení, reálné číslo, nebo vektor

Nepovinné argumenty:

MEAN: střední hodnota, reálné číslo, implicitně MEAN=0, SDEV: směrodatná odchylka, kladné reálné číslo, implicitně SDEV=1.

Příklad

```
>100*normalp(vec(-3,3))
0.13498980316301
99.865010196837

>x=seq(-4,4,count=200)
>plot(x,normalp(x),type=line)
>lineadd(h=1)
```



Viz také

NORMALD, NORMALQ, NORMALR

NORMALQ(P [, MEAN=] [, SDEV=])

Normal quantile function. Kvantilová funkce normálního rozdělení.

Funkce vrací hodnotu kvantilové funkce normálního rozdělení pro pravděpodobnost P.

Povinné argumenty

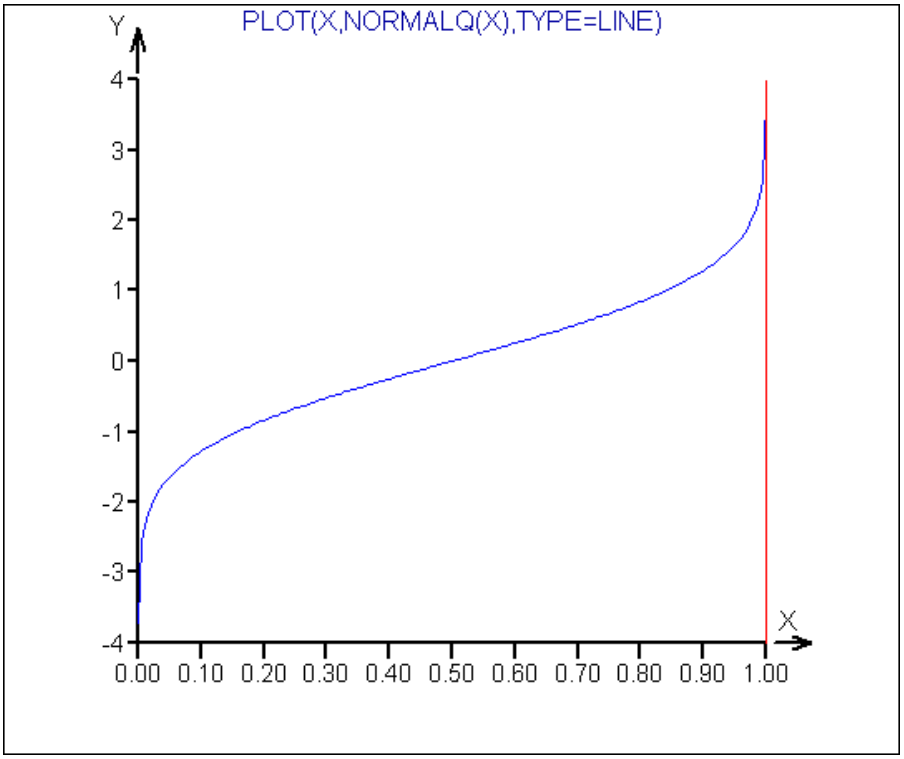
P, pravděpodobnost, reálné číslo mezi 0 a 1, nebo vektor

Nepovinné argumenty

MEAN: střední hodnota, reálné číslo, implicitně MEAN=0, SDEV: směrodatná odchylka, kladné reálné číslo, implicitně SDEV=1.

Příklad

```
>normalq(vec(0.025,0.975))
-1.95996399862641
1.95996399862641

x=seq(0.0001,0.9999,count=200)
plot(x,normalq(x),type="line")
lineadd(v=1,color=3)
```



Viz také

NORMALP, NORMALD, NORMALR

NORMALR(N [, MEAN=X] [, SDEV=S])

Normal random number. Náhodné číslo z normálního rozdělení.

Funkce vrací výběr N náhodných čísel z normálního rozdělení.

Povinné argumenty

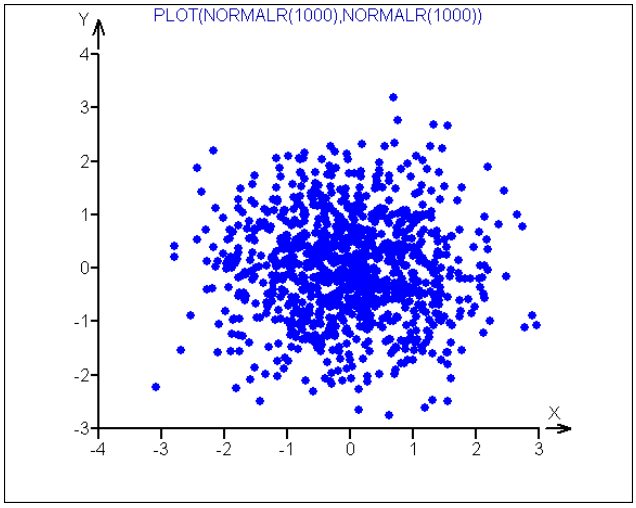
N, počet náhodných čísel (rozsah výběru), kladné celé číslo.

Nepovinné argumenty

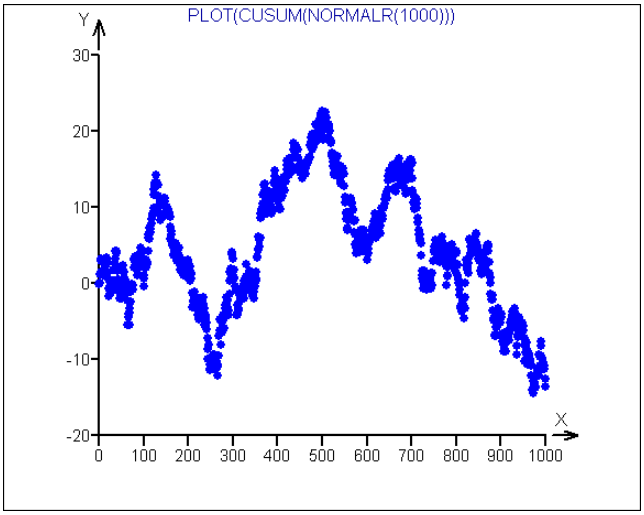
MEAN=X: střední hodnota, reálné číslo, implicitně MEAN=0, SDEV=S: směrodatná odchylka, kladné reálné číslo, implicitně SDEV=1.

Příklad

```
>normalr(5)
-2.07503566751396
-0.323798724462547
-1.0041748046037
0.386989024304513
0.54385775513808

>plot(normalr(1000),normalr(1000))
```




```
// náhodné kráčení (nestacionární proces):
>plot(cusum(normalr(1000)))
```



Viz také

NORMALP, NORMALQ, NORMALD, RANDOM, RND

NROWS (X)

Number of rows. Počet řádků.

Počet řádků vektoru nebo matice.

Povinné argumenty

X: vektor nebo matice

Příklad

```
>ncols(1:4)
4
>ncols(transp(1:4))
1
>ncols(unit(5))
```

```
5
```

Viz také

NCOLS, DIM, COUNT

ONES (N)

Vector of ones. Jedničkový vektor.

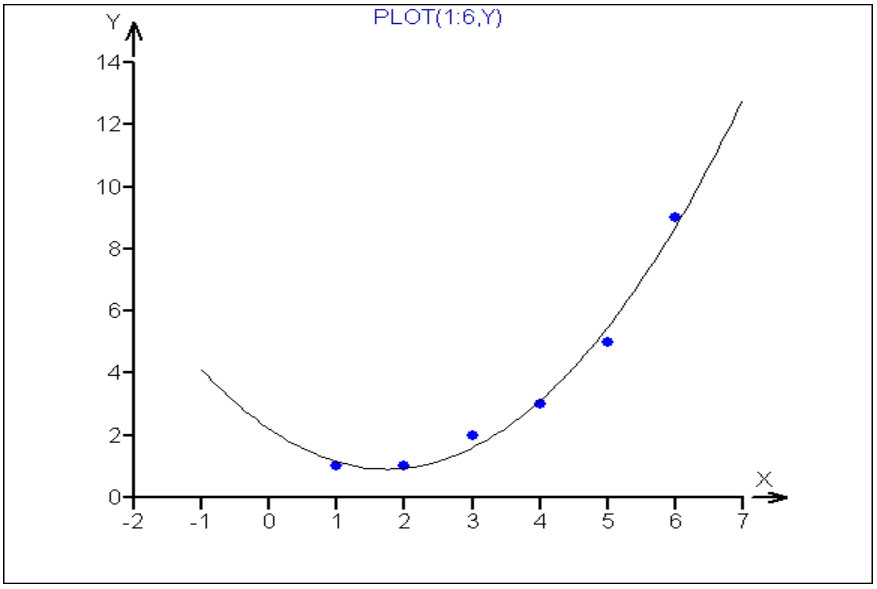
Funkce vrací vektor jedniček délky N.

Povinné argumenty

N: Délka jedničkového vektoru. Kladné celé číslo.

Příklad

```
>ones(5)
1
1
1
1
1

// Kvadratická regrese, kde vektor ones(6)
// reprezentuje sloupec pro absolutní člen modelu
// y=a1+a2*x+a3*x^2
>x=bind(bind(ones(6),1:6),(1:6)^2)
>y=vec(1,1,2,3,5,9)
>a=inv(transp(x)#x)#transp(x)#y

// Bodové odhady koeficientů a1, a2, a3 regresního modelu:
>round(a,3)
2.2
-1.486
0.429

>plot(1:6,y)
>x1=seq(-1,7,count=100)
>plotadd(x1,a[1]+a[2]*x1+a[3]*x1^2,type="line",color=4)
```



Viz také

REP, UNIT, VEC, MATRIX, []

OR(X1, X2)

OR, Logical sum, Logický součet

Logický součet numerických pravdivostních hodnot X1 a X2. Číslo 0 se považuje za nepravdu (false), číslo různé od 0 (typicky 1) se považuje za pravdu (true). Hodnoty výsledku uvádí pravdivostní tabulka:

X1	X2	OR(X1,X2)
0	0	0
0	1	1
1	0	1
1	1	1

Povinné argumenty

X1, X2: číslo, numerický vektor, nebo matice obsahující pravdivostní hodnoty.

Příklad

```
>or(ge(5,3),ge(1,2))
1

// Výběr prvků Y splňující zadané podmínky
>x=sample(0:1,10,repl=1)
>y=normalr(10)
>y[[ or (not(zero(x)) , ge(y,1)) ]]
-0.0933326548934273
-0.168102567972266
1.67348898570269
-0.845369170316211
```

Viz také

AND, NOT, XOR, GT, LT, GE, LE, EQ, NE

ORDER(I,X)

Order of vector elements. Pořadí prvků vektoru.

Pořadí prvků vektoru, nebo zvoleného sloupce matice, nebo vektoru. Argument I určuje sloupec matice (pro vektor je I=1), X je tříděná matice nebo vektor. Znaménko argumentu I určuje směr třídění (plus pro vzestupné pořadí, mínus pro sestupné pořadí). Je-li I vektor, třídí se řádky nejprve podle I[1]-tého sloupce, pak podle I[2]-tého sloupce, atd. matice X. Zápis `X[ORDER(1,X)]` dá stejný výsledek jako `SORT(1,X)` (`X` je sloupcový vektor).

Povinné argumenty

I je nenulové celé číslo nebo vektor určující sloupec, nebo sloupce, v podle nichž se má určit pořadí.

X je vektor nebo matice, z níž se má určit pořadí řádků.

Příklad

```
>x=vec(3,2,5,6,1)
>x[order(-1,x)] // Sestupně setříděné podle 1. sloupce
6
5
3
2
1

>x=vec(1,1,3,3,5)
>x=bind(x,vec(5,4,3,2,1))
>x=bind(x,vec(4,2,8,3,0))
>x // Nesetříděná matice x
1   5     4
1   4     2
3   3     8
3   2     3
5   1     0

>ii=order(vec(1,2),x)
>ii // řádkové indexy setříděné matice x podle 1. a 2.
sloupce
2
1
4
3
5

>x[ii,] // setříděná matice x
1   4     2
1   5     4
3   2     3
3   3     8
5   1     0
```

Viz také

SORT, MIN, MAX, SEQ

PARSE(S)

Parse and execute expression or command. Vyhodnocení a provedení výrazu nebo příkazu

Tento příkaz předá jazyku k provedení řetězec S. Může to být přiřazovací příkaz, nebo výraz s platnou syntaxí. Výsledkem je provedení příkazu, nebo hodnota výrazu.

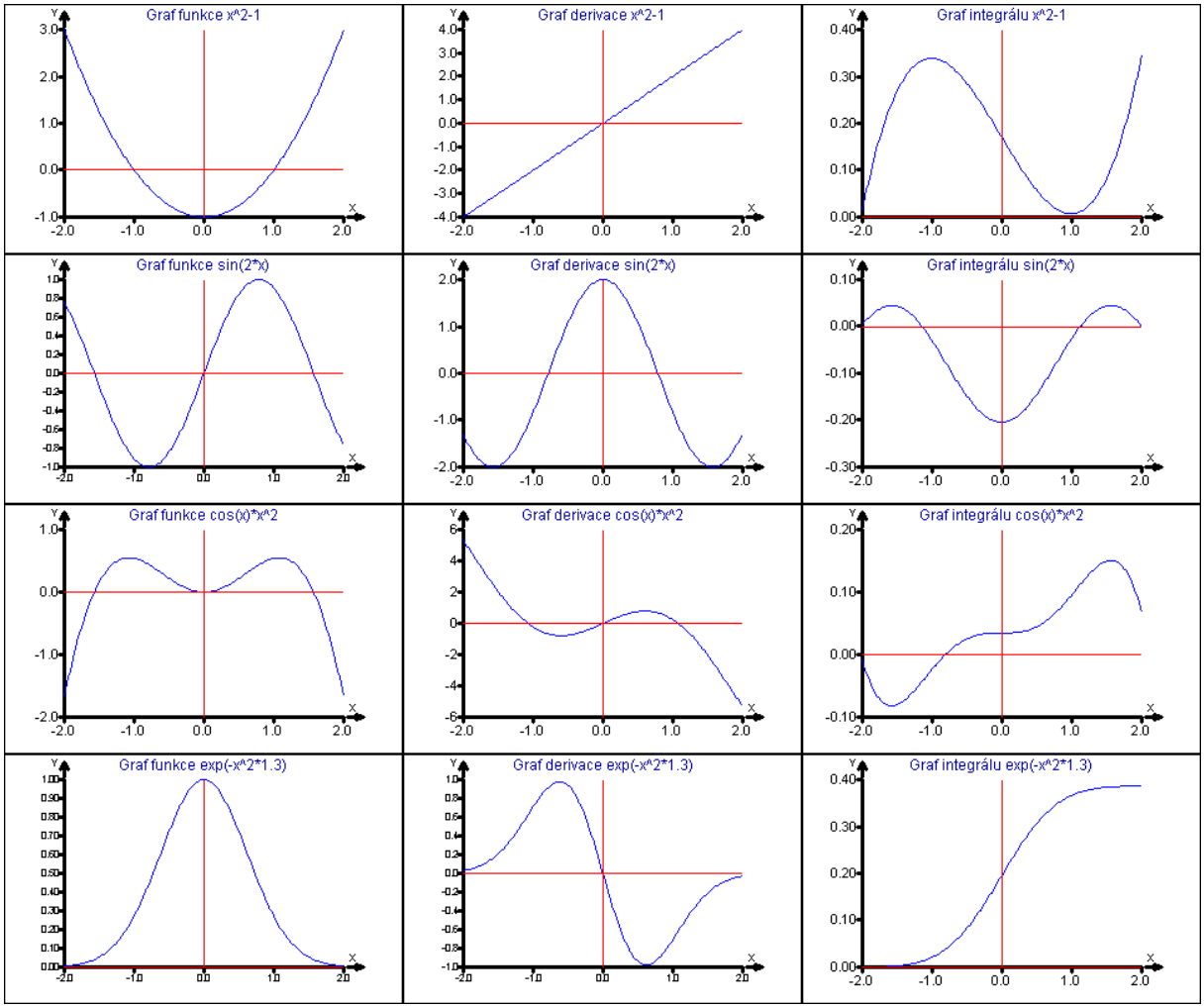
Povinné argumenty

S: Textový řetězec obsahující syntakticky platný výraz nebo příkaz.

Příklad

```
>parse("5/7")
0.714285714285714

// Dva zápisy s týmž výsledkem:
>a=parse("5/7")
>parse("a=5/7")

>v1="A"
>ex="sqrt(2)"
>parse(v1+"="+ex)
>a
1.4142135623731

// Použití PARSE ve spojení s voláním uživatelské funkce
//*****************************************************
// Seznam funkcí, které se mají zobrazit
// včetně derivace a integrálu v daném intervalu <a1, a2>:
@funkce=vec( "x^2-1", "sin(2*x)", "cos(x)*x^2",
"exp(-x^2*1.3)" );
a1=-1.999
a2=2
nf=count(funkce)
graphsheet(cols=3)
for(i=1:nf)
{
z=plotfun(a1,a2,funkce[i])
}
//*****************************************************

// Funkce definovaná ve funkčním listu:
//-------------------------------------
function PlotFun(x1,x2,funx)
{
// Graf funkce
x=seq(x1,x2,count=200)
```

```
y=parse(funx)
plot(x,y,type="line",main="Graf funkce "+funx)
lineadd(h=0,v=0,color=3)

// Graf derivace funkce
dx=0.00001
x=x-dx
y1=parse(funx)
dy=(y-y1)/dx
plot(x,dy,type="line",main="Graf derivace "+funx)
lineadd(h=0,v=0,color=3)

// Graf (přibližného) integrálu funkce
y2=cusum(y/200)
plot(x,y2,type="line",main="Graf integrálu "+funx)
lineadd(h=0,v=0,color=3)
}
```



Viz také

FUNCTION, ASTEXT

PASTE(S1 [, S2, S3, ...] [, SEPARATOR=""])

Paste string arguments into one string. Spojí textové argumenty do jediného řetězce.

Spojí jednotlivé textové řetězce, nebo číselné hodnoty ve vektoru, matici, nebo několika objektech do jediného textového řetězce. Je-li například S1 vektor textových řetezců, je výsledkem jediný řetězec vzniklý spojením všech řetězců obsažených v S1. Jednotlivé řetězce mohou být odděleny oddělovacím znakem definovaným v argumentu SEPARATOR. Obsahují-li argumenty numerické hodnoty, převedou se na řetězce.

Povinné argumenty

S1: Textový řetězec nebo numerická hodnota, vektor, nebo matice.

Nepovinné argumenty

S2, S3, …: Další textové nebo numerické objekty, jejichž obsah bude připojen do výsledného řetězce.

SEPARATOR: Textový řetězec, který se vloží mezi spojované hodnoty. Implicitní hodnota je prázdný znak ("").

Příklad

```
// Náhodný text
N=30 //Počet slov
M=6  // Max délka slova
ini(st)
lex= letters("a", 1:26)
for(i=1,N)
{  K=sample(1:M,1)
   st=vec( st,substr(lex,sample(1:26, K, repl=1)) )
}
paste("The",st,separator=" ")

"The ctdz zmqse naayl bdpi ffu uvocsf bmr qgga zqygpx uoxxn
hj roydcf s hotjy rs zaxupr jy hzx zszn c vnt zmeov i rhkrlf
ddbi jbo ttoq t wanbj ewmgw"

// ********************************************
// Řetězec 50 náhodných cifer 0-9:
r=sample(0:9,50,repl=1)
paste(r)
"10109683931479993351010733131367766071022965181430"
```

Viz také

VEC, ASTEXT, +

PAUSE(T)

Pause execution for T seconds. Zdržení provádění skriptu na T sekund.

Tento příkaz pozdrží provádění skriptu v daném místě na T sekund. Lze jej v kombinaci s příkazem TRACEON využít při odlaďování programu. T může být desetinné číslo.

Povinné argumenty

T: Kladné reálné číslo, čas v sekundách, na který se má pozastavit provádění příkazů v daném místě skriptu.

Příklad

```
// Sledujte obsah proměnných A a I.
a=0
traceon
for(i=1,10)
{
pause(0.5)
a=a+i
pause(0.5)
}
```

Viz také

TRACEON, TRACEOFF, STOP

PDFBEGIN(FILE[,Margins=Vec(Left,Top,Right,Bottom), ORIENTATION=PORTRAIT|LANDSCAPE,TITLE=,AUTHOR=, SUBJECT=,KEYWORDS=])

Create new PDF document. Založí nový dokument PDF

Vytvoří soubor typu PDF s názvem FILE. Volitelně lze zadat okraje, orientaci dokumentu a systémovou hlavičku dokumentu obsahující titul, autora, předmět a klíčová slova. Vytvořený dokument bude obsahovat jednu prázdnou stranu A4. Do takto vytvořeného dokumentu lze zapisovat, dokud není uzavřen příkazem PDFEND. Hotový dokument je vždy nutné uzavřít, jinak se nemusí později správně zobrazit.

Povinné argumenty

FILE: Název souboru PDF včetně cesty.

Nepovinné argumenty

MARGINS: Vektor se čtyřmi prvky obsahující šířky levého, horního, pravého a spodního okraje stránky v milimetrech.

ORIENTATION: Řetězec udávající orientaci stránky, portrait (na výšku), nebo landscape (na šířku).

TITLE: Řetězec titulu dokumentu, který má být zapsán do PDF-hlavičky dokumentu (není to název souboru).

AUTHOR: Řetězec obsahující jméno autora, který má být zapsán do PDF-hlavičky dokumentu.

SUBJECT: Řetězec obsahující předmět dokumentu, který má být zapsán do PDF-hlavičky dokumentu.

KEYWORDS: Řetězec obsahující klíčová slova.

Příklad

```
// Nutno vytvořit adresář C:\temp:
@PDFBEGIN("c:\temp\Report.pdf",
margins=vec(20,40,20,40), orientation="portrait");
PDFTEXT(\n,\n,chr(sample(vec(65:90,rep(32,8)),4800,repl=1)))
PDFEND(launch=1)
```

Viz také

PDFEND, PDFFONT, PDFFOOTER, PDFHEADER, PDFIMAGE, PDFNEWPAGE, PDFPLOT, PDFTEXT, PDFTABLE, PRINT, EXPORTGRAPH

PDFEND ([LAUNCH=0|1])

Close created document. Uzavře hotový dokument

Po uzavření dokumentu nelze do něj již zapisovat, dokument lze otevřít zadáním argumentu LAUNCH=1.

Povinné argumenty

Nejsou.

Nepovinné argumenty

LAUNCH: Je-li hodnota parametru 1, dojde bezprostředně po uzavření dokumentu k jeho otevření v příslušném programu (např. PDF Viewer, Acrobat Reader), přičemž běh skriptu se nezastaví.

Příklad

```
PDFEND()
PDFEND(launch=1)
```

Viz také

PDFBEGIN, PDFFONT, PDFFOOTER, PDFHEADER, PDFIMAGE, PDFNEWPAGE, PDFPLOT, PDFTEXT, PDFTABLE, PRINT, EXPORTGRAPH

PDFFONT([NAME="Tahoma", SIZE=12, ITALIC=0|1,BOLD=0|1, UNDERLINE=0|1, COLOR=])

Sets font. Nastaví font.

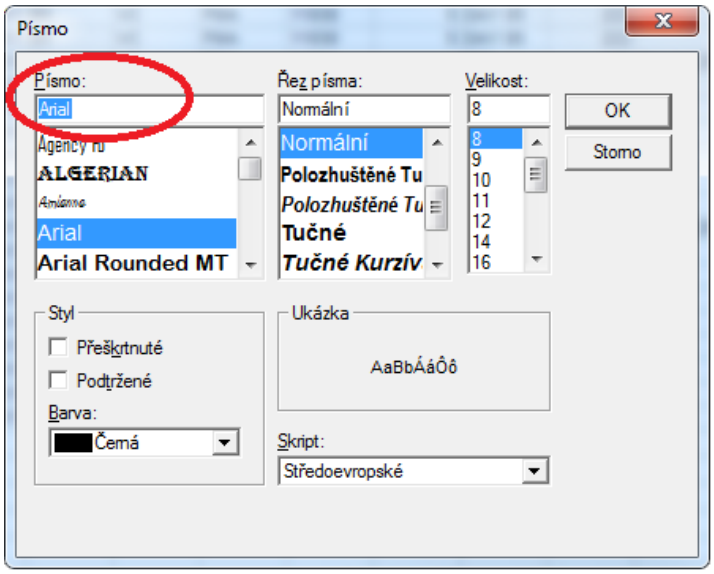
Nastavení fontu pro následující tisk do dokumentu. Nastavený font platí až do dalšího použití příkazu PDFFONT. Při dalším použití příkazu PDFFONT lze zadat jen ty argumenty, které chceme měnit, ostatní nastavení fontu zůstane zachováno.

Povinné argumenty

Nejsou.

Nepovinné argumenty

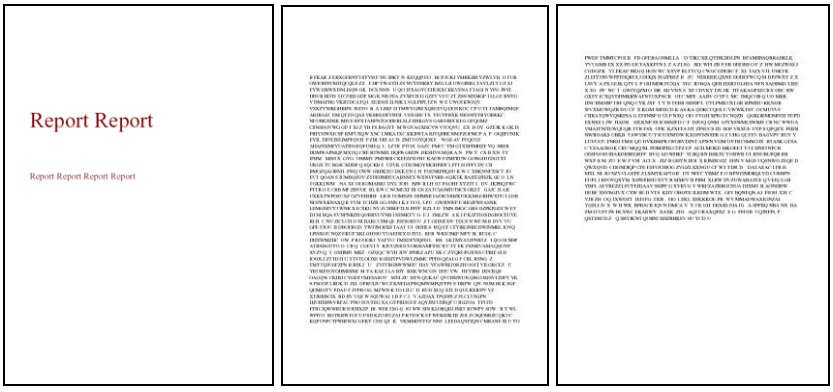
NAME: Řetězec obsahující název fontu. Přesný název fontu lze získat ze systémového dialogu nastavení fontu kterékoliv aplikace, například v QCExpertu: Formát: Písmo.



SIZE: Celé číslo, velikost fontu v bodech, implicitní hodnota 12.
ITALIC: Číslo 0, nebo 1, nastaví kurzívu, implicitní hodnota 1.
BOLD: Číslo 0, nebo 1, nastaví tučné písmo, implicitní hodnota 0.
UNDERLINE: Číslo 0, nebo 1, nastaví podtržení textu, implicitní hodnota 0.
COLOR: Celé číslo barvy textu, viz příklad u PLOTADD.

Příklad

```
@PDFBEGIN("c:\temp\Report.pdf",
margins=vec(20,40,20,40), orientation="portrait");
PDFFONT(NAME="Times New Roman",SIZE=48,COLOR=8)
PDFTEXT(\n,\n,"Report Report")
PDFFONT(NAME="Times New Roman",SIZE=20,COLOR=8)
PDFTEXT(\n,\n,\n,"Report Report Report Report")
PDFFONT(COLOR=4,SIZE=11)
PDFNewPage()
PDFTEXT(\n,\n,chr(sample(vec(65:90,rep(32,8)),4800,repl=1)))
PDFEND(launch=1)
```



Viz také

PDFBEGIN, PDFEND, PDFFOOTER, PDFHEADER, PDFIMAGE, PDFNEWPAGE, PDFPLOT, PDFTEXT, PDFTABLE, PRINT, EXPORTGRAPH

PDFFOOTER("text left","text right"[,LINE=0|1])

Define footer. Definice zápatí stránky.

Definuje zápatí stránky, které se bude opakovat až do konce dokumentu. Zápatí se vytvoří až od stránky, kdy je příkaz PDFFOOTER použit, je tedy možné nastavit zápatí až od určité stránky. Do zápatí lze vložit číslo stránky pomocí vyhrazeného řetězce "%d". Druhé použití znaků "%d" v rámci téhož řetězce zobrazí celkový počet stran dokumentu.

Povinné argumenty

Dva řetězce textu, který se umístí vlevo a vpravo v zápatí.

Nepovinné argumenty

LINE: Číslo 0, nebo 1. Zobrazení oddělovací linky zápatí.

Příklad

```
PDFBEGIN("c:\temp\Report.pdf")
PDFFOOTER("Date: "+strDate(0),"Strana: %d / %d",LINE=1)
```

Viz také

PDFBEGIN, PDFEND, PDFFONT, PDFHEADER, PDFIMAGE, PDFNEWPAGE, PDFPLOT, PDFTEXT, PDFTABLE, PRINT, EXPORTGRAPH

PDFHEADER("text vlevo","text vpravo"[,LINE=0|1])

Define header. Definice záhlaví stránky.

Definuje záhlaví stránky, které se bude opakovat až do konce dokumentu. Záhlaví se vytvoří až od stránky, kdy je příkaz PDFFOOTER použit, je tedy možné nastavit záhlaví až od určité stránky.

Povinné argumenty

Dva řetězce textu, který se umístí vlevo a vpravo v záhlaví.

Nepovinné argumenty

LINE: Číslo 0, nebo 1. Zobrazení oddělovací linky záhlaví.

Příklad

```
PDFBEGIN("c:\temp\Report.pdf")
PDFHEADER("Date: "+strDate(0),"-- Název firmy --",LINE=1)
```

Viz také

PDFBEGIN, PDFEND, PDFFONT, PDFFOOTER, PDFIMAGE, PDFNEWPAGE, PDFPLOT, PDFTEXT, PDFTABLE, PRINT, EXPORTGRAPH

PDFIMAGE(filename,[WIDTHMM=,ALIGN=LEFT|RIGHT|CENTER])

Place bitmap image from file. Umístění obrázku ze souboru.

Vloží do dokumentu obrázek ze souboru ve formátu JPG, GIF, BMP, nebo WMF. Lze nastavit velikost a pozici obrázku. Obrázek se umístí v dokumentu jako znak na samostatný řádek. Lze použít například pro umístění loga firmy do dokumentu.

Povinné argumenty

Řetězec obsahující cestu a název souboru s obrázkem.

Nepovinné argumenty

WIDTHMM: Numerická hodnota, vodorovný rozměr (šířka) vloženého obrázku v milimetrech. Výška obrázku respektuje původní poměr stran.

ALIGN: Řetězec určující zarovnání vloženého obrázku vlevo ("LEFT"), vpravo ("RIGHT"), nebo doprostřed ("CENTER").

Příklad

```
// Vytvoříme soubor JPG pomocí exportu:
plot(normalr(1000),main="Uncorrelated normal noise")
EXPORTGRAPH("C:\temp\FIG_1.jpg",resize=vec(800,480))

PDFBEGIN("c:\temp\Report.pdf")
PDFHEADER("Date: "+strDate(0),"-- Název firmy --",LINE=1)
PDFFOOTER("Prepared by: "+"J.B.","Strana: %d / %d",LINE=1)
PDFFONT(NAME="Times New Roman",SIZE=48,COLOR=8)
PDFTEXT(\n,\n,\n,\n,"Report Report")
PDFFONT(NAME="Times New Roman",SIZE=20,COLOR=8)
PDFTEXT(\n,\n,\n,"Report Report Report Report")
PDFFONT(COLOR=4,SIZE=11)
PDFNewPage()
PDFTEXT(\n,\n,chr(sample(vec(65:90,rep(32,8)),800,repl=1)))
PDFTEXT(\n,\n)
PDFIMAGE("C:\temp\FIG_1.jpg",align="Center")
PDFEND(launch=1)
```

Viz také

PDFBEGIN, PDFEND, PDFFONT, PDFFOOTER, PDFHEADER, PDFNEWPAGE, PDFPLOT, PDFTEXT, PDFTABLE, PRINT, EXPORTGRAPH

PDFNEWPAGE()

Begin new page. Nová stránka.

Založí novou prázdnou stránku dokumentu včetně případného dříve definovaného záhlaví a zápatí.

Povinné argumenty

Nejsou.

Nepovinné argumenty

Nejsou.

Příklad

```
PDFBEGIN("c:\temp\Report.pdf")
PDFFONT(name="Times New Roman", size=48)
PDFTEXT(\n,\n,\n,\n, "Weekly Report")
PDFNEWPAGE()
```

Viz také

PDFBEGIN, PDFEND, PDFFONT, PDFFOOTER, PDFHEADER, PDFIMAGE, PDFPLOT, PDFTEXT, PDFTABLE, PRINT, EXPORTGRAPH

PDFPLOT([SHEETNAME=CURRENT|"název listu", RESIZE=vec(width, height), WIDTHMM=, ALIGN=LEFT|RIGHT|CENTER])

Place graphics from graphsheet. Vloží graf z grafického listu.

Tento příkaz vloží na aktuální pozici buď poslední nakreslený graf v grafickém listu, nebo obsah celého grafického listu. Před vložením lze obrázek přeformátovat do jiného rozlišení. Šířku vloženého obrázku lze nastavit, obrázek zachovává poměr stran. Umístění obrázku je dáno současnou pozicí na stránce, obrázek se vkládá „jako znak“.

Povinné argumenty

Nejsou.

Nepovinné argumenty

SHEETNAME: Řetězec, název grafického listu (podle názvu na záložce), který se má vložit do dokumentu. Prázdný znak "" vloží aktuální grafický list. Vynechá-li se argument SHEETNAME, vloží se pouze poslední nakreslený graf.

RESIZE: Číselný vektor délky 2. Šířka a délka bitové mapy, do níž se celý graf transformuje před tiskem. Tento parametr má vliv na relativní velikost písma a rozlišení grafu.

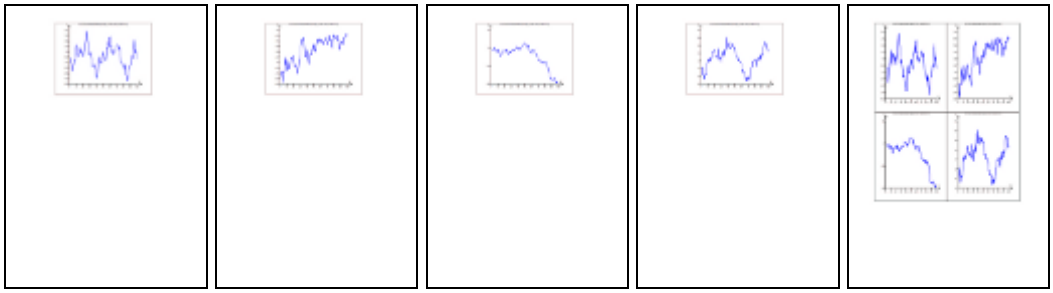
WIDTHMM: Číselná hodnota. Šířka grafu na stránce PDF v milimetrech. Poměr stran je dán argumentem RESIZE.

ALIGN: Textový řetězec "LEFT", "RIGHT", nebo "CENTER". Zarovnání obrázku. na řádku.

Příklad

```
// Do PDF se vloží každý graf zvlášť a nakonec všechny grafy
// dohromady
PDFBegin("C:\temp\Report.pdf",margins=vec(15,20,20,20))
graphsheet(cols=2)
for(i=1,4)
{
plot(cusum(normalr(100)),type="line",width=3)
pdfplot(align="center")
pdfnewpage()
```

```
}
pdfplot(sheetname="",resize=vec(600,600),widthmm=160)
pdfend(launch=1)
```



Viz také

PDFBEGIN, PDFEND, PDFFONT, PDFFOOTER, PDFHEADER, PDFIMAGE, PDFNEWPAGE, PDFTEXT, PDFTABLE, PRINT, EXPORTGRAPH

PDFTABLE(par1[,BORDER=1|0,HEADER=])

Create table from matrix or vector. Vytvoří tabulku z matice nebo vektoru.

Vytiskne zadanou matici nebo vektor v podobě tabulky. Formátování sloupců tabulky je automatické podle jeho obsahu s dodržením nastaveného fontu. Velikost tabulky lze ovlivnit volbou vhodného fontu, orientace stránky (ORIENTATION v příkazu PDFBEGIN), případně úpravou obsahu matice (zaokrouhlení, apod). Řádky lze zvýraznit vodorovnými linkami (argument BORDER). Zarovnání ve sloupci je vždy doleva. Jedním příkazem PDFTABLE se vytvoří pouze jedna tabulka z jediné matice nebo vektoru. Nevejde-li se tabulka na šířku strany, uřízne se zprava. Nevejde-li se na výšku strany, pokračuje se na nové straně se zopakovanou hlavičkou.

Povinné argumenty

Matice, nebo vektor, který se má vytisknout.

Nepovinné argumenty

BORDER: Číslo 0 nebo 1 udávající zda se mají oddělit řádky tabulky linkami.
HEADER: Vektor se stejným počtem

Příklad

```
PDFBegin("C:\trilobyte\Report.pdf",margins=vec(15,20,20,20))
a=matrix(round(normalr(64),4),ncols=8)
b=matrix(round(normalr(1000),8),ncols=10)
bheader="Column "+(1:10)
PDFFONT(NAME="Arial",SIZE=14,COLOR=4)
PDFTEXT("Table - 1")
PDFTABLE(a,border=1)
PDFTEXT(\n,"Table - 2")
PDFFONT(NAME="Arial",SIZE=7,COLOR=5)
PDFTABLE(b,header=bheader)
pdfend(launch=1)
```

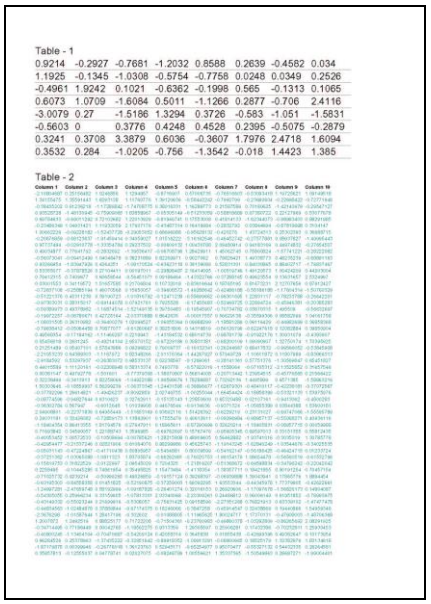
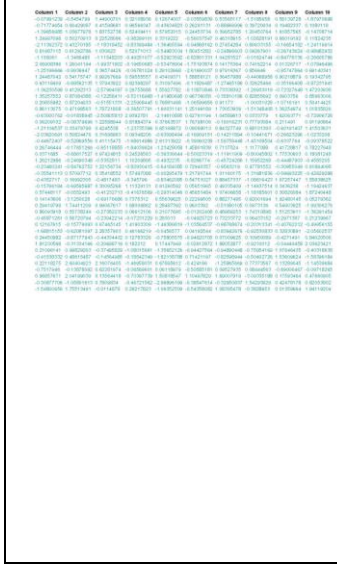


Table - 1

0.9214	-0.2927	-0.7681	-1.2032	0.8588	0.2639	-0.4582	0.034
1.1925	-0.1345	-1.0308	-0.5754	-0.7758	0.0248	0.0349	0.2526
-0.4961	1.9242	0.1021	-0.6362	-0.1998	0.565	-0.1313	0.1065
0.6073	1.0709	-1.6084	0.5011	-1.1266	0.2877	-0.706	2.4116
-3.0079	0.27	-1.5186	1.3294	0.3726	-0.583	-1.051	-1.5831
-0.5603	0	0.3776	0.4248	0.4528	0.2395	-0.5075	-0.2879
0.3241	0.3708	3.3879	0.6036	-0.3607	1.7976	2.4718	1.6094
0.3532	0.284	-1.0205	-0.756	-1.3542	-0.018	1.4423	1.385

Table - 2



Viz také

PDFBEGIN, PDFEND, PDFFONT, PDFFOOTER, PDFHEADER, PDFIMAGE, PDFNEWPAGE, PDFPLOT, PDFTEXT, PRINT, EXPORTGRAPH

PDFTEXT(par1[,...,parN] [,ALIGN=LEFT|RIGHT|CENTER|JUSTIFY])

Insert text. Vloží text.

Na aktuální pozici se vloží text. Použije se poslední definovaný formát textu (font, velikost, barva, atd.). Na konci řádku se text automaticky zalamuje. Seznam parametrů může obsahovat kód \n, který vloží nový řádek.

Povinné argumenty

Jeden nebo více textových řetězců, které se mají vytisknout.

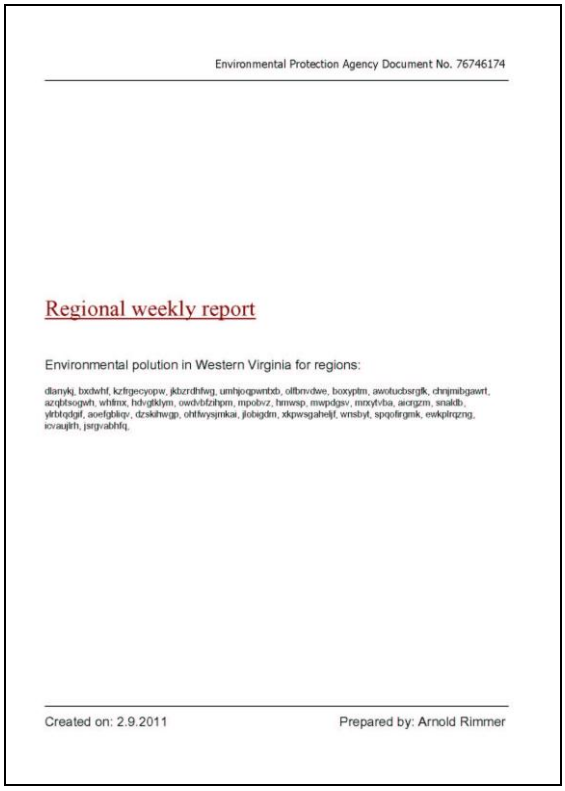
Nepovinné argumenty

ALIGN: Zarovnání textu vlevo, vpravo, doprostřed, nebo do bloku.

Příklad

```
regions=""
for(i=1,30){regions=regions+chr(sample(97:122,sample(5:12,1)
))+", "}
PDFBegin("C:\trilobyte\Report.pdf",margins=vec(15,20,20,20))
PDFHeader("","Environmental Protection Agency Document No.
"+int(random(0)*1e8),line=1)
PDFFONT(name="Times New Roman",SIZE=24,COLOR=8,underline=1)
PDFTEXT(\n,\n,\n,\n,\n,\n,\n,"Regional weekly report")
PDFFONT(NAME="Arial",SIZE=14,COLOR=4,underline=0)
PDFFooter("Created on: "+strDate(0),"Prepared by: Arnold
Rimmer",line=1)
```

```
PDFTEXT(\n,\n,"Environmental polution in Western Virginia
for regions:")
PDFFONT(SIZE=10)
PDFTEXT(\n,regions)
pdfend(launch=1)
```



Viz také

PDFBEGIN, PDFEND, PDFFONT, PDFFOOTER, PDFHEADER, PDFIMAGE, PDFNEWPAGE, PDFPLOT, PDFTABLE PRINT, EXPORTGRAPH

PI

Number Pi. Číslo Pi.

Hodnota čísla π, 3.14159265358979.

Povinné argumenty

Žádné.

Příklad

```
// Chyba numerických aproximací Pi:

>sqrt(6*sum(1/(1:50000)^2))-pi
-1.9098460222633E-5

>sqrt(sqrt(90*sum(1/(1:1000)^4)))-pi
-2.41522801758265E-10
```

```
>sqrt(sqrt(sqrt(9450*sum(1/(1:100)^8))))-pi
-4.44089209850063E-16
```

PINV(X)

Pseudoinverse of a matrix. Pseudoiverze matice.

Funkce vrací Moore-Penroseovu pseudoinverzní matici X^- k matici X. Pro regulární (čtvercovou) matici X platí, že pinv(X) = inv(X), pro singulární matici X neexistuje jednoznačná inverzní matice inv(X), avšak existuje matice pseudoinverzní pinv(X), pro níž platí $X X^- X = X$. Pseudoinverze je mimo jiné užitečná pro numerickou stabilizaci výpočtů se singulárními nebo skoro singulárními maticemi, kdy klasická inverze může selhat.

Povinné argumenty

X je matice reálných čísel

Příklad

```
>x=matrix(vec(1,1,2,2,0,1,3,1,3),ncols=3)
>x
1   2     3
1   0     1
2   1     3
>inv(x)
Error : "Nelze spočítat inverzní matici - vstupní matice je
singulární"
>pinv(x)
-0.333333333333333   0.333333333333333 0.333333333333333
0.452380952380952    -0.309523809523809      -
0.238095238095238
0.119047619047619    0.0238095238095238
    0.0952380952380953
>x#pinv(x)#x
1   2     3
1   -2.77555756156289E-16   0.999999999999999
2   1     3
```

Viz také

INV, DET, EIGENVAL, EIGENVEC

PLOT([X] [, Y][, TYPE=POINT|LINE|POINTLINE], [MAIN=, LABX=, LABY=, COLOR=, SHADE=1..100,WIDTH=, PTCOLOR=, PTSHADE=1..100, PTTYPE=, PTSIZE=, BOX=1, AXES=1])

Create new plot from given data or create empty plot frame. Vytvoření nového grafu ze zadaných dat, nebo vytvoření nového prázdného grafu.

Argumenty a použití jsou popsány u následujícího příkazu PLOTADD. Použití `PLOT()` bez argumentů, případně pouze s relevantními nepovinnými argumenty (např. main=)

vytvoří prázdný graf, do něhož lze přidávat další prvky pomocí příkazu PLOTADD, což je výhodné v cyklech FOR.

Argumenty

Viz PLOTADD.

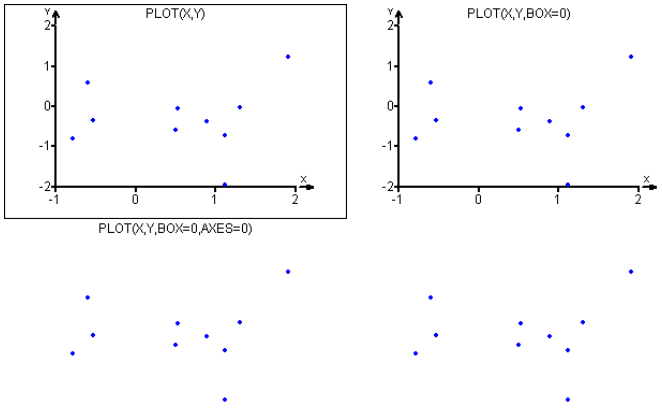
Další nepovinné argumenty

BOX: Logická numerická hodnota 0 nebo 1. Určuje, zda se kolem grafu vytvoří rámeček. Implicitní hodnota je 1.

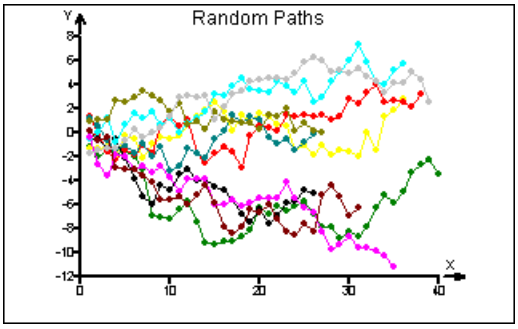
AXES: Logická numerická hodnota 0 nebo 1. Určuje, zda se v grafu zobrazí souřadnicové osy x,y s popisem. Implicitní hodnota je 1.

Příklad

```
graphsheet(cols=2) // Dva sloupce grafů
x=normalr(10)
plot(x,y)
plot(x,y,box=0)
plot(x,y,box=0,axes=0)
plot(x,y,box=0,axes=0,main="")
```




```
// Použití prázdného grafu PLOT()
plot(main="Random Paths") // Initiate plot without data
for(i=1,10)
{
n=sample(20:40,1) // Random length of a series
x=cusum(normalr(n)) // Generate random data
plotadd(x,type="pointline",color=i)
}
```



Viz také

PLOTADD, PLOTPOLY, PLOTTEXT

PLOTADD(X [,Y] [, TYPE=POINT|LINE|POINTLINE], [COLOR=, SHADE=1..100, WIDTH=, PTTYPE=])

Add new data to existing plot. Přidání dalších dat do existujícího (naposled vytvořeného) grafu.

V grafickém listu vytvoří funkce PLOT nový graf ze zadaných dat, funkce PLOTADD přidá data do naposledy vytvořeného grafu. Funkce PLOTADD nelze použít samostatně, musí být vždy spuštěna společně s funkcí vytvářející nový graf (PLOT, PLOTTEXT, apod.). Je-li zadán pouze argument X, vynáší se hodnoty X jako svislá souřadnice, vodorovnou souřadnicí je pořadové číslo. Je-li X matice, vynesou se jako nezávislé série dat všechny sloupce matice. Pokud jsou zadány oba argumenty X a Y, použije se vektor X (případně první sloupec X) jako společná vodorovná (x-ová) souřadnice a jednotlivé sloupce Y jsou y-souřadnice. Formát grafu (nadpisy, barva, spojení bodů, atd.) jsou dány dalšími argumenty funkce PLOT.

Povinné argumenty

X: vektor nebo matice číselných hodnot (pokud je zadáno Y, může být X pouze vektor).

Nepovinné argumenty

Y: vektor nebo matice číselných hodnot, y-souřadnice bodů. Musí mít stejný počet řádků jako X.

TYPE: Textový řetězec. Typ grafu, jedna z možností "point": vynese data jako body; "line": souřadnice jsou pouze spojeny čárou; "pointline": vynesou se body a spojí se čárou.

MAIN: Hlavní nadpis grafu, textový řetězec (aktivní pouze v příkazu PLOT).

LABX: Popis osy X, textový řetězec (aktivní pouze v příkazu PLOT).

LABY: Popis osy Y, textový řetězec (aktivní pouze v příkazu PLOT).

COLOR: Barva jednotlivých sérií dat, celé číslo, nebo vektor. Pokud je zadán vektor hodnot, použijí se příslušné barvy pro jednotlivé série dat (sloupce matice Y). Pak musí mít tento vektor stejný počet prvků jako počet sloupců matice Y.

SHADE: vektor stejné délky jako X s hodnotami od 0 do 100, určuje intenzitu barvy jednotlivých sérií zadané v argumentu COLOR.

WIDTH: číselný vektor se stejným počtem prvků jako počet sloupců matice X, je-li TYPE="line", nebo TYPE="pointline", určuje tloušťku čáry pro jednotlivé série matice Y. Hodnota WIDTH=1 odpovídá nejtenčí čáře.

PTCOLOR: celočíselný vektor stejné délky jako počet řádků X, určuje barvu každého bodu.

PTSHADE: jedno číslo, nebo vektor stejné délky jako X s hodnotami od 0 do 100, určuje intenzitu barvy jednotlivých bodů zadané v argumentu COLOR.

PTTYPE: vektor celých čísel od 1 do 7 stejné délky jako počet sloupců Y, popřípadě celé číslo, je-li zadáno pouze X. Určuje tvar bodu pro jednotlivé série Y. Tvary bodů a příslušná čísla jsou uvedeny v následující tabulce. Je-li vektor PTTYPE delší, použijí se jen první odpovídající hodnoty.

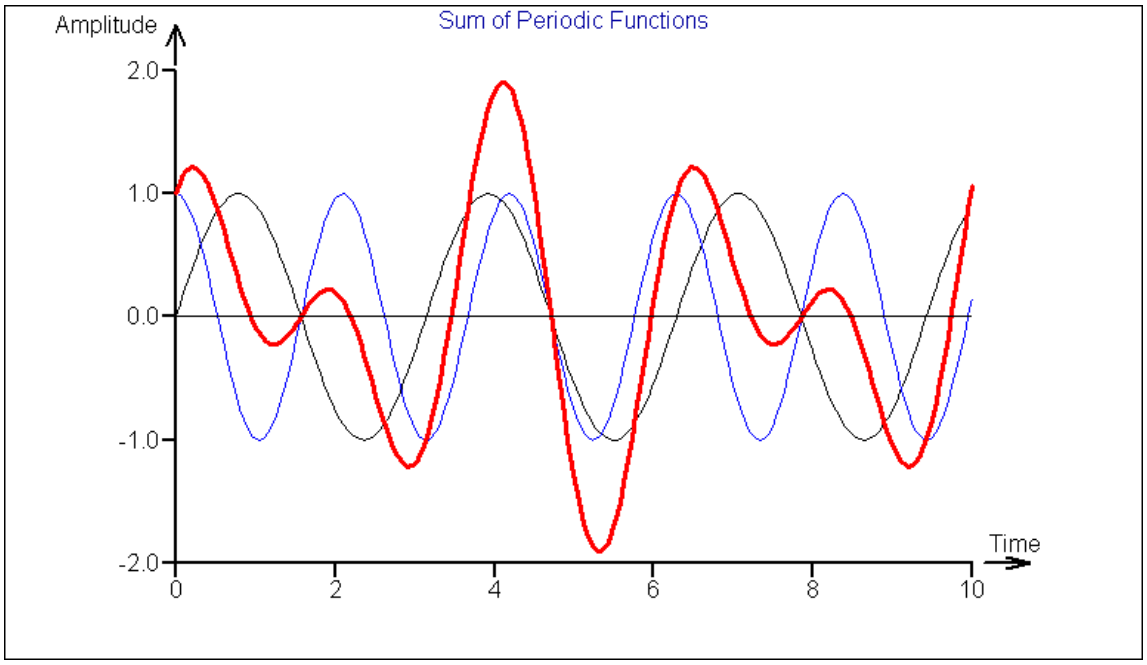
0	1	2	3	4	5	6	7
[bez znaku]							

PTSIZE jedno kladné číslo, nebo vektor stejné délky jako X, určuje relativní velikost jednotlivých bodů.

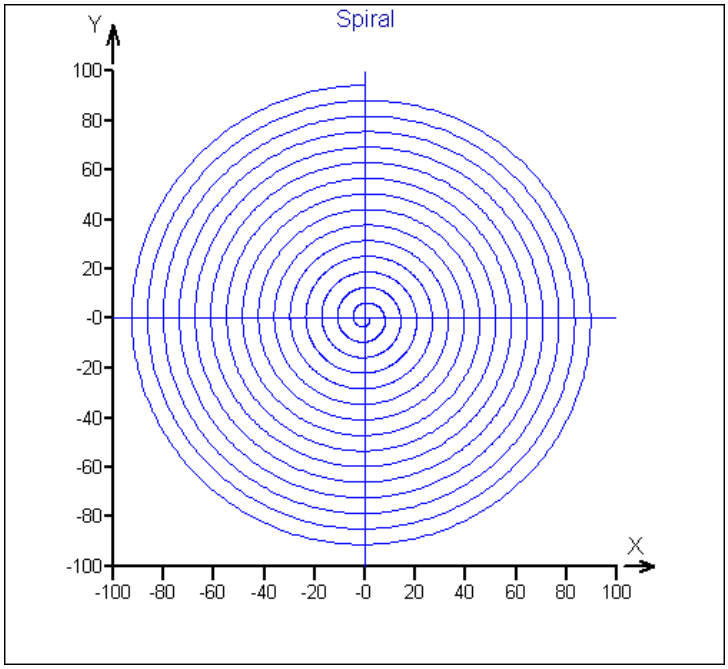
Příklad

```
// Barevná škála: Doporučujeme barevně vytisknout
// a nalepit na nástěnku.
plot(1:30, rep(1,30), ptcolor=0:29, ptsize=50, pttype=2)
plottextadd(1:30, rep(1,30)+0.15, 0:29, textsize=1.2)
lineadd(h=vec(0,2))
```

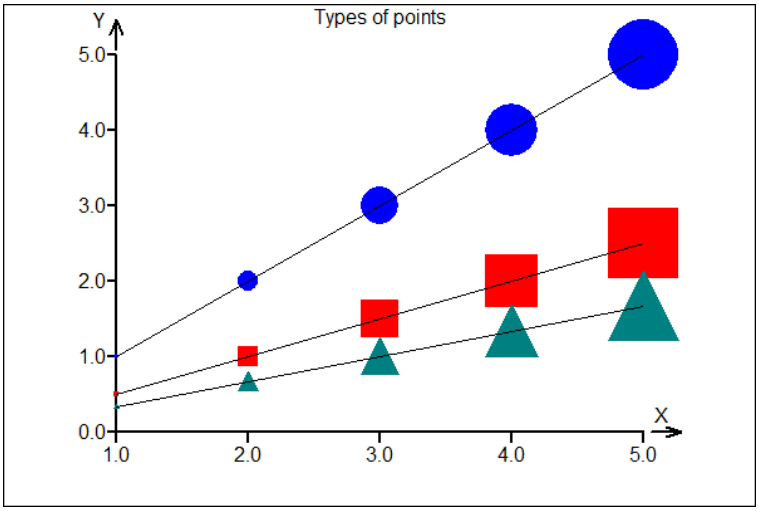

```
// Součet periodických funkcí
x=seq(0, 10, count=200)
y=sin(2*x)
y=bind(y, cos(3*x))
y=bind(y, cos(3*x)+sin(2*x))
@plot(x, y, type="line", color=vec(4,0,3),
main="Sum of Periodic Functions",
labx="Time", laby="Amplitude", width=vec(1,1,3));
lineadd(h=0,color=4)
```



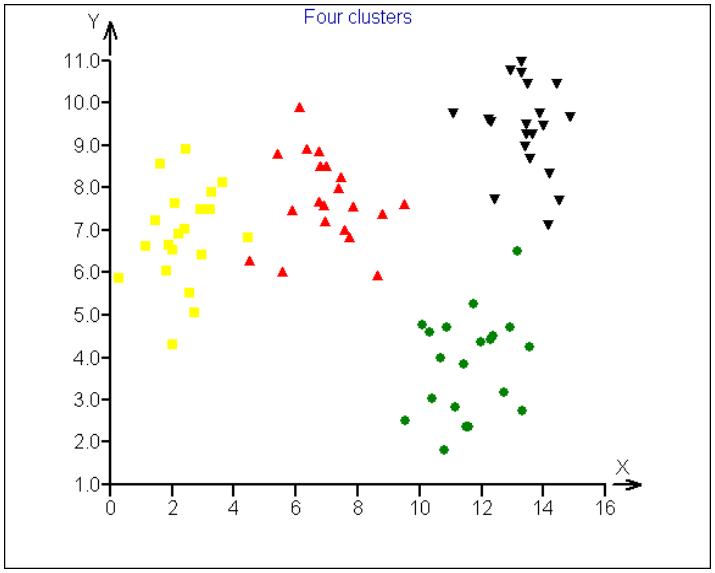

```
// Spirála
phi=seq(0,30*pi,count=5000)
x=phi*sin(phi)
y=phi*cos(phi)
plot(x,y,type="line",main="Spiral")
lineadd(v=0,h=0)
```




```
x=bind(bind(1:5,(1:5)/2),(1:5)/3)
@plot(x,color=vec(0,3,5),pttype=vec(1,2,3),
ptsize=1:5);
plotadd(x,type="line",color=4)
```




```
// Čtyři náhodné shluky různými barvami a různými symboly
x=normalr(20)+15*random(1)
y=normalr(20)+15*random(1)
plot(x,y,pttype=1,color=1,main="Four clusters")
for(i=2,4)
{
x=normalr(20)+15*random(1)
y=normalr(20)+15*random(1)
plotadd(x,y,pttype=i,color=i)
}
```



Viz také

PLOT, PLOTTEXT, PLOTTEXTADD, LINEADD, GRAPHSHEET

PLOTBAR(X, [MAIN=, LABX=, LABY=, ORIENTATION="VERTCAL"|"HORIZONTAL", GAP=, STACK=0|1, LABELS=, KEY=, COLOR=, WIDTH=, FILL=1|0, FILLCOLOR=])

Plot bars. Nakreslení sloupcového grafu.

Z vektoru nebo matice X nakreslí sloupcový graf. Je-li X matice, zobrazí se sloupce pro každý sloupec matice X. Sloupce lze uspořádat vedle sebe (STACK=0, implicitní způsob), nebo na sebe (STACK=1). Do grafu je možné vložit legendu pomocí parametru KEY. Orientace sloupců v grafu může být horizontální, nebo vertikální. Sloupce lze oddělit mezerou (GAP).

Povinné argumenty

X: vektor nebo matice číselných hodnot.

Nepovinné argumenty

MAIN: Hlavní nadpis grafu, textový řetězec.
LABX: Popis osy X, textový řetězec.
LABY: Popis osy Y, textový řetězec.
ORIENTATION: Textový řetězec "VERTCAL", nebo "HORIZONTAL", určuje orientaci sloupců.
GAP: Číselná hodnota od 0 do 0.99. Určuje relativní šířku mezery mezi sloupci. GAP=0 zobrazí sloupce bez mezery (implicitní hodnota), při GAP=0.5 je mezera stejně široká jako sloupec.
LABELS: Textový vektor, popisy jednotlivých řádků X. Popisy se zobrazí na ose grafu.
KEY: Textový vektor, zobrazí se jako legenda grafu.
COLOR: Celočíselná hodnota, Barva obrysové čáry. Je-li X matice, může být COLOR číselný vektor stejné délky, jako je počet sloupců X, hodnoty vektoru pak určují barvu obrysové čáry pro jednotlivé sloupce matice X. Není-li argument COLOR zadán, použijí se systémové barvy 0, 1, …

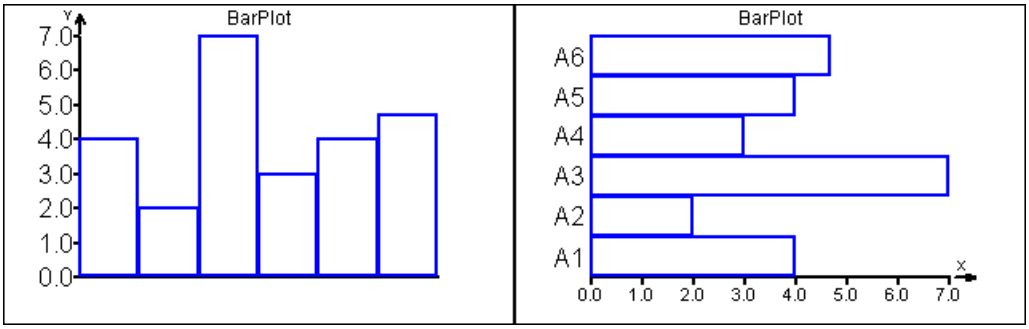
WIDTH: Šířka obrysové čáry. Je-li zadáno WIDTH=0, obrysová čára se nezobrazí.

FILL: Číselná hodnota 0, nebo 1. Výplň sloupce: je-li FILL=0 (implicitní hodnota), zobrazí se pouze obrys sloupců, je-li FILL=1, vyplní se sloupce barvou určenou argumentem FILLCOLOR. Není-li argument FILLCOLOR zadán, použijí se systémové barvy 0, 1, …

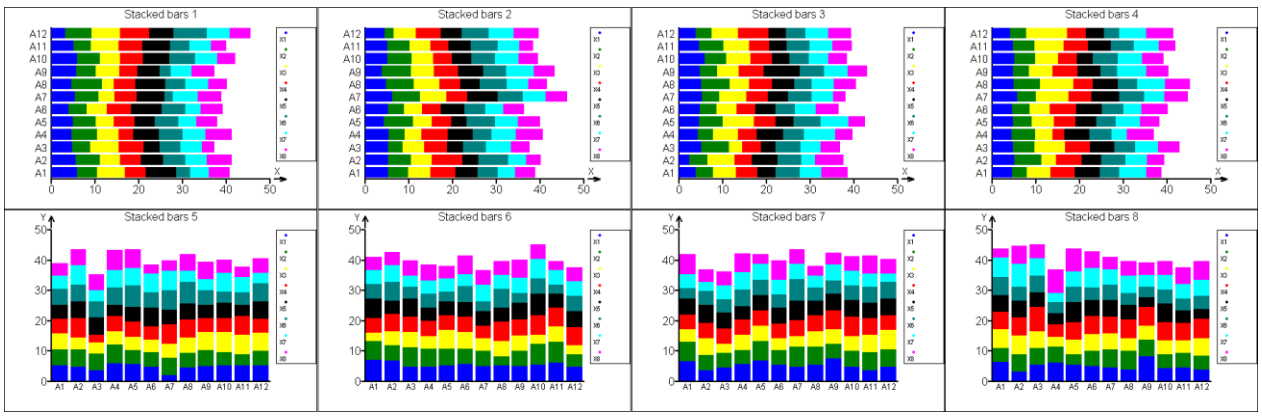
FILLCOLOR: Celočíselná hodnota, Barva výplně sloupce. Je-li X matice, může být FILLCOLOR číselný vektor stejné délky, jako je počet sloupců X, hodnoty vektoru pak určují barvu sloupce pro jednotlivé sloupce matice X. Není-li argument zadán, použijí se systémové barvy 0, 1, …

Příklad

```
x=vec(4,2,7,3,4,4.7)
labs= "A"+(1:5)
graphsheet(cols=2)
plotbar(x,main="BarPlot")
plotbar(x,main="BarPlot",orientation="horizontal",labels=labs)
```




```
n=12
m=8
ori=vec("horizontal","vertical")
graphsheet(cols=4)
for(i=1,8)
{
x=matrix(normalr(n*m),ncols=m)+5
@PLOTBAR(x,MAIN="Stacked bars "+i,width=0,
gap=0.1,fill=1,labels="A"+(1:n),KEY="X"+(1:m),
STACK=1,orientation=ori[i/4+0.9]);
}
```



Viz také

PLOT, PLOTADD, PLOTPOLY, PLOTTEXTADD, LINEADD, GRAPHSHEET

PLOTPOLY(X, Y, [MAIN=, LABX=, LABY=, COLOR=, WIDTH=, FILL=1|0, FILLCOLOR=, AXES=1|0, BOX=1|0])

Plot polygon. Nakreslení polygonu.

Ze souřadnic X a Y nakreslí uzavřený polygon (mnohoúhelník) do nového grafu. Polygon je tvořen obrysovou čárou a výplní, které mohou mít odlišnou barvu. Pokud nejsou poslední souřadnice X[n] a Y[n] stejné jako první X[1] a Y[1], tvar se automaticky doplní do uzavřeného polygonu. Obrysová čára může být potlačena zadáním nulové tloušťky, **WIDTH=0**. Další atributy jsou obdobné jako u příkazu PLOT.

Povinné argumenty

X: vektor číselných hodnot x-souřadnic polygonu.
Y: vektor číselných hodnot x-souřadnic polygonu.

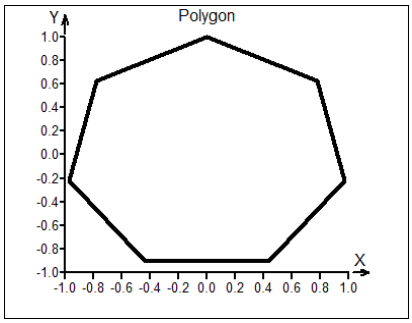
Nepovinné argumenty

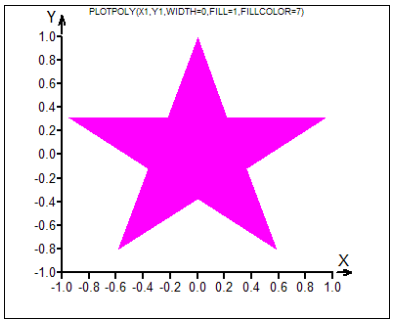
MAIN: Hlavní nadpis grafu, textový řetězec.
LABX: Popis osy X, textový řetězec.
LABY: Popis osy Y, textový řetězec.
COLOR: Barva obrysové čáry.
WIDTH: Šířka obrysové čáry. Je-li zadáno WIDTH=0, obrysová čára se nezobrazí.
FILL: Číselná hodnota 0, nebo 1. Je-li FILL=1, nakreslený polygon je vybarven barvou určenou argumentem FILLCOLOR.
FILLCOLOR: Barva výplně polygonu.
AXES: Číselná hodnota 0, nebo 1. Určuje, zda se mají zobrazit osy.
BOX: Číselná hodnota 0, nebo 1. Určuje, zda se má zobrazit rámeček kolem grafu.

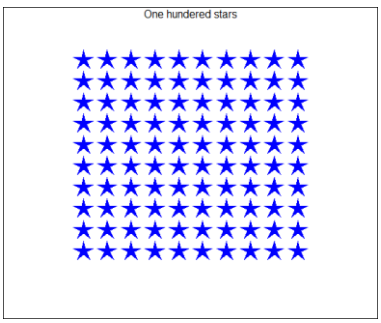
Příklad

```
// ****** N-úhelník
N=7
x=seq(0,(2-2/n)*pi,count=n)
x1=sin(x)
y1=cos(x)
PLOTPOLY(x1,y1,width=4,main="Polygon *"+N )

//****** Hvězda
N=5
x=seq(0,(2-1/n)*pi,count=2*n)
r=rep(vec(1,0.38),n)
x1=r*sin(x)
y1=r*cos(x)
PLOTPOLY(x1,y1,width=0,FILL=1,FILLCOLOR=7)

//****** Hvězdy
N=5;M=10
x=seq(0,(2-1/n)*pi,count=2*n)
r=rep(vec(1,0.38),n)
for(i=1,M)
```

```
{
for(j=1,M)
{
x1=r*sin(x)
y1=r*cos(x)
if(eq(i*j,1)){PLOTPOLY(x1+2*(i-1),y1+2*(j-1), width=0, FILL=1,
FILLCOLOR=0, main="One hundered stars", axes=0)}
if(ne(i*j,1)){PLOTPOLYADD(x1+2*(i-1),y1+2*(j-1), width=0,
FILL=1, FILLCOLOR=0)}
}
}
```







Viz také

PLOT, PLOTADD, PLOTTEXTADD, LINEADD, PLOTBAR, GRAPHSHEET

PLOTPOLYADD(par1,par2,[COLOR=,WIDTH=,FILL=1|0,FILLCOLOR=)

Add polygon to a plot. Přidání polygonu do grafu.

Vytvoří polygon v posledním nakresleném grafu a podle potřeby upraví rozsah os grafu. Význam argumentů viz příkaz PLOTPOLY.

PLOTTEXT(X, Y, S [, MAIN=, LABX=, LABY=, ALIGN=CENTER|LEFT|RIGHT, TEXTSIZE=, COLOR=, SHADE=1..100, ANGLE=])

Plot text. Vytvoření nového grafu s textem na dané pozici.

Popis funkce PLOTTEXT viz PLOTTEXTADD.

Argumenty

Viz PLOTTEXTADD.

Další nepovinné argumenty

BOX: Logická numerická hodnota 0 nebo 1. Určuje, zda se kolem grafu vytvoří rámeček. Implicitní hodnota je 1.

AXES: Logická numerická hodnota 0 nebo 1. Určuje, zda se v grafu zobrazí souřadnicové osy x,y s popisem. Implicitní hodnota je 1.

Viz také

PLOT, PLOTADD, PLOTTEXTADD, LINEADD, GRAPHSHEET

PLOTTEXTADD(X, Y, S [, ALIGN=CENTER|LEFT|RIGHT, TEXTSIZE=, COLOR=, SHADE=1..100, ANGLE=])

Add text to plot. Přidání textu do existujícího grafu.

Umístí daný text do grafu. Na pozice dané souřadnicemi X a Y se umístí text v řetězcovém vektoru S.

V grafickém listu vytvoří funkce PLOTTEXT nový graf ze zadaných dat, funkce PLOTTEXTADD přidá data do naposledy vytvořeného grafu. Funkce PLOTTEXTADD nelze použít samostatně, musí být vždy spuštěna společně s funkcí vytvářející nový graf (PLOT, PLOTTEXT, apod.). Musí být vždy zadány alespoň X, Y a S. Je-li X matice, vynesou se jako nezávislé série dat všechny sloupce matice. Pokud jsou zadány oba argumenty X a Y, použije se vektor X (případně první sloupec X) jako společná vodorovná (x-ová) souřadnice a jednotlivé sloupce Y jsou y-souřadnice. Formát grafu (nadpisy, barva, spojení bodů, atd.) jsou dány dalšími argumenty funkce PLOT.

Povinné argumenty

X: vektor číselných hodnot (pokud je zadáno Y, může být X pouze vektor).

Y: vektor číselných hodnot, y-souřadnice bodů. Musí mít stejný počet řádků jako X.

S: Vektor textových řetězců, případně číselný vektor, jednotlivé prvky vektoru S se umístí v grafu na souřadnice dané vektory X a Y.

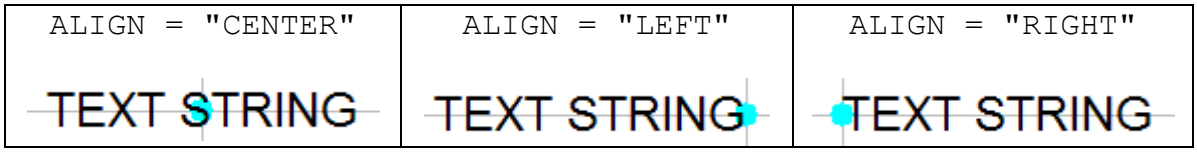
Nepovinné argumenty

MAIN: Hlavní nadpis grafu, textový řetězec.

LABX: Popis osy X, textový řetězec.

LABY: Popis osy Y, textový řetězec.

ALIGN: Zarovnání textového řetězce vzhledem k souřadnicím (x,y). CENTER: souřadnice bodu je ve středu textového řetězce; LEFT: souřadnice bodu je na pravém okraji textového řetězce (řetězec se nachází vlevo od bodu (x,y); RIGHT: souřadnice bodu je na levém okraji textového řetězce (řetězec se nachází vpravo od bodu (x,y), viz následující ilustrace.



TEXTSIZE: Číslo, nebo číselný vektor udávající velikost textu. Je-li TEXTSIZE vektor, musí mít stejnou délku jako S. Základní velikost textu je TEXTSIZE=1. Hodnota 2 znamená dvojnásobnou velikost textu.

COLOR: Barva jednotlivých dat, celé číslo, nebo vektor. Pokud je zadán vektor hodnot. použijí se příslušné barvy pro jednotlivá data. Pak musí mít tento vektor stejný počet prvků jako počet sloupců matice Y. Je-li COLOR jediné číslo, použije se odpovídající barva pro všechna data.

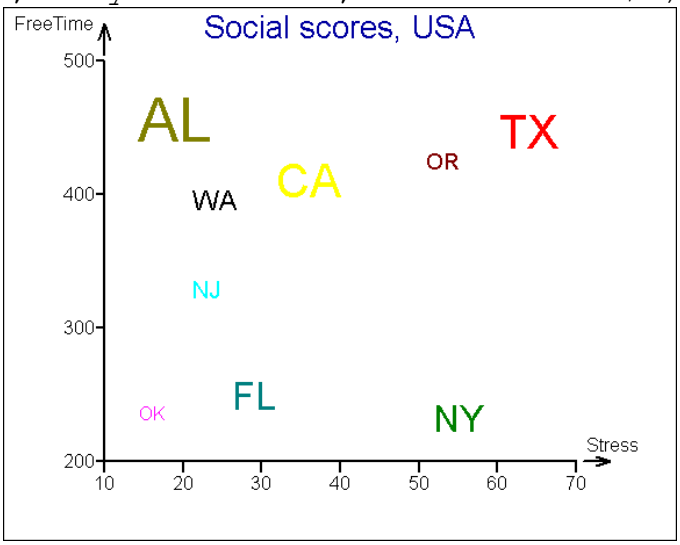
SHADE: vektor stejné délky jako X s hodnotami od 0 do 100, určuje intenzitu barvy jednotlivých sérií zadané v argumentu COLOR. Je-li SHADE jediné číslo, použije se odpovídající odstín pro všechna data.

PTCOLOR: celočíselný vektor stejné délky jako počet řádků X, určuje barvu každého bodu.

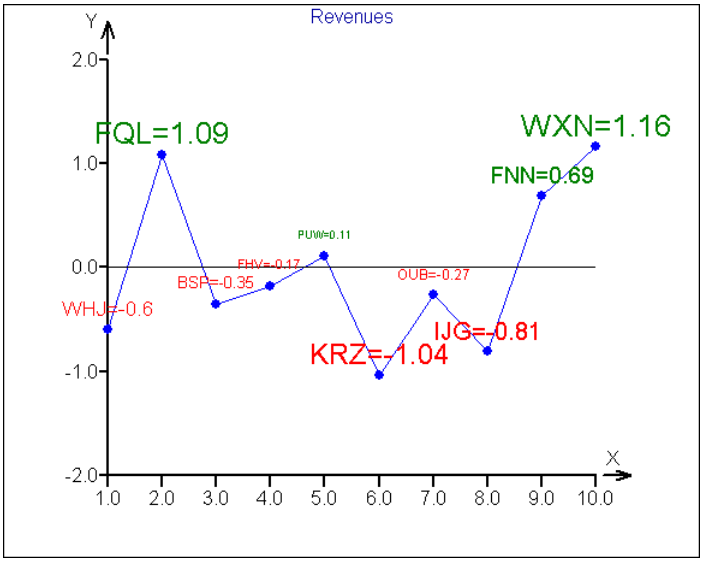
ANGLE: Číslo, úhel natočení textu ve stupních, 0 = vodorovně; 90, 270 = svisle.

Příklad

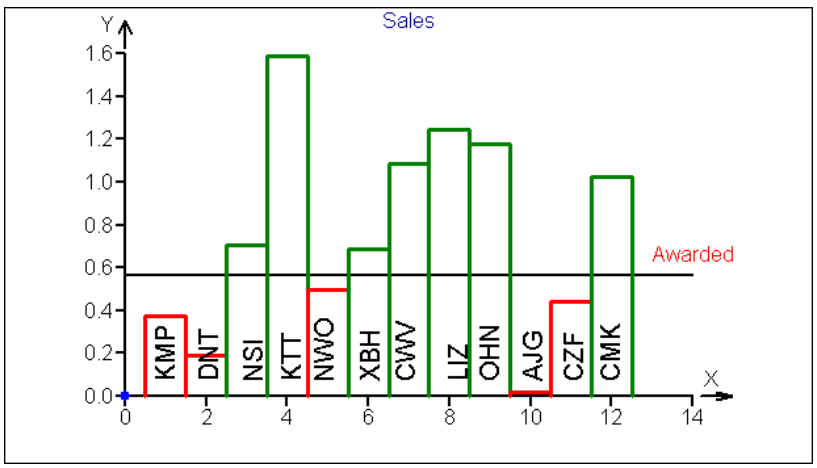
```
x=vec(55,36,64,24,29,23,16,53,19)
y=vec(231,410,446,395,249,328,236,424,455)
size=vec(20,25,25,15,21,14,10,12,34)
state=vec("NY","CA","TX","WA","FL","NJ","OK","OR","AL")
@plottext(x, y, state, color=1:9,
color=4, main="Social scores, USA",
labx="Stress", laby="FreeTime", textsize=size/2);
```




```
N=10
delete(ss)
for(i=1,N){ss[i]=chr(sample(65:90,3,repl=1))}
x=1:N
y=normalr(N)
ss1=ss+"="+round(y,2)
plot(x,y,type="pointline",main="Revenues")
plottextadd(x,y+0.2,ss1,textsize=abs(y)+0.5,color=-sign(y)+2)
lineadd(h=0,color=4)
```




```
N=12
delete(ss)
for(i=1,N){ss[i]=chr(sample(65:90,3,repl=1))}
plot(0,0,main="Sales")
sle=abs(normalr(N))
ave75=0.75*average(sle)
for(i=1,N)
{
icol=1; if (lt(sle[i],ave75)) {icol=3}
x=vec(i-0.5,i-0.5,i+0.5,i+0.5)
y=vec(0,sle[i],sle[i],0)
plotadd(x,y,type="line",width=3,color=icol)
}
@plottextadd((1:N),rep(0.2,N),ss,angle=90,
textsize=1.3,align="center");
lineadd(h=ave75,width=2,color=4)
plottextadd(N+2,0.75*average(sle)+0.1,"Awarded",color=3)
```



Viz také

PLOT, PLOTADD, PLOTTEXT, LINEADD, GRAPHSHEET

PLOT3DPOINTS(X, Y, Z, [MAIN=, ANGLEX=, ANGLEY=, ANGLEZ=, BOX=0|1|2, AXES=0|1, ISOMETRIC=0|1, ORIGIN=vec(x,y,z)])

Dynamic 3d point plot. Dynamický 3d bodový graf.

Vytvoří trojrozměrný graf v grafickém listu s počátkem souřadnic v hodnotě aritmetického průměru dat. Po dvojitém kliknutí lze grafem otáčet v samostatném okně. Další podrobnosti o 3D-Grafu viz Uživatelský manuál QCExpert.

Povinné argumenty

X, Y, Z: Číselné vektory, které mají být zobrazeny v grafu.

Nepovinné argumenty

MAIN: Textový řetězec, nadpis grafu.

ANGLEX, ANGLEY, ANGLEZ: Číselné hodnoty. Počáteční úhly natočení podle jednotlivých os. Tyto hodnoty odpovídají hodnotám X, Y, Z v okně 3D-grafu.

BOX: Číselná hodnota, 0, 1, nebo 2. Při hodnotě 0 se nezobrazí ohraničující krychle, při hodnotě 1 se zobrazí jen obrys krychle, při hodnotě 2 se zobrazí krychle s viditelnými stěnami. Odpovídá volbě Ohraničení v okně 3D grafu.

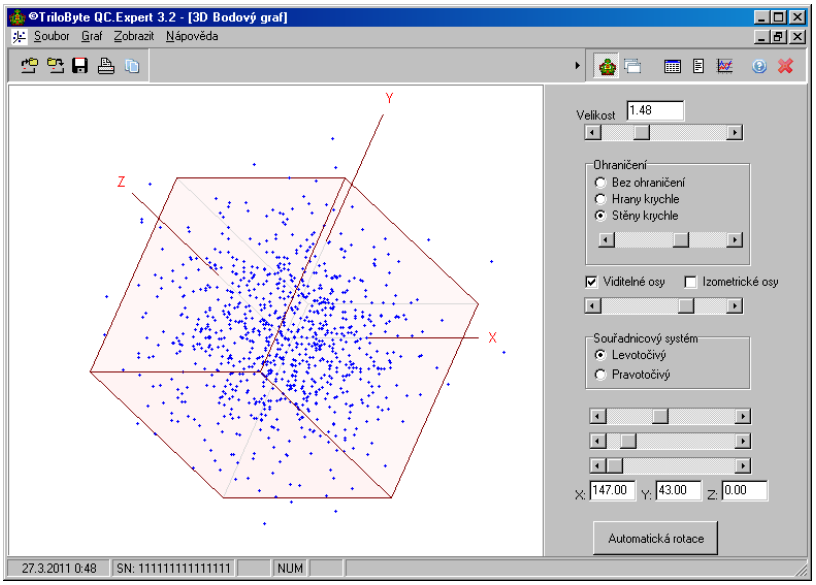
AXES: Číselná hodnota 0 nebo 1, určuje viditelnost os x, y, z v grafu. Odpovídá volbě Viditelné osy v okně 3D grafu.

ISOMETRIC: Číselná hodnota 0 nebo 1, hodnota 1 normuje rozpětí dat, 0 zobrazí data v původním měřítku, odpovídá volbě Izometrické osy v okně 3D grafu.

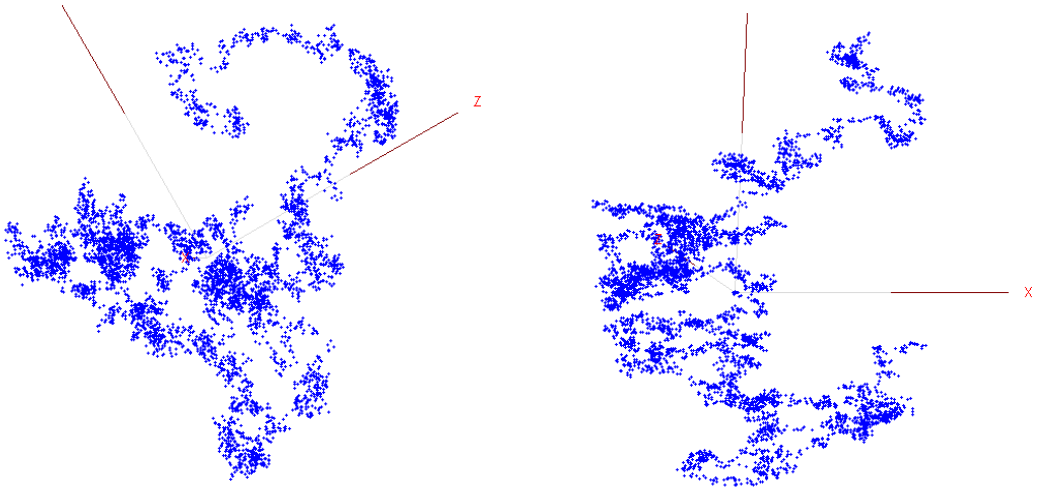
ORIGIN: Číselný vektor délky 3. Definuje souřadnice počátku.

Příklad

```
R=matrix(normalr(3*900),ncols=3) //náhodná matice 900 x 3
plot3dpoints(R[,1],R[,2],R[,3],main="3d Plot")
```




```
@R=bind(bind(cusum(normalr(5000)),
cusum(normalr(5000))),cusum(normalr(5000)));
plot3dpoints(R[,1], R[,2], R[,3], main="3d Plot", box=0)
```



Viz také

PLOT,GRAPHSHEET,PLOT3DSURFACE,PLOT3DSPLINE, PLOT3DDENSITY

PLOT3DSURFACE(Z, [MAIN=, XLIM=vec(x1, x1), YLIM=vec(x1, x2), ANGLEX=, ANGLEZ=, BOX=0|1|2, COLRANGE=vec(col1, col2 [, col3]), GRIDCOLOR=])

Plot surface from matrix data. Graf trojrozměrného povrchu z matice.

Trojrozměrný síťový graf je vytvořen z hodnot v matici Z o rozměrech [N x M]. Po otevření grafu dvojitým kliknutím lze grafem otáčet v samostatném okně. Další podrobnosti o 3D-Grafu viz Uživatelský manuál QCExpert.

Povinné argumenty

Z: Matice reálných čísel o rozměrech [N x M]. Hodnoty matice Z se vynesou v grafu do uzlových bodů (průsečíků mřížky). Hustota sítě je tedy určena velikostí matice Z. K určení x-ové a y-ové souřadnic bodů se použijí celočíselné indexy jednotlivých prvků matice Z (1 až N a 1 až M), nebo se použijí argumenty XLIM a YLIM, jsou-li zadány.

Nepovinné argumenty

MAIN: Textový řetězec, nadpis grafu.

XLIM, YLIM: Číselné vektory se dvěma prvky. Meze pro x-ové a y-ové souřadnice grafu. Souřadnice se vygenerují jako ekvidistantní body v intervalu x1 až x2, resp. y1 až y2.

ANGLEX, ANGLEZ: Číselné hodnoty. Počáteční úhly natočení podle jednotlivých os ve stupních. Tyto hodnoty odpovídají hodnotám Úhel X, Úhel Z v okně 3D-grafu. Hodnoty ANGLEX=0, ANGLEZ=270 odpovídají 2d-pohledu shora. Po vypnutí mřížky v interaktivním okně grafu lze vhodnou kontrastní volbou barev zvýraznit vrstevnice. Nastavení lze měnit interaktivně po otevření grafu.

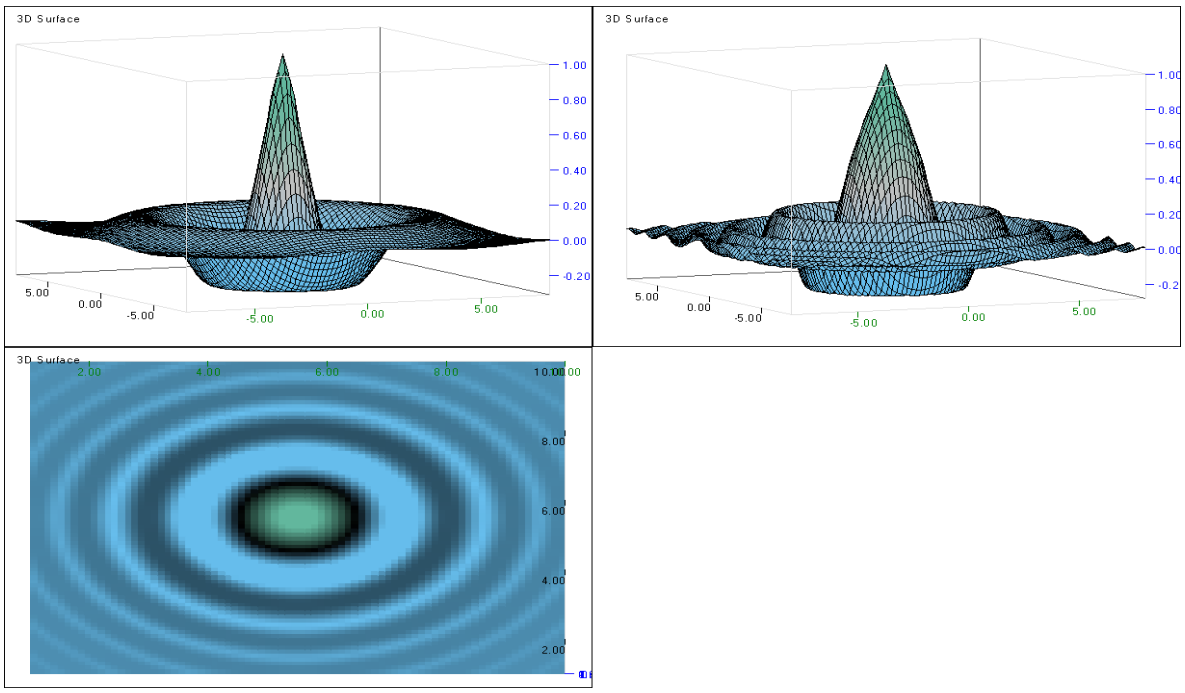
BOX: Číselná hodnota, 0, 1, nebo 2. Při hodnotě 0 se nezobrazí ohraničující krychle, při hodnotě 1 se zobrazí jen obrys krychle, při hodnotě 2 se zobrazí krychle s viditelnými stěnami. Odpovídá volbě Ohraničení v okně 3D grafu. Nastavení lze měnit interaktivně po otevření grafu.

COLRANGE: Celočíselný vektor se dvěma nebo třemi prvky. Prvky vektoru jsou barvy nízkých, (středních) a vysokých hodnot plochy. Pokud je zadán dvouprvkový vektor, použije se pouze dvoubarevný přechod od nízkých hodnot (barva col1) k vysokým hodnotám (barva col2). Pokud je zadán tříprvkový vektor, použije se tříbarevný přechod od nízkých hodnot (barva col1) ke středním (barva col2) a k vysokým hodnotám (barva col3). Nastavení lze měnit interaktivně po otevření grafu.

GRIDCOLOR : Celé číslo. Barva mřížky v grafu.

Příklad

```
delete(z1,z2)
z1[81,81]=0
z2[81,81]=0
for(i=-40,40)
{for (j=-40,40)
{ x=i/5;y=j/5;t=sqrt(x*x+y*y)
  z1[i+41,j+41]=(cos(t))*exp(-0.4*t)
  z2[i+41,j+41]=(cos(t*t*0.3))*exp(-0.4*t)
}}
graphsheet(cols=2)
@plot3dsurface(z1,main="3D Surface",
colrange=vec(22,10,23),xlim=vec(-8,8),
ylim=vec(-8,8),anglex=80,anglez=245,box=1);
@plot3dsurface(z2,main="3D Surface",
colrange=vec(22,10,23),xlim=vec(-8,8),
ylim=vec(-8,8),anglex=80,anglez=245,box=1);
@plot3dsurface(z2,main="3D Surface",
colrange=vec(22,4,23),anglex=0,anglez=270,box=1);
```



Viz také

PLOT,PLOT3DPOINTS,GRAPHSHEET,PLOT3DSPLINE,PLOT3DDENSITY

PLOT3DSPLINE(X, Y, Z, [MAIN=, SMOOTH=1, GRID=vec(a, b), ANGLEX=, ANGLEZ=, BOX=0|1|2, COLRANGE=vec(col1, ..), GRIDCOLOR=])

Plot SMOOTHED surface from data. Graf trojrozměrného vyhlazeného povrchu.

Trojrozměrný síťový graf jádrového vyhlazení zadaných dat. Data jsou zadána pomocí tří vektorů X, Y, Z, k vyhlazení se používá Gaussovy jádrové funkce

$$z(x,y)=\frac{\sum_{i=1}^{n} z_i \exp\left\{\frac{1}{h}\left[\frac{(x_i-x)^2}{s_x^2}+\frac{(y_i-y)^2}{s_y^2}\right]\right\}}{\sum_{i=1}^{n} \exp\left\{\frac{1}{h}\left[\frac{(x_i-x)^2}{s_x^2}+\frac{(y_i-y)^2}{s_y^2}\right]\right\}},$$

kde h je vyhlazovací koeficient (čím vyšší, tím hladší povrch), s_x a s_y jsou výběrové směrodatné odchylky vektoru X a Y. x_i, y_i, z_i jsou i-té prvky vektorů X, Y, Z.

Po otevření grafu dvojitým kliknutím lze grafem otáčet v samostatném okně. Další podrobnosti o 3D-Grafu viz Uživatelský manuál QCExpert.

Povinné argumenty

X, Y, Z: Číselné vektory x-, y-, z-ových souřadnic bodů, které mají být vyhlazeny.

Nepovinné argumenty

MAIN: Textový řetězec, nadpis grafu.

SMOOTH: Kladné reálné číslo, míra vyhlazení. Čím je SMOOTH větší, tím bude výsledná plocha hladší, a naopak. Implicitní hodnota je 1.

GRID: Celočíselný dvouprvkový vektor. Prvky vektoru představují počty čar mřížky ve směru X a Y.

ANGLEX, ANGLEZ: Číselné hodnoty. Počáteční úhly natočení podle jednotlivých os ve stupních. Tyto hodnoty odpovídají hodnotám Úhel X, Úhel Z v okně 3D-grafu. Hodnoty ANGLEX=0, ANGLEZ=270 odpovídají 2d-pohledu shora. Po vypnutí mřížky v interaktivním okně grafu lze vhodnou kontrastní volbou barev zvýraznit vrstevnice. Nastavení lze měnit interaktivně po otevření grafu.

BOX: Číselná hodnota, 0, 1, nebo 2. Při hodnotě 0 se nezobrazí ohraničující krychle, při hodnotě 1 se zobrazí jen obrys krychle, při hodnotě 2 se zobrazí krychle s viditelnými stěnami. Odpovídá volbě Ohraničení v okně 3D grafu. Nastavení lze měnit interaktivně po otevření grafu.

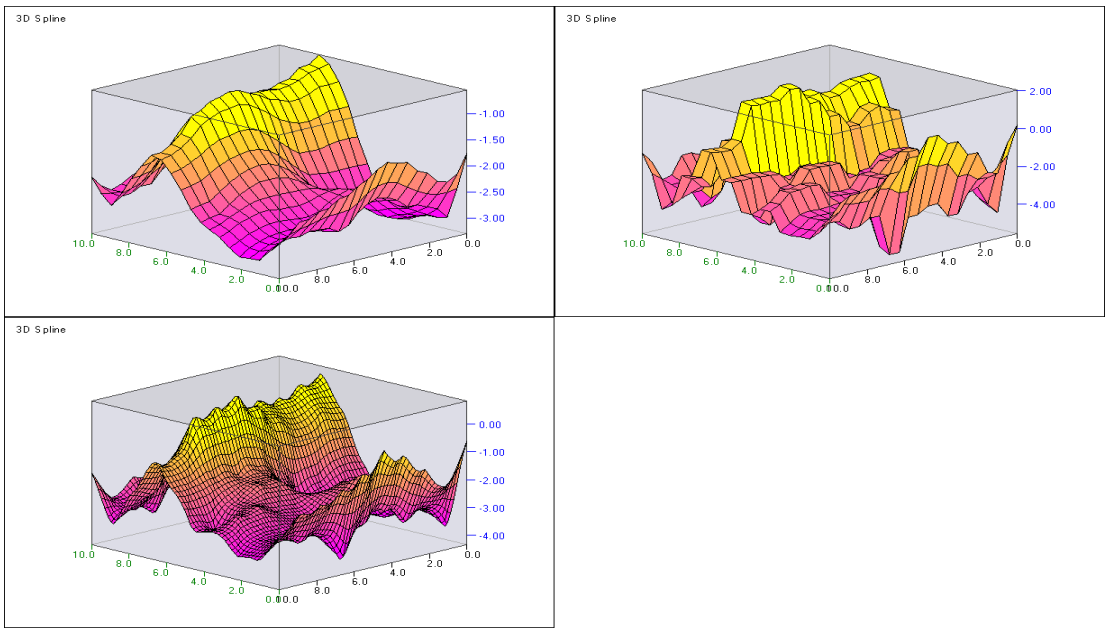
COLRANGE: Celočíselný vektor se dvěma nebo třemi prvky. Prvky vektoru jsou barvy nízkých, (středních) a vysokých hodnot plochy. Pokud je zadán dvouprvkový vektor, použije se pouze dvoubarevný přechod od nízkých hodnot (barva col1) k vysokým hodnotám (barva col2). Pokud je zadán tříprvkový vektor, použije se tříbarevný přechod od nízkých hodnot (barva col1) ke středním (barva col2) a k vysokým hodnotám (barva col3). Nastavení lze měnit interaktivně po otevření grafu.

GRIDCOLOR : Celé číslo. Barva mřížky v grafu.

Příklad

```
graphsheet(cols=2)
```

```
x=rep(1:10,10)
y=sort(1,x)
z=cusum(normalr(100))
plot3Dspline(x,y,z)
plot3Dspline(x,y,z,smooth=0.1)
plot3Dspline(x,y,z,smooth=0.5,grid=vec(60,60))
```



Viz také

PLOT,PLOT3DPOINTS,GRAPHSHEET,PLOT3DSURFACE,PLOT3DDENSITY

PLOT3DDENSITY(X, Y [, MAIN=, SMOOTH=1, GRID=vec(a, b), ANGLEX=, ANGLEZ=, BOX=0|1|2, COLRANGE=vec(col1, ..), GRIDCOLOR=])

Plot kernel-smoothed density of a 2-dimensional distribution. Graf jádrového odhadu hustoty pravděpodobnosti 2-rozměrného rozdělení.

Trojrozměrný síťový graf jádrového odhadu hustoty rozdělení $f(\mathbf{x})$ zadaných dat, kde $\mathbf{x}=(x,y)$ je dvourozměrná náhodná veličina. Data jsou zadána pomocí dvou vektorů X, Y, k vyhlazení se používá Gaussovské jádrové funkce

$$f(\mathbf{x}) = (2\pi n)^{-1} |h\boldsymbol{\Sigma}|^{-\frac{1}{2}} \sum_{i=1}^{n} \exp\left\{-\frac{1}{2}(\mathbf{x}-\mathbf{x}_i)^T (h\boldsymbol{\Sigma})^{-1} (\mathbf{x}-\mathbf{x}_i)\right\},$$

kde h je vyhlazovací koeficient (čím vyšší, tím hladší odhad hustoty, implicitně $h = 1$), $\boldsymbol{\Sigma}$ je výběrová kovarianční matice vektoru X a Y. $\mathbf{x}_i = (x_i, y_i)$ jsou i-té prvky vektorů X, Y.

Po otevření grafu dvojitým kliknutím lze grafem otáčet v samostatném okně. Další podrobnosti o tomto grafu viz Uživatelský manuál QCExpert.

Povinné argumenty

X, Y: Číselné vektory x-, y-ových souřadnic bodů.

Nepovinné argumenty

MAIN: Textový řetězec, nadpis grafu.

SMOOTH: Kladné reálné číslo, míra vyhlazení. Čím je SMOOTH větší, tím bude výsledná plocha hladší, a naopak. Implicitní hodnota je 1.

GRID: Celočíselný dvouprvkový vektor. Prvky vektoru představují počty čar mřížky ve směru X a Y.

ANGLEX, ANGLEZ: Číselné hodnoty. Počáteční úhly natočení podle jednotlivých os ve stupních. Tyto hodnoty odpovídají hodnotám Úhel X, Úhel Z v okně 3D-grafu. Hodnoty ANGLEX=0, ANGLEZ=270 odpovídají 2d-pohledu shora. Po vypnutí mřížky v interaktivním okně grafu lze vhodnou kontrastní volbou barev zvýraznit vrstevnice. Nastavení lze měnit interaktivně po otevření grafu.

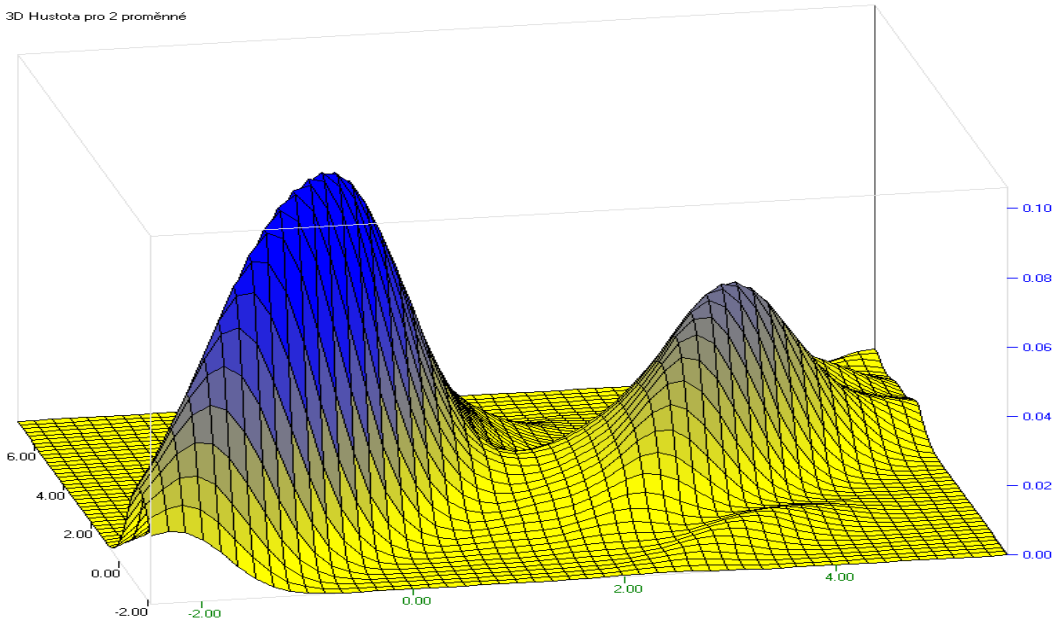
BOX: Číselná hodnota, 0, 1, nebo 2. Při hodnotě 0 se nezobrazí ohraničující krychle, při hodnotě 1 se zobrazí jen obrys krychle, při hodnotě 2 se zobrazí krychle s viditelnými stěnami. Odpovídá volbě Ohraničení v okně 3D grafu. Nastavení lze měnit interaktivně po otevření grafu.

COLRANGE: Celočíselný vektor se dvěma nebo třemi prvky. Prvky vektoru jsou barvy nízkých, (středních) a vysokých hodnot plochy. Pokud je zadán dvouprvkový vektor, použije se pouze dvoubarevný přechod od nízkých hodnot (barva col1) k vysokým hodnotám (barva col2). Pokud je zadán tříprvkový vektor, použije se tříbarevný přechod od nízkých hodnot (barva col1) ke středním (barva col2) a k vysokým hodnotám (barva col3). Nastavení lze měnit interaktivně po otevření grafu.

GRIDCOLOR : Celé číslo. Barva mřížky v grafu.

Příklad

```
x=vec(normalr(50),normalr(30)+5)
y=vec(x[1:50]+0.5*normalr(50),x[51:80]+normalr(30)-3)
plot3Ddensity(x,y,grid=vec(50,50),colrange=vec(2,0),box=1)
```



Viz také

PLOT, PLOT3DPOINTS, GRAPHSHEET, PLOT3DSURFACE, PLOT3DSPLINE

POLYREG(X,Y,W=REP(1,NROWS(X)),XNEW=X,POLDEG=0:2,

ALPHA=0.05)

Fit polynomial regression. Polynomická regrese.

Pro daný vektor délky *N* nezávisle proměnné *x* a k ní příslušný vektor délky *N* závisle proměnné *y* vypočítá odhady koeficientů **a** polynomického regresního modelu pro jednu nezávisle proměnnou *x*. Polynomické členy modelu jsou určeny argumentem POLDEG, který obsahuje zvolené mocniny ve tvaru vektoru délky *M*.

$$y = a_1 x^{p_1} + a_2 x^{p_2} + \ldots + a_m x^{p_m},$$

kde p_i jsou uživatelem zvolené mocniny zahrnuté do modelu. Jednotlivé mocniny mohou být libovolná reálná čísla, včetně desetinných a záporných, takže je možné zadat i modely obsahující členy $1/x$, $x^{1/2}$, a podobně. Implicitní hodnoty argumentů odpovídají kvadratickému modelu $y = a_1 + a_2 x + a_3 x^2$. Výsledkem je seznam.

Povinné argumenty

X: Vektor reálných čísel délky *N* obsahující hodnoty nezávisle proměnné.

Y: Vektor reálných čísel délky *N* obsahující hodnoty závisle proměnné.

Nepovinné argumenty

W: Vektor nezáporných reálných čísel délky *N* obsahující hodnoty relativních vah pro jednotlivá měření. Na hodnoty vah nejsou kladena jiná omezení (normování se provede automaticky).

XNEW: Vektor reálných čísel délky N_1 obsahující nové hodnoty nezávisle proměnné, pro něž má být počítána predikce a interval spolehlivosti predikce, implicitní hodnota je shodná s argumentem X.

POLDEG: Vektor reálných čísel délky *M* obsahující hodnoty exponentů jednotlivých členů modelu. Vektor nesmí obsahovat stejné hodnoty, Prvky tohoto vektoru mohou být necelá i záporná čísla. Implicitní hodnota je 0:2, což odpovídá absolutnímu členu $a_1 x^0 = a_1$, lineárnímu členu $a_2 x$ a kvadratickému členu $a_3 x^2$. Pokud bychom chtěli například model

$$y = a_1 + a_2 \frac{1}{x} + a_3 \sqrt{x} + a_4 x + a_5 x^2,$$

zadali bychom POLDEG=vec(0, –1, 0.5, 1, 2).

ALPHA: kladná numerická hodnota menší než 0.5, zvolená hladina významnosti pro výpočet intervalu spolehlivosti, implicitní hodnota ALPHA=0.05.

Struktura výsledného seznamu

$A: Bodové odhady regresních koeficientů, číselný vektor délky *M* (podle délky argumentu POLDEG).

$YPRED: Predikované hodnoty predikce *Y* pro zadané hodnoty **X**. Numerický vektor délky *N*.

$VARA: Kovarianční matice (*M*x*M*) odhadu koeficientů **a**, rozptyly koeficientů jsou na diagonále.

$CORA: Korelační matice (*M*x*M*) odhadu koeficientů **a**.

$YNEW: Predikované hodnoty pro nový vektor nezávisle proměnné XNEW, numerický vektor délky N_1.

$CI: Polovina intervalu spolehlivosti predikce YNEW. Interval spolehlivosti predikované hodnoty YNEW[i] se tedy získá jako YNEW[i]-CINEW[i] a YNEW[i]+CINEW[i], numerický vektor délky N_1.

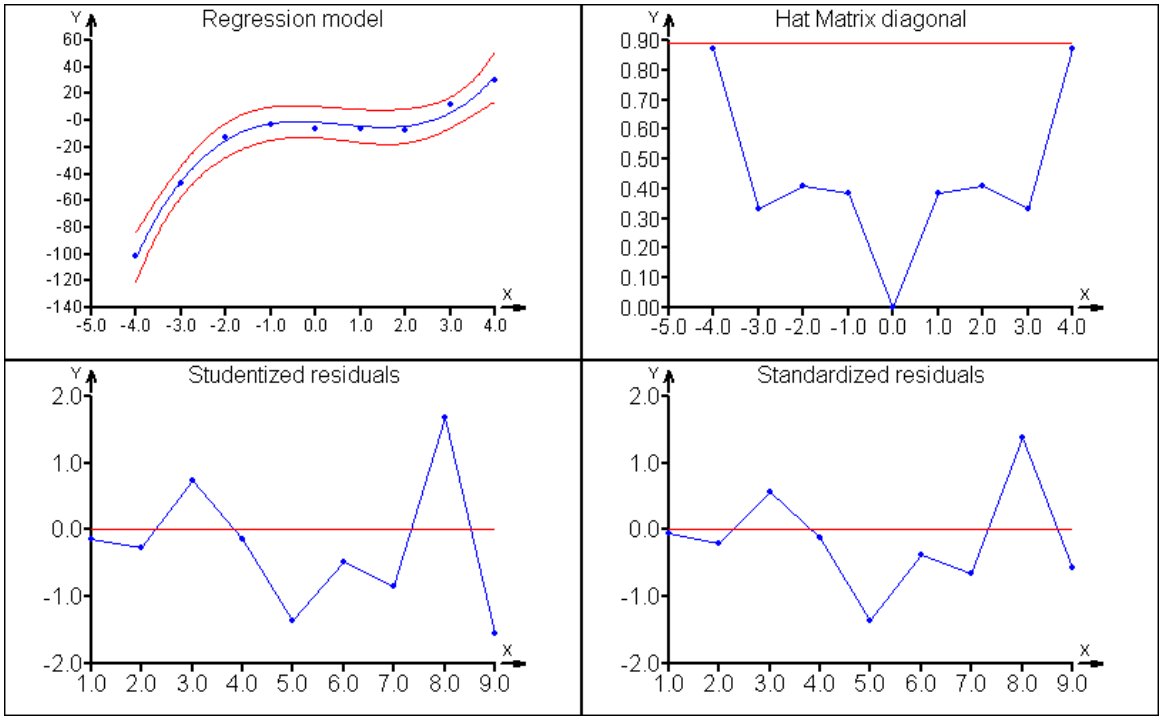
$HATDIAG: Diagonální elementy projekční matice („Hat-matrix“) $H = X(X^TX)^{-1}X^T$ používané k posouzení vlivu jednotlivých dat, numerický vektor délky N.

$SIG2: Odhad reziduálního rozptylu.

$RESID: Regresní rezidua (Y - YPRED), numerický vektor délky N.

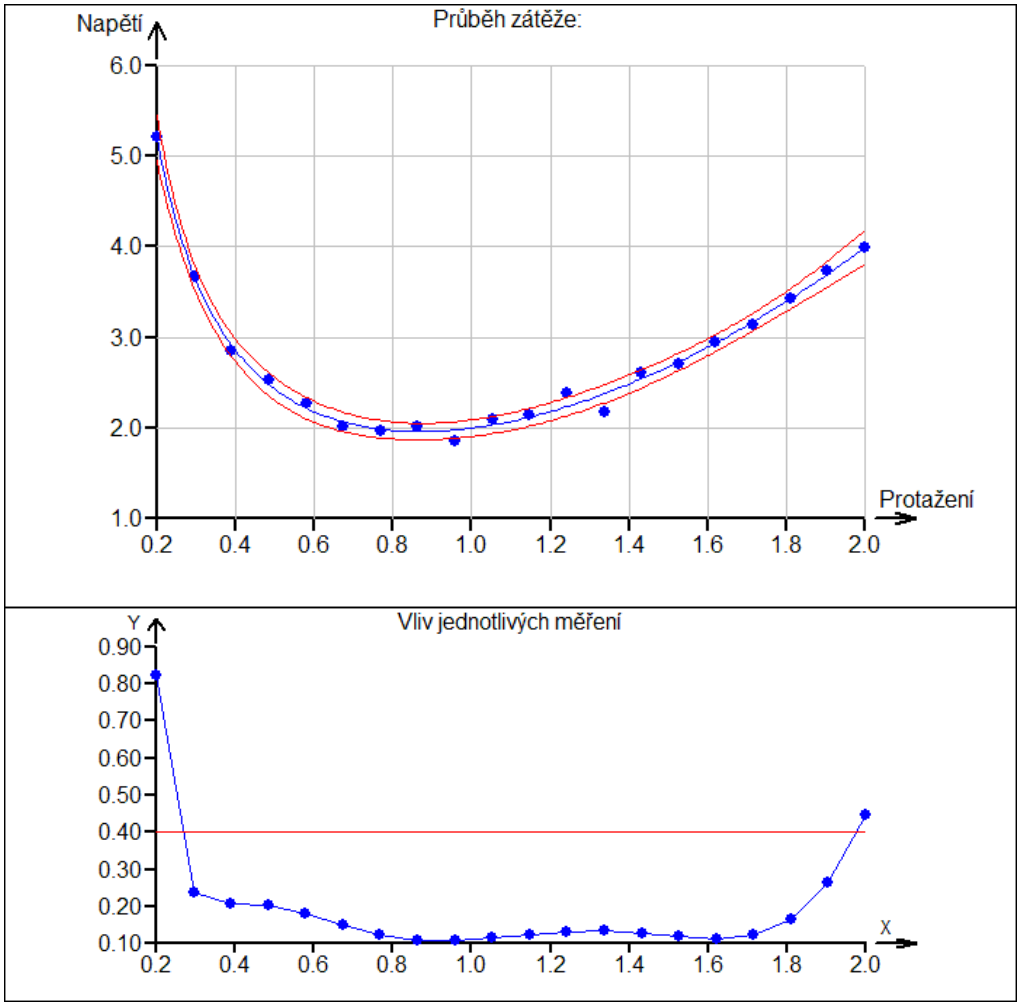
Příklad 1

```
// Vstupy: X, Y a mocniny pro polynom třetího stupně
err=normalr(9)*5
x=-4:4
y=x^3-2*x^2+x-1 + err
deg=0:3
// Počet dat a členů regresního modelu:
n=count(y)
m=count(deg)
// Nové X pro predikci a graf (XNEW):
xg=seq(min(x),max(x),count=200)
// Výpočet regrese:
pr=polyreg(x,y,xnew=xg,poldeg=deg)
// Grafy do 2 sloupců:
graphsheet(cols=2)
// Graf dat a regresní funkce s intervalem spolehlivosti
plot(x,y,main="Regression model")
plotadd(xg,PR$YNEW,type="line",main="Regression function")
plotadd(xg,PR$YNEW+PR$CI,type="line",color=3)
plotadd(xg,PR$YNEW-PR$CI,type="line",color=3)
// Graf vlivu jednotlivých dat:
plot(x,pr$hatdiag,type="pointline",main="Hat Matrix diagonal")
lineadd(h=2*m/n,color=3)
// Graf studentizovaných a klasických reziduí:
rstu=PR$RESID/sqrt(PR$SIG2*(1-PR$HATDIAG))
plot(rstu,type="pointline",main="Studentized residuals")
lineadd(h=0,color=3)
plot(PR$RESID/sqrt(PR$SIG2),type="pointline",main="Stdized residuals")
lineadd(h=0,color=3)
```



Příklad 2

```
// Model: Y = a1*1/x+a2*sqrt(x)+a3*x+a4*x^2
x=seq(.2,2,count=20)
y=1/x+sqrt(x)+x^2-x+normalr(20)/10
n=count(y)
// Data:
xg=seq(0.2,2,count=200)
//Definice exponentů modelu (bez absolutního členu):
deg=vec(-1,0.5,1,2)
M=count(deg)
// Výpočet regrese:
pr=polyreg(x,y,xnew=xg,poldeg=deg)
plot(x,y,main="Průběh zátěže: ",labx="Protažení",laby="Napětí")
plotadd(xg,PR$YNEW,type="line")
plotadd(xg,PR$YNEW+PR$CI,type="line",color=3)
plotadd(xg,PR$YNEW-PR$CI,type="line",color=3)
plot(x,PR$HATDIAG,type="pointline",main="Vliv jednotlivých měření")
lineadd(h=2*M/N,color=3)
// Sestavení tabulky výsledků
table=bind("A"+(1:M),deg,round(PR$A,3),round(sqrt(diag(PR$VARA)),3))
tableh=bind("Koeficient","Mocnina","Odhad","Sm.odch")
// Tisk tabulky výsledků do protokolu:
print(tableh,\n,table)
```



Koeficient	Mocnina	Odhad	Sm.odch
A1	-1.0	1.017	0.035
A2	0.5	0.995	0.575
A3	1.0	-1.0	0.714
A4	2.0	0.996	0.168

Viz také

`LINREG, LOCALREG, SPLINE1, SPLINE2`

POS(S1,S2)

Position in string. Pozice v textovém řetězci.

Vrací pozice všech výskytů řetězce S1 v řetězci S2 v podobě číselného vektoru. Pokud se S1 v S2 nevyskytuje, vrací hodnotu 0.

Povinné argumenty

S1, S2: textový řetězec

Příklad

```
>pos("a","Sagarmatha")
```

```
2 4 7 10

>pos("AA","AAAA")
1 2 3
```

Viz také

REPLACE, SUBSTR, ASTEXT, LENGTH

POWER(X1,X2)

Power of a real number. Mocnina reálného čísla.

Mocnina X1^X2.

Povinné argumenty

X1, X2: reálná čísla, nebo vektory, mocněnec a mocnitel. Pokud není mocnitel X2 celé číslo, musí být X1 nezáporné. Jsou-li X1 a X2 vektory, musí mít stejnou délku, výsledkem je pak vektor s prvky X1[1]^X2[1], X1[2]^X2[2], atd.

Příklad

```
>power(10,1:6)
10
100
1000
10000
100000
1000000

>power(1:5,2)
1
4
9
16
25
```

Viz také

INTPOWER, ^, EXP

PRINT(par1[, ..., parN][FILE=, APPEND=1|0, DELIMITER="\t"])

Print to output sheet. Tisk do výsledkového listu v okně Protokol.

Do aktuálního listu v okně Protokol, nebo do souboru vytiskne objekty par1, … parN. Při tisku do listu lze použít v seznamu řídicí znaky pro posun o buňku vpravo (\t) a přechod na nový řádek (\n). První tisk při každém každém spuštění skriptu smaže obsah listu a začne tisk od prvního řádku listu. Objekty typu vektor, nebo matice se tisknou v podobě tabulky, objekty typu list lze tisknout pouze po jednotlivých objektech. Použití PRINT() bez argumentů je formálně možné, nic však netiskne.

Cílový list, do kterého se tiskne, má implicitně název shodný s názvem skriptu, ze kterého se tiskne. Chceme-li tisknout do jiného listu, je nutné jej vytvořit v rámci téhož spuštění skriptu příkazem PRINTSHEET. Tiskne-li se do souboru, který ještě neexistuje, je nutné zadat hodnotu argumentu APPEND=0.

Nepovinné parametry

par1, par2, …: čísla, řetězce, vektory, nebo matice. Objekty, které se mají vytisknout.

FILE: Textový řetězec, název textového souboru, do něhož se má tisknout (zapisovat).

APPEND: Číslo 0, nebo 1, implicitně 1. Má-li se cílový soubor před zápisem vymazat, zadá se APPEND=0, má-li se pokračovat v zápisu do již existujícího souboru, zadá se APPEND=1. DELIMITER: Textový řetězec, oddělovač hodnot zapisovaných do souboru, implicitní hodnota je tabulátor, "\t".

Příklad

```
print(bind(1:5,sqrt(1:5)))
```

1	1
2	1.414213562
3	1.732050808
4	2
5	2.236067977

```
printsheet(NAME="ASDF")
print("Letter",\t,"No",\t,"i^2",\t,"i^3",\t,"i^4",\t,"log
i",\n,\n)
for(i=1,10)
{
print(letters("A",i),\t,"**"+i+"**",\t,i*i,\t,i^3,\t,i^4,\t,
log(i),\n)
print()
}
```

Letter	No	i^2	i^3	i^4	log i
A	**1**	1	1	1	0
B	**2**	4	8	16	0.3010299957
C	**3**	9	27	81	0.4771212547
D	**4**	16	64	256	0.6020599913
E	**5**	25	125	625	0.6989700043
F	**6**	36	216	1296	0.7781512504
G	**7**	49	343	2401	0.84509804
H	**8**	64	512	4096	0.903089987
I	**9**	81	729	6561	0.9542425094
J	**10**	100	1000	10000	1

```
// Tisk do souboru:
print(bind(1:10,sample(1950:2010,10)),file="C:\temp\QCE_outp
ut.txt",append=0)
print(\n,"********",\n,file="C:\temp\QCE_output.txt",append=
1)
```

```
for(i=1,10)
{
print(i+10,\t,sample(1950:2010,1),\n,file="C:\temp\QCE_outpu
t.txt",append=1)
}
```

QCE_output.txt:

```
1  1982
2  1999
3  1962
4  1963
5  1997
6  1980
7  1990
8  1989
9  1970
10 1958
********
11 2006
12 1985
13 2001
14 1996
15 1972
16 2002
17 1995
18 1972
19 2003
20 2008
```

Viz také

COPY, EXPORT, PLOT, PRINTSHEET

PRINTSHEET([NAME=, COLWIDTH=, ROWHEIGHT])

Create new print sheet. Vytvoří nový list pro tisk v okně Protokol.

V rámci jednoho spuštění skriptu se výstupy příkazu PRINT zapisují do definovaného listu. Bez použití příkazu PRINTSHEET se výstupy zapisují do listu s názvem shodným s názvem skriptu. Do tohoto listu lze směrovat výstup příkazem bez parametrů, PRINTSHEET(). Provedením příkazu PRINTSHEET(NAME=jmeno) se případný obsah již existujícího listu „jmeno“ vymaže.

Nepovinné argumenty

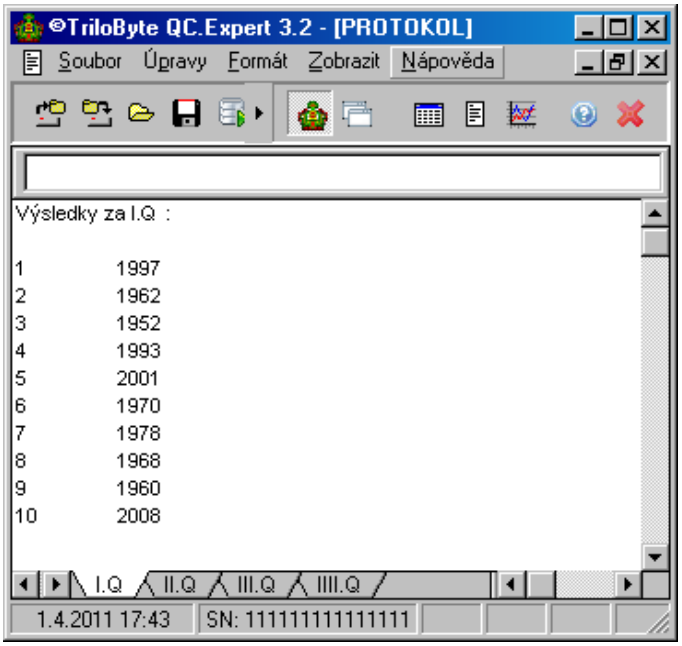
NAME: Textový řetězec, název listu v okně Protokol, do něhož se bude zapisovat příkazem PRINT.

COLWIDTH: Kladné reálné číslo, šířka sloupců listu v bodech, implicitně COLWIDTH=235.

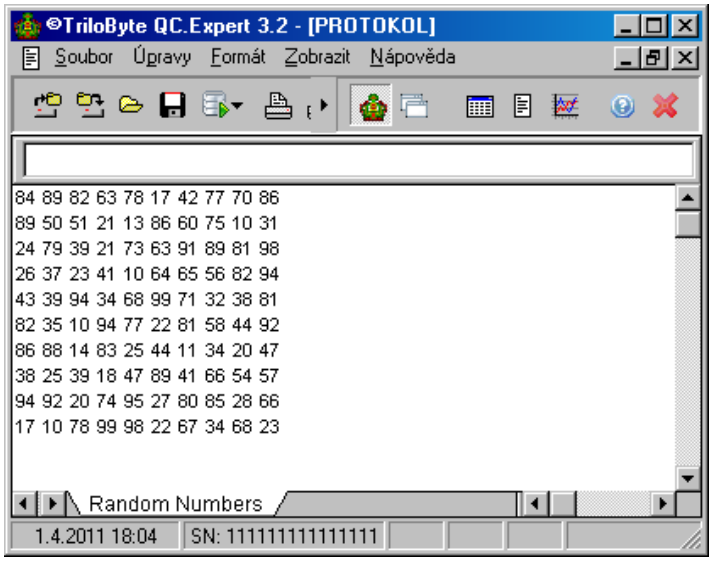
ROWHEIGHT: Kladné reálné číslo, výška řádků v bodech, implicitně ROWHEIGHT =22.

Příklad

```
delete(kva)
for(i=1,4)
{
kva[i]=letters("I",rep(1,i))
}
kva=kva+".Q"
for(i=1,4)
{
printsheet(NAME=kva[i])
print("Výsledky za "+kva[i]," :",\n,\n)
print(bind(1:10,sample(1:20,10)))
}
```




```
printsheet(NAME="Random Numbers",colwidth=150)
for(i=1,10)
{ print(transp(sample(10:99,10)),\n) }
```



Viz také

PRINT, COPY, EXPORT, PLOT

PROD (X)

Product. Součin.

Součin prvků vektoru nebo matice

Povinný argument

X: Číselný vektor, nebo matice

Příklad

```
>prod(1:10) // Faktorial 10
3628800

prod(dim(X)) // Počet prvků matice nebo vektoru X
```

Viz také

SUM, AVERAGE, FACT

PUTIMAGE(filename, X [, FORMAT=24|1|4|8|16|32])

Create an save BMP bitmap from data matrix. Vytvoří grafickou bitmapu a uloží do souboru BMP.

Z matice X (NxM) se sestrojí grafická bitová RGB mapa a uloží se jako soubor typu BMP. Přípona souboru musí být součástí názvu souboru. Každá hodnota v matici odpovídá jednomu pixelu obrázku. Velikost obrázku bude tedy stejná jako rozměry matice X. Kódování barev odpovídá 24-bitovému systému RGB, barva se skládá ze tří intenzit: R (red – červená), G (green – zelená) a B (blue – modrá). Každá z intenzit má celočíselnou hodnotu od 0 do 255. Nula znamená nepřítomnost, 255 znamená maximální jas barvy. Intenzity se kombinují do jediného čísla podle vztahu R + 256*G + 65536*B. Maximální intenzity všech tří barevných kanálů (R=255, G=255, B=255) dají bílou. Nulové intenzity odpovídají černé. Červená barva má tedy kód 255, zelená 65280, modrá 16711680.

Povinné argumenty

filename: Text, název souboru včetně cesty a přípony BMP.

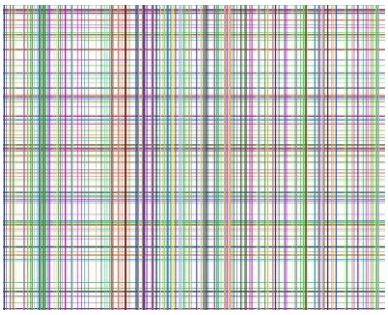
X: Numerická matice obsahující celočíselné hodnoty od 0 do 16777215, Hodnoty X[i,j] odpovídají požadované barvě pixelu [i,j] v obrázku. Pozice [0,0] odpovídá levému hornímu rohu obrázku, [N,0] odpovídá levému dolnímu rohu obrázku, [0,M] odpovídá pravému hornímu rohu obrázku, [N,M] odpovídá pravému dolnímu rohu obrázku.

Nepovinné argumenty

FORMAT: celočíselná hodnota, hloubka barvy. Přípustné hodnoty jsou 1, 4, 8, 16, 24, 32. Implicitní hodnota je 24 (16777216 barev). Tento argument ovlivňuje kvalitu obrázku a velikost souboru.

Příklad

```
n=480;m=640
x%=matrix(rep(16777215,n*m),ncols=m)
k=sample(1:n,100)
for(i=k)
{ r=int(random(1)*16000000)
x%[i,]=r
}
k=sample(1:m,200)
for(i=k)
{ r=int(random(1)*16000000)
x%[,i]=r
}
putimage("c:\temp\linky.bmp",x%)
```



Viz také

GETIMAGE, EXPORTGRAPH

PUTSHEET(P1, P2[,HEADER=, AUTOSIZE=0|1])

Put variable to data sheet. Vloží obsah proměnné do listu datového editoru.

Zkopíruje proměnnou P1 do nového datového listu QCExpertu s názvem P2. Pokud list s tímto názvem existoval, bude přepsán bez upozornění.

Povinné argumenty

P1: Numerická, nebo textová hodnota, vektor, nebo matice
P2: Textový řetězec, název listu v datovém editoru QCExpertu.

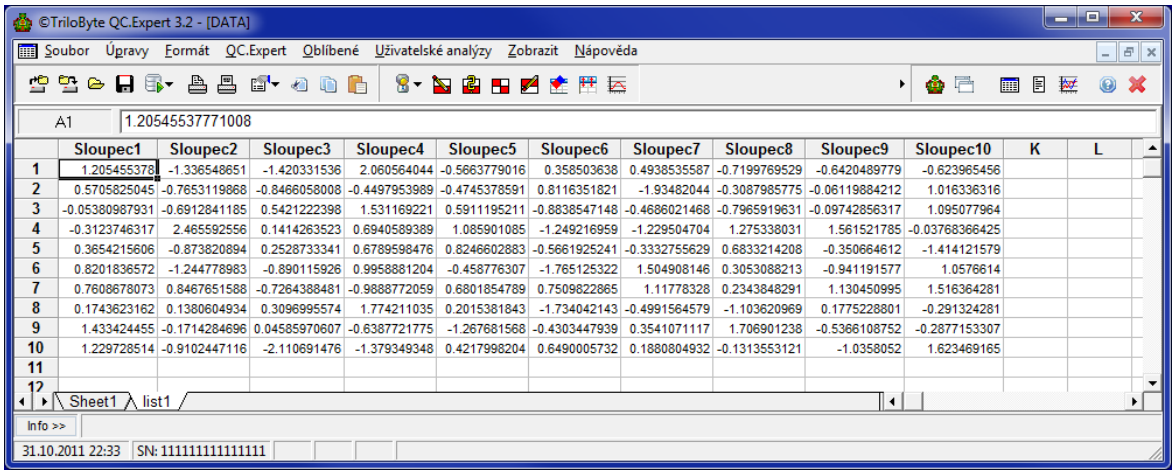
Nepovinné argumenty

HEADER: Textový vektor obsahující názvy sloupců pro datový list. Prvky vektoru se vloží jako názvy sloupců do datového listu P2. Délka vektoru nemusí odpovídat počtu sloupců proměnné P1.

AUTOSIZE: Číselná hodnota 0, nebo 1. Logická hodnota, která určí, zda se má nastavit automaticky šířka sloupců podle obsahu výsledného datového listu.

Příklad

```
a=matrix(normalr(100),ncols=10)
hh="Sloupec"+(1:10)
putsheet(a,"list1",header=hh,autosize=1)
```



	Sloupec1	Sloupec2	Sloupec3	Sloupec4	Sloupec5	Sloupec6	Sloupec7	Sloupec8	Sloupec9	Sloupec10	K	L
1	1.205455378	-1.336548651	-1.420331536	2.060564044	-0.5663779016	0.358503638	0.4938535587	-0.7199769529	-0.6420489779	-0.623965456		
2	0.5705825045	-0.7653119868	-0.8466058008	-0.4497953989	-0.4745378591	0.8116351821	-1.93482044	-0.3087985775	-0.06119884212	1.016336316		
3	-0.05380987931	-0.6912841185	0.5421222398	1.531169221	0.5911195211	-0.8838547148	-0.4686021468	-0.7965919631	-0.09742856317	1.095077964		
4	-0.3123746317	2.465592556	0.1414263523	0.6940589389	1.085901085	-1.249216959	-1.229504704	1.275338031	1.561521785	-0.03768366425		
5	0.3654215606	-0.873820894	0.2528733341	0.6789598476	0.8246602883	-0.5661925241	-0.3332755629	0.6833214208	-0.350664612	-1.414121579		
6	0.8201836572	-1.244778983	-0.890115926	0.9958881204	-0.458776307	-1.765125322	1.504908146	0.3053088213	-0.941191577	1.0576614		
7	0.7608678073	0.8467651588	-0.7264388481	-0.9888772059	0.6801854789	0.7509822865	1.11778328	0.2343848291	1.130450995	1.516364281		
8	0.1743623162	0.1380604934	0.3096995574	1.774211035	0.2015381843	-1.734042143	-0.4991564579	-1.103620969	0.1775228801	-0.291324281		
9	1.433424455	-0.1714284696	0.04585970607	-0.6387721775	-1.267681568	-0.4303447939	0.3541071117	1.706901238	-0.5366108752	-0.2877153307		
10	1.229728514	-0.9102447116	-2.110691476	-1.379349348	0.4217998204	0.6490005732	0.1880804932	-0.1313553121	-1.0358052	1.623469165		
11												
12												

Viz také

PRINT, GRAPHSHEET, PRINTSHEET, EXPORT, EXPORTGRAPH, GETSHEET, GETSHEETHEADER, GETSHEETNAMES

RANDOM(X)

Uniform random number (0,1). Náhodné číslo z rovnoměrného rozdělení z intervalu (0,1).

Vygeneruje vektor nebo matici pseudonáhodných čísel z rovnoměrného rozdělení z otevřeného intervalu (0,1) stejného rozměru jako argument X. Na samotných hodnotách vektoru X přitom nezáleží. Efektivní (zaručeně pseudonáhodný) počet desetinných míst je 9.

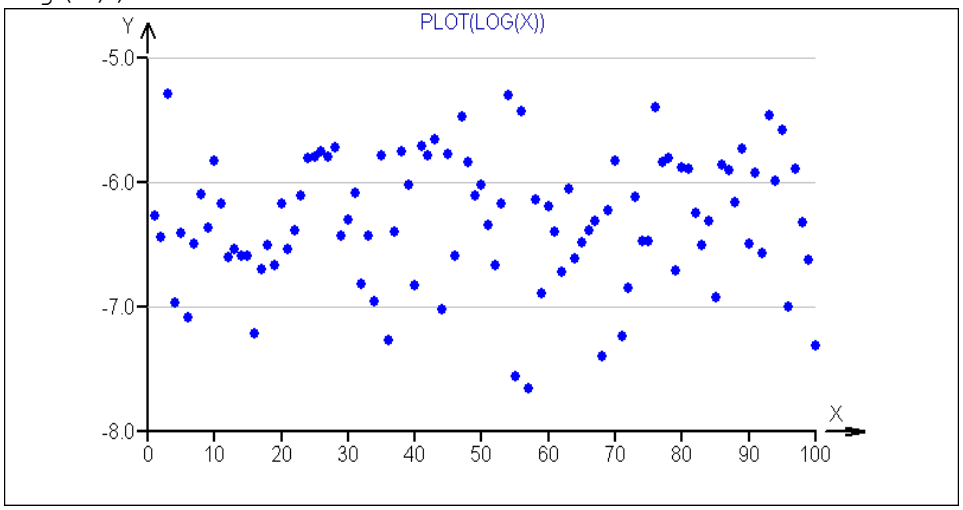
Povinný argument

X: Číselný vektor, nebo matice.

Příklad

```
>random(0) // Náhodné číslo
0.148753047920763

>random(1:5) // Náhodný vektor délky 5
0.514378784922883
0.927933687344193
0.121901484439149
0.671193222980946
0.307347321650013

>round(random(unit(5)),5) //Náhodná matice 5x5
0.40577   0.95026     0.80035     0.34749     0.10891
0.00575   0.99457     0.9912      0.13474     0.39285
0.32817   0.19191     0.51177     0.69527     0.67564
0.8511    0.1427      0.15281     0.6175      0.52945
0.87074   0.93055     0.47401     0.0525      0.69541

// *** Minimální hodnoty z milionu náhodných čísel ***
x=1:100
for(i=1,100)
```

```
{
x[i]=min(random(1:1000000))
}
plot(log(x))
```



Viz také

RND, NORMALR, SAMPLE

REP(X, N [, BYROWS=1|0])

Repeat value. Opakování hodnoty.

Argument X se opakuje N-krát. Argument BYROWS určuje směr opakování vodorovně (1, do sloupců), nebo svisle (0, do řádků). Výsledkem je vektor, nebo matice,

Povinné argumenty

X: Číslo, řetězec, vektor, nebo matice, která se bude opakovat.
N: Počet opakování. Objekt X se zopakuje N-krát.

Nepovinné argumenty

BYROWS: Směr, kterým se budou přidávat opakované objekty. Při BYROWS=0 se budou přidávat sloupce (rozšiřování doprava). Při BYROWS=1 (implicitní hodnota) se budou přidávat řádky (rozšiřování dolů).

Příklad

```
>rep(4,3)
4
4
4
>rep(1:3,2)
1
2
3
1
2
3
```

```
>rep(1:3,5,byrows=0)
1   1     1     1     1
2   2     2     2     2
3   3     3     3     3

>rep(unit(4),2,byrows=0)
1   0     0     0     1     0     0     0
0   1     0     0     0     1     0     0
0   0     1     0     0     0     1     0
0   0     0     1     0     0     0     1
```

Viz také: bindv, bind, seq

Viz také

BIND, BINDV, SEQ, VEC, ONES

REPLACE(S, Z, P)

Replace characters in string. Nahrazení znaků v řetězci.

Nahradí znak textového řetězce S na pozici P znakem, nebo řetězcem Z. Obsahuje-li řetězec Z více než 1 znak, nahradí se i znaky řetězce S na pozicích P+1, atd.

Povinné argumenty

S: textový řetězec v němž se má nahrazovat.
Z: Textový řetězec, kterým se má nahradit.
P: Pozice v řetězci S, od které se má nahrazovat.

Příklad

```
>replace("ABC","XXX",3)
"ABXXX"
// Nahrazení všech mezer pomlčkou:
>s1=replace(s," -",p)
>s="aaa bbb C  "
>p=pos(" ",s)
>s1=replace(s,"-",p)
>s
"aaa bbb C  "
>s1
"aaa-bbb-C--"
```

Viz také

REPLACES, POS, SUBSTR, LENGTH, ASTEXT, CHR

REPLACES(S, OLD, NEW [,ALL=0|1] [,NOCASE=0|1])

Replace substring by another substring in a string. Nahrazení znaků v řetězci.

Pokud řetězec S obsahuje řetězec OLD, nahradí se jeden nebo všechny (podle hodnoty ALL) výskyty řetězce OLD v řetězci S řetězcem NEW.

Povinné argumenty

S: Textový řetězec v němž se má nahrazovat.
OLD: Část textového řetězce, která se má nahradit.
NEW: Textový řetězec, kterým se má nahradit OLD.

Nepovinné argumenty

ALL: Logická hodnota 0 nebo 1. Je-li ALL=0, nahradí se pouze první výskyt OLD. Je-li ALL=1, nahradí se všechny výskyty OLD v řetězci S.

NOCASE: Logická hodnota 0 nebo 1. Je-li NOCASE=0, bere se při hledání výskytu OLD v S ohled na velikost písmen. Je-li NOCASE=1, ignoruje se rozdíl mezi malými a velkými písmeny.

Příklad

```
// Odstranění mezer:
>replaces("A BB FFFF R TTTT"," ","",all=1)
"ABBFFFFRTTTT"

// Nahrazení středníku tabulátorem:
>replaces("A;BB;FFFF;R;TTTT",";",chr(9),all=1)
"A  BB    FFFF  R     TTTT"
```

Viz také

REPLACE, POS, SUBSTR, ASTEXT, CHR

RETURN (...)

Return result of a function. Předání výsledku funkce.

Poslední příkaz algoritmu funkce, který jako svůj argument definuje výsledek dané funkce, který bude předán jako výsledek v instanci, která funkci volá. Tělo jedné funkce může obsahovat více příkazů RETURN.

Povinné argumenty

Libovolný výraz, jehož hodnota bude vrácena jako výsledek funkce (nejedná se vlastně v pravém smyslu o skutečný argument). Výraz může být číslo, nebo textový řetězec, vektor, matice, nebo seznam. Seznamem lze například vyřešit situaci, kdy je třeba, aby funkce vracela několik odlišných datových struktur, které nelze uložit do jednoho vektoru nebo matice. Příkaz RETURN lze použít pouze v těle definice funkce ve funkčním skriptovém listu, viz odstavec 6.3, str. 33. Nemá-li příkaz RETURN( ) v závorkách žádný argument, vrací funkce vždy nulu.

Příklad

```
function squareplusone(a)
{
return(a^2+1)
}
```

Viz také

FUNCTION

REV (X)

Reverse order of rows. Obrácení pořadí řádků.

Obrátí pořadí řádků argumentu.

Povinné argumenty

X: Vektor nebo matice.

Příklad

```
>A=matrix(vec(3,6,2,0,0,7,8,9,2,1,0,5),ncols=4)
>A
3   0     8     1
6   0     9     0
2   7     2     5

// Obrácené pořadí řádků:
>rev(A)
2   7     2     5
6   0     9     0
3   0     8     1

// Obrácené pořadí sloupců:
>transp(rev(transp(A)))
1   8     0     3
0   9     0     6
5   2     7     2
```

Viz také

SORT, ORDER, [], SPLIT

RND (X)

Uniform random number. Náhodné číslo z rovnoměrného rozdělení z intervalu (0, X).

Vygeneruje vektor nebo matici náhodných čísel z rovnoměrného rozdělení z otevřeného intervalu (0,X[i, j]) stejného rozměru jako argument X. Na samotných hodnotách vektoru X přitom nezáleží. Efektivní počet desetinných míst je 9.

Povinné argumenty

X: Vektor nebo matice reálných čísel x[i, j], která definují horní (případně spodní, pokud je x[i, j] záporné) mez rovnoměrného rozdělení. Druhá mez je vždy nula.

Příklad

```
>rnd(vec(5,-5,100))
4.71171439043246
-2.59433450177312
```

```
33.4084410453215

>int(rnd(unit(3)*100))
72  0     0
0   34    0
0   0     69
```

Viz také

RANDOM, NORMALR, SAMPLE

ROUND(X, N)

Round to N decimal places. Zaokrouhlení na N desetinných míst.

Zaokrouhlí X na N desetinných míst. Poslední platná číslice je řádu 10^{-N}.

Povinné argumenty

X reálné číslo, vektor, nebo matice.
N: celé číslo, nebo vektor celých čísel

Příklad

```
>round(24.555555,2)
24.56

>round(pi,-1:7)
0
3
3.1
3.14
3.142
3.1416
3.14159
3.141593
3.1415927

>round(rnd(10^(1:5)),6:2)
7.393619
29.70178
888.8496
1169.889
10868.99
```

Viz také

INT, FLOOR, TRUNC, FRAC

ROW(X, N)

N-th row. N-tý řádek matice nebo vektoru.

Vrací N-tý řádek matice nebo vektoru X. Je-li N vektor, vrací všechny řádky odpovídající hodnotám N ve tvaru matice, nebo vektoru.

Povinné argumenty

X: Matice nebo vektor
N: Kladné celé číslo, nebo vektor celých kladných čísel

Příklad

```
>A=bind(bind(vec(1,2,3),vec(10,20,30)),vec(100,200,300))
>a
1   10    100
2   20    200
3   30    300
>row(A,3)+row(A,1)
4   40    400
>row(A,3:1)
3   30    300
2   20    200
1   10    100
```

Viz také

COL, SPLIT, SPLITV, [], MATRIX

SAMPLE(X, N [, REPL=0|1])

Sample N values from X with or without replacement. Výběr N hodnot z vektoru nebo matice X s vracením, nebo bez vracení.

Provede náhodný výběr N prvků matice nebo vektoru, výsledkem je vždy vektor. Každý prvek X má stejnou pravděpodobnost, že bude vybrán. Je-li argument REPL=0 (implicitní hodnota), nemůže být již vybraný prvek vybrán znovu a maximální rozsah výběru je dán počtem prvků X. Je-li argument REPL=1, může být každý prvek X vybrán více než jednou a maximální rozsah výběru není omezen počtem prvků X.

Povinné argumenty

X: Vektor, nebo matice. Základní soubor, z něhož se prování výběr.
N: Celé číslo. Rozsah výběru. Je-li REPL=1, může být N větší, než počet prvků X.

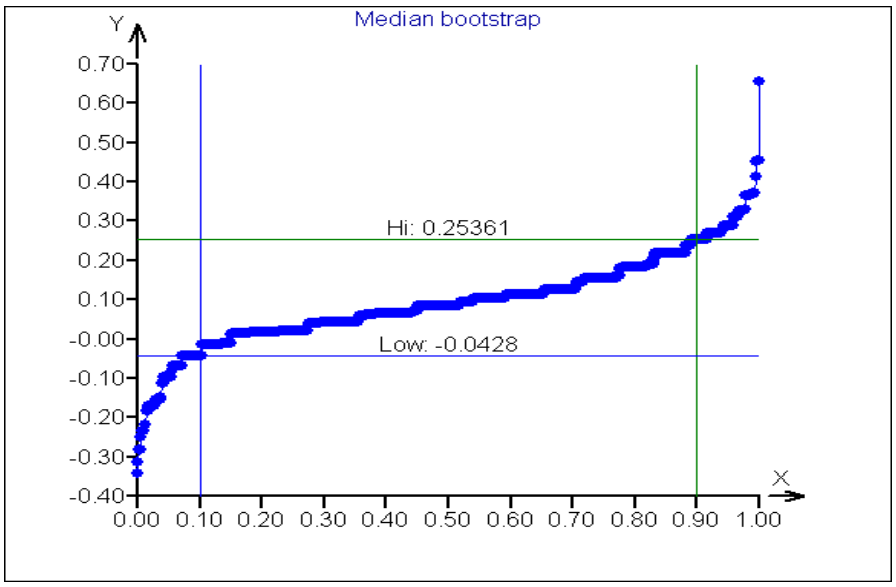
Nepovinné argumenty

REPL: Číslo 0 nebo 1, „replace“. Při REPL=0 (implicitní hodnota) se nemůže vybrat dvakrát tentýž prvek. Při REPL=1 se vybrané prvky v základním souboru ponechávají a mohou být vybrány vícekrát.

Příklad

```
// Náhodný výběr z lichých čísel:
>x=(1:5)*2-1
>transp(sample(X,10,repl=1))
3   5     5     5     9     3     3     5     9     1
```

```
// Bootstrapový odhad 80% intervalu spolehlivosti mediánu x.
x=normalr(30)
nb=1000
bs=rep(0,nb)
for(i=1,nb)
{
bs[i]=median(sample(x,30,repl=1))
}
bs=sort(1,bs)
@plot(seq(0,1,count=nb),bs,type="pointline",
main="Median bootstrap");
a=vec(0.1,0.9)
b=round(vec(bs[0.1*nb],bs[0.9*nb]),5)
@plottextadd(vec(0.5,0.5),b+0.03,
vec("Low: "+b[1],"Hi: "+b[2]));
lineadd(v=a)
lineadd(h=b)
// Interval spolehlivosti mediánu:
>b
-0.0428
0.25361
```



Viz také

REP, RANDOM, NORMALR, SEQ

SENDMAIL([message_text1[,...,message_textn],] ACCOUNT=List(HOST=, USER=, PASSWORD=, FROMEMAIL=, FROMNAME=), TOEMAIL=email|VEC(emails), [SUBJECT=, ATTACHMENT=attachment|VEC(attachments)])

Send an e-mail with attachment through SMTP given sender account details. Poslání e-mailu s možnou přílohou pomocí SMTP.

Pošle e-mail na všechny adresy v textovém vektoru TOEMAIL pomocí protokolu SMTP. Zpráva může obsahovat jednu nebo více příloh v podobě řetězce odkazujícího na přiložený soubor.

Povinné argumenty

ACCOUNT: Seznam obsahující čtyři textové argumenty, HOST: Název nebo IP adresa SMTP serveru; USER: uživatelské jméno účtu; PASSWORD: Heslo SMTP; FROMMAIL: E-mailová adresa odesílatele (nemusí odpovídat skutečnosti); FROMNAME: Jméno odesílatele.

TOEMAIL: Řetězec, nebo vektor řetězců obsahující emaily adresáta nebo adresátů.

Nepovinné argumenty

Řetězec nebo řetězce oddělení čárkami obsahující text zprávy. Syntaxe textu zprávy je podobná, jako u příkazu PRINT, lze používat řídicí kódy \n (nový řádek) a \t (tabulátor).

SUBJECT: Řetězec s předmětem zprávy.

ATTACHMENT: Řetězec obsahující cestu a název souboru, který má být přiložen ke zprávě. Je-li tento argument textový vektor, přidají se k zprávě všechny soubory uvedené v tomto vektoru.

Příklad

```
@sendmail("The report is attached",\n,"Regards, J.B."
ACCOUNT=List(HOST="0.0.0.111",USER="username",
PASSWORD="my_password",
FROMEMAIL="my_name@company.com",FROMNAME="My Name"),
TOEMAIL="my.boss@company.com",SUBJECT="DW_Message",
ATTACHMENT="C:\temp\report.pdf");
```

SEQ(X1, X2 [, COUNT= | STEP=])

Sequence from X1 to X2. Numerická sekvence od X1 do X2.

Vrací numerický vektor ekvidistantních hodnot s počátkem X1, koncem X2. Počet hodnot je dán hodnotou argumentu COUNT, krok je dán hodnotou argumentu STEP, oba argumenty nemohou být zadány současně.

Povinné argumenty

X1, X2: číselná hodnota začátku a konce sekvence.

Nepovinné argumenty

COUNT: Celé kladné číslo větší než 1, počet hodnot sekvence.

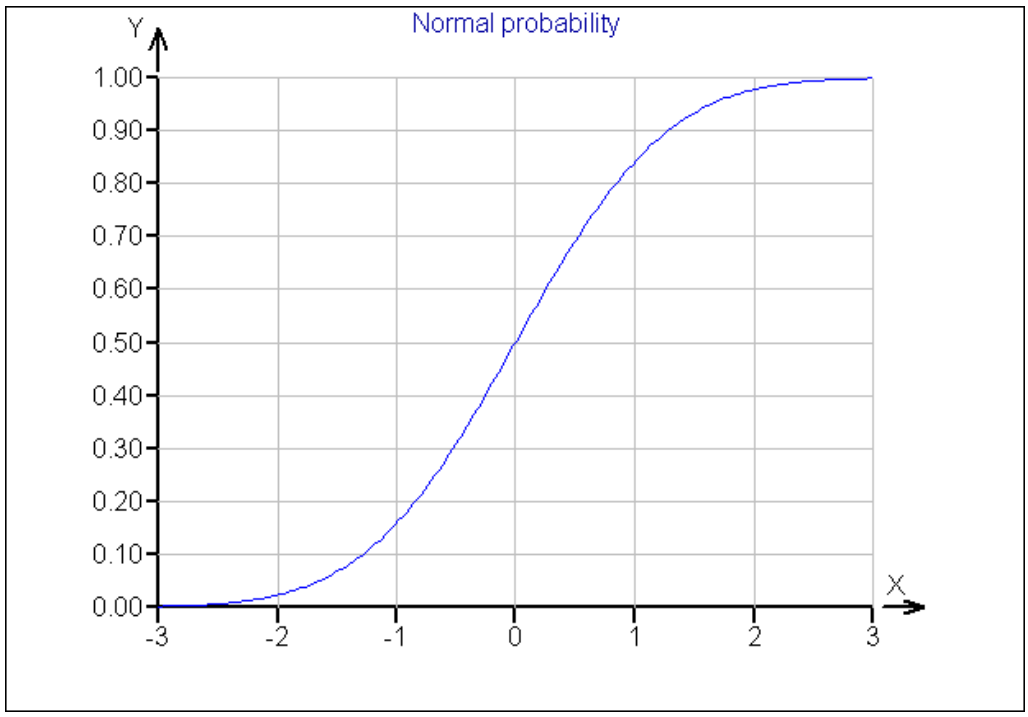
STEP: Reálné kladné číslo, krok hodnot sekvence, poslední hodnota sekvence je vždy menší nebo rovna X2.

Argumenty COUNT a STEP nemohou být zadány současně.

Příklad

```
>seq(0,1,count=10)
0
0.111111111111111
0.222222222222222
```

```
0.333333333333333
0.444444444444444
0.555555555555556
0.666666666666667
0.777777777777778
0.888888888888889
1
>seq(0,1,step=0.3)
0
0.3
0.6
0.9

x=seq(-3,3,count=200)
plot(x,normalp(x),type=line,main="Normal probability")
```



Viz také

REP, RANDOM, VEC

SIGN(X)

Sign of a real number. Znaménko čísla.

Funkce vrací –1, je-li argument X záporný, 0, je-li argument X nula, +1, je-li argument X kladný.

Povinný argument

X: reálné číslo, vektor, nebo matice. Pokud je X vektor, nebo matice, je výsledkem rovněž vektor nebo matice.

Příklad

```
>sign(-5)
-1
>sign(1e8)
1
>transp(sign(normalr(10)))
-1  -1    1     -1    1     1     -1    1     1     1
```

Viz také

ZERO, INT

SIN(X)

Sine. Sinus.

Funkce sinus, sin(x)

Povinný argument

X: reálné číslo (X je v radiánech), vektor, nebo matice

Příklad

```
>sin(pi/(1:4))
1.22460635382238E-16
1
0.866025403784439
0.707106781186547
```

Viz také

COS, TAN, ASIN

SINH(X)

Hyperbolic sine. Hyperbolický sinus.

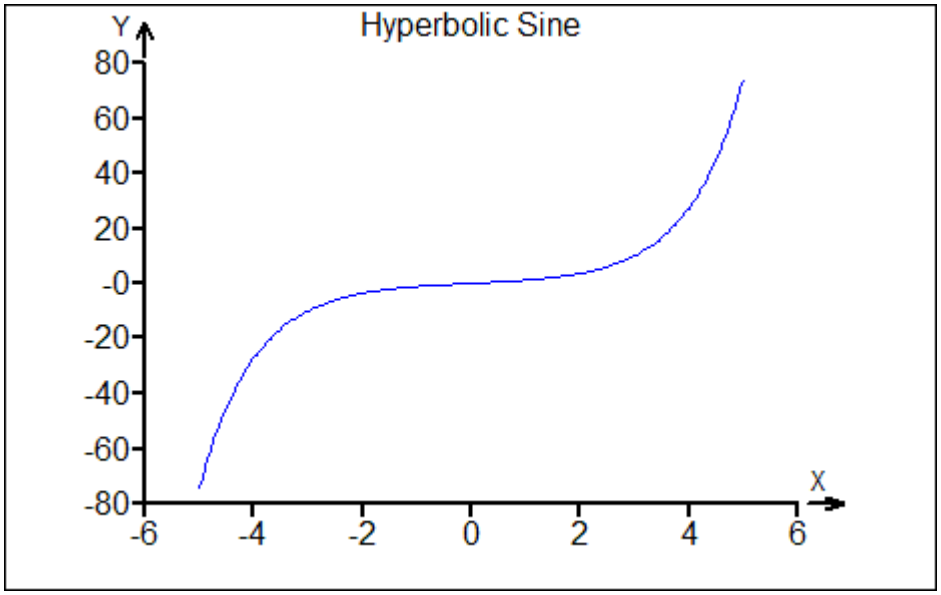
Funkce $\sinh(x) = \frac{e^x - e^{-x}}{2}$

Povinný argument

X: reálné číslo, vektor, nebo matice

Příklad

```
x=seq(-5,5,count=200)
plot(x,sinh(x),type="line",main="Hyperbolic Sine")
```



Viz také

COSH, TANH, ASINH

SMALL(S)

Convert string to small letters. Konverze řetězce na malá písmena.

Všechny velká písmena obsažená v řetězci S jsou převedena na malá. Ostatní znaky nejsou změněny.

Povinný argument

S: řetězec, vektor, nebo matice řetězců.

Příklad

```
>small("AbCdEfGh")
"abcdefgh"

>s=vec("SsSs","ĚŘŤŽŠĎČŇ","1234567890")
>small(s)
"ssss"
"ěřťžšďčň"
"1234567890"
```

Viz také

CAPS, LETTERS, ASTEXT

SORT(N, X)

Sort rows of X. Třídění řádků X.

Setřídí řádky vektoru, nebo matice podle N-tého sloupce. Násobné třídění (tj. např. nejprve podle 1. sloupce, pak podle 3. sloupce, apod.) je možné, je-li N vektor s čísly sloupců.

Povinné argumenty

N: celé číslo, nebo vektor celých čísel odpovídající sloupcům matice podle nichž se má třídit. Je-li N kladné, třídí se vzestupně, je-li N záporné, třídí se sestupně. Má-li se například třídit matice X nejdříve podle 1. sloupce a pak podle 3. sloupce, zadá se jako první argument vec(1,3). Třídí-li se sloupcový vektor, nebo se třídí podle jednoho sloupce, musí být N rovno 1, popř. číslu sloupce matice.

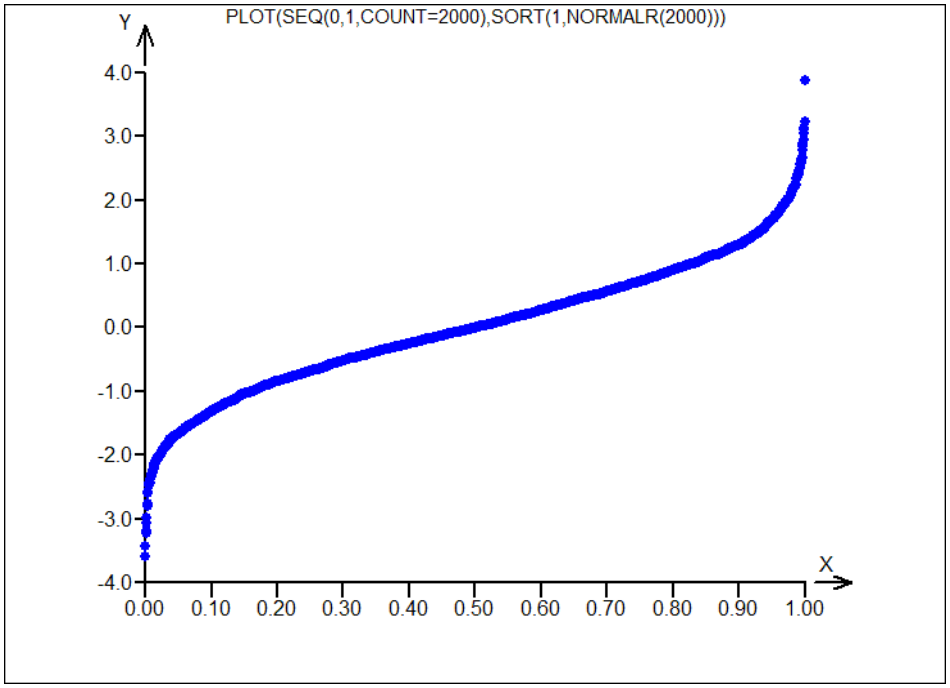
X: Sloupcový vektor, nebo matice reálných čísel.

Příklad

```
>A=bind(vec(1,3,3,4,2,1,1),vec(7,6,5,4,3,2,1))
```

>A		>sort(1,A)		>sort(2,A)		>sort(1:2,A)	
1	7	1	7	1	1	1	1
3	6	1	2	1	2	1	2
3	5	1	1	2	3	1	7
4	4	2	3	4	4	2	3
2	3	3	6	3	5	3	5
1	2	3	5	3	6	3	6
1	1	4	4	1	7	4	4

```
// Empirická kvantilová funkce normálního rozdělení:
plot(seq(0,1,count=2000),sort(1,normalr(2000)))
```



Viz také

ORDER, SEQ, MIN, MAX, REP

SPLINE1(X, Y, X1, W)

SPLINE2(X, Y, X1)

Interpolation spline. Interpolační spline.

Pro zadaná data X a Y vrací interpolované hodnoty ležící na spline-křivce v bodech X1.

Povinné argumenty

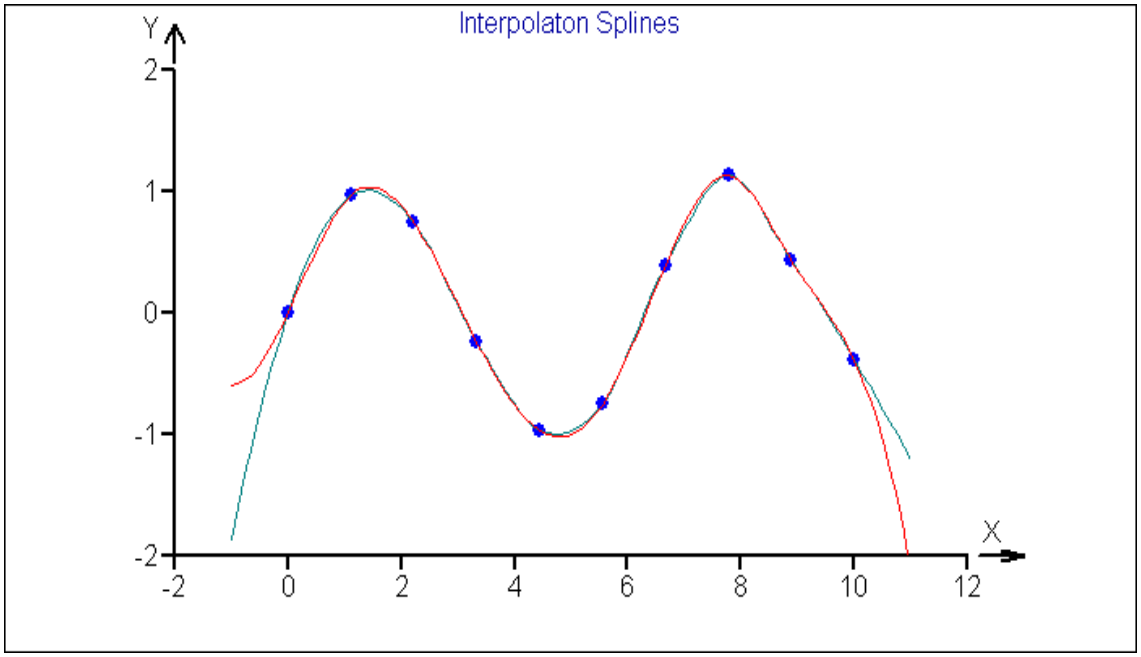
X, Y: Reálné vektory stejné délky. Dvojice dat (x_i ,y_i), které mají být proloženy křivkou.

X1: Reálný vektor, polohy bodů, pro něž se mají počítat interpolované hodnoty. Délka vektoru X1 nezávisí na délce X a Y.

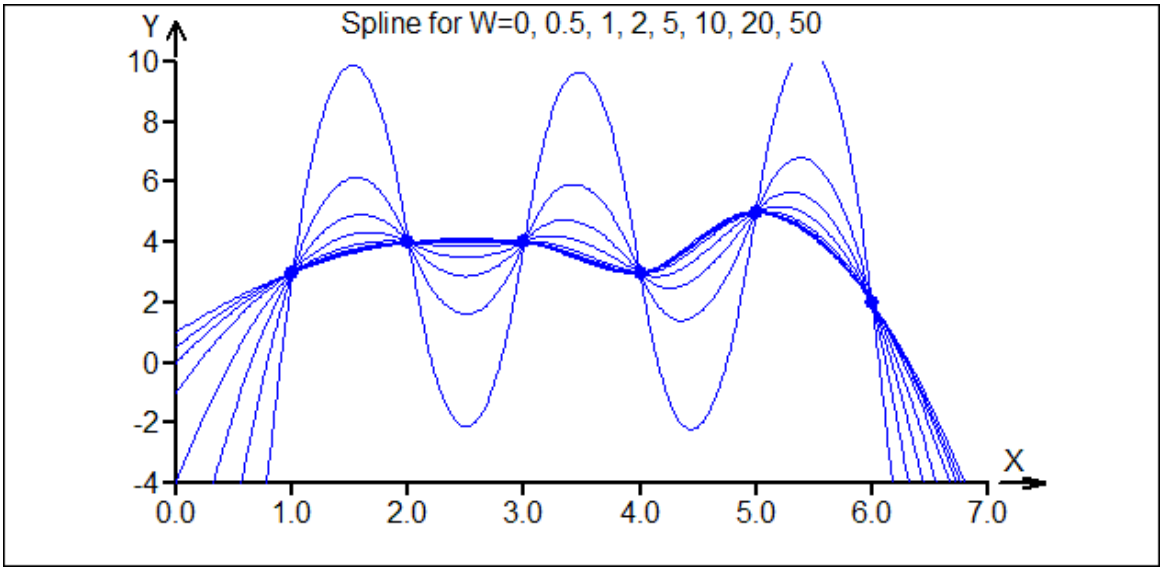
W: Reálné číslo, omezení na druhou derivaci křivky. Pro W=0 je výsledná křivka nejhladší.

Příklad

```
N=10
x=seq(0,10,count=N)
y=sin(x)+normalr(N)/10
x1=seq(-1,11,count=10*N)
plot(x,y,main="Interpolaton Splines")
plotadd(x1,spline1(x,y,x1,0),type="line",color=5)
plotadd(x1,spline2(x,y,x1),type="line",color=3)
```




```
x=1:6
y=vec(3,4,4,3,5,2)
x1=seq(0,7,count=300)
plot(x,y,main="Spline for W=0,0.5,1,2,5,10,20,50")
for(w=vec(0,0.5,1,2,5,10,20,50))
{
y1=spline1(x,y,x1,w)
plotadd(x1,y1,type="line")
}
```



Viz také

PLOT3DSPLINE

SPLIT(X, N)

Split horizontal. Odříznutí pravé části matice.

Ponechá pouze prvních N sloupců matice nebo vektoru X. Je-li N větší, než počet sloupců X, je výsledkem původní vektor, nebo matice X.

Povinné argumenty

X: Matice nebo vektor.
N: Počet sloupců, které má funkce z původního objektu vrátit.

Příklad

```
>split(unit(4),2)
1       0
0       1
0       0
0       0
```

Viz také

SPLITV, [], BIND, BINDV, ROW, COL

SPLITV(X, N)

Split vertical. Odříznutí spodní části matice.

Ponechá pouze prvních N řádků matice nebo vektoru X. Je-li N větší, než počet řádků X, je výsledkem původní vektor, nebo matice X.

Povinné argumenty

X: Matice nebo vektor.

N: Počet řádků, které má funkce z původního objektu vrátit.

Příklad

```
>splitv(unit(4),2)
1       0       0       0
0       1       0       0
```

Viz také

SPLIT, [], BIND, BINDV, ROW, COL

SQR (X)

Square of a number. Čtverec čísla (druhá mocnina).

Funkce x^2, shodná s X^2
Povinný argument
X: reálné číslo, vektor, nebo matice.

Příklad

```
>sqr(sqrt(2))-2
4.44089209850063E-16
```

Viz také

SQRT, POWER, INTPOWER, ^

SQRT (X)

Square root of a number. Druhá odmocnina čísla.

Funkce $\sqrt{X}$
Povinný argument
X: nezáporné reálné číslo, vektor, nebo matice.

Příklad

```
sqr(sqrt(2))-2
4.44089209850063E-16
```

Viz také

SQR, POWER, INTPOWER, ^

STOP

Stop execution. Zastavení programu.

Zastaví provádění skriptu, ekvivalentní stisknutí *F10*, nebo tlačítka *Stop – F10*.

Příklad

```
// Ilustrační příklad ošetření chyby:
```

```
r=normalr(1)
if (lt(r,0)) {
message("Chyba - záporné X:",\n,"X= ",r);stop
}
print(sqrt(r))
```

Viz také

PAUSE, TRACEON

STRDATE (M)

STRDATETIME (M)

STRTIME (M)

Date and time stamp. Datové a časové razítko.

Tyto tři funkce vrací datum, čas a datum, resp. čas odvozené od daného okamžiku ve formě textového řetězce. Argument je počet minut, které se přičtou k současnému času.

Povinné argumenty

M: Reálné číslo, čas v minutách, který se přičte k současnému systémovému času. M může být záporné.

Příklad:

```
>STRDATE(0)
"11.11.2011"
>STRDATETIME(0)
"11.11.2011 11:11:09"
>STRTIME(0)
"23:59:59"

print("Čas zpracování:",\t,STRDATETIME(0))
```

Čas zpracování:	1.4.2011 12:12:12

```
// Zjištění doby výpočtu
a=0
t1=strtime(0)
for(i=1,50000)
{
a=a+i
}

>t1
"11:37:00"
>strtime(0)
"11:37:04"
```

Viz také

ASTEXT

STUDENTP(x, N)

Student cumulative distribution function. Distribuční funkce t-rozdělení.

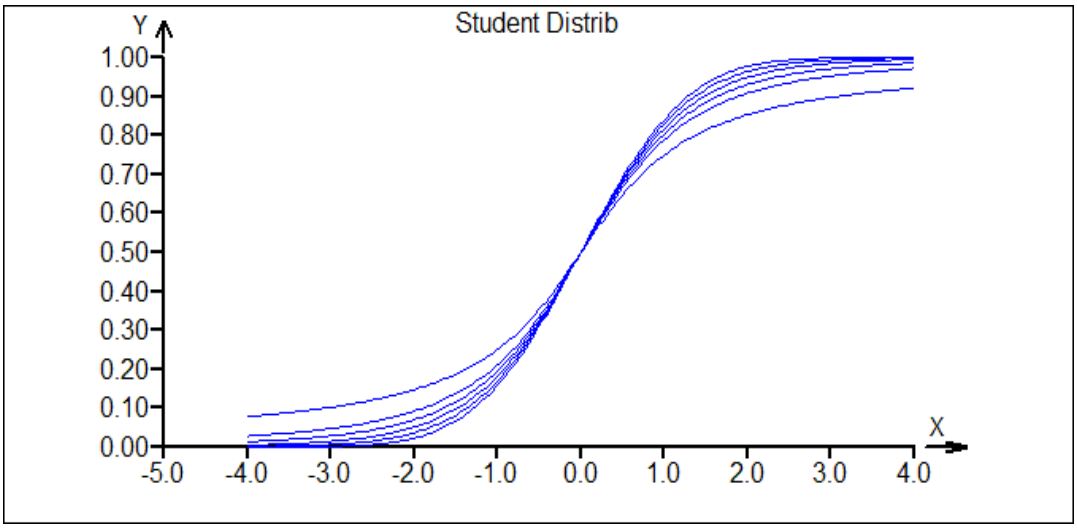
Hodnota distribuční funkce Studentova t-rozdělení s N stupni volnosti.

Povinné argumenty

X: reálné číslo, nebo vektor, kvantil t-rozdělení.
N: Počet stupňů volnosti t-rozdělení

Příklad

```
>studentp(-2, 5)
0.0509697394149279

// Grafy distribuční funkce t-rozdělení
// s 1, 2, 3, 5, 10 a 500 stupni volnosti
x=seq(-4,4,count=200)
plot(x,studentp(x,1),type="line", main="Student Distrib")
for(i=vec(2,3,5,10,500))
{
plotadd(x,studentp(x,i),type="line")
}
```



Viz také

STUDENTQ, NORMALP, NORMALQ

STUDENTQ(P, N)

Student quantile function. Kvantilová funkce t-rozdělení.

Hodnota kvantilové funkce Studentova t-rozdělení s N stupni volnosti.

Povinné argumenty

P: reálné číslo, 0 < P < 1, nebo vektor, pravděpodobnost
N: Počet stupňů volnosti t-rozdělení

Příklad

```
// kvantily 0.025 a 0.975 (95% interval spolehlivosti)
// pro (nulový) průměr ze 3 dat s rozptylem 1 (N-1=2):
>studentq(vec(0.025,0.975),2)/sqrt(3)
-2.48413771173866
2.48413771173866
```

Viz také

STUDENTP, NORMALP, NORMALQ

SUBSTR(S, N)

Substring of a string. Část textového řetězce.

Část textového řetězce S definovaná pořadím N

Povinné argumenty

S: Textový řetězec.
N: Celé číslo, nebo vektor celých čísel, pořadová čísla znaků, které se mají vybrat z řetězce S.

Příklad

```
>substr("ABCDEFGH",2:5)
"BCDE"

// K-tice náhodných znaků
K=4
substr(letters("A", 1:26),sample(1:26, K, repl=1))
"OXZU"
```

Viz také

REPLACE, POS, LENGTH, LETTERS

SUM(X)

Sum of X. Součet prvků X.

Součet prvků vektoru, nebo matice X

Povinné argumenty

X: vektor nebo matice reálných čísel.

Příklad

```
// Součet čísel 1 až 100
>sum(1:100)
5050

// Stopa matice A^TA:
>A=int(10*random(unit(5)))
```

```
>sum(diag(transp(A)#A))
792
```

Viz také

AVERAGE, CUSUM, PROD, APPLY

SVDU(X) ; SVDD(X) ; SVDV(X)

Singular value decomposition. Singulární rozklad matice.

Rozklad matice **X**(*n* x *m*) **X** = **U D** $\mathbf{V}^T$, kde **U**(*n* x *m*) a **V**(*m* x *m*) jsou ortogonální matice a D(*m* x *m*) je diagonální matice se singulárními hodnotami na hlavní diagonále.

Funkce SVDU(X) vrací matici **U**, SVDD(X) vrací matici **D**, SVDV(X) vrací matici **V**.

Povinné argumenty

X: matice reálných čísel

Příklad

```
// Rozklad náhodné matice A(5 x 5):
>A=int(10*random(unit(5)))
>A
4   4     7     9     0
8   4     6     9     4
8   6     0     0     7
7   4     0     2     0
3   0     6     4     1

>U=SVDU(a)
>D=SVDD(a)
>V=SVDV(a)
>round(U,4)
-0.5139   -0.4571     -0.2295     -0.4843     -0.4897
-0.6442   -0.0959     0.1637      -0.051      0.7392
-0.3824   0.772       0.3554      -0.1645     -0.3232
-0.2956   0.3073      -0.8034     0.4155      -0.0111
-0.2956   -0.3024     0.3857      0.7505      -0.3304

>round(D,4)
22.4824   0      0      0      0
0   11.0063      0      0      0
0   0     4.3658       0      0
0   0     0     2.5263       0
0   0     0     0     1.7199

>round(V,4)
-0.5882   0.4384      -0.2822     0.593       0.1748
-0.3607   0.3316      -0.308      -0.5805     -0.5729
-0.4108   -0.5078     0.3872      0.3193      -0.567
-0.5425   -0.5062     -0.1502     -0.3899     0.5243
-0.2468   0.4287      0.8081      -0.2396     0.2118
```

```
// Rekonstrukce původní matice
// zaokrouhlená na 13 desetinných míst:
>round(U#D#transp(V),13)
4   4     7     9     0
8   4     6     9     4
8   6     0     0     7
7   4     0     2     0
3   0     6     4     1

// Singulární hodnoty A:

>diag(D)
22.4824282622287
11.0063489190708
4.3658195391934
2.52631411500781
1.71990105788584
```

Viz také

EIGENVAL, EIGENVEC, INV, PINV, DET

SVMC (X, Y[, KERNEL="LINEAR"|"POLYNOMIAL"|"RBF"|"TANH", PREDICT=0|1, NEWDATA=X1, TYPE="COST"|"NU", DEGREE=, GAMMA=, COST=, NU=, R=, SHRINK=1|0, PROBABILITY=0|1, CPLOT=0|1, BPLOT=0|1, PREDPLOT=0|1, ROCPLOT=0|1])

Support Vector Classification. Klasifikace modelem Support Vector Machines

Funkce vytvoří model SVM-C (klasifikace pomocí SVM) se zadaným jádrem pro zadané hodnoty prediktoru X a odezvy Y se zadanými parametry. Prediktor je numerická matice nebo vektor, odezva je nominální a může být reprezentována hodnotami textového vektoru. Pomocí argumentu NEWDATA lze zadat nové, jiné hodnoty prediktoru, který musí mít stejný počet sloupců jako X ale může mít jiný počet řádků. Pro NEWDATA jsou pak vypočítány predikované hodnoty odezvy včetně rozdělení pravděpodobnosti. Výsledkem funkce SVMC je seznam. Podrobněji je metoda SVM-C a význam jejích argumentů popsána v uživatelském manuálu QCExpert a v další doporučené literatuře.

Povinné argumenty

X: reálný číselný vektor (N x 1) nebo matice (N x M), obsahuje v řádcích hodnoty prediktoru.

Y: Textový, nebo číselný vektor (N x 1) obsahující L různých hodnot (úrovní faktoru) odezvy pro každý řádek X. Y musí mít stejný počet řádků jako X. Je-li Y číselný vektor, berou se jeho hodnoty jako textové řetězce, hodnoty Y mohou tedy být například celočíselné hodnoty, avšak ty se v klasifikaci považují pouze za označení různých stavů (úrovní) odezvy.

Příklad vstupních dat X a Y (zde N=6, M=4, L=3):

X				Y
2.6	2553	12.0	0.0224	modra
5.3	3564	17.4	0.0644	zelena
7.1	4701	14.3	0.0703	zelena
2.7	3386	18.3	0.0369	cervena
4.3	3703	18.1	0.0296	zelena
3.3	2994	16.3	0.0529	modra

Nepovinné argumenty

KERNEL: Textový řetězec obsahující typ použitého jádra SVM. Možné typy jádra jsou: "LINEAR" (lineární jádro), "POLYNOMIAL" (polynomické jádro, jehož stupeň je určen argumentem DEGREE), "RBF" (Gaussovské jádro, Radial Base Function), "TANH" (hyperbolický tangens, sigmoidální jádro). Není-li argument uveden, použije se implicitní hodnota "LINEAR".

PREDICT: Logická hodnota 0 nebo 1 určující zda se mají vypočítat predikované hodnoty pro NEWDATA. Implicitní hodnota je 0. Je-li zadána hodnota 1 a není zadán argument NEWDATA, dojde k chybě.

NEWDATA: Nová data pro predikci, data musí mít stejný počet sloupců jako X. Pro tato data funkce SVMC vypočítá predikované hodnoty odezvy.

TYPE: Textový řetězec "COST" nebo "NU", určuje metodu výpočtu ztrátové funkce. Je-li zadán typ "COST", ignoruje se případná hodnota argumentu NU, a naopak je-li zadán typ "NU", ignoruje se případná hodnota argumentu COST. Implicitní hodnota je "COST".

DEGREE: Je-li zadáno polynomické jádro (KERNEL="POLYNOMIAL"), určuje tento argument stupeň polynomu, jinak je ignorován. Hodnota parametru musí být kladné celé číslo, nejčastěji 1, 2, 3. Implicitní hodnota je 2.

GAMMA: Reálné číslo, parametr strmosti jádra, implicitní hodnota je 0, obvyklé hodnoty jsou od 0 do 10, případně i vyšší.

COST: Hodnota koeficientu ztrátové funkce. Argument je ignorován, je-li zadáno TYPE="NU". Implicitní hodnota je 1.

NU: Hodnota koeficientu ný (ν). Argument je ignorován, je-li zadáno TYPE="COST". Implicitní hodnota je 1.

R: Parametr polynomického a sigmoidálního jádra

SHRINK: Logická hodnota 0 nebo 1, zkrácení odhadu, někdy může vést ke snížení počtu support vektorů a mírnému zlepšení stability modelu.

PROBABILITY: Logická hodnota 0 nebo 1, je-li zadána hodnota 1, je ve výsledkovém seznamu funkce položka Probability, která obsahuje odhady pravděpodobností pro jednotlivé úrovně odezvy.

CPLOT: Logická hodnota 0 nebo 1, určuje, zda funkce SVMC vytvoří graf predikovanách úrovní pro X pokud má matice prediktoru X právě 2 sloupce. Při jiné dimenzi prediktoru se graf CPLOT nevytváří.

BPLOT: Logická hodnota 0 nebo 1, určuje, zda funkce SVMC vytvoří graf hranic predikovanách úrovní pro X pokud má matice prediktoru X právě 2 sloupce. Při jiné dimenzi prediktoru se graf BPLOT nevytváří. Tento graf má totožný význam jako CPLOT, pouze zvýrazňuje hranici mezi predikovanými klasifikacemi.

PREDPLOT: : Logická hodnota 0 nebo 1, určuje, zda funkce SVMC vytvoří graf predikovanách úrovní pro hodnoty zadané v argumentu NEWDATA. Vytváří se jen pokud má matice NEWDATA právě 2 sloupce. Při jiné dimenzi prediktoru se graf PREDPLOT nevytváří.

ROCPLOT: Logická hodnota 0 nebo 1, určuje, zda funkce SVMC vytvoří graf ROC.

Struktura výsledného seznamu

$Levels: Vektor obsahující rozdílné úrovně (hodnoty) odezvy v pořadí, v jakém se poprvé vyskytují ve vektoru odezvy Y.

$Misclass: Počet misklasifikací (chybných klasifikací) modelu.

$NewPrediction: Vektor obsahující predikované hodnoty odezvy pro zadané hodnoty v argumentu NEWDATA. Tato položka se vytváří pouze je-li zadán argument NEWDATA.

$NewProbability: Matice obsahující pravděpodobnosti jednotlivých možných úrovní (hodnot) odezvy pro zadané hodnoty v argumentu NEWDATA. Vektor $NewPrediction obsahuje ty úrovně, které mají nejvyšší pravděpodobnost. Pořadí sloupců odpovídá pořadí úrovní v položce $Levels. Tato položka se vytváří pouze je-li zadán argument NEWDATA.

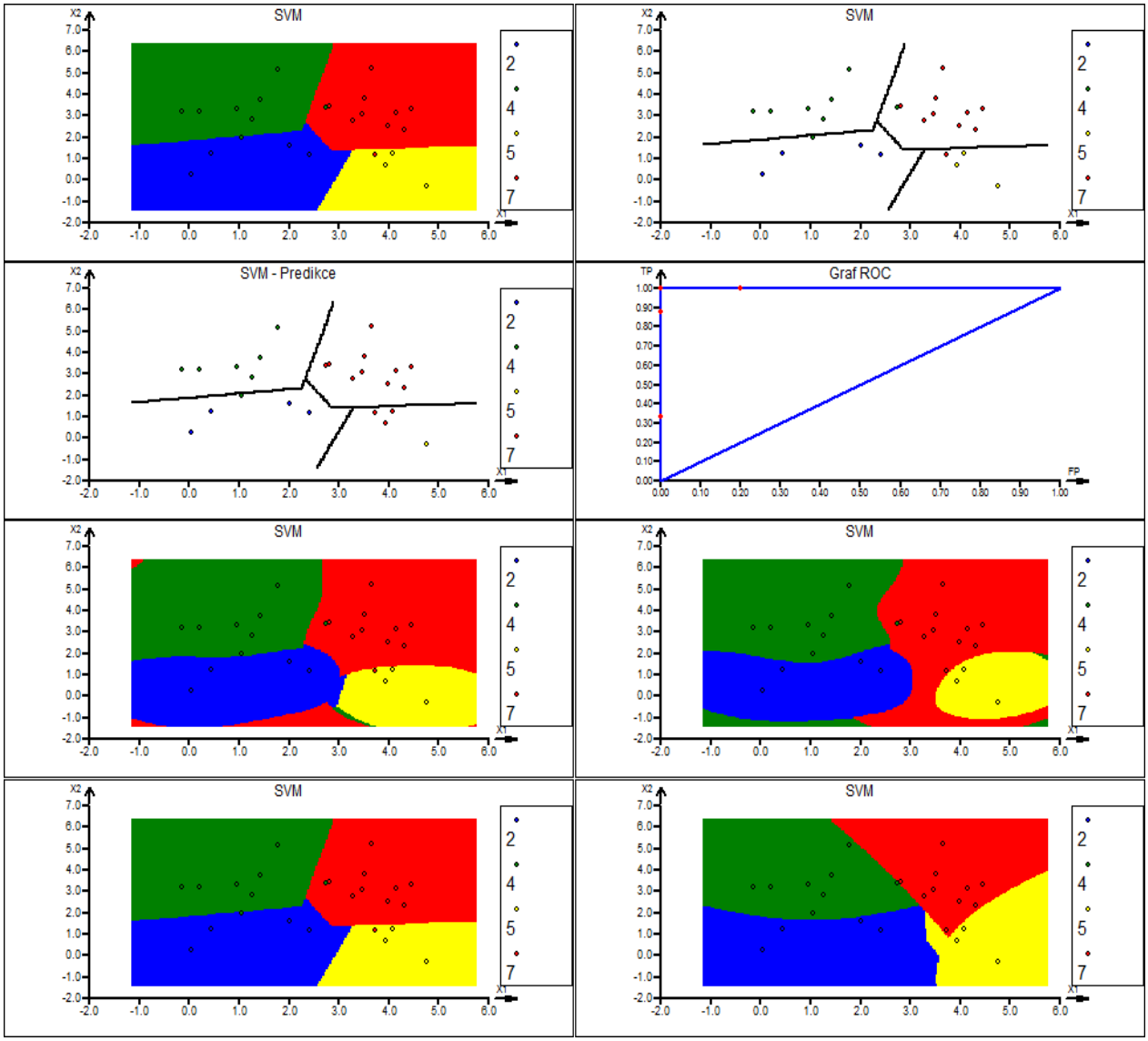
$Prediction: Vektor obsahující predikované hodnoty odezvy pro zadané hodnoty prediktoruX. Tato položka se vytváří pouze je-li zadáno PREDICT=1.

$Probability: Matice obsahující pravděpodobnosti jednotlivých možných úrovní (hodnot) odezvy pro zadané hodnoty prediktoru X. Vektor $Prediction obsahuje ty úrovně, které mají pro odpovídající řádek prediktoru nejvyšší pravděpodobnost. Pořadí sloupců odpovídá pořadí úrovní v položce $Levels. Tato položka se vytváří pouze je-li zadáno PREDICT=1.

$Residuals: Číselný vektor délky n obsahující nulu pokud predikce modelu souhlasí s

Příklad

```
n=25
graphsheet(cols=2)
x=matrix(normalr(2*n),ncols=2)
xm1=sample(vec(1,4),n,repl=1)
xm2=sample(vec(1,3),n,repl=1)
xm=bind(xm1,xm2)
x=x+xm
y=apply(xm,"sum",dir=1)
@sv1=svmc(x,y,predict=1,newdata=X,PROBABILITY=1,
CPLOT=1,BPLOT=1,PREDPLOT=1,ROCPLOT=1)
;
sv1=svmc(x,y,KERNEL="RBF",CPLOT=1)
sv1=svmc(x,y,KERNEL="RBF",CPLOT=1,type="COST",cost=10)
sv1=svmc(x,y,KERNEL="LINEAR",CPLOT=1)
sv1=svmc(x,y,KERNEL="POLYNOMIAL",CPLOT=1)
```



SVMR(X, Y[, KERNEL="LINEAR"|"POLYNOMIAL"|"RBF"|"TANH", PREDICT=0|1, NEWDATA=X1, TYPE="EPSILON"|"NU", DEGREE=, GAMMA=, COST=, NU=, R=, EPSILON=, SHRINK=1|0, FPLOT=0|1,PREDPLOT=0|1,RESPLOT=0|1])

Support Vector Regression. Regrese modelem Support Vector Machines

Funkce vytvoří model SVM-R (regresní model SVM) se zadaným jádrem pro zadané hodnoty prediktoru X a odezvy Y se zadanými parametry. Prediktor je numerická matice nebo vektor, odezva je numerický reálný vektor. Pomocí argumentu NEWDATA lze zadat nové, jiné hodnoty prediktoru, který musí mít stejný počet sloupců jako X ale může mít jiný počet řádků. Pro NEWDATA jsou pak vypočítány predikované hodnoty odezvy. Výsledkem funkce SVMR je seznam. Podrobněji je metoda SVM-R a význam jejích argumentů popsána v uživatelském manuálu QCExpert a v další doporučené literatuře.

Povinné argumenty

X: reálný číselný vektor (N x 1) nebo matice (N x M), obsahuje v řádcích hodnoty prediktoru.

Y: Číselný vektor (N x 1) obsahující hodnotu odezvy pro každý řádek X. Y musí mít stejný počet řádků jako X.

Příklad vstupních dat X a Y (zde N=6, M=4):

X				Y
2.6	2553	12.0	0.0224	12.4
5.3	3564	17.4	0.0644	8.5
7.1	4701	14.3	0.0703	20.7
2.7	3386	18.3	0.0369	17.2
4.3	3703	18.1	0.0296	15.6
3.3	2994	16.3	0.0529	9.7

Nepovinné argumenty

KERNEL: Textový řetězec obsahující typ použitého jádra SVM. Možné typy jádra jsou: "LINEAR" (lineární jádro), "POLYNOMIAL" (polynomické jádro, jehož stupeň je určen argumentem DEGREE), "RBF" (Gaussovské jádro, Radial Base Function), "TANH" (hyperbolický tangens, sigmoidální jádro). Není-li argument uveden, použije se implicitní hodnota "LINEAR".

PREDICT: Logická hodnota 0 nebo 1 určující zda se mají vypočítat predikované hodnoty pro NEWDATA. Implicitní hodnota je 0. Je-li zadána hodnota 1 a není zadán argument NEWDATA, dojde k chybě.

NEWDATA: Nová data pro predikci, data musí mít stejný počet sloupců jako X. Pro tato data funkce SVMC vypočítá predikované hodnoty odezvy.

TYPE: Textový řetězec "COST" nebo "NU", určuje metodu výpočtu ztrátové funkce. Je-li zadán typ "COST", ignoruje se případná hodnota argumentu NU, a naopak je-li zadán typ "NU", ignoruje se případná hodnota argumentu COST. Implicitní hodnota je "COST".

DEGREE: Je-li zadáno polynomické jádro (KERNEL="POLYNOMIAL"), určuje tento argument stupeň polynomu, jinak je ignorován. Hodnota parametru musí být kladné celé číslo, nejčastěji 1, 2, 3. Implicitní hodnota je 2.

GAMMA: Reálné číslo, parametr strmosti jádra, implicitní hodnota je 0, obvyklé hodnoty jsou od 0 do 10, případně i vyšší.

COST: Hodnota koeficientu ztrátové funkce. Argument je ignorován, je-li zadáno TYPE="NU". Implicitní hodnota je 1.

NU: Hodnota koeficientu ný (ν). Argument je ignorován, je-li zadáno TYPE="COST". Implicitní hodnota je 1.

R: Parametr polynomického a sigmoidálního jádra

SHRINK: Logická hodnota 0 nebo 1, zkrácení odhadu, někdy může vést ke snížení počtu support vektorů a mírnému zlepšení stability modelu.

PROBABILITY: Logická hodnota 0 nebo 1, je-li zadána hodnota 1, je ve výsledkovém seznamu funkce položka Probability, která obsahuje odhady pravděpodobností pro jednotlivé úrovně odezvy.

CPLOT: Logická hodnota 0 nebo 1, určuje, zda funkce SVMC vytvoří graf predikovanách úrovní pro X pokud má matice prediktoru X právě 2 sloupce. Při jiné dimenzi prediktoru se graf CPLOT nevytváří.

BPLOT: Logická hodnota 0 nebo 1, určuje, zda funkce SVMC vytvoří graf hranic predikovanách úrovní pro X pokud má matice prediktoru X právě 2 sloupce. Při jiné dimenzi prediktoru se graf BPLOT nevytváří. Tento graf má totožný význam jako CPLOT, pouze zvýrazňuje hranici mezi predikovanými klasifikacemi.

PREDPLOT: : Logická hodnota 0 nebo 1, určuje, zda funkce SVMC vytvoří graf predikovanách úrovní pro hodnoty zadané v argumentu NEWDATA. Vytváří se jen pokud má matice NEWDATA právě 2 sloupce. Při jiné dimenzi prediktoru se graf PREDPLOT nevytváří.

ROCPLOT: Logická hodnota 0 nebo 1, určuje, zda funkce SVMC vytvoří graf ROC.

Struktura výsledného seznamu

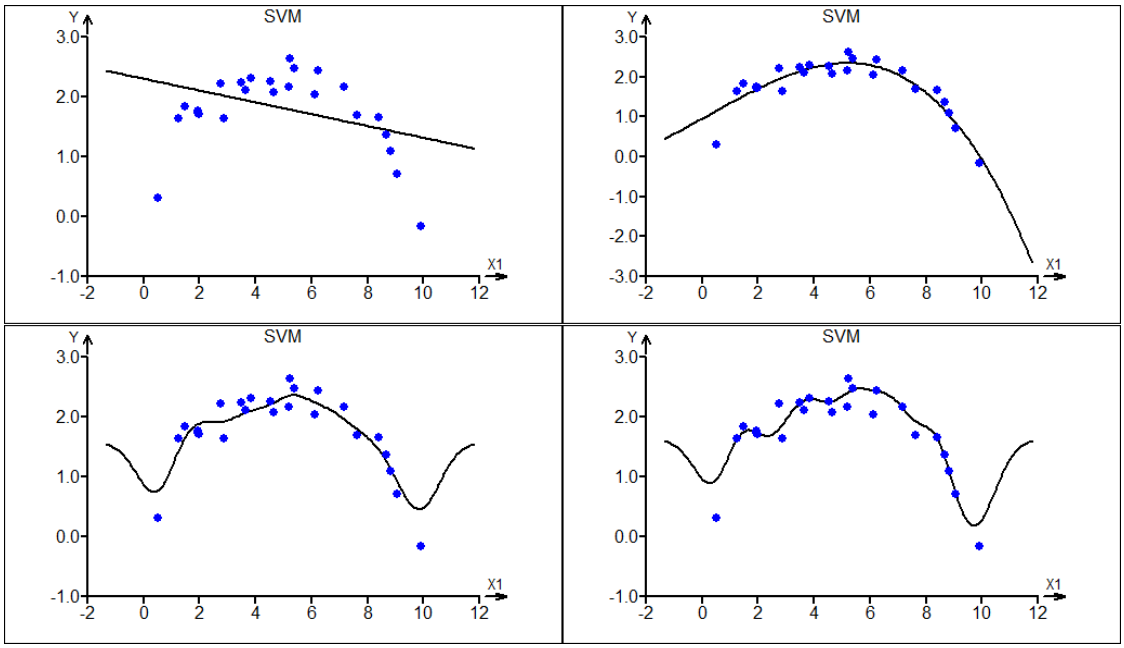
$NewPrediction: Vektor obsahující predikované hodnoty odezvy pro zadané hodnoty v argumentu NEWDATA. Tato položka se vytváří pouze je-li zadán argument NEWDATA a argument PREDICT=1.

$Prediction: Vektor obsahující predikované hodnoty odezvy pro zadané hodnoty prediktoruX.

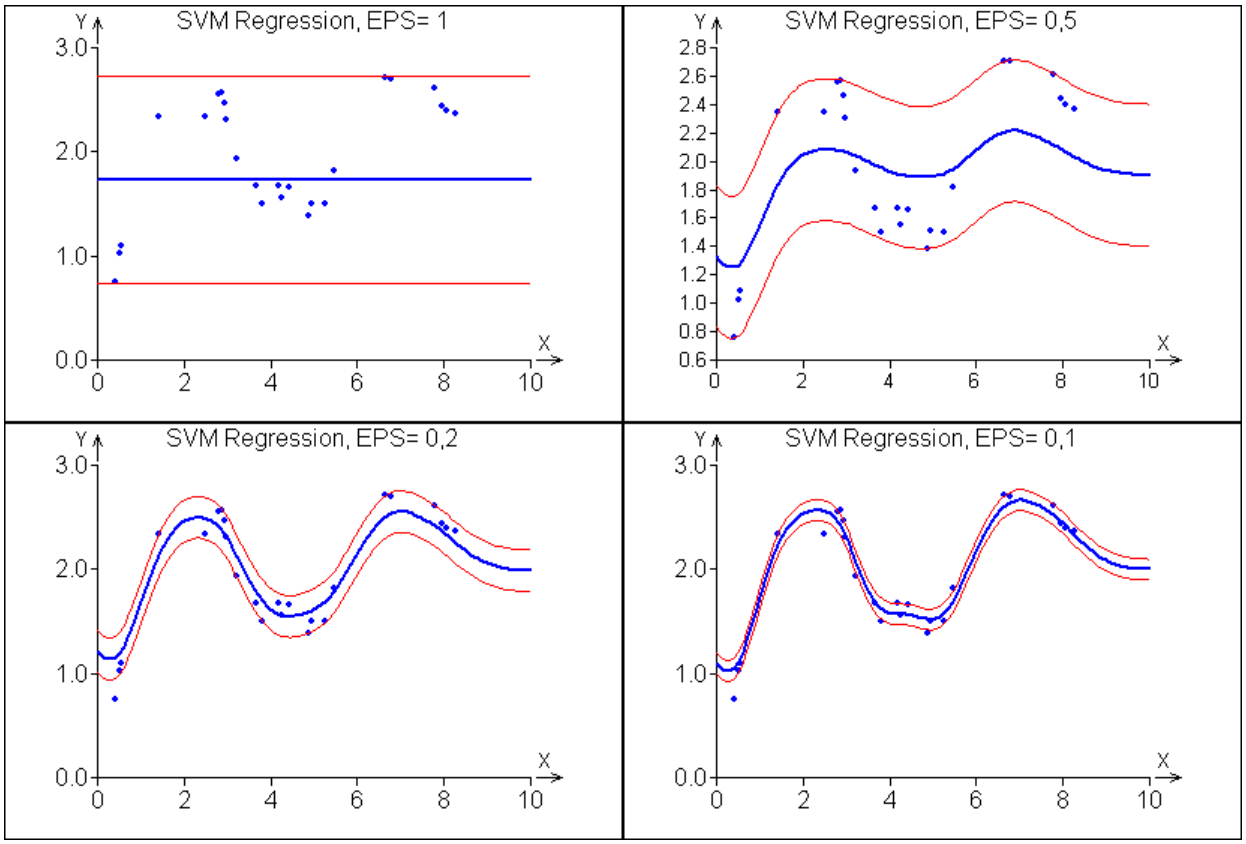
$Residuals: Číselný vektor délky N obsahující rozdíl zadaných hodnot odezvy od predikce.

Příklad

```
deletevars
// Regresní model s různými jádry a epsilony:
n=25
noise=normalr(n)*0.2
x=random(1:n)*10
y=x-0.1*x^2+0.3*sin(x)+noise
x1=seq(0,10,count=100)
graphsheet(cols=2)
ker=vec("LINEAR","POLYNOMIAL","RBF","RBF")
ep=vec(0.7,0.2,0.2,0.01)
ii=1
for(k=ker) {
@sv1=svmr(x,y,KERNEL=k,FPLOT=1,epsilon=ep[ii],
degree=3,R=1,NEWDATA=x,predict=1)
ii=ii+1
}
```



```
// Stejna data s ruznym epsilon s vyznačeným
// intervalem +/- epsilon.
graphsheet(cols=2)
for(e=vec(1,0.5,0.2,0.1))
{
@sv2=svmr(x,y,KERNEL="RBF",epsilon=e,newdata=x1,
predict=1);
plot(x,y,main="SVM Regression, EPS= "+e)
plotadd(x1,sv2$newprediction,type="line",width=2)
plotadd(x1,sv2$newprediction+e,type="line",width=1,color=3)
plotadd(x1,sv2$newprediction-e,type="line",width=1,color=3)
}
```



TAN(X)

Tangens. Tangent.

Funkce tangens, tan(x)

Povinný argument

X: reálné číslo (X je v radiánech), vektor, nebo matice

Příklad

```
>tan(pi/4)
1
```

Viz také

SIN, COS, ATAN

TANH(X)

Hyperbolic tangent. Hyperbolický tangens.

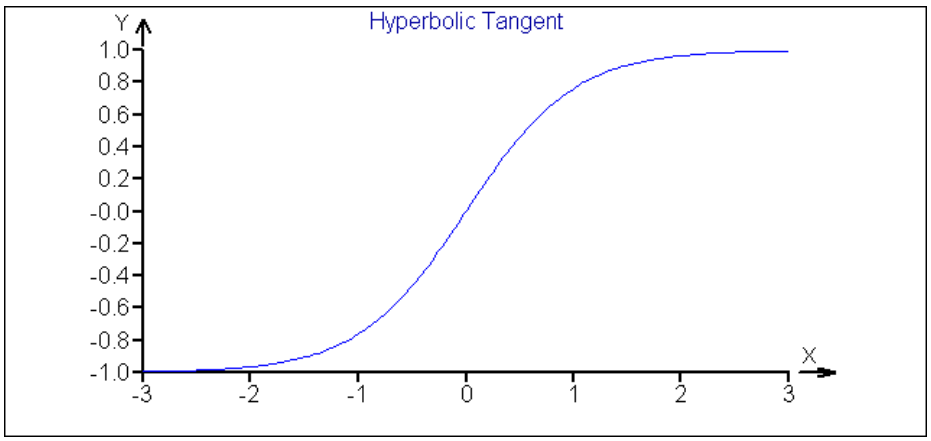
Funkce hyperbolický tangent, hyptan(x), $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

Povinný argument

X: reálné číslo (X je v radiánech), vektor, nebo matice.

Příklad

```
>tanh(vec(-1,0,1))
-0.761594155955765
0
0.761594155955765

x=seq(-3,3,count=200)
plot(x,tanh(x),type="line",main="Hyperbolic Tangent")
```



Viz také

COSH, SINH, ATANH, TAN

TIMEDIFN(D)

Convert Time in hour – text format to numeric hours format. Převod textového časového formátu na numerický počet hodin.

Vrací počet dnů v desetinném formátu z textového časového údaje. Vstupní textový formát času je "H:M:S:t", přičemž počty hodin, minut ani sekund nejsou omezeny a sčítají se. Část řetězce "t" za desetinnou tečkou se chápe jako počet milisekund, tedy "100" je 100 milisekund, ale "00100" je rovněž 100 milisekund, "0:00:01.001" je 1.001 sekundy, ale "0:00:01.1" je rovněž 1.001 sekundy. Výsledkem funkce TIMEDIFN je reálné číslo. Jedná se o inverzní funkci k TIMEDIFS.

Povinné argumenty

D: Textový řetězec, nebo vektor textových řetězců. Doba (chápe se jako časový interval, doba trvání, nikoli čas na hodinkách) (počet hodin, minut, sekund a milisekund) v textovém formátu, počty nejsou omezeny, např. `"18:29:52.009"`. znamená 18 hodin, 29 minut 52 sekund a 9 milisekund, ale je přípustné i `"148:240:5200.5000"`, což znamená 148 hodin plus 240 minut plus 5200 sekund plus 5000 milisekund.

Příklad

```
>timedifn("148:24:52.500")
6.18393524305556
```

Viz také

DATETIMEDIFN, TIMEDIFS, STRDATETIME, DATETIMEN, DATETIMES

TIMEDIFS(X)

Convert Time numeric format to text time format. Převod numerického počtu hodin na textový časový formát.

Doba (počet dnů) ve formě desetinného čísla se převede na textový časový údaj ve formátu "H:M:S:t", Výsledkem funkce je textový řetězec. Jedná se o inverzní funkci k TIMEDIFN. Doba (chápe se jako časový interval, doba trvání, nikoli čas na hodinkách).

Povinné argumenty

X: Reálné číslo, nebo reálný vektor představující počet dní v desetinném formátu. X může být výsledkem funkce DATETIMEDIFN nebo TIMEDIFN.

Příklad

```
>timedifs(5.59)
"134:9:36.0"
```

Viz také

DATETIMEDIFN, TIMEDIFN, STRDATETIME, DATETIMEN, DATETIMES

TRACEOFF

Set program tracing off. Vypnutí trasování programu.

Vypne režim trasování programu

Příklad

```
a=0;b=1
traceon
while(gt(b,1e-10))
{
b=b/2
a=a+b
}
traceoff
print(a)
```

Viz také

TRACEON, STOP, PAUSE

TRACEON

Set program tracing on. Zapnutí trasování programu.

Zapne režim trasování programu. Tento režim lze zapnout pouze v rámci spuštěného skriptu. V režimu trasování se zobrazují všechny změny hodnot proměnných v seznamu proměnných i v datové tabulce. Toho lze využít sledování, případně ladění programu, nebo při demonstraci funkce programu.

Příklad

viz TRACEOFF.

Viz také

TRACEOFF, STOP, PAUSE

TRANSP (X)

Transpose matrix or vector. Transpozice matice nebo vektoru.

Zamění řádky matice X nebo vektoru za sloupce, výsledkem je transponovaná matice $\mathbf{X}^T$.

Povinné argumenty

X: číselná nebo řetězcová matice nebo vektor.

Příklad

```
>transp(1:20)
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

>transp(1:20)#(1:20) // Skalární součin vektorů
2870
```

Viz také

UNIT, INV, NORM

TRUNC (X)

Truncate decimal part. Useknutí desetinné části.

Usekne desetinnou část X.

Povinné argumenty

X: reálné číslo, vektor, nebo matice.

Příklad

```
>trunc(vec(-3.9,-1.1,-0.1,0.1,3.9))
```

```
-3
-1
0
0
3
```

Viz také

INT, FLOOR, CEIL, FRAC

UNIT(N)

Unit square matrix. Jednotková matice.

Vytvoří čtvercovou jednotkovou matici dimenze N x N.

Povinné argumenty

N: Celé kladné číslo, rozměr jednotkové matice

Příklad

```
>unit(5)
1   0     0     0     0
0   1     0     0     0
0   0     1     0     0
0   0     0     1     0
0   0     0     0     1

>rad=5
>R=int(10*random(unit(rad)))
>R
1   1     4     1     9
7   0     5     7     2
6   6     9     3     4
1   1     0     0     8
0   9     7     7     5
>R1=R#inv(R)-unit(rad)
>R1
2.220E-016     -2.220E-016 9.020E-017  -2.775E-016 0
2.220E-016     -3.330E-016 2.220E-016  -3.885E-016 0
5.551E-017     0           0           -3.330E-016 0
2.775E-017     -1.387E-016 1.110E-016  0     5.551E-017
2.220E-016     -2.220E-016 -5.551E-017 0     0

>norm(R1)
1.44755372248954E-15
```

Viz také

ONES, BIND, TRANSP, MATRIX

VAR (X)

Variance of X. Rozptyl X.

Je-li X vektor, vypočítá výběrový rozptyl $s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$. Je-li argument X matice s rozměry *n* x *m*, vrací kovarianční matici **S** typu *m* x *m* s prvky $S_{ij} = \frac{1}{n-1}\sum_{k=1}^{n}(x_{ki} - \bar{\mathbf{x}}_i)(x_{kj} - \bar{\mathbf{x}}_j)$, kde $\bar{\mathbf{x}}_i$ je průměr *i*-tého sloupce matice **X**. Na diagonále kovarianční matice jsou rozptyly sloupců matice **X**. Směrodatná odchylka výběru X se získá jako `SQRT(VAR(X))`.

Povinné argumenty

X: Numerický vektor, nebo matice.

Příklad

```
>var(normalr(1000))
1.08816147429982

>z=matrix(normalr(100),ncols=4)
>cc=round(var(z),3)
>cc
1.139     -0.123      0.242       0.073
-0.123    0.854       0.171       0.093
0.242     0.171       1.234       0.173
0.073     0.093       0.173       0.675
```

Viz také

`COR, MEAN, AVERAGE, APPLY`

VARS ( )

Get existing variables. Seznam definovaných proměnných.

Vrací matici s informacemi o existujících proměnných. Výsledná matice obsahuje názvy, počet řádků a sloupců, v případě skaláru i obsah všech definovaných proměnných. U proměnných typu „seznam“ se zde všechny prvky seznamu chápou jako samostatné proměnné. Maximální počet standardních proměnných je omezen na 256. Počet proměnných *BigData* není omezen.

Povinné argumenty

nejsou

Příklad

```
deletevars
a=diag(1:10)
b=normalr(10)
c="ABCDEFGH"
d=1/sqrt(2)
```

```
data=random(1:100)
data2=list(A=123,b="QWERTY",c=1:10)
ini(e)
x%=1:1000000
v=vars()

> v
```

A	10	10	
B	10	1	
C	1	1	ABCDEFGH
D	1	1	0.707106781186547
DATA	100	1	
DATA2$A	1	1	123
DATA2$B	1	1	QWERTY
DATA2$C	10	1	
E	0	0	Nedefinováno
X%	1000000	1	

Viz také

DEL, DELETEVARS, ISDEFINED, INI

VEC(X1[, ..., XN][, BYROWS=1|0])

Vector. Vektor.

Vytvoří vektor z prvků stejného typu (buď číselného nebo řetězcového)

Povinné argumenty

X1, …, XN: vektor nebo matice, počet argumentů je 1 nebo více. Ze zadaných argumentů se vytvoří sloupcový vektor. Jsou-li argumenty matice (jedna nebo více), vytvoří se z nich vektor buď po řádcích, nebo po sloupcích, podle hodnoty argumentu BYROWS.

Nepovinné argumenty

BYROWS: Číslo 0 nebo 1. Je-li mezi argumenty X1, …, XN matice, tento argument určuje v jakém pořadí se budou načítat prvky této matice. Je-li BYROWS=1 (implicitní hodnota), načítá se matice po řádcích, je-li BYROWS=0, matice se načítá po sloupcích.

Příklad

```
>vec(bind(1:3,11:13),byrows=0)
1
2
3
11
12
13

>vec(bind(1:3,11:13),byrows=1)
1
11
```

```
2
12
3
13

>vec(1:3,5,7,seq(8,10,count=5))
1
2
3
5
7
8
8.5
9
9.5
10
```

Viz také

MATRIX, SEQ, REP, :, BIND, BINDV, FOR, ONES, DIAG

WHILE(EXPR) {}

While cycle. Cyklus While.

Cyklus while opakuje celou sekvenci příkazů v příkazových závorkách { }, dokud je výsledek výrazu EXPR nenulový. Výraz EXPR se vyhodnocuje vždy před začátkem sekvence. Příkazové závorky musí být použity vždy, i když je v sekvenci jen jeden příkaz.

Povinné argumenty

EXPR: Číslo, číselný výraz. Typicky logický výraz, jako gt(X, 0.001), not(zero(N)), a podobně. Je-li tento výraz nulový, přeskočí se sekvence příkazů v příkazových závorkách. Dokud je výsledek výrazu nenulový (typicky má logickou hodnotu 1), provádí se opakovaně sekvence v příkazových závorkách.

Příklad

```
// Přibližný součet členů posloupnosti R^i
R=0.99; a=1; b=0; n=0
while(gt(a,1e-10))
{
n=n+1
a=a*R
b=b+a
}
>b // Výsledný součet:
98.9999999901935
>n //Počet iterací:
2292
```

Viz také

FOR, IF, {}

XOR(X1, X2)

Logical exclusive OR, Exkluzivní OR

Exkluzivní OR numerických pravdivostních hodnot X1 a X2. Číslo 0 se považuje za nepravdu (false), číslo různé od 0 (typicky 1) se považuje za pravdu (true). Hodnoty výsledku uvádí pravdivostní tabulka:

X1	X2	XOR(X1,X2)
0	0	0
0	1	1
1	0	1
1	1	0

Povinné argumenty

X1, X2: číslo, numerický vektor, nebo matice obsahující pravdivostní hodnoty.

Příklad

```
>xor(vec(1,1,0,0,1),vec(0,1,0,1,0))
1
0
0
1
1
```

Viz také

AND, OR, NOT, GT, LT, GE, LE, EQ, NE

ZERO(X)

Zero? Nula?

Logická funkce, vrací 1, je-li X=0, jinak vrací 1.

Povinné argumenty

X: číslo, číselný vektor, nebo matice.

Příklad

```
// Vybrat nenulové prvky vektoru
>A=vec(2,0,4,0,2,1,0,0,0,2)
>A[[ not(zero(A)) ]]
2
4
2
1
2
```

```
// Indexy nulových prvků A
>n=count(A)
>(1:n)[[ zero(A) ]]
2
4
7
8
9
```

Viz také

EQ, NE, LT, GT, LE, GE

8. Přílohy a tabulky

8.1. Seznam funkcí a příkazů podle skupin

8.1.1. Základní matematické funkce

ABS	Absolutní hodnota
ACOS	Arc kosinus
ACOSH	Arc kosinus hyperbolický
ARG	Arg funkce
ASIN	Arc sinus
ASINH	Arc sinus hyperbolický
ATAN	Arc tangens
ATANH	Arc tangens hyperbolický
CEIL	Celočíselný strop čísla
COS	Kosinus
COSH	Hyperbolický kosinus
COTAN	Kotangens
EXP	Exponenciála
FLOOR	Celočíselné dno čísla
HEAV	Heavisidův operátor
INT	Celá část
INTPOWER	Celočíselná mocnina
LN	Přirozený logaritmus
LOG	Dekadický logaritmus
LOG2	Dvojkový logaritmus
LOGN	Logaritmus při základu N
PI	Ludolfovo číslo π
POWER	Mocnina
ROUND	Zaokrouhlení
SIGN	Znaménko
SIN	Sinus
SINH	Hyperbolický sinus
SQR	Čtverec
SQRT	Druhá odmocnina
TAN	Tangens
TANH	Hyperbolický tangens
TRUNC	Celá část

8.1.2. Operátory a zvláštní znaky

+, -, *, /, ^	Skalární aritmetické operace
-	Unární minus
#	Maticové násobení
:	Celočíselná sekvence
%	Suffix proměnné BigData
[ ]	Indexové závorky
[[ ]]	Logické indexové závorky
$	Selektor prvku seznamu

;	Oddělovač příkazů na řádku
{ }	Příkazové závorky
//	Řádkový komentář
/*	Začátek blokového komentáře
*/	Konec blokového komentáře
@ ;	Víceřádkový příkaz

8.1.3. Statistické funkce

AVERAGE	Průměr
DIFF	Diference
CHISQP	Chi^2, distribuční funkce
CHISQQ	Chi^2, kvantilová funkce
COR	Korelační matice
COUNT	Počet prvků
CUSUM	Kumulativní součet
FACTORIAL	Faktoriál
FISHERP	F-rozdělení, distribuční funkce
FISHERQ	F-rozdělení, kvantilová funkce
GAMMA	Gamma funkce
GAMMALN	Logaritmus gamma funkce
MAD	Mediánová odchylka
MADS	Mediánová směrodatná odchylka
MEDIAN	Medián
NORMALD	Normální rozdělení, hustota
NORMALP	Normální rozdělení, distribuční funkce
NORMALQ	Normální rozdělení, kvantilová funkce
NORMALR	Normální rozdělení, náhodné číslo
PROD	Součin
RANDOM	Náhodné číslo, rovnoměrné
RND	Náhodné číslo, rovnoměrné
SAMPLE	Náhodný výběr
STUDENTP	t-rozdělení, distribuční funkce
STUDENTQ	t-rozdělení, kvantilová funkce
SUM	Součet
VAR	Rozptyl, kovarianční matice

8.1.4. Logické a relační funkce

AND	Logický součin AND
EQ	Rovno
GE	Větší nebo rovno
GT	Větší než
ISNUM	Test typu číslo
ISTEXT	Test typu text
ISUNDEF	Test typu „nedefinováno“
LE	Menší nebo rovno
LT	Menší než
NE	Není rovno
OR	Logický součet OR

XOR	Logická operace XOR (exkluzivní OR)
ZERO	Test nulové hodnoty

8.1.5. Textové funkce

ASCII	ASCII kód
ASNUMERIC	Převod na číslo
ASTEXT	Převod na text
CAPS	Velká písmena
CHR	Znak ASCII
LENGTH	Délka textového řetězce
LETTERS	Posloupnost ASCII
POS	Pozice v řetězci
REPLACE	Nahrazení na pozici v řetězci
REPLACES	Nahrazení podřetězce v řetězci
SMALL	Malá písmena
SUBSTR	Část řetězce

8.1.6. Časové/datové funkce

DATETIMEDIFN	Rozdíl časů
DATETIMEN	Konverze času z textového do numerického formátu
DATETIMES	Konverze času z numerického do textového formátu
DAYINWEEK	Den v týdnu
DAYINYEAR	Den v roce
TIMEDIFN	Konverze textové časové diference na hodiny
TIMEDIFS	Konverze počtu hodin na textový formát
STRDATE	Datum
STRTIME	Čas
STRDATETIME	Datum a čas

8.1.7. Třídění a pořadí

MAX	Maximum
MIN	Minimum
ORDER	Pořadí
REV	Opačné pořadí
SORT	Třídění

8.1.8. Základní maticové a vektorové funkce

APPLY	Aplikace funkce na řádky nebo sloupce matice
BIND	Spojit sloupce
BINDV	Spojit řádky
COL	Sloupec
DIAG	Diagonála
DIM	Dimenze
MATRIX	Matice
NCOLS	Počet sloupců
NROWS	Počet řádků
ONES	Jedničky

REP	Opakování hodnoty
ROW	Řádek
SEQ	Sekvence
SPLIT	Oddělit sloupce
SPLITV	Oddělit řádky
UNIT	Jednotková matice
VEC	Vektor

8.1.9. Lineární algebra

#	Maticové násobení
DET	Determinant
EIGENVAL	Vlastní čísla
EIGENVEC	Vlastní vektory
INV	Inverze matice
NORM	Norma matice
PINV	Pseudoinverze matice
SVDD	Sigulární rozklad, D
SVDU	Sigulární rozklad, U
SVDV	Sigulární rozklad, V
TRANSP	Transpozice matice

8.1.10. Grafické příkazy

GRAPHSHEET	Nastavení grafického listu
LINEADD	Přidat přímku do grafu
PLOT	Graf
PLOTADD	Přidat graf
PLOTBAR	Sloupcový graf
PLOTPOLY	Graf polygonu
PLOTPOLYADD	Přidat polygon
PLOTTEXT	Graf s textem
PLOTTEXTADD	Přidat text
PLOT3DPOINTS	3D graf, bodový
PLOT3DSURFACE	3D graf, povrch
PLOT3DSPLINE	3D graf, spline
PLOT3DDENSITY	3D graf, hustota rozdělení

8.1.11. Export, import, tisk, soubory

DBCONNECT	Připojit databázi
DBDISCONNECT	Odpojit databázi
DBGETFIELDS	Názvy sloupců databáze
DBGETTABLES	Názvy tabulek databáze
DBIMPORT	Import tabulky s SQL dotazem
DBIMPORTTABLE	Import tabulky
DELETESHEET	Smazání datového listu QCE
EXPORT	Export proměnné
EXPORTGRAPH	Export grafu
FILECOPY	Kopírování souborů

FILEDELETE	Smazání souborů
FILEEXISTS	Hledání souborů
FILEFIND	Nalezení souborů
FILEMOVE	Přesouvání souborů
IMPORT	Import proměnné
GETIMAGE	Načtení grafického souboru BMP
GETSHEET	Načtení obsahu datového listu QCE
GETSHEETHEADER	Načtení záhlaví datového listu QCE
GETSHEETNAMES	Načtení názvů datových listů QCE
PDFBEGIN	Nový dokument PDF
PDFEND	Konec dokumentu PDF
PDFFONT	Nastavení fontu PDF
PDFFOOTER	Definice zápatí PDF
PDFHEADER	Definice záhlaví PDF
PDFIMAGE	Vložení bitové mapy do PDF
PDFNEWPAGE	Nová stránka PDF
PDFPLOT	Vložení grafu do PDF
PDFTABLE	Zapsání tabulky do PDF
PDFTEXT	Zapsání textu do PDF
PRINT	Tisk do listu
PRINTSHEET	Nastavení listu pro tisk
PUTIMAGE	Export bitové mapy BMP
PUTSHEET	Kopírování do datového listu QCE
SENDMAIL	Odeslání e-mailu

8.1.12. Definice proměnných

DELETE	Smazání proměnné
DELETEVARS	Smazání všech proměnných
INI	Inicializace nedefinované proměnné
LIST	Seznam
VARS	Výpis proměnných

8.1.13. Řízení programu

{ }	Příkazové závorky
DIALOG	Interaktivní dialogové okno
EXEC	Spuštění externí aplikace
FOR	Cyklus for
FUNCTION	Uživatelská funkce
IF	Podmínka if
MESSAGE	Zobrazení informačního okna
PARSE	Vyhodnocení řetězce jako výrazu nebo příkazu
PAUSE	Časová prodleva
RETURN	Vrácení výsledku funkce
STOP	Ukončení programu
TRACEOFF	Vypnout trasování
TRACEON	Trasování programu
WHILE	Cyklus While

8.1.14. Statistické metody a modely

LINREG	Lineární regrese
LOCALREG	Lokální regrese
MULTIVAR	Vícerozměrná analýza dat
NNLEARN	Natrénovat neuronovou síť
NNPREDICT	Predikovat neuronovou sítí
NNTIMELEARN	Natrénovat časovou neuronovou síť
POLYREG	Polynomická regrese
SPLINE1	Kubický spline
SPLINE2	Kubický spline
SVMC	Support vector machines: klasifikace
SVMR	Support vector machines: regrese

8.2. Formální definice funkcí a jejich argumentů

```
ABS(arg1)
ACOS(arg1)
ACOSH(arg1)
AND(arg1, arg2)
APPLY(arg1, arg2, DIR=1|2)
ARG(arg1, arg2)
ASCII(arg1)
ASIN(arg1)
ASINH(arg1)
ASNUMERIC(arg1)
ASTEXT(arg1)
ATAN(arg1)
ATANH(arg1)
AVERAGE(arg1)
AVG(arg1)
BIND(arg1, arg2)
BINDV(arg1, arg2)
CAPS(arg1)
CEIL(arg1)
COL(arg1, arg2)
COR(arg1)
COS(arg1)
COSH(arg1)
COTAN(arg1)
COUNT(arg1)
CUSUM(arg1)
DATETIMEDIFN(arg1, arg2)
DATETIMEN(arg1)
DATETIMES(arg1)
DAYINWEEK(arg1)
DAYINYEAR(arg1)
DBCONNECT(USER=, PSWD= [, SERVER=, DB=, ROLE=, LOCALE=])
DBCREATE(USER=,PSWD=,SERVER=,DB=[,LOCALE="WIN1250"])
DBCREATETABLE(NAME=[, FORMAT=, MODE="APPEND"|ERASE"|"DROP",
      DATA="name", SECURITY=1|0])
DBDISCONNECT()
DBGETFIELDS(arg1 [, VALIDONLY=1|0, SYSFIELDS=0|1])
DBGETTABLES()
DBIMPORT(query)
DBIMPORTTABLE(table [, STARTDATE=, ENDDATE=, FIELDS=, VALIDONLY=1|0,
      SYSFIELDS=0|1])
DEL(arg1 [, arg2, ..., argN])
DELETE(arg1, [arg2, ..., argN])
DELETESHEET(arg1)
DELETEVARS
DET(arg1)
DIAG(arg1)
```

```
DIALOG(arg1, ……)
DIFF(arg1)
DIM(arg1)
EIGENVAL(arg1)
EIGENVEC(arg1)
EQ(arg1, arg2)
EXEC(filename[,PARAMS=,DIR=,WAIT=1|0,HIDE=0|1])
EXP(arg1)
EXPORT(arg1, filename, [DELIMITER="\t", DECIMALSEPAR="."])
EXPORTGRAPH(file [, RESIZE=vec(width, height), SHEETNAME=])
FACT(arg1)
FILECOPY(srcdir,destdir,filename)
FILEDELETE(dir,filename)
FILEEXISTS(dir,filename)
FILEFIND(dir,filename)
FILEMOVE(srcdir,destdir,filename)
FISHERP(arg1, arg2, arg3)
FISHERQ(arg1, arg2, arg3)
FLOOR(arg1)
FOR(INDEX=start, end| INDEX =vektor) {…}
FRAC(arg1)
GAMMA(arg1)
GAMMALN(arg1)
GE(arg1, arg2)
GETIMAGE(arg1)
GETSHEET(arg1)
GETSHEETHEADER(arg1[, ALL=0|1])
GETSHEETNAMES( )
GRAPHSHEET(COLS=arg1)
GT(arg1, arg2)
HEAV(arg1)
CHISQP(arg1, arg2)
CHISQQ(arg1, arg2)
CHR(arg1)
IF(func=0|<>0) {}
IMPORT(arg1, filename [, DELIMITER="\t", DECIMALSEPAR=".", STARTROW=,
      ENDROW=])
INI(var1[, var2,...,varN])
INT(arg1)
INTPOWER(arg1, arg2)
INV(arg1)
ISNUM(arg1)
ISTEXT(arg1)
ISUNDEF(var1)
LE(arg1, arg2)
LENGTH(arg1)
LETTERS(arg1, arg2)
LINEADD([H|V|A, B=] [, COLOR=] [, SHADE=1..100] [, WIDTH=])
LINREG(arg1, arg2 [, W=REP(1,NROWS(par1)), XNEW=par1, ABSOLUTE=1|0,
      ALPHA=0.05])
```

LIST([name1=]arg1 [..., [nameN=]argN])
LN(arg1)
LOCALREG(arg1, arg2[, POLDEG=2, KERNEL="QUAD"|"GAUSS", KWIDTH=1,
XNEW=, ALPHA=0.05])
LOG(arg1)
LOG2(arg1)
LOGN(arg1, arg2)
LT(arg1, arg2)
MAD(arg1)
MADS(arg1)
MATRIX(arg1, ncols=|nrows=[, byrows=0|1])
MAX(arg1)
MEDIAN(arg1)
MESSAGE([LABEL=][, par1, ..parN])
MIN(arg1)
MULTIVAR(arg1 [, CORREL=0|1, BIPLOT=0|1, LOADINGS=0|1, VARIANCES=0|1,
COMPO=0|1, NORMAL=0|1, ANDREWS=0|1 , MAHALA=0|1])
NCOLS(arg1)
NE(arg1, arg2)
NNLEARN(arg1, arg2 [, IDENT=, XNAMES=, YNAMES=, LAYERS=, MODELFILE=,
ITERATIONS=, USEFORTEACH=, EXPONENT=, ALPHA=, MOMENTUM=,
TEACHRATE=, IDENTERROR=, MEANERR=0|1, RESIDUALS=1|0, GRNET=0|1,
GRPREDICT=0|1], BESTMODEL=0|1])
NNPREDICT(arg1, MODELFILE=|Model= [, FORECAST=, ALPHA=])
NNTIMELEARN(arg1, [, IDENT=, MODELTYPE=AR|DIFF, MODELDEPTH=,
LAYERS=, MODELFILE=, ITERATIONS=, EXPONENT=, ALPHA=,
MOMENTUM=, TEACHRATE=, IDENTERROR=, MEANERR=0|1,
RESIDUALS=1|0, GRNET=0|1, GRPREDICT=0|1], BESTMODEL=0|1])
NORM(arg1)
NORMALD(arg1, [MEAN=], [SDEV=])
NORMALP(arg1, [MEAN=], [SDEV=])
NORMALQ(arg1, [MEAN=], [SDEV=])
NORMALR(arg1, [MEAN=], [SDEV=])
NROWS(arg1)
ONES(arg1)
OR(arg1)
ORDER(arg1, arg2)
ORDER1(arg1, arg2)
PARSE(arg1)
PASTE(arg1[,..,argN][,SEPARATOR=""])
PAUSE(arg1)
PDFBEGIN(FILE [,Margins=Vec(Left,Top,Right,Bottom),
ORIENTATION=PORTRAIT|LANDSCAPE, TITLE=, AUTHOR=, SUBJECT=,
KEYWORDS=])
PDFEND([LAUNCH=0|1])
PDFFONT([NAME="Tahoma", SIZE=12, ITALIC=0|1, BOLD=0|1, UNDERLINE=0|1,
COLOR=])
PDFFOOTER("text left", "text right"[, LINE=0|1])
PDFHEADER("text vlevo", "text vpravo"[,LINE=0|1])
PDFIMAGE(filename, [WIDTHMM=, ALIGN=LEFT|RIGHT|CENTER])

PDFNEWPAGE()
PDFPLOT([SHEETNAME=CURRENT|"sheet name listu", RESIZE=vec(width,height),
WIDTHMM=, ALIGN=LEFT|RIGHT|CENTER])
PDFTABLE(arg1[, BORDER=1|0, HEADER=])
PDFTEXT(arg1[,...,argN][, ALIGN=LEFT|RIGHT|CENTER|JUSTIFY])
PH(arg1)
PI
PINV(arg1)
PLOT(arg1[, arg2][, TYPE=POINT|LINE|POINTLINE], [MAIN=, LABX=, LABY=,
COLOR=, SHADE=1..100, WIDTH=, PTCOLOR=, PTSHADE=1..100, PTTYPE=,
PTSIZE=, BOX=1|0, AXES=1|0])
PLOTADD(arg1, arg2, [TYPE=POINT|LINE|POINTLINE], [COLOR=, SHADE=1..100,
WIDTH=, PTTYPE=])
PLOTBAR(arg1, [MAIN=, LABX=, LABY=,
ORIENTATION="VERTCAL"|"HORIZONTAL", GAP=, STACK=0|1, LABELS=,
KEY=, COLOR=, WIDTH=, FILL=1|0, FILLCOLOR=],)
PLOTPOLY(arg1, arg2 [, MAIN=, LABX=, LABY=, COLOR=, WIDTH=, FILL=1|0,
FILLCOLOR=, AXES=1|0, BOX=1|0, TIMEAXIS=1|0])
PLOTPOLYADD(arg1, arg2[, COLOR=, WIDTH=, FILL=1|0, FILLCOLOR=,
TIMEAXIS=1|0])
PLOTTEXT(arg1, arg2, arg3, [MAIN=, LABX=, LABY=, ALIGN=CENTER|LEFT|RIGHT,
TEXTSIZE=, COLOR=, SHADE=1..100] , BOX=1|0, AXES=1|0)
PLOTTEXTADD(arg1, arg2, arg3, [ALIGN=CENTER|LEFT|RIGHT, TEXTSIZE=,
COLOR=, SHADE=1..100])
PLOT3DPOINTS(arg1, arg2, arg3, [MAIN=, ANGLEX=, ANGLEY=, ANGLEZ=,
BOX=0|1|2, AXES=0|1, ISOMETRIC=0|1, ORIGIN=vec(x, y, z)])
PLOT3DSURFACE(arg1, [MAIN=, XLIM=vec(a, b), YLIM=vec(a, b), ANGLEX=,
ANGLEZ=, BOX=0|1|2, COLRANGE=vec(col1, ..), GRIDCOLOR=])
PLOT3DSPLINE(arg1, arg2, arg3, [MAIN=, SMOOTH=1, GRID=vec(a, b), ANGLEX=,
ANGLEZ=, BOX=0|1|2, COLRANGE=vec(col1, ..), GRIDCOLOR=])
PLOT3DDENSITY(arg1, arg2, [MAIN=, SMOOTH=1, GRID=vec(a, b), ANGLEX=,
ANGLEZ=, BOX=0|1|2, COLRANGE=vec(col1, ..), GRIDCOLOR=])
POLYREG(arg1, arg2 [, W=REP(1,NROWS(par1)), XNEW=par1, POLDEG=2,
ALPHA=0.05])
POS(arg1, arg2)
POWER(arg1, arg2)
PRINT(arg1[, ..., argN][FILE=, APPEND=1|0, DELIMITER="\t"])
PRINTSHEET([NAME=, COLWIDTH=, ROWHEIGHT])
PROD(arg1)
PUTIMAGE(filename,data[,FORMAT=24|1|4|8|16|32])
PUTSHEET(arg1, arg2[, HEADER=, AUTOSIZE=0|1])
RANDOM(arg1)
REP(arg1, arg2[, BYROWS=1|0])
REPLACE(arg1, arg2, arg3)
REPLACES(arg1, arg2, arg3 [, ALL=0|1, NOCASE=0|1])
RETURN(…)
REV(arg1)
RND(arg1)
ROUND(arg1, arg2)
ROW(arg1, arg2)

SAMPLE(arg1, arg2[, REPL=0|1])
SENDMAIL([text_zpravy1[,...,text_zpravy],]
ACCOUNT=List(HOST=,USER=,PASSWORD=,FROMEMAIL=,FROMNAME=),
TOEMAIL=email|VEC(emails),
[SUBJECT=,ATTACHMENT=attachment|VEC(attachments)])
SEQ(arg1, arg2, [COUNT|STEP=])
SIGN(arg1)
SIN(arg1)
SINH(arg1)
SMALL(arg1)
SORT(arg1, arg2)
SPLINE1(arg1, arg2, arg3, arg4)
SPLINE2(arg1, arg2, arg3)
SPLIT(arg1, arg2)
SPLITV(arg1, arg2)
SQR(arg1)
SQRT(arg1)
STRDATE()
STRTIME()
STRDATETIME()
STOP
STUDENTP(arg1, arg2)
STUDENTQ(arg1, arg2)
SUBSTR(arg1, arg2)
SUM(arg1)
SVDD(arg1)
SVDU(arg1)
SVDV(arg1)
SVMC(arg1, arg2[, KERNEL="LINEAR"|"POLYNOMIAL"|"RBF"|"TANH",
PREDICT=0|1, NEWDATA=X1, TYPE="COST"|"NU", DEGREE=, GAMMA=,
COST=, NU=, R=, SHRINK=1|0, PROBABILITY=0|1, CPLOT=0|1, BPLOT=0|1,
PREDPLOT=0|1, ROCPLOT=0|1])
SVMR(arg1, arg2 [, KERNEL="LINEAR"|"POLYNOMIAL"|"RBF"|"TANH", REDICT=0|1,
NEWDATA=, TYPE="EPSILON"|"NU", DEGREE=, GAMMA=, COST=, NU=, R=,
EPSILON=, SHRINK=1|0, FPLOT=0|1, PREDPLOT=0|1, RESPLOT=0|1])
TAN(arg1)
TANH(arg1)
TIMEDIFS(arg1)
TIMEDIFN(arg1)
TRACEOFF
TRACEON
TRANSP(arg1)
TRUNC(arg1)
UNIT(arg1)
VAR(arg1)
VARS()
VEC(arg1 [, ..., argN][, byrows=1|0])
WHILE(func=0|<>0) {}
XOR(arg1)
ZERO(arg1)

 Poznámky:

 Poznámky:

9. Rejstřík pojmů, funkcí a příkazů

(Názvy funkcí jsou uvedeny velkými písmeny.)

Q

R

Ř

S

T

U

V

W

X

Z