CMAT Newsletter: July 2006

Wolfgang M. Hartmann

August 2006

Contents

1 General Remarks

Some bugs are fixed - again. Among others, some related to the `glim` function applied to models with a string type response variable.
A new language feature, a grammar modification for the already existing matrix literals, was implemented. Inside the brackets `[` and `]` of a matrix literal, identifiers (names starting with a letter and containing only letters, digits and underscores) are treated as object names. Strings had to be embedded into quotes. This syntax is consistent with the C language and is among others similar in Matlab. Now, the two new tokens `["` and `]"` can be used as brackets of a matrix literal where identifiers are treated as strings and not as object names. Such a form of matrix literal is similar to that used in SAS/IML and will probably find lots of use for specifying small matrices of mixed, string and numeric, data type.

1.1 New Functions

The following new functions are implemented:

canals for canonical correlation analysis of two sets of categorical variables. (See GIFI project.)

cuclcr compute cubic cluster criterion and R^2 for given cluster decomposition (Sarle, 1983)

demreg computes Deming regression which is a specific for of error-in-variables linear regression.

dixonr computes pdf, cdf and critical values for Dixon's r, useful for onedimensional outlier tests. (Mc Bane, 2006)

fica implements the original algorithm of the (Fast) Independent Component Analysis which is a type of nonlinear principal component analysis (Gaevert, Hurri, Saerelae, & Hyvaerinen, 2005)

histplot for plotting a histogram. It is usually used for the distribution of eigen or singular values.

homals for homogeneity analysis of categorical data. (See GIFI project.)

ita for item tree analysis of binary data.

overals for canonical correlation analysis of more than two sets of categorical variables. (See GIFI project.)

primals for one-dimensional homogeneity analysis of categorical data. (See GIFI project.)

sdd implements the semidiscrete decomposition of a matrix (Kolda & O'Leary, 1999)

sgmanova for multivariate analysis of variance based on spatial signs signs (robust estimation method). (Nevalainen & Oja, 2006)

xyplot for plotting a xy diagram. It is usually used for plotting a set of points in the xy plane, e.g. by the GIFI functions (`anacor`,`princals`,...).

zoverw for normal distrubuted variables z and w the probability and density of the normal distributed ratio z/w is computed as long as the distribution of z/w is unimodal and not bimodal. (Marsaglia, 2006)

1.2 Fixed Bugs

1. The `srand()` function did not reset the seed value of a number of random generators. (Instead the time of day was used resulting in different random values even when `srand()` was called. This is fixed now.

2. A number of bugs in the `nlp` function were fixed.

2 Modifications of Features

2.1 New Language Feature

A new language feature, a grammar modification for the already existing matrix literals, was implemented. Inside the brackets `[` and `]` of a matrix literal, identifiers (names starting with a letter and containing only letters, digits and underscores) are treated as object names. Strings had to be embedded into quotes. This syntax is consistent with the C language and is among others similar in Matlab. For example, the input

```
a = [ 1 2 , 3 4 ];
print "A=", a;
b = [ a a , a a ];
print "B=",b;
```

leads into

```
A=
   |   1   2
-----------
 1 |   1   2
 2 |   3   4
```

```
B=
   |   1   2   3   4
-------------------
 1 |   1   2   1   2
 2 |   3   4   3   4
 3 |   1   2   1   2
 4 |   3   4   3   4
```

Now, the two new token `["` and `]"` can be used as brackets of a matrix literal where identifiers are treated as strings and not as object names. Such a form of matrix literal is similar to that used in SAS/IML and will probably find lots of

use for specifying small matrices of mixed, string and numeric, data type. The following two examples result in the same mixed data type matrix:

```
school = [ 1 "regular"   "self"  10 ,  1 "regular"   "team"  17 ,
           1 "regular"   "class" 26 ,  1 "afternoon" "self"   5 ,
           1 "afternoon" "team"  12 ,  1 "afternoon" "class" 50 ,
           2 "regular"   "self"  21 ,  2 "regular"   "team"  17 ,
           2 "regular"   "class" 26 ,  2 "afternoon" "self"  16 ,
           2 "afternoon" "team"  12 ,  2 "afternoon" "class" 36 ,
           3 "regular"   "self"  15 ,  3 "regular"   "team"  15 ,
           3 "regular"   "class" 16 ,  3 "afternoon" "self"  12 ,
           3 "afternoon" "team"  12 ,  3 "afternoon" "class" 20 ];


cnam = [ "School" "Program" "Style" "Count" ];
school = cname(school,cnam);
print "Data 1=", school;
```

The new form does not need the so many around each string for telling CMAT that the identifiers are strings. Please note, that there should be no white space separating the brackets and strings, the two symbols are recognized as one token.

```
school2 = [" 1 regular   self  10 ,  1 regular   team  17 ,
             1 regular   class 26 ,  1 afternoon self   5 ,
             1 afternoon team  12 ,  1 afternoon class 50 ,
             2 regular   self  21 ,  2 regular   team  17 ,
             2 regular   class 26 ,  2 afternoon self  16 ,
             2 afternoon team  12 ,  2 afternoon class 36 ,
             3 regular   self  15 ,  3 regular   team  15 ,
             3 regular   class 16 ,  3 afternoon self  12 ,
             3 afternoon team  12 ,  3 afternoon class 20 "];
school2 = cname(school2,cnam);
print "Data 2=", school2;
```

Using Yacc and Lex, this language extension was programmed and tested in about two hours.

2.2 Extensions to `nlp()` Function

< xr,rp > = nlp(func,x0<,keyopt<,nlcon,<grad, <hess,<jcon>>>>>)

1. If the input vector `x0` has column (or row) names attached, they are copied to the columns of the return argument `xr`. And in any case, row names are attached to the `rp` output vector.

2. The limited memory BFGS method is now well tested and works very well for boundary contraints. There are two algorithm implemented, one unconstrained method and one constrained. The implementation is very smilar to the LBFGS and LBFGS-B Fortran code by Byrd and Nocedal. The only specific option for this method is the maximum number of (corrected) point x and gradient g vectors which is being used for computing the search direction. This integer number is specified with the `"corrs"` or the `"correcti"` option. As larger this number is closer becomes the LBFGS search direction to the commmon quasi Newton BFGS search direction. For very nonlinear functions this number should be selected larger than for fuctions close to linearity to permit a better approximation of the Hessian of the objective function. Default is $MAX(n, 5)$ for small n and 10 for larger n.

3. The test examples nlp01.inp,....,nlp36.inp now also contain calls to the unconstrained limited memory BFGS method. The test examples nlp41.inp,....,nlp87.inp now also contain calls to the bound constrained limited memory BFGS method.

2.3 Extensions to `nnmf()` Function

1. A printout of the residual matrix $\mathbf{A} - \mathbf{WV}$‘ was added. In addition, the *Lagrange multipliers* and Kuhn-Tucker conditions for the matrices $\mathbf{W}$ and $\mathbf{H}$ can now be computed and printed. For optimality the Lagrange multipliers should be smaller or equal (close to) zero and the Kuhn Tucker condition values should be small (close to zero exceot for rounding errors).

2. Some new classical optimization methods were added, like (dual) Quasi Newton (`"tech" "DUQUAN"`), Double Dogleg (`"tech" "DBLDOG"`), limited memory BFGS (`"tech" "LMEMQN"`), and conjugate gradient (`"tech" "CONGRA"`). They can be used to estimate the $\mathbf{W}$ and $\mathbf{H}$ matrices minimizing the MSE criterion

$$\|\mathbf{A} - \mathbf{WH}\|_2^2$$

   treating the nonnegativity constraints in two different ways:

   - as explicit boundary constraints (default) with zero lower bounds,
   - as unconstrained, but reparametrized problem (specifying the `"uncon"` option).

   These methods can also be used for the divergence criterion but only with the explicit boundary constraints (the reparametrized and unconstrained version is not implemented yet). For large problems where the number of parameters $Np = (N + n) * p$, (N is the number of rows of $\mathbf{A}$, n is the number of columns of $\mathbf{A}$ and p is the number of factors) is large, only limited memory BFGS and conjugate gradient should be used, since all other methods require $O(Np^2)$ memory for an (approximate) Hessian matrix.

3. In addition to the common MSE problem

$$f^{L2} = \|\mathbf{A} - \mathbf{WH}`\|_2^2$$

methods for estimating the parameters which optimize the *divergence criterion* (see Lee & Seung, 2001)

$$f^{div} = \sum_{ij}(\mathbf{A}_{ij} log(\mathbf{A}_{ij}/(\mathbf{WH^T})_{ij}) - \mathbf{A}_{ij} + (\mathbf{WH^T})_{ij}$$

were added. Two estimation methods are available:

- the Lee and Seung (2001) iterative gradient descent technique,
- the classical optimization methods like (dual) Quasi Newton (`"tech" "DUQUAN"`), Double Dogleg (`"tech" "DBLDOG"`), limited memory BFGS (`"tech" "LMEMQN"`), and conjugate gradient (`"tech" "CONGRA"`) for the boundary constrained problem.

The option `"div"` or `"diverg"` must be specified to select the algorithms using the divergence criterion.

The following shows some applications of the NLP algorithms to the small sparse binary data set. Also, fitting the divergence criterion using both, the Lee and Seung and the NLP algorithm is illustrated: These are the data:

```
print "Sparse Binary Test Example: nrow=29, ncol=9";
v =  [ 4  1  1,   7  1  1,   3  1  1,   5  1  1,   1  1  1,
       6  1  1,   2  1  2,   7  2  1,   9  2  1,  10  2  1,
      11  2  1,   8  2  1,   2  2  2,  13  3  1,  12  3  1,
      14  3  1,  11  3  1,   9  4  1,  15  4  1,   8  4  2,
       2  4  1,  16  4  1,   7  5  1,   9  5  1,  17  5  1,
      18  5  1,  10  5  1,  19  5  1,   2  5  2,  21  6  1,
       7  6  1,  12  6  1,  23  6  1,   3  6  1,  14  6  1,
      24  6  1,  22  6  1,   2  6  1,  20  6  1,   7  7  1,
       9  7  1,  17  7  1,  18  7  1,  15  7  1,  19  7  1,
       8  7  1,   2  7  1,   5  8  1,  15  8  1,  25  8  1,
      26  8  1,  28  8  1,   2  8  2,  27  8  1,  29  9  1,
      25  9  1,  26  9  1,   2  9  1 ];


nr = 29; nc = 9; nf = 2;
a = spmat(nr,nc,v[,1],v[,2],v[,3]);

rnam = [ "Row1" : "Row29" ];
cnam = [ "Col1" : "Col9" ];
a = rname(a,rnam);
a = cname(a,cnam);
print "A=", a; /* attrib(a); */
```

1. Using the NLP - CONGRA algorithm for MVN Criterion: Reparametrized Unconstrained:

```
optn = [ "print"          3 ,      /* ipri */
         "meth"      "nlpm" ,      /* imet=NLPM */
         "tech"    "congra" ,      /* opt tech=CONGRA */
         "uncon"            ,      /* unconstrained */
         "maxit"        400 ,      /* maxiter */
         "tol"        1.e-4 ,      /* term */
         "seed"           5 ];     /* seed */
 < w,h > = nnmf(a,nf,optn);
print "Matrix W:", w;
print "Matrix H:", h;
```

```
                Conjugate-Gradient Optimization
      Automatic Restart Update (Powell, 1977; Beale, 1972)
                    User Specified Gradient


  Iteration Start:
  N. Variables             76
  Criterion       98.78941989        Max Grad Entry  40.13129165


Iter rest nfun act   optcrit  difcrit maxgrad   alpha    slope
   1    0    3   0  54.16194 44.62748 8.77708 0.01126 -11212.1
   2    1    7   0  43.24937 10.91256 10.5465 0.05634 -233.691
   3    1    9   0  40.83906 2.410312 8.38129 0.01000 -334.147
   4    2   11   0  39.31714 1.521921 11.0225 0.03840 -307.139
   5    2   14   0  35.52007 3.797068 10.1440 0.04043 -171.877
   6    3   17   0  34.32169 1.198387 15.3566 0.02800 -206.596
   7    3   20   0  33.97451 0.347180 11.0643  3e-003 -189.736
   8    4   23   0  32.52891 1.445599 6.29884 0.01123 -563.637
   9    4   25   0  31.73161 0.797298 7.22337 0.03693 -45.8187
  10    5   27   0  31.02710 0.704514 4.09008 0.01000 -106.559
  11    5   29   0  30.65107 0.376030 1.44054 0.02136 -27.4217
  12    6   31   0  30.49364 0.157427 1.38370 0.02350 -13.3090
  13    6   33   0  30.32159 0.172052 1.29311 0.03305 -9.43735
  14    6   35   0  30.15385 0.167737 1.48065 0.03694 -9.89529
  15    7   37   0  30.05164 0.102213 1.19335 0.02268 -8.10923
  16    7   39   0  29.98208 0.069559 1.12017 0.02508 -5.52725
  17    7   41   0  29.88584 0.096240 1.35682 0.03776 -4.86831
  18    7   43   0  29.79602 0.089820 1.03646 0.03580 -5.12546
  19    7   45   0  29.73785 0.058168 0.76943 0.02705 -4.26396
  20    7   47   0  29.70338 0.034469 0.59084 0.02681 -2.46288
```

```
21    7   48   0  29.65957 0.043813 1.33480 0.10000 -1.29789
22    7   50   0  29.64220 0.017370 0.72617 0.01145 -3.19017
23    8   52   0  29.61886 0.023334 0.87540 0.02424 -2.05690
24    8   54   0  29.59832 0.020547 0.59364 0.02957 -1.35263
25    8   56   0  29.58143 0.016886 0.43855 0.03426 -0.95931
26    8   58   0  29.57247 9.0e-003 0.33651 0.02307 -0.73714
27    8   60   0  29.55741 0.015056 0.28047 0.04275 -0.69247
28    8   62   0  29.55327 4.1e-003 0.27504 0.02502 -0.32323
29    8   64   0  29.54609 7.2e-003 0.36385 0.04645 -0.31101
30    8   66   0  29.53893 7.2e-003 0.33628 0.03199 -0.43288
31    8   67   0  29.53301 5.9e-003 0.90068 0.10000 -0.33002
32    8   69   0  29.52596 7.1e-003 0.45535 0.01279 -0.89885
33    9   71   0  29.51819 7.8e-003 0.29178 0.01854 -0.81291
34    9   73   0  29.51205 6.1e-003 0.37153 0.03211 -0.38695
35    9   75   0  29.50219 9.9e-003 0.53196 0.04621 -0.42223
36    9   77   0  29.49242 9.8e-003 0.28976 0.02272 -0.78272
37   10   79   0  29.48616 6.3e-003 0.28503 0.03729 -0.33118
38   10   81   0  29.48164 4.5e-003 0.18476 0.02903 -0.31278
39   10   83   0  29.47748 4.2e-003 0.25359 0.04642 -0.17733
40   10   85   0  29.47243 5.0e-003 0.10675 0.02678 -0.34795

41   11   87   0  29.47094 1.5e-003 0.20433 0.04319 -0.06981
42   11   89   0  29.46954 1.4e-003 0.08816 0.02850 -0.09741
43   11   91   0  29.46829 1.3e-003 0.21823 0.04789 -0.05197
44   11   93   0  29.46685 1.4e-003 0.11105 0.02744 -0.10353
45   11   95   0  29.46586 9.9e-004 0.16885 0.02907 -0.06828
46   11   97   0  29.46439 1.5e-003 0.16292 0.03643 -0.07963
47   11   99   0  29.46313 1.3e-003 0.10837 0.02843 -0.08886
48   11  100   0  29.46234 7.9e-004 0.24001 0.10000 -0.05092
49   11  102   0  29.46168 6.6e-004 0.12783 0.01142 -0.11624
50   12  104   0  29.46065 1.0e-003 0.05952 0.02802 -0.07362
51   12  106   0  29.46032 3.3e-004 0.08907 0.03464 -0.01891
52   12  108   0  29.45971 6.1e-004 0.11512 0.04313 -0.02837
53   12  110   0  29.45925 4.6e-004 0.04608 0.01896 -0.04840
54   12  112   0  29.45898 2.7e-004 0.07201 0.04432 -0.01208
55   12  114   0  29.45872 2.7e-004 0.05001 0.02863 -0.01863
56   12  115   0  29.45850 2.2e-004 0.14630 0.10000 -0.01059
57   12  117   0  29.45832 1.8e-004 0.05900 0.01207 -0.02960
58   13  119   0  29.45812 2.0e-004 0.04333 0.02919 -0.01409
59   13  121   0  29.45801 1.0e-004 0.05519 0.03870  -5e-003
60   13  123   0  29.45784 1.7e-004 0.03520 0.03610  -9e-003
61   13  125   0  29.45781 3.6e-005 0.02610 0.01981  -4e-003
62   13  127   0  29.45773 7.5e-005 0.03056 0.04736  -3e-003
63   13  129   0  29.45769 3.7e-005 0.02628 0.01979  -4e-003
64   13  131   0  29.45764 5.4e-005 0.01483 0.05112  -2e-003
```

```
 65   13  133   0  29.45762 1.6e-005 0.01019 0.02264  -1e-003

 Successful Termination After    65 Iterations
 GCONV convergence criterion satisfied.
 Criterion       29.45762302       Max Grad Entry  0.010194410
 Slope SDirect. -0.001437778
 N. Function Calls       134       N. Gradient Calls        70
 N. Line Searches         65       Preproces. Time           0
 Time for Method           0       Effective Time            0


      L2 Precision : 29.4576   L1 Precision : 57.9905
                 Computer Time : 0 seconds
```

Matrix W:

	1	2
Row1	0.24994	0.31258
Row2	4.03134	1.20531
Row3	0.14168	1.23294
Row4	0.25011	0.31247
Row5	0.63815	0.26804
Row6	0.24976	0.31223
Row7	1.52230	1.11202
Row8	1.55421	0.00000
Row9	1.70504	1e-007
Row10	0.94914	0.00012
Row11	0.43227	0.22790
Row12	0.00002	1.14962
Row13	0.00002	0.24306
Row14	0.00002	1.14866
Row15	1.14134	0.00000
Row16	0.33919	0.00004
Row17	0.90924	0.00001
Row18	0.90939	0.00001
Row19	0.90932	0.00001
Row20	0.00000	0.90738
Row21	0.00001	0.90644
Row22	0.00000	0.90726
Row23	0.00000	0.90731
Row24	0.00000	0.90744
Row25	0.55946	0.00015
Row26	0.55973	0.00015
Row27	0.38654	0.00055
Row28	0.38482	0.00052

```
  Row29 |   0.17542   0.00426


Matrix H:
       |         1         2
---------------------------
  Col1 |   0.28410   0.34150
  Col2 |   0.45809   0.03805
  Col3 |    2e-007   0.24173
  Col4 |   0.33921    2e-008
  Col5 |   0.49155    4e-007
  Col6 |    4e-008   0.90747
  Col7 |   0.41659   0.00000
  Col8 |   0.38543   0.00001
  Col9 |   0.17508   0.00000
```

2. Using the NLP - LMEMQN algorithm for MVN Criterion: Reparametrized Unconstrained:

```
optn = [ "print"          3 ,      /* ipri */
         "ktc"              ,      /* print LM and KTc */
         "meth"      "nlpm" ,      /* imet=NLPM */
         "tech"    "lmemqn" ,      /* opt tech=LMEMQN */
         "uncon"            ,      /* unconstrained */
         "maxit"        400 ,      /* maxiter */
         "tol"        1.e-2 ,      /* term */
         "seed"           5 ];      /* seed */
< w,h > = nnmf(a,nf,optn);
print "Matrix W:", w;
print "Matrix H:", h;


              Limited Memory Quasi Newton Method
                    User Specified Gradient

  Iteration Start:
  N. Variables            76
  Criterion       98.78941989        Max Grad Entry  40.13129165
  N. Grad Storage          10


Iter nfun act  optcrit  difcrit maxgrad   alpha    slope   gcrit
   1    2   0 54.67382 44.11560 9.38814  9e-003 -11212.1 16.0070
   2    4   0 43.72293 10.95089 7.00302 5.00000 -2.78845 15.6427
```

```
 3    6   0 39.95295 3.769978 10.5931 0.24870 -21.3839 15.3948
 4    7   0 38.06904 1.883906 15.8151 1.00000 -6.79442 22.9104
 5    8   0 36.17317 1.895874 6.10007 1.00000 -4.21848 9.23589
 6    9   0 35.16416 1.009007 3.95639 1.00000 -1.13842 7.73968
 7   10   0 33.73835 1.425811 17.8595 1.00000 -5.31282 21.4752
 8   11   0 32.22885 1.509496 1.94392 1.00000 -3.60207 5.18031
 9   12   0 31.81764 0.411218 1.28720 1.00000 -0.50466 4.22842
10   13   0 31.00392 0.813719 1.51212 1.00000 -1.36124 4.21562
 ..........................................................


101  106   0 29.47366 1.8e-005 0.01878 1.00000  -3e-005 0.03585
102  107   0 29.47365 1.7e-005 0.01695 1.00000  -3e-005 0.03470
103  108   0 29.47362 2.1e-005 0.01247 1.00000  -3e-005 0.03208
104  109   0 29.47356 6.3e-005 0.02084 1.00000  -1e-004 0.05023
105  110   0 29.47350 5.7e-005 0.03011 1.00000  -9e-005 0.06669
106  111   0 29.47341 9.0e-005 0.03538 1.00000  -1e-004 0.07797
107  112   0 29.47309 3.2e-004 0.09527 1.00000  -6e-004 0.19618
108  113   0 29.47280 2.9e-004 0.16062 1.00000  -6e-004 0.27992
109  114   0 29.47212 6.8e-004 0.15812 1.00000  -9e-004 0.26738
110  116   0 29.46972 2.4e-003 0.17617 0.52357  -7e-003 0.29421
 ..........................................................


191  201   0 29.45722 7.9e-008  1e-003 1.00000  -1e-007  2e-003
192  202   0 29.45721 1.2e-007  1e-003 1.00000  -2e-007  2e-003
193  203   0 29.45721 5.6e-008  2e-003 1.00000  -2e-007  4e-003
194  204   0 29.45721 8.3e-008  9e-004 1.00000  -1e-007  2e-003
195  205   0 29.45721 6.8e-008  6e-004 1.00000  -1e-007  1e-003
196  206   0 29.45721 6.7e-008  1e-003 1.00000  -1e-007  2e-003
197  207   0 29.45721 2.7e-008  2e-003 1.00000  -1e-007  4e-003

 Successful Termination After   197 Iterations
 FCONV convergence criterion satisfied.
 Criterion       29.45721464        Max Grad Entry  0.002062513
 N. Grad Storage          10
 N. Function Calls       208        N. Gradient Calls       208
 N. Line Searches        197        Preproces. Time           0
 Time for Method           0        Effective Time            0


                 Lagrange Multipliers for W

                              1            2
                Row1   -0.000111482 -0.000113863
                Row2    0.000153769  8.2441e-005
```

Row3	-8.7993e-005	1.8062e-006
Row4	-9.1778e-005	-5.7975e-005
Row5	5.8933e-005	4.6990e-005
Row6	-1.2878e-005	0.000118787
Row7	-7.9420e-006	8.1444e-006
Row8	-0.000108417	-0.139545170
Row9	5.3087e-005	-0.157002651
Row10	-2.0423e-005	-0.070407456
Row11	-9.6579e-006	3.8041e-005
Row12	-0.131713804	-1.3901e-005
Row13	-0.027757473	-9.0297e-005
Row14	-0.131710316	1.6546e-005
Row15	-1.4619e-005	-0.130783805
Row16	4.1597e-005	-0.038865140
Row17	-2.2285e-005	-0.104082350
Row18	-1.2464e-005	-0.104081224
Row19	-1.0560e-005	-0.104081006
Row20	-0.103964947	1.2052e-006
Row21	-0.103964896	1.6516e-006
Row22	-0.103964766	2.7879e-006
Row23	-0.103964933	1.3228e-006
Row24	-0.103964806	2.4372e-006
Row25	-3.9460e-005	-0.064239762
Row26	-3.8248e-005	-0.064239693
Row27	3.0456e-005	-0.044174886
Row28	-5.3610e-005	-0.044184749
Row29	7.0584e-005	-0.020060559

Lagrange Multipliers for H

	1	2
Col1	0.000903317	0.000327090
Col2	-9.7933e-005	0.000252230
Col3	-1.319991808	-3.6869e-005
Col4	-2.0443e-005	-1.248136120
Col5	-0.000336134	-0.032602896
Col6	-0.867880447	-8.5427e-006
Col7	0.000760818	-0.694732073
Col8	8.8973e-005	-0.108804085
Col9	2.0108e-005	-0.060396633

Large positive Lagrange multipliers indicate nonoptimal solution.

Kuhn-Tucker Conditions for W

	1	2
Row1	-2.7633e-005	-3.5702e-005
Row2	0.000619784	9.9409e-005
Row3	-1.2540e-005	2.2266e-006
Row4	-2.2748e-005	-1.8175e-005
Row5	3.7620e-005	1.2624e-005
Row6	-3.1911e-006	3.7219e-005
Row7	-1.2092e-005	9.0649e-006
Row8	-0.000168397	-3.0299e-009
Row9	9.0543e-005	-3.6001e-009
Row10	-1.9400e-005	-1.4927e-008
Row11	-4.1699e-006	8.7898e-006
Row12	-3.6296e-010	-1.5977e-005
Row13	-6.3885e-011	-2.1871e-005
Row14	-5.9483e-010	1.9016e-005
Row15	-1.6683e-005	-3.6350e-009
Row16	1.4107e-005	-1.5677e-009
Row17	-2.0240e-005	-3.8310e-010
Row18	-1.1320e-005	-3.2544e-010
Row19	-9.5908e-006	-3.1996e-010
Row20	-1.0521e-011	1.0934e-006
Row21	-1.4305e-012	1.4983e-006
Row22	-4.8314e-012	2.5293e-006
Row23	-3.4757e-012	1.2001e-006
Row24	-3.1171e-011	2.2110e-006
Row25	-2.2120e-005	-1.5672e-008
Row26	-2.1441e-005	-2.0252e-008
Row27	1.1740e-005	-3.5382e-009
Row28	-2.0670e-005	-1.3791e-008
Row29	1.2348e-005	-2.4284e-007

Kuhn-Tucker Conditions for H

	1	2
Col1	0.000256262	0.000111814
Col2	-4.4876e-005	9.6990e-006
Col3	-2.4392e-010	-8.9269e-006
Col4	-6.9338e-006	-8.8310e-012
Col5	-0.000165271	-1.1305e-009
Col6	-3.3033e-012	-7.7501e-006
Col7	0.000316906	-9.3292e-010
Col8	3.4300e-005	-1.5378e-009
Col9	3.5193e-006	-1.5760e-011

Kuhn-Tucker values should be close to zero for optimality.

Matrix W:

	1	2
Row1	0.24787	0.31355
Row2	4.03062	1.20583
Row3	0.14251	1.23273
Row4	0.24786	0.31350
Row5	0.63835	0.26864
Row6	0.24780	0.31333
Row7	1.52260	1.11302
Row8	1.55324	2e-008
Row9	1.70558	2e-008
Row10	0.94993	2e-007
Row11	0.43176	0.23106
Row12	3e-009	1.14935
Row13	2e-009	0.24222
Row14	5e-009	1.14932
Row15	1.14124	3e-008
Row16	0.33914	4e-008
Row17	0.90824	4e-009
Row18	0.90823	3e-009
Row19	0.90823	3e-009
Row20	1e-010	0.90721
Row21	1e-011	0.90721
Row22	5e-011	0.90721
Row23	3e-011	0.90721
Row24	3e-010	0.90721
Row25	0.56056	2e-007
Row26	0.56056	3e-007
Row27	0.38548	8e-008
Row28	0.38556	3e-007
Row29	0.17495	0.00001

Matrix H:

	1	2
Col1	0.28369	0.34184
Col2	0.45823	0.03845
Col3	2e-010	0.24213
Col4	0.33918	7e-012
Col5	0.49168	3e-008
Col6	4e-012	0.90722
Col7	0.41653	1e-009
Col8	0.38551	1e-008
Col9	0.17502	3e-010

3. Using the NLP - LMEMQN algorithm for MVN Criterion: Boundary Constrained

```
optn = [ "print"          3 ,      /* ipri */
         "ktc"              ,      /* print LM and KTc */
         "meth"      "nlpm" ,      /* imet=NLPM */
         "tech"    "lmemqn" ,      /* opt tech=LMEMQN */
         "maxit"        400 ,      /* maxiter */
         "tol"        1.e-2 ,      /* term */
         "seed"           5 ];     /* seed */
< w,h > = nnmf(a,nf,optn);
print "Matrix W:", w;
print "Matrix H:", h;
```

```
              Limited Memory Quasi Newton Method
                    User Specified Gradient


  Iteration Start:
  N. Variables             76
  N. Bound. Constr.         0        N. Mask  Constr.          0
  Criterion       98.78941989        Max Grad Entry  22.64559517
  N. Active Constraints     0        N. Grad Storage          10


Iter nfun act  optcrit  difcrit maxgrad   alpha    slope   gcrit
   1    2  65 73.47906 25.31036 2.44961 0.10137 -317.404 2.02314
   2    3   2 47.14930 26.32976 4.17193 1.00000 -44.2226 3.28535
   3    4   2 41.02734 6.121966 1.82697 1.00000 -8.50631 1.82697
   4    5   2 37.17527 3.852066 2.48471 1.00000 -4.79325 2.48471
   5    6   6 33.53237 3.642900 1.31315 1.00000 -5.03331 1.04296
   6    7   9 32.13014 1.402233 1.44021 1.00000 -1.74902 1.44021
   7    8   9 31.39028 0.739858 1.45387 1.00000 -1.04539 0.76790
   8    9  11 30.51717 0.873110 0.75889 1.00000 -1.13048 0.61035
   9   10  18 29.94711 0.570062 0.48903 1.00000 -0.75366 0.48903
  10   11  21 29.72476 0.222347 0.65612 1.00000 -0.42923 0.65612


Iter nfun act  optcrit  difcrit maxgrad   alpha    slope   gcrit
  61   64  29 29.01868 7.4e-009  1e-004 1.00000  -2e-008  1e-004
  62   65  29 29.01868 1.9e-009  1e-004 1.00000  -3e-009  1e-004
  63   66  29 29.01868 1.0e-009  6e-005 1.00000  -2e-009  6e-005
```

```
64   67  29 29.01868 5.0e-010  3e-005 1.00000  -9e-010  3e-005
65   68  29 29.01868 2.4e-010  2e-005 1.00000  -5e-010  2e-005
66   69  29 29.01868 1.4e-010  2e-005 1.00000  -2e-010  2e-005
67   70  29 29.01868 1.3e-010  1e-005 1.00000  -3e-010  1e-005
68   71  29 29.01868 1.9e-011  1e-005 1.00000  -6e-011  1e-005
69   72  29 29.01868 1.3e-011  5e-006 1.00000  -2e-011  5e-006

Successful Termination After    69 Iterations
ABSGCONV convergence criterion satisfied.
Criterion       29.01868024        Max Grad Entry  4.8328e-006
N. Active Constraints    29        N. Grad Storage          10
Skipped Updates           0        Segments Explored       153
N. Function Calls        73        N. Gradient Calls        73
N. Line Searches         69        Preproces. Time           0
Time for Method           0        Effective Time            0


                    ********************
                    Optimization Results
                    ********************

                    Parameter Estimates
                    -------------------

       Parameter      Estimate   Gradient   Active BC

         1          0.00000000  0.2011190  Lower BC
         2          0.45851455  1.21e-006
         3          1.96376751  6.77e-007
         4          2.45597201 -6.02e-007
         5          0.00000000  0.3785930  Lower BC
         6          0.80855975  1.73e-006
         7          0.00000000  0.2011190  Lower BC
         8          0.45851455  1.23e-006
         9          0.00000000  0.3877336  Lower BC
        10          0.88553457 -4.71e-007
        11          0.00000000  0.2011190  Lower BC
        12          0.45851447  9.61e-007
        13          1.06797600 -1.05e-006
        14          0.86032370  1.63e-006
        15          1.47157964  1.91e-006
        16          0.00000000  0.3331886  Lower BC
        17          1.53195271  3.77e-007
        18          0.00000000  0.0575196  Lower BC
        19          0.74881785 -3.64e-006
        20          0.09889650  2.95e-006
```

```
21          0.34080603  2.18e-007
22          0.12101720  1.26e-006
23          0.00000000  0.2033707  Lower BC
24          0.40112323  1.13e-006
25          0.00000000  0.0258967  Lower BC
26          0.05107792  2.64e-007
27          0.00000000  0.2033706  Lower BC
28          0.40112308  6.44e-007
29          0.72929756 -5.38e-007
30          0.31932234  2.80e-006


31          0.34432504  1.11e-006
32          0.00000000  0.1745738  Lower BC
33          0.82868748 -1.24e-006
34          0.00000000  0.1138688  Lower BC
35          0.82868752 -1.13e-006
36          0.00000000  0.1138688  Lower BC
37          0.82868752 -1.11e-006
38          0.00000000  0.1138688  Lower BC
39          0.00000000  0.1774740  Lower BC
40          0.35004520  5.06e-007
41          0.00000000  0.1774740  Lower BC
42          0.35004533  9.36e-007
43          0.00000000  0.1774740  Lower BC
44          0.35004524  6.27e-007
45          0.00000000  0.1774740  Lower BC
46          0.35004521  5.34e-007
47          0.00000000  0.1774740  Lower BC
48          0.35004522  5.49e-007
49          0.00000000  0.2910077  Lower BC
50          0.63292235 -2.03e-006
51          0.00000000  0.2910077  Lower BC
52          0.63292234 -2.05e-006
53          0.00000000  0.1866146  Lower BC
54          0.42702016 -1.21e-006
55          0.00000000  0.1866146  Lower BC
56          0.42702013 -1.30e-006
57          0.00000000  0.1043929  Lower BC
58          0.20590201 -1.43e-006
59          0.01567463 -2.42e-006
60          0.78709378 -2.45e-006


61          0.60723137  2.50e-006
62          0.20645340  2.99e-006
```

```
  63          0.00000000  0.3863343  Lower BC
  64          0.08768115  1.02e-006
  65          0.58250612 -2.42e-006
  66          0.00000000  1.5442693  Lower BC
  67          0.68464211 -1.78e-006
  68          0.15313912 -3.43e-007
  69          0.00000000  1.2553296  Lower BC
  70          0.60089369  3.49e-006
  71          0.71727801 -1.22e-006
  72          0.00000000  1.4651282  Lower BC
  73          0.01494294  4.83e-006
  74          0.73303115  1.45e-006
  75          0.00000000  0.3775169  Lower BC
  76          0.35345585 -5.88e-007

  Value of Objective Function =      29.0187

L2 Precision : 29.0187   L1 Precision : 56.2957
          Computer Time : 0 seconds


          Lagrange Multipliers for W

                      1            2
       Row1   -0.077313790 -4.6338e-007
       Row2   -2.6039e-007  2.2965e-007
       Row3   -0.145538006 -6.5872e-007
       Row4   -0.077313791 -4.6756e-007
       Row5   -0.149051811  1.7978e-007
       Row6   -0.077313776 -3.6669e-007
       Row7    4.0411e-007 -6.2325e-007
       Row8   -7.3342e-007 -0.127152012
       Row9   -1.4511e-007 -0.021950748
       Row10   1.3988e-006 -1.1251e-006
       Row11  -8.3792e-008 -4.8018e-007
       Row12  -0.078179381 -4.3126e-007
       Row13  -0.009955145 -1.0086e-007
       Row14  -0.078179354 -2.4564e-007
       Row15   2.0692e-007 -1.0671e-006
       Row16  -4.2594e-007 -0.066621181
       Row17   4.7789e-007 -0.043454824
       Row18   4.3331e-007 -0.043454830
       Row19   4.2780e-007 -0.043454831
       Row20  -0.068224216 -1.9298e-007
       Row21  -0.068224240 -3.5738e-007
       Row22  -0.068224223 -2.3924e-007
```

Row23	-0.068224217	-2.0381e-007
Row24	-0.068224218	-2.0959e-007
Row25	-0.111868624	7.7428e-007
Row26	-0.111868622	7.8408e-007
Row27	-0.071738048	4.6241e-007
Row28	-0.071738043	4.9569e-007
Row29	-0.040130541	5.4667e-007

Lagrange Multipliers for H

	1	2
Col1	1.5749e-006	1.6069e-006
Col2	-1.6269e-006	-1.9590e-006
Col3	-0.251246158	-6.7066e-007
Col4	1.5767e-006	-1.011649104
Col5	1.1563e-006	2.2470e-007
Col6	-0.816382941	-2.2876e-006
Col7	7.9642e-007	-0.959803821
Col8	-3.1429e-006	-9.5089e-007
Col9	-0.245511884	3.8534e-007

Large positive Lagrange multipliers indicate nonoptimal solution.

Kuhn-Tucker Conditions for W

	1	2
Row1	0.000000000	-2.7837e-007
Row2	-6.6508e-007	7.3896e-007
Row3	0.000000000	-6.9783e-007
Row4	0.000000000	-2.8088e-007
Row5	0.000000000	2.0859e-007
Row6	0.000000000	-2.2028e-007
Row7	5.6134e-007	-7.0253e-007
Row8	-1.4038e-006	0.000000000
Row9	-2.8914e-007	0.000000000
Row10	1.3624e-006	-1.4578e-007
Row11	-3.7143e-008	-7.6135e-008
Row12	0.000000000	-2.2665e-007
Row13	0.000000000	-6.7500e-009
Row14	0.000000000	-1.2910e-007
Row15	1.9628e-007	-4.4643e-007
Row16	-1.9076e-007	0.000000000
Row17	5.1509e-007	0.000000000
Row18	4.6704e-007	0.000000000
Row19	4.6111e-007	0.000000000

Row20	0.000000000	-8.8505e-008
Row21	0.000000000	-1.6390e-007
Row22	0.000000000	-1.0972e-007
Row23	0.000000000	-9.3472e-008
Row24	0.000000000	-9.6124e-008
Row25	0.000000000	6.4208e-007
Row26	0.000000000	6.5020e-007
Row27	0.000000000	2.5871e-007
Row28	0.000000000	2.7733e-007
Row29	0.000000000	1.4748e-007

Kuhn-Tucker Conditions for H

	1	2
Col1	1.8979e-008	9.6536e-007
Col2	-7.5955e-007	-3.0869e-007
Col3	0.000000000	-4.4882e-008
Col4	7.0614e-007	0.000000000
Col5	6.0866e-007	2.6264e-008
Col6	0.000000000	-1.0491e-006
Col7	4.3920e-007	0.000000000
Col8	-3.6108e-008	-5.3200e-007
Col9	0.000000000	1.0395e-007

Kuhn-Tucker values should be close to zero for optimality.

Matrix W:

	1	2
Row1	0.00000	0.60074
Row2	2.55421	3.21781
Row3	0.00000	1.05937
Row4	0.00000	0.60074
Row5	0.00000	1.16023
Row6	0.00000	0.60074
Row7	1.38908	1.12719
Row8	1.91404	0.00000
Row9	1.99256	0.00000
Row10	0.97396	0.12957
Row11	0.44328	0.15856
Row12	0.00000	0.52555
Row13	0.00000	0.06692
Row14	0.00000	0.52555
Row15	0.94857	0.41838
Row16	0.44785	0.00000

```
  Row17 |   1.07785   0.00000
  Row18 |   1.07785   0.00000
  Row19 |   1.07785   0.00000
  Row20 |   0.00000   0.45863
  Row21 |   0.00000   0.45863
  Row22 |   0.00000   0.45863
  Row23 |   0.00000   0.45863
  Row24 |   0.00000   0.45863
  Row25 |   0.00000   0.82925
  Row26 |   0.00000   0.82925
  Row27 |   0.00000   0.55948
  Row28 |   0.00000   0.55948
  Row29 |   0.00000   0.26977


Matrix H:
       |         1         2
----------------------------
  Col1 |   0.01205   0.60074
  Col2 |   0.46686   0.15757
  Col3 |   0.00000   0.06692
  Col4 |   0.44785   0.00000
  Col5 |   0.52638   0.11688
  Col6 |   0.00000   0.45863
  Col7 |   0.55147   0.00000
  Col8 |   0.01149   0.55948
  Col9 |   0.00000   0.26977
```

4. Using the Divergence Criterion: Lee and Seung (2001) Algorithm

```
  /*--- without start: Lee and Seung : diverg crit --------------------*/
  optn = [ "print"          2 ,      /* ipri */
           "meth"     "le_se" ,      /* imet=Lee&Seung */
           "div"              ,      /* divergence crit */
           "maxit"        400 ,      /* maxiter */
           "tol"        1.e-4 ,      /* term */
           "spar"       1.e-6 ,      /* spar: this must be small */
           "seed"           5 ];     /* seed */
  < w,h > = nnmf(a,nf,optn);
  print "Matrix W:", w;
  print "Matrix H:", h;


                  Nonnegative Matrix Factorization
```

Lee and Seung : Divergence Criterion

Iter	DivCrit	DiffCrit	L2Crit	L1Crit
1	67.2530413	64.1090130	37.9950500	74.3988213
2	57.4070642	9.84597706	33.4726096	67.5121960
3	52.0537402	5.35332407	31.8483313	62.9296746
4	50.5408866	1.51285358	31.5692414	61.5683179
5	50.0387588	0.50212783	31.4813819	61.1506904
6	49.8152613	0.22349752	31.4787125	60.9825576
7	49.6935221	0.12173911	31.5045851	60.9029977
8	49.6185150	0.07500711	31.5338144	60.8576293
9	49.5682335	0.05028151	31.5600457	60.8251475
10	49.5321499	0.03608361	31.5823211	60.7979974
11	49.5048965	0.02725336	31.6009318	60.7740396
12	49.4836833	0.02121322	31.6163814	60.7529290
13	49.4668965	0.01678681	31.6291190	60.7346336
14	49.4535425	0.01335397	31.6395387	60.7190971
15	49.4429223	0.01062020	31.6479879	60.7061439
16	49.4345008	0.00842158	31.6547805	60.6955103
17	49.4278484	0.00665240	31.6601982	60.6868902
18	49.4226065	0.00524183	31.6644930	60.6799655
19	49.4185069	0.00409962	31.6678694	60.6744606
20	49.4152974	0.00320955	31.6705212	60.6701023
21	49.4128007	0.00249668	31.6725874	60.6666746
22	49.4108578	0.00194290	31.6741977	60.6639860
23	49.4093487	0.00150909	31.6754482	60.6618834
24	49.4081783	0.00117040	31.6764174	60.6602430
25	49.4072716	9.067e-004	31.6771676	60.6589657
26	49.4065698	7.018e-004	31.6777476	60.6579725
27	49.4060270	5.428e-004	31.6781955	60.6572011
28	49.4056074	4.196e-004	31.6785413	60.6566025
29	49.4052832	3.243e-004	31.6788081	60.6561383
30	49.4050327	2.505e-004	31.6790139	60.6557785
31	49.4048392	1.935e-004	31.6791725	60.6554998
32	49.4046898	1.494e-004	31.6792948	60.6552840
33	49.4045745	1.154e-004	31.6793891	60.6551169
34	49.4044854	8.906e-005	31.6794618	60.6549876

Divergence : 49.4045 L2 Precision : 31.6795 L1 Precision : 60.655

Computer Time : 0 seconds

	1	2

Row1	0.00000	0.52068
Row2	3.21009	2.60315
Row3	0.00000	1.04137
Row4	0.00000	0.52068
Row5	0.00000	1.04137
Row6	0.00000	0.52068
Row7	1.37565	1.04137
Row8	1.83420	0.00000
Row9	1.83420	0.00000
Row10	0.91710	0.00000
Row11	0.45855	0.52068
Row12	0.00000	1.04137
Row13	0.00000	0.52068
Row14	0.00000	1.04137
Row15	0.91727	0.52049
Row16	0.45855	0.00000
Row17	0.91710	0.00000
Row18	0.91710	0.00000
Row19	0.91710	0.00000
Row20	0.00000	0.52068
Row21	0.00000	0.52068
Row22	0.00000	0.52068
Row23	0.00000	0.52068
Row24	0.00000	0.52068
Row25	0.45863	0.52059
Row26	0.45863	0.52059
Row27	0.00000	0.52068
Row28	0.00000	0.52068
Row29	0.45855	0.00000

Matrix H:

	1	2
Col1	0.00000	0.51217
Col2	0.46257	0.00000
Col3	0.00000	0.25608
Col4	0.39649	0.00000
Col5	0.52866	0.00000
Col6	0.00000	0.64021
Col7	0.52866	0.00000
Col8	0.00008	0.51209
Col9	0.26433	0.00000

5. Using the Divergence Criterion: NLP - LMEMQN Algorithm

```
/*--- without start: NLP LMEMQN : diverg crit --------------------*/
optn = [ "print"          2 ,      /* ipri */
         "meth"      "nlpm" ,      /* imet=NLP */
         "tech"    "lmemqn" ,      /* NLPtech=LMEMQN */
         "div"              ,      /* divergence crit */
         "maxit"        400 ,      /* maxiter */
         "tol"        1.e-2 ,      /* term */
         "spar"       1.e-6 ,      /* spar: this must be small */
         "seed"           5 ];     /* seed */
< w,h > = nnmf(a,nf,optn);
print "Matrix W:", w;
print "Matrix H:", h;
```

```
              Limited Memory Quasi Newton Method
                    User Specified Gradient

  Iteration Start:
  N. Variables             76
  N. Bound. Constr.         0        N. Mask  Constr.          0
  Criterion       131.3620543        Max Grad Entry  14.15900144
  N. Active Constraints     0        N. Grad Storage          10

Iter nfun act  optcrit  difcrit maxgrad   alpha    slope   gcrit
   1    2  62 118.6154 12.74664 8.23326 0.06940 -339.868 1.62024
   2    3   5 77.96629 40.64912 12.7533 1.00000 -67.2015 15.8777
   3    4   5 71.77752 6.188777 7.49549 1.00000 -14.7915 6.16743
   4    5   8 66.75111 5.026408 4.57577 1.00000 -5.75082 3.38992
   5    6  11 62.56379 4.187322 2.84446 1.00000 -4.57609 2.01025
   6    7  18 55.41199 7.151797 7.65013 1.00000 -8.50998 7.65013
   7    8  19 53.53373 1.878265 3.17197 1.00000 -2.58694 2.24079
   8    9  22 52.04522 1.488505 3.20784 1.00000 -1.55402 1.43159
   9   10  27 50.37416 1.671061 1.85432 1.00000 -1.71509 0.89163
  10   11  30 49.59162 0.782539 1.79799 1.00000 -0.80473 0.91625
  11   12  31 48.59144 1.000178 1.87873 1.00000 -1.12037 0.93079
  12   13  32 48.37295 0.218492 1.87039 1.00000 -0.22589 0.66562
  13   14  33 48.07979 0.293162 1.86324 1.00000 -0.33388 0.51116
  14   15  34 47.66714 0.412655 0.53287 1.00000 -0.46159 0.53287
  15   16  34 47.62529 0.041845 0.36378 1.00000 -0.05168 0.36378
  16   17  34 47.58050 0.044795 0.24929 1.00000 -0.07346 0.24929
  17   18  34 47.56526 0.015234 0.24582 1.00000 -0.01976 0.24582
  18   19  34 47.54966 0.015602 0.11308 1.00000 -0.02615 0.11308
  19   20  34 47.54673 2.9e-003 0.11045 1.00000  -5e-003 0.11045
  20   21  34 47.54578 9.6e-004 0.06170 1.00000  -2e-003 0.06170
  21   22  34 47.54534 4.4e-004 0.04650 1.00000  -5e-004 0.04650
  22   23  34 47.54490 4.4e-004 0.02411 1.00000  -8e-004 0.02411
```

```
  23   24  34 47.54484 5.6e-005 0.03282 1.00000  -1e-004 0.03282
  24   25  34 47.54481 3.5e-005 0.01169 1.00000  -5e-005 0.01169
  25   26  34 47.54479 1.9e-005  5e-003 1.00000  -3e-005  5e-003
  26   27  34 47.54478 7.4e-006  5e-003 1.00000  -1e-005  5e-003
  27   28  34 47.54477 7.9e-006  2e-003 1.00000  -1e-005  2e-003
  28   29  34 47.54477 1.7e-006  2e-003 1.00000  -2e-006  2e-003
  29   30  34 47.54477 7.6e-007  1e-003 1.00000  -1e-006  1e-003
  30   31  34 47.54477 5.4e-007  1e-003 1.00000  -8e-007  1e-003


Iter nfun act  optcrit  difcrit maxgrad   alpha    slope   gcrit
  31   32  34 47.54477 3.7e-007  5e-004 1.00000  -6e-007  5e-004
  32   33  34 47.54477 1.4e-007  7e-004 1.00000  -2e-007  7e-004
  33   34  34 47.54477 5.1e-008  7e-004 1.00000  -1e-007  7e-004
  34   35  34 47.54477 3.5e-008  3e-004 1.00000  -5e-008  3e-004
  35   36  34 47.54477 2.1e-008  2e-004 1.00000  -3e-008  2e-004
  36   37  34 47.54477 7.1e-009  9e-005 1.00000  -1e-008  9e-005
  37   38  34 47.54477 2.8e-009  1e-004 1.00000  -5e-009  1e-004
  38   39  34 47.54477 2.0e-010  1e-004 1.00000  -2e-009  1e-004
  39   40  34 47.54477 4.7e-010  1e-005 1.00000  -9e-010  1e-005
  40   41  34 47.54477 4.5e-011  9e-006 1.00000  -6e-011  9e-006

  Successful Termination After    40 Iterations
  ABSGCONV convergence criterion satisfied.
  Criterion       47.54476724        Max Grad Entry  8.8336e-006
  N. Active Constraints    34        N. Grad Storage          10
  Skipped Updates           0        Segments Explored       132
  N. Function Calls        42        N. Gradient Calls        42
  N. Line Searches         40        Preproces. Time           0
  Time for Method           0        Effective Time            0


                     ********************
                     Optimization Results
                     ********************

                     Parameter Estimates
                     -------------------

         Parameter      Estimate   Gradient   Active BC

           1          0.00000000  3.5185713  Lower BC
           2          0.52768677  1.72e-006
           3          2.55785523 -1.53e-006
           4          1.58305793 -1.12e-006
           5          0.00000000  3.5185713  Lower BC
```

```
 6          1.05536898 -6.46e-006
 7          0.00000000  3.5185713  Lower BC
 8          0.52768610 -6.53e-007
 9          0.28420624 -2.52e-007
10          0.52768788  5.72e-006
11          0.00000000  3.5185713  Lower BC
12          0.52768660  1.14e-006
13          0.85261916  1.58e-006
14          1.05537474  3.89e-006
15          1.13682393 -3.42e-006
16          0.00000000  1.8950654  Lower BC
17          1.13682406 -3.04e-006
18          0.00000000  1.8950654  Lower BC
19          0.56841267  9.10e-007
20          0.00000000  1.8950654  Lower BC
21          0.28420665  4.89e-006
22          0.52768758  4.65e-006
23          0.00000000  3.5185713  Lower BC
24          1.05536765 -8.83e-006
25          0.00000000  3.5185713  Lower BC
26          0.52768662  1.20e-006
27          0.00000000  3.5185713  Lower BC
28          1.05537257 -8.81e-009
29          0.85262005  5.25e-006
30          0.00000000  1.8950654  Lower BC
31          0.28420657  3.91e-006
32          0.00000000  1.8950654  Lower BC
33          0.56841170 -5.09e-006
34          0.00000000  1.8950654  Lower BC
35          0.56841325  4.55e-006
36          0.00000000  1.8950654  Lower BC
37          0.56841341  5.52e-006
38          0.00000000  1.8950654  Lower BC
39          0.00000000  3.5185713  Lower BC
40          0.52768721  3.31e-006
41          0.00000000  3.5185713  Lower BC
42          0.52768625 -1.20e-007
43          0.00000000  3.5185713  Lower BC
44          0.52768659  1.08e-006
45          0.00000000  3.5185713  Lower BC
46          0.52768719  3.24e-006
47          0.00000000  3.5185713  Lower BC
48          0.52768670  1.50e-006
49          0.56841201 -3.14e-006
50          0.00000000  1.8950654  Lower BC
```

```
          51          0.56841240 -7.33e-007
          52          0.00000000  1.8950654  Lower BC
          53          0.28420633  9.10e-007
          54          0.00000000  1.8950654  Lower BC
          55          0.28420682  6.91e-006
          56          0.00000000  1.8950654  Lower BC
          57          0.28420652  3.25e-006
          58          0.00000000  1.8950654  Lower BC
          59          0.00000000  5.0091395  Lower BC
          60          0.68911470 -5.57e-007
          61          0.60073168  2.15e-007
          62          0.00000000  4.3973784  Lower BC
          63          0.00000000  10.089324  Lower BC
          64          0.34455738  6.45e-007
          65          0.51491288  4.48e-007
          66          0.00000000  10.407145  Lower BC
          67          0.68655041 -1.15e-006
          68          0.00000000  8.0032380  Lower BC
          69          0.00000000  8.8388160  Lower BC
          70          0.86139327 -1.89e-006
          71          0.68655045 -5.28e-007
          72          0.00000000  8.9047022  Lower BC
          73          0.68655042 -1.04e-006
          74          0.00000000  7.1017684  Lower BC
          75          0.34327548  8.13e-006
          76          0.00000000  9.8061707  Lower BC

          Value of Objective Function =      47.5448


       L2 Precision : 31.4442   L1 Precision : 59.3171
                 Computer Time : 0 seconds


Matrix W:
        |         1         2
----------------------------
   Row1 |   0.00000   0.60984
   Row2 |   3.75744   1.82951
   Row3 |   0.00000   1.21967
   Row4 |   0.00000   0.60984
   Row5 |   0.41749   0.60984
   Row6 |   0.00000   0.60984
   Row7 |   1.25248   1.21968
   Row8 |   1.66997   0.00000
   Row9 |   1.66997   0.00000
```

```
  Row10 |   0.83499   0.00000
  Row11 |   0.41749   0.60984
  Row12 |   0.00000   1.21967
  Row13 |   0.00000   0.60984
  Row14 |   0.00000   1.21967
  Row15 |   1.25248   0.00000
  Row16 |   0.41749   0.00000
  Row17 |   0.83499   0.00000
  Row18 |   0.83499   0.00000
  Row19 |   0.83499   0.00000
  Row20 |   0.00000   0.60984
  Row21 |   0.00000   0.60984
  Row22 |   0.00000   0.60984
  Row23 |   0.00000   0.60984
  Row24 |   0.00000   0.60984
  Row25 |   0.83499   0.00000
  Row26 |   0.83499   0.00000
  Row27 |   0.41749   0.00000
  Row28 |   0.41749   0.00000
  Row29 |   0.41749   0.00000
```

```
Matrix H:
       |         1         2
---------------------------
  Col1 |   0.00000   0.59628
  Col2 |   0.40894   0.00000
  Col3 |   0.00000   0.29814
  Col4 |   0.35052   0.00000
  Col5 |   0.46736   0.00000
  Col6 |   0.00000   0.74536
  Col7 |   0.46736   0.00000
  Col8 |   0.46736   0.00000
  Col9 |   0.23368   0.00000
```

This solution with objective function 47.54476724 seems to be much better than that obtained by the Lee and Seung algorithm with objective function 49.4045.

3 New Developments

3.1 Function `canals`

<gof,eval,xload,yload,xscor,yscor,xcqua,ycqua> = canals(xydata,scale,xind,yind,optn)

Purpose: The `canals` function performs canonical correlation analysis for categorical data, like data with nominally or ordinally scaled variables. The iterative algorithm is based on the Gifi (1990) suite of programs and the input data can have some missing values. The results are not only variable loadings and observation scores, estimated are also general category quantifications of all variables and the optimally scaled data set.

Input: data This argument must be a $N \times n$ matrix containing the measured levels of n categorical variables. The levels can be integer or string coded. A sufficiently small number of missing values are permitted.

scale This argument must be an n integer vector each entry defining the scale property of a variable. The following scaling options are permitted:

scale=0 nominal

scale=1 ordinal

scale=2 numerical

Note that the dimension of this vector must be the same as the column number of the data matrix provided for the first argument.

xind Numbers of columns of data matrix which belong to **X** matrix. Must be either int scalar or vector.

yind Numbers of columns of data matrix which belong to **Y** matrix. Must be either int scalar or vector.

optn This argument must be specified in form of a two column matrix where the first column defines the option as string value (in quotes) and the second column can be used for a numeric or string specification of the option. See table below for content.

Option	Second Column	Meaning
"ftol"	real	stress termination tolerance for iteration (default is 0.001)
"maxit"	int	maximum number of iterations for iterations (default is 200)
"nfac"	int	number of factors (dimension) of decomposition
"orig"		original version of algorithm
"phis"		print optimization history
"print"	int	indicates the amount of printed output (=0: no printed output, default is 2)
"plot"	int	indicates the amount of plot output (=0: no plot output is default)
"seed"	int	seed for random generator (valid only when "orig" is not specified)
"xtol"	real	termination tolerance for (intermediate) weight iteration (default is 0.0001)

The orig option currently refers only to one small difference: it specifies that the original (and obviously not so great) random generator is being used for generating initial values of the object score matrix before the initial iteration can start.

Output: There are at most eight results returned:

gof a vector of scalar results, like an inidicator for the success of the iterations, the computer time used and the values of differently defined *stress* for the final solution.

eval a vector of $nfac$ eigenvalues.

xload the $nx \times nfac$ matrix of factor loadings of **X**

yload the $ny \times nfac$ matrix of factor loadings of **Y**

xscor the $N \times nfac$ matrix of object scores of **X**

yscor the $N \times nfac$ matrix of object scores of **Y**

xcqua the $Ncatx$ vector of category quantifications of all variables of **X**

ycqua the $Ncaty$ vector of category quantifications of all variables of **Y**

Restrictions: 1. String data for levels are permitted. The second, third, and fourth input arguments can have no missing values.

2. Many missing values or difficult missing value patterns may affect the convergence of the iterative processes.

Relationships: cancorr(), overals(), homals(), anacor(), anaprof(), princals(), primals()

Examples: 1. Economic Inequality and Polit. Instability: nobs=47, nvar=9, nx=5, ny=4 (Gifi, 1990, p. 225)

```
print "CANALS: Economic Inequality and Polit. Instability";
print "Third Var Missing=9: Cats: 6 5 5 8 5 6 3 5 3";
data =
 [ 1 1 1 4 4 1 2 1 3,   1 1 1 5 3 3 2 2 3,  1 1 2 6 2 5 1 1 1,   1 2 2 3 2 6 2 2
   1 2 2 7 1 4 2 1 1,   1 2 3 7 1 3 1 1 1,  2 3 5 1 4 2 3 2 1,   2 3 4 3 3 5 2 4
   2 3 3 7 1 3 1 1 1,   2 3 4 6 2 6 3 2 2,  2 3 5 6 1 6 2 2 1,   2 3 2 5 2 5 2 1
   2 3 2 6 3 6 2 1 2,   3 3 5 6 1 5 2 1 1,  3 3 3 7 2 5 1 1 1,   3 4 4 2 3 1 2 1
   3 3 2 6 2 5 2 1 1,   3 4 3 2 4 4 3 5 3,  3 4 2 6 1 2 2 1 2,   3 4 2 1 4 5 2 1
   4 5 3 8 1 5 2 1 1,   4 4 5 6 1 5 2 1 1,  4 4 3 4 3 6 2 3 3,   4 5 2 5 2 5 2 1
   4 5 3 2 4 6 3 2 3,   4 5 3 3 3 6 2 2 2,  4 5 3 2 4 5 3 4 3,   4 5 . 4 4 5 2 2
   4 5 4 7 1 5 1 1 1,   4 5 5 4 3 1 2 2 3,  4 5 5 4 3 5 3 5 3,   4 5 3 3 3 4 2 3
   5 5 4 5 2 6 3 2 2,   5 5 4 5 2 5 2 2 1,  5 5 3 3 4 6 2 2 3,   5 5 2 4 4 6 3 2
   5 5 3 4 3 5 3 4 2,   5 5 3 3 4 5 3 3 3,  5 5 4 4 2 5 3 4 2,   5 5 3 3 3 6 3 2
   5 5 . 2 3 5 2 3 3,   5 5 5 3 5 6 2 4 3,  5 5 2 4 3 5 2 3 2,   6 5 3 6 3 5 2 4
   6 5 . 7 1 4 1 1 1,   6 5 3 3 2 5 2 2 2,  6 5 3 1 4 6 3 5 3 ];


cnam = [ "GINI" "FARM" "RENT" "GNPR" "LABO"
         "INST" "ECKS" "DEAT" "DEMO" ];
rnam = [ "Yug" "Pol" "Den" "Jap" "Can" "Swi" "Ind" "Phi" "Swe" "Fra"
         "Bel" "Ire" "Fin" "Net" "Lux" "Tai" "Nor" "SVi" "WGe" "Lib"
         "USA" "UnK" "Pan" "Aut" "Egy" "Gre" "Hon" "Nic" "NZe" "Spa"
         "Cub" "DoR" "Ita" "Uru" "ElS" "Bra" "Col" "Gua" "Arg" "Ecua"
         "Per" "Ira" "Cos" "Ven" "Aus" "Chi" "Bol" ];
data = cname(data,cnam);
data = rname(data,rnam);
/* print "Data=", data; */


ctyp = [ 9 # 0 ];
xind = [ 1:5 ]; yind = [ 6:9 ];
optn = [ "nfac"       2 ,
         "orig"         ,
         "maxit"    100 ,
         "xtol"   1.e-4 ,
         "ftol"   1.e-3 ,
         "plot"       3 ,
         "print"      3 ,
         "phis"         ];
< gof,eval,xload,yload,xscor,yscor,xcatq,ycatq >
     = canals(data,ctyp,xind,yind,optn);


            ************************************
            Number of Observations for Class Levels
            ************************************
```

Variable	Value	Nobs	Proportion
C[1]	1	6	12.765957
	2	7	14.893617
	3	7	14.893617
	4	12	25.531915
	5	11	23.404255
	6	4	8.510638
C[2]	1	3	6.382979
	2	3	6.382979
	3	10	21.276596
	4	6	12.765957
	5	25	53.191489
C[3]	1	2	4.255319
	2	11	23.404255
	3	17	36.170213
	4	7	14.893617
	5	7	14.893617
	_Missing	3	6.382979
C[4]	1	3	6.382979
	2	5	10.638298
	3	9	19.148936
	4	9	19.148936
	5	5	10.638298
	6	9	19.148936
	7	6	12.765957
	8	1	2.127660
C[5]	1	10	21.276596
	2	11	23.404255
	3	14	29.787234
	4	11	23.404255
	5	1	2.127660
C[6]	1	3	6.382979
	2	2	4.255319
	3	3	6.382979
	4	4	8.510638
	5	22	46.808511
	6	13	27.659574
C[7]	1	6	12.765957
	2	28	59.574468

```
                    3      13   27.659574

C[8]                1      18   38.297872
                    2      15   31.914894
                    3       5   10.638298
                    4       6   12.765957
                    5       3    6.382979

C[9]                1      16   34.042553
                    2      12   25.531915
                    3      19   40.425532


       Summary on Variables in X
       *************************

Var    Type   N_Lev   Frequ  N_Miss  N_Nonz
  1 nominal       6      47       0      47
  2 nominal       5      47       0      47
  3 nominal       5      44       3      47
  4 nominal       8      47       0      47
  5 nominal       5      47       0      47


       Summary on Variables in Y
       *************************

Var    Type   N_Lev   Frequ  N_Miss  N_Nonz
  1 nominal       6      47       0      47
  2 nominal       3      47       0      47
  3 nominal       5      47       0      47
  4 nominal       3      47       0      47
```

The output of the input data is not shown here.

```
Preiteration takes 4 iterations to reach convergence with stress
                        0.583297.

     ************************************
     History of Iterations (Initial Fit=0)
     ************************************

Iteration         Fit      OldFit     FitDiff
        1  0.58329744  1.01000000  0.42670256
        2  0.38845859  0.46613290  0.07767432
        3  0.32115378  0.33290164  0.01174786
```

```
          4  0.28544515  0.29475377  0.00930863
          5  0.25494595  0.26455467  0.00960872
          6  0.23035057  0.23885139  0.00850083
          7  0.21349771  0.21976780  0.00627009
          8  0.20363188  0.20755284  0.00392096
          9  0.19817058  0.20046732  0.00229674
         10  0.19489383  0.19631838  0.00142455
         11  0.19262240  0.19361108  9.887e-004

Stress for complete numerical solution: 0.192622
         Stress for second set: 0.191599


          ******************************
          Category Quantifications for X
          ******************************

Var   Scale     Categories

  1 nominal  6 -0.3703995  7 -1.3873688  7 -1.3410108
            12  0.6195379 11  1.1793939  4  0.2283166
  2 nominal  3 -1.7868654  3 -0.2103452 10 -1.5057049
             6  0.4698922 25  0.7291731
  3 nominal  2 -4.0165578 11 -0.0522989 17  0.2763235
             7  0.8742372  7 -0.4464698
    Missing 28 -0.7077428 41 -0.7436578 45  2.3679319
  4 nominal  3 -1.0205256  5  1.0001798  9  0.1380634
             9 -0.2639056  5 -2.2446592  9  0.6731147
             6  1.0034776  1 -1.6623439
  5 nominal 10 -1.8123309 11 -0.0723741 14  0.6950846
            11  0.7182014  1  1.2880241


          ******************************
          Category Quantifications for Y
          ******************************

Var   Scale     Categories

  1 nominal  3 -0.0999414  2 -2.5356249  3 -2.7647140
             4  0.0211377 22  0.0721680 13  0.9225362
  2 nominal  6 -2.2808089 28  0.6902222 13 -0.4339513
  3 nominal 18 -0.8937251 15  0.2394206  5  1.6820481
             6  1.3897078  3 -1.4175815
  4 nominal 16 -1.2641995 12  0.0041169 19  1.0619889
```

Optimally Scaled X Variables

	GINI	FARM	RENT	GNPR
Yug	-0.370399508	-1.786865450	-4.016557808	-0.263905570
Pol	-0.370399508	-1.786865450	-4.016557808	-2.244659181
Den	-0.370399508	-1.786865450	-0.052298886	0.673114669
Jap	-0.370399508	-0.210345210	-0.052298886	0.138063363
Can	-0.370399508	-0.210345210	-0.052298886	1.003477584
.....				
Ven	0.228316637	0.729173102	0.276323523	0.673114669
Aus	0.228316637	0.729173102	2.367931864	1.003477584
Chi	0.228316637	0.729173102	0.276323523	0.138063363
Bol	0.228316637	0.729173102	0.276323523	-1.020525624

Optimally Scaled X Variables

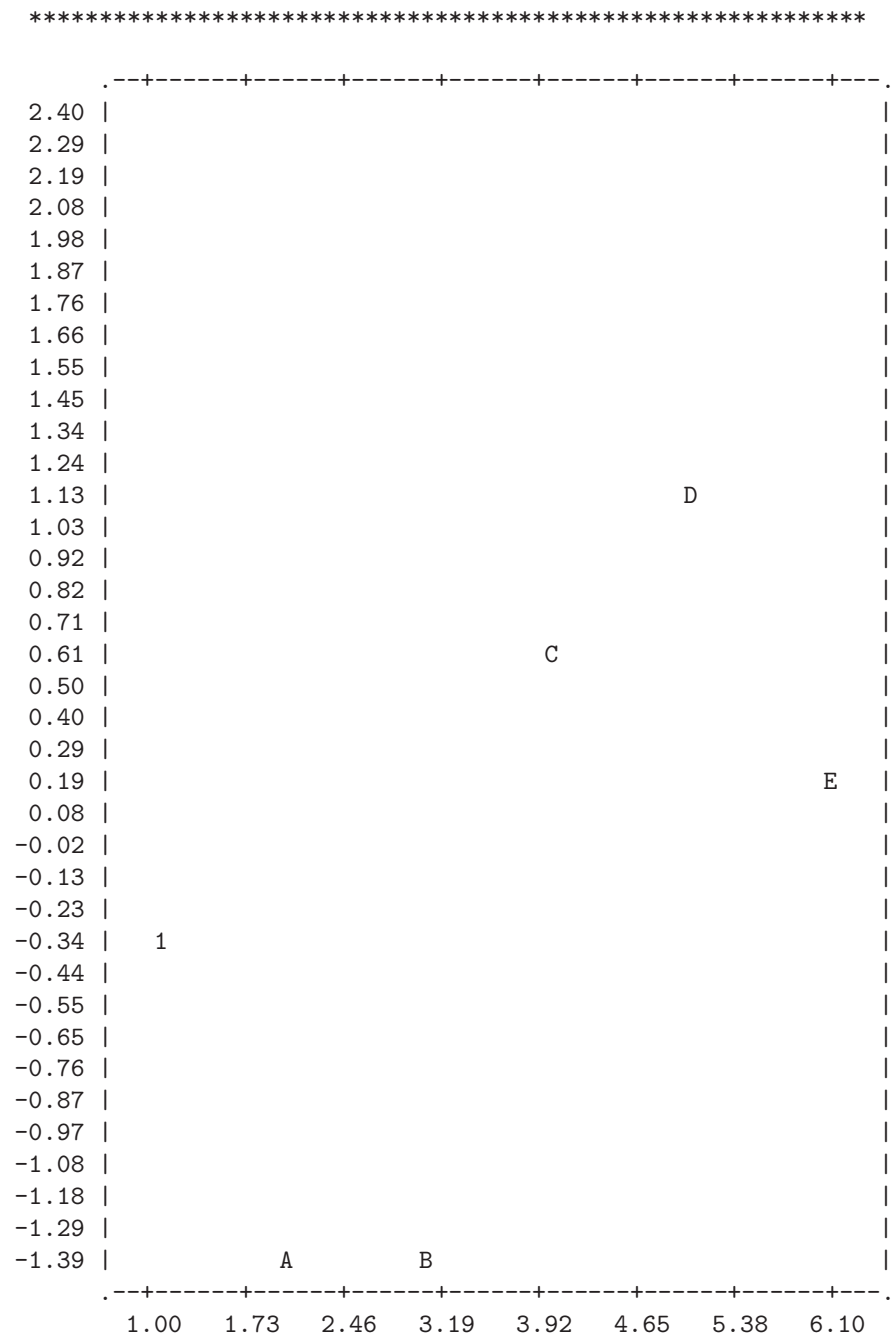
	LABO
Yug	0.718201386
Pol	0.695084641
Den	-0.072374113
Jap	-0.072374113
Can	-1.812330905
.....	
Ven	0.695084641
Aus	-1.812330905
Chi	-0.072374113
Bol	0.718201386

Optimally Scaled Y Variables

	INST	ECKS	DEAT	DEMO
Yug	-0.099941409	0.690222162	-0.893725060	1.061988936
Pol	-2.764714010	0.690222162	0.239420570	1.061988936
Den	0.072167953	-2.280808907	-0.893725060	-1.264199527
Jap	0.922536190	0.690222162	0.239420570	0.004116887
Can	0.021137654	0.690222162	-0.893725060	-1.264199527
.....				
Aus	0.021137654	-2.280808907	-0.893725060	-1.264199527
Chi	0.072167953	0.690222162	0.239420570	0.004116887
Bol	0.922536190	-0.433951315	-1.417581466	1.061988936

We show only the plot w.r.t. the first X Variable:

Category Numbers (hor.) vs. Cat. Quantif. (ver.): Var=1 of X


2.40
2.29
2.19
2.08
1.98
1.87
1.76
1.66
1.55
1.45
1.34
1.24
1.13
1.03
0.92
0.82
0.71
0.61
0.50
0.40
0.29
0.19
0.08
-0.02
-0.13
-0.23
-0.34
-0.44
-0.55
-0.65
-0.76
-0.87
-0.97
-1.08
-1.18
-1.29
-1.39
1
A
B
C
D
E
1.00
1.73
2.46
3.19
3.92
4.65
5.38
6.10
Canonical Correlations for each Dimension

```
          Fac_1       Fac_2
1  0.915975836  0.882774581
```

Variable Loadings of X

```
             Fac_1       Fac_1
GINI   0.815147661 -0.938600683
FARM  -1.184451574  0.500535938
RENT   0.821612228 -0.484183316
GNPR   0.004981056 -0.394209482
LABO  -0.445846455 -0.618604122
```

Variable Loadings of Y

```
             Fac_1       Fac_1
INST   0.296746967 -0.540499646
ECKS  -0.403678711  0.542129581
DEAT   0.269179024 -0.657611829
DEMO  -0.907932148 -0.284632365
```

Correlations between Optimally Scaled Variables of X

and Canonical Variates of X for Each Dimension

```
             Fac_1       Fac_2
GINI  -0.126457348 -0.678940980
FARM  -0.329478793 -0.483252933
RENT   0.532666827 -0.340749177
GNPR   0.193661875 -0.193202610
LABO  -0.615053712 -0.587586814
```

Correlations between Optimally Scaled Variables of Y

and Canonical Variates of X for Each Dimension

```
             Fac_1       Fac_2
INST   0.025190640 -0.526277724
ECKS  -0.510771547  0.119654530
DEAT  -0.094804378 -0.632561360
DEMO  -0.801639957 -0.412740846
```

Correlations between Optimally Scaled Variables of X

and Canonical Variates of Y for Each Dimension

	Fac_1	Fac_2
GINI	-0.116770547	-0.599149019
FARM	-0.304516171	-0.422797472
RENT	0.485706002	-0.297300734
GNPR	0.176630451	-0.165470359
LABO	-0.561935540	-0.521926736

Correlations between Optimally Scaled Variables of Y

and Canonical Variates of Y for Each Dimension

	Fac_1	Fac_2
INST	0.027496406	-0.596149312
ECKS	-0.557620284	0.135534003
DEAT	-0.103497027	-0.716569813
DEMO	-0.875175879	-0.467549537

****
Projections of Category Quantifications on Canonical Variates X
****

Var	Level	Nobs	Fac_1	Fac_2
1	1	6	0.0468397	0.2514794
	2	7	0.1754430	0.9419416
	3	7	0.1695807	0.9104672
	4	12	-0.0783451	-0.4206297
	5	11	-0.1491430	-0.8007388
	6	4	-0.0288723	-0.1550135
2	1	3	0.5887343	0.8635080
	2	3	0.0693043	0.1016499
	3	10	0.4960978	0.7276363
	4	6	-0.1548195	-0.2270768
	5	25	-0.2402471	-0.3523750
3	1	2	-2.1394871	1.3686388
	2	11	-0.0278579	0.0178208
	3	17	0.1471884	-0.0941570
	4	7	0.4656772	-0.2978956
	5	7	-0.2378196	0.1521342
4	1	3	-0.1976369	0.1971682
	2	5	0.1936967	-0.1932373
	3	9	0.0267376	-0.0266742
	4	9	-0.0511084	0.0509872
	5	5	-0.4347049	0.4336740

		6	9	0.1303566	-0.1300475
		7	6	0.1943354	-0.1938745
		8	1	-0.3219326	0.3211692
	5	1	10	1.1146808	1.0649017
		2	11	0.0445140	0.0425261
		3	14	-0.4275144	-0.4084226
		4	11	-0.4417324	-0.4220057
		5	1	-0.7922040	-0.7568260

	1	1	3	-0.0025176	0.0525969
		2	2	-0.0638740	1.3344429
		3	3	-0.0696449	1.4550074
		4	4	5.32e-004	-0.0111243
		5	22	0.0018180	-0.0379804
		6	13	0.0232393	-0.4855102
	2	1	6	1.1649723	-0.2729091
		2	28	-0.3525458	0.0825882
		3	13	0.2216500	-0.0519242
	3	1	18	0.0847290	0.5653359
		2	15	-0.0226981	-0.1514482
		3	5	-0.1594655	-1.0639986
		4	6	-0.1317504	-0.8790754
		5	3	0.1343929	0.8967073
	4	1	16	1.0134329	0.5217868
		2	12	-0.0033003	-0.0016992
		3	19	-0.8513328	-0.4383262

****
Projections of Category Quantifications on Canonical Variates Y
****

Var	Level	Nobs	Fac_1	Fac_2
1	1	6	0.0432518	0.2219245
	2	7	0.1620038	0.8312407
	3	7	0.1565906	0.8034653
	4	12	-0.0723438	-0.3711955
	5	11	-0.1377185	-0.7066327
	6	4	-0.0266607	-0.1367957
2	1	3	0.5441294	0.7554822
	2	3	0.0640535	0.0889334
	3	10	0.4585115	0.6366082
	4	6	-0.1430898	-0.1986693
	5	25	-0.2220450	-0.3082925
3	1	2	-1.9508662	1.1941256

```
        2    11 -0.0254019  0.0155485
        3    17  0.1342120 -0.0821512
        4     7  0.4246223 -0.2599114
        5     7 -0.2168531  0.1327358
4       1     3 -0.1802559  0.1688667
        2     5  0.1766622 -0.1655001
        3     9  0.0243862 -0.0228454
        4     9 -0.0466138  0.0436685
        5     5 -0.3964752  0.3714246
        6     9  0.1188925 -0.1113805
        7     6  0.1772447 -0.1660458
        8     1 -0.2936206  0.2750686
5       1    10  1.0184131  0.9459040
        2    11  0.0406696  0.0377740
        3    14 -0.3905928 -0.3627833
        4    11 -0.4035829 -0.3748485
        5     1 -0.7237865 -0.6722542

---------------------------------------
1       1     3 -0.0027480  0.0595800
        2     2 -0.0697206  1.5116110
        3     3 -0.0760197  1.6481824
        4     4  5.81e-004 -0.0126012
        5    22  0.0019844 -0.0430229
        6    13  0.0253664 -0.5499693
2       1     6  1.2718253 -0.3091272
        2    28 -0.3848819  0.0935486
        3    13  0.2419801 -0.0588152
3       1    18  0.0924979  0.6404164
        2    15 -0.0247793 -0.1715616
        3     5 -0.1740870 -1.2053049
        4     6 -0.1438306 -0.9958226
        5     3  0.1467155  1.0157961
4       1    16  1.1063969  0.5910759
        2    12 -0.0036030 -0.0019248
        3    19 -0.9294271 -0.4965324

Canonical Scores for each Dimension for X

             Fac_1        Fac_1
Yug   -1.807049779  1.057768875
Pol   -1.806609505  1.852900844
Den    1.807176462 -0.741988156
Jap   -0.062800538  0.258039180
Can    0.717263707  0.993229132
```

```
                    ..............................
                    Aus    2.080983007 -0.270298920
                    Chi   -0.417572228  0.007232990
                    Bol   -0.775818508 -0.025093508


                Canonical Scores for each Dimension for Y

                                  Fac_1        Fac_1
                    Yug   -1.513071239  0.713655896
                    Pol   -1.998815391  1.408794574
                    Den    1.849364973 -0.327944459
                    Jap    0.055840965 -0.283058231
                    Can    0.634879894  1.310321229
                    ..............................
                    Aus    1.834221887 -0.300362600
                    Chi   -0.196503230  0.176565500
                    Bol   -0.896860368 -0.103946411


  print "GOF=", gof;
  print "Eval=",eval;
  print "Xload=",xload;
  print "Yload=",yload;
  print "Xscor=",xscor;
  print "Yscor=",yscor;
  print "Xcatq=",xcatq;
  print "Ycatq=",ycatq;


GOF=
              |         1
----------------------
    Failure |   0.00000
       Time |   0.00000
     TotFit |   0.19262
  Dimension |   2.00000
     unused |   0.00000
     unused |   0.00000
     unused |   0.00000
     unused |   0.00000
     unused |   0.00000
     unused |   0.00000


Eval=
   |     Fac_1     Fac_2
```

```
------------------------
 1 |   0.83901   0.77929


Xload=
       |     Fac_1     Fac_2
-----------------------------
  GINI |   0.81515  -0.93860
  FARM |  -1.18445   0.50054
  RENT |   0.82161  -0.48418
  GNPR |   0.00498  -0.39421
  LABO |  -0.44585  -0.61860


Yload=
       |     Fac_1     Fac_2
-----------------------------
  INST |   0.29675  -0.54050
  ECKS |  -0.40368   0.54213
  DEAT |   0.26918  -0.65761
  DEMO |  -0.90793  -0.28463


Xcatq=
           |         1
-----------------------
  GINI_C_1 |  -0.37040
  GINI_C_2 |  -1.38737
  GINI_C_3 |  -1.34101
  GINI_C_4 |   0.61954
  GINI_C_5 |   1.17939
  GINI_C_6 |   0.22832
  FARM_C_1 |  -1.78687
  FARM_C_2 |  -0.21035
  FARM_C_3 |  -1.50570
  FARM_C_4 |   0.46989
  FARM_C_5 |   0.72917
  RENT_C_1 |  -4.01656
  RENT_C_2 |  -0.05230
  RENT_C_3 |   0.27632
  RENT_C_4 |   0.87424
  RENT_C_5 |  -0.44647
  GNPR_C_1 |  -1.02053
  GNPR_C_2 |   1.00018
  GNPR_C_3 |   0.13806
  GNPR_C_4 |  -0.26391
  GNPR_C_5 |  -2.24466
```

```
  GNPR_C_6 |   0.67311
  GNPR_C_7 |   1.00348
  GNPR_C_8 |  -1.66234
  LABO_C_1 |  -1.81233
  LABO_C_2 |  -0.07237
  LABO_C_3 |   0.69508
  LABO_C_4 |   0.71820
  LABO_C_5 |   1.28802


Ycatq=
           |         1
-----------------------
  INST_C_1 |  -0.09994
  INST_C_2 |  -2.53562
  INST_C_3 |  -2.76471
  INST_C_4 |   0.02114
  INST_C_5 |   0.07217
  INST_C_6 |   0.92254
  ECKS_C_1 |  -2.28081
  ECKS_C_2 |   0.69022
  ECKS_C_3 |  -0.43395
  DEAT_C_1 |  -0.89373
  DEAT_C_2 |   0.23942
  DEAT_C_3 |   1.68205
  DEAT_C_4 |   1.38971
  DEAT_C_5 |  -1.41758
  DEMO_C_1 |  -1.26420
  DEMO_C_2 |   0.00412
  DEMO_C_3 |   1.06199
```

3.2 Function `cuclcr`

< gof,ccc,sv > = cuclcr(data,cstr<,optn<,sv<,rsq>>>)

Purpose: For a $N \times n$ data set $\mathbf{X}$ and a given cluster structure (index vector assigning a cluster number to each row/observation of the data $\mathbf{X}$) the `cuclcr` function computes R^2 estimates and computes the cubic cluster criterion (Sarle, 1983). We assume $K < N$ disjoint cluster, and usually $K << N$.

Input: data specifies $N \times n$ data set $\mathbf{X}$

cstr specifies N vector with indices specifying the cluster number $k \in \{1, \ldots, K\}$ of each observation (row of $\mathbf{X}$).

optn options vector

1. int specifying the amount of printed output (def=2)
2. int specifying the version of SVD computation (def=-1)
3. real estimate of R^2 (useful only if $K = 1$) (def=missing)

sv optional input of estimates for the largest nonzero singular values of **X**

rsq optional input of K vector for R^2 estimate of each cluster

Output: **gof** vector of some goodness-of-fit measures

ccc a table containing the cubic cluster criterion, the expected R^2 and the effective number of clusters

sv the estimates of the largest nonzero singular values used for the computation of the CCC

Restrictions: 1. The input data cannot contain missing, string, or complex data.

2. There should be less clusters K than observations N.

Relationships: cluster(), clmix()

Examples: 1. Iris Data: Using standard deviations rather than singular values

```
cstr = [ 1 2 3 2 3 1 2 3 3 1   3 3 2 3 2 3 2 1 3 2
         2 3 2 2 3 1 2 3 3 3   1 3 3 3 2 1 1 3 2 1
         2 1 3 1 2 3 1 3 3 2   1 1 3 1 1 1 2 2 1 1
         1 3 3 1 1 3 3 1 1 3   3 1 1 2 2 3 3 2 1 1
         2 2 3 2 3 3 3 1 1 2   3 1 3 3 3 1 1 3 3 3
         1 1 2 3 2 3 1 1 3 3   2 2 1 3 3 1 3 3 3 3
         3 3 2 2 1 1 2 2 3 3   2 2 2 3 1 1 1 3 1 1
         3 3 3 1 1 1 3 2 2 1 ];

optn = [  3      /* ipri: print */
         -1      /* ivrs: version svd */
          1 ];   /* istd: use std and not svd */
< gof,ccc,sv > = cuclcr(iris2,cstr,optn);
```

Cluster Centroids

	1	2	3
SepalLength	50.06000000	68.50000000	59.01612903
SepalWidth	34.28000000	30.73684211	27.48387097
PetalLength	14.62000000	57.42105263	43.93548387
PetalWidth	2.460000000	20.71052632	14.33870968

Mean, Standard Deviation, R-Square

Variable	Mean	Total_StD	Within_StD	R-Square
SepalLength	58.4333333	8.28066128	4.36528701	0.72209551
SepalWidth	30.5733333	4.35866285	3.22628913	0.45210162
PetalLength	37.5800000	17.6529823	4.18593450	0.94377251
PetalWidth	11.9933333	7.62237669	2.43592090	0.89787183
Over All	34.6450000	10.6922367	3.63732149	0.88427525

Euclidean Distances between Cluster Centroids

	1	2	3
1	0.000000000	50.17568520	33.56934547
2	50.17568520	0.000000000	17.97181799
3	33.56934547	17.97181799	0.000000000

Maximum Distance from Cluster Entry to Centroid

	1	2	3
1	12.48030448	15.29710381	16.60640336

```
************************************************************
Cubic Cluster Criterion for 1 Structures (Total SS=953929)
************************************************************
```

Struct	N_Cluster	Rsquare	Eff_Rsqu	CubClCrit	N_Eff_Var
1	3	0.88427525	0.62727897	25.0210651	2

2. Iris Data: Using singular values

```
optn = [  3      /* ipri: print */
         -1      /* ivrs: version svd */
          0 ];   /* istd: use svd */
< gof,ccc,sv > = cuclcr(iris2,cstr,optn);
```

Cluster Centroids

	1	2	3
SepalLength	50.06000000	68.50000000	59.01612903
SepalWidth	34.28000000	30.73684211	27.48387097
PetalLength	14.62000000	57.42105263	43.93548387
PetalWidth	2.460000000	20.71052632	14.33870968

Mean, Standard Deviation, R-Square

Variable	Mean	Total_StD	Within_StD	R-Square
SepalLength	58.4333333	8.28066128	4.36528701	0.72209551
SepalWidth	30.5733333	4.35866285	3.22628913	0.45210162
PetalLength	37.5800000	17.6529823	4.18593450	0.94377251
PetalWidth	11.9933333	7.62237669	2.43592090	0.89787183
Over All	34.6450000	10.6922367	3.63732149	0.88427525

Euclidean Distances between Cluster Centroids

	1	2	3
1	0.000000000	50.17568520	33.56934547
2	50.17568520	0.000000000	17.97181799
3	33.56934547	17.97181799	0.000000000

Maximum Distance from Cluster Entry to Centroid

	1	2	3
1	12.48030448	15.29710381	16.60640336

Singular Values

1 : 959.6 177.6 34.61 18.85

****

Cubic Cluster Criterion for 1 Structures (Total SS=953929)

****

Struct	N_Cluster	Rsquare	Eff_Rsqu	CubClCrit	N_Eff_Var
1	3	0.88427525	0.86222785	1.80174582	1

3.3 Function demreg

<gof,estim,yxprd> = demreg(data<,opt>)

Purpose: The demreg function computes parameter estimates and variances of parameter estimates of the univariate Deming regression problem. Only the univariate regression with one predictor variable x is implemented as in Reed (1989, 1992). This form of regression is appropriate when both,

the y and x data are measured with uncertainty. Numerical weights for y and x are then assigned "as the inverse squares of the measurement uncertainties" (Reed, 1992).

Input: **data** The data matrix must have inbetween two and four columns and should have $n > 2$ rows (observations). The columns must contain:

1. always: the measurements of the response variable y.
2. always: the measurements of the predictor variable x.
3. optional: the weights of the response variable y.
4. optional: the weights of the predictor variable x.

If weights for the predictor variable are given but no weights for response variable are given the third column should be filled with default weights of 1.

optn The optional second input argument should be a vector specifying some of the following options:

Output: **gof** Is a vector of scalar results containing the sum of squares error of the solution in its forst entry.

estim Is a 2×2 matrix containing the parameter estimates of intercept and slope in its first column and the variances of the corresponding parameters in its second column.

yxprd Is a $n \times 2$ matrix containing the predicted values for the response y (in first column) and the predictor x (in second column).

Restrictions: 1. No missing values are permitted in the input data. Default options are used for missing values in the option vector.

2. Only univariate regression with one predictor variable x is implemented.

Relationships:

Examples: 1. First example by Reed (1989), n=27, with unit weights:

```
data = [  .67   .89 ,  .64  1.00 ,  .76   .92 ,  .61   .87 ,
          .74   .90 ,  .61   .86 ,  .77  1.08 ,  .61   .86 ,
          .99  1.25 ,
          .77  1.01 ,  .73   .86 ,  .64   .85 ,  .62   .88 ,
          .63   .84 ,  .57   .79 ,  .66   .88 ,  .53   .70 ,
          .46   .81 ,
          .79   .88 ,  .77   .92 ,  .70   .92 ,  .88  1.01 ,
          .62   .88 ,  .80   .92 ,  .74   .96 ,  .64   .85 ,
          .93  1.04 ];
```

```
   optn = [   4 ,   /* ipri */
              1 ,   /* minmet */
            100 ];  /* maxfun */
   < gof,estim,yxprd > = demreg(data,optn);


   print "GOF=", gof;
   print "Estimates=", estim;
   print "PredValues=", yxprd;
```

```
                    Algorithm found 2 Real Zeros
                    ****************************

                      N       Slope         SSE

                      0  1.16683674  0.05780470
                      1 -0.85701776  0.59684715


                  Deming Regression with SSE=0.0578047
                  ************************************

     Coefficient    Estimate    AsStdErr     LowerCI     UpperCI

       Intercept -0.36515514  0.15613232 -0.67116887 -0.05914141
           Slope  1.16683674  0.17044460  0.83277147  1.50090200


GOF=
       |         1
-----------------
   SSE |   0.05780
  TIME |   0.00000
       |   0.00000
       |   0.00000


Estimates=
           |   Estimate   AsStdErr    LowerCI    UpperCI
-----------------------------------------------------------
  Intercept |   -0.36516    0.15613   -0.67117   -0.05914
      Slope |    1.16684    0.17044    0.83277    1.50090


PredValues=
   |     Ypred     Xpred
------------------------
```

```
 1 |   0.67141   0.88835
 2 |   0.70847   0.92011
 3 |   0.73812   0.94553
 4 |   0.62694   0.85024
 5 |   0.71671   0.92718
 6 |   0.62199   0.84600
 7 |   0.82294   1.01822
 8 |   0.62199   0.84600
 9 |   1.03378   1.19891
10 |   0.78836   0.98858
11 |   0.69118   0.90530
12 |   0.63435   0.85659
13 |   0.63764   0.85941
14 |   0.62364   0.84742
15 |   0.56435   0.79660
16 |   0.66070   0.87918
17 |   0.49681   0.73872
18 |   0.51081   0.75072
19 |   0.73565   0.94341
20 |   0.74389   0.95047
21 |   0.70353   0.91588
22 |   0.85178   1.04293
23 |   0.63764   0.85941
24 |   0.76118   0.96529
25 |   0.74636   0.95258
26 |   0.63435   0.85659
27 |   0.89543   1.08034
```

```
modl = "1 = 2";
optn = [ "print"      2 ,
         "meth"    "xpx"];
< gof,parm > = reg(data,modl,"l_2",optn);
print "GOF=", gof;
print "Estimates=", parm;
```

```
                ******************************
                Least-Squares (L_2) Regression
                ******************************

SSQ Model            0.247981926    SSQ Total            0.368585185
SSQ Error (SSE)      0.120603259    L2 Norm = sqrt(SSE)  0.347279799
Model DF                       1    Error DF                      25

Root MSE             0.069455960    Res.Err.Var. (MSE)   0.004824130
F Value              51.40448297    Probability>F        1.6309e-007
```

```
Dep. Var.            0.699259259    Coef. Var. (C.V.)    9.932790860
R Squared            0.672794069    Adj. R Squared       0.659705832
Akaike IC (AIC)     -142.0993177    Bayesian IC (BIC)   -139.7921177
Schwarz BC (SBC)    -152.6909914    Mallows CP           2.000000000

 Var        Estimate   AStdErr T_Value    Prob  Low_W_CI  Upp_W_CI

 Intercep -0.1500719  0.119213 -1.2589  0.2197 -0.383725  0.083581
 X1        0.9310573  0.129860  7.1697  0.0000  0.676536  1.185578
```

GOF=

	1
ErrReturn	0.000
Nobs	27.000
Nparm	2.000
Nullity	0.000
EDF	25.000
SS_model	0.248
SS_total	0.369
SS_error	0.121
MSE	0.005
RMSE	0.069
Rsquared	0.673
AdjRsqu	0.660
Fvalue	51.404
Fprob	2e-007
DepMean	0.699
CoefVar	9.933
AIC	-142.099
BIC	-139.792
SBC	-152.691
CPMallow	2.000

2. Second example by Reed (1989), n=10, with uncertainties:

```
print "Example 2 (Reed, 1989) : N=10";
print " col[1]=y, col2=y_uncert col3=x col4=x_uncert";
data = [ -1.367   .297    1.333  2.469 ,
          7.232  4.672   -1.009  6.363 ,
         -0.593  2.014    9.720  6.045 ,
          7.124   .022   -2.079  4.061 ,
          0.468  6.868    8.920  5.325 ,
          8.664  2.834   -0.938  5.865 ,
```

```
          5.854  4.647   10.94   3.993 ,
         13.35   4.728    5.138  3.787 ,
          4.279  2.274   11.37   3.693 ,
         11.63   4.659    9.421  4.687 ];
/* weights are inverse squares of uncertainties */
ywgt = 1. / (data[,2] .* data[,2]);
xwgt = 1. / (data[,4] .* data[,4]);
dat2 = data[,1] -> data[,3] -> ywgt -> xwgt;


optn = [   4 ,   /* ipri */
           1 ,   /* minmet */
         100 ];  /* maxfun */
< gof,estim,yxprd > = demreg(dat2,optn);
print "GOF=", gof;
print "Estimates=", estim;
print "PredValues=", yxprd;
```

Algorithm found 2 Real Zeros

N	Slope	SSE
0	0.00165927	833.514121
1	4.54365682	13.9556261

Deming Regression with SSE=13.9556

Coefficient	Estimate	AsStdErr	LowerCI	UpperCI
Intercept	-17.4835233	233.016318	-474.187115	439.220068
Slope	4.54365682	48.2885088	-90.1000813	99.1873949

PredValues=

	Ypred	Xpred
1	-1.3740	3.5455
2	6.4863	5.2755
3	-0.4471	3.7495
4	7.1240	5.4158
5	2.1516	4.3214
6	8.3239	5.6799
7	7.4775	5.4936

```
 8 |   12.8243    6.6704
 9 |    4.8182    4.9083
10 |   12.2554    6.5452
```

3. Example by Reed (1992), n=10, with specified weights:

```
data = [ 5.9  0.0    1.0  1000. ,
         5.4  0.9    1.8  1000. ,
         4.4  1.8    4.0   500. ,
         4.6  2.6    8.0   800. ,
         3.5  3.3   20.0   200. ,
         3.7  4.4   20.0    80. ,
         2.8  5.2   70.0    60. ,
         2.8  6.1   70.0    20. ,
         2.4  6.5  100.0     1.8 ,
         1.5  7.4  500.0     1.0 ];


optn = [   4 ,   /* ipri */
           1 ,   /* minmet */
         100 ];  /* maxfun */
< gof,estim,yxprd > = demreg(data,optn);
print "GOF=", gof;
print "Estimates=", estim;
print "PredValues=", yxprd;


                  Algorithm found 3 Real Zeros
                  ****************************

                    N       Slope         SSE

                    0 -0.48053341  11.8663532
                    1  0.24878711  231.099889
                    2  0.01045388  453.491111


                Deming Regression with SSE=11.8664
                **********************************

     Coefficient    Estimate    AsStdErr     LowerCI     UpperCI

       Intercept  5.47991022  0.35555715  4.78303102  6.17678943
           Slope -0.48053341  0.07017169 -0.61806740 -0.34299942


PredValues=
```

```
   |     Ypred     Xpred
-----------------------
 1 |   5.48001  -0.00020
 2 |   5.04758   0.89970
 3 |   4.61455   1.80082
 4 |   4.23137   2.59823
 5 |   3.88525   3.31851
 6 |   3.38382   4.36202
 7 |   2.94269   5.28000
 8 |   2.66100   5.86622
 9 |   2.39685   6.41591
10 |   1.50364   8.27470
```

3.4 Function `dixonr`

r = dixonr(sopt,i,j,n,r|R|α <,opt>)

Purpose: The `dixonr` function computes pdf, cdf, and critical values for Dixon's r,

$$ret(i,j,n) = \frac{x_n - x_{n-j}}{x_n - x_i} \quad forn \geq i + j + 1$$

where $x_1 \leq x_2 \leq \ldots \leq x_{n-1} \leq x_n$ are the ordered (onedimensional) observations. Usually i and j are small integer values.

Input: sopt This is a string option, which can have the following values:

"pdf" specifies computation of pdf $P(r)$ values for all (i, j, n, r) tuples, for one or more r values with $0 \leq r \leq 1$.

"cdf" specifies computation of cdf values $G(R)$ for all (i, j, n, R)] tuples where

$$G(R) = \int_0^R P(r), \quad 0 \leq r \leq 1, 0 < R < 1.$$

Here is $G(0) = 0$ and $G(1) = 1$ and $G(R)$ increases monotonically from zero to one.

"pcdf" specifies the computation of both, pdf and cdf, and returns a two column matrix.

"crit" specifies the computation of critical values which are the roots R of of the cdf for

$$(1. - \alpha) - G(R) = 0$$

for given value(s) of α. Since $G(R)$ increases monotonically from zero to one there is exactly one critical value R for each value of α.

i is the index of the "small" test value.

j is the index of the "large" test value.

n is the number of data values.

r|R|alpha This input depends on the specification of the `"sopt"` input argument:

"pdf" fourth input argument is Dixon's r for $P(r)$

"cdf" fourth input argument is upper limit R for integral $G(R)$

"pcdf" fourth input argument is Dixon's r for $P(r)$ (first column of output) and upper limit R for integral $G(R)$ (second column of output)

"crit" fourth input argument is α for critical values

opt This optional input argument is a vector specifying the following:

1 specifying printed output (=0: no printed output).

2 specifying the optimization method that is used for critical values:

- opt[2]=0: original method
- opt[2]=1: different version of Brent's method
- opt[2]=2: different method

3 termination criterion: max function calls (def=200)

4 termination criterion: relative precision (def=1.e-6)

5 termination criterion: absolute precision (def=1.e-8)

Note: The input arguments $(i, j, n, r|R|\alpha)$ must be either scalars or vectors with the same number of entries.

Output: The return argument *ret* is either a m vector or a $m \times 2$ matrix depending on the specification of the `"sopt"` input argument:

"pdf" returns $P(r)$ values of pdf for all (i, j, n, r)

"cdf" returns $G(R)$ values of cdf for all (i, j, n, R)

"pcdf" returns both returns both $P(r)$ and $G(R)$ values for all $(i, j, n, r|R)$

"crit" returns critical values R which are roots of $(1. - \alpha) - G(R) = 0$ for all given α

Restrictions:

1. There must be $n \geq i + j + 1$.
2. Missing values for input arguments are not permitted.
3. Complex data for input are not permitted.
4. String data for all arguments except the first are not permitted.

Relationships:

Examples:

1. Example 1 from JSS paper submission:

```
/* Test 1: Table (r P(r) G(R)) */
j = 1; i = 2; n = 7;
np = 50;
npm = np - 1.; dp = 1. / npm;
prob = [ 0. : dp : 1. ];
r = dixonr("pcdf",i,j,n,prob);
r2 = prob`; r2 = r2 -> r;
print "Test 1:", r2;
```

PDF and CDF Values
****

I=2 J=1 N=7

Number	R|r	P(r)	G(R)
1	0.000000000	2.184490534	0.000000000
2	0.020408163	2.179629832	0.044537528
3	0.040816327	2.171321982	0.088941032
4	0.061224490	2.159364541	0.133138086
5	0.081632653	2.143562830	0.177052206
6	0.102040816	2.123733414	0.220603044
7	0.122448980	2.099707964	0.263706654
8	0.142857143	2.071337486	0.306275846
9	0.163265306	2.038496861	0.348220625
10	0.183673469	2.001089659	0.389448727
11	0.204081633	1.959053140	0.429866253
12	0.224489796	1.912363336	0.469378414
13	0.244897959	1.861040090	0.507890369
14	0.265306122	1.805151892	0.545308168
15	0.285714286	1.744820335	0.581539798
16	0.306122449	1.680223982	0.616496299
17	0.326530612	1.611601424	0.650092965
18	0.346938776	1.539253298	0.682250591
19	0.367346939	1.463543023	0.712896745
20	0.387755102	1.384896047	0.741967049
21	0.408163265	1.303797383	0.769406423
22	0.428571429	1.220787302	0.795170255
23	0.448979592	1.136455078	0.819225473
24	0.469387755	1.051430753	0.841551462
25	0.489795918	0.966375010	0.862140796
26	0.510204082	0.881967301	0.880999747
27	0.530612245	0.798892511	0.898148531
28	0.551020408	0.717826530	0.913621275

```
29 0.571428571 0.639421197 0.927465681
30 0.591836735 0.564289159 0.939742379
31 0.612244898 0.492989225 0.950523976
32 0.632653061 0.426012830 0.959893818
33 0.653061224 0.363772184 0.967944491
34 0.673469388 0.306590632 0.974776100
35 0.693877551 0.254695662 0.980494380
36 0.714285714 0.208214861 0.985208695
37 0.734693878 0.167174980 0.989029981
38 0.755102041 0.131504132 0.992068709
39 0.775510204 0.101036953 0.994432916
40 0.795918367 0.075522445 0.996226375
41 0.816326531 0.054634101 0.997546949
42 0.836734694 0.037981785 0.998485166
43 0.857142857 0.025124840 0.999123053
44 0.877551020 0.015585833 0.999533246
45 0.897959184 0.008864369 0.999778379
46 0.918367347 0.004450483 0.999910754
47 0.938775510 0.001837145 0.999972277
48 0.959183673 0.000531523 0.999994630
49 0.979591837 6.4747e-005 0.999999671
50 1.000000000-2.4938e-045 0.999999999

   Time for Computing Table: 3
```

2. Example 2 from JSS paper submission:

```
/* Test 2: some critical values: orig min */
j = 1; i = 2; n = [ 4:15 ];
alpha = .1;
opt = [ 2,    /* ipri*/
        0,    /* minmet */
      200 ];  /* maxf */
r = dixonr("crit",i,j,n,alpha,opt);
r2 = (real)n'; r2 = r2 -> r;
print "Test 1:", r2;
```

```
 Critical Values R for alpha=0.1
 *******************************

    I=2 J=1 alpha=       0.1

Number       N    Nfun           R

     1       4       6 0.910476664
```

```
         2       5      10 0.727990676
         3       6      10 0.609709149
         4       7      11 0.532943474
         5       8      10 0.479547988
         6       9       9 0.440202619
         7      10      10 0.409904488
         8      11      11 0.385770039
         9      12      11 0.366026632
        10      13      11 0.349527303
        11      14      11 0.335496399
        12      15      11 0.323391031

    Total  Number of Function Calls: 121
         Time for Computing Table: 3
```

```
/* Test 2: some critical values: Kahane-Moler-Nash */
j = 1; i = 2; n = [ 4:15 ];
alpha = .1;
opt = [ 2,    /* ipri*/
        1,    /* minmet */
      200 ];  /* maxf */
r = dixonr("crit",i,j,n,alpha,opt);
r2 = (real)n'; r2 = r2 -> r;
print "Test 1:", r2;
```

```
       Critical Values R for alpha=0.1
       *******************************

          I=2 J=1 alpha=       0.1

      Number       N    Nfun           R

           1       4       6 0.910476554
           2       5      10 0.727990826
           3       6      10 0.609708990
           4       7       9 0.532943488
           5       8       7 0.479548035
           6       9       8 0.440202539
           7      10       9 0.409904745
           8      11      10 0.385770037
           9      12      10 0.366026770
          10      13      11 0.349527462
          11      14      11 0.335496673
          12      15      11 0.323391045
```

```
            Total  Number of Function Calls: 112
                 Time for Computing Table: 5
```

```
/* Test 2: some critical values: ratio zer */
j = 1; i = 2; n = [ 4:15 ];
alpha = .1;
opt = [ 2,    /* ipri*/
        2,    /* minmet */
      200 ];  /* maxf */
r = dixonr("crit",i,j,n,alpha,opt);
r2 = (real)n'; r2 = r2 -> r;
print "Test 1:", r2;
```

```
              Critical Values R for alpha=0.1
              *******************************

                 I=2 J=1 alpha=       0.1

             Number       N    Nfun           R

                  1       4       8 0.910476554
                  2       5      11 0.727990765
                  3       6      11 0.609708993
                  4       7      11 0.532943474
                  5       8      10 0.479547987
                  6       9      10 0.440202611
                  7      10      11 0.409904738
                  8      11      12 0.385770040
                  9      12      12 0.366026638
                 10      13      12 0.349527353
                 11      14      13 0.335496640
                 12      15      13 0.323391030

            Total  Number of Function Calls: 134
                 Time for Computing Table: 3
```

3. Example 4 from JSS paper submission:

```
/* Test 4: Table of all critical values by Dixon, 1951:
           for 14 alpha values */
ni = 3; nj = 2; nn = 28;
iv = [ 1:3 ]; jv = [ 1 2 ]; nv = [ 3:30 ];
alpha =  [ 0.005, 0.01, 0.02, 0.05, 0.10, 0.20, 0.30, 0.40,
           0.50, 0.60, 0.70, 0.80, 0.90, 0.95 ];
```

```
np = nrow(alpha);
optn = 0;

for (j = 1; j <= nj; j++) {
  jj = jv[j];
  for (i = 1; i <= ni; i++) {
    ii = iv[i];
    nmin = ii + jj + 1;
    nmax = 30;
    tab = cons(nn,np);
    for (n = nmin; n <= nmax; n++) {
      r = dixonr("crit",ii,jj,n,alpha,optn);
      tab[n-2,] = r;
    }
    print "Table [j=",j, ", i=",i,"]:", tab;
} }
```

We only show the first table here:

```
Table [j= 1, i= 1]:
```

	1	2	3	4	5
1	0.99397	0.99397	0.99397	0.99397	0.99397
2	0.92066	0.92066	0.92066	0.92066	0.92066
3	0.82320	0.82320	0.82320	0.82320	0.82320
4	0.74270	0.74270	0.74270	0.74270	0.74270
5	0.68108	0.68108	0.68108	0.68108	0.68108
6	0.63363	0.63363	0.63363	0.63363	0.63363
7	0.59627	0.59627	0.59627	0.59627	0.59627
8	0.56613	0.56613	0.56613	0.56613	0.56613
9	0.54129	0.54129	0.54129	0.54129	0.54129
10	0.52043	0.52043	0.52043	0.52043	0.52043
11	0.50262	0.50262	0.50262	0.50262	0.50262
12	0.48721	0.48721	0.48721	0.48721	0.48721
13	0.47372	0.47372	0.47372	0.47372	0.47372
14	0.46179	0.46179	0.46179	0.46179	0.46179
15	0.45115	0.45115	0.45115	0.45115	0.45115
16	0.44158	0.44158	0.44158	0.44158	0.44158
17	0.43291	0.43291	0.43291	0.43291	0.43291
18	0.42502	0.42502	0.42502	0.42502	0.42502
19	0.41780	0.41780	0.41780	0.41780	0.41780
20	0.41115	0.41115	0.41115	0.41115	0.41115
21	0.40501	0.40501	0.40501	0.40501	0.40501
22	0.39931	0.39931	0.39931	0.39931	0.39931
23	0.39400	0.39400	0.39400	0.39400	0.39400

24	0.38904	0.38904	0.38904	0.38904	0.38904
25	0.38440	0.38440	0.38440	0.38440	0.38440
26	0.38003	0.38003	0.38003	0.38003	0.38003
27	0.37592	0.37592	0.37592	0.37592	0.37592
28	0.37204	0.37204	0.37204	0.37204	0.37204
29	0.34236	0.34236	0.34236	0.34236	0.34236
30	0.30939	0.30939	0.30939	0.30939	0.30939
31	0.25945	0.25945	0.25945	0.25945	0.25945
32	0.21544	0.21544	0.21544	0.21544	0.21544
33	0.16422	0.16422	0.16422	0.16422	0.16422
34	0.12995	0.12995	0.12995	0.12995	0.12995
35	0.10322	0.10322	0.10322	0.10322	0.10322
36	0.08087	0.08087	0.08087	0.08087	0.08087
37	0.06142	0.06142	0.06142	0.06142	0.06142
38	0.04404	0.04404	0.04404	0.04404	0.04404
39	0.02821	0.02821	0.02821	0.02821	0.02821
40	0.01361	0.01361	0.01361	0.01361	0.01361
41	0.00670	0.00670	0.00670	0.00670	0.00670

3.5 Function `fica`

<gof,Aica,Wica,Scor,Eval,Evec> = fica(data,optn<,Eini<,Vini<,Agues>>>)

Purpose: The `fica` function implements the original algorithm of (Fast) Independent Component Analysis which is a nonlinear extension of principal component analysis. The algorithm was modified in a way to analyze large sparse data matrices, i.e. no dense copy of the data is needed. The algorithm is a two step method:

1. The first step computes a k factor principal component analysis of the m *timesn* (centered) data matrix $\mathbf{Z}$,

$$\mathbf{Z} = \mathbf{U}\mathbf{V}^{\mathbf{T}} \quad \text{, where} \quad \mathbf{U} \in \mathcal{R}^{m \times k}, \mathbf{V} \in \mathcal{R}^{n \times k}$$

2. The second step is an iterative fix point algorithm imposing a nonlinear function on the relationship of matrices $\mathbf{U}$ and $\mathbf{V}$.

Since the first step, PCA, can be a significant part of the computational cost, the `fica()` function permits the input of eigenvalues and eigenvectors from a preliminary PCA of the centered data $\mathbf{Z}$. There are two versions of the algorithm:

1. The *symmetric* or *parallel* approach: The fit is expressed taking all k components into account.

2. The *deflation* approach: estimates independent components one-by-one like in projection pursuit. The results are usually inferior to those of the parallel approach which uses all components in the fit.

There is no guarantee that either of the two approaches converges, especially not for a step size 1 as the example shows. Convergence properties should be better as more precise the model, nonlinear kernel function, fits the data. The `fica()` function is closely related to the `kpca()` function, which is another nonlinear modification of principal component analysis. Note: different from the original Matlab and R implementations of this method, CMAT only computes the k *truncated* evd or svd in the first step of the algorithm. This is of course no loss of generalizity, since additional components computed by the PCA do not have any influence on the ICA components.

Input: data This should be a $m \times n$, $m > n$, dense or sparse data matrix $\mathbf{Z}$ which cannot contain string or complex data or missing values.

optn This argument must be specified in form of a two column matrix where the first column defines the option as string value (in quotes) and the second column can be used for a numeric or string specification of the option. See table below for content.

Eini If the principal component analysis of $\mathbf{Z}$ is known from a preliminary analysis, this is a k vector of the largest eigenvalues of $\mathbf{Z}$.

Vini If the principal component analysis of $\mathbf{Z}$ is known from a preliminary analysis, this is a $n \times k$ matrix $\mathbf{V}$ of the k eigenvectors corresponding to the largest eigenvalues of $\mathbf{Z}$.

Agues This argument specifies a starting value for the mixing matrix **Aica**. Commonly this should be the result of a former analysis of the the same or similar data matrix $\mathbf{Z}$. If this argument is not specified a random generated normal matrix is used as start for mixing matrix $\mathbf{A}$.

Option	Second Column	Meaning
`"appr"`	string	compute variance matrices
	"par"	use parallel version of algorithm
	"def"	use deflation version of algorithm
`"finet"`	string	nonlinear kernel function
	"skew"	second order polynomial
	"pow3"	third order polynomial
	"gaus"	exponential function (also in R)
	"tanh"	sigmoid function (logcosh in R)
`"kern"`	string	nonlinear kernel function
	"skew"	second order polynomial
	"pow3"	third order polynomial
	"gaus"	exponential function (also in R)
	"tanh"	sigmoid function (logcosh in R)
`"kfp1"`	real	kernel parameter for `"tanh"`
`"kfp2"`	real	kernel parameter for `"gaus"`
`"maxit"`	int	maximum number of iterations
`"maxtun"`	int	maximum number of tuning steps
`"norm"`		standardize data $\mathbf{Z}$ (centering is default)
`"nfac"`		number of components k (factors, dimension)
`"orig"`		use the original version of the algorithm
`"phis"`		print iteration history
`"print"`	int	amount of printed output (default=2)

Option	Second Column	Meaning
"sampsiz"	real	if smaller than one and larger than zero then use random sample (default=1, i.e. full data)
"seed"	int	seed for random generator
"stabon"		stabilization on: step size can be changed during iteration (default)
"stabof"		stabilization off: algorithm works with step size 1
"stan"		standardize data **Z** (centering is default)
"step"	real	stepsize should be in $(0., 1]$ (default=1)
"tolit"	real	termination criterion for iteration (default=1.e-5)
"vers"	string	version of eigen value or singular value method for pca
	"cev1"	eigen values of $n \times n$ cov matrix: dense, all n values and vectors
	"cev2"	eigen values of $n \times n$ cov matrix: dense, all values but only k vectors
	"cev3"	eigen values of $n \times n$ cov matrix: dense Lapack version
	"cev4"	eigen values of $n \times n$ cov matrix: sparse Arpack version
	"rsv1"	singular values of $m \times n$ raw matrix: dense, all values and vectors
	"rsv2"	singular values of $m \times n$ raw matrix: sparse Arpack version
	"rsv3"	singular values of $m \times n$ raw matrix: sparse Block Lanczos
	"rsv4"	singular values of $m \times n$ raw matrix: sparse subspace iteration (Rutishauser)

Output: The function returns at most five arguments:

gof this is a vector of scalar results.

Aica this is the $n \times k$ *mixing* matrix **A** of ICA.

Wica this is the $n \times k$ *separating* matrix **W** of ICA.

Scor this is the $m \times k$ *independent component* matrix of the ICA.

Eval This is a k vector of the largest eigenvalues of **Z**.

Evec This is a $n \times k$ matrix **V** of the k eigenvectors corresponding to the largest eigenvalues of **Z**.

Restrictions: 1. Missing values for **Z** are not permitted.

2. String or complex data for **Z** are not permitted.

Relationships: pca(), kpca(), svd(), nnmf()

Examples: The first 30 observations of the Boston Housing Data, excluding column 4 (which is constant zero for the selected observations) is ued as an illustration. We specify the number of components with $k = 3$.

```
fid = fopen("..\\tdata\\housing.dat","r");
form = "%g %g %g %g %g %g %g %g %g %g %g %g %g %g";
hous = fscanf(fid,form,506,14);
cind = [ 1:3 5:14 ];
data = hous[1:30,cind ];
vna0 = [ "crim" "zn" "indus" "chas" "nox" "rm" "age"
         "dis" "rad" "tax" "ptrat" "b" "lstat" "medv" ];
vnam = [ "crim" "zn" "indus" "nox" "rm" "age"
         "dis" "rad" "tax" "ptrat" "b" "lstat" "medv" ];
data = cname(data,vnam);
```

For this example, the common algorithm for the *deflation* approach converges for the `"skew"`, `"gaus"`, and `"tanh"` functions, but fails to converge (at the second component) for the `"pow3"` function with step size 1. Convergence is obtained, however, when the step size is reduced to .5. The common algorithm for the *parallel* approach also converges for the `"skew"`, `"gaus"`, and `"tanh"` functions with default step size 1. For `"pow3"` the step size was reduced to .5 to obtain convergence.

1. Deflation approach : Second order polynomial `"skew"`

```
optn = [ "print"           3 ,
         "orig"              ,
         "nfac"            3 ,
         "seed"          123 ,
         "appr"        "def" ,
         "finet"       "off" ,
         "staboff"           ,
         "step"           1. ,
         "kern"       "skew" ,
         "tolit"       1.e-4 ,
         "maxit"         100 ];
< gof,aica,wica,scor,eval,evec > = fica(data,optn);


             ************************************
             (Fast) Independent Component Analysis
             ************************************

  Number Signals (Observations) . . . . . .        30
  Data Dimensionality (Number Variables). . . . .  13
  Number Components (Factors) . . . . . .          3
```

```
Algorithmic Approach. . . . . . . . . . . . . . . . Deflation
(Nonlinear) Kernel Function . . . . . . . Polynom 2nd D (skew)
Maximum Number Iterations . . . . . . . . . . . . .        100
Iteration Tolerance (Precision) . . . . . . . . . .     0.0001
(Starting) Step Size. . . . . . . . . . . . . . . .     1.0000
```

Eigenvalues of Covariance Matrix

	1
Fac_1	298.9788208
Fac_2	738.4512646
Fac_3	1082.087139

Eigenvectors from PCA

	crim	zn	indus	nox
Fac_1	0.001229976	-0.076004462	-0.008719439	-0.000384571
Fac_2	-0.000783432	0.078012878	0.015949698	0.000242652
Fac_3	-0.007262904	-0.022803679	-0.046025946	-0.000745953

Eigenvectors from PCA

	rm	age	dis	rad
Fac_1	0.004772388	0.867108437	0.003661115	-0.007957442
Fac_2	4.0445e-006	0.360244165	0.004363819	0.009304303
Fac_3	0.010048966	-0.278456839	0.010490921	-0.016671057

Eigenvectors from PCA

	tax	ptrat	b	lstat
Fac_1	-0.433709511	0.014406183	-0.191961088	0.119033488
Fac_2	0.350545506	-0.024633434	0.856487838	0.068692754
Fac_3	-0.812379842	-0.014874382	0.470053483	-0.123756051

Eigenvectors from PCA

	medv
Fac_1	-0.053287718
Fac_2	-0.044703300
Fac_3	0.151106020

****

```
Deflation (Projection Pursuit) Iteration for Dimension 1
*********************************************************

          Iteration      D_minus       D_plus
                  1  1.000000000  1.000000000
                  2  1.493909576  1.329749668
                  3  0.490944424  1.938807255
                  4  0.279717325  1.980342955
                  5  0.260585372  1.982951150
                  6  0.168285154  1.992907451
                  7  0.069157590  1.998803949
                  8  0.021021828  1.999889518
                  9  0.005603145  1.999992151
                 10  0.001435044  1.999999485
                 11  0.000363513  1.999999967
                 12  9.1837e-005  1.999999998

Iteration for Dimension 1 Converged after 12 Iterations


*********************************************************
Deflation (Projection Pursuit) Iteration for Dimension 2
*********************************************************

          Iteration      D_minus       D_plus
                  1  1.000000000  1.000000000
                  2  0.324322780  1.973528499
                  3  0.070294886  1.998764276
                  4  0.025682933  1.999835090
                  5  0.008693955  1.999981104
                  6  0.003030223  1.999997704
                  7  0.001045967  1.999999726
                  8  0.000362276  1.999999967
                  9  0.000125329  1.999999996
                 10  4.3375e-005  2.000000000

Iteration for Dimension 2 Converged after 10 Iterations


*********************************************************
Deflation (Projection Pursuit) Iteration for Dimension 3
*********************************************************

          Iteration      D_minus       D_plus
                  1  1.000000000  1.000000000
                  2  1.6653e-016  2.000000000
```

Iteration for Dimension 3 Converged after 2 Iterations

Mixing Matrix A

	crim	zn	indus	nox
Fac_1	0.133863641	-0.188731148	-0.066681915	-1.535891407
Fac_2	0.302469235	-0.011910701	-3.775150762	-8.569211494
Fac_3	0.399702955	0.828689739	-0.048299199	-0.245958372

Mixing Matrix A

	rm	age	dis	rad
Fac_1	-1.939505599	0.814478340	0.341945531	-1.527803475
Fac_2	-17.80023624	-0.267778071	0.255744999	-0.007279223
Fac_3	-27.92112529	1.687603368	3.061705264	0.372831783

Mixing Matrix A

	tax	ptrat	b	lstat
Fac_1	-0.227502955	0.005802884	-0.025611485	0.000446330
Fac_2	0.037491511	-0.618014998	-0.015790806	4.256093774
Fac_3	-3.816083941	-3.097098443	-1.344534541	4.567635510

Mixing Matrix A

	medv
Fac_1	-0.156368093
Fac_2	-29.03175498
Fac_3	2.088287433

Separating Matrix W

	crim	zn	indus	nox
Fac_1	0.000133168	-0.002307278	0.000142403	2.4287e-006
Fac_2	-0.000157393	-0.003002800	-0.001588287	-2.8978e-005
Fac_3	-0.000110150	0.003701779	0.000116372	1.5589e-005

Separating Matrix W

	rm	age	dis	rad
Fac_1	-0.000139000	-0.006172870	-0.000287933	-6.9962e-005
Fac_2	0.000329250	-0.000811284	0.000251118	-0.000703952

```
Fac_3  -0.000204385 -0.052186697 -0.000162371  0.000292266
```

Separating Matrix W

```
              tax        ptrat            b        lstat
Fac_1  -0.000163658  0.001035038 -0.034804233 -0.000221101
Fac_2  -0.033229348  0.000237571 -0.005032225 -0.002653979
Fac_3   0.017356266 -0.000769808  0.009174562 -0.007800156
```

Separating Matrix W

```
                medv
Fac_1  -0.000846325
Fac_2   0.003905271
Fac_3   0.004163807
```

Independent Component Matrix

```
              Fac_1        Fac_2        Fac_3
Row_01  -14.31365208 -11.85717625  5.490767018
Row_02  -14.34377350 -10.04777087  3.728304231
Row_03  -14.10230615 -9.947876376  4.714135583
Row_04  -14.06601856 -9.274629337  5.183960349
Row_05  -14.19978987 -9.288232190  4.759399961
Row_06  -14.12434364 -9.307094351  4.468912884
Row_07  -14.26745598 -12.36926202  5.584831719
Row_08  -14.49996812 -12.40102758  4.022219707
Row_09  -14.15996947 -12.42267204  3.596328676
Row_10  -14.07511452 -12.36795488  4.442809495
Row_11  -14.32659305 -12.42807572  4.015333541
Row_12  -14.41030680 -12.40671650  4.722797088
Row_13  -13.91924322 -12.33466645  6.947855405
Row_14  -14.24409973 -12.19882093  5.749040699
Row_15  -13.79525717 -12.14620722  4.384784246
Row_16  -14.16638160 -12.19067980  6.010224472
Row_17  -13.69549440 -12.10700130  7.377240715
Row_18  -14.01249567 -12.19234944  4.555377381
Row_19  -10.33305375 -11.64565167  6.046760858
Row_20  -14.08304614 -12.19204976  5.260080497
Row_21  -13.75727414 -12.18683721  3.540457691
Row_22  -14.26146895 -12.21717049  4.232339754
Row_23  -14.42631656 -12.27136114  4.085429958
Row_24  -14.39509879 -12.27207134  3.618740455
Row_25  -14.35164341 -12.25228036  3.957174056
```

```
Row_26  -11.13430573 -11.79528986  3.552814269
Row_27  -13.72144650 -12.15347886  4.011158825
Row_28  -11.25744041 -11.81109994  3.415829057
Row_29  -14.13278485 -12.19994597  3.921723626
Row_30  -13.82256851 -12.14308882  4.238708497
```

2. Deflation approach : Exponential function "gaus"

```
optn = [ "print"           3 ,
         "orig"              ,
         "nfac"            3 ,
         "seed"          123 ,
         "appr"            0 ,
         "finet"       "off" ,
         "staboff"           ,
         "step"           1. ,
         "kern"       "gaus" ,
         "kfp1"           1. ,
         "kfp2"           1. ,
         "tolit"       1.e-4 ,
         "maxit"         100 ];
< gof,aica,wica,scor,eval,evec > = fica(data,optn);


             ************************************
             (Fast) Independent Component Analysis
             ************************************

  Number Signals (Observations) . . . . . .          30
  Data Dimensionality (Number Variables). . . . .    13
  Number Components (Factors) . . . . . .            3
  Algorithmic Approach. . . . . .  Deflation
  (Nonlinear) Kernel Function . . . . .  Radial Base (gauss)
  Function Parameter. . . . . .      1.0000
  Maximum Number Iterations . . . . . .        100
  Iteration Tolerance (Precision) . . . . . .      0.0001
  (Starting) Step Size. . . . . .      1.0000


    ********************************************************
    Deflation (Projection Pursuit) Iteration for Dimension 1
    ********************************************************

              Iteration      D_minus       D_plus
                     1  1.000000000  1.000000000
                     2  1.684503927  1.078168132
```

```
         3  0.406525395  1.958248478
         4  0.465632844  1.945041402
         5  0.578109104  1.914625254
         6  0.853029618  1.808961158
         7  1.483625954  1.341213641
         8  1.994912080  0.142568549
         9  1.991874539  0.180099478
        10  1.986730009  0.230008415
        11  1.986115952  0.235251831
        12  1.998587498  0.075153251
        13  1.999963203  0.012132108
        14  1.999997784  0.002977092
        15  1.999999879  0.000696416
        16  1.999999993  0.000164911
        17  2.000000000  3.8941e-005

Iteration for Dimension 1 Converged after 17 Iterations


*********************************************************
Deflation (Projection Pursuit) Iteration for Dimension 2
*********************************************************

   Iteration      D_minus       D_plus
           1  1.000000000  1.000000000
           2  1.985323935  0.241844728
           3  1.999407862  0.048664152
           4  1.999818906  0.026913614
           5  1.999952947  0.013718909
           6  1.999986509  0.007346112
           7  1.999996311  0.003841323
           8  1.999998965  0.002035048
           9  1.999999713  0.001070893
          10  1.999999920  0.000565559
          11  1.999999978  0.000298120
          12  1.999999994  0.000157303
          13  1.999999998  8.2958e-005

Iteration for Dimension 2 Converged after 13 Iterations


*********************************************************
Deflation (Projection Pursuit) Iteration for Dimension 3
*********************************************************

   Iteration      D_minus       D_plus
           1  1.000000000  1.000000000
```

```
                    2  2.000000000  1.4499e-014

 Iteration for Dimension 3 Converged after 2 Iterations


                        Mixing Matrix A

              crim           zn        indus          nox
Fac_1  -0.134552197 -0.190003408  0.061480157  1.594608644
Fac_2   0.304388271  0.017443744  3.052212925 -8.435760833
Fac_3  -0.436908777 -0.838511314 -0.057087284  0.207833670


                        Mixing Matrix A

                rm          age          dis          rad
Fac_1  -1.924802270 -0.732378892 -0.333972363 -1.530684738
Fac_2   18.00148590  0.264115490  0.258989299  0.018197195
Fac_3   28.01061544  2.009289320 -1.783104365 -0.469957161


                        Mixing Matrix A

               tax        ptrat            b        lstat
Fac_1   0.219851068 -0.005472896 -0.025681503 -0.000573280
Fac_2  -0.030868387 -0.618337550  0.017416029 -3.889559263
Fac_3  -3.805088970  3.097411361  1.385200503  4.573239927


                        Mixing Matrix A

                          medv
                Fac_1   0.152065708
                Fac_2  -29.08259502
                Fac_3  -2.049074871
```

3. Deflation approach : Sigmoid Function "tanh"

```
optn = [ "print"          3 ,
         "orig"             ,
         "nfac"           3 ,
         "seed"         123 ,
         "appr"           0 ,
         "finet"      "off" ,
         "staboff"          ,
         "step"          1. ,
         "kern"       "tanh" ,
```

```
        "kfp1"           1. ,
        "kfp2"           1. ,
        "tolit"       1.e-4 ,
        "maxit"          1000 ];
< gof,aica,wica,scor,eval,evec > = fica(data,optn);


           ************************************
           (Fast) Independent Component Analysis
           ************************************

  Number Signals (Observations) . . . . . .          30
  Data Dimensionality (Number Variables). . .        13
  Number Components (Factors) . . . . . .            3
  Algorithmic Approach. . . . . .  Deflation
  (Nonlinear) Kernel Function . . . . .       Sigmoid (tanh)
  Function Parameter. . . . . .                 1.0000
  Maximum Number Iterations . . . . . .         1000
  Iteration Tolerance (Precision) . . . . .     0.0001
  (Starting) Step Size. . . . . .               1.0000

    ********************************************************
    Deflation (Projection Pursuit) Iteration for Dimension 1
    ********************************************************

             Iteration      D_minus       D_plus
                     1  1.000000000  1.000000000
                     2  1.633098300  1.154551836
                     3  0.305378168  1.976548551
                     4  0.385639722  1.962468345
                     5  0.490828333  1.938836648
                     6  0.687981754  1.877945981
                     7  1.019998125  1.720349914
                     8  1.806714398  0.857777991
                     9  1.972366914  0.331313685
                    10  1.987595466  0.222405625
                    11  1.999831431  0.025966267
                    12  1.999941032  0.015358015
                    13  1.999980668  0.008793549
                    14  1.999993079  0.005261548
                    15  1.999997432  0.003204868
                    16  1.999998931  0.002067618
                    17  1.999999500  0.001414389
                    18  1.999999733  0.001033846
                    19  1.999999843  0.000793370
                    20  1.999999901  0.000629841
```

```
                21  1.999999935  0.000508982
                22  1.999999957  0.000415626
                23  1.999999971  0.000340891
                24  1.999999980  0.000280385
                25  1.999999987  0.000230815
                26  1.999999991  0.000190166
                27  1.999999994  0.000156686
                28  1.999999996  0.000129139
                29  1.999999997  0.000106427
                30  1.999999998  8.7723e-005

 Iteration for Dimension 1 Converged after 30 Iterations


 ********************************************************
 Deflation (Projection Pursuit) Iteration for Dimension 2
 ********************************************************

          Iteration      D_minus       D_plus
                  1  1.000000000  1.000000000
                  2  1.999944824  0.014856033
                  3  1.999999939  0.000493318
                  4  2.000000000  1.4514e-005

 Iteration for Dimension 2 Converged after 4 Iterations


 ********************************************************
 Deflation (Projection Pursuit) Iteration for Dimension 3
 ********************************************************

          Iteration      D_minus       D_plus
                  1  1.000000000  1.000000000
                  2  2.000000000  1.5686e-015

 Iteration for Dimension 3 Converged after 2 Iterations


                    Mixing Matrix A

              crim           zn        indus          nox
Fac_1   0.205967847  0.106405674  0.065118945  1.700303918
Fac_2  -0.111895871  0.013874527  8.002781700 -4.791285798
Fac_3  -0.489176141  0.165765637  0.815673487  0.238230001


                    Mixing Matrix A
```

```
                rm          age          dis          rad
Fac_1  -1.801978789 -0.803653146  1.561488505  0.124386457
Fac_2   17.81938901 -0.291025373 -0.228831840  0.010188636
Fac_3  -5.629037100 -27.42767161 -2.801437698  3.839782002

                    Mixing Matrix A

               tax        ptrat            b        lstat
Fac_1   0.221696767  0.026173304  0.002117292 -0.000544126
Fac_2   0.617149589 -0.050172343  0.014374327  29.34053602
Fac_3  -0.143277312  3.086918323 -4.718168135 -0.703346709

                    Mixing Matrix A

                    medv
Fac_1  -0.321507846
Fac_2   0.102694574
Fac_3  -2.067898582
```

4. Deflation approach : Third order Polynomial "pow3"

```
optn = [ "print"           3 ,
         "orig"              ,
         "nfac"            3 ,
         "seed"          123 ,
         "appr"        "def" ,
         "finet"       "off" ,
         "staboff"           ,
         "step"           .5 ,
         "kern"       "pow3" ,
         "tolit"       1.e-4 ,
         "maxit"         30 ];
< gof,aica,wica,scor,eval,evec > = fica(data,optn);


             ************************************
             (Fast) Independent Component Analysis
             ************************************

  Number Signals (Observations) . . . . . .         30
  Data Dimensionality (Number Variables). . . . .   13
  Number Components (Factors) . . . . . .          3
  Algorithmic Approach. . . . . .  Deflation
  (Nonlinear) Kernel Function . . . . . Polynom 3rd D (pow3)
```

```
Maximum Number Iterations . . . . . .        30
Iteration Tolerance (Precision) . . . . . .    0.0001
(Starting) Step Size. . . . . .    0.5000

  ********************************************************
  Deflation (Projection Pursuit) Iteration for Dimension 1
  ********************************************************

          Iteration      D_minus       D_plus
                  1  1.000000000  1.000000000
                  2  0.605626171  1.906099929
                  3  0.971993994  1.747920958
                  4  0.239063550  1.985660751
                  5  0.120647375  1.996357736
                  6  0.075103563  1.998589366
                  7  0.046624954  1.999456455
                  8  0.028852409  1.999791874
                  9  0.017811118  1.999920689
                 10  0.010976890  1.999969877
                 11  0.006757696  1.999988583
                 12  0.004157390  1.999995679
                 13  0.002556565  1.999998366
                 14  0.001571728  1.999999382
                 15  0.000966110  1.999999767
                 16  0.000593788  1.999999912
             Reducing step size to 0.25
                 17  0.000182465  1.999999992
                 18  0.000147300  1.999999995
                 19  0.000118912  1.999999996
                 20  9.5994e-005  1.999999998

  Iteration for Dimension 1 Converged after 20 Iterations

  ********************************************************
  Deflation (Projection Pursuit) Iteration for Dimension 2
  ********************************************************

          Iteration      D_minus       D_plus
                  1  1.000000000  1.000000000
                  2  0.148402043  1.994486609
                  3  0.106151586  1.997180973
                  4  0.076442606  1.998538598
                  5  0.053214238  1.999291936
                  6  0.037428464  1.999649747
                  7  0.025934642  1.999831842
```

```
                 8  0.018094466  1.999918146
                 9  0.012541107  1.999960680
                10  0.008724734  1.999980970
                11  0.006051373  1.999990845
                12  0.004205145  1.999995579
                13  0.002918046  1.999997871
                14  0.002026797  1.999998973
                15  0.001406812  1.999999505
                16  0.000976922  1.999999761
              Reducing step size to 0.25
                17  0.000339089  1.999999971
                18  5.1791e-005  1.999999999

 Iteration for Dimension 2 Converged after 18 Iterations


 ********************************************************
 Deflation (Projection Pursuit) Iteration for Dimension 3
 ********************************************************

          Iteration      D_minus       D_plus
                  1  1.000000000  1.000000000
                  2  1.1102e-016  2.000000000

 Iteration for Dimension 3 Converged after 2 Iterations


                    Mixing Matrix A

              crim           zn        indus          nox
Fac_1   0.119232266 -0.181125446 -0.104702618 -1.429164618
Fac_2   0.187483108  0.240898998 -6.534112630 -18.77164960
Fac_3  -23.48504869  0.781731667 -0.333784819  0.164433305


                    Mixing Matrix A

                rm          age          dis          rad
Fac_1  -0.278354863 -2.159635713  0.278574980 -1.118873116
Fac_2   3.076053091 -0.261554279  0.178507405  0.192062626
Fac_3  -27.11846572  7.094121637 -2.464454719 -0.157716323


                    Mixing Matrix A

               tax        ptrat            b        lstat
Fac_1  -1.083229768  0.005376087 -0.015351822 -0.020621351
```

```
Fac_2   0.024390101 -0.378305200 -0.489783421  3.782806501
Fac_3  -4.758122755 -1.276288507 -0.941087026  4.515637368
```

```
                    Mixing Matrix A

                                medv
                Fac_1  -0.151321646
                Fac_2  -17.18312418
                Fac_3   2.399038569
```

5. Parallel approach : Second order polynomial "skew"

```
 optn = [ "print"           3 ,
          "orig"              ,
          "nfac"            3 ,
          "seed"          123 ,
          "appr"        "par" ,
          "finet"       "off" ,
          "staboff"           ,
          "step"           1. ,
          "kern"        "skew" ,
          "tolit"       1.e-4 ,
          "maxit"         20 ];
< gof,aica,wica,scor,eval,evec > = fica(data,optn);
```

```
             ************************************
             (Fast) Independent Component Analysis
             ************************************

  Number Signals (Observations) . . . . . .          30
  Data Dimensionality (Number Variables). . . . .    13
  Number Components (Factors) . . . . . .            3
  Algorithmic Approach. . . . . .             Parallel
  (Nonlinear) Kernel Function . . . . . Polynom 2nd D (skew)
  Maximum Number Iterations . . . . . .            100
  Iteration Tolerance (Precision) . . . . . .   0.0001
  (Starting) Step Size. . . . . .               1.0000

           ***************************************
           Parallel (Symmetric) Fix Point Iteration
           ***************************************

            Iteration        Diff1        Diff2
                    1  0.000000000  0.000000000
```

```
 2  0.562431160  0.000000000
 3  0.958088628  0.760776261
 4  0.997938250  0.960953058
 5  0.999836124  0.997555566
 6  0.999987888  0.999825368

Iteration Converged (Criterion: 0.0001) after 6 Iterations

                    Mixing Matrix A

              crim           zn        indus          nox
Fac_1   0.126612204 -1.733511828  0.258535470  0.003997982
Fac_2  -0.039121209  0.794607302 -0.081074070  0.002875458
Fac_3  -0.201057900 -1.774135093 -1.558670653 -0.025798619

                    Mixing Matrix A

                rm          age          dis          rad
Fac_1  -0.133819369 -2.628646570 -0.248052752 -0.002828727
Fac_2  -0.049345187 -17.42260887 -0.052530314  0.032758735
Fac_3   0.309415437 -9.699199792  0.269951820 -0.618479074

                    Mixing Matrix A

               tax        ptrat            b        lstat
Fac_1   2.236144566  0.843230743 -27.90880422  0.408886690
Fac_2   2.921500413 -0.157583469  0.119740200 -2.760191848
Fac_3  -29.11325105 -0.116980264  3.591043211 -4.062963399

                    Mixing Matrix A

                        medv
Fac_1  -1.227218323
Fac_2   1.607455755
Fac_3   4.789784220


                  Separating Matrix W

              crim           zn        indus          nox
Fac_1   0.000132173 -0.002846432  2.3501e-005 -1.0233e-006
Fac_2  -8.4591e-005  0.003659398  0.000244554  1.7807e-005
Fac_3  -0.000173245 -0.002559325 -0.001579911 -2.7758e-005
```

Separating Matrix W

	rm	age	dis	rad
Fac_1	-9.6417e-005	-0.001209690	-0.000253594	-0.000144699
Fac_2	-0.000240542	-0.052346683	-0.000208119	0.000333906
Fac_3	0.000319911	-0.004535990	0.000254374	-0.000673373

Separating Matrix W

	tax	ptrat	b	lstat
Fac_1	-0.004052435	0.001117343	-0.035779706	0.000345577
Fac_2	0.019608965	-0.000677419	0.005982132	-0.007570725
Fac_3	-0.031694204	0.000114005	-0.002203915	-0.003240312

Separating Matrix W

	medv
Fac_1	-0.000974849
Fac_2	0.003764860
Fac_3	0.004263824

Independent Component Matrix

	Fac_1	Fac_2	Fac_3
Row_01	-15.53693465	4.872192444	-10.51128866
Row_02	-15.27710713	2.989415444	-8.847326338
Row_03	-15.12456106	3.984661555	-8.684957247
Row_04	-15.08803062	4.405857007	-7.980277465
Row_05	-15.18132472	3.972132791	-8.019146982
Row_06	-15.07996694	3.692790212	-8.065244483
Row_07	-15.53446695	5.007180127	-11.01625037
Row_08	-15.61854497	3.435542401	-11.15665332
Row_09	-15.24171926	3.048530977	-11.23198776
Row_10	-15.23446865	3.893073316	-11.11614401
Row_11	-15.44751695	3.448029093	-11.19448764
Row_12	-15.59667320	4.140146479	-11.11266579
Row_13	-15.31625911	6.392141724	-10.89563005
Row_14	-15.51546184	5.160150533	-10.83514506
Row_15	-14.93607518	3.847493277	-10.91678785
Row_16	-15.46262876	5.426529848	-10.81118521
Row_17	-15.11966359	6.824192295	-10.64875112
Row_18	-15.17119819	3.998357246	-10.93630503

```
Row_19  -11.62230775  5.807328069 -10.49540008
Row_20  -15.30843443  4.690576049 -10.87644448
Row_21  -14.82058469  3.016373639 -11.02579671
Row_22  -15.38934133  3.654650846 -10.97146991
Row_23  -15.54270129  3.496272407 -11.02706522
Row_24  -15.46724941  3.036334749 -11.06628906
Row_25  -15.45502659  3.375098522 -11.02261911
Row_26  -12.19037396  3.263044215 -10.79219353
Row_27  -14.82763559  3.484647366 -10.95778231
Row_28  -12.30067056  3.115917892 -10.81131325
Row_29  -15.23076287  3.358057025 -10.98642440
Row_30  -14.94906305  3.699575271 -10.92352107
```

6. Parallel approach : Exponential function "gaus"

```
optn = [ "print"           3 ,
         "orig"              ,
         "nfac"            3 ,
         "seed"          123 ,
         "appr"        "par" ,
         "finet"       "off" ,
         "staboff"           ,
         "step"           1. ,
         "kern"       "gaus" ,
         "kfp1"           1. ,
         "kfp2"           1. ,
         "tolit"       1.e-4 ,
         "maxit"         20 ];
< gof,aica,wica,scor,eval,evec > = fica(data,optn);


             ************************************
             (Fast) Independent Component Analysis
             ************************************

  Number Signals (Observations) . . . . . .         30
  Data Dimensionality (Number Variables). . . . . .         13
  Number Components (Factors) . . . . . .          3
  Algorithmic Approach. . . . . .   Parallel
  (Nonlinear) Kernel Function . . . . . .  Radial Base (gauss)
  Function Parameter. . . . . .     1.0000
  Maximum Number Iterations . . . . . .         20
  Iteration Tolerance (Precision) . . . . . .     0.0001
  (Starting) Step Size. . . . . .     1.0000


          ****************************************
```

```
          Parallel (Symmetric) Fix Point Iteration
          ****************************************

            Iteration        Diff1        Diff2
                    1  0.000000000  0.000000000
                    2  0.860274699  0.000000000
                    3  0.060608007  0.031679718
                    4  0.932015898  0.155546222
                    5  0.984273676  0.980590864
                    6  0.997381717  0.994026336
                    7  0.999352445  0.999208095
                    8  0.999858138  0.999812933
                    9  0.999965361  0.999963697

Iteration Converged (Criterion: 0.0001) after 9 Iterations

                    Mixing Matrix A

              crim           zn        indus          nox
Fac_1   0.136186630  0.074134913  0.828435216  0.010000851
Fac_2   0.156930588  1.949909657  1.327907145  0.023999625
Fac_3   0.121702057 -1.725217579  0.230706539  0.003717773

                    Mixing Matrix A

                rm          age          dis          rad
Fac_1  -0.110770013  19.80159983 -0.094224644  0.267048605
Fac_2  -0.294360788  0.194684300 -0.260451635  0.558714396
Fac_3  -0.131004633 -3.519986014 -0.245866324 -0.010989437

                    Mixing Matrix A

               tax        ptrat            b        lstat
Fac_1   11.46721822  0.232023289 -3.076140819  4.382848235
Fac_2   26.94397041  0.021485174 -2.903730786  2.243275449
Fac_3   1.908470261  0.833821682 -27.81936353  0.227762051

                    Mixing Matrix A

                          medv
                Fac_1  -3.756410394
                Fac_2  -3.427532402
                Fac_3  -1.083242482
```

7. Parallel approach : Exponential function "tanh"

```
 optn = [ "print"           3 ,
          "orig"              ,
          "nfac"            3 ,
          "seed"          123 ,
          "appr"        "par" ,
          "finet"       "off" ,
          "staboff"           ,
          "step"           1. ,
          "kern"       "tanh" ,
          "kfp1"           1. ,
          "kfp2"           1. ,
          "tolit"       1.e-4 ,
          "maxit"          20 ];
< gof,aica,wica,scor,eval,evec > = fica(data,optn);


              ************************************
              (Fast) Independent Component Analysis
              ************************************

   Number Signals (Observations) . . . . . .          30
   Data Dimensionality (Number Variables). . . . .    13
   Number Components (Factors) . . . . . .            3
   Algorithmic Approach. . . . . .          Parallel
   (Nonlinear) Kernel Function . . . . .       Sigmoid (tanh)
   Function Parameter. . . . . .              1.0000
   Maximum Number Iterations . . . . . .           20
   Iteration Tolerance (Precision) . . . . . .     0.0001
   (Starting) Step Size. . . . . .            1.0000


             ***************************************
             Parallel (Symmetric) Fix Point Iteration
             ***************************************

     Iteration        Diff1        Diff2
             1  0.000000000  0.000000000
             2  0.837784301  0.000000000
             3  0.850686271  0.445966900
             4  0.960644543  0.955747141
             5  0.990955396  0.967177503
             6  0.998165431  0.996291854
             7  0.999509490  0.999380299
             8  0.999868487  0.999883696
             9  0.999963362  0.999970653
```

```
Iteration Converged (Criterion: 0.0001) after 9 Iterations

                    Mixing Matrix A

               crim          zn        indus          nox
Fac_1   0.157232915  1.968662417  1.335053511  0.024075108
Fac_2   0.135920161  0.029123546  0.815875859  0.009768761
Fac_3   0.121609698 -1.705150860  0.234193302  0.003844654

                    Mixing Matrix A

                rm          age          dis          rad
Fac_1  -0.294249644  0.464327547 -0.258943628  0.561898528
Fac_2  -0.109031727  19.75073410 -0.094370907  0.260376558
Fac_3  -0.132701593 -3.771533049 -0.247398295 -0.008461297

                    Mixing Matrix A

               tax        ptrat            b        lstat
Fac_1   27.05532575  0.015449512 -2.647268539  2.292107029
Fac_2   11.17726636  0.242542023 -3.402390810  4.359080988
Fac_3   2.048159983  0.830957097 -27.80695935  0.195389998

                    Mixing Matrix A

                          medv
                Fac_1  -3.459868539
                Fac_2  -3.730067366
                Fac_3  -1.071409738
```

8. Parallel approach : Third order Polynomial "pow3"

```
optn = [ "print"           3 ,
         "orig"              ,
         "nfac"            3 ,
         "seed"          123 ,
         "appr"        "par" ,
         "finet"       "off" ,
         "staboff"           ,
         "step"           .5 ,
         "kern"       "pow3" ,
```

```
         "tolit"       1.e-4 ,
         "maxit"          30 ];
< gof,aica,wica,scor,eval,evec > = fica(data,optn);


            ************************************
            (Fast) Independent Component Analysis
            ************************************

   Number Signals (Observations) . . . . . .          30
   Data Dimensionality (Number Variables). . . . .    13
   Number Components (Factors) . . . . . .            3
   Algorithmic Approach. . . . . .  Parallel
   (Nonlinear) Kernel Function . . . . . Polynom 3rd D (pow3)
   Maximum Number Iterations . . . . . .          30
   Iteration Tolerance (Precision) . . . . . .    0.0001
   (Starting) Step Size. . . . . .    0.5000


           ***************************************
           Parallel (Symmetric) Fix Point Iteration
           ***************************************

             Iteration        Diff1        Diff2
                     1  0.000000000  0.000000000
                     2  0.552866206  0.000000000
                     3  0.009248534  0.227395448
                     4  0.573514542  0.557855303
                     5  0.045591390  0.637215443
                     6  0.291428397  0.648088503
                     7  0.953495508  0.358031713
                     8  0.997192832  0.938990762
                     9  0.997965542  0.998053447
                    10  0.998003842  0.999579360
                    11  0.997797433  0.999907588
                    12  0.999434218  0.999462209
                    13  0.999996811  0.999468030

   Iteration Converged (Criterion: 0.0001) after 13 Iterations


                        Mixing Matrix A

                 crim           zn        indus          nox
    Fac_1   0.059677641  2.502082779  0.928955134  0.017701353
    Fac_2   0.178965214  0.306727715  1.114418549  0.015266696
    Fac_3   0.149651026 -0.655377162  0.630857714  0.011975431
```

```
                    Mixing Matrix A

                rm          age          dis          rad
Fac_1  -0.178329073 -0.943869596 -0.096766146  0.453483129
Fac_2  -0.184840704  18.88732187 -0.173774550  0.378202482
Fac_3  -0.223859642 -6.849068692 -0.312413677  0.186851371


                    Mixing Matrix A

               tax        ptrat            b        lstat
Fac_1   20.91239030 -0.408834380  11.08379490  1.227138382
Fac_2   17.12975378  0.319045858 -6.586356444  4.761317000
Fac_3   11.41754619  0.693267053 -25.01161093  0.342798427


                    Mixing Matrix A

                      medv
Fac_1  -1.939555612
Fac_2  -4.499389364
Fac_3  -1.739434124
```

3.6 Function `histplot`

histplot(y<,titl<,optn<,fpath>>>)

Purpose: The `histplot` function plots horizontal or vertical histograms and/or writes the output to a text file with user specified filename.

Input: **y** This argument must be a vector or a matrix. A matrix with c columns will generate c plots, i.e. one for each column.

titl This argument must be a single string argument specifying the title for the plot.

optn This argument must be a numeric scalar or a vector specifying the output options:

1 amount of printed output (=0: no printed output)

2 the y_{min} value for the plot; if missing this is the minimum of the input values.

3 the y_{max} value for the plot; if missing this is the maximum of the input values.

4 =0: prints horicontal plots, =1: prints vertical plots

fpath This argument must be a single string argument specifying the (path and) file name for a text output file containing the plot.

Output: This function has no return arguments. However, it has printed and/or file output.

Restrictions: 1. Missing values for y are not permitted.

2. String or complex data for y are not permitted.

Relationships: xyplot()

Examples: 1. Eigenvalue plot:

```
eval = [ 0.27628586 0.07152085 0.02729783 ];
titl = "Eigenvalues from ANACOR";
optn = [ 2 ,
         0. ];
histplot(eval,titl,optn);
```

```
                    ***********************
                    Eigenvalues from ANACOR
                    ***********************

  N      Value +-----------+-----------+-----------+-----------+
  1 0.27628586 ***********************************************
  2 0.07152085 ************
  3 0.02729783 ****
```

3.7 Function `homals`

< gof,eval,discr,obscor,cquant,stquant > = homals(a,optn)

Purpose: The `homals` function

Input: data This argument must be a $N \times n$ matrix containing the measured levels of n categorical variables. The levels can be integer or string coded. A sufficiently small number of missing values are permitted.

optn This argument must be specified in form of a two column matrix where the first column defines the option as string value (in quotes) and the second column can be used for a numeric or string specification of the option. See table below for content.

Option	Second Column	Meaning
"dimadd"		perform computation in $nfac+1$ dimensions but print only p
"maxit"	int	maximum number of iterations (default is 200)
"nfac"	int	number of factors (dimension) of decomposition
"nowgtm"		no weighting of observations with missings
"nonort"	int	number subiterations without orthogonalizations (default = 1)
"phis"		print optimization history
"print"	int	indicates the amount of printed output (=0: no printed output, default is 2)
"plot"	int	indicates the amount of plot output (=0: no plot output is default)
"seed"	int	seed for random generator (valid only when "orig" is not specified)
"stan"		compute and print standardized category quantif.
"tolit"	real	termination tolerance for final iteration (default is $5.e-4$)
"wgtmis"		weight observations with missing values (default)

Output: There are at most six results returned:

gof a vector of scalar results, like an inidicator for the success of the iterations, the computer time used and the values of differently defined *stress* for the final solution.

eval a vector of $nfac$ eigenvalues.

discrim the $n \times nfac$ matrix of discrimination measures.

obscor the $N \times nfac$ matrix of object scores.

cquant the $Ncat \times nfac$ matrix of category quantifications, where $Ncat$ is the total number of categories of all n variables

stquant the $Ncat \times nfac$ matrix of standardized category quantifications.

Restrictions: 1. String data for levels are permitted. The second input argument can have no missing values.

2. Many missing values or difficult missing value patterns may affect the convergence of the iterative processes.

Relationships: princals(), primals(), anacor(), anaprof() canals(), overals(), pca(), svd()

Examples: 1. Small Example by Gifi (1990), $N = 10$, $n = 3$:

```
data = [  1   1   1,   2   2   2,   1   3   2,
          1   1   1,   2   1   2,   3   1   2,
          1   1   1,   1   1   2,   3   1   2,
          1   1   2 ];
```

```
optn = [ "nfac"       3 ,
         "nonort"     1 ,
         "stand"        ,
         "maxit"    100 ,
         "tolit"  1.e-4 ,
         "plot"       0 ,
         "print"      3 ,
         "phis"         ];
< gof,eval,disc,scor,quant,stqnt > = homals(data,optn);
```

****
Number of Observations for Class Levels
****

Variable	Value	Nobs	Proportion
C[1]	1	6	60.000000
	2	2	20.000000
	3	2	20.000000
C[2]	1	8	80.000000
	2	1	10.000000
	3	1	10.000000
C[3]	1	3	30.000000
	2	7	70.000000

Input Data Matrix: 10 by 3

	Col_1	Col_2	Col_3
Row_01	0.000000000	0.000000000	0.000000000
Row_02	1.000000000	1.000000000	1.000000000
Row_03	0.000000000	2.000000000	1.000000000
Row_04	0.000000000	0.000000000	0.000000000
Row_05	1.000000000	0.000000000	1.000000000
Row_06	2.000000000	0.000000000	1.000000000
Row_07	0.000000000	0.000000000	0.000000000
Row_08	0.000000000	0.000000000	1.000000000
Row_09	2.000000000	0.000000000	1.000000000
Row_10	0.000000000	0.000000000	1.000000000

****
History of Iterations (Initial Fit=0)
****

Iteration	Fit	OldFit	FitDiff
1	0.33178094	0.00000000	0.33178094
2	1.29656348	0.33178094	0.96478254
3	1.43677297	1.29656348	0.14020949
4	1.44292187	1.43677297	0.00614891
5	1.44319024	1.44292187	2.684e-004
6	1.44320634	1.44319024	1.610e-005

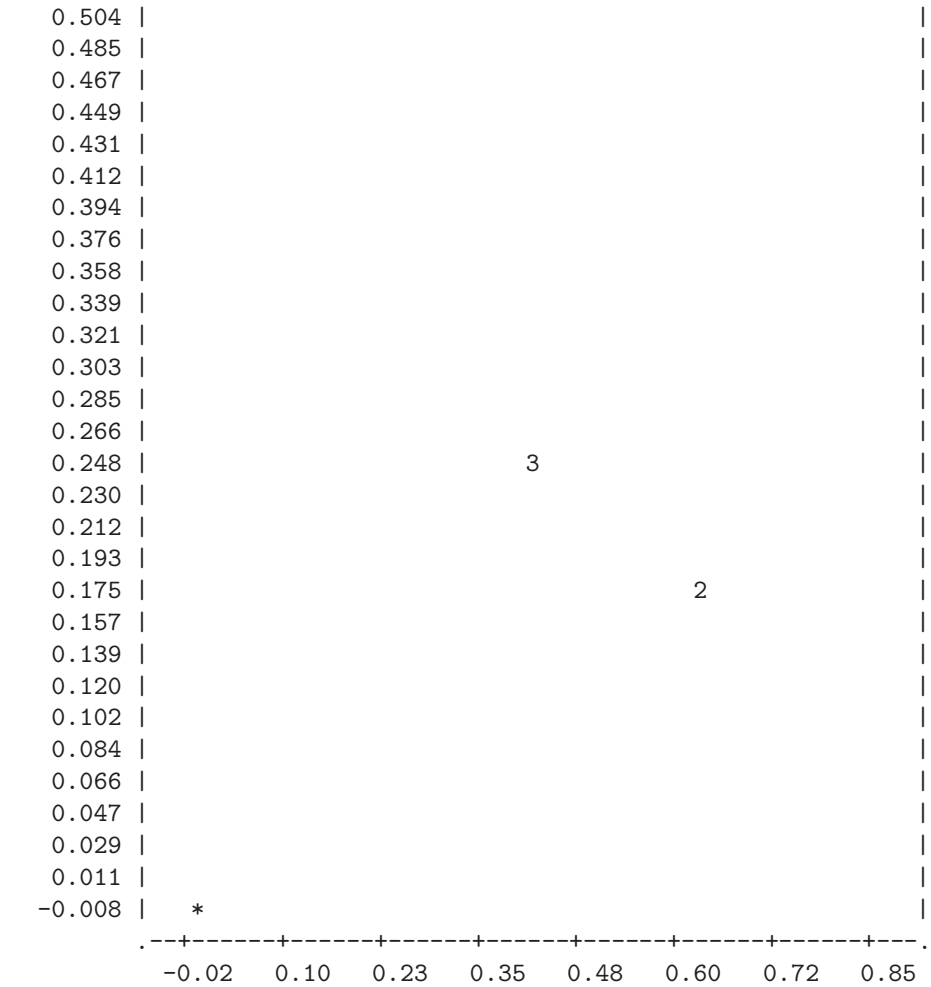
Eigenvalues
---

1 : 0.6285 0.4255 0.3891

Discrimination Measures

	Fac_1	Fac_2	Fac_3
Col_1	0.809151003	0.850489710	0.071774858
Col_2	0.638170635	0.174084683	0.929987134
Col_3	0.438309688	0.252070656	0.165580662

****
Plot of the Discrimination Measures (* = Origin)
****



Object Scores

	Row_01	Row_02	Row_03	Row_04
Fac_1	1.011297486	-2.395991510	0.214007060	1.011297486
Fac_2	0.766919072	1.199780950	0.221256041	0.766919072
Fac_3	-0.621574515	-0.317800677	2.893072883	-0.621574515

Object Scores

	Row_05	Row_06	Row_07	Row_08
Fac_1	-0.979032658	-0.181943388	1.011297486	0.245505713
Fac_2	0.124296899	-1.832373356	0.766919072	-0.090672197
Fac_3	-0.319429038	-0.337461382	-0.621574515	0.141901571

```
                    Object Scores

                        Row_09       Row_10
                Fac_1  -0.181943388  0.245505713
                Fac_2  -1.832373356 -0.090672197
                Fac_3  -0.337461382  0.141901571


                    Category Quantifications
                    ************************

Var   Cat MargFrq       Fac_1       Fac_2       Fac_3
  1     1       6  0.62315182  0.39011148  0.21869208
  1     2       2 -1.68751208  0.66203892 -0.31861486
  1     3       2 -0.18194339 -1.83237336 -0.33746138
Var   Cat MargFrq       Fac_1       Fac_2       Fac_3
  2     1       8  0.27274806 -0.17762962 -0.32190903
  2     2       1 -2.39599151  1.19978095 -0.31780068
  2     3       1  0.21400706  0.22125604  2.89307288
Var   Cat MargFrq       Fac_1       Fac_2       Fac_3
  3     1       3  1.01129749  0.76691907 -0.62157452
  3     2       7 -0.43341321 -0.32867960  0.26638908


          Standardized Category Quantifications
          *************************************

Var   Cat MargFrq       Fac_1       Fac_2       Fac_3
  1     1       6  0.76320200  0.47778703  0.26784200
  1     2       2 -0.84375604  0.33101946 -0.15930743
  1     3       2 -0.09097169 -0.91618668 -0.16873069
Var   Cat MargFrq       Fac_1       Fac_2       Fac_3
  2     1       8  0.54549611 -0.35525925 -0.64381805
  2     2       1 -0.79866384  0.39992698 -0.10593356
  2     3       1  0.07133569  0.07375201  0.96435763
Var   Cat MargFrq       Fac_1       Fac_2       Fac_3
  3     1       3  0.66204961  0.50206639 -0.40691604
  3     2       7 -0.66204961 -0.50206639  0.40691604
```

We also show the returned objects:

```
print "GOF=",  gof;
print "Eigenvalues=", eval;
print "Discrim. Meas.=", disc;
print "Object Scores=", scor;
```

```
  print "Categ. Coord.=", quant;
  print "Stand. Cat. Coord=", stqnt;
```

GOF=

	1
Failure	0.00000
Time	0.00000
TotFit	1.44321
Dimension	3.00000
unused	0.00000
unused	0.00000
unused	0.00000
unused	0.00000
unused	0.00000
unused	0.00000

Eigenvalues=

	Fac_1	Fac_2	Fac_3
1	0.62854	0.42555	0.38911

Discrim. Meas.=

	Fac_1	Fac_2	Fac_3
Col_1	0.80915	0.85049	0.07177
Col_2	0.63817	0.17408	0.92999
Col_3	0.43831	0.25207	0.16558

Object Scores=

	Fac_1	Fac_2	Fac_3
Row_01	1.01130	0.76692	-0.62157
Row_02	-2.39599	1.19978	-0.31780
Row_03	0.21401	0.22126	2.89307
Row_04	1.01130	0.76692	-0.62157
Row_05	-0.97903	0.12430	-0.31943
Row_06	-0.18194	-1.83237	-0.33746
Row_07	1.01130	0.76692	-0.62157
Row_08	0.24551	-0.09067	0.14190
Row_09	-0.18194	-1.83237	-0.33746
Row_10	0.24551	-0.09067	0.14190

```
Categ. Coord.=
          |     Fac_1     Fac_2     Fac_3
----------------------------------------
  V_1_C_1 |   0.62315   0.39011   0.21869
  V_1_C_2 |  -1.68751   0.66204  -0.31861
  V_1_C_3 |  -0.18194  -1.83237  -0.33746
  V_2_C_1 |   0.27275  -0.17763  -0.32191
  V_2_C_2 |  -2.39599   1.19978  -0.31780
  V_2_C_3 |   0.21401   0.22126   2.89307
  V_3_C_1 |   1.01130   0.76692  -0.62157
  V_3_C_2 |  -0.43341  -0.32868   0.26639


Stand. Cat. Coord=
          |     Fac_1     Fac_2     Fac_3
----------------------------------------
  V_1_C_1 |   0.76320   0.47779   0.26784
  V_1_C_2 |  -0.84376   0.33102  -0.15931
  V_1_C_3 |  -0.09097  -0.91619  -0.16873
  V_2_C_1 |   0.54550  -0.35526  -0.64382
  V_2_C_2 |  -0.79866   0.39993  -0.10593
  V_2_C_3 |   0.07134   0.07375   0.96436
  V_3_C_1 |   0.66205   0.50207  -0.40692
  V_3_C_2 |  -0.66205  -0.50207   0.40692
```

3.8 Function `ita`

< gof,bmat,ecor,caval,cqord,cstat > = ita(data,optn)

< gof,bmat,eqor,diff,cqord,cstat > = ita(data,optn)

Purpose: The `ita` function implements two algorithms for classical (Van Leeuwe, 1974) and inductive (Schrepp, 1999, 2003, 2006) item tree analysis. Relationships among the items of a $N \times n$ matrix of binary data. The data matrix may contain a column with frequencies.

Input: data $N \times n$ matrix of binary data, i.e. values can only be one, zero, or missing. The data may have an additional column with frequency values.

optn This argument must be specified in form of a two column matrix where the first column defines the option as string value (in quotes) and the second column can be used for a numeric or string specification of the option. See table below for content.

Option	Second Column	Meaning
"cspac"		compute compatible space
"freq"	int	column number for frequency value
"meth"	"class"	for classical method
	"induc"	for inductive method
"print"	int	indicates the amount of printed output (=0: no printed output, default is 2)

Output: **gof** some goodness-of-fit measures

bmat The $n \times n$ **B** matrix where b_{ij} is the number of data rows for which item $i = 0$ and item $j = 1$

ecor only for classical method: symmetric $n \times n$ matrix of empirical correlations of the n items (columns)

eqor only for inductive method: unsymmetric $n \times n$ matrix of quasi order for lowest level with $i \geq_L j$

caval only for classical method: table of CA values

diff only for inductive method: table of Diff values

cqord constructed quasi orders based on the best level

cstat compatible states with best fitting quasi order

Restrictions: 1. The data matrix may contain missing values. But otherwise the data must be either zero or one.

2.

Relationships:

Examples: 1. Classical ITA of 200×5 binary data matrix by Schrepp (2006).

```
optn = [ "cspac"          ,
         "meth"   "class" ,
         "print"        2 ];
< gof,bmat,ecor,caval,cqord,cstat > = ita(data,optn);


                  *********************************
                  Classical Item Tree Analysis (ITA)
                  *********************************

  Number Rows of Matrix . . . . . .          200
  Number Columns of Matrix  . . . . . .        5


                  Distribution of Values per Column
                  *********************************
```

Column	Freq(1)	Freq(0)	Freq(.)
1	0.205	0.795	0.000
2	0.335	0.665	0.000
3	0.600	0.400	0.000
4	0.595	0.405	0.000
5	0.875	0.125	0.000

Table of the A

	V 1	V 2	V 3	V 4
V 1	159.0000000	123.0000000	71.00000000	78.00000000
V 2	123.0000000	133.0000000	72.00000000	74.00000000
V 3	71.00000000	72.00000000	80.00000000	43.00000000
V 4	78.00000000	74.00000000	43.00000000	81.00000000
V 5	23.00000000	20.00000000	16.00000000	19.00000000

Table of the A

	V 5
V 1	23.00000000
V 2	20.00000000
V 3	16.00000000
V 4	19.00000000
V 5	25.00000000

Table of the B

	V 1	V 2	V 3	V 4
V 1	0.000000000	36.00000000	88.00000000	81.00000000
V 2	10.00000000	0.000000000	61.00000000	59.00000000
V 3	9.000000000	8.000000000	0.000000000	37.00000000
V 4	3.000000000	7.000000000	38.00000000	0.000000000
V 5	2.000000000	5.000000000	9.000000000	6.000000000

Table of the B

	V 5
V 1	136.0000000
V 2	113.0000000
V 3	64.00000000
V 4	62.00000000
V 5	0.000000000

Table of the C

	V 1	V 2	V 3	V 4
V 1	0.000000000	10.00000000	9.000000000	3.000000000
V 2	36.00000000	0.000000000	8.000000000	7.000000000
V 3	88.00000000	61.00000000	0.000000000	38.00000000
V 4	81.00000000	59.00000000	37.00000000	0.000000000
V 5	136.0000000	113.0000000	64.00000000	62.00000000

Table of the C

	V 5
V 1	2.000000000
V 2	5.000000000
V 3	9.000000000
V 4	6.000000000
V 5	0.000000000

Table of the D

	V 1	V 2	V 3	V 4
V 1	41.00000000	31.00000000	32.00000000	38.00000000
V 2	31.00000000	67.00000000	59.00000000	60.00000000
V 3	32.00000000	59.00000000	120.0000000	82.00000000
V 4	38.00000000	60.00000000	82.00000000	119.0000000
V 5	39.00000000	62.00000000	111.0000000	113.0000000

Table of the D

	V 5
V 1	39.00000000
V 2	62.00000000
V 3	111.0000000
V 4	113.0000000
V 5	175.0000000

Empirical Correlations R

	V 1	V 2	V 3	V 4
V 1	0.000000000	0.453046590	0.187083553	0.343259101
V 2	0.453046590	0.000000000	0.406526636	0.434512643
V 3	0.187083553	0.406526636	0.000000000	0.220386128
V 4	0.343259101	0.434512643	0.220386128	0.000000000

```
V 5   0.117030887  0.108106429  0.185164020  0.273333800
```

Empirical Correlations R

	V 5
V 1	0.117030887
V 2	0.108106429
V 3	0.185164020
V 4	0.273333800
V 5	0.000000000

CA-values

Level	Size	CA	REP-PO	Trans
0	5	0.9105	1.0000	yes
2	6	0.9113	0.9900	yes
3	7	0.9225	0.9750	yes
5	8	0.9211	0.9600	yes
6	9	0.9252	0.9500	yes
7	10	0.9418	0.9250	yes
8	11	0.9553	0.9000	yes
9	13	0.9493	0.8650	yes
10	14	0.9630	0.8400	yes
36	15	0.9399	0.7050	yes
37	16	0.8856	0.5650	yes
38	17	0.8840	0.4150	yes
59	18	0.8543	0.3100	no
61	19	0.8221	0.3100	no
62	20	0.7727	0.1900	no
64	21	0.7140	0.1900	no
81	22	0.6715	0.1900	no
88	23	0.6106	0.1900	no
113	24	0.5336	0.1900	no
136	25	0.4562	0.1900	yes

Constructed Quasi-order for Level 10

Col -> Row	B[i,j]
1 2	10
1 3	9
2 3	8
1 4	3

```
  2      4         7
  1      5         2
  2      5         5
  3      5         9
  4      5         6


*****************
Compatible States
*****************

     0 0 0 0 0
     0 0 0 0 1
     0 0 1 0 1
     0 0 0 1 1
     0 1 1 1 1
     1 1 1 1 1

 6 States are compatible with the best fitting quasi-order.
     These are 18.75 per cent of all possible states.


                      Fit Indices
 Maximal CA-Value Found at Level . . . . . .        10
 Mean Violation of an Implication. . . . . .    6.5556
 Reproducibility Coefficient . . . . . .        0.9450
 Correlational Agreement Coefficient . . . . .  0.9630
 REP-PO Coefficient. . . . . .                  0.8400
```

2. Inductive ITA of 200×5 binary data matrix by Schrepp (2006).

```
optn = [ "cspac"          ,
         "meth"   "induc" ,
         "print"        2 ];
< gof,bmat,eqor,diff,cqord,cstat > = ita(data,optn);


               *********************************
               Inductive Item Tree Analysis (ITA)
               *********************************

 Number Rows of Matrix . . . . . .        200
 Number Columns of Matrix  . . . . .        5


               Distribution of Values per Column
               *********************************
```

Column	Freq(1)	Freq(0)	Freq(.)
1	0.205	0.795	0.000
2	0.335	0.665	0.000
3	0.600	0.400	0.000
4	0.595	0.405	0.000
5	0.875	0.125	0.000

Table of the B

	V 1	V 2	V 3	V 4
V 1	0.000000000	36.00000000	88.00000000	81.00000000
V 2	10.00000000	0.000000000	61.00000000	59.00000000
V 3	9.000000000	8.000000000	0.000000000	37.00000000
V 4	3.000000000	7.000000000	38.00000000	0.000000000
V 5	2.000000000	5.000000000	9.000000000	6.000000000

Table of the B

	V 5
V 1	136.0000000
V 2	113.0000000
V 3	64.00000000
V 4	62.00000000
V 5	0.000000000

Quasi-Orders (lowest level with i <=L j)

	V 1	V 2	V 3	V 4
V 1	0.000000000	36.00000000	136.0000000	136.0000000
V 2	10.00000000	0.000000000	136.0000000	136.0000000
V 3	9.000000000	8.000000000	0.000000000	37.00000000
V 4	3.000000000	7.000000000	38.00000000	0.000000000
V 5	2.000000000	5.000000000	9.000000000	6.000000000

Quasi-Orders (lowest level with i <=L j)

	V 5
V 1	136.0000000
V 2	136.0000000
V 3	136.0000000
V 4	136.0000000
V 5	0.000000000

Diff-Values
****

L	Size	Diff
0	0	154.507635
2	1	110.279245
3	2	94.5722731
5	3	91.6990120
6	4	90.6900445
7	5	70.1424417
8	6	53.5741768
9	8	49.0126842
10	9	40.8740545
36	10	87.8806028
37	11	111.413533
38	12	132.676554
136	20	850.971717

Constructed Quasi-order for Level 10
****

Col -> Row		Support	Confidence	B[i,j]
1	2	0.1550	0.7561	10
1	3	0.1600	0.7805	9
2	3	0.2950	0.8806	8
1	4	0.1900	0.9268	3
2	4	0.3000	0.8955	7
1	5	0.1950	0.9512	2
2	5	0.3100	0.9254	5
3	5	0.5550	0.9250	9
4	5	0.5650	0.9496	6

****
Compatible States
****

0 0 0 0 0
0 0 0 0 1
0 0 1 0 1
0 0 0 1 1
0 1 1 1 1
1 1 1 1 1

6 States are compatible with the best fitting quasi-order.
These are 18.75 per cent of all possible states.

```
                        Fit Indices
Minimal Diff-Value Found at Level . . . . . .        10
Mean Violation of an Implication. . . . . .     6.5556
Reproducibility Coefficient . . . . . .         0.9450
```

3.9 Function `overals`

< gof,eval,cquan,load,scor > = overals(data,vtyp,sets,optn)

Purpose: The `overals` function implements a K set generalization of the common 2 set canonical correlation analysis. It requires that the n variables (columns) of the data are assigned to $K < n$ variable sets.

Input: data This argument must be a $N \times n$ matrix containing the measured levels of n categorical variables. The levels can be integer or string coded. A sufficiently small number of missing values are permitted.

vtyp This argument must be a n vector of integers between 0 and 3

=0 single nominal

=1 single ordinal

=2 single numeric

=3 multiple nominal

sets This argument must be a n vector of integers specifying the set number of each variable (column).

optn This argument must be specified in form of a two column matrix where the first column defines the option as string value (in quotes) and the second column can be used for a numeric or string specification of the option. See table below for content.

Option	Second Column	Meaning
`"ftol"`	real	termination tolerance for final iteration (default is $5.e-4$)
`"init"`		perform preprocessing iteration
`"maxit"`	int	maximum number of iterations (default is 200)
`"nfac"`	int	number of factors (dimensions)
`"phis"`		print optimization history
`"print"`	int	indicates the amount of printed output (=0: no printed output, default is 2)
`"plot"`	int	indicates the amount of plot output (=0: no plot output is default)
`"seed"`	int	seed for random generator (valid only when `"orig"` is not specified)

Output: There are at most five results returned:

gof a vector of scalar results, like an inidicator for the success of the iterations, the computer time used and the values of differently defined *stress* for the final solution.

eval a vector of p eigenvalues.

cquan the $Ncat \times p$ matrix of category quantifications, where $Ncat$ is the total number of categories of all n variables

load the $n \times p$ matrix of factor loadings

scor the $N \times p$ matrix of object scores

Restrictions: 1. String data for levels are permitted. The second input argument can have no missing values.

2. Many missing values or difficult missing value patterns may affect the convergence of the iterative processes.

Relationships: canals(), cancorr(), princals(), primals(), homals()

Examples: 1. OVERALS: Economic Inequality and Polit. Instability, $N = 47$, $n = 9$:

```
print "OVERALS: Economic Inequality and Polit. Instability";
print "Third Var Missing=9: Cats: 6 5 5 8 5 6 3 5 3";
data =
 [ 1 1 1 4 4 1 2 1 3,   1 1 1 5 3 3 2 2 3,  1 1 2 6 2 5 1 1 1,   1 2 2 3 2 6 2 2
   1 2 2 7 1 4 2 1 1,   1 2 3 7 1 3 1 1 1,  2 3 5 1 4 2 3 2 1,   2 3 4 3 3 5 2 4
   2 3 3 7 1 3 1 1 1,   2 3 4 6 2 6 3 2 2,  2 3 5 6 1 6 2 2 1,   2 3 2 5 2 5 2 1
   2 3 2 6 3 6 2 1 2,   3 3 5 6 1 5 2 1 1,  3 3 3 7 2 5 1 1 1,   3 4 4 2 3 1 2 1
   3 3 2 6 2 5 2 1 1,   3 4 3 2 4 4 3 5 3,  3 4 2 6 1 2 2 1 2,   3 4 2 1 4 5 2 1
   4 5 3 8 1 5 2 1 1,   4 4 5 6 1 5 2 1 1,  4 4 3 4 3 6 2 3 3,   4 5 2 5 2 5 2 1
   4 5 3 2 4 6 3 2 3,   4 5 3 3 3 6 2 2 2,  4 5 3 2 4 5 3 4 3,   4 5 . 4 4 5 2 2
   4 5 4 7 1 5 1 1 1,   4 5 5 4 3 1 2 2 3,  4 5 5 4 3 5 3 5 3,   4 5 3 3 3 4 2 3
   5 5 4 5 2 6 3 2 2,   5 5 4 5 2 5 2 2 1,  5 5 3 3 4 6 2 2 3,   5 5 2 4 4 6 3 2
   5 5 3 4 3 5 3 4 2,   5 5 3 3 4 5 3 3 3,  5 5 4 4 2 5 3 4 2,   5 5 3 3 3 6 3 2
   5 5 . 2 3 5 2 3 3,   5 5 5 3 5 6 2 4 3,  5 5 2 4 3 5 2 3 2,   6 5 3 6 3 5 2 4
   6 5 . 7 1 4 1 1 1,   6 5 3 3 2 5 2 2 2,  6 5 3 1 4 6 3 5 3 ];


cnam = [ "GINI" "FARM" "RENT" "GNPR" "LABO"
         "INST" "ECKS" "DEAT" "DEMO" ];
rnam = [ "Yug" "Pol" "Den" "Jap" "Can" "Swi" "Ind" "Phi" "Swe" "Fra"
         "Bel" "Ire" "Fin" "Net" "Lux" "Tai" "Nor" "SVi" "WGe" "Lib"
         "USA" "UnK" "Pan" "Aut" "Egy" "Gre" "Hon" "Nic" "NZe" "Spa"
         "Cub" "DoR" "Ita" "Uru" "ElS" "Bra" "Col" "Gua" "Arg" "Ecua"
         "Per" "Ira" "Cos" "Ven" "Aus" "Chi" "Bol" ];
data = cname(data,cnam);
```

```
data = rname(data,rnam);
/* print "Data=", data; */


ctyp = [ 9#0 ];
sets = [ 5#1 4#2 ];
optn = [ "nfac"       2 ,
         "orig"         ,
         "maxit"    100 ,
         "ftol"   1.e-3 ,
         "plot"       3 ,
         "print"      3 ,
         "phis"         ];
< gof,eval,catq,load,scor > = overals(data,ctyp,sets,optn);
```

```
            ************************************
            Number of Observations for Class Levels
            ************************************

            Variable       Value    Nobs  Proportion
            C[1]               1       6   12.765957
                               2       7   14.893617
                               3       7   14.893617
                               4      12   25.531915
                               5      11   23.404255
                               6       4    8.510638

            C[2]               1       3    6.382979
                               2       3    6.382979
                               3      10   21.276596
                               4       6   12.765957
                               5      25   53.191489

            C[3]               1       2    4.255319
                               2      11   23.404255
                               3      17   36.170213
                               4       7   14.893617
                               5       7   14.893617
                        _Missing       3    6.382979

            C[4]               1       3    6.382979
                               2       5   10.638298
                               3       9   19.148936
                               4       9   19.148936
                               5       5   10.638298
                               6       9   19.148936
```

	7	6	12.765957
	8	1	2.127660
C[5]	1	10	21.276596
	2	11	23.404255
	3	14	29.787234
	4	11	23.404255
	5	1	2.127660
C[6]	1	3	6.382979
	2	2	4.255319
	3	3	6.382979
	4	4	8.510638
	5	22	46.808511
	6	13	27.659574
C[7]	1	6	12.765957
	2	28	59.574468
	3	13	27.659574
C[8]	1	18	38.297872
	2	15	31.914894
	3	5	10.638298
	4	6	12.765957
	5	3	6.382979
C[9]	1	16	34.042553
	2	12	25.531915
	3	19	40.425532

Summary on Data Set Variables
****

Var	Set	Type	N_Lev	Frequ	N_Miss	N_Nonz
1	1	single nominal	6	47	0	47
2	1	single nominal	5	47	0	47
3	1	single nominal	6	44	3	47
4	1	single nominal	8	47	0	47
5	1	single nominal	5	47	0	47
6	2	single nominal	6	47	0	47
7	2	single nominal	3	47	0	47
8	2	single nominal	5	47	0	47
9	2	single nominal	3	47	0	47

```
************************************
History of Iterations (Initial Fit=0)
************************************
```

Iteration	Crit	CritDiff	TotFit
1	0.86498901	1.01591603	1.13501099
2	0.41864204	0.44634697	1.58135796
3	0.28494251	0.13369953	1.71505749
4	0.22246591	0.06247660	1.77753409
5	0.18542246	0.03704345	1.81457754
6	0.16052930	0.02489316	1.83947070
7	0.14331404	0.01721526	1.85668596
8	0.13137447	0.01193957	1.86862553
9	0.12295429	0.00842018	1.87704571
10	0.11684820	0.00610609	1.88315180
11	0.11229099	0.00455721	1.88770901
12	0.10880092	0.00349006	1.89119908
13	0.10606587	0.00273505	1.89393413
14	0.10387591	0.00218996	1.89612409
15	0.10208512	0.00179080	1.89791488
16	0.10058941	0.00149570	1.89941059
17	0.09931332	0.00127609	1.90068668
18	0.09820158	0.00111174	1.90179842
19	0.09721357	9.880e-004	1.90278643
20	0.09631955	8.940e-004	1.90368045

Eigenvalues

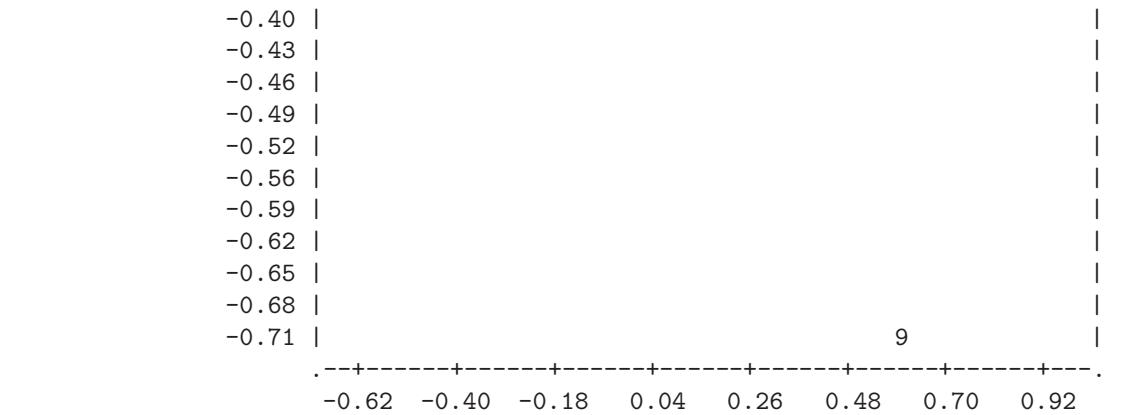
	1
Fac_1	0.962847624
Fac_2	0.933719189

```
*********************************************************************
*     P A R T I T I O N I N G   L O S S                              *
* FOR SINGLE VARIABLES : DISCRIMINATION MATRIX = SINGLE FIT         *
*  TOTAL LOSS = TOTAL DISPERSION - SINGLE FIT                       *
*  TOTAL LOSS = MULTIPLE LOSS + SINGLE LOSS                         *
*  MULTIPLE LOSS = TOTAL DISPERSION - MULTIPLE FIT                  *
*  SINGLE LOSS= MULTIPLE FIT - SINGLE FIT                           *
*  ------------------------------------------------------------     *
* FOR MULTIPLE VARIABLES : DISCRIMINATION MATRIX = MULTIPLE FIT     *
*  TOTAL LOSS = TOTAL DISPERSION - MULTIPLE FIT                     *
*********************************************************************
```

This first part is shown here only for the first variable:

```
*******************************************
Loss and Fit: Variable 1 Type: single nominal
*******************************************


     Discrimination Matrix

             Fac_1        Fac_2
Fac_1   0.043244867 -0.024692610
Fac_2  -0.024692610  0.014099362


        Total Dispersion

             Fac_1        Fac_2
Fac_1   0.346884825  0.014237564
Fac_2   0.014237564  0.139205296


          Total Loss

             Fac_1        Fac_2
Fac_1   0.303639958  0.038930174
Fac_2   0.038930174  0.125105934


          Multiple Fit

             Fac_1        Fac_2
Fac_1   0.043412400 -0.024429598
Fac_2  -0.024429598  0.014512266


         Multiple Loss

             Fac_1        Fac_2
Fac_1   0.303472425  0.038667162
Fac_2   0.038667162  0.124693029


          Single Loss

             Fac_1        Fac_2
Fac_1   0.000167533  0.000263012
```

```
Fac_2   0.000263012  0.000412905


                  ******************
                  Component Loadings
                  ******************

          Horizontal Variate: 1 Vertical Variate: 2
          ****************************************

      .--+------+------+------+------+------+------+------+---.
 0.68 |                                                       |
 0.65 |                                                       |
 0.62 |                                                       |
 0.59 |   5                                                   |
 0.56 |                                                       |
 0.53 |                                                       |
 0.49 |                                                       |
 0.46 |                                                       |
 0.43 |                                                       |
 0.40 |                  3                      6             |
 0.37 |             1                                         |
 0.33 |                           2                           |
 0.30 |                                                   7   |
 0.27 |                                                       |
 0.24 |                                             4         |
 0.21 |                                                       |
 0.18 |                                                       |
 0.14 |                                                       |
 0.11 |                                                       |
 0.08 |                                                       |
 0.05 |                                                       |
 0.02 |                                                       |
-0.01 |                                                       |
-0.05 |                                                       |
-0.08 |                                                       |
-0.11 |                                                       |
-0.14 |                                                       |
-0.17 |                                                       |
-0.21 |                                                       |
-0.24 |                                                       |
-0.27 |                                                       |
-0.30 |                                                       |
-0.33 |                                           8           |
-0.36 |                                                       |
```



Weights

	Fac_1	Fac_2
GINI	-0.207954001	0.118740732
FARM	0.318424520	0.477392684
RENT	0.010660347	0.433136579
GNPR	0.655843977	0.562908736
LABO	-0.372951564	0.669671295
INST	0.409735260	0.165918502
ECKS	0.550044485	0.680374512
DEAT	0.283220645	-0.281629106
DEMO	0.288868591	-0.819879477

Component Loadings for Single Variables

	Fac_1	Fac_2
GINI	-0.280036108	0.375176845
FARM	0.084625036	0.331467055
RENT	-0.122802874	0.387375045
GNPR	0.779020535	0.250786379
LABO	-0.623803555	0.579856020
INST	0.584138039	0.383450872
ECKS	0.886717170	0.316038147
DEAT	0.723605439	-0.348488426
DEMO	0.577408958	-0.721703844

****
Multiple Category Quantifications: Variable 1 Type: single nominal
****

Category Quantifications for Variable 1

	Fac_1	Fac_2
GINI_C_1	0.003725883	-0.015911973
GINI_C_2	-0.425645939	0.260763463
GINI_C_3	-0.127726193	0.034728299
GINI_C_4	0.105517976	-0.078350114
GINI_C_5	0.072021140	0.000654924
GINI_C_6	0.401470405	-0.207043850

Category Centroids

****

Cat	Frequ		
1	6	-1.05781071	-0.26983922
2	7	-0.30637873	0.84948227
3	7	-0.32885515	0.21792270
4	11	0.21625362	-0.20981717
5	10	0.64096493	-0.05083366
6	3	0.86107875	-0.52230792

****
Single Category Quantifications: Variable 1 Type: single nominal
****

Categ	Frequ	Quantification
1	6	-0.0488787195
2	7	2.0866326332
3	7	0.5283955936
4	11	-0.5480648770
5	10	-0.2524916380
6	3	-1.8807228401

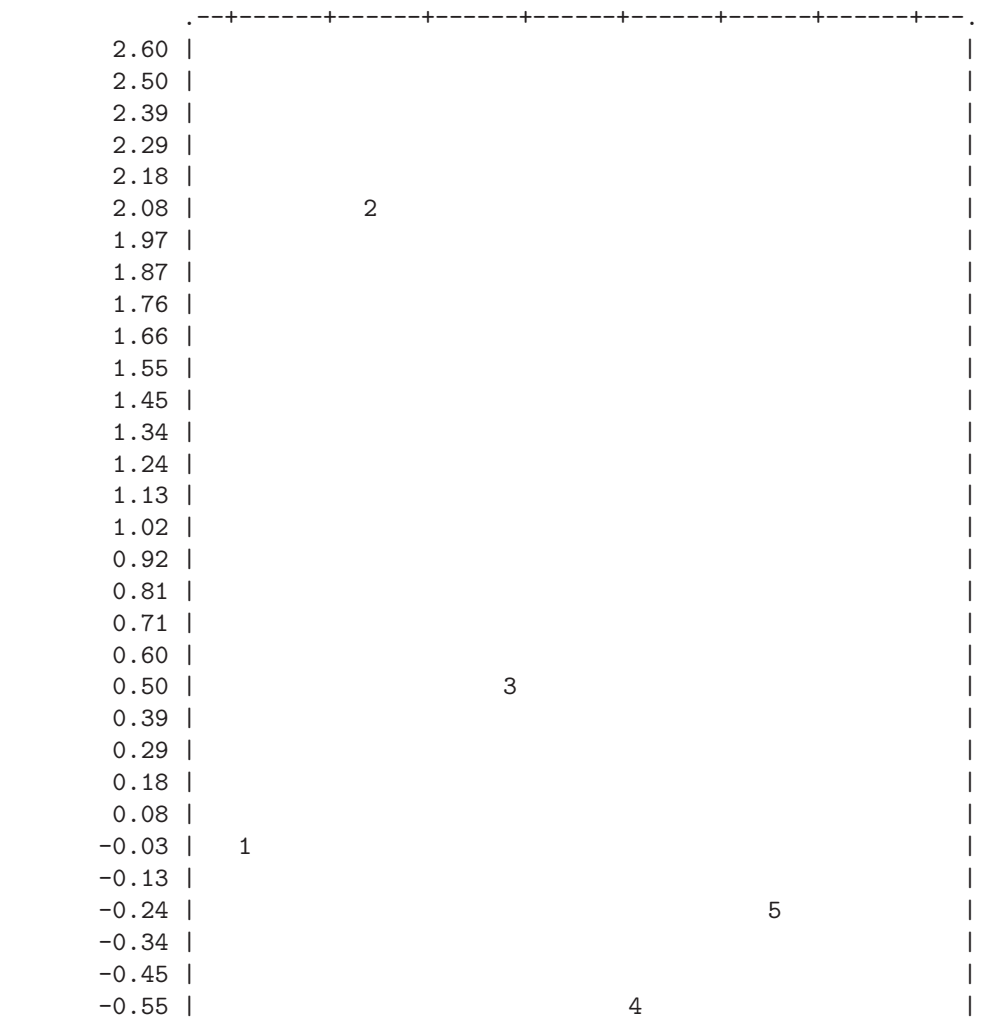
Rank One Category Quantifications

	Fac_1	Fac_2
GINI_C_1	0.010164525	-0.005803895
GINI_C_2	-0.433923605	0.247768287
GINI_C_3	-0.109881978	0.062742080
GINI_C_4	0.113972284	-0.065077625
GINI_C_5	0.052506646	-0.029981042
GINI_C_6	0.391103840	-0.223318408

Average Rank One Category Quantifications

	Fac_1	Fac_2
GINI_C_1	0.013687806	-0.018338164
GINI_C_2	-0.584332482	0.782856249
GINI_C_3	-0.147969846	0.198241792
GINI_C_4	0.153477955	-0.205621252
GINI_C_5	0.070706776	-0.094729016
GINI_C_6	0.526670305	-0.705603662

Cat. Numbers (hor.) vs. Cat. Quant. (ver.): Var=0 of SINGLE NOMINAL

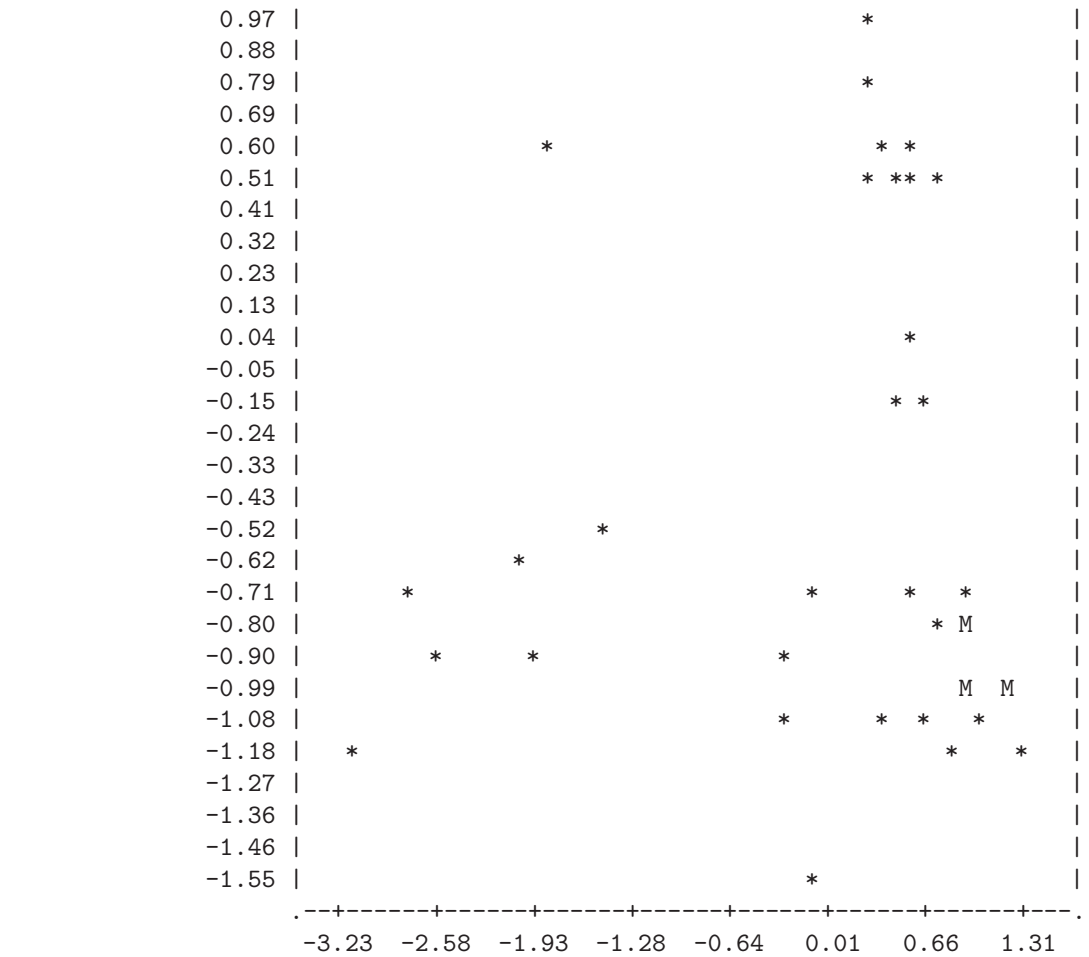


Object Scores

	Fac_1	Fac_2
Yug	-0.283278362	-0.906609568
Pol	-0.287814562	-1.101900504
Den	-1.527004252	-0.523824373
Jap	0.273256960	1.195166916
Can	-1.911524177	0.634497391
Swi	-2.610499860	-0.916365187
Ind	0.252108204	0.504179865
Phi	0.477878643	0.624205135
Swe	-2.786764484	-0.692361401
Fra	0.032907227	1.034321295
Bel	-0.130951473	1.505843178
Ire	-0.143557075	1.835055778
Fin	0.153727822	1.135132023
Net	-0.171255149	1.615666741
Lux	-1.962599188	-0.894582805
Tai	-0.098671794	-0.685550722
Nor	-0.000886608	1.695910145
SVi	-0.155620928	-1.525363356
WGe	-0.357181218	1.501652862
Lib	0.444228855	-0.182273952
USA	0.051338634	1.439214648
UnK	-0.261659573	1.237984424
Pan	0.859144549	-0.986288788
Aut	0.105011493	1.343101664
Egy	0.632283718	-1.070284957
Gre	0.558929413	0.066251429

```
Hon    0.796700599 -1.220412976
Nic    0.922746249 -0.770072798
NZe   -2.105351402 -0.592146824
Spa    0.558014959 -0.689447883
Cub    0.890139718 -0.715655324
DoR    0.294237663 -1.120304239
Ita    0.217531282  0.759228234
Uru    0.228500342  1.005781265
ElS    0.848466902 -0.800204728
Bra    0.546996594  0.464748128
Col    0.651145851 -0.115228748
Gua    0.911523060 -0.984469848
Arg    0.375981635  0.603814529
Ecua   0.736296047 -0.780532818
Per    1.149948421 -0.994964841
Ira    1.217011800 -1.188528352
Cos    0.676195739  0.527055771
Ven    1.130691952 -0.967376490
Aus   -3.229042355 -1.174062994
Chi    0.424280007  0.484488845
Bol    1.028264279 -1.084036110
```

```
             *************
             Object Scores
             *************

      Horizontal Variate: 1 Vertical Variate: 2
      ****************************************

     .--+------+------+------+------+------+------+------+---.
2.38 |                                                       |
2.28 |                                                       |
2.19 |                                                       |
2.10 |                                                       |
2.00 |                                                       |
1.91 |                                                       |
1.82 |                                  *                    |
1.72 |                                   *                   |
1.63 |                                  *                    |
1.54 |                                 * *                   |
1.44 |                                    *                  |
1.35 |                                    *                  |
1.26 |                                   *                   |
1.16 |                                       **              |
1.07 |                                    *                  |
```




```
print "GOF=", gof;
print "Eval=",eval;
print "Catq=",catq;
print "Load=",load;
print "Scor=",scor;
```

2. Fish data by Gifi (1990, p.102, table ...), $N = 24$, $n = 17$:

3.10 Function `primals`

< gof,eval,obscor,cquant,loada,scora,loadb,scorb > = primals(a,optn)

Purpose: The `primals` function performs one-dimensional homogeneity analysis and treats therefore a special application of the `homals` function.

Input: **data** This argument must be a $N \times n$ matrix containing the measured levels of n categorical variables. The levels can be integer or string coded. A sufficiently small number of missing values are permitted.

optn This argument must be specified in form of a two column matrix where the first column defines the option as string value (in quotes) and the second column can be used for a numeric or string specification of the option. See table below for content.

Option	Second Column	Meaning
`"eigmax"`		maximize largest eigenvalue (default)
`"eigmin"`		minimize smallest eigenvalue
`"maxit"`	int	maximum number of iterations (default is 200)
`"misest"`	int	substitute for missing data (default is 0) =0: substitute object scores =1: substitute 0. for missing data =2: substitute average of object scores
`"missol"`	int	=0: missing data passive categories (default is 0) =1: missing data multiple categories
`"normcat"`		normalize categories in centroid of objects
`"normobj"`		normalize objects in centroid of categories (valid only for `"eigmax"`)
`"phis"`		print optimization history
`"print"`	int	indicates the amount of printed output (=0: no printed output, default is 2)
`"plot"`	int	indicates the amount of plot output (=0: no plot output is default)
`"seed"`	int	seed for random generator (valid only when `"orig"` is not specified)
`"tolit"`	real	termination tolerance for final iteration (default is $5.e-4$)

Output: There are at most eight results returned:

gof a vector of scalar results, like an inidicator for the success of the iterations, the computer time used and the values of differently defined *stress* for the final solution.

eval a vector of n eigenvalues.

obscor the N vector of object scores

catqua the $Ncat$ vector of category quantifications, where $Ncat$ is the total number of categories of all n variables

loada the $n \times 2$ matrix of factor loadings after transformation.

scora the $N \times 2$ matrix of object scores after transformation.

loadb the $n \times 2$ matrix of factor loadings before transformation.

scorb the $N \times 2$ matrix of object scores before transformation.

Restrictions: 1. String data for levels are permitted.

2. Many missing values or difficult missing value patterns may affect the convergence of the iterative processes.

Relationships: princals(), homals(), anacor(), anaprof() canals(), overals(), pca(), svd()

Examples: 1. Small Artificial Data by Gifi (1990, p.65, table 2.1), $N = 10$, $n = 3$:

```
data = [  1   1   1,   2   2   2,   1   3   2,
          1   1   1,   2   1   2,   3   1   2,
          1   1   1,   1   1   2,   3   1   2,
          1   1   2 ];


print "Maximizing Largest Eigenvalue, Normalize Cats";
optn = [ "normcat"      ,
         "eigmax"       ,
         "maxit"    100 ,
         "tolit"  1.e-4 ,
         "plot"       2 ,
         "print"      3 ,
         "phis"         ];
< gof,eval,obscor,cquant,cloada,cscora,cloadb,cscorb > =
   primals(data,optn);
```

```
*************************************
Number of Observations for Class Levels
*************************************

Variable       Value    Nobs  Proportion
C[1]               1       6   60.000000
                   2       2   20.000000
                   3       2   20.000000

C[2]               1       8   80.000000
                   2       1   10.000000
                   3       1   10.000000

C[3]               1       3   30.000000
                   2       7   70.000000
```

Input Data Matrix: 10 by 3

	Col_1	Col_2	Col_3
Row_01	0.000000000	0.000000000	0.000000000
Row_02	1.000000000	1.000000000	1.000000000
Row_03	0.000000000	2.000000000	1.000000000
Row_04	0.000000000	0.000000000	0.000000000
Row_05	1.000000000	0.000000000	1.000000000
Row_06	2.000000000	0.000000000	1.000000000
Row_07	0.000000000	0.000000000	0.000000000
Row_08	0.000000000	0.000000000	1.000000000
Row_09	2.000000000	0.000000000	1.000000000
Row_10	0.000000000	0.000000000	1.000000000

```
*****************************************************
*** PRIMALS: One-Dimensional Homogeneity Analysis ***
*****************************************************

Number of Objects or Individuals . . . . . .         10
Number of Variables. . . . . .          3
Maximum Number of Categories . . . . . .          3
Total Number of Categories . . . . . .          8
Normalization: Categories in Centroid of Objects
Maximizing Largest Eigenvalue  . . . . . .          0
Maximum Number of Iterations . . . . . .        100
Missing Data Passive Categories (Deleted). . . . . .          0
Substitution for Missing Data (Object Scores). . . .          0
Output: Data Matrix. . . . . .(1,0,0)
Output: Object Scores. . . . . .(1,0,1)
Output: Category Quantifications . . . . . .(1,0,1)
Output: Transformed Data Matrix. . . . . .(1,0,0)
Output: Correlations bef. Transform. . . . . .(1,0,0)
Output: Correlations aft. Transform. . . . . .(1,0,0)
Output: Component Loadings bef. Transform. . . . . .(1,0,1)
Output: Component Loadings aft. Transform. . . . . .(1,0,1)
Output: Component Scores bef. Transform. . . . . .(1,0,1)
Output: Component Scores aft. Transform. . . . . .(1,0,1)
Plot: Comp. Loadings Bef. and Aft. Transform . . . .          0

          ****************************************
          History of Iterations (Initial Fit=2.74955)
          ****************************************

        Iteration         Fit      OldFit     FitDiff
                1  0.38033644  2.74954542 -2.36920898
```

2	0.57056689	0.38033644	0.19023046
3	0.60080825	0.57056689	0.03024136
4	0.61522703	0.60080825	0.01441878
5	0.62236021	0.61522703	0.00713318
6	0.62571906	0.62236021	0.00335886
7	0.62726153	0.62571906	0.00154247
8	0.62796271	0.62726153	7.012e-004
9	0.62828041	0.62796271	3.177e-004
10	0.62842433	0.62828041	1.439e-004
11	0.62848957	0.62842433	6.525e-005

Discrimination Measures for Variables

	1
Col_1	0.810511724
Col_2	0.645635002
Col_3	0.429396447

Object Scores

	1
Row_01	-1.000962059
Row_02	2.410021209
Row_03	-0.217962740
Row_04	-1.000962059
Row_05	0.982491392
Row_06	0.160712692
Row_07	-1.000962059
Row_08	-0.246544534
Row_09	0.160712692
Row_10	-0.246544534

Category Quantifications

****

Var	V_1_C_1	V_1_C_2	V_1_C_3
1	-0.61898966	1.69625630	0.16071269
Var	V_2_C_1	V_2_C_2	V_2_C_3
2	-0.27400731	2.41002121	-0.21796274
Var	V_3_C_1	V_3_C_2	
3	-1.00096206	0.42898374	

Eigenvalues

```
-----------

  1 :       1.518      1.138     0.3434
```

Component Scores Bef. Transformation

	1	2
Row_01	-1.392056542	-0.215800071
Row_02	0.867648134	0.725967828
Row_03	0.549118804	2.564041002
Row_04	-1.392056542	-0.215800071
Row_05	0.551880538	-0.528521723
Row_06	1.186177465	-1.112105346
Row_07	-1.392056542	-0.215800071
Row_08	-0.082416390	0.055061899
Row_09	1.186177465	-1.112105346
Row_10	-0.082416390	0.055061899

Component Loadings Bef. Transformation

	1	2
Col_1	0.770529817	-0.531371910
Col_2	0.307019772	0.914248985
Col_3	0.911314925	0.141274309

```
Eigenvalues
-----------

  1 :       1.886     0.7903     0.3241
```

Component Scores Aft. Transformation

	1	2
Row_01	-1.006982879	-1.118032589
Row_02	2.401849034	-1.501280330
Row_03	-0.214442342	0.859241053
Row_04	-1.006982879	-1.118032589
Row_05	0.983195687	0.593016833
Row_06	0.169238037	0.798588525
Row_07	-1.006982879	-1.118032589
Row_08	-0.244064909	0.902971580
Row_09	0.169238037	0.798588525
Row_10	-0.244064909	0.902971580

```
            Component Loadings Aft. Transformation

                                 1           2
               Col_1   0.899840440 -0.095251104
               Col_2   0.800808910 -0.495488895
               Col_3   0.659225038  0.731924138


print "Minimizing Smallest Eigenvalue, Normalize Cats";
optn = [ "nfac"       3 ,
         "normcat"      ,
         "eigmin"       ,
         "maxit"    100 ,
         "tolit"  1.e-4 ,
         "plot"       2 ,
         "print"      3 ,
         "phis"         ];
< gof,eval,obscor,cquant,cloada,cscora,cloadb,cscorb > =
   primals(data,optn);


          ******************************************
          History of Iterations (Initial Fit=-1.74955)
          ******************************************

         Iteration         Fit      OldFit     FitDiff
                1  0.24527792 -1.74954542  1.99482334
                2  0.11977793  0.24527792 -0.12549999
                3  0.07002848  0.11977793 -0.04974945
                4  0.05106089  0.07002848 -0.01896759
                5  0.04205932  0.05106089 -0.00900157
                6  0.03644000  0.04205932 -0.00561932
                7  0.03211163  0.03644000 -0.00432837
                8  0.02837937  0.03211163 -0.00373226
                9  0.02501339  0.02837937 -0.00336598
               10  0.02194297  0.02501339 -0.00307042
               11  0.01914894  0.02194297 -0.00279403
               12  0.01662517  0.01914894 -0.00252377
               13  0.01436532  0.01662517 -0.00225985
               14  0.01235918  0.01436532 -0.00200614
               15  0.01059241  0.01235918 -0.00176677
               16  0.00904755  0.01059241 -0.00154486
               17  0.00770522  0.00904755 -0.00134233
               18  0.00654528  0.00770522 -0.00115994
               19  0.00554771  0.00654528 -9.976e-004
               20  0.00469329  0.00554771 -8.544e-004
               21  0.00396405  0.00469329 -7.292e-004
```

```
   22  0.00334352  0.00396405 -6.205e-004
   23  0.00281683  0.00334352 -5.267e-004
   24  0.00237076  0.00281683 -4.461e-004
   25  0.00199367  0.00237076 -3.771e-004
   26  0.00167537  0.00199367 -3.183e-004
   27  0.00140705  0.00167537 -2.683e-004
   28  0.00118112  0.00140705 -2.259e-004
   29  9.910e-004  0.00118112 -1.901e-004
   30  8.313e-004  9.910e-004 -1.598e-004
   31  6.970e-004  8.313e-004 -1.342e-004
   32  5.843e-004  6.970e-004 -1.127e-004
   33  4.898e-004  5.843e-004 -9.458e-005
```

Discrimination Measures for Variables

```
                        1
        Col_1   0.000646904
        Col_2   0.000355369
        Col_3   0.000283413
```

Object Scores

```
                         1
        Row_01   1.447781657
        Row_02  -0.043347351
        Row_03   0.040913972
        Row_04  -1.965176655
        Row_05   0.130445260
        Row_06   1.294426200
        Row_07   0.594542079
        Row_08  -0.543766103
        Row_09  -1.265292534
        Row_10   0.309473475
```

Category Quantifications

****

```
 Var     V_1_C_1     V_1_C_2     V_1_C_3
   1 -0.01937193  0.04354895  0.01456683
 Var     V_2_C_1     V_2_C_2     V_2_C_3
   2  3.042e-004 -0.04334735  0.04091397
 Var     V_3_C_1     V_3_C_2
   3  0.02571569 -0.01102101
```

```
Eigenvalues
-----------

  1 :       1.755     0.9895      0.255


              Component Scores Aft. Transformation

                                 1            2
                Row_01  -0.956188293  1.135356362
                Row_02   2.062831185  1.047383443
                Row_03  -1.038851567 -1.943052180
                Row_04  -0.956188293  1.135356362
                Row_05   1.137572523 -0.499694604
                Row_06   0.531573129 -0.501575279
                Row_07  -0.956188293  1.135356362
                Row_08  -0.178066761 -0.503777594
                Row_09   0.531573129 -0.501575279
                Row_10  -0.178066761 -0.503777594


             Component Loadings Aft. Transformation

                                 1            2
                Col_1   0.933597797  0.001633138
                Col_2  -0.701457288 -0.661107461
                Col_3  -0.625972176  0.743265210


print "Maximizing Largest Eigenvalue, Normalize Objs";
optn = [ "nfac"       3 ,
         "normobj"      ,
         "eigmax"       ,
         "maxit"    100 ,
         "tolit"  1.e-4 ,
         "plot"       2 ,
         "print"      3 ,
         "phis"         ];
< gof,eval,obscor,cquant,cloada,cscora,cloadb,cscorb > =
   primals(data,optn);


          ****************************************
          History of Iterations (Initial Fit=1.49778)
          ****************************************

         Iteration         Fit      OldFit     FitDiff
```

1	0.36379645	1.49777613	-1.13397968
2	0.42010282	0.36379645	0.05630638
3	0.43006756	0.42010282	0.00996473
4	0.43707721	0.43006756	0.00700965
5	0.44949061	0.43707721	0.01241340
6	0.47144032	0.44949061	0.02194972
7	0.50437140	0.47144032	0.03293107
8	0.54313904	0.50437140	0.03876764
9	0.57764849	0.54313904	0.03450945
10	0.60147151	0.57764849	0.02382302
11	0.61514710	0.60147151	0.01367559
12	0.62217103	0.61514710	0.00702393
13	0.62557156	0.62217103	0.00340052
14	0.62717057	0.62557156	0.00159901
15	0.62791215	0.62717057	7.416e-004
16	0.62825388	0.62791215	3.417e-004
17	0.62841089	0.62825388	1.570e-004
18	0.62848293	0.62841089	7.204e-005

Discrimination Measures for Objects

	1
Row_01	0.632539786
Row_02	3.644083065
Row_03	0.028366998
Row_04	0.632539786
Row_05	0.605729151
Row_06	0.016571031
Row_07	0.632539786
Row_08	0.038082562
Row_09	0.016571031
Row_10	0.038082562

Object Scores

	1
Row_01	-0.795323699
Row_02	1.908948157
Row_03	-0.168425050
Row_04	-0.795323699
Row_05	0.778286034
Row_06	0.128728517
Row_07	-0.795323699
Row_08	-0.195147539
Row_09	0.128728517

```
                    Row_10  -0.195147539


                  Category Quantifications
                  ************************

              Var     V_1_C_1     V_1_C_2     V_1_C_3
                1 -0.77838578  2.14191494  0.19324239
              Var     V_2_C_1     V_2_C_2     V_2_C_3
                2 -0.34721538  3.04477098 -0.26704792
              Var     V_3_C_1     V_3_C_2
                3 -1.26036993  0.54015854


Eigenvalues
-----------

  1 :       1.886       0.79     0.3245


             Component Scores Aft. Transformation

                              1           2
              Row_01  -1.006150644 -1.119191852
              Row_02   2.403810040 -1.498075857
              Row_03  -0.210282389  0.855402500
              Row_04  -1.006150644 -1.119191852
              Row_05   0.983563213  0.590391879
              Row_06   0.164529566  0.800166394
              Row_07  -1.006150644 -1.119191852
              Row_08  -0.243849031  0.904762123
              Row_09   0.164529566  0.800166394
              Row_10  -0.243849031  0.904762123


            Component Loadings Aft. Transformation

                              1           2
               Col_1   0.899607728 -0.096532131
               Col_2   0.801496506 -0.493779792
               Col_3   0.658680213  0.732683054
```

3.11 Function sdd

< gof,D,X,Y,Rho > = sdd(data,nfac<,optn<,wgt>>)

Purpose: The `sdd` function performs unweigthed and weighted SemiDiscrete decomposition of a real m *timesn* matrix $\mathbf{A}$. For a specified number K of factors the unweighted model (Kolda & O'Leary, 1999) is

$$\mathbf{A} = \sum_{k=1}^{K} x_k * d_k * y_k^T + e_k$$

where the entries of $x_k \in \mathcal{S}^m$ and $y_k \in \mathcal{S}^n$, $\mathcal{S} = -1, 0, 1$, consist only of values $-1, 0, 1$ and the d_k are positive scalars.

The SemiDiscrete decomposition (sdd) is a discrete analogon to the singular value decomposition (svd). The algorithm computes stepwise for $k = 1, \ldots, K$ the $m \times n$ matrices $\mathbf{A}_k$

$$\mathbf{A}_k = \sum_{l=1}^{k} x_l * d_l * y_l^T = \mathbf{X}_k * \mathbf{D}_k \mathbf{Y}_k^T$$

by fitting the residual matrices $\mathbf{R}_k = \mathbf{A} - \mathbf{A}_{k-1}$

$$\min F_k(d, x, y) = \|\mathbf{R}_k - dxy^T\|_F^2$$

for $x \in \mathcal{S}^m$, $y \in \mathcal{S}^n$, and $d > 0$ using an iterative algorithm of *mixed integer programming*. Since

$$\|\mathbf{R}_{k+1}\|_F^2 = \|\mathbf{R}_k\|_F^2 - \frac{x_k^T \mathbf{R}_k y_k}{\|x\|_2^2 \|y\|_2^2}$$

and with $\rho_k = \|\mathbf{R}_{k+1}\|_F^2$ and $\beta_k = \frac{x_k^T \mathbf{R}_k y_k}{\|x\|_2^2 \|y\|_2^2}$

$$\rho_k = \rho_{k-1} - \beta_k$$

the iteration stops when β_k no longer changes (becomes nearly constant). For the weighted model see Kolda & O'Leary (1999b).

Note: For sparse input data the unweighted algorithm works with sparse data which can save memory when $K \ll \min(m, n)$. The weighted algorithm always works on dense data.

Input: data The $M \times N$ input data set should not contain string or complex data and no missing values.

nfac The number of factors (dimensions, components).

optn This argument must be specified in form of a two column matrix where the first column defines the option as string value (in quotes) and the second column can be used for a numeric or string specification of the option. See table below for content.

wgt The optional $M \times N$ weight matrix set should contain only nonnegative entries.

Options Matrix Argument: The option argument is specified in form of a two column matrix:

Option Name	Second Column	Meaning
`"init"`	string	the method for obtaining initial values
	"thr"	threshold method (default)
	"cyc"	cycling method
	"one"	set all entries equal to one
	"spl"	periodic ones
`"maxit"`	int	the maximum number of iterations for each factor
`"rhomin"`	real	terminate generating further dimensions when ρ becames smaller than *rhomin*
`tolit`	real	termination criterion for iteration w.r.t. relative change in β

Output: **gof** some scalar results (see output of examples)

D the K column vector of positive diagonal entries

X the $m \times K$ matrix $\mathbf{X}$ with entries in S

Y the $n \times K$ matrix $\mathbf{Y}$ with entries in S

Rho the K column vector of positive values ρ_k (Frobenius norm of residual matrices $\mathbf{R}_k$)

Note, that the output and input values of K must not be the same when *rhomin* is specified large enough for an early termination of generating factors.

Restrictions:

1. No missing values are permitted in input data or weight matrix.
2. The data and weight matrix must be real and cannot have string or complex entries.
3. The weight matrix must have the same dimension as the input data set and should contain only nonnegative values.

Relationships: nnmf(), fica(), svd(), pca(), factor()

Examples: 1. 5×5 matrix from *MatrixMarket* see Kolda & O'Leary (1999):

```
spd = [   1    1    5.827917e-01,  2    1    4.234963e-01,
          3    1    5.155118e-01,  4    1    3.339515e-01,
          5    1    4.329066e-01,  1    2    2.259499e-01,
          2    2    5.798069e-01,  3    2    7.603650e-01,
          4    2    5.298231e-01,  5    2    6.405265e-01,
          1    3    2.090694e-01,  2    3    3.798184e-01,
          3    3    7.833286e-01,  4    3    6.808458e-01,
          5    3    4.610951e-01,  1    4    5.678287e-01,
          2    4    7.942107e-01,  3    4    5.918259e-02,
          4    4    6.028691e-01,  5    4    5.026880e-02,
```

```
 1    5    4.153749e-01,  2    5    3.049987e-01,
 3    5    8.743672e-01,  4    5    1.500950e-02,
 5    5    7.679504e-01 ];


nr = nc = 5;
data = spmat(nr,nc,spd[,1],spd[,2],spd[,3]);
cnam = [ "c1" : "c5" ] ;
data = cname(data,cnam);
print "Data=", data;
```

(a) UNWeighted Application: `rhomin = 0.`

```
optn = [ "print"           3 ,
         "init"        "thr" ,
         "rhomin"        0. ,
         "tolit"       1.e-4 ,
         "maxit"          30 ];
< gof,d,x,y,rho > = sdd(data,10,optn);
```

```
                  ************************************
                  Standard Semi Discrete Decomposition
                  ************************************

    Number Rows of Matrix . . . . . .          5
    Number Columns of Matrix  . . . . .        5
    Number Components (Factors) . . . . .     10
    Initial Method. . . . . .           Threshold
    Rhomin Parameter. . . . . .           0.0000
    Maximum Number Iterations . . . . .       30
    Iteration Tolerance (Precision) . . . .   0.0001

                       Input Data Matrix
                       *****************

                    Matrix: Dense Storage

      |          1         2         3         4         5
    ---------------------------------------------------------
    1 |   0.5827917 0.2259499 0.2090694 0.5678287 0.4153749
    2 |   0.4234963 0.5798069 0.3798184 0.7942107 0.3049987
    3 |   0.5155118 0.7603650 0.7833286 0.0591826 0.8743672
    4 |   0.3339515 0.5298231 0.6808458 0.6028691 0.0150095
    5 |   0.4329066 0.6405265 0.4610951 0.0502688 0.7679504

                  ****************************
```

```
         SDD Iteration for Dimension 1
         ****************************

Iteration      AbsDiff      RelDiff            D
        1  5.751696297  0.152566649  0.479653888
        2  5.751696297  0.000000000  0.479653888

         ****************************
         SDD Iteration for Dimension 2
         ****************************

Iteration      AbsDiff      RelDiff            D
        1  0.851471922  0.249469568  0.326242226
        2  0.851471922  0.000000000  0.326242226

         ****************************
         SDD Iteration for Dimension 3
         ****************************

Iteration      AbsDiff      RelDiff            D
        1  0.338516965  0.290440862  0.193940702
        2  0.338516965  0.000000000  0.193940702

         ****************************
         SDD Iteration for Dimension 4
         ****************************

Iteration      AbsDiff      RelDiff            D
        1  0.086034185  0.000000000  0.084672991

         ****************************
         SDD Iteration for Dimension 5
         ****************************

Iteration      AbsDiff      RelDiff            D
        1  0.031007883  0.000000000  0.101665929

         ****************************
         SDD Iteration for Dimension 6
         ****************************

Iteration      AbsDiff      RelDiff            D
        1  0.044970089  0.290312719  0.070687174
        2  0.051803544  0.151955555  0.058767079
        3  0.051803544  0.000000000  0.058767079
```

```
            ****************************
            SDD Iteration for Dimension 7
            ****************************

     Iteration      AbsDiff      RelDiff            D
             1  0.024015511  0.000000000  0.038742346

            ****************************
            SDD Iteration for Dimension 8
            ****************************

     Iteration      AbsDiff      RelDiff            D
             1  0.012635886  0.684476759  0.032449815
             2  0.012635886  0.000000000  0.032449815

            ****************************
            SDD Iteration for Dimension 9
            ****************************

     Iteration      AbsDiff      RelDiff            D
             1  0.006062158  0.000000000  0.055055237

            ****************************
            SDD Iteration for Dimension 10
            ****************************

     Iteration      AbsDiff      RelDiff            D
             1  0.002253059  0.000000000  0.023733201

                  Diagonal Matrix D
                  *****************

            Dense Column Vector (nrow=10)

C |      Fac_01     Fac_02     Fac_03     Fac_04     Fac_05
      0.4796539  0.3262422  0.1939407  0.0846730  0.1016659

C |      Fac_06     Fac_07     Fac_08     Fac_09     Fac_10
      0.0587671  0.0387423  0.0324498  0.0550552  0.0237332

                Left Discrete Matrix X
                **********************

                Dense Matrix (5 by 10)
```

	Fac_01	Fac_02	Fac_03	Fac_04	Fac_05
Row_1	1.0000000	0.0000000	-1.0000000	-1.0000000	1.0000000
Row_2	1.0000000	1.0000000	0.0000000	1.0000000	-1.0000000
Row_3	1.0000000	-1.0000000	1.0000000	1.0000000	0.0000000
Row_4	1.0000000	1.0000000	1.0000000	-1.0000000	-1.0000000
Row_5	1.0000000	-1.0000000	0.0000000	0.0000000	0.0000000

	Fac_06	Fac_07	Fac_08	Fac_09	Fac_10
Row_1	1.0000000	-1.0000000	0.0000000	0.0000000	1.0000000
Row_2	1.0000000	-1.0000000	-1.0000000	0.0000000	-1.0000000
Row_3	-1.0000000	1.0000000	-1.0000000	0.0000000	-1.0000000
Row_4	-1.0000000	0.0000000	1.0000000	0.0000000	-1.0000000
Row_5	1.0000000	1.0000000	0.0000000	1.0000000	0.0000000

Right Discrete Matrix Y

****

Dense Matrix (5 by 10)

	Fac_01	Fac_02	Fac_03	Fac_04	Fac_05
c1	1.0000000	0.0000000	0.0000000	0.0000000	1.0000000
c2	1.0000000	0.0000000	1.0000000	1.0000000	0.0000000
c3	1.0000000	0.0000000	1.0000000	0.0000000	0.0000000
c4	1.0000000	1.0000000	-1.0000000	1.0000000	0.0000000
c5	1.0000000	-1.0000000	0.0000000	1.0000000	0.0000000

	Fac_06	Fac_07	Fac_08	Fac_09	Fac_10
c1	0.0000000	0.0000000	-1.0000000	-1.0000000	0.0000000
c2	1.0000000	1.0000000	0.0000000	1.0000000	0.0000000
c3	-1.0000000	1.0000000	-1.0000000	0.0000000	1.0000000
c4	-1.0000000	-1.0000000	1.0000000	0.0000000	0.0000000
c5	0.0000000	-1.0000000	-1.0000000	0.0000000	0.0000000

Rho Values

****

Dense Column Vector (nrow=10)

C	Fac_01	Fac_02	Fac_03	Fac_04	Fac_05
	1.4088220	0.5573500	0.2188331	0.1327989	0.1017910

```
     C |      Fac_06     Fac_07     Fac_08     Fac_09     Fac_10
           0.0499875  0.0259720  0.0133361  0.0072739  0.0050209
```

```
GOF=
             |         1
-------------------------
     Failure |    0.0000
        Time |    0.0000
      TotFit |    0.0050
   Dimension |   10.0000
  Iterations |   26.0000
      unused |    0.0000
      unused |    0.0000
      unused |    0.0000
      unused |    0.0000
      unused |    0.0000
```

(b) UNWeighted Application: `rhomin = 0.01`

```
  optn = [ "print"           3 ,
           "init"        "thr" ,
           "rhomin"        .01 ,
           "tolit"       1.e-4 ,
           "maxit"          30 ];
 < gof,d,x,y,rho > = sdd(data,10,optn);
```

```
                  ****************************
                  SDD Iteration for Dimension 9
                  ****************************

          Iteration      AbsDiff      RelDiff            D
                  1  0.006062158  0.000000000  0.055055237

     Iteration terminated: rho=0.00727391 smaller rhomin=0.01

                        Diagonal Matrix D
                        *****************

                   Dense Column Vector (nrow=9)

     C |      Fac_01     Fac_02     Fac_03     Fac_04     Fac_05
           0.4796539  0.3262422  0.1939407  0.0846730  0.1016659

     C |      Fac_06     Fac_07     Fac_08     Fac_09
           0.0587671  0.0387423  0.0324498  0.0550552
```

```
                          Rho Values
                          **********

                    Dense Column Vector (nrow=9)

     C |      Fac_01     Fac_02     Fac_03     Fac_04     Fac_05
           1.4088220  0.5573500  0.2188331  0.1327989  0.1017910

     C |      Fac_06     Fac_07     Fac_08     Fac_09
           0.0499875  0.0259720  0.0133361  0.0072739

GOF=
             |         1
-------------------------
     Failure |    1.0000
        Time |    0.0000
      TotFit |    0.0073
   Dimension |    9.0000
  Iterations |   24.0000
      unused |    0.0000
      unused |    0.0000
      unused |    0.0000
      unused |    0.0000
      unused |    0.0000
```

(c) Weighted Application: `rhomin = 0.`

```
  /*--- Weighted Application ----*/
  wgt = cons(nr,nc,1.) - ide(nr);
  optn = [ "print"           3 ,
           "init"        "thr" ,
           "rhomin"         0. ,
           "tolit"       1.e-4 ,
           "maxit"          30 ];
 < gof,d,x,y,rho > = sdd(data,5,optn,wgt);

                 ***********************************
                 Weighted Semi Discrete Decomposition
                 ***********************************

    Number Rows of Matrix . . . . . .          5
    Number Columns of Matrix  . . . . .        5
    Number Components (Factors) . . . . .      5
    Initial Method. . . . . .          Threshold
    Rhomin Parameter. . . . . .           0.0000
```

```
Maximum Number Iterations . . . . . .        30
Iteration Tolerance (Precision) . . . . . .    0.0001
```

Input Data Matrix

****

Matrix: Dense Storage

	1	2	3	4	5
1	0.5827917	0.2259499	0.2090694	0.5678287	0.4153749
2	0.4234963	0.5798069	0.3798184	0.7942107	0.3049987
3	0.5155118	0.7603650	0.7833286	0.0591826	0.8743672
4	0.3339515	0.5298231	0.6808458	0.6028691	0.0150095
5	0.4329066	0.6405265	0.4610951	0.0502688	0.7679504

Input Weight Matrix

****

Dense Matrix wgt

	1	2	3	4	5
1	0.0000000	1.0000000	1.0000000	1.0000000	1.0000000
2	1.0000000	0.0000000	1.0000000	1.0000000	1.0000000
3	1.0000000	1.0000000	0.0000000	1.0000000	1.0000000
4	1.0000000	1.0000000	1.0000000	0.0000000	1.0000000
5	1.0000000	1.0000000	1.0000000	1.0000000	0.0000000

****

SDD Iteration for Dimension 1

****

Iteration	AbsDiff	RelDiff	D
1	3.762434683	0.367971357	0.433730024
2	3.762434683	0.000000000	0.433730024

****

SDD Iteration for Dimension 2

****

Iteration	AbsDiff	RelDiff	D
1	0.739612064	0.049280937	0.351096393
2	0.739612064	0.000000000	0.351096393

```
                ****************************
                SDD Iteration for Dimension 3
                ****************************

      Iteration      AbsDiff      RelDiff            D
              1  0.255155464  0.293209415  0.190920875
              2  0.255155464  0.000000000  0.190920875

                ****************************
                SDD Iteration for Dimension 4
                ****************************

      Iteration      AbsDiff      RelDiff            D
              1  0.026750500  0.000000000  0.163555800

                ****************************
                SDD Iteration for Dimension 5
                ****************************

      Iteration      AbsDiff      RelDiff            D
              1  0.049446224  0.000000000  0.222365069

                     Diagonal Matrix D
                     *****************

                Dense Column Vector (nrow=5)

  C |       Fac_1      Fac_2      Fac_3      Fac_4      Fac_5
        0.4337300  0.3510964  0.1909209  0.1635558  0.2223651

                    Left Discrete Matrix X
                    **********************

                    Dense Matrix (5 by 5)

      |       Fac_1      Fac_2      Fac_3      Fac_4      Fac_5
    ------------------------------------------------------------
Row_1 |   1.0000000  0.0000000 -1.0000000  0.0000000  0.0000000
Row_2 |   1.0000000  1.0000000  0.0000000  0.0000000  1.0000000
Row_3 |   1.0000000 -1.0000000  1.0000000  0.0000000  0.0000000
Row_4 |   1.0000000  1.0000000  1.0000000  0.0000000  0.0000000
Row_5 |   1.0000000 -1.0000000  1.0000000 -1.0000000  0.0000000

                   Right Discrete Matrix Y
                   ***********************
```

```
                    Dense Matrix (5 by 5)

      |       Fac_1      Fac_2      Fac_3      Fac_4      Fac_5
 -----------------------------------------------------------------
   c1 |   1.0000000  0.0000000  0.0000000  0.0000000  0.0000000
   c2 |   1.0000000  0.0000000  1.0000000  0.0000000  0.0000000
   c3 |   1.0000000  0.0000000  1.0000000  1.0000000  0.0000000
   c4 |   1.0000000  1.0000000  0.0000000  0.0000000  0.0000000
   c5 |   1.0000000 -1.0000000  0.0000000  0.0000000  1.0000000

                          Rho Values
                          **********

                 Dense Column Vector (nrow=5)

   C |       Fac_1      Fac_2      Fac_3      Fac_4      Fac_5
         1.1554587  0.4158466  0.1606912  0.1339407  0.0844945

GOF=
             |         1
-------------------------
     Failure |    0.0000
        Time |    0.0000
      TotFit |    0.0845
   Dimension |    5.0000
  Iterations |   13.0000
      unused |    0.0000
      unused |    0.0000
      unused |    0.0000
      unused |    0.0000
      unused |    0.0000
```

3.12 Function sgmanova

< gof,parm,shape,ase,cov > = sgmanova(data,modl<,class<,optn>>)

Purpose: The sgmanova function performs robust multivariate analysis of variance based on spatial signs. The classical MANOVA model is for a $N \times p$ data matrix $\mathbf{X} = (x_{ij})$,

$$x_{ij} = \mu_j + \epsilon_{ij} \quad i = 1, \ldots, N, j = 1, \ldots, p$$

For the one-sample case, $H_0 : \mu = 0$, the Hotelling T^2 test is used,

$$T^2 = N\bar{x}^T \mathbf{S}^{-1} \bar{x}$$

where $\frac{N-p}{Np}T^2$ has a $F_{p,N-p}$ distribution.

For the multisample case, $H_0 : \mu_1 = \ldots = \mu_c$, the Hotelling's trace statistic is used,

$$T^2 = (N-c)trace(\mathbf{BW}^-1)$$

where **B** is the between samples sums of squares matrix and **W** is the within samples sums of squares matrix. Under H_0 the test statistics is $\chi^2_{p(c-1)}$ distributed.

The estimation is based on estimating the scatter matrix **V** by Tyler (1987). For more details see Nevalainen & Oja (2006).

Input: data specifies the data with N rows and a set of n columns which can be defined as effects using the model string argument.

modl specifies the p effects of the MANOVA model.

class specifies which of the columns of the input data matrix **X** are nominally scaled CLASS variables. (This argument can be missing value.)

optn This argument must be specified in form of a two column matrix where the first column defines the option as string value (in quotes) and the second column can be used for a numeric or string specification of the option. See table below for content.

Option	Second Column	Meaning
"cov"	string	compute covariance matrix or matrices of estimates and standard errors
	"asym"	based on asymptotic assumptions
	"boot"	based on Bootstrap
	"jack"	based on Jackknife
"maxit"	int	maximum number of iterations (def=400)
"nboot"	int	number bootstrap samples (def=500)
"nopr"		no printed output
"perm"	int	number permutations for sign test (def=1000)
"print"	int	amount of printed output (def=2)
"seed"	int	seed for random generator (global default)
"setvar"	int	column number specifying sample number for multisample analysis (def¡0, one sample)
"samp"	int	number bootstrap samples (def=500)
"xtol"	real	termination criterion (def=1.e-6)

For multi sample estimation, the input data must contain one column containing ints specifying the sample number. The number of this column must be specified with the "stevar" option. For standard errors and/or covariance matrices of estimates, the "cov" option must be specified. Note, that Jackknife is implemented only for the one sample case.

Output: gof This is a vector of scalar data, containing the fit statistics, containing

1. the $Q2$ test statistics,
2. the p value based on the limiting distribution,
3. the p value of the conditionally distribution-free permutation test,
4. and its standard error (from the binomial distribution of the signs).

parm For the one-sample case this is a p vector of the parameter estimates μ. For the multisample case this is a $c \times p$ matrix of the c row vectors μ_j.

shape This is the $p \times p$ symmetric *shape* matrix $\mathbf{V}$, see Tyler (1987).

ase For the one-sample case this is a p vector of the standard error estimates se. For the multisample case this is a $c \times p$ matrix of the c row vectors se_j.

cov For the one-sample case this is the $p \times p$ symmetric covariance matrix $\mathbf{C}$. For the multisample case this is a $cp \times p$ matrix of the c rowwise stacked $p \times p$ covariance matrices $\mathbf{C}_j$.

Restrictions:
1. The data must not contain missing values.
2. The data should be int or real, but CLASS variables may be string.

Relationships: anova()

Examples: 1. One sample application with 100×3 matrix of normal random generated data:

```
print "Random Generated: nobs=100, nvar=3";
srand(1);
x1 = rand(100,3,'g',"norm");


/* define desired covariance matrix */
sc = cons(3,3,1.);
sc[2,2] = sc[3,3] = 3.;
< eval,evec > = eig(sc);
sh = evec * sqrt(diag(eval)) * evec';
/* introduce the covariance structure into the data set */
/* [1] mu(dat1) = (0,0,0) */
dat1 = x1 * sh;
/* [2] mu(set1(dat2)) = (0,0,0);
       mu(set2(dat2)) = (1,1,1) */
x0 = cons(50,3,0.) |> cons(50,3,1.);
set = [ 50#1 50#2 ];
dat2 = set -> (dat1 + x0);
```

```
/* [1] Asymptotic: mu(dat1) = (0,0,0) */
modl = " 1:3 ";
optn = [ "print"      3 ,
         "xtol"   1.e-9 ,
         "perm"    1000 ,
         "cov"   "asym" ];
< gof,parm,shap,ase,covm > = sgmanova(dat1,modl,.,optn);


print "GOF=", gof;
pase = parm‘ -> ase‘;
cnam = [ "Estimate" "StdErr" ];
pase = cname(pase,cnam);
print "Parm=", pase;
print "COVM=",covm;
```

****
Model Information
****

Number Valid Observations 100
Number Response Variables 0
N Independend Variables 3
Significance Level: 0.0500000
Design Coding: Rank-Deficient

****
Model Effects
****

X1 + X2 + X3

****
Simple Statistics
****

Column	Nobs	Mean	Std Dev	Skewness	Kurtosis
X[1]	100	-0.1524848	1.0072575	0.0032078	0.0193308
X[2]	100	-0.1837237	1.8836723	-0.0637110	0.2525209
X[3]	100	-0.0141710	1.6074002	0.0154456	-0.1817052

****
Parameter Information
****

Parameter	Meaning
1	X1
2	X2
3	X3

****
Iterate Parameter Vector and Covariance Matrix
****

Iteration	F_parm	F_covm
1	0.002895709	0.787140492
2	0.000528193	0.088551751
3	0.000242354	0.018838679
4	8.5131e-005	0.004223412
5	2.3503e-005	0.000946418
6	5.7194e-006	0.000211326
7	1.3058e-006	4.7040e-005
8	2.8855e-007	1.0446e-005
9	6.2712e-008	2.3156e-006
10	1.3516e-008	5.1277e-007
11	2.9010e-009	1.1349e-007
12	6.2146e-010	2.5119e-008
13	1.3299e-010	5.5616e-009
14	2.8437e-011	1.2322e-009
15	6.0752e-012	2.7325e-010

Successful estimation after 15 iterations wrt. termination criterion 1e-009.

****
Iterate Covariance Matrix for Sign Test
****

Iteration	F_covm
1	0.732058891
2	0.080385362
3	0.016710788
4	0.003792285
5	0.000872202
6	0.000201543
7	4.6766e-005
8	1.0903e-005
9	2.5550e-006

10	6.0203e-007
11	1.4264e-007
12	3.3978e-008
13	8.1354e-009
14	1.9573e-009
15	4.7301e-010

Successful estimation after 15 iterations wrt. termination criterion 1e-009.

Parameter Estimates and Asympt. Standard Errors

	Estimate	AStdErr
VAR_1	-0.169911314	0.110774197
VAR_2	-0.190354819	0.209159238
VAR_3	0.011283515	0.174701194

Covariance Matrix and Standard Errors Computed by Asympt. Theory

Test Statistics Q2	3.617311535
Large sample p-value	0.305862890
Estimated p-value from sign change distribution. .	0.308000000
Precision of p-value from binomial distribution. .	0.014599178

Shape Matrix

	VAR_1	VAR_2	VAR_3
VAR_1	0.425389258	0.480977242	0.417240187
VAR_2	0.480977242	1.516573279	0.470591433
VAR_3	0.417240187	0.470591433	1.058037463

Asymptotic Covariance Matrix

	VAR_1	VAR_2	VAR_3
VAR_1	0.012270923	0.013874433	0.012035852
VAR_2	0.013874433	0.043747587	0.013574840
VAR_3	0.012035852	0.013574840	0.030520507

GOF=

	1
Failure	0.0000
Iterat.	15.0000

```
      Time |    0.0000
        Q2 |    3.6173
      P_as |    0.3059
     P_prm |    0.3080
      SE_P |    0.0146
    unused |     .
    unused |     .
    unused |     .


Parm=
        |   Estimate     StdErr
-------------------------------
  VAR_1 |   -0.16991    0.11077
  VAR_2 |   -0.19035    0.20916
  VAR_3 |    0.01128    0.17470


COVM=
    SYM |     VAR_1     VAR_2     VAR_3
---------------------------------------
  VAR_1 |   0.01227
  VAR_2 |   0.01387   0.04375
  VAR_3 |   0.01204   0.01357   0.03052


  /* [2] Bootstrap: one sample */
  modl = " 1:3 ";
  optn = [ "print"      3 ,
           "xtol"   1.e-9 ,
           "perm"    1000 ,
           "cov"   "boot" ];
  < gof,parm,shap,ase,covm > = sgmanova(dat1,modl,.,optn);


           ******************************************
           Iterate Parameter Vector and Covariance Matrix
           ******************************************

                Iteration       F_parm       F_covm
                        1  0.002895709  0.787140492
                        2  0.000528193  0.088551751
                        3  0.000242354  0.018838679
                        4  8.5131e-005  0.004223412
                        5  2.3503e-005  0.000946418
                        6  5.7194e-006  0.000211326
                        7  1.3058e-006  4.7040e-005
```

```
 8  2.8855e-007  1.0446e-005
 9  6.2712e-008  2.3156e-006
10  1.3516e-008  5.1277e-007
11  2.9010e-009  1.1349e-007
12  6.2146e-010  2.5119e-008
13  1.3299e-010  5.5616e-009
14  2.8437e-011  1.2322e-009
15  6.0752e-012  2.7325e-010

Successful estimation after 15 iterations wrt. termination
                  criterion 1e-009.


*************************************
Iterate Covariance Matrix for Sign Test
*************************************

         Iteration       F_covm
                 1  0.732058891
                 2  0.080385362
                 3  0.016710788
                 4  0.003792285
                 5  0.000872202
                 6  0.000201543
                 7  4.6766e-005
                 8  1.0903e-005
                 9  2.5550e-006
                10  6.0203e-007
                11  1.4264e-007
                12  3.3978e-008
                13  8.1354e-009
                14  1.9573e-009
                15  4.7301e-010

Successful estimation after 15 iterations wrt. termination
                  criterion 1e-009.
  Bootstrap of 500 samples: 8128 iterations, 1 second(s).


     Parameter Estimates and Asympt. Standard Errors

                   Estimate      AStdErr
          VAR_1  -0.169911314  0.110885337
          VAR_2  -0.190354819  0.211548068
          VAR_3   0.011283515  0.168239956

Covariance Matrix and Standard Errors Computed by Bootstrap
```

```
Test Statistics Q2 . . . . . . . . . . . . . . . . . 3.617311535
Large sample p-value . . . . . . . . . . . . . . . . 0.305862890
Estimated p-value from sign change distribution. . 0.308000000
Precision of p-value from binomial distribution. . 0.014599178
```

Shape Matrix

	VAR_1	VAR_2	VAR_3
VAR_1	0.425389258	0.480977242	0.417240187
VAR_2	0.480977242	1.516573279	0.470591433
VAR_3	0.417240187	0.470591433	1.058037463

Covariance Matrix from Bootstrap

	VAR_1	VAR_2	VAR_3
VAR_1	0.012295558	0.014545545	0.011645648
VAR_2	0.014545545	0.044752585	0.014614700
VAR_3	0.011645648	0.014614700	0.028304683

```
/* [3] Jackknife: one sample */
modl = " 1:3 ";
optn = [ "print"      3 ,
         "xtol"   1.e-9 ,
         "perm"    1000 ,
         "cov"   "jack" ];
< gof,parm,shap,ase,covm > = sgmanova(dat1,modl,.,optn);

         ********************************************
         Iterate Parameter Vector and Covariance Matrix
         ********************************************

              Iteration       F_parm       F_covm
                      1  0.002895709  0.787140492
                      2  0.000528193  0.088551751
                      3  0.000242354  0.018838679
                      4  8.5131e-005  0.004223412
                      5  2.3503e-005  0.000946418
                      6  5.7194e-006  0.000211326
                      7  1.3058e-006  4.7040e-005
                      8  2.8855e-007  1.0446e-005
                      9  6.2712e-008  2.3156e-006
```

```
                10  1.3516e-008  5.1277e-007
                11  2.9010e-009  1.1349e-007
                12  6.2146e-010  2.5119e-008
                13  1.3299e-010  5.5616e-009
                14  2.8437e-011  1.2322e-009
                15  6.0752e-012  2.7325e-010

Successful estimation after 15 iterations wrt. termination
                  criterion 1e-009.


        ************************************
        Iterate Covariance Matrix for Sign Test
        ************************************

                Iteration       F_covm
                        1  0.732058891
                        2  0.080385362
                        3  0.016710788
                        4  0.003792285
                        5  0.000872202
                        6  0.000201543
                        7  4.6766e-005
                        8  1.0903e-005
                        9  2.5550e-006
                       10  6.0203e-007
                       11  1.4264e-007
                       12  3.3978e-008
                       13  8.1354e-009
                       14  1.9573e-009
                       15  4.7301e-010

Successful estimation after 15 iterations wrt. termination
                  criterion 1e-009.
     Jackknife takes: 1488 iterations, 0 second(s).


     Parameter Estimates and Asympt. Standard Errors

                        Estimate      AStdErr
            VAR_1  -0.169911314  0.109925383
            VAR_2  -0.190354819  0.210549827
            VAR_3   0.011283515  0.164919453

Covariance Matrix and Standard Errors Computed by Jackknife
```

```
Test Statistics Q2 . . . . . . . . . . . . . . . . . 3.617311535
Large sample p-value . . . . . . . . . . . . . . . . 0.305862890
Estimated p-value from sign change distribution. . 0.308000000
Precision of p-value from binomial distribution. . 0.014599178
```

Shape Matrix

	VAR_1	VAR_2	VAR_3
VAR_1	0.425389258	0.480977242	0.417240187
VAR_2	0.480977242	1.516573279	0.470591433
VAR_3	0.417240187	0.470591433	1.058037463

Covariance Matrix from Jackknife

	VAR_1	VAR_2	VAR_3
VAR_1	0.012083590	0.015079762	0.011652283
VAR_2	0.015079762	0.044331230	0.015319253
VAR_3	0.011652283	0.015319253	0.027198426

2. Two sample application with 100×3 matrix of normal random generated data:

```
print "Random Generated: nobs=100, nvar=3";
srand(1);
x1 = rand(100,3,'g',"norm");

/* define desired covariance matrix */
sc = cons(3,3,1.);
sc[2,2] = sc[3,3] = 3.;
< eval,evec > = eig(sc);
sh = evec * sqrt(diag(eval)) * evec';
/* introduce the covariance structure into the data set */
/* [1] mu(dat1) = (0,0,0) */
dat1 = x1 * sh;
/* [2] mu(set1(dat2)) = (0,0,0);
       mu(set2(dat2)) = (1,1,1) */
x0 = cons(50,3,0.) |> cons(50,3,1.);
set = [ 50#1 50#2 ]';
dat2 = (dat1 + x0) -> set;
print "DAT2=", dat2;


/* [1] Asymptotic */
modl = " 1:3 ";
optn = [ "print"      3 ,
```

```
         "setvar"     4 ,
         "xtol"   1.e-9 ,
         "perm"    1000 ,
         "cov"   "asym" ];
< gof,parm,shap,ase,covm > = sgmanova(dat2,modl,.,optn);


print "GOF=", gof;
print "Parm=", parm;
print "ASE=", ase;
print "COVM=",covm;
```

****
Sample Sizes (Nobs=100)
****

N	Sample	Entries
1	1	50
2	2	50

****
Iterate Parameter Vector and Covariance Matrix
****

Iteration	F_parm	F_covm
1	0.035971471	0.768636189
2	0.006433164	0.090865619
3	0.001778615	0.021068100
4	0.000570752	0.005177108
5	0.000180947	0.001278926
6	5.4608e-005	0.000317014
7	1.5785e-005	7.8869e-005
8	4.4248e-006	1.9692e-005
9	1.2151e-006	4.9331e-006
10	3.2929e-007	1.2398e-006
11	8.8520e-008	3.1255e-007
12	2.3685e-008	7.9033e-008
13	6.3218e-009	2.0044e-008
14	1.6858e-009	5.0982e-009
15	4.4951e-010	1.3005e-009
16	1.1993e-010	3.3267e-010

Successful estimation after 16 iterations wrt. termination criterion 1e-009.

```
*******************************************
Iterate Parameter Vector and Covariance Matrix
*******************************************

     Iteration       F_parm       F_covm
             1  0.011530429  0.779828895
             2  0.001719311  0.086193398
             3  0.000422777  0.016209131
             4  0.000116202  0.003159214
             5  3.0079e-005  0.000614417
             6  7.2660e-006  0.000120186
             7  1.6788e-006  2.3804e-005
             8  3.7873e-007  4.7908e-006
             9  8.4544e-008  9.8108e-007
            10  1.8828e-008  2.0441e-007
            11  4.2034e-009  4.3291e-008
            12  9.4330e-010  9.3042e-009
            13  2.1309e-010  2.0257e-009
            14  4.8478e-011  4.4594e-010

   Successful estimation after 14 iterations wrt. termination
                     criterion 1e-009.


    Parameter Estimates and Asympt. Stand. Errors of Sample 1

                      Estimate      AStdErr
              VAR_1  -0.053733243  0.148688416
              VAR_2   0.030349437  0.278683932
              VAR_3   0.088199494  0.239947140


    Parameter Estimates and Asympt. Stand. Errors of Sample 2

                      Estimate      AStdErr
              VAR_1   0.712496424  0.148688416
              VAR_2   0.553524926  0.278683932
              VAR_3   0.933640220  0.239947140

Covariance Matrix and Standard Errors Computed by Asympt. Theory


  Test Statistics Q2 . . . . . . . . . . . . . . . . . 12.35036728
  Large sample p-value . . . . . . . . . . . . . . . . 0.006274435
  Estimated p-value from sign change distribution. . . 0.005000000
```

```
 Precision of p-value from binomial distribution. . 0.002230471
```

Shape Matrix

	VAR_1	VAR_2	VAR_3
VAR_1	0.421517275	0.465986306	0.432294415
VAR_2	0.465986306	1.480760993	0.500577849
VAR_3	0.432294415	0.500577849	1.097721731

Asymptotic Covariance Matrix for Sample 1

	VAR_1	VAR_2	VAR_3
VAR_1	0.022108245	0.024440610	0.022673497
VAR_2	0.024440610	0.077664734	0.026254909
VAR_3	0.022673497	0.026254909	0.057574630

Asymptotic Covariance Matrix for Sample 2

	VAR_1	VAR_2	VAR_3
VAR_1	0.022108245	0.024440610	0.022673497
VAR_2	0.024440610	0.077664734	0.026254909
VAR_3	0.022673497	0.026254909	0.057574630

Shift Estimates and ASEs of Sample 2 w.r.t. First Set

	Estimate	AStdErr
VAR_1	0.766229667	0.210277175
VAR_2	0.523175489	0.394118596
VAR_3	0.845440726	0.339336499

```
/* [2] Bootstrap */
modl = " 1:3 ";
optn = [ "print"      3 ,
         "setvar"     4 ,
         "xtol"   1.e-9 ,
         "perm"    1000 ,
         "cov"   "boot" ,
         "samp"    500 ];
< gof,parm,shap,ase,covm > = sgmanova(dat2,modl,.,optn);


print "GOF=", gof;
print "Parm=", parm;
```

```
print "ASE=", ase;
print "COVM=",covm;
```

```
*********************************************
Iterate Parameter Vector and Covariance Matrix
*********************************************

     Iteration       F_parm       F_covm
             1  0.035971471  0.768636189
             2  0.006433164  0.090865619
             3  0.001778615  0.021068100
             4  0.000570752  0.005177108
             5  0.000180947  0.001278926
             6  5.4608e-005  0.000317014
             7  1.5785e-005  7.8869e-005
             8  4.4248e-006  1.9692e-005
             9  1.2151e-006  4.9331e-006
            10  3.2929e-007  1.2398e-006
            11  8.8520e-008  3.1255e-007
            12  2.3685e-008  7.9033e-008
            13  6.3218e-009  2.0044e-008
            14  1.6858e-009  5.0982e-009
            15  4.4951e-010  1.3005e-009
            16  1.1993e-010  3.3267e-010

  Successful estimation after 16 iterations wrt. termination
                      criterion 1e-009.
```

```
*********************************************
Iterate Parameter Vector and Covariance Matrix
*********************************************

     Iteration       F_parm       F_covm
             1  0.011530429  0.779828895
             2  0.001719311  0.086193398
             3  0.000422777  0.016209131
             4  0.000116202  0.003159214
             5  3.0079e-005  0.000614417
             6  7.2660e-006  0.000120186
             7  1.6788e-006  2.3804e-005
             8  3.7873e-007  4.7908e-006
             9  8.4544e-008  9.8108e-007
            10  1.8828e-008  2.0441e-007
            11  4.2034e-009  4.3291e-008
            12  9.4330e-010  9.3042e-009
```

```
                13  2.1309e-010  2.0257e-009
                14  4.8478e-011  4.4594e-010

Successful estimation after 14 iterations wrt. termination
                    criterion 1e-009.
  Bootstrap of 500 samples: 8564 iterations, 1 second(s).


 Parameter Estimates and Asympt. Stand. Errors of Sample 1

                      Estimate      AStdErr
             VAR_1  -0.053733243  0.145543504
             VAR_2   0.030349437  0.279227870
             VAR_3   0.088199494  0.242502047


 Parameter Estimates and Asympt. Stand. Errors of Sample 2

                      Estimate      AStdErr
             VAR_1   0.712496424  0.152723732
             VAR_2   0.553524926  0.285019663
             VAR_3   0.933640220  0.240609908

 Covariance Matrix and Standard Errors Computed by Bootstrap


Test Statistics Q2 . . . . . . . . . . . . . . . . . 12.35036728
Large sample p-value . . . . . . . . . . . . . . . . 0.006274435
Estimated p-value from sign change distribution. . . 0.005000000
Precision of p-value from binomial distribution. . . 0.002230471


                      Shape Matrix

                 VAR_1        VAR_2        VAR_3
       VAR_1   0.421517275  0.465986306  0.432294415
       VAR_2   0.465986306  1.480760993  0.500577849
       VAR_3   0.432294415  0.500577849  1.097721731


       Covariance Matrix from Bootstrap for Sample 1

                 VAR_1        VAR_2        VAR_3
       VAR_1   0.021182912  0.022713391  0.022610751
       VAR_2   0.022713391  0.077968203  0.025878858
       VAR_3   0.022610751  0.025878858  0.058807243
```

```
Covariance Matrix from Bootstrap for Sample 2

              VAR_1        VAR_2        VAR_3
VAR_1   0.023324538  0.026994456  0.022958273
VAR_2   0.026994456  0.081236208  0.028928557
VAR_3   0.022958273  0.028928557  0.057893128


Shift Estimates and ASEs of Sample 2 w.r.t. First Set

                 Estimate      AStdErr
VAR_1   0.766229667  0.210967889
VAR_2   0.523175489  0.399004275
VAR_3   0.845440726  0.341614359
```

3.13 Function `xyplot`

xyplot(y,x<,titl<,optn<,labl<,fpath>>>>)

Purpose: The `xyplot` function prints one or more (x, y) diagrams for specified vectors x and y.

Input: **y** This argument must be a vector or a matrix with c_y columns.

x This argument must be a vector or a matrix with c_x columns.

titl This argument must be a single string argument specifying the title for the plot.

optn This argument must be a numeric scalar or a vector specifying the output options:

1 amount of printed output (=0: no printed output)

2

3 specifying how to place the labels if they are specified:

4

fpath This argument must be a single string argument specifying the (path and) file name for a text output file containing the plot.

Output: This function has no return arguments. However, it has printed and/or file output.

Restrictions: 1. Missing values in y and x are not permitted.

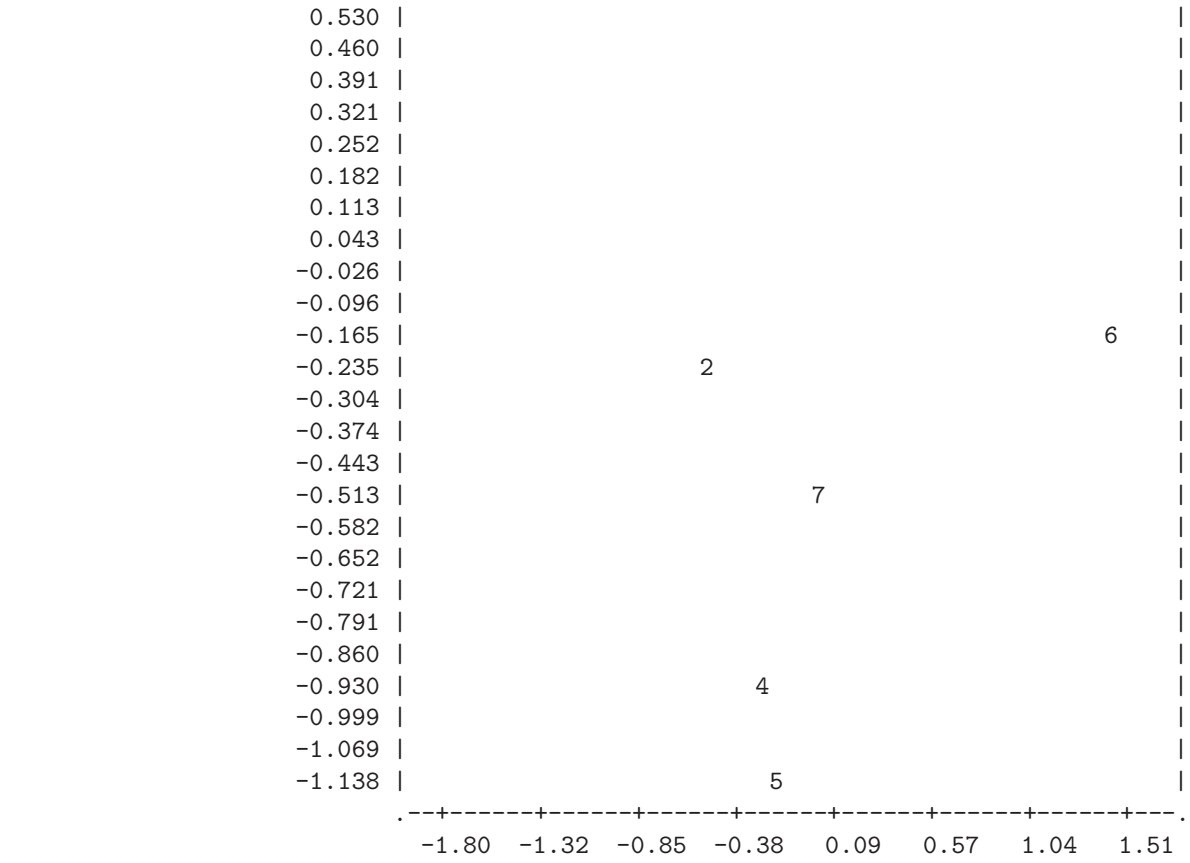
2. String or complex data for x and y are not permitted.

Relationships: histplot()

Examples: 1. Plot of object scores from PRINCALS output:

```
Scores= [ 1.25915  -1.79503,
         -0.25081  -0.47219,
          1.71736   1.17960,
         -0.94401  -0.23549,
         -1.11559  -0.15896,
         -0.18344   1.44263,
         -0.48266   0.03943 ];
 rnam = [ "Crowd" "Audience" "Public" "Mob"
          "Primary" "Secondary" "Modern" ];
 Scores = rname(Scores,rnam);
 nr = nrow(Scores); nc = ncol(Scores);
 y =  Scores[,1];
 x =  Scores[,2];
 titl = "Scores from PRINCALS";
 labels = [ 1:nr ];
 optn = [ 2 ,
          0 ,
          1 ];
 xyplot(y,x,titl,optn,labels);
```

```
                    ********************
                    Scores from PRINCALS
                    ********************

       .--+------+------+------+------+------+------+------+---.
 1.920 |                                                       |
 1.850 |                                                       |
 1.781 |                                                       |
 1.711 |                                             3         |
 1.642 |                                                       |
 1.572 |                                                       |
 1.503 |                                                       |
 1.433 |                                                       |
 1.364 |                                                       |
 1.294 |                                                       |
 1.225 |  1                                                    |
 1.155 |                                                       |
 1.086 |                                                       |
 1.016 |                                                       |
 0.947 |                                                       |
 0.877 |                                                       |
 0.808 |                                                       |
 0.738 |                                                       |
 0.669 |                                                       |
 0.599 |                                                       |
```



3.14 Function `zoverw`

< r1,r2,r3,r4 > = zoverw(sopt,muz,sigz,muw,sigw,rho<,r><,opt>)

Purpose: The `zoverw` function computes probability and density of the ratio z/w of two normal distributed variables z and w. As Marsaglia (1965, 2006) states the distribution of z/w is either unimodal or bimodal normal. By transforming z/w into $\frac{a+x}{b+y}$ Marsaglia finds that the distribution of z/w is unimodal for the tall and narrow region $a < 2.5$ and $b > 4$ in the (a, b) plane. The mean and standard deviation, the probability and density of the distribution of z/w are only computed when the input arguments `muz`, `sigz`, `muw`, `sigw`, and `rho` result in a tranformation (a, b) indicating unimodality. There are two versions for the computation of probability and density

1. a numerically inexpensive which assumes that all values of the de-

nominator w variable can only be positive.

2. a numerically slightly more expensive version which is not based on this assumption.

Input: sopt the string option specifies the method and can have the following values:

"Fprob" compute probability

"Gprob" compute probability under assumption that all w are positive

"fdens" compute density

"gdens" compute density under assumption that all w are positive

muz a scalar or n vector of means of z

sigz a scalar or n vector of standard deviation of z

muw a scalar or n vector of means of w

sigw a scalar or n vector of standard deviation of w

rho the product moment correlation among z and w

r this argument can be missing value, a scalar or a n vector of values in $[0, 1]$. It specifies the t used in the computation of the probability $F(t) = Prob(z/w < t)$ or the density $f(t) = F'(t)$.

opt a real scalar or vector specifying the following options:

1. controls the printed output (=0: no printed output)
2. number of random generated z/w values for second output argument `r2`
3. ngphi: number of generated $\Phi((z/w - mu)/sigma)$ values for chi^2 values or third output argument `r3`
4. nggf: number of generated $G(z/w)$ values for chi^2 values for fourth output argument `r4`
5. seed for random generator (default=123456789).

Output: r1 The first and default return argument should be a matrix with n rows and a minimum of 4 and a maximum of 7 columns.

1. contains the value a of the transformation $\frac{(a+x)}{(b+y)}$
2. contains the value b of the transformation $\frac{(a+x)}{(b+y)}$
3. contains the mean μ of the z/w distribution
4. contains the standard deviation σ of the z/w distribution
5. optional only if argument r is not missing:
6. optional only if `optn`$[3] > 0$: contains $chi^2(Phi((z/w - \mu)/\sigma)$ for random generated z/w
7. optional only if `optn`$[4] > 0$: contains $chi^2(G(z/w))$ or $chi^2(F(z/w))$ for random generated z/w

r2 only if optn[2] $>$ 0: contains a vector of opt[2] values of random generated z/w; otherwise returns missing value.

r3 only if optn[3] $>$ 0: contains a vector of opt[3] values of random generated $Phi((z/w - \mu)/\sigma)$; otherwise returns missing value.

r4 only if optn[4] $>$ 0: contains a vector of opt[4] values of random generated $g(z/w)$ or $F(z/w)$; otherwise returns missing value.

Restrictions: 1. The input arguments muz, sigz, muw, sigw, and rho must be either scalars or vectors of the same dimension n.

2. No missing values are permitted in first five input arguments.

Relationships:

Examples: 1. Example by Marsaglia, 2006: a=.5, b=8, mu=0.64, sigm=.1637:

```
muz = 30.5; sigz = 5.; muw = 32.; sigw = 4.; rho = .8;
opt = [    2 ,   /* ipri  */
          20 ,   /* ngen  */
       10000 ,   /* ngenP */
     1000000 ];  /* ngenG */
r1 = zoverw("Gprob",muz,sigz,muw,sigw,rho,.,opt);
< r1,r2,r3,r4 > = zoverw("Gprob",muz,sigz,muw,sigw,rho,.,opt);


            Chi Square Values of Generated z/w Values
            *****************************************

     Zmu=    31 Zsig=     5 Wmu=    32 Wsig=     4 Rho=0.800

Number         A         B   Mu(z/w)     Sigma  Chi(Phi)    Chi(G)

     1 0.5000000 8.0000000 0.9519766 0.0959551 126.54000 98.904000

               Chisquare: 70 < chisq < 130 is OK!


            Chi Square Values of Generated z/w Values
            *****************************************

     Zmu=    31 Zsig=     5 Wmu=    32 Wsig=     4 Rho=0.800

Number         A         B   Mu(z/w)     Sigma  Chi(Phi)    Chi(G)

     1 0.5000000 8.0000000 0.9519766 0.0959551 95.800000 95.595200

               Chisquare: 70 < chisq < 130 is OK!
```