CMAT Newsletter: September 2003

Wolfgang M. Hartmann

September 2003

1 General Remarks

Nobody, except Thomas and me, is using the forum. Of course, this has to do with the fact that nobody, except myself, is working with CMAT. And that, of course, has to do with the fact that CMAT was (and maybe is) buggy. You should see me here on my computer putting ashes on my head! The most important thing is fixing bugs and finishing up the two language features, multidimensional arrays and structures. Also, implementing dynamic binding of user supplied C and Fortran routines.

2 Modifications of Features

2.1 Fixed Bugs

A number of bugs were fixed:

1. some very bad ones for specific dense matrix multiplication (especially those involving dense upper triangular matrices).
2. Reading and writing sparse vectors using the `import()` and `export()` functions.
3. Fixing problems with return arguments of the `sem` function. Especially, bootstrap results of indirect and total effects.
4. `rand()` function was extended for generating nonsquare diagonal, lower, and upper triangular matrices (except for orthogonal type).

2.2 Changes in `import()` and `export()`

The functions

<a,b,c,...> = import("path","ftyp")

export(a,"path","ftyp"<,optn>)

were extended for reading and writing Matlab (version 5) .MAT files by specifying the `"ftyp"` keyword `"mat"`. The following keywords are available now:

"mat" for Matlab (version 5 .MAT) files

"sas" for SAS XPORT data sets

"sps" for SPSS PORT and SAVE data sets (writing is still in the works and not finished)

"spl" for S-Plus3 data objects

"sta" for STATA data sets

"mtb" for MINITAB data sets

At this time no string data and no missing values can be read from or written to .MAT files. Otherwise, almost all kinds of .MAT files, like numeric constants, sparse and dense arrays, cell arrays, and structures) can be read, but we had no interest to include general *Object Arrays*.

2.3 Outlier Identification by Jackknife in `sem`

The new option `"outlier"` with an integer `n` was added to the `sem` function. For `n=0` a Jackknife (leave-one-out) process is performed and a table of smallest chi^2 and corresponding p values is printed indicating the strongest outliers. Stepwise removing the observation which contributed to the largest χ^2 reduction will decrease the χ^2 values and increase the p values corresponding to the reduced data sets. This a very tempting feature, since it permits to increase p values of a model. For `n>0` an n-step process is performed where in each step the observation with the largest reduction of the χ^2 value is hypothetically removed. A summary table shows the stepwise reduction in the χ^2 value and the increase of the p value for the reduced data sets. Note, the reduction is not performed in reality, it is just simulated. If you really want to delete the outlier observation you must do so by changing the data set and rerun the `sem` function.
We illustrate the results using an example from the EQS Manual. The $N = 50$ observation artificial data set with one huge outlier as observation 50 shows a χ^2 value

```
Normal Th. Chi-square (df = 8). 16.6917 Prob>chi**2 = 0.0335
```

When analyzing the data without the outlier observation 50 shows the much smaller χ^2 value:

```
Normal Th. Chi-square (df = 8).  2.7012 Prob>chi**2 = 0.9517
```

The EQS manual illustrates how to identify the outlier observation with its contribution to the kurtosis. The approach here is more general in that it identifies outliers which are *model misfits* which still fit in the multivariate normal distributed range and have no excessive kurtosis or skewness.

```
  print "SEM12: Simulated Confirmatory Factor Analysis, BENTLER, 1985, p.105";
  print "Test WLS and DWLS estimation and INWGT= and OUTWGT data sets";

#include "..\\tdata\\sem12.dat"

 d12 = d12[ ,1:6];
 name = [ "V1" : "V6" ];
 d12 = cname(d12,name);

  ram = [ 1   1   7   "Z1" ,
          1   2   7   "Z2" ,
          1   3   7   "Z3" ,
          1   4   8   "Z4" ,
          1   5   8   "Z5" ,
          1   6   8   "Z6" ,
          2   1   1   "EPS1" ,
          2   2   2   "EPS2" ,
          2   3   3   "EPS3" ,
          2   4   4   "EPS4" ,
          2   5   5   "EPS5" ,
          2   6   6   "EPS6" ,
```

```
        2   7   7      1. ,
        2   8   8      1. ,
        2   8   7   "COR" ];

parms = [ "EPS1"   .   0. ,   "EPS2"   .    0. ,
          "EPS3"   .   0. ,   "EPS4"   .    0. ,
          "EPS5"   .   0. ,   "EPS6"   .    0. ];
cpnam = [ "_PNAM" "_INIT" "_LBC" ];
parms = cname(parms,cpnam);
```

The "outlier" option with n=0 performs a Jackknife (leave-one-out) process for the 50 observations indicating clearly observation 50 as an outlier:

```
/* Jackknife for outliers: */
optns = [ "data"           "raw" ,
          "anal"           "cov" ,
          "meth"            "ml" ,
          "cl"                 1 ,
          "outli"              0 ,
          "nomod"                ,
          "pall"                ];
gof = sem(d12,ram,optns,parms);
```

```
               Rank Order of Outlier Observations
               **********************************

          Function=0.3406 Chisquare=16.69 Pvalue= 0.0335

           Rank    Nobs     OptCrit  Chi-square    pval
              1      50  0.05627578  2.70123733 0.95169
              2      15  0.31474078  15.1075576 0.05709
              3      43  0.31556869  15.1472972 0.05634
              4      12  0.31656969  15.1953451 0.05546
              5       8  0.32461098  15.5813270 0.04878
              6       6  0.32480679  15.5907262 0.04863
              7      13  0.32613436  15.6544493 0.04760
              8      37  0.32654850  15.6743280 0.04729
              9       7  0.32752192  15.7210520 0.04655
             10      31  0.32778227  15.7335490 0.04636

   Number failing optimizations 0 during Loo outlier detection.
         (Number Iterations 1256 and Function Calls 1359)
```

The "outlier" option with n=3 performs a 3 step Jackknife (leave-one-out) process for 50, 49, 48, and 47 observations where in each step the observation with smallest chi^2 value is removed from the data:

```
/* Jackknife for outliers: */
optns = [ "data"           "raw" ,
          "anal"           "cov" ,
          "meth"            "ml" ,
          "cl"                 1 ,
          "outli"              3 ,
```

```
          "nomod"                ,
          "pall"                ];
gof = sem(d12,ram,optns,parms);

          ****************************************
          Rank Order of Outlier Observations (Nobs=50)
          ****************************************

         Function=0.3406 Chisquare=16.69 Pvalue= 0.0335

          Rank    Nobs     OptCrit  Chi-square    pval
             1      50  0.05627578  2.70123733 0.95169
             2      15  0.31474078  15.1075576 0.05709
             3      43  0.31556869  15.1472972 0.05634
             4      12  0.31656969  15.1953451 0.05546
             5       8  0.32461098  15.5813270 0.04878
             6       6  0.32480679  15.5907262 0.04863
             7      13  0.32613436  15.6544493 0.04760
             8      37  0.32654850  15.6743280 0.04729
             9       7  0.32752192  15.7210520 0.04655
            10      31  0.32778227  15.7335490 0.04636
```

It is interesting to note that after removing the strong outlier observation 50 in the second step observation 28 is considerd the worst outlier which did not even occur in the rank order of the ten largest outliers in the first step. That obviously has to do with the well-known masking effects in stepwise outlier procedures.

```
          ****************************************
          Rank Order of Outlier Observations (Nobs=49)
          ****************************************

        Function=0.05628 Chisquare=2.701 Pvalue= 0.9517

          Rank    Nobs     OptCrit  Chi-square    pval
             1      28  0.04358613  2.04854810 0.97949
             2       7  0.04428755  2.08151505 0.97841
             3      36  0.04519250  2.12404738 0.97697
             4       9  0.05043014  2.37021647 0.96751
             5       5  0.05175661  2.43256083 0.96480
             6      17  0.05252000  2.46843989 0.96319
             7      35  0.05275171  2.47933054 0.96269
             8      30  0.05349017  2.51403776 0.96107
             9      16  0.05366883  2.52243518 0.96068
            10       3  0.05370540  2.52415363 0.96060

          ****************************************
          Rank Order of Outlier Observations (Nobs=48)
          ****************************************

        Function=0.04359 Chisquare=2.049 Pvalue= 0.9795
```

```
  Rank    Nobs     OptCrit  Chi-square    pval
     1      36  0.03219261  1.48086016 0.99302
     2      16  0.03842558  1.76757676 0.98733
     3       7  0.03866238  1.77846955 0.98707
     4      47  0.03886566  1.78782016 0.98684
     5      27  0.03946553  1.81541450 0.98616
     6       8  0.04062435  1.86872009 0.98478
     7      44  0.04081860  1.87765569 0.98454
     8      24  0.04116463  1.89357288 0.98410
     9      14  0.04148702  1.90840294 0.98369
    10      10  0.04159397  1.91332281 0.98355

  ****************************************
  Rank Order of Outlier Observations (Nobs=47)
  ****************************************

Function=0.03219 Chisquare=1.481 Pvalue= 0.9930

  Rank    Nobs     OptCrit  Chi-square    pval
     1      16  0.02633829  1.18522321 0.99679
     2       8  0.02832111  1.27445004 0.99585
     3       7  0.02854093  1.28434199 0.99574
     4      47  0.02863760  1.28869179 0.99569
     5      24  0.02951064  1.32797893 0.99521
     6      14  0.02961938  1.33287188 0.99515
     7      27  0.02986343  1.34385447 0.99501
     8      10  0.02989699  1.34536450 0.99499
     9      44  0.03039086  1.36758867 0.99469
    10      17  0.03056321  1.37534445 0.99459

           *************************
           Stepwise Outlier Reduction
           *************************

        Step    Nobs  Chi-square    pval
           1      50  2.70123733 0.95169
           2      28  2.04854810 0.97949
           3      36  1.48086016 0.99302
           4      16  1.18522321 0.99679

   Number failing optimizations 0 during Loo outlier detection.
         (Number Iterations 2082 and Function Calls 2474)
```

It is interesting to note that the existence of strong outliers in the data contributes considerably to the computational cost (number of iterations and function calls until convergence). After removing the outlier 50 we notice much faster optimizations to fit the model.

2.4 SRMR Goodness-of-Fit Coefficient Added in `sem`

For $p \times p$ input covariance matrix $S = (s_{ij})$ and model matrix $C = (c_{ij})$ the *Standardized Root Mean Residual* (SRMR) coefficient is defined as:

$$SRMR = \sqrt{\frac{2}{p(p+1)} \sum_{i=1}^{p} \sum_{j=1}^{i} \frac{(s_{ij} - c_{ij})^2}{s_{ii} s_{jj}}}$$

For multiple sample analysis, the coefficient is added across the samples.

2.5 Outlier Identification by Jackknife in `svm`

Similar to the `sem` function the `outlier` option was added to `svm` function. The leave-one-out algorithm did exist already but did not show a table of outlier observations. The table now shows how the classification error can be improved by dropping outlier observations. The old `cloo` option is still producing the fast summary statistics similar to the *SVM Light* program by Joachims. The Jackknife of the algorithm for new `outlier` option takes more computer time.

2.6 Added new Algorithms for `ode()`

Two new algorithms were added to the `ode()` function:

1. Fehlberg (4,5) Runge Kutta method (see Forsythe, Malcolm, and Moler, 1977)
2. Verner (5,6) Runge Kutta method (Hull, Enright, and Jackson, Univ. of Toronto).

The following example is taken from Forsythe, Malcolm, and Moler (1977, pp. 129):

```
ecc = .25;
alfa = .25 * macon("pi");
alfsq = alfa * alfa;
y0 = cons(4);
y0[1] = 1. - ecc;
y0[4] = alfa * sqrt((1. + ecc) / (1. - ecc));
tab = [ 0. : .5 : 12. ];

function orbit(t,y) global(alfsq) {
  r = y[1] * y[1] + y[2] * y[2];
  r = r * sqrt(r) / alfsq;
  ydot = cons(4);
  ydot[1] = y[3]; ydot[2] = y[4];
  ydot[3] = -y[1] / r;
  ydot[4] = -y[2] / r;
  return(ydot);
}

/* oderkf45: Runge-Kutta (4,5) Fehlberg by
             Watts and Shampine: nfun=1723 */
par = cons(1,9,.);
par[1] = 2; par[2] = 1e-9;
par[4] = 4;
< y,t > = ode(orbit,tab,y0,par);
print t, y;

         Solve Initial Value Problem dY_i/dT = F(Y_i,T)
    Use Runge-Kutta (4,5) Fehlberg Method (Watts & Shampine)
```

```
          Start Iteration at T=0  Rel. Precision=1e-009

            Iter Nstep   Nfunc   Curr.Time   Rel.Error

               0     0       1  0.00000000  1.000e-009
               1     1     119  0.50000000  1.000e-009
               2     1     192  1.00000000  1.000e-009
               3     1     287  1.50000000  1.000e-009
               4     1     365  2.00000000  1.000e-009
               5     1     420  2.50000000  1.000e-009
               6     1     469  3.00000000  1.000e-009
               7     1     518  3.50000000  1.000e-009
               8     1     579  4.00000000  1.000e-009
               9     1     664  4.50000000  1.000e-009
              10     1     713  5.00000000  1.000e-009
              11     1     762  5.50000000  1.000e-009
              12     1     811  6.00000000  1.000e-009
              13     1     883  6.50000000  1.000e-009
              14     1     978  7.00000000  1.000e-009
              15     1    1051  7.50000000  1.000e-009
              16     1    1148  8.00000000  1.000e-009
              17     1    1263  8.50000000  1.000e-009
              18     1    1336  9.00000000  1.000e-009
              19     1    1431  9.50000000  1.000e-009
              20     1    1509  10.0000000  1.000e-009
              21     1    1564  10.5000000  1.000e-009
              22     1    1613  11.0000000  1.000e-009
              23     1    1662  11.5000000  1.000e-009
              24     1    1723  12.0000000  1.000e-009
       Sucessful Termination at T=12  Rel. Precision=1e-009
```

	1	2	3	4	5
1	0.75000	0.61977	0.29442	-0.10518	-0.49030
2	0.00000	0.47779	0.81218	0.95804	0.93987
3	0.00000	-0.49525	-0.76260	-0.80631	-0.71918
4	1.01394	0.84521	0.47923	0.11427	-0.17238

	6	7	8	9	10
1	-0.81394	-1.05403	-1.20074	-1.25000	-1.20074
2	0.79959	0.57571	0.30016	-6e-010	-0.30016
3	-0.56845	-0.38883	-0.19672	5e-011	0.19672
4	-0.37586	-0.50910	-0.58415	-0.60837	-0.58415

	11	12	13	14	15
1	-1.05403	-0.81394	-0.49030	-0.10518	0.29442
2	-0.57571	-0.79959	-0.93987	-0.95804	-0.81218
3	0.38883	0.56845	0.71918	0.80631	0.76260
4	-0.50910	-0.37586	-0.17238	0.11427	0.47923

```
   |        16        17        18        19        20
---------------------------------------------------------
 1 |   0.61977   0.75000   0.61977   0.29442  -0.10518
 2 |  -0.47779    5e-009   0.47779   0.81218   0.95804
 3 |   0.49525   -6e-009  -0.49525  -0.76260  -0.80631
 4 |   0.84521   1.01394   0.84521   0.47923   0.11427

   |        21        22        23        24        25
---------------------------------------------------------
 1 |  -0.49030  -0.81394  -1.05403  -1.20074  -1.25000
 2 |   0.93987   0.79959   0.57571   0.30016   -1e-008
 3 |  -0.71918  -0.56845  -0.38883  -0.19672    6e-009
 4 |  -0.17238  -0.37586  -0.50910  -0.58415  -0.60837
```

```
/* odevrk: Verner 5-6 Runge-Kutta by
           K.R. Jackson: nfun=1105 */
par = cons(1,9,.);
par[1] = 2; par[2] = 1e-9;
par[4] = 5;
< y,t > = ode(orbit,tab,y0,par);
print t, y;
```

```
          Solve Initial Value Problem dY_i/dT = F(Y_i,T)
          Use Runge-Kutta (5,6) Verner Method (Jackson)
          Start Iteration at T=0  Rel. Precision=1e-009

            Iter Nstep   Nfunc   Curr.Time   Rel.Error

               0     0       1  0.00000000  1.000e-009
               1     1      65  0.50000000  1.000e-009
               2     1     121  1.00000000  1.000e-009
               3     1     169  1.50000000  1.000e-009
               4     1     209  2.00000000  1.000e-009
               5     1     249  2.50000000  1.000e-009
               6     1     289  3.00000000  1.000e-009
               7     1     329  3.50000000  1.000e-009
               8     1     369  4.00000000  1.000e-009
               9     1     409  4.50000000  1.000e-009
              10     1     449  5.00000000  1.000e-009
              11     1     489  5.50000000  1.000e-009
              12     1     529  6.00000000  1.000e-009
              13     1     569  6.50000000  1.000e-009
              14     1     617  7.00000000  1.000e-009
              15     1     673  7.50000000  1.000e-009
              16     1     737  8.00000000  1.000e-009
              17     1     801  8.50000000  1.000e-009
              18     1     857  9.00000000  1.000e-009
              19     1     905  9.50000000  1.000e-009
              20     1     945  10.0000000  1.000e-009
              21     1     985  10.5000000  1.000e-009
```

```
              22     1    1025  11.0000000  1.000e-009
              23     1    1065  11.5000000  1.000e-009
              24     1    1105  12.0000000  1.000e-009
       Sucessful Termination at T=12  Rel. Precision=1e-009
```

	1	2	3	4	5
1	0.75000	0.61977	0.29442	-0.10518	-0.49030
2	0.00000	0.47779	0.81218	0.95804	0.93987
3	0.00000	-0.49525	-0.76260	-0.80631	-0.71918
4	1.01394	0.84521	0.47923	0.11427	-0.17238

	6	7	8	9	10
1	-0.81394	-1.05403	-1.20074	-1.25000	-1.20074
2	0.79959	0.57571	0.30016	3e-010	-0.30016
3	-0.56845	-0.38883	-0.19672	5e-009	0.19672
4	-0.37586	-0.50910	-0.58415	-0.60837	-0.58415

	11	12	13	14	15
1	-1.05403	-0.81394	-0.49030	-0.10518	0.29442
2	-0.57571	-0.79959	-0.93987	-0.95804	-0.81218
3	0.38883	0.56845	0.71918	0.80631	0.76260
4	-0.50910	-0.37586	-0.17238	0.11427	0.47923

	16	17	18	19	20
1	0.61977	0.75000	0.61977	0.29442	-0.10518
2	-0.47779	7e-008	0.47779	0.81218	0.95804
3	0.49525	-6e-008	-0.49525	-0.76260	-0.80631
4	0.84521	1.01394	0.84521	0.47923	0.11427

	21	22	23	24	25
1	-0.49030	-0.81394	-1.05403	-1.20074	-1.25000
2	0.93988	0.79959	0.57571	0.30016	1e-008
3	-0.71918	-0.56845	-0.38883	-0.19672	8e-009
4	-0.17238	-0.37586	-0.50910	-0.58415	-0.60837

```
 /* Default Kamona method: nfun=1329 */
 par = cons(1,9,.);
 par[1] = 2; par[2] = 1e-9;
 < y,t > = ode(orbit,tab,y0,par);
 print t, y;

          Solve Initial Value Problem dY_i/dT = F(Y_i,T)
               Use Gear's Method for Stiff Systems
          Start Iteration at T=0  Rel. Precision=1e-009
```

Iter	Nstep	Nfunc	Njac	CurrentTime	AvStpSiz	LStpSiz	AvOrd	LOrd
1	75	111	12	0.50000000	6.9e-003	0.017235	3.59	5
2	101	138	13	1.00000000	0.010025	0.020477	3.95	5
3	127	176	15	1.50000000	0.011878	0.023084	4.17	5
4	154	212	17	2.00000000	0.013044	0.022652	4.31	5
5	172	231	18	2.50000000	0.014571	0.032947	4.38	5
6	186	245	18	3.00000000	0.016164	0.039567	4.43	5
7	199	258	18	3.50000000	0.017768	0.042245	4.47	5
8	210	269	18	4.00000000	0.019050	0.042245	4.50	5
9	270	369	33	4.50000000	0.016684	0.035993	4.13	5
10	283	389	34	5.00000000	0.017790	0.044831	4.17	5
11	294	410	34	5.50000000	0.018774	0.036921	4.20	5
12	309	428	35	6.00000000	0.019505	0.030269	4.24	5
13	333	457	36	6.50000000	0.019557	0.018610	4.29	5
14	357	482	36	7.00000000	0.019640	0.022682	4.34	5
15	379	505	36	7.50000000	0.019807	0.018789	4.38	5
16	409	538	37	8.00000000	0.019576	0.012497	4.42	5
17	442	572	38	8.50000000	0.019265	0.016564	4.47	5
18	468	598	38	9.00000000	0.019258	0.020103	4.50	5
19	495	632	41	9.50000000	0.019197	0.020655	4.52	5
20	524	667	43	10.0000000	0.019091	0.022136	4.55	5
21	542	686	44	10.5000000	0.019375	0.032319	4.56	5
22	556	701	45	11.0000000	0.019785	0.039157	4.58	5
23	569	714	45	11.5000000	0.020262	0.042144	4.59	5
24	900	1329	198	12.0000000	0.013333	2.5e-014	3.51	1

Sucessful Termination at T=12 Rel. Precision=1e-009

	1	2	3	4	5
1	0.75000	0.61977	0.29442	-0.10518	-0.49030
2	0.00000	0.47779	0.81218	0.95804	0.93988
3	0.00000	-0.49525	-0.76260	-0.80631	-0.71918
4	1.01394	0.84521	0.47923	0.11427	-0.17238

	6	7	8	9	10
1	-0.81394	-1.05403	-1.20074	-1.25000	-1.20074
2	0.79959	0.57571	0.30016	2e-008	-0.30016
3	-0.56845	-0.38883	-0.19672	-2e-008	0.19672
4	-0.37586	-0.50910	-0.58415	-0.60837	-0.58415

	11	12	13	14	15
1	-1.05403	-0.81394	-0.49030	-0.10518	0.29442
2	-0.57571	-0.79959	-0.93988	-0.95804	-0.81218
3	0.38883	0.56845	0.71918	0.80631	0.76260
4	-0.50910	-0.37586	-0.17238	0.11427	0.47923

	16	17	18	19	20

```
  1 |   0.61977   0.75000   0.61977   0.29442  -0.10518
  2 |  -0.47779   -3e-007   0.47779   0.81218   0.95804
  3 |   0.49525    3e-007  -0.49525  -0.76260  -0.80631
  4 |   0.84521   1.01394   0.84521   0.47923   0.11427

    |        21        22        23        24        25
----------------------------------------------------------
  1 |  -0.49030  -0.81394  -1.05403  -1.20074  -1.25000
  2 |   0.93988   0.79959   0.57571   0.30016    5e-007
  3 |  -0.71918  -0.56845  -0.38883  -0.19672   -3e-007
  4 |  -0.17238  -0.37586  -0.50910  -0.58415  -0.60837
```

For this example the (5,6) Verner Runge-Kutta method by Jackson needs the smallest number (1105) function calls for integrating the interval. The default implementation by Kahane takes 1329 function calls from which more than 600 were needed in the last interval (which could still be a bug). The worst was the (4,5) Fehlberg Runge-Kutta method found at Forsythe, Malcolm, and Moler (1977) which needs 1723 function calls.

2.7 New Functions

New functions are implemented:

generead not finished yet.

tri2vec returns a vector containing the values of a lower or upper triangle of an input object.

polychor computes polychoric correlation matrix, ML threshold estimates and their asymptotic covariance matrices. It also computes a test for the distribution assumption of multivariate normality for the ordinal variables.

3 New Developments

New developments in CMAT are described in more detail in a corresponding `newdev.pdf` file on this site. Here only some short comments.

3.1 Function `generead`

a = generead(”path”,rstr<,optn<,code>>)

Still not finished yet.

3.2 Function `tri2vec`

b = tri2vec(a<,opt>)

Purpose:

Input: a The first input argument is usually a $n \times m$ matrix.

”sopt” the second input argument is optional. If used it must have one of the following string values:

”dlow” the values of the lower triangle of `a` are returned in `b`.

”slow” the values of the strict lower triangle of `a` are returned in `b`.

”dupp” the values of the upper triangle of `a` are returned in `b`.

"supp" the values of the strict upper triangle of `a` are returned in `b`.

Output: The result `b` is a vector containing the values of the (strict) lower or upper triangle of input object `a`.

Restrictions: 1. For input scalar or vector and the strict options the result should be an empty object of zero size. Currently a missing value is returned in such cases.

Relationships: shape(), tri2sym()

Examples: 1. Diagonal Matrix:

```
srand(1);
print "adia=", adia = rand(4,4,'d');
print "vd1=", vd1 = tri2vec(adia,"dlow");
print "vd2=", vd2 = tri2vec(adia,"slow");
print "vd3=", vd3 = tri2vec(adia,"dupp");
print "vd4=", vd4 = tri2vec(adia,"supp");
```

```
adia=
 D |         1         2         3         4
-------------------------------------------
 1 |   0.18496         0         0         0
 2 |         0   0.97009         0         0
 3 |         0         0   0.39982         0
 4 |         0         0         0   0.25940
```

```
vd1=
   |         1
--------------
 1 |   0.18496
 2 |   0.00000
 3 |   0.97009
 4 |   0.00000
 5 |   0.00000
 6 |   0.39982
 7 |   0.00000
 8 |   0.00000
 9 |   0.00000
10 |   0.25940
```

```
vd2=
 Z |   1
--------
 1 |   0
 2 |   0
 3 |   0
 4 |   0
 5 |   0
 6 |   0
```

```
vd3=
   |         1
--------------
 1 |   0.18496
 2 |   0.00000
 3 |   0.97009
 4 |   0.00000
 5 |   0.00000
 6 |   0.39982
 7 |   0.00000
 8 |   0.00000
 9 |   0.00000
10 |   0.25940
```

```
vd4=
 Z |   1
--------
 1 |   0
 2 |   0
 3 |   0
 4 |   0
 5 |   0
 6 |   0
```

2. SCOMP Matrices: SYM, LOW, UPP:

```
print "asym=", asym = rand(4,4,'s');
print "vs1=", vs1 = tri2vec(asym,"dlow");
print "vs2=", vs2 = tri2vec(asym,"slow");
print "vs3=", vs3 = tri2vec(asym,"dupp");
print "vs4=", vs4 = tri2vec(asym,"supp");
```

```
asym=
 S |         1         2         3         4
-------------------------------------------
 1 |   0.92160
 2 |   0.96928   0.54298
 3 |   0.53169   0.04979   0.06657
 4 |   0.81932   0.52387   0.85339   0.06718
```

```
vs1=
   |         1
--------------
 1 |   0.92160
 2 |   0.96928
 3 |   0.54298
 4 |   0.53169
 5 |   0.04979
 6 |   0.06657
 7 |   0.81932
 8 |   0.52387
 9 |   0.85339
10 |   0.06718
```

```
vs2=
   |         1
--------------
 1 |   0.96928
 2 |   0.53169
 3 |   0.04979
 4 |   0.81932
 5 |   0.52387
 6 |   0.85339
```

vs3=

	1
1	0.92160
2	0.96928
3	0.53169
4	0.81932
5	0.54298
6	0.04979
7	0.52387
8	0.06657
9	0.85339
10	0.06718

vs4=

	1
1	0.96928
2	0.53169
3	0.81932
4	0.04979
5	0.52387
6	0.85339

```
  print "alow=", alow = rand(4,4,'l');
  print "vl1=", vl1 = tri2vec(alow,"dlow");
  print "vl2=", vl2 = tri2vec(alow,"slow");
  print "vl3=", vl3 = tri2vec(alow,"dupp");
  print "vl4=", vl4 = tri2vec(alow,"supp");
```

alow=

L	1	2	3	4
1	0.95702	0	0	0
2	0.29719	0.27261	0	0
3	0.68993	0.97676	0.22651	0
4	0.68824	0.41276	0.55855	0.28723

vl1=

	1
1	0.95702
2	0.29719
3	0.27261
4	0.68993
5	0.97676
6	0.22651
7	0.68824
8	0.41276
9	0.55855
10	0.28723

vl2=

	1
1	0.29719
2	0.68993
3	0.97676
4	0.68824
5	0.41276
6	0.55855

```
vl3=                                    vl4=
   |         1                           Z |   1
--------------                          --------
 1 |   0.95702                           1 |   0
 2 |   0.00000                           2 |   0
 3 |   0.00000                           3 |   0
 4 |   0.00000                           4 |   0
 5 |   0.27261                           5 |   0
 6 |   0.00000                           6 |   0
 7 |   0.00000
 8 |   0.22651
 9 |   0.00000
10 |   0.28723
```

```
  print "aupp=", aupp = rand(4,4,'u');
  print "vu1=", vu1 = tri2vec(aupp,"dlow");
  print "vu2=", vu2 = tri2vec(aupp,"slow");
  print "vu3=", vu3 = tri2vec(aupp,"dupp");
  print "vu4=", vu4 = tri2vec(aupp,"supp");
```

```
aupp=
 U |         1         2         3         4
-------------------------------------------
 1 |   0.47579   0.84499   0.59036   0.72836
 2 |         0   0.63452   0.58258   0.50660
 3 |         0         0   0.37701   0.93121
 4 |         0         0         0   0.92912
```

```
vu1=                                    vu2=
   |         1                           Z |   1
--------------                          --------
 1 |   0.47579                           1 |   0
 2 |   0.00000                           2 |   0
 3 |   0.63452                           3 |   0
 4 |   0.00000                           4 |   0
 5 |   0.00000                           5 |   0
 6 |   0.37701                           6 |   0
 7 |   0.00000
 8 |   0.00000
 9 |   0.00000
10 |   0.92912
```

```
vu3=
   |         1
-------------
 1 |   0.47579
 2 |   0.84499
 3 |   0.59036
 4 |   0.72836
 5 |   0.63452
 6 |   0.58258
 7 |   0.50660
 8 |   0.37701
 9 |   0.93121
10 |   0.92912
```

```
vu4=
   |         1
-------------
 1 |   0.84499
 2 |   0.59036
 3 |   0.72836
 4 |   0.58258
 5 |   0.50660
 6 |   0.93121
```

3. SDENS Matrices: Square and Rectangular:

```
print "arsq=", arsq = rand(4,4);
print "v11=", v11 = tri2vec(arsq,"dlow");
print "v12=", v12 = tri2vec(arsq,"slow");
print "v13=", v13 = tri2vec(arsq,"dupp");
print "v14=", v14 = tri2vec(arsq,"supp");
```

```
arsq=
   |         1         2         3         4
-----------------------------------------------
 1 |   0.58966   0.29722   0.39104   0.47243
 2 |   0.67953   0.16809   0.16653   0.87110
 3 |   0.29879   0.93464   0.90047   0.56878
 4 |   0.04955   0.13559   0.51132   0.43320
```

```
v11=
   |         1
-------------
 1 |   0.58966
 2 |   0.67953
 3 |   0.16809
 4 |   0.29879
 5 |   0.93464
 6 |   0.90047
 7 |   0.04955
 8 |   0.13559
 9 |   0.51132
10 |   0.43320
```

```
v12=
   |         1
-------------
 1 |   0.67953
 2 |   0.29879
 3 |   0.93464
 4 |   0.04955
 5 |   0.13559
 6 |   0.51132
```

```
v13=
   |         1
-------------
 1 |   0.58966
 2 |   0.29722
 3 |   0.39104
 4 |   0.47243
 5 |   0.16809
 6 |   0.16653
 7 |   0.87110
 8 |   0.90047
 9 |   0.56878
10 |   0.43320
```

```
v14=
   |         1
-------------
 1 |   0.29722
 2 |   0.39104
 3 |   0.47243
 4 |   0.16653
 5 |   0.87110
 6 |   0.56878
```

```
  print "arlo=", arlo = rand(6,3);
  print "v21=", v21 = tri2vec(arlo,"dlow");
  print "v22=", v22 = tri2vec(arlo,"slow");
  print "v23=", v23 = tri2vec(arlo,"dupp");
  print "v24=", v24 = tri2vec(arlo,"supp");
```

```
arlo=
   |         1         2         3
-----------------------------------
 1 |   0.17611   0.66504   0.40482
 2 |   0.12455   0.45349   0.19955
 3 |   0.57484   0.73847   0.43981
 4 |   0.04937   0.52238   0.34337
 5 |   0.02271   0.71289   0.93706
 6 |   0.44599   0.94694   0.71290
```

```
v21=
   |         1
-------------
 1 |   0.17611
 2 |   0.12455
 3 |   0.45349
 4 |   0.57484
 5 |   0.73847
 6 |   0.43981
 7 |   0.04937
 8 |   0.52238
 9 |   0.34337
10 |   0.00000
11 |   0.02271
12 |   0.71289
13 |   0.93706
14 |   0.00000
15 |   0.00000
```

```
v22=
   |         1
-------------
 1 |   0.12455
 2 |   0.57484
 3 |   0.73847
 4 |   0.04937
 5 |   0.52238
 6 |   0.34337
 7 |   0.02271
 8 |   0.71289
 9 |   0.93706
10 |   0.00000
11 |   0.44599
12 |   0.94694
```

```
v23=
   |         1
-------------
 1 |   0.17611
 2 |   0.66504
 3 |   0.40482
 4 |   0.45349
 5 |   0.19955
 6 |   0.43981
```

```
v24=
   |         1
-------------
 1 |   0.66504
 2 |   0.40482
 3 |   0.19955
```

```
print "arhi=", arhi = rand(3,6);
print "v31=", v31 = tri2vec(arhi,"dlow");
print "v32=", v32 = tri2vec(arhi,"slow");
print "v33=", v33 = tri2vec(arhi,"dupp");
print "v34=", v34 = tri2vec(arhi,"supp");
```

```
arhi=
   |         1         2         3         4         5         6
-----------------------------------------------------------------
 1 |   0.10327   0.17517   0.26891   0.61486   0.42071   0.07174
 2 |   0.35849   0.71143   0.18985   0.14797   0.56184   0.27011
 3 |   0.32520   0.56918   0.04259   0.43921   0.91744   0.52584
```

```
v31=
   |         1
-------------
 1 |   0.10327
 2 |   0.35849
 3 |   0.71143
 4 |   0.32520
 5 |   0.56918
 6 |   0.04259
```

```
v32=
   |         1
-------------
 1 |   0.35849
 2 |   0.32520
 3 |   0.56918
```

```
v33=
   |         1
-------------
 1 |   0.10327
 2 |   0.17517
 3 |   0.26891
 4 |   0.61486
 5 |   0.42071
 6 |   0.07174
 7 |   0.71143
 8 |   0.18985
 9 |   0.14797
10 |   0.56184
11 |   0.27011
12 |   0.04259
13 |   0.43921
14 |   0.91744
15 |   0.52584
```

```
v34=
   |         1
-------------
 1 |   0.17517
 2 |   0.26891
 3 |   0.61486
 4 |   0.42071
 5 |   0.07174
 6 |   0.18985
 7 |   0.14797
 8 |   0.56184
 9 |   0.27011
10 |   0.43921
11 |   0.91744
12 |   0.52584
```

4. SDENS with lower and upper triangular matrices: Note, here it was necessary to extend the `rand` function to rectangular lower and upper triangular matrices.

```
print "alo2=", alo2 = rand(6,3,'l');
print "v51=", v51 = tri2vec(alo2,"dlow");
print "v52=", v52 = tri2vec(alo2,"slow");
print "v53=", v53 = tri2vec(alo2,"dupp");
print "v54=", v54 = tri2vec(alo2,"supp");
```

```
alo2=
 L |         1         2         3
-----------------------------------
 1 |   0.73182         0         0
 2 |   0.18794   0.33133         0
 3 |   0.12156   0.18067   0.27064
 4 |   0.65321   0.42137   0.03798
 5 |   0.27081   0.42773   0.82010
 6 |   0.84345   0.87691   0.26722
```

v51=

	1
1	0.73182
2	0.18794
3	0.33133
4	0.12156
5	0.18067
6	0.27064
7	0.65321
8	0.42137
9	0.03798
10	0.00000
11	0.27081
12	0.42773
13	0.82010
14	0.00000
15	0.00000

v52=

	1
1	0.18794
2	0.12156
3	0.18067
4	0.65321
5	0.42137
6	0.03798
7	0.27081
8	0.42773
9	0.82010
10	0.00000
11	0.84345
12	0.87691

v53=

	1
1	0.73182
2	0.00000
3	0.00000
4	0.33133
5	0.00000
6	0.27064

v54=

Z	1
1	0
2	0
3	0

```
  print "vup2=", aup2 = rand(3,6,'u');
  print "v61=", v61 = tri2vec(aup2,"dlow");
  print "v62=", v62 = tri2vec(aup2,"slow");
  print "v63=", v63 = tri2vec(aup2,"dupp");
  print "v64=", v64 = tri2vec(aup2,"supp");
```

vup2=

U	1	2	3	4	5	6
1	0.30602	0.39705	0.34905	0.76593	0.54340	0.61257
2	0	0.73591	0.37186	0.64565	0.55718	0.87504
3	0	0	0.14843	0.02544	0.81722	0.65822

```
v61=
   |         1
-------------
 1 |   0.30602
 2 |   0.00000
 3 |   0.73591
 4 |   0.00000
 5 |   0.00000
 6 |   0.14843
```

```
v62=
 Z |   1
--------
 1 |   0
 2 |   0
 3 |   0
```

```
v63=
   |         1
-------------
 1 |   0.30602
 2 |   0.39705
 3 |   0.34905
 4 |   0.76593
 5 |   0.54340
 6 |   0.61257
 7 |   0.73591
 8 |   0.37186
 9 |   0.64565
10 |   0.55718
11 |   0.87504
12 |   0.14843
13 |   0.02544
14 |   0.81722
15 |   0.65822
```

```
v64=
   |         1
-------------
 1 |   0.39705
 2 |   0.34905
 3 |   0.76593
 4 |   0.54340
 5 |   0.61257
 6 |   0.37186
 7 |   0.64565
 8 |   0.55718
 9 |   0.87504
10 |   0.02544
11 |   0.81722
12 |   0.65822
```

5. Row and column vector:

```
print "rowv=", vrow = rand(1,5);
print "vr1=", vr1 = tri2vec(vrow,"dlow");
print "vr2=", vr2 = tri2vec(vrow,"slow");
print "vr3=", vr3 = tri2vec(vrow,"dupp");
print "vr4=", vr4 = tri2vec(vrow,"supp");
```

```
rowv=
   |         1         2         3         4         5
-----------------------------------------------------------
 1 |   0.73182   0.90522   0.57600   0.18794   0.33133
```

```
vr1= 0.7318
```

```
vr2= .
```

```
vr3=                                          vr4=
   |         1                                   |         1
--------------                                --------------
 1 |   0.73182                                 1 |   0.90522
 2 |   0.90522                                 2 |   0.57600
 3 |   0.57600                                 3 |   0.18794
 4 |   0.18794                                 4 |   0.33133
 5 |   0.33133
```

```
  print "colv=", vcol = rand(5,1);
  print "vc1=", vc1 = tri2vec(vcol,"dlow");
  print "vc2=", vc2 = tri2vec(vcol,"slow");
  print "vc3=", vc3 = tri2vec(vcol,"dupp");
  print "vc4=", vc4 = tri2vec(vcol,"supp");
```

```
colv=
   |         1
--------------
 1 |   0.69887
 2 |   0.12156
 3 |   0.18067
 4 |   0.27064
 5 |   0.65321
```

```
vc1=                                          vc2=
   |         1                                   |         1
--------------                                --------------
 1 |   0.69887                                 1 |   0.12156
 2 |   0.12156                                 2 |   0.18067
 3 |   0.18067                                 3 |   0.27064
 4 |   0.27064                                 4 |   0.65321
 5 |   0.65321

vc3= 0.6989                                   vc4= .
```

3.3 Function `polychor`

< corr,tau,ccov,tcov,ctco,gof > = polychor(data<,ind<,optn>>)

Purpose: Assuming the $N \times n$ `data` matrix contains the observations n ordinal scaled valiables in its rows. The `polychor` function computes the $n \times n$ matrix of polychoric correlations. In addition, the $n \times m$ matrix of threshold estimates is computed where m is the maximum of levels of the n ordinal variables. Also the $\begin{pmatrix} n \\ 2 \end{pmatrix} \times \begin{pmatrix} n \\ 2 \end{pmatrix}$ asymptotic covariance matrix `ccov` of the polychoric correlations is computed. If not all variables are binary, tests for the distributional restrictions (multivariate normality) are computed using the Satorra-Bentler mean and mean-variance corrected χ^2 values as proposed by Maydeu-Olivares (2003).

Input: data The first input argument is

1. either an $N \times n$ data set (matrix) containing the observations of ordinal variables in its n columns

2. or a $nr \times nc$ cell frequency (contingency) table of two ordinal variables with nr row and nc column levels. (The number of vars is only $n = 2$.)

The first form is the default. To specify the second form of data input the second entry of the options vector `optn[2]` must be set to nonzero.

ind The second input argument is valid only for the first form of data input. The argument `ind` is an index vector specifying a subset of at least two columns for which the correlations should be computed.

optn Only the first four entries of the options vector are used currently:

1. The first entry should be a nonnegative integer specifying the amount of printed output. A zero specifies that no output is printed. Increasing positive values add output.
2. The second (nonzero) entry is necessary when the first input argument `data` contains a $nr \times nc$ cell frequency (contingency) table of two ordinal variables with nr resp. nc levels.
3. This entry specifies whether asymptotic covariances and the Satora-Bentler corrected χ^2 values are computed. A value of zero supresses both, the asymptotic covariances and the Satorra-Bentler χ^2 measures. A value of one suppreses the computation of the mean-variance corrected Satorra-Bentler χ^2 value which for large applications can reduce the computer time significantly.
4. The next entry specifies a column number which is used as a frequency variable. This specification is invalid for frequency table input.
5. This option should not be specified or specified to the default (missing value). This entry specifies a column number which is used as a weight variable. This specification is invalid for frequency table input.

It should be noted that the algorithm for the asymptotic covariances and the Satorra-Bentler χ^2 values needs the computation of fourth order frequencies, i.e. the excution of four nested loops each across the number of variables on each observation of the data set. Therefore, if the data set contains many duplicate observations the computation could become faster when providing a data set with a reduced number of observations which contains a frequency variable. A sketch of this approach is shown in the *Illustration* section at the end of the Newsletter.

Output: corr The first return argument specifies the polychoric correlation(s). For the first form of data input, this is a $n \times n$ symmetric matrix. For the second input argument this is a scalar.

tau The second return argument contains the threshold estimates for the levels of each ordinal variable. The n rows correspond to the ordinal variables, the q columns correspond to the maximum number of levels.

ccov valid only for the first data input form: Returns the $\begin{pmatrix} n \\ 2 \end{pmatrix} \times \begin{pmatrix} n \\ 2 \end{pmatrix}$ asymptotic covariance matrix `ccov` of the polychoric correlations.

tcov valid only for the first data input form: Returns the $p \times p$ asymptotic covariance matrix `tcov` of the p thresholds, where p is the sum of the levels of all n variables.

ctco valid only for the first data input form: Returns the $\begin{pmatrix} n \\ 2 \end{pmatrix} \times p$ asymptotic covariance matrix between polychoric correlations and the p thresholds.

gof This output vector contains some goodness-of-fit values:

`gof[1]` the mean corrected S-B χ^2 value

`gof[2]` the degrees of freedom for the mean corrected S-B χ^2 value

`gof[3]` the p value for the mean corrected S-B χ^2 value

`gof[4]` the mean-variance corrected S-B χ^2 value

`gof[5]` the degrees of freedom for the mean-variance corrected S-B χ^2 value

`gof[6]` the p value for the mean-variance corrected S-B χ^2 value

Restrictions: 1. No string or missing values are permitted in the data input argument.
2.

Relationships: sem(), noharm(), factor(), bivar()

Examples: 1. Tetrachoric correlation matrix (Example by Albert Maydeu):

```
  data = [
#include "..\\tdata\\sin_pau.dat"
         ];
  data = shape(data,.,6);

  ind = [ 1:6 ]; ipri = 3;
  < corr,tau,ccov,tcov,ctco > = polychor(data,ind,ipri);
```

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

                Variable       Value    Nobs  Proportion
                V_1                0      51  0.92727273
                                   1       4  0.07272727

                V_2                0      49  0.89090909
                                   1       6  0.10909091

                V_3                0      40  0.72727273
                                   1      15  0.27272727

                V_4                0      14  0.25454545
                                   1      41  0.74545455

                V_5                0      15  0.27272727
                                   1      40  0.72727273

                V_6                0      25  0.45454545
                                   1      30  0.54545455


                  **********
                  Thresholds
                  **********

                1 V_1   1.45577601
                2 V_2   1.23137719
                3 V_3   0.60458538
                4 V_4  -0.66025393
                5 V_5  -0.60458538
                6 V_6  -0.11418530
```

Polychoric Correlation(s)

	V_1	V_2	V_3	V_4	V_5
V_1	1.0000000				
V_2	0.3075789	1.0000000			
V_3	0.3182734	0.1025720	1.0000000		
V_4	-0.6359546	0.9727632	-0.4823171	1.0000000	
V_5	-0.6120377	0.2021511	0.5692998	0.6064232	1.0000000
V_6	-0.0600628	-0.9910507	0.6541807	-0.6203729	0.2952584

Polychoric Correlation(s)

	V_6
V_6	1.0000000

Indefinite Matrix: 2 Neg. 0 Zero 4 Pos. Eigenvalues

Note that here are no S-B χ^2 values for only binary variables.

Note that the asymptotic covariances with the correlations $[4, 2]$ are set to zero due to the small density estimate:

Asymptotic Covariance Matrix of Correlations

	V_1_1	V_2_1	V_2_2	V_3_1	V_3_2
V_1_1	0.0000000				
V_2_1	0.0000000	6.8911523			
V_2_2	0.0000000	0.0000000	0.0000000		
V_3_1	0.0000000	3.0299803	0.0000000	5.0050583	
V_3_2	0.0000000	3.1134314	0.0000000	1.1977655	4.7723053
V_3_3	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000
V_4_1	0.0000000	3.9267701	0.0000000	1.7736195	1.4199131
V_4_2	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000
V_4_3	0.0000000	0.3919733	0.0000000	-0.6733170	0.9488935
V_4_4	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000
V_5_1	0.0000000	4.0016623	0.0000000	2.0828976	1.6424740
V_5_2	0.0000000	0.7273981	0.0000000	0.4038405	1.5517100
V_5_3	0.0000000	0.9742057	0.0000000	-1.8852524	0.3165638
V_5_4	0.0000000	-0.3862395	0.0000000	0.1151992	0.4891980
V_5_5	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000
V_6_1	0.0000000	-3.4289777	0.0000000	0.1806828	-1.4871574
V_6_2	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000
V_6_3	0.0000000	-1.3214782	0.0000000	-0.7242817	-2.0769837
V_6_4	0.0000000	-0.2753889	0.0000000	-0.7478389	0.1100339
V_6_5	0.0000000	0.1467489	0.0000000	-0.7400205	-0.4472438
V_6_6	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000

Asymptotic Covariance Matrix of Correlations

	V_3_3	V_4_1	V_4_2	V_4_3	V_4_4
V_3_3	0.0000000				

V_4_1	0.0000000	2.8355415			
V_4_2	0.0000000	0.0000000	0.0000000		
V_4_3	0.0000000	0.1013805	0.0000000	2.1573639	
V_4_4	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000
V_5_1	0.0000000	2.7242247	0.0000000	0.4024652	0.0000000
V_5_2	0.0000000	0.3619484	0.0000000	0.6025383	0.0000000
V_5_3	0.0000000	0.9796486	0.0000000	0.7896657	0.0000000
V_5_4	0.0000000	-0.4596787	0.0000000	1.2111667	0.0000000
V_5_5	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000
V_6_1	0.0000000	-2.1792916	0.0000000	-0.7715017	0.0000000
V_6_2	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000
V_6_3	0.0000000	-0.8101757	0.0000000	-0.5875449	0.0000000
V_6_4	0.0000000	-0.4538338	0.0000000	0.7070677	0.0000000
V_6_5	0.0000000	-0.0517763	0.0000000	-0.4946981	0.0000000
V_6_6	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000

Asymptotic Covariance Matrix of Correlations

	V_5_1	V_5_2	V_5_3	V_5_4	V_5_5
V_5_1	3.0246290				
V_5_2	0.2643203	5.1672053			
V_5_3	1.0157202	-0.1679629	2.2824795		
V_5_4	-0.4724068	0.7129292	-0.6100452	1.6245540	
V_5_5	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000
V_6_1	-2.0467157	-0.2427067	-1.6780416	0.1335218	0.0000000
V_6_2	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000
V_6_3	-0.9033592	-0.6563151	-0.1018227	-0.4749758	0.0000000
V_6_4	-0.3887577	-0.0235441	0.1416777	0.5686402	0.0000000
V_6_5	0.2041586	-0.9871726	0.8058345	-1.2471383	0.0000000
V_6_6	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000

Asymptotic Covariance Matrix of Correlations

	V_6_1	V_6_2	V_6_3	V_6_4	V_6_5
V_6_1	5.5223146				
V_6_2	0.0000000	0.0000000			
V_6_3	0.7963563	0.0000000	1.3971679		
V_6_4	-1.1818899	0.0000000	-0.0270297	1.5890567	
V_6_5	-1.1414722	0.0000000	0.4386091	0.9364679	2.4871616
V_6_6	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000

Asymptotic Covariance Matrix of Correlations

	V_6_6
V_6_6	0.0000000

Asymptotic Covariance Matrix of Thresholds

```
           V_1_1      V_2_1      V_3_1      V_4_1      V_5_1
V_1_1   3.5275884
V_2_1   0.3965253  2.7817500
V_3_1   0.3597440  0.1064425  1.7961829
V_4_1  -0.8123436  0.4630779 -0.5426591  1.8437017
V_5_1  -0.7554624  0.1862744  0.5089185  0.7132091  1.7961829
V_6_1  -0.0603229 -0.8031864  0.6651261 -0.6239634  0.3011892
```

Asymptotic Covariance Matrix of Thresholds

```
           V_6_1
V_6_1   1.5782588
```

Here, as above, the asymptotic covariances with the correlations $[4, 2]$ are set to zero due to the small density estimate:

Asymptotic Covariances Between Correlations and Thresholds

```
           V_1_1      V_2_1      V_3_1      V_4_1      V_5_1
V_1_1   0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
V_2_1  -0.2990292 -0.2039940 -0.6544304 -0.6371008 -0.7714572
V_2_2   0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
V_3_1  -0.3122561 -0.8454922 -0.0547561 -0.6630431 -0.0995213
V_3_2  -1.1181136 -0.0767267 -0.0262592 -0.3470638  0.0211036
V_3_3   0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
V_4_1   0.4881952 -0.7911013 -0.5528285 -0.0353381 -0.4832137
V_4_2   0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
V_4_3  -0.5246000 -0.2233592  0.0915205 -0.1080555  0.5621692
V_4_4   0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
V_5_1   0.4847762 -0.9452228 -0.1163229 -0.5294841 -0.0262238
V_5_2  -0.8587961 -0.1511162 -0.0887325  0.3381420  0.0643427
V_5_3   0.9160078 -0.0882955 -0.1611747  0.1158829  0.1611747
V_5_4  -0.8009046  0.0233167  0.4743032  0.1144233  0.0940773
V_5_5   0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
V_6_1   0.0626437  0.4685463  0.3595079  0.1630244  0.6334682
V_6_2   0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
V_6_3   0.5588047  0.7568806 -0.1510259  0.2027964  0.0684822
V_6_4  -0.3252731  0.3039012  0.2101319 -0.1623065  0.0071399
V_6_5   0.6747174  0.3842094 -0.1973161 -0.5296006  0.0782290
V_6_6   0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
```

Asymptotic Covariances Between Correlations and Thresholds

```
           V_6_1
V_1_1   0.0000000
V_2_1   0.0960954
V_2_2   0.0000000
V_3_1   0.3593578
V_3_2   0.4372653
```

```
V_3_3   0.0000000
V_4_1   0.0595033
V_4_2   0.0000000
V_4_3  -0.0214446
V_4_4   0.0000000
V_5_1   0.4673272
V_5_2   0.2326695
V_5_3   0.0703174
V_5_4  -0.1983459
V_5_5   0.0000000
V_6_1  -0.0018214
V_6_2   0.0000000
V_6_3   0.0723762
V_6_4  -0.0716343
V_6_5   0.0028117
V_6_6   0.0000000
```

2. Agresti (1992): Three variables each with 3 levels, nobs=61

```
  agres = [
#include "..\\tdata\\agresti.dat"
         ];

  agres = shape(agres,.,3);
  cnam = [ "co_cl" "co_pe" "cl_pe" ];
  agres = cname(agres,cnam);
  nobs = nrow(agres);
  nvar = ncol(agres);
  print "nrow=", nobs;
  ipri = 3;
  < corr,tau,ccov,tcov,ctco,gof > = polychor(agres,.,ipri);
```

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

            Variable       Value    Nobs  Proportion
            co_cl              0      18  0.29508197
                               1      11  0.18032787
                               2      32  0.52459016

            co_pe              0      20  0.32786885
                               1      12  0.19672131
                               2      29  0.47540984

            cl_pe              0      28  0.45901639
                               1      14  0.22950820
                               2      19  0.31147541
```

```
**********
Thresholds
**********

     1 co_cl  -0.53859850 -0.06167749
     2 co_pe  -0.44580556  0.06167749
     3 cl_pe  -0.10291204  0.49167260


Polychoric Correlation(s)

             co_cl      co_pe      cl_pe
co_cl   1.0000000
co_pe   0.1790111  1.0000000
cl_pe  -0.3202509  0.1056275  1.0000000
Matrix is Positive Definite: Determinant=0.842126
```

The test statistics shows that the hypothesis of joint categorized bivariate normality cannot be rejected at $\alpha = 0.01$:

```
       S-B Mean-Corr. MVN Test: chi^2=7.57 df=9 pval= 0.5780
S-B Mean-Var-Corr. MVN Test: chi^2=6.42673 df=7.64076 pval= 0.5616
                 (See A. Maydeu-Olivares, 2003)


           Asymptotic Covariance Matrix of Correlations

                [co_cl,co_ [co_pe,co_ [co_pe,co_ [cl_pe,co_
[co_cl,co_cl]   0.0000000
[co_pe,co_cl]   0.0000000  1.7050838
[co_pe,co_pe]   0.0000000  0.0000000  0.0000000
[cl_pe,co_cl]   0.0000000  0.0423212  0.0000000  1.5848976
[cl_pe,co_pe]   0.0000000 -0.7755111  0.0000000  0.2686906
[cl_pe,cl_pe]   0.0000000  0.0000000  0.0000000  0.0000000


           Asymptotic Covariance Matrix of Correlations

                    [cl_pe,co_ [cl_pe,cl_
[cl_pe,co_pe]   1.8684276
[cl_pe,cl_pe]   0.0000000  0.0000000


            Asymptotic Covariance Matrix of Thresholds

            co_cl_1    co_cl_2    co_pe_1    co_pe_2    cl_pe_1
co_cl_1   1.7468070
co_cl_2   1.1265763  1.5729694
co_pe_1   0.0129366  0.1700372  1.6890669
co_pe_2   0.1858069  0.2881517  1.0837538  1.5729694
cl_pe_1  -0.1511133 -0.3435559  0.2081131  0.2398088  1.5768549
cl_pe_2  -0.3216315 -0.4620141 -0.0989163 -0.0038183  1.0191197
```

```
        Asymptotic Covariance Matrix of Thresholds

                              cl_pe_2
                   cl_pe_2   1.7159723


    Asymptotic Covariances Between Correlations and Thresholds

                     co_cl_1    co_cl_2    co_pe_1    co_pe_2
[co_cl,co_cl]   0.0000000  0.0000000  0.0000000  0.0000000
[co_pe,co_cl]  -0.0554799 -0.0659489 -0.0606661 -0.0948283
[co_pe,co_pe]   0.0000000  0.0000000  0.0000000  0.0000000
[cl_pe,co_cl]   0.0553091  0.0933981 -0.4906571 -0.1316125
[cl_pe,co_pe]  -0.3668860 -0.3321084 -0.0221411 -0.0409370
[cl_pe,cl_pe]   0.0000000  0.0000000  0.0000000  0.0000000


    Asymptotic Covariances Between Correlations and Thresholds

                              cl_pe_1    cl_pe_2
           [co_cl,co_cl]   0.0000000  0.0000000
           [co_pe,co_cl]  -0.3670009 -0.2590001
           [co_pe,co_pe]   0.0000000  0.0000000
           [cl_pe,co_cl]   0.0292260  0.1860359
           [cl_pe,co_pe]   0.1479196  0.1981278
           [cl_pe,cl_pe]   0.0000000  0.0000000
```

3. Contingency Table Input: Example by Olsson, page 456:

```
tabl = [ 13     6     0 ,
         69   113    22 ,
         41   132   104 ];
optn = [ 2 1 ];
< corr,tau > = polychor(tabl,.,optn);


                          **********
                          Thresholds
                          **********

              1  col_lev -0.68713131  0.66820933
              2  row_lev -1.77438183 -0.13577393

          Polychoric Correlation(s)  0.49137196
```

Transpose the contingency table:

```
tabl = [ 13    69    41 ,
          6   113   132 ,
```

```
          0    22   104 ];
optn = [ 2 1 ];
< corr, tau > = polychor(tabl,.,optn);
```

```
                          **********
                          Thresholds
                          **********

                1  col_lev -1.77438183 -0.13577393
                2  row_lev -0.68713131  0.66820933

             Polychoric Correlation(s)  0.49137196
```

4. LOT Data (Scheier & Carver, 1985): Data Matrix Input: Eight 5-level variabels:

```
  options NOECHO;
  lot8 = [
#include "..\\tdata\\lot.dat"
           ];
  options ECHO;


  lot = shape(lot8,.,8);
  cnam = [ "i1" "i4" "i5" "i11" "i3" "i8" "i9" "i12" ];
  lot = cname(lot,cnam);
  print cov = bivar(lot,"cov");
  ipri = 3;
  < corr,tau,ccov,tcov,ctco,gof > = polychor(lot,.,ipri);


  print "Pol. Corrs=", corr;
  print "Thresholds=", tau;
  print "ACOV Corrs=", ccov;
  print "ACOV Thres=", tcov;
  print "ACOV (Corrs,Thres)=", ctco;
```

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

            Variable       Value    Nobs  Proportion
            i1                 0      13  0.03341902
                               1      84  0.21593830
                               2     125  0.32133676
                               3     129  0.33161954
                               4      38  0.09768638

            i4                 0      12  0.03084833
                               1      56  0.14395887
                               2     121  0.31105398
```

		3	149	0.38303342
		4	51	0.13110540
	i5	0	11	0.02827763
		1	46	0.11825193
		2	96	0.24678663
		3	172	0.44215938
		4	64	0.16452442
	i11	0	21	0.05398458
		1	52	0.13367609
		2	121	0.31105398
		3	165	0.42416452
		4	30	0.07712082
	i3	0	29	0.07455013
		1	139	0.35732648
		2	108	0.27763496
		3	89	0.22879177
		4	24	0.06169666
	i8	0	70	0.17994859
		1	172	0.44215938
		2	85	0.21850900
		3	48	0.12339332
		4	14	0.03598972
	i9	0	75	0.19280206
		1	180	0.46272494
		2	79	0.20308483
		3	44	0.11311054
		4	11	0.02827763
	i12	0	69	0.17737789
		1	186	0.47814910
		2	60	0.15424165
		3	58	0.14910026
		4	16	0.04113111

****
Thresholds
****

1	i1	-1.83276143	-0.67651357	0.17814153	1.29484771
2	i4	-1.86846944	-0.93533747	-0.03544818	1.12118112
3	i5	-1.90673221	-1.05143443	-0.27068619	0.97603151
4	i11	-1.60738861	-0.88654975	-0.00322189	1.42470796
5	i3	-1.44271677	-0.17159847	0.55195841	1.54068570
6	i8	-0.91556101	0.31102177	0.99699679	1.79924808
7	i9	-0.86761683	0.40028580	1.07410343	1.90673221

```
      8 i12  -0.92540400  0.40028580  0.87704399  1.73770831
```

```
                  Polychoric Correlation(s)

              i1         i4         i5        i11         i3
i1    1.0000000
i4    0.5577219  1.0000000
i5    0.4859851  0.5877967  1.0000000
i11   0.2803616  0.3723243  0.2652058  1.0000000
i3   -0.1636465 -0.2297360 -0.2780625 -0.1216389  1.0000000
i8   -0.3080950 -0.3925849 -0.3718812 -0.1979306  0.5536118
i9   -0.2654568 -0.3005806 -0.3364019 -0.2294538  0.5587520
i12  -0.2585184 -0.3656555 -0.3312995 -0.3016573  0.4891194
```

```
                  Polychoric Correlation(s)

                       i8         i9        i12
            i8    1.0000000
            i9    0.7644719  1.0000000
            i12   0.5980033  0.5932275  1.0000000
      Matrix is Positive Definite: Determinant=0.0414436
```

The test statistics shows that the hypothesis of joint categorized bivariate normality is to be rejected at $\alpha = 0.01$:

```
    S-B Mean-Corr. MVN Test: chi^2=1070.92 df=420 pval= 0.0000
S-B Mean-Var-Corr. MVN Test: chi^2=252.142 df=98.8868 pval= 0.0000
                (See A. Maydeu-Olivares, 2003)
```

```
           Asymptotic Covariance Matrix of Correlations

              [i1,i1]    [i4,i1]    [i4,i4]    [i5,i1]    [i5,i4]
 [i1,i1]     0.0000000
 [i4,i1]     0.0000000  0.7498818
 [i4,i4]     0.0000000  0.0000000  0.0000000
 [i5,i1]     0.0000000  0.4048918  0.0000000  0.8915829
 [i5,i4]     0.0000000  0.2292698  0.0000000  0.3972917  0.8766739
 [i5,i5]     0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
 [i11,i1]    0.0000000  0.2405817  0.0000000  0.1779357  0.0593687
 [i11,i4]    0.0000000 -0.0075237  0.0000000  0.0184799  0.1356258
 [i11,i5]    0.0000000  0.0262209  0.0000000  0.2540914  0.3162464
 [i11,i11]   0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
 [i3,i1]     0.0000000 -0.2863215  0.0000000 -0.2368921 -0.1296937
 [i3,i4]     0.0000000 -0.1427675  0.0000000 -0.1182177 -0.3459550
 [i3,i5]     0.0000000 -0.0589008  0.0000000 -0.1429217 -0.1225855
 [i3,i11]    0.0000000 -0.1157351  0.0000000 -0.0248331 -0.0210622
 [i3,i3]     0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
 [i8,i1]     0.0000000 -0.3688731  0.0000000 -0.3287197 -0.1999898
```

[i8,i4]	0.0000000	-0.1967475	0.0000000	-0.1977907	-0.2240274
[i8,i5]	0.0000000	-0.1810822	0.0000000	-0.3079560	-0.3598340
[i8,i11]	0.0000000	-0.1290327	0.0000000	-0.1371755	-0.1056421
[i8,i3]	0.0000000	-0.0760800	0.0000000	-0.1111845	0.0488294
[i8,i8]	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000
[i9,i1]	0.0000000	-0.3105226	0.0000000	-0.3479250	-0.1866036
[i9,i4]	0.0000000	-0.1942187	0.0000000	-0.1838424	-0.3812420
[i9,i5]	0.0000000	-0.0949266	0.0000000	-0.1418713	-0.1298395
[i9,i11]	0.0000000	-0.0967689	0.0000000	-0.1491504	-0.1158284
[i9,i3]	0.0000000	-0.0188932	0.0000000	-0.1152172	-0.0857580
[i9,i8]	0.0000000	0.1037371	0.0000000	0.0430391	0.1894119
[i9,i9]	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000
[i12,i1]	0.0000000	-0.3648634	0.0000000	-0.3505401	-0.2171904
[i12,i4]	0.0000000	-0.1133516	0.0000000	-0.1697314	-0.3161368
[i12,i5]	0.0000000	-0.1340803	0.0000000	-0.2826039	-0.4107149
[i12,i11]	0.0000000	-0.0386469	0.0000000	-0.0924975	-0.1142642
[i12,i3]	0.0000000	0.0123570	0.0000000	-4.28e-004	0.1104869
[i12,i8]	0.0000000	0.0057381	0.0000000	0.0135609	-0.0328625
[i12,i9]	0.0000000	0.0651554	0.0000000	0.0685473	0.1110437
[i12,i12]	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000

Asymptotic Covariance Matrix of Correlations

	[i5,i5]	[i11,i1]	[i11,i4]	[i11,i5]	[i11,i11]
[i5,i5]	0.0000000				
[i11,i1]	0.0000000	1.2349966			
[i11,i4]	0.0000000	0.6028863	1.0754266		
[i11,i5]	0.0000000	0.4894728	0.6353338	1.2712612	
[i11,i11]	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000
[i3,i1]	0.0000000	-0.0741255	-0.0812807	-0.0162155	0.0000000
[i3,i4]	0.0000000	-0.1094284	-0.0054655	-0.0046508	0.0000000
[i3,i5]	0.0000000	0.0179225	0.0752562	-0.0482004	0.0000000
[i3,i11]	0.0000000	-0.2063255	-0.1102967	-0.2718515	0.0000000
[i3,i3]	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000
[i8,i1]	0.0000000	-0.1700084	-0.1059726	-0.1481728	0.0000000
[i8,i4]	0.0000000	-0.1282863	-0.0257298	-0.0955624	0.0000000
[i8,i5]	0.0000000	-0.1182317	-0.0653591	-0.2327645	0.0000000
[i8,i11]	0.0000000	-0.4569260	-0.4436340	-0.4746940	0.0000000
[i8,i3]	0.0000000	-0.0921453	-0.0499180	-0.1971167	0.0000000
[i8,i8]	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000
[i9,i1]	0.0000000	-0.1795266	-0.1427058	-0.2135695	0.0000000
[i9,i4]	0.0000000	-0.1545288	-0.2010502	-0.2127320	0.0000000
[i9,i5]	0.0000000	-0.1499175	-0.1345073	-0.2502288	0.0000000
[i9,i11]	0.0000000	-0.3714228	-0.3622265	-0.4314383	0.0000000
[i9,i3]	0.0000000	-0.0623710	-0.1912168	-0.1794872	0.0000000
[i9,i8]	0.0000000	0.1087765	0.0828143	0.0838046	0.0000000
[i9,i9]	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000
[i12,i1]	0.0000000	-0.2950317	-0.1309337	-0.1790665	0.0000000
[i12,i4]	0.0000000	-0.1287031	-0.1990726	-0.1960277	0.0000000
[i12,i5]	0.0000000	-0.1332658	-0.1528485	-0.3478692	0.0000000

[i12,i11]	0.0000000	-0.3276557	-0.4006596	-0.4661341	0.0000000
[i12,i3]	0.0000000	-0.0555323	-0.0951417	-0.0910821	0.0000000
[i12,i8]	0.0000000	0.0340219	-0.0126943	-0.0070939	0.0000000
[i12,i9]	0.0000000	0.0571162	0.0270429	0.0303823	0.0000000
[i12,i12]	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000

Asymptotic Covariance Matrix of Correlations

	[i3,i1]	[i3,i4]	[i3,i5]	[i3,i11]	[i3,i3]
[i3,i1]	1.5319365				
[i3,i4]	0.8128574	1.4866689			
[i3,i5]	0.7079951	0.7557139	1.4135463		
[i3,i11]	0.4498536	0.4510179	0.3162209	1.4843595	
[i3,i3]	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000
[i8,i1]	0.6319901	0.1646241	0.0668746	0.0780265	0.0000000
[i8,i4]	0.1725779	0.5031405	0.1935542	0.0980560	0.0000000
[i8,i5]	0.1029051	0.2651749	0.5031759	-0.0717984	0.0000000
[i8,i11]	0.0874087	0.1118584	-0.0943589	0.6899845	0.0000000
[i8,i3]	-0.4157167	-0.5249943	-0.2542621	-0.1696359	0.0000000
[i8,i8]	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000
[i9,i1]	0.6288916	0.2897875	0.1308063	0.1174729	0.0000000
[i9,i4]	0.3047282	0.6672341	0.2015062	-0.0144276	0.0000000
[i9,i5]	0.1424844	0.2196218	0.5694547	-0.0660590	0.0000000
[i9,i11]	0.1077278	-0.0358359	-0.1063441	0.6682305	0.0000000
[i9,i3]	-0.3390711	-0.3597474	-0.2645399	-0.2451779	0.0000000
[i9,i8]	-0.1142829	-0.2447004	-0.0929038	0.0459624	0.0000000
[i9,i9]	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000
[i12,i1]	0.6820948	0.3725107	0.2585437	0.1556770	0.0000000
[i12,i4]	0.3713219	0.7292402	0.3828550	0.1240947	0.0000000
[i12,i5]	0.2912302	0.4559998	0.6211546	0.0777316	0.0000000
[i12,i11]	0.1260872	0.0928542	0.0173656	0.6809596	0.0000000
[i12,i3]	-0.3721283	-0.5376121	-0.3162389	-0.2844092	0.0000000
[i12,i8]	-0.1144864	-0.1788057	-0.0944707	7.98e-004	0.0000000
[i12,i9]	-0.0603375	-0.1336428	0.0170095	0.0572206	0.0000000
[i12,i12]	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000

Asymptotic Covariance Matrix of Correlations

	[i8,i1]	[i8,i4]	[i8,i5]	[i8,i11]	[i8,i3]
[i8,i1]	1.2549031				
[i8,i4]	0.5695256	1.2485521			
[i8,i5]	0.4459450	0.5777676	1.3594328		
[i8,i11]	0.3345971	0.3138431	0.2750605	1.3873143	
[i8,i3]	-0.0111589	-0.0345924	-0.1794221	-0.0475399	1.1789095
[i8,i8]	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000
[i9,i1]	0.9574562	0.4901448	0.3912273	0.3664686	-0.0533193
[i9,i4]	0.5160140	0.8574390	0.6183569	0.3033155	-0.2356927
[i9,i5]	0.3622442	0.5420093	0.9694835	0.2413860	-0.1391496
[i9,i11]	0.3330091	0.2352393	0.2096605	1.1308393	0.0633053

[i9,i3]	0.0064622	-0.0913039	-0.0951166	0.0375120	0.7707556
[i9,i8]	-0.1279645	0.0066725	-0.1835236	-0.1152053	0.3628543
[i9,i9]	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000
[i12,i1]	0.8085333	0.3458511	0.2415704	0.2663639	-0.0392306
[i12,i4]	0.3120361	0.6174747	0.1931184	0.1787972	-0.0860377
[i12,i5]	0.2321602	0.2006386	0.5922744	0.1411479	-0.0817471
[i12,i11]	0.1859681	0.1004781	0.0720642	0.8481616	0.0834291
[i12,i3]	0.0033380	-0.0536168	-0.0604514	0.0269829	0.5670552
[i12,i8]	-0.1719014	-0.2909316	-0.2577605	-0.2306699	0.3070131
[i12,i9]	-0.2173663	-0.1700851	-0.3367805	-0.1504180	0.3273647
[i12,i12]	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000

Asymptotic Covariance Matrix of Correlations

	[i8,i8]	[i9,i1]	[i9,i4]	[i9,i5]	[i9,i11]
[i8,i8]	0.0000000				
[i9,i1]	0.0000000	1.2861276			
[i9,i4]	0.0000000	0.6880367	1.4253940		
[i9,i5]	0.0000000	0.5510879	0.7569680	1.3289465	
[i9,i11]	0.0000000	0.3201945	0.2989657	0.1577934	1.5098476
[i9,i3]	0.0000000	0.0199866	-0.0731508	-0.1437007	0.0518889
[i9,i8]	0.0000000	-0.2144186	-0.4266682	-0.2748334	0.0730791
[i9,i9]	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000
[i12,i1]	0.0000000	0.8468306	0.4568463	0.3467562	0.2886922
[i12,i4]	0.0000000	0.4251613	0.6832509	0.3915667	0.1923690
[i12,i5]	0.0000000	0.3665784	0.4543963	0.6333431	0.1487443
[i12,i11]	0.0000000	0.2401755	0.1684544	0.1075715	0.8721981
[i12,i3]	0.0000000	0.0328808	-0.0816266	0.0349870	0.1052581
[i12,i8]	0.0000000	-0.2643480	-0.2829903	-0.3865786	-0.1365622
[i12,i9]	0.0000000	-0.2364777	-0.3704358	-0.3043424	-0.2001257
[i12,i12]	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000

Asymptotic Covariance Matrix of Correlations

	[i9,i3]	[i9,i8]	[i9,i9]	[i12,i1]	[i12,i4]
[i9,i3]	1.0756640				
[i9,i8]	0.0884760	0.7916846			
[i9,i9]	0.0000000	0.0000000	0.0000000		
[i12,i1]	0.0375875	-0.1822188	0.0000000	1.5554362	
[i12,i4]	-0.0158532	-0.0871089	0.0000000	0.8841297	1.4562270
[i12,i5]	0.0513678	-0.2717609	0.0000000	0.7433665	0.8757251
[i12,i11]	0.1498875	0.0138580	0.0000000	0.4513734	0.4015361
[i12,i3]	0.5463665	0.1629711	0.0000000	-0.0749384	-0.1777648
[i12,i8]	0.2660921	0.1937387	0.0000000	-0.4404389	-0.4746750
[i12,i9]	0.2942678	0.2188327	0.0000000	-0.3104457	-0.2945387
[i12,i12]	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000

Asymptotic Covariance Matrix of Correlations

	[i12,i5]	[i12,i11]	[i12,i3]	[i12,i8]	[i12,i9]
[i12,i5]	1.4716725				
[i12,i11]	0.3965216	1.3349460			
[i12,i3]	-0.2639191	0.0308148	1.2336999		
[i12,i8]	-0.4820321	-0.0704266	0.3689865	1.2471911	
[i12,i9]	-0.3500347	0.0075353	0.3329128	0.8182657	1.0101311
[i12,i12]	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000

Asymptotic Covariance Matrix of Correlations

	[i12,i12]
[i12,i12]	0.0000000

Asymptotic Covariance Matrix of Thresholds

	i1_1	i1_2	i1_3	i1_4	i4_1
i1_1	5.8373174				
i1_2	1.0626482	1.8586562			
i1_3	0.4911705	0.8590962	1.5890303		
i1_4	0.2543806	0.4449320	0.8229700	2.9615754	
i4_1	2.2823982	0.5825741	0.3903408	0.2508539	6.1659873
i4_2	0.9025218	0.7875594	0.5132046	0.2685633	1.4191857
i4_3	0.5793321	0.6475705	0.6755321	0.4284170	0.5713003
i4_4	0.2767956	0.3699293	0.4647483	0.6315101	0.2729582
i5_1	1.4042611	0.5323404	0.2751276	0.2471719	3.2259987
i5_2	0.6165333	0.6983023	0.4982926	0.3614716	1.1647336
i5_3	0.4391108	0.6286990	0.5053025	0.3078690	0.6028571
i5_4	0.2983097	0.2929058	0.4366266	0.7666782	0.1451718
i11_1	1.0398028	0.2781292	0.1800996	0.2788703	0.7922174
i11_2	0.5852027	0.3548558	0.2756640	0.2285828	0.2396407
i11_3	0.3912509	0.2766564	0.2672518	0.1475817	0.2790491
i11_4	6.14e-004	0.1950047	0.2771373	0.5224289	-0.0190345
i3_1	0.2528150	-0.1282801	0.0209020	-0.0176704	0.2896183
i3_2	-0.3177319	-0.3893005	-0.1478299	0.0914015	-0.2988788
i3_3	-0.6278152	-0.2559177	-0.1053300	-0.0016773	-0.7020089
i3_4	-1.4753927	-0.4667789	-0.0700849	0.2026223	-1.8981233
i8_1	-0.3081392	-0.2301891	-0.3231229	-0.2363890	-0.1631477
i8_2	-0.7352681	-0.4335750	-0.2602154	-0.0533111	-0.6279484
i8_3	-1.4136392	-0.4853074	-0.1245440	-0.1876649	-1.0781074
i8_4	-1.5439652	-0.2570880	-0.0836630	-0.2577730	-2.6002086
i9_1	-0.3163338	-0.2278578	-0.1865477	-0.2543371	-0.0422863
i9_2	-0.7059094	-0.3428810	-0.1871625	-0.0845746	-0.4878925
i9_3	-1.2587732	-0.3356856	-0.1931813	-0.2242914	-0.8737797
i9_4	-1.9377158	-0.2822433	-0.0730036	-0.2471719	-2.0862044
i12_1	-0.3065016	-0.2245146	-0.3116929	-0.3587685	-0.3022523
i12_2	-0.6120615	-0.3208818	-0.1160453	-0.0036410	-0.7886694
i12_3	-0.8305678	-0.3444557	0.0055776	-0.0674241	-1.3210010
i12_4	-2.5346835	-0.6442465	-0.1388067	0.0738771	-3.1438916

Asymptotic Covariance Matrix of Thresholds

	i4_2	i4_3	i4_4	i5_1	i5_2
i4_2	2.1739013				
i4_3	0.8751148	1.5715138			
i4_4	0.4181159	0.7508442	2.5159162		
i5_1	1.0902498	0.4633813	0.2689517	6.5478422	
i5_2	1.0015419	0.5984979	0.3933231	1.6230704	2.3736265
i5_3	0.5515239	0.5811428	0.4416350	0.6885928	1.0070187
i5_4	0.2895030	0.4708983	1.1024107	0.2898561	0.4238942
i11_1	0.7583003	0.3409911	0.3034432	0.8710959	0.4008090
i11_2	0.6758915	0.3958251	0.2947658	0.2852420	0.3037264
i11_3	0.4024307	0.3999047	0.1645706	0.2500854	0.2126058
i11_4	0.2929303	0.3824225	0.5068984	0.2328225	0.1855752
i3_1	-0.1465637	-0.0498766	-0.1027089	0.0506783	-0.1787826
i3_2	-0.3139661	-0.2890803	0.0622537	-0.0757747	-0.3309608
i3_3	-0.3567637	-0.3406292	0.0417928	-0.4407218	-0.3740704
i3_4	-0.6397350	-0.2827712	-0.1137693	-1.4085102	-0.6883786
i8_1	-0.1611516	-0.3197525	-0.3141738	-0.1481482	-0.2671067
i8_2	-0.4542956	-0.4169294	-0.1675777	-0.2968301	-0.4555218
i8_3	-0.8701689	-0.5015012	-0.1557329	-0.5308789	-0.5036919
i8_4	-0.8271657	-0.5054988	-0.2804869	-1.8091084	-0.5593719
i9_1	-0.1133700	-0.1987394	-0.2279820	-0.1624401	-0.2858965
i9_2	-0.3137245	-0.4184043	-0.1170652	-0.3460213	-0.4064930
i9_3	-0.6852354	-0.4108486	-0.1731130	-0.6100625	-0.5468448
i9_4	-0.7821443	-0.3638477	-0.0824594	-2.2597772	-0.7586476
i12_1	-0.1942933	-0.3354137	-0.2301896	-0.1451804	-0.2632256
i12_2	-0.6118429	-0.3833835	-0.1170652	-0.4537894	-0.4673223
i12_3	-0.7373220	-0.2860137	0.0132658	-0.7171073	-0.6663100
i12_4	-0.8155034	-0.3091432	-0.0133885	-2.0472901	-0.8456224

Asymptotic Covariance Matrix of Thresholds

	i5_3	i5_4	i11_1	i11_2	i11_3
i5_3	1.6132635				
i5_4	0.6790868	2.2390780			
i11_1	0.2280852	0.2323751	4.2504375		
i11_2	0.1808905	0.0774500	1.4855979	2.1020026	
i11_3	0.2629967	0.2839408	0.6188420	0.8756120	1.5708023
i11_4	0.2680984	0.5786576	0.2626806	0.3716721	0.6667604
i3_1	-0.1615788	-0.3850094	-0.0941290	-0.0976994	-0.0668917
i3_2	-0.2903686	-0.2206419	-0.2427720	-0.1584925	-0.1603784
i3_3	-0.3230225	-0.1087665	-0.1300540	-0.0778641	-0.1061869
i3_4	-0.3601779	-0.0808378	-0.5209361	-0.1173060	-0.0545598
i8_1	-0.2683375	-0.6123225	-0.2484125	-0.1504985	-0.1205981
i8_2	-0.4780091	-0.2507159	-0.4358589	-0.3117560	-0.1981589
i8_3	-0.4300421	-0.1368225	-0.4496215	-0.4470102	-0.2145222
i8_4	-0.2953793	0.0914286	-0.3690951	-0.2864990	-0.1644834
i9_1	-0.3539402	-0.4039586	-0.0898345	-0.1420499	-0.1742348
i9_2	-0.3684144	-0.2267153	-0.3035474	-0.2035900	-0.2830061

i9_3	-0.2794422	-0.1411726	-0.7358737	-0.5827303	-0.2752316
i9_4	-0.2758615	0.0304678	-0.8710959	-0.5799542	-0.1506138
i12_1	-0.3377972	-0.4249250	-0.1555984	-0.2184109	-0.3076161
i12_2	-0.3139572	-0.2267153	-0.5583046	-0.3332078	-0.3005056
i12_3	-0.1943131	-0.0830892	-0.2595190	-0.2851806	-0.2632663
i12_4	-0.3569308	0.1609753	-0.5683672	-0.2163104	-0.2208133

Asymptotic Covariance Matrix of Thresholds

	i11_4	i3_1	i3_2	i3_3	i3_4
i11_4	3.4042713				
i3_1	-0.6010314	3.4748452			
i3_2	-0.0019765	0.7646139	1.5877074		
i3_3	-0.0889869	0.4486317	0.9315758	1.7562224	
i3_4	0.3138311	0.2681095	0.5567246	1.0495467	3.9054850
i8_1	-0.2440657	0.9583579	0.5425999	0.2669849	0.2671014
i8_2	-0.0157515	0.3340040	0.7309278	0.6771157	0.4960909
i8_3	-0.0572484	0.0467647	0.5598073	0.8035861	0.8852103
i8_4	0.2069650	-0.2206836	0.3346948	0.4682531	1.6388816
i9_1	-0.2088149	0.8267889	0.5877361	0.4052423	0.2025985
i9_2	-0.1126999	0.2967611	0.7080759	0.7351570	0.5007426
i9_3	-0.0985177	0.0081629	0.5472962	0.7710327	0.7167919
i9_4	0.3160742	-0.0506783	0.3786139	0.6723986	1.4085102
i12_1	-0.6618636	0.9021393	0.4829433	0.3187569	0.2645217
i12_2	-0.2575466	0.2472160	0.5660042	0.6128840	0.5007426
i12_3	-0.1117490	0.1018922	0.4565284	0.6494654	0.7335384
i12_4	0.1545131	-0.1670674	0.2900737	0.5407711	1.4403245

Asymptotic Covariance Matrix of Thresholds

	i8_1	i8_2	i8_3	i8_4	i9_1
i8_1	2.1439645				
i8_2	0.6819080	1.6271390			
i8_3	0.4504426	1.0748263	2.2746258		
i8_4	0.3122484	0.7450735	1.5767788	5.5510960	
i9_1	1.2705432	0.6259428	0.3850530	0.2017953	2.0758245
i9_2	0.5352132	1.0719210	0.8239142	0.6338094	0.6587179
i9_3	0.3453412	0.8214509	1.3819261	1.3090487	0.4443104
i9_4	0.1481482	0.5056310	1.1849150	3.8169348	0.3073694
i12_1	1.0772821	0.5482178	0.3258251	0.1855175	1.0001753
i12_2	0.3223319	0.8515216	0.8239142	0.7221160	0.3782548
i12_3	0.1196535	0.5736854	1.0221184	1.1179616	0.2166730
i12_4	-0.1245936	0.3033521	0.7750516	1.6320568	0.1155466

Asymptotic Covariance Matrix of Thresholds

	i9_2	i9_3	i9_4	i12_1	i12_2
i9_2	1.6653752				
i9_3	1.1233087	2.4178790			

```
i9_4    0.7770935  1.6726640  6.5478422
i12_1   0.5576869  0.3422451  0.1451804  2.1587264
i12_2   0.8690948  0.7805894  0.6693255  0.6382440  1.6653752
i12_3   0.5529357  0.6986306  0.8632334  0.4779158  1.2470297
i12_4   0.4346865  0.7467953  1.5970946  0.3183560  0.8306889


          Asymptotic Covariance Matrix of Thresholds

                        i12_3      i12_4
      i12_3   2.0887426
      i12_4   1.3913826  5.0759601
```

The output of asymptotic covariances between correlations and thresholds is skipped due to space.

4 Illustration

4.1 Using FREQ Variable in `polychor`

For very large data sets with many duplicate observations the use of a frequency variable can speed up the computation of the asymptotic covariances of polychoric correlations. Here we show a small sketch of the approach even though this data set is much too small to show an improvement in the computer time. We compress the Agresti data set so that there will be no duplicate observations. There should not be more than $3^3 = 27$ obeservations in the new data set.

```
  agres = [
#include "..\\tdata\\agresti.dat"
          ];

  agres = shape(agres,.,3);
  cnam = [ "co_cl" "co_pe" "cl_pe" ];
  agres = cname(agres,cnam);
  nobs = nrow(agres);
  nvar = ncol(agres);
  print "nrow=", nobs;

  key = cons(nobs,1,.);
  for (i = 1; i <= nobs; i++)
  key[i] = (agres[i,1]*10. + agres[i,2])*10. + agres[i,3];
  kind = key[<:];
  /* print "Sorting key=", tmp = kind -> key; */
  agre2 = agres[kind,];
  print "Sorted data agre2=", agre2;
```

```
   |   1   2   3
---------------
 1 |   0   0   0
 2 |   0   0   1
 3 |   0   0   2
 4 |   0   0   2
 5 |   0   0   2
 6 |   0   0   2
 7 |   0   1   0
 8 |   0   1   0
 9 |   0   1   0
10 |   0   1   1
11 |   0   1   2
12 |   0   2   0
13 |   0   2   0
14 |   0   2   0
15 |   0   2   1
16 |   0   2   2
17 |   0   2   2
18 |   0   2   2
19 |   1   0   0
20 |   1   0   0
21 |   1   0   2
22 |   1   0   2
23 |   1   0   2
24 |   1   1   0
25 |   1   1   1
26 |   1   2   1
27 |   1   2   1
28 |   1   2   2
29 |   1   2   2
30 |   2   0   0
31 |   2   0   0
32 |   2   0   0
33 |   2   0   0
34 |   2   0   0
35 |   2   0   0
36 |   2   0   0
37 |   2   0   0
38 |   2   0   1
39 |   2   1   0
40 |   2   1   0
41 |   2   1   1
42 |   2   1   2
43 |   2   1   2
44 |   2   2   0
45 |   2   2   0
46 |   2   2   0
47 |   2   2   0
48 |   2   2   0
49 |   2   2   0
50 |   2   2   0
51 |   2   2   0
52 |   2   2   1
53 |   2   2   1
54 |   2   2   1
55 |   2   2   1
56 |   2   2   1
57 |   2   2   1
58 |   2   2   2
59 |   2   2   2
60 |   2   2   2
61 |   2   2   2
```

```
  agre3 = cons(27,4,.);
  i = ir = 1;
lrep:
  match = agre2[i,];
  for (j = i+1; j <= nobs; j++)
  if (agre2[j,] != match) break;
  freq = j - i;
  /* options debug="excon*=4"; */
  agre3[ir,] = freq -> match;
  i = j;
  if (i < nobs) { ir++; goto lrep; }
  agre3 = agre3[1:ir,];
  print "New data agre3=", agre3;
  tobs = agre3[+,1];
  print "nobs=",nobs,tobs;
```

The new data set: the first column contains the frequencies:

```
New data agre3=
   |         1         2         3         4
-------------------------------------------
 1 |   1.00000   0.00000   0.00000   0.00000
 2 |   1.00000   0.00000   0.00000   1.00000
 3 |   4.00000   0.00000   0.00000   2.00000
 4 |   3.00000   0.00000   1.00000   0.00000
 5 |   1.00000   0.00000   1.00000   1.00000
 6 |   1.00000   0.00000   1.00000   2.00000
 7 |   3.00000   0.00000   2.00000   0.00000
 8 |   1.00000   0.00000   2.00000   1.00000
 9 |   3.00000   0.00000   2.00000   2.00000
10 |   2.00000   1.00000   0.00000   0.00000
11 |   3.00000   1.00000   0.00000   2.00000
12 |   1.00000   1.00000   1.00000   0.00000
13 |   1.00000   1.00000   1.00000   1.00000
14 |   2.00000   1.00000   2.00000   1.00000
15 |   2.00000   1.00000   2.00000   2.00000
16 |   8.00000   2.00000   0.00000   0.00000
17 |   1.00000   2.00000   0.00000   1.00000
18 |   2.00000   2.00000   1.00000   0.00000
19 |   1.00000   2.00000   1.00000   1.00000
20 |   2.00000   2.00000   1.00000   2.00000
21 |   8.00000   2.00000   2.00000   0.00000
22 |   6.00000   2.00000   2.00000   1.00000
23 |   4.00000   2.00000   2.00000   2.00000

nobs= 61 61.000
```

The results should agree with those of the raw data set `agres` above:

```
cnam = [ "freq" "co_cl" "co_pe" "cl_pe" ];
agre3 = cname(agre3,cnam);
print "New data agre3=", agre3;

ind = [ 2:4 ];
popt = [ 3 0 2 1 ];
< corr,tau,ccov,tcov,ctco > = polychor(agre3,ind,popt);
```

```
          ***********************************************
          Sums of Weights and Frequencies for Class Levels
          ***********************************************

         Variable       Value    Nobs  Proportion      Freq
         co_cl              0       9  0.14754098        18
                            1       6  0.09836066        11
                            2       8  0.13114754        32

         co_pe              0       7  0.11475410        20
```

```
                    1       8  0.13114754        12
                    2       8  0.13114754        29

cl_pe               0       8  0.13114754        28
                    1       8  0.13114754        14
                    2       7  0.11475410        19


                     **********
                     Thresholds
                     **********

               1 co_cl  -0.53859850 -0.06167749
               2 co_pe  -0.44580556  0.06167749
               3 cl_pe  -0.10291204  0.49167260


                Polychoric Correlation(s)

                     co_cl      co_pe      cl_pe
          co_cl   1.0000000
          co_pe   0.1790111  1.0000000
          cl_pe  -0.3202509  0.1056275  1.0000000
        Matrix is Positive Definite: Determinant=0.842126


       S-B Mean-Corr. MVN Test: chi^2=7.57 df=9 pval= 0.5780
S-B Mean-Var-Corr. MVN Test: chi^2=6.42673 df=7.64076 pval= 0.5616
                  (See A. Maydeu-Olivares, 2003)


            Asymptotic Covariance Matrix of Correlations

                  [co_cl,co_ [co_pe,co_ [co_pe,co_ [cl_pe,co_
     [co_cl,co_cl]   0.0000000
     [co_pe,co_cl]   0.0000000  1.7050838
     [co_pe,co_pe]   0.0000000  0.0000000  0.0000000
     [cl_pe,co_cl]   0.0000000  0.0423212  0.0000000  1.5848976
     [cl_pe,co_pe]   0.0000000 -0.7755111  0.0000000  0.2686906
     [cl_pe,cl_pe]   0.0000000  0.0000000  0.0000000  0.0000000


            Asymptotic Covariance Matrix of Correlations

                        [cl_pe,co_ [cl_pe,cl_
           [cl_pe,co_pe]   1.8684276
           [cl_pe,cl_pe]   0.0000000  0.0000000


             Asymptotic Covariance Matrix of Thresholds

              co_cl_1    co_cl_2    co_pe_1    co_pe_2    cl_pe_1
```

```
co_cl_1   1.7468070
co_cl_2   1.1265763  1.5729694
co_pe_1   0.0129366  0.1700372  1.6890669
co_pe_2   0.1858069  0.2881517  1.0837538  1.5729694
cl_pe_1  -0.1511133 -0.3435559  0.2081131  0.2398088  1.5768549
cl_pe_2  -0.3216315 -0.4620141 -0.0989163 -0.0038183  1.0191197
```

Asymptotic Covariance Matrix of Thresholds

```
          cl_pe_2
cl_pe_2   1.7159723
```

Asymptotic Covariances Between Correlations and Thresholds

```
                  co_cl_1    co_cl_2    co_pe_1    co_pe_2
[co_cl,co_cl]   0.0000000  0.0000000  0.0000000  0.0000000
[co_pe,co_cl]  -0.0554799 -0.0659489 -0.0606661 -0.0948283
[co_pe,co_pe]   0.0000000  0.0000000  0.0000000  0.0000000
[cl_pe,co_cl]   0.0553091  0.0933981 -0.4906571 -0.1316125
[cl_pe,co_pe]  -0.3668860 -0.3321084 -0.0221411 -0.0409370
[cl_pe,cl_pe]   0.0000000  0.0000000  0.0000000  0.0000000
```

Asymptotic Covariances Between Correlations and Thresholds

```
                          cl_pe_1    cl_pe_2
[co_cl,co_cl]   0.0000000  0.0000000
[co_pe,co_cl]  -0.3670009 -0.2590001
[co_pe,co_pe]   0.0000000  0.0000000
[cl_pe,co_cl]   0.0292260  0.1860359
[cl_pe,co_pe]   0.1479196  0.1981278
[cl_pe,cl_pe]   0.0000000  0.0000000
```