CMAT Newsletter: July 2008, Part I

Wolfgang M. Hartmann

July 2008

Contents

1 General Remarks

Most of the time during this last half year I spent on the following problems:

1. Adding more complicate examples to the `tnlp` directory of the `nlp` function. This uncovered many bugs, mostly connected to some features of `nlp` but also some general language problems.

2. Extending the functionality of the MDS function.

3. Adding some basic statistics routines based on code by my former class mate Andreas Hahnewald-Busch from the TU Dresden.

During some work on my wifes Microchip data which contains some rather long string data including special symbols, tabs and spaces, I had to improve the printout for vectors and matrices. Now long strings can wrap around in columns of tables taking multiple lines.
One of class mates when I studied Math from 1966 until 1971 was Andreas Hahnewald-Busch. I met Andeas already before we studied at the TU Dresden in a French language class at the evening school *Volkshochschule* in Dresden. Even though Andreas choose *Probabilistic Statistics* as his special topic and I choose *Numerical Mathematics* we never lost touch until his tragic dead. When I was employed at the Academy of Sciences in Berlin I needed a computer in Dresden to develop my software and Andreas gave me permission to use his computer account on the BESM-6 computer of the TU Dresden. Later when I applied for emigration I made him to my successor in a well paid side job teaching Pharmacists in Math and Computer Science at a College in Dresden. When I had to emigrate, I ask'd him to keep a few valuable things for me which I was not able to move to the West. Already working at SAS in the US, I called him rather often and mailed him at his wish a selection of SAS Software Manuals. The first mailing was obviously stolen by the East German authorities, however, the second mailing arrived. From talking to him on the phone, he was planning to write some new statistics software which was not available in SAS at that time.
From Hungary Andreas mailed me some listings of my older computer programs to the US, which was rather dangerous. I would never have ask'd him for this since I felt not even comfortable to pack such things into my moving boxes.
In memory of Andreas I implemented all his functions from the V. Nollau and A. Hahnewald-Busch book *"Statistische Analysen"* (1975) with the set of `hb....` functions. Note, the book and the original Algol code are from Pre-SAS time, and not much statistics software was available at that time. Even though much is covered by the `reg`, `discrim`, `glmod`, and `anova` functions already and my idea of CMAT is not to replicate what other software is already doing better I think of enhancing this software if I'll find some time in the near future.

1.1 New Functions

The following new functions are implemented:

codapp implements an important applications of the complete orthogonal decomposition (COD) for the solution of rank-deficient linear least squares problems (similar to the `APPCORT` function which I put into SAS/IML many years ago; as the `CMAT` function `cod` is similar to the `COMPORT` function which I wrote for SAS/IML).

hbanova implements nine different versions of analysis of variance for models I and II (chapters 4 and 7, A4.3, A4.4)

hbaddtst test for additivity by Tukey (chapter 4.5, A4.2)

hbbartlett Bartlett test (chapter 4.4, A4.1)

hbscheffe Scheffè (1959) test (chapter 4.3, A4.5)

hbcovar one way classification with one covariable and unrepeated two way classification with one covariable (chapters 5.1 and 5.2, A5) (AHB did not implement the (2,2) model of chapter 5.3)

hbdiscrim two class discriminance analysis (chapter 6.1, A3) (AHB did not implement the N class analysis which is described in chapter 6.2)

hblreg linear regression (simple version of the `reg` function) (chapters 3.1 to 3.3, A2.2)

hbltst linearity test by R. A. Fisher (chapter 2.9, A2.1)

hbqlreg simple (one predictor) quasi linear polynomial regression (chapter 3.5, A2.3)

hbrcmp compare the coefficients of two simple linear regressions (chapter 3.3, A2.4)

mds computes a variety of methods for *metric* and *nonmetric* multidimensional scaling (MDS) for 2-and 3-way data tables

selc sequential elimination of level combinations algorithm (JSS 317)

strquot reading data sets containing string data where strings are unquoted but separated by a specific separator string like a new line symbol

urd1out analysis of unreplicated 2^k and 2^{k-p} designs with a possible outlier

1.2 Fixed Bugs

By adding rather large examples for the `nlp` function, like those for neural nets, the efficient frontier of the Markowitz stock portfolio model, and some examples which I designed for an ASA meeting long time ago together with Ying So and Minbo Kim, the CMAT language was tested very extensively and still a large number of bugs were found. Some rather serious bugs were found not only in the `ode` and `nlp` functions, also in some more general language features.
The first time we dealt with long and non standard strings (containing special symbols and white space) in data describing genes. The `["` and `"]` operators work only for standard strings (not containing white space or special symbols). A new function `strquot` was written recognicing a specific separator string (can be a new line) indicating the end of a string. Bugs had to be fixed also in printing functions for objects with oversize columns, i.e. long strings not fitting the specified line size.

2 Modifications of Features

2.1 Extensions to Various Functions

2.1.1 Extension of `min` and `max` Functions

z = max(a,b)

z = max(a)

z = max(a,b,c,...)

Purpose: The arguments can be scalars or data objects like vectors, matrices, or tensors.

z=max(a,b) This is the common form:

- If one of the two arguments is scalar and the other argument is a matrix or vector, then the function is performed using the scalar and each entry of the other operand. The result is of the same size as the matrix or vector argument.
- If both arguments are vectors or matrices, then both must have the same dimension (same number of rows and columns). The function is performed pairwise with corresponding pairs of entries of the two operands.

For integer and real arguments the result is

$$z = \max(a, b) = \begin{cases} a & \text{if } a >= b \\ b & \text{if } a < b \end{cases}$$

For complex arguments a or b the decision is based on the modulos (absolute values) of the arguments.

z=max(a) If `a` is an object, the largest nonmissing numerical entry of a is returned.

z=max(a,b,c,...) The arguments can be scalars or data objects not necessarily of the same size. The result z is the largest scalar entry of all arguments.

If the arguments have no nonmissing values (strings), a missing value is returned.

Input: The arguments a and b must be numeric scalars, vectors, matrices, or tensors.

Output: **z=max(a,b)** The result z is a scalar if both, a and b are scalars. Otherwise it is a data object of the same size as one or both of the arguments.

z=max(a) The result z is a scalar.

z=max(a,b,c,...) The result z is a scalar.

The data type of the result z is the same as the maximum data type of all arguments in the order integer, real, complex. For complex arguments a or b the returned value is complex, either a or b, however the pairwise comparison is based on their modulos (the absolute values).

Restrictions: 1. Missing value is returned if arguments contain string data or missing values.

2. For `z = max(a,b)` and if a or b is an object then the other argument must be either scalar or an object of the same size.

Relationships: min()

Examples: 1. Example for 2-argument form `max(a,b)`:

```
a = [  3., -3.,  2., -2. ];
b = [ -1.,  1., -3.,  3. ];
d = min(a,b);
print "Min=", d;
e = max(a,b);
print "Max=", e;
```

```
Min=
   |         1
---------------
 1 |  -1.00000
 2 |  -3.00000
 3 |  -3.00000
 4 |  -2.00000
```

```
Max=
   |         1
---------------
 1 |   3.00000
 2 |   1.00000
 3 |   2.00000
 4 |   3.00000
```

2. Example for 1-argument form `max(a)`:

```
q = [ 0. 1. 1. 3. 5. -3. ];
print "Min=", qmin = min(q);
print "Max=", qmax = max(q);
```

```
Min=-3.0000
Max= 5.0000
```

3. Example for more than two argument form `max(a,b,c,...)`:

```
a = [ 1 2 3 ]; b = [ 3 4 , 5 6 ]; c = 7.;
print "Min=", d = min(a,b,c);
print "Max=", e = max(a,b,c);
```

```
Min= 1
Max= 7
```

z = min(a,b)

z = min(a)

z = min(a,b,c,...)

Purpose: The arguments can be scalars or data objects like vectors, matrices, or tensors.

z=min(a,b) This is the common form:

- If one of the two arguments is scalar and the other argument is a matrix or vector, then the function is performed using the scalar and each entry of the other operand. The result is of the same size as the matrix or vector argument.
- If both arguments are vectors or matrices, then both must have the same dimension (same number of rows and columns). The function is performed pairwise with corresponding pairs of entries of the two operands.

For integer and real argument the result is

$$z = \min(a, b) = \begin{cases} a & \text{if } a <= b \\ b & \text{if } a > b \end{cases}$$

For complex arguments a or b the decision is based on the modulos (absolute values) of the arguments.

z=min(a) If `a` is an object, the smallest nonmissing numerical entry of a is returned.

z=min(a,b,c,...) The arguments can be scalars or data objects not necessarily of the same size. The result z is the smallest scalar entry of all arguments.

If the arguments have no nonmissing values (strings), a missing value is returned.

Input: The arguments a and b must be numeric scalars, vectors, matrices, or tensors.

Output: **z=min(a,b)** The result z is a scalar if both, a and b are scalars. Otherwise it is a data object of the same size as one or both of the arguments.

z=min(a) The result z is a scalar.

z=min(a,b,c,...) The result z is a scalar.

The data type of the result z is the same as the maximum data type of all arguments in the order integer, real, complex. For complex arguments a or b the returned value is complex, either a or b, however the pairwise comparison is based on their modulos (the absolute values).

Restrictions: 1. Missing value is returned if arguments contain string data or missing values.

2. For `z = min(a,b)` and if a or b is an object then the other argument must be either scalar or an object of the same size.

Relationships: max()

Examples: 1. Example for 2-argument form `max(a,b)`:

```
a = [  3., -3.,  2., -2. ];
b = [ -1.,  1., -3.,  3. ];
d = min(a,b);
print "Min=", d;
e = max(a,b);
print "Max=", e;
```

```
Min=
   |         1
---------------
 1 |  -1.00000
 2 |  -3.00000
 3 |  -3.00000
 4 |  -2.00000
```

```
Max=
   |         1
---------------
 1 |   3.00000
 2 |   1.00000
 3 |   2.00000
 4 |   3.00000
```

2. Example for 1-argument form `max(a)`:

```
q = [ 0. 1. 1. 3. 5. -3. ];
print "Min=", qmin = min(q);
print "Max=", qmax = max(q);
```

```
Min=-3.0000
Max= 5.0000
```

3. Example for more than two argument form `max(a,b,c,...)`:

```
a = [ 1 2 3 ]; b = [ 3 4 , 5 6 ]; c = 7.;
print "Min=", d = min(a,b,c);
print "Max=", e = max(a,b,c);
```

```
Min= 1
Max= 7
```

2.1.2 Extension of `irtml` Function

1. When the new `"size"` option is specified the data matrix can have an additional column specifying pattern probability values (normally $0 \leq p_i \leq 1$ and $\sum_i p_i = 1$). The sample size is then specified by `"size"` with an int or real scalar. The pattern of the input matrix must be distinct.

2. The $R1$ goodness of fit measure is computed and returned when all items are dichotomous and the `"nor1"` option is not specified.

2.1.3 Extension of `cod` Function

The function for the complete orthogonal decomposition
`r = cod(a)` or `<q,r,v,pi> = cod(a)`
was extended for a second and third input argument to `r = cod(a<,b<sopt>>` or `<q,r,v,pi,lind> = cod(a<,b<sopt>>)`.
If the second input argument `b` is specified other than missing, it must be a vector or matrix $\mathbf{B}$ having the same number of rows as the first argument `a`. If `b` is specified nonmissing, the result `q` returns $\mathbf{Q}^T\mathbf{B}$ rather than $\mathbf{Q}$. This extension is useful for the major aplication of this function for solving rank deficient least sqares problems, see also the new function `codapp()`.
The third input argument `sopt` is a string option specifying one of two algorithms.

"old" an older algorithm which was already available and should be used together with the LAPACK option.

"new" a newly developed algorithm.

At this time no sufficient test computations were made to determine which of the algorithms is the preferred one.

2.1.4 Extension of `mds` Function

The `mds` function was extended by features of the MULTISCALE algorithm by Jim Ramsay (1969, 1977). Those include:

- The spline transform among dissimilarity data and model distances.
- Three different model types for subjective weights:
  1. the identity model,
  2. the d diagonal weighting model, which includes the estimation of $nsub \times d$ weights.
  3. and the full $d \times d$ symmetric weighting matrix, which includes the estimation of $nsub \times d * (d + 1)/2$ weights.
- The computation of asymptotic standard errors for the point coordinates. (Not yet for the weight estimates.) This is implemented for four different data distribution assumptions:

  1.

  Asymptotic standard errors and confidence ellipsoids are at this time only for the unrotated point coordinates as with the MULTISCALE program.

Instead of listing here the additions to the function we prefer to give a complete new documentation of the function in the *New Developments* section below.

2.1.5 Extensions to `ode` Function

The old form of the `ode` call utilized a numeric vector for specifying a number of options. Now, the `optn` 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.

y = ode(func,tab,y0<,optn<,root<,afun<,jacm>>>)

<y,t> = ode(func,tab,y0<,optn<,root<,afun<,jacm>>>)

All other features of the `ode` function are unchanged.

Option	Second Column	Meaning
"maxord"	int	maximum order of interpolation; by default: "maxord"=5 for Gear's method (i.e. "gear" or "dyna", and "maxord"=12 for Adams method (i.e. "adam") this option does not apply to methods "rk45" and "rk56"
"maxst"	int	maximum number of internal steps performed within any of the integration intervals; the default is "maxst"=1000
"met"	string	integration method: stiff-nonstiff
	"adam"	Adam's method: only for nonstiff ODE's
	"gear"	Gear's method: also for stiff ODE's (default)
	"dyna"	method choosing dynamically between "adam" and "gear", (note, that "dyna" is not possible for "nroot" > 0)
	"rk45"	Fehlberg's (4,5) order Runge-Kutta (similar to Watts and Shampine)
	"rk56"	Verner's (5,6) order Runge-Kutta method (similar to an implementation by Jackson)
"minval"	real	specifies the smallest nonzero value for solution y_i; the ODE function will not approximate exactly values for y_i smaller than "maxst"; default is 0. this option does not apply to methods "rk45" and "rk56"
"nequ"	int	only valid for "type"=2: specifies the number m of ODE's of the hybrid problem; default is "nequ"=n this option does not apply to methods "rk45" and "rk56"
"nolog"		supresses output of warnings into the log
"nroot"	int	number of roots t_r returned by user function x=ROOT(i,t,y); the iteration is stopped when the first specified root $t_0 \le t_r \le t_N$ is reached; default is 0 this option does not apply to methods "rk45" and "rk56"
"print"	int	controls printed output
	0	no printed output
	1	print information at begin and end of iteration
	2	print iteration history additionally

Option	Second Column	Meaning
`"reltol"`	real	relative precision of integration (def=1.e-4)
`"task"`	string	compute y values at $t_1, \ldots t_N$ exactly or permit for interpolation; default is `"task" = "int"`:
	"int"	for `t[1]` until `t[N-1]` interpolated (like `"task" = "eff"`), for `t[N]` precisely (like `"task" = "exa"`)
	"eff"	interpolation is permitted for the y values at $t_1, \ldots, t_N$
	"exa"	all $t_1, \ldots, t_N$ are matched precisely; this is computationally very expensive
`"type"`	int	which kind of problem is to be solved
	0	explicit initial value problem: $$\frac{dy_i}{dt} = f_i(y,t), \quad i = 1, \ldots, n$$ this is default
	1	implicit initial value problem: $$\mathbf{A}\frac{dy}{dt} = f(y,t)$$ where function `a=AFUN(t,y)` must return return $n \times n$ matrix $\mathbf{A}$ for given t and n vector y
	2	hybrid system of m ODE's and $n-m$ algebraic equations $$\begin{aligned} \mathbf{a}_i\frac{dy_i}{dt} &= f_i(y,t), \quad i = 1, \ldots, m \\ 0 &= f_i(y,t), \quad i = m+1, \ldots, n \end{aligned}$$ where the user function `a=AFUN(t,y)` must return n vector a for given t and n vector y

2.1.6 Extensions to `nlp` and `nle`: New Returns

The following was the old syntax for calling `nle` and `nlp`:

< xr,rp > = nle(func,x0<,sopt<,par,<jac>>>)

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

We added some more return arguments to the two functions:

< xr,rp,der1 > = nle(func,x0<,sopt<,par,<jac>>>)

< xr,rp,der1,der2,acon,dpro,jhnlc > = nlp(func,x0<,keyopt<,lau<,. . . ,<jcon>>>>>>>)

The following results are returned only if available for the specific optimization technique:

der1 except for derivative free methods, `der1` contains in its first row the gra-

dient at the solution `xr`. If an $m \times n$ Jacobian was used it is stored in rows $2, \ldots, m+1$.

der2 if the Hessian or cross-product Jacobian was used by the optimization technique, `der2` contains this $n \times n$ matrix evaluated at the solution `xr`.

acon returns a column vector or a 3 or 5 column matrix indicating

1. in column 1: active boundary constraints
2. in column 2 and 3: values of linear constraints and which are active at the solution
3. in column 4 and 5: values of nonlinear constraints which are active at the solution

dpro contains only for constrained optimization: the vector of projected gradient in its first row and if available the projected Hessian matrix in additional rows.

jhnlc contains only for nonlinearly constrained optimization: either the Jacobian or the Hessian of the nonlinear constraints.

2.1.7 Extensions to `nlp` Function

Modifying the `nlp` Function: Some Options The default value for the `nlcrmit` option specifying the maximum number of preprocessor iterations for reducing the violation of nonlinear constraints at the starting point was set to zero (from former 10 iterations). Also constraints are no longer scaled by default, it must now be invoked by specifying the `cscale` option.
In a postprocessing step, the optimality of a solution may be tested by evaluating the function on points of a circle around the parameter estimate. The `rtop` option specifies a radius of the circle. Infeasible circle points are not evaluated.

Extensions to `nlp` Function: Grid Search Until now, the input argument `x0` for the starting point had to be a n vector. We now allow also the input of a $n \times k$ matrix where each row in its first $k_i, i = 1, \ldots, n$ columns specifies grid points for an initial estimate. Usually, $k = max_{i=1}^{n} k_i$. The remaining columns $k_i, \ldots, k$ of row i must be set to missing values. During a preprocessor step, all $\prod_{i=1}^{n} k_i$ starting points are evaluated and the best is selected for the starting point. Note, missing values and multiple grid values are removed from the grid. The new `"best"` option specifies an integer number of best points to be printed with their corresponding function values.
If there are constraints specified, the `nofea` option drops infeasible points from the grid evaluation.

```
function frosbr2(x) {
  f = cons(2);
  f[1] = 10. * (x[2] - x[1] * x[1]);
```

```
  f[2] = 1. - x[1];
  return(f);
}


x1 = [ -1.2  -5.:1.:9.  -2. ];
x2 = [  1.:2.:5. 14#. ];
x0 = x1 |> x2; print "Grid=",x0;


/* Levenberg-Marquardt */
mopt = [ "tech"   "levmar" ,
         "pbest"        10 ,
         "print"         3 ];
< xr,rp,der1 > = nlp(frosbr2,x0,mopt);
print "LM LSQ: DER1=",der1;
```

Extensions to `nlp` Function: New Techniques Three additional conjugate gradient methods were added

"bm" spectral conjugate gradient method by Birgin & Martinez (2001) with restart as proposed by Andrei (2007)

"spr" scaled Polak-Ribière method (Andrei, 2007)

"sfr" scaled Fletcher-Reeves method (Andrei, 2007)

The three methods are implemented with two different restart methods which can be specified by the `"vers"` option:

0 performs angle restart testing the size of descent:

$$g_r^T d_r \geq -0.001 \parallel g_r \parallel\parallel d_r \parallel$$

1 performs Powell-Beale restart testing the orthonality of successive search directions:

$$\|g_r^T g_{r-1}\| \geq 0.2 \parallel g_r \parallel$$

2 performs restart as specified by the `"rest"` option specifying an integer number of iterations after which a restart is done.

The new methods are applied to the $n = 2$ Rosenbrock function:

```
real function frosbr1(x)
{
  /* crit = .5 * f' * f */
  s = 0.;
```

```
  r1 = 10. * (x[2] - x[1] * x[1]);
  r2 = 1. - x[1];
  crit = r1 * r1 + r2 * r2;
  return(crit);
}


real function grosbr(x)
{
  /* gradient= J' * f */
  f = cons(2);
  f[1] = 10. * (x[2] - x[1] * x[1]);
  f[2] = 1. - x[1];
  jacm = cons(2,2,1.);
  jacm[1,1] = -20. * x[1]; jacm[1,2] = 10.;
  jacm[2,1] = -1.;         jacm[2,2] = 0.;
  g = 2. * jacm' * f;
  return(g);
}


x0 = [ -1.2 1. ];
```

1. Compare the Birgin-Martinez method with our old Powell-Beale restart method:

```
print "Scaled Birgin-Martinez Procedure";
mopt = [ "tech" "congra" ,
         "upd"      "bm" ];
< xr, rp> = nlp(frosbr1,x0,mopt,.,.,.,grosbr);
```

```
                Conjugate-Gradient Optimization
          Scaled Perry Update (Birgin & Martinez, 2001)
                          Angle Restart
                     User Specified Gradient

   Iteration Start:
   N. Variables              2
   Criterion       24.20000000        Max Grad Entry  215.6000000


  Iter rest nfun act   optcrit  difcrit maxgrad   alpha    slope
     1    0    3   0  4.225209 19.97479 12.0012  8e-004 -54227.4
     2    0    5   0  4.123315 0.101894 1.47559 1.23881 -0.16427
```

```
   3    0   11    0  3.153859 0.969456 15.3256 366.441  -3e-003
   4    0   13    0  2.935328 0.218531 18.1200  5e-003 -95.0228
   5    0   14    0  1.939282 0.996047 1.97128 0.03898 -53.7121
   6    0   16    0  1.651216 0.288065 5.72925 0.92313 -0.45306
   7    0   18    0  1.504818 0.146398 8.83740 0.04532 -5.50434
   8    0   19    0  1.047402 0.457416 2.06775 0.12171 -6.38630
   9    0   23    0  0.533302 0.514099 5.50337 6.22620 -0.12101
  10    0   24    0  0.242992 0.290310 5.45446 0.25691 -2.87669
  11    0   26    0  0.172798 0.070195 0.76193 0.01766 -7.95091
  12    0   29    0  0.107325 0.065473 3.19427 39.5250  -2e-003
  13    0   30    0  0.059212 0.048114 4.13722 0.01550 -5.51292
  14    0   32    0  0.038274 0.020937 0.49901  4e-003 -11.0304
  15    0   35    0  0.017715 0.020559 2.07891 14.8899  -2e-003
  16    0   36    0  3.6e-003 0.014159 1.39582  6e-003 -4.76617
  17    0   38    0  2.6e-003 9.7e-004 0.19049  8e-004 -2.35155
  18    0   41    0  2.7e-004 2.3e-003 0.53477 1.93390  -2e-003
  19    0   43    0  4.9e-005 2.2e-004 0.05115  6e-004 -0.69930
  20    0   45    0  4.8e-007 4.9e-005 0.02647 0.45229  -2e-004
  21    0   47    0  1.2e-010 4.8e-007  1e-004  5e-004  -2e-003

   Successful Termination After    21 Iterations
   ABSGCONV convergence criterion satisfied.
   Criterion       1.2240e-010        Max Grad Entry  0.000130196
   Slope SDirect. -0.002012364
   N. Function Calls        48        N. Gradient Calls        48
   Preproces. Time           0        Time for Method           1
   Effective Time            1


print "Powell-Beale Restart Procedure";
mopt = [ "tech" "congra" ,
         "upd"      "pb" ];
< xr, rp> = nlp(frosbr1,x0,mopt,.,.,.,grosbr);


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

   Iteration Start:
   N. Variables              2
   Criterion       24.20000000        Max Grad Entry  215.6000000


  Iter rest nfun act   optcrit  difcrit maxgrad   alpha    slope
     1    0    5   0  12.21263 11.98737 102.605  1e-003 -54227.4
```

```
 2    1    9   0  4.465179 7.747455 31.8199 0.08022 -158.404
 3    2   12   0  1.721788 2.743390 7.51549  4e-003 -1652.33
 4    3   14   0  1.261716 0.460073 5.14185 0.04001 -17.1437
 5    4   18   0  1.225464 0.036251 8.25795  3e-003 -25.8111
 6    5   20   0  0.811041 0.414424 5.04328 0.01000 -68.4931
 7    5   22   0  0.750370 0.060671 7.06388 0.02346 -4.41542
 8    6   24   0  0.460088 0.290282 1.56525 0.01000 -52.9473
 9    6   29   0  0.205633 0.254455 5.67494 0.28754 -1.36086
10    7   32   0  0.156114 0.049519 6.44849  3e-003 -24.1788
11    7   35   0  0.153919 2.2e-003 7.55008  1e-003 -7.63691
12    8   38   0  0.067512 0.086407 5.88641  4e-003 -84.5981
13    9   43   0  0.015795 0.051717 3.39118  4e-003 -20.6174
14   10   46   0  0.013784 2.0e-003 3.66717  1e-003 -3.22118
15   11   49   0  2.9e-003 0.010924 0.53580  1e-003 -17.8777
16   12   53   0  3.4e-004 2.5e-003 0.35287 0.02593 -0.15384
17   12   56   0  3.3e-004 1.3e-005 0.55227  2e-003 -0.04146
18   13   59   0  4.7e-005 2.8e-004 0.11433  3e-003 -0.33638
19   14   62   0  3.8e-005 8.4e-006  4e-003  1e-003 -0.01702
20   14   64   0  1.3e-005 2.5e-005 0.12006 0.25028  -2e-004
21   14   67   0  5.9e-006 7.2e-006 0.09242  1e-003 -0.01590
22   15   70   0  3.9e-008 5.8e-006  3e-003  1e-003 -0.01098
23   15   72   0  3.8e-008 8.7e-010  2e-003 0.11041  -2e-008
24   16   74   0  3.0e-013 3.8e-008  8e-006 0.02064  -4e-006

Successful Termination After    24 Iterations
ABSGCONV convergence criterion satisfied.
Criterion       3.0413e-013        Max Grad Entry  8.0078e-006
Slope SDirect. -3.6728e-006
N. Function Calls        75        N. Gradient Calls        36
N. Line Searches         24        Preproces. Time           0
Time for Method           0        Effective Time            0
```

2. Compare the scaled Polak-Ribiere method with our old method:

```
 print "Scaled Polak-Ribiere Procedure";
 mopt = [ "tech" "congra" ,
          "upd"     "spr" ];
/* options debug="nlpcog*=4"; */
 < xr, rp > = nlp(frosbr1,x0,mopt,.,.,.,grosbr);


                  Conjugate-Gradient Optimization
```

```
   Scaled Polak & Ribiere Update (Birgin & Martinez, 2001)
                        Angle Restart
                    User Specified Gradient


 Iteration Start:
 N. Variables              2
 Criterion       24.20000000        Max Grad Entry  215.6000000


Iter rest nfun act   optcrit  difcrit maxgrad   alpha    slope
   1    0    3   0  4.225209 19.97479 12.0012  8e-004 -54227.4
   2    1    5   0  4.123326 0.101883 1.47590 1.23607 -0.16461
   3    1   11   0  3.398703 0.724623 16.6396 295.210  -3e-003
   4    1   13   0  3.148488 0.250214 19.5048  4e-003 -101.613
   5    1   14   0  2.194519 0.953969 3.67853 0.02960 -70.6406
   6    1   16   0  2.000833 0.193686 6.31449 0.33425 -1.02907
   7    1   18   0  1.968548 0.032285 7.82874 0.02153 -2.79197
   8    1   21   0  1.530584 0.437964 9.44477 0.12103 -6.78695
   9    1   23   0  1.229539 0.301046 1.87472 0.08012 -7.74129
  10    1   25   0  0.995246 0.234292 7.40442 12.3664 -0.03376
  11    1   27   0  0.987355 7.9e-003 8.53973 0.01906 -0.95770
  12    1   29   0  0.748566 0.238789 2.06697 0.17265 -2.83409
  13    1   31   0  0.581345 0.167221 5.46329 5.19952 -0.05272
  14    1   33   0  0.562095 0.019250 6.86845 0.03178 -1.18031
  15    1   34   0  0.418152 0.143943 1.62207 0.05961 -3.65117
  16    2   36   0  0.410735 7.4e-003 1.35378 0.32197 -0.04607
  17    2   39   0  0.289956 0.120779 4.32608 18.3293 -0.01033
  18    2   41   0  0.256048 0.033907 5.49160 0.01997 -3.10424
  19    2   42   0  0.169400 0.086648 4.26835 0.02998 -9.01681
  20    2   46   0  0.033225 0.136175 1.58459 0.59975 -0.15382
  21    2   47   0  0.015496 0.017729 4.94562 0.02157 -5.09254
  22    2   49   0  1.4e-003 0.014068 0.26340  5e-004 -59.8030
  23    3   51   0  1.4e-003 3.7e-005 0.03097  4e-003 -0.02008
  24    3   53   0  1.8e-004 1.2e-003 0.53704 2713.69  -1e-006
  25    3   55   0  4.1e-005 1.4e-004 0.10794  5e-004 -0.54580
  26    3   57   0  1.0e-005 3.0e-005 0.07347  7e-003  -8e-003
  27    3   58   0  1.9e-007 1.0e-005  2e-003  4e-003  -5e-003
  28    4   60   0  1.9e-007 2.2e-009  4e-004  6e-004  -8e-006

 Successful Termination After    28 Iterations
 ABSGCONV convergence criterion satisfied.
 Criterion       1.8798e-007        Max Grad Entry  0.000379533
 Slope SDirect. -7.9765e-006
 N. Function Calls        61        N. Gradient Calls        61
 Preproces. Time           0        Time for Method           0
```

```
   Effective Time            0


print "UNscaled Polak-Ribiere Procedure";
mopt = [ "tech" "congra" ,
         "upd"      "pr" ];
< xr, rp > = nlp(frosbr1,x0,mopt,.,.,.,grosbr);


                Conjugate-Gradient Optimization
          Polak & Ribiere Update (Fletcher, 1980, 4.1.12)
                     User Specified Gradient


   Iteration Start:
   N. Variables              2
   Criterion       24.20000000        Max Grad Entry  215.6000000


  Iter rest nfun act   optcrit  difcrit maxgrad   alpha    slope
     1    0    4   0  16.80653 7.393470 123.995  2e-003 -54227.4
     2    1    9   0  6.986146 9.820384 74.6509  2e-003 -20689.4
     3    2   13   0  4.230184 2.755962 15.2832  1e-003 -6631.44
     4    3   17   0  4.065223 0.164961 1.83761  1e-003 -326.639
     5    3   21   0  3.430994 0.634229 17.3148 0.28126 -3.07935
     6    3   25   0  3.181065 0.249929 20.3821  1e-003 -347.942
     7    3   30   0  1.399645 1.781420 4.06284  1e-002 -566.262
     8    3   33   0  1.381114 0.018531 2.77742  2e-003 -24.2291
     9    3   37   0  0.989928 0.391186 6.02301 0.05952 -9.44289
    10    3   40   0  0.825493 0.164434 12.6224 0.02400 -36.8003
    11    3   45   0  0.815083 0.010410 12.9909  7e-004 -31.0781
    12    3   47   0  0.517657 0.297426 5.87687  5e-003 -239.950
    13    4   50   0  0.433281 0.084376 3.81449  5e-003 -63.1391
    14    5   53   0  0.410084 0.023197 2.10443  5e-003 -16.9923
    15    6   56   0  0.399479 0.010605 1.54651  5e-003 -5.73771
    16    6   59   0  0.386033 0.013446 4.51900 0.03124 -1.32944
    17    6   62   0  0.272963 0.113070 10.5626 0.02400 -21.4661
    18    6   64   0  0.121671 0.151292 10.4183  5e-003 -86.3239
    19    6   67   0  0.029634 0.092037 1.49443  1e-003 -149.357
    20    7   71   0  0.027814 1.8e-003 0.15268  1e-003 -2.71767
    21    7   75   0  0.012246 0.015568 2.01672 0.83495 -0.02958
    22    7   77   0  4.8e-004 0.011766 0.40836  5e-003 -5.67809
    23    7   80   0  2.2e-004 2.6e-004 0.14575  1e-003 -0.31798
    24    7   82   0  1.5e-004 7.0e-005 0.43475  5e-003 -0.04037
    25    7   85   0  9.4e-005 6.0e-005 0.38651  8e-004 -0.16030
    26    7   88   0  1.0e-007 9.4e-005  6e-003  1e-003 -0.18617
```

```
 27    7   91    0  3.7e-008 6.6e-008  6e-003 0.02500  -9e-006
 28    7   94    0  1.2e-008 2.4e-008  9e-005  1e-003  -5e-005

Successful Termination After    28 Iterations
ABSGCONV convergence criterion satisfied.
Criterion       1.2493e-008        Max Grad Entry  8.6491e-005
Slope SDirect. -4.9026e-005
N. Function Calls        95        N. Gradient Calls        51
N. Line Searches         28        Preproces. Time           0
Time for Method           0        Effective Time            0
```

3. Compare the scaled Fletcher-Reeves method with our old method:

```
print "Scaled Fletcher-Reeves Procedure";
mopt = [ "tech" "congra" ,
         "upd"     "sfr" ];
< xr, rp> = nlp(frosbr1,x0,mopt,.,.,.,grosbr);
```

```
                Conjugate-Gradient Optimization
    Scaled Fletcher & Reeves Update (Birgin & Martinez, 2001)
                          Angle Restart
                    User Specified Gradient


Iteration Start:
N. Variables              2
Criterion       24.20000000        Max Grad Entry  215.6000000


 Iter rest nfun act   optcrit  difcrit maxgrad   alpha    slope
    1    0    3   0  4.225209 19.97479 12.0012  8e-004 -54227.4
    2    0    5   0  4.123006 0.102203 1.46651 1.31975 -0.15465
    3    0    9   0  2.952149 1.170858 13.0280 407.726  -3e-003
    4    0   11   0  2.748430 0.203719 16.4498  5e-003 -90.5729
    5    0   13   0  2.698100 0.050330 16.6332  3e-003 -29.1203
    6    0   14   0  2.609541 0.088559 15.7797  4e-003 -33.0735
    7    0   15   0  2.504919 0.104622 16.3277  5e-003 -33.9579
    8    0   16   0  2.411523 0.093397 17.3825  6e-003 -28.0149
    9    0   17   0  2.362683 0.048840 18.8530  8e-003 -19.3104
   10    0   19   0  2.347059 0.015624 19.3736  3e-003 -9.20253
   11    0   20   0  2.310751 0.036308 19.6798  4e-003 -13.8259
   12    0   22   0  2.237142 0.073608 20.2813  8e-003 -17.5790
   13    0   24   0  2.202357 0.034785 20.7744  6e-003 -11.4053
   14    0   25   0  2.168168 0.034190 21.2433  6e-003 -10.8573
```

```
    15    0   26   0  2.137800 0.030368 21.7134  7e-003 -10.1502
    16    0   27   0  2.113663 0.024137 22.2011  7e-003 -9.29385
    17    0   28   0  2.097332 0.016331 22.7153  7e-003 -8.37092
    18    0   30   0  2.083319 0.014013 22.9657  4e-003 -7.41977
    19    0   31   0  2.055238 0.028082 23.0764  4e-003 -11.6201
    20    0   33   0  1.995047 0.060190 23.2723  7e-003 -16.6854
    ..........................................................
    51    0   82   0  1.2e-005 5.8e-005 0.10173  2e-003 -0.05639
    52    0   84   0  3.0e-006 9.0e-006 0.01539  6e-004 -0.02994
    53    0   86   0  2.8e-006 2.0e-007  3e-003 0.01278  -3e-005
    54    0   88   0  2.8e-006 1.3e-008  2e-003 0.30383  -9e-008
    55    0   90   0  2.8e-006 2.7e-008  4e-003 4.54979  -1e-008
    56    0   92   0  2.3e-006 4.7e-007 0.02129 1.85485  -5e-007
    57    0   94   0  5.0e-007 1.8e-006 0.01871 0.02257  -2e-004
    58    0   96   0  1.3e-007 3.8e-007  5e-003  1e-003  -5e-004
    59    0   98   0  1.1e-007 1.8e-008  8e-004  6e-003  -6e-006

   Successful Termination After    59 Iterations
   ABSGCONV convergence criterion satisfied.
   Criterion       1.1014e-007        Max Grad Entry  0.000777436
   Slope SDirect. -5.8224e-006
   N. Function Calls        99        N. Gradient Calls        99
   Preproces. Time           0        Time for Method           0
   Effective Time            0
```

```
print "UNScaled Fletcher-Reeves Procedure";
mopt = [ "tech" "congra" ,
         "upd"      "fr" ];
< xr, rp> = nlp(frosbr1,x0,mopt,.,.,.,grosbr);
```

```
                Conjugate-Gradient Optimization
        Fletcher & Reeves Update (Fletcher, 1980, 4.1.4)
                    User Specified Gradient


   Iteration Start:
   N. Variables              2
   Criterion       24.20000000        Max Grad Entry  215.6000000


  Iter rest nfun act   optcrit  difcrit maxgrad   alpha    slope
     1    0    5   0  4.128117 20.07188 1.55692  8e-004 -54227.4
     2    0   11   0  2.848166 1.279950 12.1975 0.43624 -3.15335
```

```
 3    0   14   0  2.648745 0.199422 14.5428  2e-003 -194.176
 4    0   19   0  2.515573 0.133171 14.5202  7e-004 -332.107
 5    0   24   0  2.422893 0.092681 14.7083  5e-004 -381.612
 6    0   29   0  2.350554 0.072339 15.8808  3e-004 -403.521
 7    0   36   0  2.283377 0.067177 16.9342  3e-004 -431.541
 8    0   43   0  2.232105 0.051272 17.7922  2e-004 -408.278
 9    0   50   0  2.187037 0.045067 18.5227  2e-004 -407.317
10    0   57   0  2.146450 0.040587 19.1559  2e-004 -406.614
11    0   64   0  2.109106 0.037344 19.7124  2e-004 -407.605
12    0   71   0  2.074072 0.035033 20.2064  2e-004 -411.182
13    0   78   0  2.040616 0.033457 20.6483  2e-004 -417.956
14    0   83   0  2.008235 0.032381 21.0080  1e-004 -428.395
15    0   88   0  1.972465 0.035770 21.3331  1e-004 -466.679
16    0   93   0  1.936408 0.036057 21.6244  1e-004 -486.531
17    0   98   0  1.899928 0.036480 21.8835  1e-004 -509.252
18    0  103   0  1.862564 0.037364 22.1107  1e-004 -537.438
19    0  108   0  1.823815 0.038749 22.3052  1e-004 -571.786
20    0  113   0  1.783149 0.040666 22.4651  1e-004 -612.851
.....................................................................
70    0  304   0  3.5e-007 1.5e-006 0.01295  3e-003  -1e-003
71    0  307   0  2.0e-007 1.6e-007  3e-003  2e-003  -2e-004
72    0  310   0  1.9e-007 8.4e-009  9e-004  2e-003  -1e-005

Successful Termination After    72 Iterations
ABSGCONV convergence criterion satisfied.
Criterion       1.8770e-007        Max Grad Entry  0.000885977
Slope SDirect. -1.1068e-005
N. Function Calls       311        N. Gradient Calls       166
N. Line Searches         72        Preproces. Time           0
Time for Method           0        Effective Time            0
```

3 New Developments

3.1 Function `codapp`

< x, lind> = codapp(a,b<,sopt>)

Purpose: The complete orthogonal decompositon of $m \times n$ matrix $\mathbf{A}$ is:

$$\mathbf{A} = \mathbf{Q} \begin{bmatrix} \mathbf{R}^{-1} \\ \mathbf{0} \end{bmatrix} \mathbf{\Pi}^T \mathbf{V}^T$$

This function computes the $n \times p$ matrix $\mathbf{X}$:

$$\mathbf{X} = \mathbf{V\Pi} \begin{bmatrix} \mathbf{R}^{-1T} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \mathbf{Q}^T \mathbf{B}$$

which is the minimum length solution of a rank-deficient linear least-sqares problem:

$$\min_{\mathbf{X}} \|\mathbf{A} - \mathbf{B}\mathbf{X}^T\|_2$$

For full rank $\mathbf{A}$, the $n \times n$ matrix $\mathbf{V}$ is the identity matrix and the solution $\mathbf{X}$ is unique. The matrix $\mathbf{\Pi}$ is a permutation matrix. For rank deficient matrix $\mathbf{A}$, $m \geq n$, a solution $\mathbf{X}$ is not unique without additional constraint(s), like that of mininum length.

Input: **a** Numeric (real or complex) $m \times n$ matrix $\mathbf{A}$.

b Numeric (real or complex) $m \times p$ matrix $\mathbf{B}$.

sopt is a string option specifying one of two algorithms:

"old" an older algorithm which was already available and should be used together with the LAPACK option.

"new" a newly developed algorithm.

At this time no sufficient test computations were made to determine which of the algorithms is the preferred one.

Output: The following matrices are returned:

prqb Returns the $n \times p$ matrix $\mathbf{X}$.

lind Gives the integer number of rank deficiencies. For full rank of $\mathbf{A}$, $rank(\mathbf{A}) = \min(m, n)$, `lindep` is zero. Therefore we obtain $rank(\mathbf{A}) = \min(m, n) - lindep$.

Restrictions: 1. No missing values or string data are permitted for arguments `a` and `b`.

2. Arguments `a` and `b` must have the same number of rows.

Relationships:

Examples: 1. Simple Example from function `cod`:

The following 4×4 matrix $\mathbf{A}$ has $rank(\mathbf{A}) = 2$

```
options nolapack;
b = [ 1 2,
      3 4,
      5 6,
      7 8];
a = b * b';
nr = nrow(a); nc = ncol(a);

< x, lind > = codapp(a,b,"new");
print "X=", x; print "Lind=", lind;
```

```
X=
   |         1         2
-----------------------
 1 |   -1.0000   0.85000
 2 |  -0.50000   0.45000
 3 |   -6e-015   0.05000
 4 |   0.50000  -0.35000


Lind= 2
```

Check this result with the `cod` function:

```
< qtb, r, v, p, lind > = cod(a,b,"new");
rank = nc - lind;
print "Qtb=", qtb; print "R=", r; print "V=", v;
print "Piv=", p; print "Lind=", lind;
```

```
Qtb=
   |         1         2
-----------------------
 1 |   -9.1530   -10.947
 2 |   0.47186  -0.41287
 3 |    3e-015   -1e-015
 4 |    3e-015   -2e-015


R=
 U |         1         2         3         4
-------------------------------------------
 1 |    181.25    92.760   0.00000   0.00000
```

```
 2 |         0   0.44138   0.00000   0.00000
 3 |         0         0   0.00000   0.00000
 4 |         0         0         0   0.00000


V=
   |         1         2         3         4
-------------------------------------------
 1 |  -0.83666   0.23905  -0.11952  -0.47809
 2 |   0.00000  -0.80178  -0.53452  -0.26726
 3 |  -0.19044  -0.50974   0.82903  -0.12886
 4 |  -0.51355  -0.20042  -0.11271   0.82668


Piv=
   |   1   2   3   4
--------------------
 1 |   0   1   0   0
 2 |   0   0   1   0
 3 |   0   0   0   1
 4 |   1   0   0   0


Lind= 2
```

```
      if (rank < nc) {
        r2 = r[1:rank,1:rank];
        rinv = inv(r2);
        rinv = (rinv -> cons(rank,lind,0.)) |> cons(lind,nc,0.);
        print "Rinv=",rinv;
      } else {
        rinv = inv(r);
      }
      x2 = p * v‘ * rinv‘ * qtb;
      res = x - x2;
      print "Res=", res;
      ss1 = ssq(res); print "SS1=",ss1;
```

```
Res=
   |            1            2
------------------------------
 1 |   0.000e+000  -1.110e-016
 2 |  -5.551e-017   0.000e+000
 3 |  -3.469e-018   1.388e-017
 4 |   1.110e-016   5.551e-017
```

```
SS1= 3.102e-032
```

3.2 Function `hbaddtst`

gof = hbaddtst(data<,optn>)

Purpose: Test for additivity by Tukey (Nollau, 1975, chapter 4.5, A4.2) for the ANOVA model I.2:

$$y_{i,k} = \mu + \alpha_i + \beta_k + E_{i,k}$$

Input: **data** The $na \times nb$ data matrix is cross frequency table for two nominal scaled variables (factors) with na and nb levels.

optn specifies a vector of options:

1. Controls the printed output. Default is 0, i.e. no printed output.
2. The probablity α, default is $\alpha = 0.05$.

Output: There is only one return argument, the vector `gof`. The entries of the vector have the following meaning:

1. Indicating error return of the function. No error means zero.
2. T test value
3. F quantile of $F(\alpha, 1, (na-1)(nb-1)-1)$ distribution
4. p probability for test value T
5. $df1 = 1$
6. $df2 = (na-1)(nb-1)-1$

Restrictions:

1. The input data cannot have missing values.
2. Test is suitable for method 2 of ANOVA model I.

Relationships: anova(), glm()

Examples:

1. Example by Nollau, p. 211:

```
print "Additivitaetstest by Tukey: p. 211 and 221";
print "Model I.2: Y[i,k]= mu + alfa[i] + beta[k] + E[i,k]";
data = [ 4203 4201 4197 4198 4196 ,
         4205 4206 4201 4201 4200 ,
         4207 4208 4202 4194 4197 ,
         4206 4201 4211 4203 4194 ,
         4212 4204 4201 4202 4198 ,
         4222 4209 4211 4203 4201 ,
         4221 4219 4213 4214 4201 ,
```

```
          4214 4216 4202 4193 4198 ,
          4226 4217 4214 4215 4201 ,
          4217 4218 4206 4214 4213 ];
  optn = 2;
  gof = hbaddtst(data,optn);
  print "GOF=", gof;


                      ************************
                      Additivity Test by Tukey
                      ************************

    Model Y[j,k] = Mu + Alpha[j] + Beta[k] + Gamma[j,k] + E[j,k]
                 F=2.13667 <= F(0.95;1,35)=4.12134:
        H: All Gamma[j,k] = 0 cannot be rejected, p=0.152735


GOF=
                |         1
-------------------------
          Error |   0.00000
           Test |    2.1367
    F Quantile |    4.1213
   Probability |   0.15273
           NDF1 |   1.00000
           NDF2 |    35.000
         unused |   0.00000
         unused |   0.00000
```

3.3 Function `hbanova`

< gof,hyp,ci > = hbanova(data<,freq<,optn>>)

Purpose: Implements the following nine different methods of analysis of variance for models I and II (Nollau, chapters 4 and 7, A4.3, A4.4):

1. Model I.1: Einfache Klassifikation: (A,B) B: Wiederholungen Y[i,k] = mu + A[i] + E[i,k]
2. Model I.2: Zweifache Klassifikation ohne Wiederholungen (Kreuzklassifikation): (A,B) B: Levels of categorical variable Y[i,k] = mu + A[i] + B[k] + E[i,k]
3. Model I.3: Zweifache Klassifikation mit Wiederholungen (Kreuzklassifikation): (A,B,C) B: Levels of categorical variable C: equal number Wiederholungen Y[i,k,l] = mu + A[i] + B[k] + G[i,k] + E[i,k,l]

4. Model I.4: Dreifache Klassifikation ohne Wiederholungen: (A,B,C) B: Levels of categorical variable C: Levels of categorical variable Y[i,k,l] = mu + A[i] + B[k] + G[l] + C[i,k] + D[i,l] + Z[k,l] + E[i,k,l]
5. Model II.5: Einfache Klassifikation orthogonal: (A,B) B: Wiederholungen Y[i,k] = mu + A[i] + E[i,k]
6. Model II.6: Einfache Klassifikation nicht orthogonal: (A,B) B: Wiederholungen Y[i,k] = mu + A[i] + E[i,k]
7. Model II.7: Zweifache Klassifikation hierarchisch: (A,B,C) B: Levels of categorical variable C: equal number Wiederholungen Y[i,k,l] = mu + A[i] + B[i,k] + E[i,k,l]
8. Model II.8: Zweifache Klassifikation ohne Wiederholungen (Kreuzklassifikation): (A,B) B: Levels of categorical variable Y[i,k] = mu + A[i] + B[k] + E[i,k]
9. Model II.9: Zweifache Klassifikation mit Wiederholungen (Kreuzklassifikation): (A,B,C) B: Levels of categorical variable C: Wiederholungen Y[i,k,l] = mu + A[i] + B[k] + C[i,k] + E[i,k,l]

For all models the factor A is a categorical variable with `na` levels. The factors B and C can be either the levels of a categorical variable or the replications depending on the analysis model.

Input: data The input data object may be a vector, a $na \times nb$ matrix, or a $na \times nb \times nc$ tensor.

freq This is a frequency vector which should be specified only for models I.1 or II.6.

optn specifies a vector of options:

1. Controls the printed output. Default is 0, i.e. no printed output.
2. The probablity α, default is $\alpha = 0.05$.
3. Specifies an integer model number inbetween 1 and 9 to select one of the models mentioned above.

Output: There are no more than three return arguments,

gof a vector with entries having the following meaning:

1. Indicating error return of the function. No error means zero.

hyp Returns a table of the results of hypothesis.

ci Returns confidence intervals for model II parameters.

Restrictions:
1. The input data cannot have missing values.
2. .

Relationships: anova(), glmod()

Examples:
1. ANOVA Model 1.1: Y[i,k]= mu + alfa[i] + E[i,k]:

```
data = [ 11.2 11.8  6.3 17.5 14.9  1.6 ,
          8.9  9.2  3.6 14.3 12.9  2.4 ,
          7.3  7.1  3.7 12.6 17.1  3.8 ,
          8.8  7.6  6.0 10.9 17.6  4.2 ,
         11.1 10.7  9.7 12.3 20.2  7.3 ,
         12.3 12.4 11.6 13.6 19.1 11.9 ,
         12.7 14.4 13.2 12.2 20.4 16.8 ,
         13.1 14.6 13.4 14.0 20.4 15.1 ,
         13.2 12.0  9.7 14.3 17.9  9.1 ,
         12.7 10.2  6.7 15.8 17.9  4.4 ,
         14.8 12.2  6.2 18.7 20.1  4.5 ,
         13.1 14.0  7.6 17.2 16.0  2.2 ];


print "ANOVA Model I.1, p.195";
nr = nrow(data); freq = cons(nr,1,6.);
optn = [ 2, .05, 1 ];
< gof,hyp > = hbanova(data,freq,optn);
print "GOF=", gof;
print "Hypothesis=", gof;
```

For models I.1 and I.3 a preliminary Bartlett test is executed:

```
            ****************************************
            Bartlett Test: Equality of All Variances
            ****************************************

             Model Y[j,k] = Mu + Alpha[j] + E[j,k]

          ChiSqu=8.26955 <= ChiSqu(0.95;11)=19.6751:
       H: Equal Variances cannot be rejected: p=0.688974


                 ***************************
                 Variance Analysis Model I.1
                 ***************************


                   Hypothesis: ALPHA[I] = 0
              F=1.36706 <= F(0.95;11,60)=1.95221:
           Hypothesis cannot be rejected: p=0.212188
```

2. ANOVA Model 1.2: Y[i,k]= mu + alfa[i] + beta[k] + E[i,k]:

```
data = [ 4203 4201 4197 4198 4196 ,
         4205 4206 4201 4201 4200 ,
         4207 4208 4202 4194 4197 ,
         4206 4201 4211 4203 4194 ,
         4212 4204 4201 4202 4198 ,
         4222 4209 4211 4203 4201 ,
         4221 4219 4213 4214 4201 ,
         4214 4216 4202 4193 4198 ,
         4226 4217 4214 4215 4201 ,
         4217 4218 4206 4214 4213 ];


freq = .;
optn = [ 2, .05, 2 ];
< gof,hyp > = hbanova(data,freq,optn);
print "GOF=", gof;
print "Hypothesis=", hyp;
```

For model I.2 a preliminary Tukey test is executed:

```
                        ************************
                        Additivity Test by Tukey
                        ************************

 Model Y[j,k] = Mu + Alpha[j] + Beta[k] + Gamma[j,k] + E[j,k]
               F=2.13667 <= F(0.95;1,35)=4.12134:
     H: All Gamma[j,k] = 0 cannot be rejected, p=0.152735


                     ***************************
                     Variance Analysis Model I.2
                     ***************************

                       Hypothesis: ALPHA[I] = 0
                    F=8.24997 > F(0.95;9,36)=2.15261:
                 Hypothesis is rejected: p=1.56402e-006

                       Hypothesis: BETA[J] = 0
                    F=13.82 > F(0.95;4,36)=2.63353:
                 Hypothesis is rejected: p=6.34246e-007
```

3. ANOVA Model I.3: Tensor data p. 224:

```
print "Cross Classification with Equal Number Replications";
print "Y[i,k] = mu + alfa[i] + beta[k] + gamma[i,k] + E[i,k]";
```

```
/* Input are Tensor Data d[na,nb,nc]: nc are replications */
real d[5,6,2];
d[1,,]= [  81.0  80.7, 146.6 100.4,  82.3 103.1,
          119.8  98.9,  98.9  66.4,  86.9  67.7 ];
d[2,,]= [ 105.4  82.3, 142.0 115.5,  77.3 105.1,
           121.4  61.9,  89.0  49.9,  77.1  66.7 ];
d[3,,]= [ 119.7  80.4, 150.7 112.2,  78.4 116.5,
          124.0  96.2,  69.1  96.7,  78.9  67.4 ];
d[4,,]= [ 109.7  87.2, 191.5 147.7, 131.3 139.9,
          140.8 125.5,  89.3  61.9, 101.8  91.8 ];
d[5,,]= [  98.3  84.2, 145.7 108.1,  89.6 129.6,
          124.8  75.7, 104.1  80.3,  96.0  94.1 ];
freq = .; optn = [ 2 .05 3 ];
< gof,hyp > = hbanova(d,freq,optn);
print "GOF=", gof;
print "Hypothesis=", hyp;


            ****************************************
            Bartlett Test: Equality of All Variances
            ****************************************

             Model Y[j,k] = Mu + Alpha[j] + E[j,k]

           ChiSqu=18.2304 <= ChiSqu(0.95;29)=42.557:
       H: Equal Variances cannot be rejected: p=0.939442


                 ***************************
                 Variance Analysis Model I.3
                 ***************************

                   Hypothesis: ALPHA[I] = 0
                F=2.89247 > F(0.95;4,30)=2.68963:
              Hypothesis is rejected: p=0.0388184

                   Hypothesis: BETA[J] = 0
                F=9.24762 > F(0.95;5,30)=2.53355:
             Hypothesis is rejected: p=2.07543e-005

                Hypothesis: ALPHA - BETA[I,J] = 0
              F=0.482954 <= F(0.95;20,30)=1.93165:
            Hypothesis cannot be rejected: p=0.953458


GOF=
          |         1
```

```
----------------------
     Error |   0.00000
      Mean |    101.09
    Mean_A |    2.8925
    Mean_B |    9.2476
    Mean_C |       .
  Mean_A*B |   0.48295
  Mean_A*C |       .
  Mean_B*C |       .
   Sigma_A |       .
   Sigma_B |       .
   Sigma_C |       .
   Sigma_E |    458.95
     Rho_1 |       .
     Rho_2 |       .
Bartl_Test |    18.230
Bartl_Qunt |    42.557
Bartl_Prob |   0.93944
    unused |   0.00000
    unused |   0.00000
    unused |   0.00000
```

```
Hypothesis=
                         |   Test Value         Ndf1         Ndf2
-------------------------------------------------------------------
           ALPHA[I] = 0 |       2.8925       4.0000       30.000
            BETA[J] = 0 |       9.2476       5.0000       30.000
  ALPHA - BETA[I,J] = 0 |          .            .            .
 ALPHA - GAMMA[I,K] = 0 |      0.48295       20.000       30.000
  BETA - GAMMA[J,K] = 0 |          .            .            .



                         |    Fquantile  Probability
-------------------------------------------------
           ALPHA[I] = 0 |       2.6896       0.0388
            BETA[J] = 0 |       2.5336       0.0000
  ALPHA - BETA[I,J] = 0 |          .            .
 ALPHA - GAMMA[I,K] = 0 |       1.9317      0.95346
  BETA - GAMMA[J,K] = 0 |          .            .
```

4. ANOVA Model I.4: p. 374:

```
print "3-Way Classification without replications";
print "Y[i,k,l] = mu + a[i] + b[k] + g[l] + d[i,k] + e[i,l] + z[k,l] + E[i,k,l]";
```

```
print "No Data Example in Book";
```

5. ANOVA Model II.5: p. 284, chapter 7.1:

```
print "1-way Classification: Orthogonal Case";
print "Equal number replications";
print "Y[i,k] = mu + A[i] + E[i,k]";
freq = .; optn = [ 2 .05 5 ];
print "No Data Example in Book";
```

6. ANOVA Model II.6: p. 284, chapter 7.1:

```
print "1-way Classification: Nonorthogonal Case";
print "Y[i,k] = mu + alfa[i] + E[i,k]";
data = [  727.7  769.0 ,  644.8  634.6 ,
         1138.3  906.7 ,  723.2  679.6 ,
          590.6  646.6 ];
freq = [ 2 2 2 2 2 ]; optn = [ 2 .05 6 ];
< gof,hyp,ci > = hbanova(data,freq,optn);
print "GOF=", gof;
print "Hypothesis=", hyp;
print "Confidence Intervals=", ci;


                  ****************************
                  Variance Analysis Model II.6
                  ****************************

              Mean = 746.11 (Sigma_A)**2 = 52988.7
                     (Sigma_E)**2 = 6048.52
                Intra Class Correlation: 0.795094

                   Hypothesis: (SIGMA_A)**2 = 0
                 F=8.76059 > F(0.95;4,5)=5.19217:
                Hypothesis is rejected: p=0.0175639


optn = [ 2 .01 6 ];
< gof,hyp,ci > = hbanova(data,freq,optn);
print "GOF=", gof;
print "Hypothesis=", hyp;
print "Confidence Intervals=", ci;


                  ****************************
                  Variance Analysis Model II.6
```

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

Mean = 746.11 (Sigma_A)**2 = 52988.7
       (Sigma_E)**2 = 6048.52
  Intra Class Correlation: 0.795094

    Hypothesis: (SIGMA_A)**2 = 0
  F=8.76059 <= F(0.99;4,5)=11.3919:
Hypothesis cannot be rejected: p=0.0175639
```

7. ANOVA Model II.7: p. 298, 303, chapter 7.3:

```
print "Hierarchical Classification: Equal Number Replications";
print "Y[i,k,l] = mu + A[i] + B[i,k] + E[i,k,l]";
real d[10,2,2];
d[ 1,,] = [ 59.8 61.2, 60.0 65.0 ];
d[ 2,,] = [ 65.0 65.8, 64.5 64.5 ];
d[ 3,,] = [ 65.0 65.2, 65.5 63.5 ];
d[ 4,,] = [ 62.5 61.9, 60.9 61.5 ];
d[ 5,,] = [ 59.8 60.9, 56.0 57.2 ];
d[ 6,,] = [ 68.8 69.0, 62.5 62.0 ];
d[ 7,,] = [ 65.2 65.5, 61.0 59.3 ];
d[ 8,,] = [ 59.6 58.5, 62.3 61.5 ];
d[ 9,,] = [ 61.0 64.0, 73.0 71.7 ];
d[10,,] = [ 65.0 64.0, 62.0 63.0 ];


freq = .; optn = [ 2 .05 7 ];
< gof,hyp,ci > = hbanova(d,freq,optn);
print "GOF=", gof;
print "Hypothesis=", hyp;
print "Confidence Intervals=", ci;
```

```
***************************
Variance Analysis Model II.7
***************************

    Mean = 63.115 (Sigma_A)**2 = 2.12014
 (Sigma_B)**2 = 9.37175 (Sigma_E)**2 = 1.3205
Intra Class Correlations 0.896936 and 0.165476

    Hypothesis: (SIGMA_A)**2 = 0
  F=1.42268 <= F(0.95;9,10)=3.02038:
Hypothesis cannot be rejected: p=0.294493
```

```
                Hypothesis: (SIGMA_B)**2 = 0
              F=15.1942 > F(0.95;10,20)=2.34788:
            Hypothesis is rejected: p=2.90441e-007


                Confidence Interval for Mean
       Value=63.115 Lower Bound=61.204 Upper Bound=65.026

              Confidence Interval for (Sigma_E)**2
      Value=1.3205 Lower Bound=0.772909 Upper Bound=2.75369

       Confidence Interval for (Sigma_B)**2 / (Sigma_E)**2
      Value=7.09712 Lower Bound=2.23901 Upper Bound=25.4711
```

GOF=

	1
Error	0.00000
Mean	63.115
Mean_A	28.545
Mean_B	20.064
Mean_C	51.250
Mean_A*B	.
Mean_A*C	.
Mean_B*C	.
Sigma_A	2.1201
Sigma_B	9.3717
Sigma_C	.
Sigma_E	1.3205
Rho_1	0.89694
Rho_2	0.16548
unused	0.00000
unused	0.00000

Hypothesis=

	Test Value	Ndf1	Ndf2
(SIGMA_A)**2 = 0	1.4227	9.0000	10.0000
(SIGMA_B)**2 = 0	15.194	10.0000	20.000
(SIGMA_AB)**2 = 0	.	.	.

	Fquantile	Probability
(SIGMA_A)**2 = 0	3.0204	0.29449

```
  (SIGMA_B)**2 = 0 |       2.3479       3e-007
 (SIGMA_AB)**2 = 0 |         .            .


Confidence Intervals=
                                  |     Value  Lower CI  Upper CI
-------------------------------------------------------------------
                         CI Mean  |    63.115    61.204    65.026
                CI (Sigma_E)**2   |    1.3205   0.77291    2.7537
 CI (Sigma_B)**2 / (Sigma_E)**2   |    7.0971    2.2390    25.471
```

8. ANOVA Model II.8: p. 292, chapter 7.2:

```
print "Cross Classification: Without Replications";
print "Y[i,k] = mu + A[i] + B[k] + E[i,k]";
freq = .; optn = [ 2 .05 8 ];
print "No Data Example in Book";
```

9. ANOVA Model II.9: p. , chapter 7.2:

```
print "Cross Classification: Equal Number Replications";
print "Y[i,k,l] = mu + A[i] + B[k] + C[i,k] + E[i,k,l]";
freq = .; optn = [ 2 .05 9 ];
print "No Data Example in Book";
```

3.4 Function `hbbartlett`

gof = hbbartlett(data<,freq<,optn>>)

Purpose: The Bartlett test (Nollau, chapter 4.4, A4.1) can be used in connection with the ANOVA Model I to test the Hypothesis: $\sigma_1^2 = \ldots = \sigma_m^2$ for m independent samples from a normal distributed population with size $n_i \geq 2$, $i = 1, \ldots, m$. The Bartlett test refers to models I.1 and I.2:

$$Y_{i,k} = \mu + \alpha_i + E_{i,k}$$

Input: **data** The data set is either an $N = \sum_i^m n_i$ vector or a matrix with m rows and $\max_i n_i$ columns containing the n_i measurements in its first n_i columns.

freq This is a m vector of integer frequencies n_i for the m samples.

optn This argument is optional and should be either a scalar or vector imposing the following run time options:

1. specifies the amount of printed output (default is 0: no printed output)

Output: There is only one return argument, the vector `gof`. The entries of the vector have the following meaning:

1. Indicating error return of the function. No error means zero.
2. T test value
3. χ^2 quantile of $\chi^2(\alpha, m-1)$ distribution
4. p probability for test value T
5. $df = m - 1$

Restrictions: 1. The input data cannot have missing values.
2. Test is suitable for methods 1 and 3 of ANOVA model I.

Relationships: anova(), hbanova(), glmod()

Examples: 1. Example by Nollau p. 195:

```
print "Bartlett Test: p. 195: t=8.27, chi=19.7, ndf=11";
print "Model 1.1: Y[i,k]= mu + alfa[i] + E[i,k]";
data = [ 11.2 11.8  6.3 17.5 14.9  1.6 ,
          8.9  9.2  3.6 14.3 12.9  2.4 ,
          7.3  7.1  3.7 12.6 17.1  3.8 ,
          8.8  7.6  6.0 10.9 17.6  4.2 ,
         11.1 10.7  9.7 12.3 20.2  7.3 ,
         12.3 12.4 11.6 13.6 19.1 11.9 ,
         12.7 14.4 13.2 12.2 20.4 16.8 ,
         13.1 14.6 13.4 14.0 20.4 15.1 ,
         13.2 12.0  9.7 14.3 17.9  9.1 ,
         12.7 10.2  6.7 15.8 17.9  4.4 ,
         14.8 12.2  6.2 18.7 20.1  4.5 ,
         13.1 14.0  7.6 17.2 16.0  2.2 ];
nr = nrow(data); freq = cons(nr,1,6.);
optn = 2;
gof = hbbartlett(data,freq,optn);
print "GOF=", gof;
```

```
**************************************
Bartlett Test: Equality of All Variances
**************************************

 Model Y[j,k] = Mu + Alpha[j] + E[j,k]

    ChiSqu=0 <= ChiSqu(0.95;11)=0:
H: Equal Variances cannot be rejected: p=0
```

```
GOF=
                |         1
--------------------------
          Error |   0.00000
           Test |    8.2696
  ChiSqu Quant |    19.675
   Probability |   0.68897
            NDF |    11.000
         unused |   0.00000
         unused |   0.00000
         unused |   0.00000
```

3.5 Function `hbcovar`

gof = hbcovar(ydata,wdata<,freq<,optn>>)

Purpose: There are two models for covariance analysis with one covariable implemented (see Nollau, 1975, chapters 5.1 and 5.2, A5):

1. One way (1,1) classification with one covariable and
2. unrepeated two way (2,1) classification with one covariable.

Chapter 5.3 also describes the (2,2) classification model.

Input: ydata 1. For (1,1) classification this can be
- either a vector Y of size `ns`, where $ns = \sum_j f_j, j = 1, \ldots, na$
- or a $na \times nb$ matrix, where $nb = sumf_j$

2. For (2,1) classification this is a $na \times nb$ cross frequency table Y of two categorical variables with na and nb levels.

wdata This is a second data object W of the same size as the first input argument Y.

freq valid only (1,1) classification: specifies a vector $f[na]$ of the frequencies f_j of measurements for each level of the data in Y and W.

optn specifies a vector of options:

1. Controls the printed output. Default is 0, i.e. no printed output.
2. The probablity α, default is $\alpha = 0.05$.

Output: The only return argument is a vector with the following entries:

1. Indicating error return of the function. No error means zero.
2. Test value for $\gamma = 0$.
3. Test criterion for $\gamma = 0$.

4. Quantile for testing equality of the means of factor A.
5. Criterion for testing equality of the means of factor A.
6. Quantile for testing equality of the means of factor B.
7. Criterion for testing equality of the means of factor B.

Restrictions: 1. The input data cannot have missing values.
2.

Relationships: anova(), glmod()

Examples: 1. Covariance Analysis: Model (1,1): p. 247:

```
print "Covariance Analysis: Model (1,1): p. 247";
Y = [ 766.8 854.0 1080.0 954.6 802.5 862.5 501.6 417.6 595.2 1033.2 ,
      612.0 942.0  365.2 276.0 731.5 487.2 369.2 498.0 464.8 667.4 ];
W = [ 7.1 7.0 7.5 7.4 7.5 7.5 4.4 4.3 6.2 8.2,
      7.2 6.2 4.4 4.0 7.7 5.6 5.2 5.3 5.6 7.1 ];
Bfrq = [ 10 10 ];
optn = 2;
gof = hbcovar(Y,W,Bfrq,optn);
print "GOF=",gof;
```

```
   ************************************************************
   (1,1) Classification: One factor A=2 with one Covariable
   ************************************************************

 Number Replications
 -------------------

   1 :        10       10

               ABS(T=6.51661) > T(0.975,17)=2.10982:
               H: Gamma = 0 is rejected: p=0.999997
                 F=4.98054 > F(0.95;1,17)=4.45132:
         H: Equal Means of Factor A is rejected, p=0.0393832

GOF=
           |         1
-----------------------
     Error |   0.00000
 TestGamma |    6.5166
 CritGamma |    2.1098
    Test_A |    4.9805
    Crit_A |    4.4513
    unused |    .
```

2. Covariance Analysis: Model (2,1): p. 254:

```
print "Covariance Analysis: Model (2,1): p. 254";
Y = [ 2.8 1.9 0.8 0.5 2.1 0.9 ,
      3.0 5.4 2.8 1.0 0.5 1.1 ,
      4.5 0.9 1.9 0.1 3.2 3.2 ,
      2.3 0.2 3.3 0.6 2.2 2.8 ];
W = [ 50 49 51 50 50 50 ,
      51 53 53 50 48 48 ,
      52 47 51 49 52 52 ,
      48 45 53 48 50 50 ];
optn = 2;
gof = hbcovar(Y,W,.,optn);
print "GOF=",gof;
```

```
*****************************************************************
(2,1) Classification: Two factors (A=4,B=6) with one Covariable
*****************************************************************

             ABS(T=12.6166) > T(0.975,14)=2.14479:
                 H: Gamma = 0 is rejected: p=1
               F=6.72921 > F(0.95;3,14)=3.34389:
      H: Equal Means of Factor A is rejected, p=0.00485774
               F=19.6477 > F(0.95;5,14)=2.95825:
     H: Equal Means of Factor B is rejected, p=7.03401e-006
```

```
GOF=
           |         1
-----------------------
     Error |   0.00000
 TestGamma |    12.617
 CritGamma |    2.1448
    Test_A |    6.7292
    Crit_A |    3.3439
    Test_B |    19.648
    Crit_B |    2.9582
    unused |     .
```

3.6 Function `hbdiscrim`

< gof,fsep,yhat,zhat> = hbdiscrim(data,idc<,optn<,zdat>>)

Purpose: The code in appendix A3 refers only to two sample discriminance

analysis (Nollau, 1975, chapter 6.1). However, chapter 6.2 also treats the general case of $K > 2$ samples.

Input: data The input data is in the form of a $nr \times nc$ matrix which contains a identity column. The identity column determines to which sample the observation belongs. The discriminance analysis is then perfomed w.r.t. only $nc - 1$ columns of the data matrix.

idc specifies the number of the identity column. (The data of this column is not included in the analysis.)

optn specifies a vector of options:

1. Controls the printed output. Default is 0, i.e. no printed output.
2. The probablity α, default is $\alpha = 0.05$.
3. If entry three is set unequal zero, then the additional data set `zdat` also contains a column `idc` specifying priors. If this entry is missing or zero, then `zdat` does not contain priors.

zdat specifies an additional data set with either nc or $nc-1$ columns (see `optn[]`).

Output: gof The entries of this vector have the following meaning:

1. Indicating error return of the function. No error means zero.
2. The second entry contains the separation threshold for the posteriors.
3. The third entry gives the Mahalanobis test value.
4. The fourth entry gives the corresponding critical F value.
5. The fifth entry gives the p value.

fsep is a vector of $nv = nc-1$ coefficients of the linear separator function.

yhat This is a vector of nr values computed as the predicted values of the separator function and which permits the prediction of the sample.

zhat This is a vector of predicted function values for the `zdat` input data matrix.

Restrictions:

1. The input data cannot have missing values.
2. The columns of `data` and `zdat` must be compatible except for the `idc` column.

Relationships: discrim()

Examples: 1. Discriminance Analysis: p. 267:

```
X = [ 63.6  2.096 1, 59.2  2.048 1, 59.2  2.144 1, 64.4  2.256 1,
      64.0  2.064 1, 65.6  2.264 1, 66.4  2.144 1, 66.8  1.984 1,
      67.2  2.192 1, 67.6  2.240 1, 67.2  2.032 1,
      60.0  1.952 2, 56.0  1.856 2, 56.4  1.984 2, 57.6  1.920 2,
      59.6  2.032 2, 57.2  2.016 2, 56.4  1.984 2, 54.4  1.984 2,
```

```
      50.4  1.776 2, 45.6  1.616 2, 54.8  1.680 2 ];

optn = 2;
< gof,tren,yhat > = hbdisc(X,3,optn);
print "GOF=",gof;
print "Tren=",tren;
print "Yhat=",yhat;
```

```
          ***************************************
          Discriminance Aanalysis for K=2 Samples
          ***************************************
 Threshold Separating Samples. . . . . .      47.8876
 Mahalanobis Distance. . . . . .      7.01903
 Test Value for Mahalanobis Distance . . . . . .      18.3372
 Critical F(4.71301e-296,2,19) Value . . . . . .      3.52189
 Probability . . . . . . . . . . . . . . . . . . 3.66801e-005
```

```
              Linear Discriminance Function

                        VAR_1        VAR_2
                1  0.590377168  6.200649587
```

```
             Prior and Posterior Probability
             *******************************

               N    Name Prior        Yhat  Post

               1  OBS_01     0 50.54454941     0
               2  OBS_02     0 47.64925869     1
               3  OBS_03     0 48.24452105     0
               4  OBS_04     0 52.00895508     0
               5  OBS_05     0 50.58227949     0
               6  OBS_06     0 52.76701288     0
               7  OBS_07     0 52.49523666     0
               8  OBS_08     0 51.73928359     0
               9  OBS_09     0 53.26516957     0
              10  OBS_10     0 53.79895162     0
              11  OBS_11     0 52.27306564     0
              12  OBS_12     1 47.52629807     1
              13  OBS_13     1 44.56952703     1
              14  OBS_14     1 45.59936105     1
              15  OBS_15     1 45.91097208     1
              16  OBS_16     1 47.78619917     1
              17  OBS_17     1 46.27008357     1
```

```
 18  OBS_18     1 45.59936105     1
 19  OBS_19     1 44.41860671     1
 20  OBS_20     1 40.76736293     1
 21  OBS_21     1 36.94144859     1
 22  OBS_22     1 42.76976010     1


*******************************
Evaluation of Training Data Fit
*******************************

Index                              Value       StdErr
Absolute Classification Error          1            .
Classification Accuracy        95.45454545          .
Goodman-Kruskal Gamma          1.000000000  0.000000000
Kendall Tau_a                  0.000000000          .
Kendall Tau_b                  0.912870929  0.081417368
Stuart Tau_c                   0.909090909  0.086678417
Somers D C|R                   0.909090909  0.086678417
Somers D R|C                   0.916666667  0.079785592

          Classification Table
          --------------------

                |    Predicted
      Observed  |    Event   NoEvt
      ----------|-----------------
         Event  |       10       1
         NoEvt  |        0      11
       Missing  |        0       0
```

3.7 Function `hblreg`

< gof,parm,yhat,zhat,test> = hblreg(xdat,ydat<,optn<,zdat<,cont>>>)

Purpose: This function implements the common multiple linear least squares regression (a simple version of the `reg` function), see Nollau, 1975, chapters 3.1 to 3.3, A2.2.

Input: **xdat** specifies an $m \times n$ data matrix X where the m rows correspond to observations and the n columns to predictor variables. Note, that the function assumes an intercept column (constant ones) in X.

ydat specifies an m response vector.

optn specifies a vector of options:

1. Controls the printed output. Default is 0, i.e. no printed output.
2. The probablity α, default is $\alpha = 0.05$.

zdat specifies an additional $r \times n$ data set which should be scored with coefficients computed from (x, y).

cont The fifth input argument specifies an optional contrast vector of size n.

Output: gof The entries of this vector have the following meaning:

1. Indicating error return of the function. No error means zero.
2. Returns the error variance. Note that the `reg` function computes many more indicators for the goodness of fit.

parm returns the $n \times 3$ matrix with the estimates for coefficients of the linear function together with lower and upper confidence limits.

yhat returns the $m \times 3$ matrix of predicted values $\hat{y}$ together with lower and upper confidence limits.

zhat if the fourth input argument z is specified, this $r \times 3$ matrix contains the predicted values together with lower and upper confidence limits.

test if the fifth input argument *cont* is specified, this n vector returns the test vector for the contrast.

Restrictions:

1. The input data cannot have missing values.
2. The size of input arguments 1,2,4, and 5 must be compatible.

Relationships:

Examples: reg()

1. Example p. 137:

```
print "Lineare Regression, p. 137";
x = [ 0  20  40  60  80  100 ];
x = cons(6,1,1.) -> x';
y = [ 70.7  88.3  104.9  124.7  148.0  176.0 ];
optn = 2;
< gof,parm,yhat > = hblreg(x,y,optn);
print "GOF=",gof;
print "Parm=",parm;
print "Yhat=",yhat;
```

```
****************************************
Linear Regression: Error Variance=19.7842
****************************************
```

```
                  Linear Regression Coefficients
                  ******************************

                      Dense Matrix (2 by 3)

                    |    Estimate   Lower CI   Upper CI
                  --------------------------------------
                  1 |   66.952381  58.014493  75.890269
                  2 |   1.0362857  0.8886813  1.1838901


                   Predicted Values of Response
                   ****************************

                      Dense Matrix (6 by 3)

                    |    Estimate   Lower CI   Upper CI
                  --------------------------------------
                  1 |   66.952381  58.014493  75.890269
                  2 |   87.678095  80.967911  94.388279
                  3 |   108.40381  103.15053  113.65709
                  4 |   129.12952  123.87625  134.38280
                  5 |   149.85524  143.14505  156.56542
                  6 |   170.58095  161.64306  179.51884

GOF=
              |         1
  ----------------------
       Error |   0.00000
   EVariance |    19.784
      unused |     .
      unused |     .
      unused |     .
      unused |     .
      unused |     .
      unused |     .
```

2. Example p. 142:

```
X = [ 6.44  6.44 11.26 11.26 12.87 14.48 16.09 16.09 16.09 17.70
     17.70 19.31 19.31 19.31 19.31 20.92 20.92 20.92 20.92 22.53
     22.53 22.53 22.53 24.14 24.14 24.14 25.74 25.74 27.35 27.35
     28.96 28.96 28.96 28.96 30.57 30.57 30.57 32.18 32.18 32.18
     32.18 32.18 35.40 37.01 38.62 38.62 38.62 38.62 40.23 ];
Y = [  .61  3.05  1.22  6.71  4.88  3.05  7.93  5.49 10.37  8.54
```

```
      5.18  6.10  4.27  7.32  8.54 10.37  7.93 10.37 14.03 10.98
      7.93 18.30 24.40 16.47  7.93  6.10  9.76 12.20 15.25 12.20
     17.08 25.62 23.18 12.81 20.74 14.03 10.98 14.64 17.08 19.52
     15.86  9.76 20.13 16.47 28.36 21.35 36.60 28.06 25.92 ];
nr = ncol(X);
X = cons(nr,1,1.) -> X';
optn = 2;
< gof,parm,yhat > = hblreg(X,Y,optn);
print "GOF=",gof;
print "Parm=",parm;
print "Yhat=",yhat;
```

```
****************************************
Linear Regression: Error Variance=21.8641
****************************************

      Linear Regression Coefficients
      ******************************

          Dense Matrix (2 by 3)

        |    Estimate   Lower CI   Upper CI
    ------------------------------------
    1 |  -5.3512523 -9.4850517 -1.2174530
    2 |   0.7493058  0.5912091  0.9074024

       Predicted Values of Response
       ****************************

          Dense Matrix (49 by 3)

        |    Estimate   Lower CI   Upper CI
    ------------------------------------
    1 |  -0.5257231 -3.7139053  2.6624591
    2 |  -0.5257231 -3.7139053  2.6624591
    3 |   3.0859308  0.5682040  5.6036575
    4 |   3.0859308  0.5682040  5.6036575
    5 |   4.2923131  1.9858293  6.5987969
    6 |   5.4986954  3.3938525  7.6035383
    7 |   6.7050777  4.7892393  8.6209161
    8 |   6.7050777  4.7892393  8.6209161
    9 |   6.7050777  4.7892393  8.6209161
   10 |   7.9114600  6.1678756  9.6550444
   11 |   7.9114600  6.1678756  9.6550444
   12 |   9.1178423  7.5243200  10.711365
```

```
 13 |   9.1178423  7.5243200  10.711365
 14 |   9.1178423  7.5243200  10.711365
 15 |   9.1178423  7.5243200  10.711365
 16 |   10.324225  8.8517720  11.796677
 17 |   10.324225  8.8517720  11.796677
 18 |   10.324225  8.8517720  11.796677
 19 |   10.324225  8.8517720  11.796677
 20 |   11.530607  10.142624  12.918590
 21 |   11.530607  10.142624  12.918590
 22 |   11.530607  10.142624  12.918590
 23 |   11.530607  10.142624  12.918590
 24 |   12.736989  11.389973  14.084006
 25 |   12.736989  11.389973  14.084006
 26 |   12.736989  11.389973  14.084006
 27 |   13.935878  12.582557  15.289200
 28 |   13.935878  12.582557  15.289200
 29 |   15.142261  13.735924  16.548598
 30 |   15.142261  13.735924  16.548598
 31 |   16.348643  14.847421  17.849865
 32 |   16.348643  14.847421  17.849865
 33 |   16.348643  14.847421  17.849865
 34 |   16.348643  14.847421  17.849865
 35 |   17.555025  15.924343  19.185708
 36 |   17.555025  15.924343  19.185708
 37 |   17.555025  15.924343  19.185708
 38 |   18.761408  16.974186  20.548630
 39 |   18.761408  16.974186  20.548630
 40 |   18.761408  16.974186  20.548630
 41 |   18.761408  16.974186  20.548630
 42 |   18.761408  16.974186  20.548630
 43 |   21.174172  19.017099  23.331245
 44 |   22.380555  20.019045  24.742065
 45 |   23.586937  21.012044  26.161830
 46 |   23.586937  21.012044  26.161830
 47 |   23.586937  21.012044  26.161830
 48 |   23.586937  21.012044  26.161830
 49 |   24.793319  21.998146  27.588492
```

3.8 Function `hbltst`

gof = hbltst(xdat,ydat,freq<,optn>)

Purpose: This function implements the linearity test by R. A. Fisher (Nollau, 1975, chapter 2.9, A2.1).

Input: **xdat** specifies an $m \times n$ data matrix X where the m rows correspond to observations and the n columns to predictor variables. Note, that the function assumes an intercept column (constant ones) in X.

ydat specifies an m response vector Y.

optn specifies a vector of options:

1. Controls the printed output. Default is 0, i.e. no printed output.
2. The probablity α, default is $\alpha = 0.05$.

Output: There is only one return argument, the vector `gof`. The entries of the vector have the following meaning:

1. Indicating error return of the function. No error means zero.
2. Test value for the Hypothesis
3. Critical value F value for the test
4. Probability

Restrictions: 1. The input data cannot have missing values.

2.

Relationships:

Examples: 1. Example p. 130:

```
print "Linearitaetstest von R. A. Fisher, p. 130-133";
X = [ 1  240 , 5  240 , 1  280 , 5  280 ];
Y = [ 61.35 63.25 65.60 64.70 67.45 70.85 78.60 74.30 ];
freq = [ 2 2 2 2 ];

nr = nrow(X); id = cons(nr,1,1.);
X1 = id -> X;
optn = 2;
gof = hbltst(X1,Y,freq,optn);
print "GOF=",gof;


                  **********************************
                  Test for Linearity by R. A. Fisher
                  **********************************

                  F=2.29794 <= F(0.95;1,4)=7.70865:
            Linearity Hypothesis not rejected (p=0.20413)


GOF=
        |         1
-------------------
```

```
 Error |   0.00000
  Test |    2.2979
FQuant |    7.7086
  Prob |   0.20413
unused |    .
unused |    .
```

3.9 Function `hbqlrg`

< gof,parm,yhat> = hblqrg(xdat,ydat,mpow,freq,<,optn>)

Purpose: Simple (one predictor) polynomial regression (Nollau, 1975, chapter 3.5, A2.3). There are two versions of the algorithm implemented:

1. The third input argument `mpow` specifies the degree of the polynomial, i.e. coefficients $parm_0, \ldots, parm_{mpow}$ are returned.
2. The third input argument `mpow` specifies an upper range for the power. Then the algorithm computes stepwise for $m = 1, \ldots, mpow$ each polynomial fit and applies Fisher's linearity test. The algorithm terminates for $m <= mpow$ with the smallest degree for which the hypothesis of linearity is not rejected.

Note, that the function assumes a 2-column matrix X with constant ones in the first column (intercept).

Input: **xdat** specifies an $m \times 2$ matrix containing constant ones in its first column and the predictor variable in its second column.

ydat specifies an $msum$ vector of response values.

mpow specifies the (maximum) degree of the polynomial.

freq specifies a m vector of frequencies $f_i \geq 1$, where $\sum_{i=1}^{m} f_i = msum$. That means, this vector maps the y_i values to the rows of X.

optn specifies a vector of options:

1. Controls the printed output. Default is 0, i.e. no printed output.
2. The probablity α, default is $\alpha = 0.05$.
3. If the third entry is not missing and unequal to zero, the third argument specifies the degree of the polynomial and the linearity test is not performed.

Output: **gof** The entries of the vector have the following meaning:

1. Indicating error return.
2. Returns the integer $ndeg$ sepcifying the degree of the computed polynomial.

parm returns the $ndeg + 1$ estimated coefficients c_k of the polynomial $f = c_0 + c_1 x + \ldots + c_{ndeg} x^{ndeg}$.

yhat Returns a $msum$ vector of predicted values.

Restrictions: 1. The input data cannot have missing values and the frequencies should not be smaller than one.

2. The input arguments 1,2, and 4 must be compatible.

Relationships: nlreg(), lreg(), nlp()

Examples: 1. QuasiLineare Regression, p. 165:

```
print "ndeg=2: f2=0.192, Fcrit(.95,8,22)=2.55";
print "alf0= 404.0, alf1=-1.386, alf2=.001";
data = [ 100 230 290 310 ,  150 195 240 250 ,
         200 150 190 195 ,  250 115 140 145 ,
         300  80 100 105 ,  350  60  70  75 ,
         400  40  50  50 ,  450  35  40  40 ,
         500  25  30  30 ,  550  20  20  20 ,
         600  15  15  15 ];
nr = nrow(data); freq = cons(nr,1,3);
x = data[,1];
y = data[,2:4]; y = shape(y,.,1);  /* print "Y=", y; */


print "Specify Degree=2";
grd = 2; optn = [ 2 . 1 ];
< gof,parm > = hbqlreg(x,y,grd,freq,optn);
print "GOF=",gof;
print "Parm=",parm;
```

```
**********************************
Quasi Linear Regression of Degree 2
**********************************


     Coefficients of Polynomial
     **************************

       Dense Matrix (3 by 3)

      | Coefficient F Quantile Crit Value
    ------------------------------------
GRAD_0 |   404.03030     .          .
GRAD_1 |  -1.3858741  8.3126949  2.3419373
GRAD_2 |   0.0012448  0.1918109  2.3965033
```

```
GOF=
       |         1
------------------
 Error |   0.00000
  Grad |    2.0000
unused |    .
unused |    .
unused |    .
unused |    .


Parm=
         |  Coefficient   F Quantile   Crit Value
-------------------------------------------------
  GRAD_0 |       404.03         .            .
  GRAD_1 |      -1.3859       8.3127       2.3419
  GRAD_2 |        0.001      0.19181       2.3965
```

```
print "Search for Best Degree in 1,...5";
grd = 5; optn = 2;
< gof,parm > = hbqlreg(x,y,grd,freq,optn);
print "GOF=",gof;
print "Parm=",parm;
```

```
          ********************************************
          Polynomial Regression up to Degree 5 : Found 2
          ********************************************


                    Coefficients of Polynomial
                    **************************

                      Dense Matrix (3 by 3)

                    | Coefficient F Quantile Crit Value
                   -----------------------------------
             GRAD_0 |   404.03030  0.0000000     .
             GRAD_1 |  -1.3858741  0.0000000  8.3126949
             GRAD_2 |   0.0012448  0.0000000  0.1918109
```

3.10 Function `hbrcmp`

< gof,parm> = hbrcmp(x1,y1,x2,y2<,optn>)

Purpose: Compare the coefficients of two simple linear regressions (Nollau, 1975, chapter 3.3, A2.4). There are the following three tests provided:

Input: **x1** $n_1 \times 2$ matrix for sample 1: containing ones in its foirst column and one predictor column

y1 n_1 response vector for sample 1

x2 $n_2 \times 2$ matrix for sample 2: containing ones in its foirst column and one predictor column

y2 n_2 response vector for sample 2

optn specifies a vector of options:

1. Controls the printed output. Default is 0, i.e. no printed output.
2. The probablity α, default is $\alpha = 0.05$.

Output: **gof**
1. n_1 size of sample 1
2. n_2 size of sample 2
3. σ_1^2 test variance for sample 1
4. σ_2^2 test variance for sample 2
5. Test value for $\sigma_1^2 = \sigma_2^2$
6. F quantile for $\sigma_1^2 = \sigma_2^2$
7. Test value for slopes $b_1 = b_2$
8. F quantile for slopes $b_1 = b_2$
9. Test value for intercepts $a_1 = a_2$
10. F quantile for intercepts $a_1 = a_2$

Note, that the test for the equality of intercepts is valid only if the slopes are found different.

parm returns a 2×2 matrix containing the two regression coefficients (intercept a and slope b) for each of the two samples

Restrictions:
1. The input data cannot have missing values.
2. Something is not correct with this implementation. I was not able to confirm the results of the text.

Relationships:

Examples: 1. Compare Variances and Slopes, p. 151:

```
data = [ 925  19.06  19.68 ,   950  18.13  19.07 ,
         975  17.19  18.13 ,  1000  16.56  17.50 ,
        1025  15.84  16.57 ,  1050  14.76  15.93 ,
        1075  13.75  15.06 ,  1100  13.03  14.38 ,
        1125  12.19  13.44 ,  1150  11.25  12.81 ,
        1175  10.42  11.88 ,  1200   9.69  11.27 ];
 x = data[,1]; y1 = data[,2]; y2 = data[,3];
 nr = nrow(data);
```

```
x = cons(nr,1,1.) -> x;
optn = 2;
< gof,parm > = hbrcmp(x,y1,x,y2,optn);
print "GOF=",gof;
print "Parm=",parm;
```

```
                ****************************
                Compare two Regression Slopes
                ****************************


                           Sample 1      Sample 2
                Nobs             12            12
           Intercept   50.77888112   48.49589744
               Slope  -0.034311888  -0.031076923
           Error Var   0.011813112   0.006631282


          Test for Equality of Error Variances:
           F=1.78142 <= F(0.95;11,11)=2.81793:
     H: Equal Variances cannot be rejected: p=0.176235

              Test for Equality of Slopes:
            F=50.7097 > F(0.95;1,20)=4.35124:
       H: Equal Slopes is rejected: p=6.70211e-007
```

```
GOF=
                |         1
----------------------------
          Error |   0.00000
        Test N1 |    12.000
        Test N2 |    12.000
      Test Var1 |    0.0118
      Test Var2 |    0.0066
 Test Var1=Var2 |    1.7814
 Qunt Var1=Var2 |    2.8179
 Test Slo1=Slo2 |    50.710
 Qunt Slo1=Slo2 |    4.3512
 Test Int1=Int2 |     .
 Qunt Int1=Int2 |    4.3512
         unused |     .


Parm=
   |         1         2
```

```
-----------------------
 1 |    50.779   -0.0343
 2 |    48.496   -0.0311
```

3.11 Function `hbscheffe`

gof = hbscheffe(data,cont<,freq<,optn>>)

Purpose: Scheffè (1959) test (Nollau, 1975, chapter 4.3, A4.5).

Input: data

cont specifies the coefficients of a contrast vector

freq

optn specifies a vector of options:

1. Controls the printed output. Default is 0, i.e. no printed output.
2. The probablity α, default is $\alpha = 0.05$.

Output: There is only one return argument, the vector `gof`. The entries of the vector have the following meaning:

1. Indicating error return.

Restrictions: 1. The input data cannot have missing values.

2.

Relationships: anova(), hbanova()

Examples: 1. :

3.12 Function `mds`

< gof,conf,... > = mds(data,optn<,xini<,wini <,tra<,ceq<,ref<,pspl<,sew> .. >)

< gof,conf,wgt,eval,evec > = mds(data,optn<,xini,...> .. >)

< gof,conf,grad,res > = mds(data,optn<,xini,...> .. >) KYST

< gof,conf,wgt,res,cov > = mds(data,optn<,xini,...> .. >) MULTISCALE

Purpose: This function implements a variety of *metric* and *nonmetric* methods for 2-way and 3-way multidimensional scaling. Similar to cluster analysis, multidimensional scaling analyzes the relations between the rows of a data set (observations, cases) and is therefore different of most statistical methods which anayze the relations between the columns (variables, items) of a data set.

The result is usually a $n \times d$ configuration matrix $\hat{\mathbf{X}}$ of n points in a small $d << p$ dimensional space, whose interpoint distances $d(x_i, x_j)$ fit in some way the distances among the larger p dimensional space of the input data. If $d = 2$ or $d = 3$, the plot of the $n \times d$ result confugration may uncover the most important relations among the observations of the data, such as clusters of similar subjects.

If the data is a tensor consisting of m data matrices belonging to different subjects or treatments different models for the model distance matrix $\mathbf{D}_i$,

$$X_i = X * W_i \quad , \text{ for } i = 1, \ldots, m$$

can be applied among them:

identity $\mathbf{W}$ is the identity $\mathbf{I}$ matrix. This model can be fitted via KYST and MULTISCALE.

diagonal $\mathbf{W}$ is a $d \times d$ positive diagonal matrix: this model is also known as Horan's model and can be fitted via COSPA, SUMSCAL, INDSCAL, and MULTISCALE.

full (symmetric) model: $\mathbf{W}$ is a $d \times d$ symmetric matrix with a positive diagonal. This model can be fitted by MULTISCALE.

The following estimation methods are implemented:

COSPA 3-way COmmon SPace Analysis by Schoenemann (1972)

INDSCAL 3-way (Metric) INdividual Differences Scaling by Carroll & Chang (1970) using the Canonical Decomposition (CANDECOMP) technique is based on Horan's (1969) model

SUMSCAL 3-way (Metric) individual differences scaling by de Leeuw & Pruzansky (1978)

KYST 2-way Kruskal-Young-Shepard-Torgerson metric and nonmetric scaling; see Young (1987)

MULTISCALE 2- and 3-way maximum likelihood method by Ramsay (1977): this method also computes asymptotic standard errors for the unrotated point coordinates from which ellipsoidal confidence intervals of the point estimates can be drawn

SMACOF 2- and 3-way metric and nonmetric method by de Leeuw (1984, 1994, 2007): these are iterative ALS methods suitable for problems with many observations

Input: data This function permits the following forms of data input:

1. $n \times p$ matrices or $K \times n \times p$ 3-way tables (tensors) of *raw* data.
2. symmetric $n \times n$ matrices or $K \times n \times n$ 3-way tables (tensors) of distance (dissimilarity) data.
3. symmetric $n \times n$ matrices or $K \times n \times n$ 3-way tables (tensors) of correlation, covariance, or scalar product (similarity) data.

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.

xini an $n \times d$ matrix of initial values for the **X** configuration of points

Option	Second Column	Meaning
`"addc"`		add constant if distance data do not satisfy triangle inequality
`"dimmax"`	int	maximum number dimensions d (def=2)
`"dimmin"`	int	minimum number dimensions d (def=2)
`"dimdif"`	int	step size dimensions d (def=1)
`"data"`	string	kind of input data
	"raw"	$n \times p$ raw data
	"diss"	dissimilarities
	"simi"	similarities
	"dist"	distances
	"corr"	correlations
	"cova"	covariances
	"scp"	scalar product data
`"gutlin"`		apply Guttman-Lingoes transformation
`"met"`	string	estimation method
	"torg"	classic Torgerson
	"cos"	COSPA
	"sums"	SUMSCAL
	"inds"	INDSCAL
	"kyst"	KYST
	"mult"	MULTISCALE

Option	Second Column	Meaning
"maxit"	int	number of iterations
"ndim"	int	number dimensions d (def=2)
"noplot"		do not plot
"nopr"		do not print
"pdat"		print input data
"pdis"		print distance matrices
"pcoef"		print X and W configurations
"phis"		print iteration history
"print"	int	controls printed output
"pres"		printed observationwise output
"plotf"	char	plotting into text file (with larger line size)
"plotco"	char	configuration plotting (dimensions pairwise)
	"non"	
	"som"	
	"all"	
"plotsc"	char	scatter plotting (data vs. model)
	"non"	
	"som"	
	"all"	
	"joi"	
"plowgt"	char	subject weights plotting (dimensions pairwise)
	"non"	
	"som"	
	"all"	
"seed"	int	seed of random generator (def: time of day)
"sing"	real	singularity criterion (def=1.e-8)
"xtol"	real	termination criterion (def=1.e-5)

Specific options for the INDSCAL, COSPA, and SUMSCAL methods:

This three methods are specifically designed for the 3-way metric MDS of a set of K scalar product matrices. For input raw data the `mds` function will produce distance or scalar product matrices which are suitable. They only fit 3-way scalar product data in a least squares sense and are therefore rather sensitive to outliers of the data. All other input data types, like (squared) Euclidean distances, are transformed internally to scalar product data.

Option	Second Column	Meaning
"dist"	string	form of distance function
	"euc"	use Euclidean distances (def)
	"squ"	use squared Euclidean distances
"norm"		use normed scalar products
"plin"	int	print only selected steps of INDSCAL iteration

Specific options for the more general KYST methods:

KYST uses more general algorithms and is applicable to a wider range of input data. KYST also permits to fit city-block distances and nonlinear or monotone regression between the data and the model distances. KYST can also deal with a reasonable amount of missing data.

Option	Second Column	Meaning
`"acsav"`	real	for termination test (def=0.66)
`"cutof"`	real	threshold for removing small distances from fit
`"cosav"`	real	for termination test (def=0.66)
`"coord"`	char	type of normalization
	"as-is"	no normalization
	"stand"	normalize at each iteration
	"rotat"	rotate to principal components (def)
`"diag"`	char	do or don't use diagonal of data matrix
	"absent"	do not use diagonal
	"present"	use diagonal (def)
`"dpow"`	real	power of diatance function (def=2.)
		should be in [1., 4.]
`"ftol"`	real	termination test for stress value (def=.01)
`"gtol"`	real	termination test for 2-norm of gradient (def=0)
`"pre-it"`	int	number of iterations for start estimates

Option	Second Column	Meaning
`"poldeg"`	int	polynomial degree of regression
		should be in $[1, 4]$
`"regres"`	char	type of regression between data and model values
	"asc"	isotone ascending (def)
	"des"	isotone descending
	"poly"	polynomial with intercept
	"poln"	polynomial without intercept
`"split"`	char	split data values in groups (for unfolding)
	"byrow"	
	"bygroup"	
	"bydeck"	
`"sfrm"`	int	kind of stress formula for optimization
		=1: default, =2: for unfolding
`"sratst"`	real	for termination test (def=.999)
`"start"`	char	method for initial estimates
	"rand"	random
	"tors"	Torgerson-Young quasi nonmetric
`"ties"`	char	type of tie processing in isotone regression
	"prim"	type 1 processing
	"seco"	type 2 processing
`"unsym"`		input of unsymmetric data matrix
		(does not refer to `"data"` `"raw"` input)

Specific options for the MULTISCALE method:

Option	Second Column	Meaning
"alpha"	real	for CI of ML estimation (def=0.05)
"ase"		does compute standard errors (default for MULTSCAL)
"ccorr"		compute and print correlations of log dissimilarities between matrices
"coord"	char	type of normalization
	"as-is"	no normalization
	"rotat"	rotate to principal components (def)
"dlogn"		error distribution: lognormal
"dnorm0"		error distribution: normal with beta=0
"dnorm5"		error distribution: normal with beta=.5
"dnorm1"		error distribution: normal with beta=1
"itxmax"	int	maximum iterations for X (coordinates) optimization
"itwmax"	int	maximum iterations for W (weights) optimization
"itpmax"	int	maximum iterations for P (spline) optimization
"itamax"	int	maximum iterations for A (stand.error) optimization
"knots"	int	number (inner) knots for spline transform (is internally increased by two more boundary knots)
"mod"	char	subjective weighting model
	"ide"	identity model
	"dia"	(positive) diagonal weights
	"ful"	full $d \times d$ weight matrices
"noase"		does not compute standard errors
"orig"	int	=2: use original version of MULTISCALE program
		=0: use modified version
"pcorr"		print $d \times d$ asymptotic correlation matrices of coordinate estimates
"plis"		print line search information
"ptab"		print some table output
"regres"	char	type of regression between data and model values
	"scale"	
	"power"	power transform
	"spline"	spline transform

Option	Second Column	Meaning
`"tech"`	char	optimization technique (see NLP)
`"vers"`	int	version of algorithm
`"varobs"`	char	stimulus variance level: A fit
	"con"	constant (default)
	"obs"	stimuliwise (fit n add. parameters)
	"pair"	pairwise (set $n*(n+1)/2$ estimates)
`"varsub"`	char	subject variance level
	"con"	constant
	"subw"	subjectwise (default) set m add. estimates

The number of parameters $p = p_1+p_2+p_3$ for the MULTISCALE approach is:

1. `model` specification:

$$p_1 = \begin{cases} n*d - d*(d+1)/2 & \text{identity} \\ n*d + (m-1)*(d-1) - d & \text{diagonal} \\ n*d + (m-1)*d(d+1)/2 - m - d + 1 & \text{full} \end{cases}$$

2. `varobs` specification:

$$p_2 = \begin{cases} 0 & \text{constant} \\ n-1 & \text{stimuliwise} \\ n*n-1)/2 - 1 & \text{pairwise} \end{cases}$$

3. `varsub` specification:

$$p_3 = \begin{cases} 0 & \text{constant} \\ m-1 & \text{subjectwise} \end{cases}$$

Specific options for the SMACOF method:

Option	Second Column	Meaning

Output: Let $s = \sum_{d=dimmin}^{dimmax} d$, i.e. for only one dimensionality $dimmin = dimmax$ there is $s = d$.

GOF a vector with some scalar results

conf this is a $n \times s$ matrix containing all of the computed $n \times d$ configurations horicontally concatenated

wgt the $w \times s$ matrix of the individual weights

eval or [i] for INDSCAL, COSPA, SUMSCAL: this a s vector of eigenvalues depending on the method [ii] for KYST, MULTISCAL, and SMACOF:

evec or grad [i] for INDSCAL, COSPA, SUMSCAL: this is a $n \times s$ matrix of eigenvectors [ii] for KYST, MULTISCAL, and SMACOF: it contains the gardient at the solution

Restrictions: 1.

2.

Relationships: cluster(), dist()

Examples: 1. Randomly generated raw data in tensor form `X[2,n,p]`:

```
srand(123);
nind = [ 2 10 10 ]; m = p = nind[1];
nway = size(nind); n = nind[2]; nd = 4;
none = cons(n,1,1.); pone = cons(p,1,1.);

/* [1] Uniform random generation of pxnd weight matrix: W[p,nd] */
wght = rand(p,nd,"g");
print "Uniform: Unscaled wght=",wght;


/* [2] Normal random generated nxnd configuration: X[n,nd] */
coef  = rand(n,nd,"g","norm",0.,1.);
coef += rand(n,nd,"g","norm",1.,1.);
mu = cons(nd,1,1.); id = ide(nd);
for (j = 1; j <= n; j++) {
  coef[j,] += mrand("mnor",mu,id)‘;
}
print "Normal: Unscaled coef=",coef;
```

```
   /* [3] Normalize data[p+n,nd]: NORCONF() */
   /* [3.1] center X[n,nd] */
   mu = coef[+,] / n;

   coef = coef - none * mu;
   print "Centered Coef=", coef;

   /* [3.2] scaling coef[n,nd] */
   tv = coef[**,]; rv = 1. ./ sqrt(tv);
   coef = coef .* (none * rv);
   print "Scaled Coef=", coef;
   print "Mean=", coef[+,];
   print "Scale=", coef[**,];

   /* [3.3] scaling wght[p,nd] */
   twgt = wght .* (pone * tv);
   print "Scaled WGT=", twgt;


   /* [4] Concatenate COEF and WGHT */
   wcof = twgt |> coef;


   /* [5] Permute dimensions according weights
          sum of squares in descending order */
   wscl = twgt[**,];
   wrnk = wscl[>:]; print "Ranks=", wrnk;
   wcof = wcof[,wrnk];
   n1 = p+1; n2 = p+n;
   wght = wcof[1:p,]; coef = wcof[n1:n2,];


   /* [6] Weighted Configuration: Y = X * W */
   real wxdat[p,n,nd];
   for (j = 1; j <= p; j++) {
     wv = wght[j,];
     wxdat[j,,] = coef .* (none * wv);
     /* print "WXdat[j,,]=", wxdat[j,,]; */
   }
   print "Tensor WXDAT=",wxdat;


Tensor WXDAT=
                    ******************************
                    Tensor wxdat with 3 Dimensions
                    ******************************
```

```
                    wxdat_1
                    *******

              Dense Matrix (10 by 4)

   |          1         2         3         4
---------------------------------------------------
 1 |  -14.554672 -1.6301023  4.9856908  0.0606599
 2 |  -3.0380098 -4.5080465  1.0150659  0.0439437
 3 |   6.4290345 -7.0272730  0.4288613 -0.6636031
 4 |  -8.4827305  6.9577941 -1.4993646  0.1756061
 5 |   18.784123  1.9255437  3.3502585  0.0904182
 6 |   13.203212  0.4243445 -3.0888729 -0.5465808
 7 |   1.3760075  3.8225599  1.3278726  0.2144417
 8 |  -6.3482351  1.5903091  2.2758083  0.0053984
 9 |  -4.2046344 -5.6754399 -5.2610451  0.3112615
10 |  -3.1640951  4.1203103 -3.5342749  0.3084545

                    wxdat_2
                    *******

              Dense Matrix (10 by 4)

   |          1         2         3         4
---------------------------------------------------
 1 |  -6.3948545 -0.7757735  3.4352285  0.2674217
 2 |  -1.3348037 -2.1454008  0.6993982  0.1937275
 3 |   2.8247109 -3.3443127  0.2954930 -2.9255223
 4 |  -3.7270388  3.3112474 -1.0330885  0.7741671
 5 |   8.2531390  0.9163754  2.3083869  0.3986123
 6 |   5.8010665  0.2019476 -2.1282877 -2.4096247
 7 |   0.6045734  1.8191745  0.9149276  0.9453753
 8 |  -2.7892102  0.7568357  1.5680719  0.0237990
 9 |  -1.8473810 -2.7009689 -3.6249525  1.3722094
10 |  -1.3902016  1.9608753 -2.4351774  1.3598348
```

(a) Analysis with COSPA:

```
optn = [ "data"      "raw" ,
         "meth"    "cospa" ,
         "ndim"          2 ,
         "norm"           ];
< gof,conf,wgt,eval,evec > = mds(wxdat,optn);
print "GOF=",gof;
print "X Configuration=",conf;
```

```
print "W Configuration=",wgt;
print "Eigenvalues=",eval;
print "Eigenvectors=",evec;

                ***************************
                3-Way Multidimension Scaling
                ***************************

                        Raw Input Data
 Number Rows of Matrix . . . . . .          10
 Dimension . . . . . .                       2
 Amount Input Information. . . . . .        90
 Number of Parameters. . . . . .            24
 Degrees of Freedom. . . . . .              66

                ************************
                Compute SCHOENEMANN COSPA
                ************************

Eigenvalues of Mean Scalar Product Matrix
-----------------------------------------

  1 :      0.9573     0.2185     0.1664    0.02963  1.191e-016
  6 :   1.633e-017 -1.537e-017 -4.755e-017 -1.052e-016 -3.128e-016

             Unrotated common space configuration

                    DIM_1        DIM_2
               O_01 -0.465592568  0.071152585
               O_02 -0.093331284  0.147963476
               O_03  0.227242154  0.246678538
               O_04 -0.278491720 -0.215828327
               O_05  0.585362887 -0.105058979
               O_06  0.434267720 -0.008289568
               O_07  0.032286775 -0.137844258
               O_08 -0.205208492 -0.042378202
               O_09 -0.128315170  0.181828134
               O_10 -0.108220302 -0.138223400

              Undiagonalized Weight Matrices:
              ******************************

                   Weight matrix:    1

                           DIM_1        DIM_2
```

```
DIM_1   1.017745116
DIM_2  -0.027635222  0.907661656

      Weight matrix:    2

              DIM_1        DIM_2
DIM_1   0.982254884
DIM_2   0.027635222  1.092338344

   Orthogonal Rotation Matrix

              DIM_1        DIM_2
DIM_1   0.230560993  0.973057875
DIM_2   0.973057875 -0.230560993

Rotated common space configuration

             DIM_1        DIM_2
O_01  -0.038111902 -0.469453525
O_02   0.122458472 -0.124931347
O_03   0.292425671  0.164245319
O_04  -0.274222781 -0.221226968
O_05   0.032733382  0.593814469
O_06   0.092058967  0.424478876
O_07  -0.126686370  0.063198410
O_08  -0.088549516 -0.189908979
O_09   0.147344825 -0.166780562
O_10  -0.159450748 -0.073435693

  Diagonalized Weight Matrices:
  ****************************

      Weight matrix:    1

              DIM_1        DIM_2
DIM_1   0.901113634
DIM_2  -3.8580e-015  1.024293137

      Weight matrix:    2

              DIM_1        DIM_2
DIM_1   1.098886366
DIM_2   2.1927e-015  0.975706863

Schoenemann's Testindizes for COSPA
```

```
Matrix  CommonSpace  Diagonality
     1  0.964379639  2.8395e-015
     2  0.885697342  1.4974e-015

*****************************
Result Normalized Configuration
*****************************

   Common Space Configuration

             DIM_1        DIM_2
O_01  -0.489977791 -0.075071833
O_02  -0.130393281  0.241215514
O_03   0.171426039  0.576012484
O_04  -0.230898897 -0.540156903
O_05   0.619775732  0.064477364
O_06   0.443036873  0.181335360
O_07   0.065961412 -0.249543518
O_08  -0.198211702 -0.174422535
O_09  -0.174072122  0.290236005
O_10  -0.076646263 -0.314081938

   Subjective Weight Matrices

             DIM_1        DIM_2
 W_1   0.940278879  0.232245207
 W_2   0.895677733  0.283217434

  Goodness-of-fit resp. SCP Matrices
 *********************************

      N  Av_SSQ_Diff  Correlation
      1  0.000125092  0.994679763
      2  0.000660168  0.972705057

Goodness-of-fit resp. Distance Matrices
***************************************

      N  Av_SSQ_Diff  Correlation
      1  2.033206077  0.990027529
      2  1.716943364  0.965981685

GOF=
          |         1
```

```
--------------------
    Error |   0.00000
     Time |   0.00000
 Ind.Inf. |    90.000
   NParms |    24.000
       DF |    66.000
   unused |     .
   unused |     .
   unused |     .
   unused |     .
   unused |     .
```

X Configuration=

	DIM_1	DIM_2
O_01	-0.48998	-0.07507
O_02	-0.13039	0.24122
O_03	0.17143	0.57601
O_04	-0.23090	-0.54016
O_05	0.61978	0.06448
O_06	0.44304	0.18134
O_07	0.06596	-0.24954
O_08	-0.19821	-0.17442
O_09	-0.17407	0.29024
O_10	-0.07665	-0.31408

W Configuration=

	DIM_1	DIM_2
W_1	0.94028	0.23225
W_2	0.89568	0.28322

Eigenvalues=

	DIM_1	DIM_2
1	0.95725	0.21846

Eigenvectors=

	DIM_1	DIM_2
O_01	-0.47588	0.15223
O_02	-0.09539	0.31657
O_03	0.23226	0.52777
O_04	-0.28464	-0.46177
O_05	0.59829	-0.22478

```
O_06 |    0.44386   -0.01774
O_07 |    0.03300   -0.29492
O_08 |   -0.20974   -0.09067
O_09 |   -0.13115    0.38902
O_10 |   -0.11061   -0.29573
```

(b) Analysis with SUMSCAL:
We run the same analysis with the SUMSCAL method, but now for two and also one dimension:

```
optn = [ "data"      "raw" ,
         "meth"    "sumsc" ,
         "dimmax"         2 ,
         "dimmin"         1 ,
         "norm"           ];
< gof,conf,wgt,eval,evec > = mds(wxdat,optn);

                ****************************
                3-Way Multidimension Scaling
                ****************************

                        Raw Input Data
 Number Rows of Matrix . . . . . .          10
 Dimension . . . . . .           2
 Amount Input Information. . . . . .        90
 Number of Parameters. . . . . .            24
 Degrees of Freedom. . . . . .              66

              *********************************
              Compute DeLEEUW-PRUZANSKY SUMSCAL
              *********************************

Eigenvalues of Mean Scalar Product Matrix
-----------------------------------------

  1 :      0.9573     0.2185     0.1664    0.02963  1.191e-016
  6 :   1.633e-017 -1.537e-017 -4.755e-017 -1.052e-016 -3.128e-016

             Unrotated common space configuration

                    D_2_1        D_2_2
              O_01  -0.465592568  0.071152585
              O_02  -0.093331284  0.147963476
              O_03   0.227242154  0.246678538
              O_04  -0.278491720 -0.215828327
              O_05   0.585362887 -0.105058979
```

```
O_06   0.434267720 -0.008289568
O_07   0.032286775 -0.137844258
O_08  -0.205208492 -0.042378202
O_09  -0.128315170  0.181828134
O_10  -0.108220302 -0.138223400

Undiagonalized Weight Matrices:
*******************************

      Weight matrix:    1

             D_2_1        D_2_2
D_2_1   1.017745116
D_2_2  -0.027635222  0.907661656


      Weight matrix:    2

             D_2_1        D_2_2
D_2_1   0.982254884
D_2_2   0.027635222  1.092338344

 Iterative Simultaneous Diagonalization:
 ***************************************


Iter  ssq-off-dia     max(ang)     min(cos)
   1  0.055270443  1.3408e+154  0.000000000
   2  4.3715e-016  0.232654169  0.973057875
   3  4.3715e-016  2.8166e-017  1.000000000

Residuals SIMDIAG: nit = 2
--------------------------

  1 :   4.627e-032  4.93e-032

          Orthogonal Rotation Matrix

             D_2_1        D_2_2
D_2_1   0.973057875 -0.230560993
D_2_2   0.230560993  0.973057875

        Rotated common space configuration

             D_2_1        D_2_2
```

```
O_01  -0.469453525 -0.038111902
O_02  -0.124931347  0.122458472
O_03   0.164245319  0.292425671
O_04  -0.221226968 -0.274222781
O_05   0.593814469  0.032733382
O_06   0.424478876  0.092058967
O_07   0.063198410 -0.126686370
O_08  -0.189908979 -0.088549516
O_09  -0.166780562  0.147344825
O_10  -0.073435693 -0.159450748

 Diagonalized Weight Matrices:
 ****************************

      Weight matrix:    1

              D_2_1        D_2_2
D_2_1   1.024293137
D_2_2  -2.1511e-016  0.901113634

      Weight matrix:    2

              D_2_1        D_2_2
D_2_1   0.975706863
D_2_2  -2.2204e-016  1.098886366

Schoenemann's Testindizes for SUMSCAL

     1  0.964379639  1.5832e-016
     2  0.885697342  1.5163e-016

  ******************************
  Result Normalized Configuration
  ******************************

    Common Space Configuration

             D_2_1        D_2_2
O_01  -0.489977791 -0.075071833
O_02  -0.130393281  0.241215514
O_03   0.171426039  0.576012484
O_04  -0.230898897 -0.540156903
O_05   0.619775732  0.064477364
O_06   0.443036873  0.181335360
O_07   0.065961412 -0.249543518
```

```
O_08  -0.198211702 -0.174422535
O_09  -0.174072122  0.290236005
O_10  -0.076646263 -0.314081938

   Subjective Weight Matrices

            D_2_1        D_2_2
W_1   0.940278879  0.232245207
W_2   0.895677733  0.283217434


 Goodness-of-fit resp. SCP Matrices
 **********************************

      N  Av_SSQ_Diff  Correlation
      1  0.000125092  0.994679763
      2  0.000660168  0.972705057

Goodness-of-fit resp. Distance Matrices
***************************************

      N  Av_SSQ_Diff  Correlation
      1  2.033206077  0.990027529
      2  1.716943364  0.965981685
```

It follows the analysis in one dimension:

```
*************************************************
Compute DeLeeuw-Pruzansky SUMSCAL for 1 Dimension(s)
*************************************************

    Unrotated common space configuration

              D_1_1
O_01  -0.465592568
O_02  -0.093331284
O_03   0.227242154
O_04  -0.278491720
O_05   0.585362887
O_06   0.434267720
O_07   0.032286775
O_08  -0.205208492
O_09  -0.128315170
O_10  -0.108220302

  Undiagonalized Weight Matrices:
```

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

      Weight matrix:    1

                   D_1_1
      D_1_1   1.017745116

      Weight matrix:    2

                   D_1_1
      D_1_1   0.982254884

Schoenemann's Testindizes for SUMSCAL

      1  0.908403857  0.000000000
      2  0.816123639  0.000000000

*******************************
Result Normalized Configuration
*******************************

   Common Space Configuration

                   D_1_1
      O_01  -0.475875172
      O_02  -0.095392504
      O_03   0.232260793
      O_04  -0.284642206
      O_05   0.598290617
      O_06   0.443858516
      O_07   0.032999828
      O_08  -0.209740518
      O_09  -0.131149009
      O_10  -0.110610345

   Subjective Weight Matrices

                 D_1_1
      W_1   0.974237878
      W_2   0.940264806

Goodness-of-fit resp. SCP Matrices
*********************************

      N  Av_SSQ_Diff  Correlation
```

```
                    1  0.000574718  0.975346430
                    2  0.001385490  0.941856456

               Goodness-of-fit resp. Distance Matrices
               ***************************************

                    N  Av_SSQ_Diff  Correlation
                    1  0.000000000  1.000000000
                    2  0.000000000  1.000000000
```

(c) Analysis with INDSCAL:

```
optn = [ "data"      "raw" ,
         "meth"    "indsc" ,
         "ndim"          2 ,
         "norm"           ];
< gof,conf,wgt,eval,evec > = mds(wxdat,optn);
```

The following output of the iteration history could be restricted to larger steps using the `plin` option (default is 10):

```
                Iterative Canonical Decomposition:
                **********************************

          Iter  Way    Max. Dev.     Maximum     Residual
             1    1  0.092338344
             1    2  0.002573734  0.092338344  0.275733651
            11    1  0.000126273
            11    2  0.001420932  0.001420932  0.275619458
            21    1  9.1720e-005
            21    2  0.001341886  0.001341886  0.275430703
            31    1  5.9351e-005
            31    2  0.001262113  0.001262113  0.275255123
            41    1  3.3093e-005
            41    2  0.001178954  0.001178954  0.275097042
             .....................................
           631    1  3.3370e-006
           631    2  1.2614e-005  1.2614e-005  0.274047651
           641    1  3.1349e-006
           641    2  1.1837e-005  1.1837e-005  0.274047522
           651    1  2.9452e-006
           651    2  1.1108e-005  1.1108e-005  0.274047402
           661    1  2.7671e-006
           661    2  1.0426e-005  1.0426e-005  0.274047292
           664    0  9.9616e-006  9.9616e-006  0.274047235

 Residuals CADECO: nit = 664
```

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

  1 :    0.006999    0.03625

            Result non-normalized configuration X:
            ***************************************

                 Common Space Configuration

                        DIM_1        DIM_2
              O_01  -0.469705827 -0.036553819
              O_02  -0.124329431  0.121437948
              O_03   0.164347235  0.294394706
              O_04  -0.222344182 -0.271971552
              O_05   0.594691055  0.028653940
              O_06   0.423981427  0.094048475
              O_07   0.063142821 -0.127689246
              O_08  -0.190385558 -0.087104099
              O_09  -0.165710781  0.144643058
              O_10  -0.073686760 -0.159859411

                 Common Space Configuration

                        DIM_1        DIM_2
              O_01  -0.469705827 -0.036553819
              O_02  -0.124329431  0.121437948
              O_03   0.164347235  0.294394706
              O_04  -0.222344182 -0.271971552
              O_05   0.594691055  0.028653940
              O_06   0.423981427  0.094048475
              O_07   0.063142821 -0.127689246
              O_08  -0.190385558 -0.087104099
              O_09  -0.165710781  0.144643058
              O_10  -0.073686760 -0.159859411

                 Subjective Weight Matrices

                        DIM_1        DIM_2
               W_1   1.023979211  0.899153581
               W_2   0.975339995  1.102091165

            Result Normalized Configuration
            *******************************

                 Common Space Configuration
```

```
             DIM_1        DIM_2
O_01  -0.489950010 -0.072139585
O_02  -0.129687993  0.239659861
O_03   0.171430553  0.580992970
O_04  -0.231927151 -0.536740493
O_05   0.620322065  0.056549038
O_06   0.442254902  0.185606268
O_07   0.065864258 -0.251996903
O_08  -0.198591120 -0.171901425
O_09  -0.172852867  0.285455541
O_10  -0.076862638 -0.315485272
```

Common Space Configuration

```
             DIM_1        DIM_2
O_01  -0.489950010 -0.072139585
O_02  -0.129687993  0.239659861
O_03   0.171430553  0.580992970
O_04  -0.231927151 -0.536740493
O_05   0.620322065  0.056549038
O_06   0.442254902  0.185606268
O_07   0.065864258 -0.251996903
O_08  -0.198591120 -0.171901425
O_09  -0.172852867  0.285455541
O_10  -0.076862638 -0.315485272
```

Subjective Weight Matrices

```
             DIM_1        DIM_2
W_1   0.941108062  0.230861820
W_2   0.896405242  0.282966978
```

Goodness-of-fit resp. SCP Matrices

****

```
      N  Av_SSQ_Diff  Correlation
      1  0.000127246  0.994584218
      2  0.000659062  0.972744353
```

Goodness-of-fit resp. Distance Matrices

****

```
      N  Av_SSQ_Diff  Correlation
      1  2.075138105  0.989819063
```

```
                              2   1.718323923   0.965938432
```

2. Scalar product and distance data in tensor form `D[2,n,n]`:

   The first part of randomly generating the `coef` and `wgt` data is the same as for the raw data tensor `wxdat` above.

   The tensor data `eudis` of distances are then obtained using the `dist` function:

```
/* [6] Weighted Euclidean Distances: WEUCDIS() */
real eudis[p,n,n];
for (j = 1; j <= p; j++) {
  wv = wght[j,];
  xcof = coef .* (none * wv);
  optn = [ 0, 0 ];
  eudis[j,,] = dist(xcof,"L2",optn);
  /* print "Eudis[j,,]=", eudis[j,,]; */
}
print "Tensor EUDIS=",eudis;
```

```
Tensor EUDIS=
                  ******************************
                  Tensor eudis with 3 Dimensions
                  ******************************


                            eudis_1
                            *******

                Dense Symmetric Matrix (10 by 10)

    S |          1          2          3          4          5
    ---------------------------------------------------------------
    1 |   0.0000000
    2 |   12.517277  0.0000000
    3 |   22.152531  9.8394964  0.0000000
    4 |   12.356763  12.940254  20.551525  0.0000000
    5 |   33.567743  22.870327  15.553271  28.148374  0.0000000
    6 |   28.987725  17.472745  10.667907  22.715945  8.6757329
    7 |   17.231406  9.4344830  12.034581  10.724700  17.628005
    8 |   9.2229700  7.0525519  15.536413  6.9026711  25.157692
    9 |   15.117752  6.4949865  12.174883  13.858909  25.699440
   10 |   15.344827  9.7586323  15.262637  6.3637807  23.108129


    S |          6          7          8          9         10
```

```
 ---------------------------------------------------------
 6 |   0.0000000
 7 |   13.096467  0.0000000
 8 |   20.315094  8.0987128  0.0000000
 9 |   18.592868  12.836619  10.690362  0.0000000
10 |   16.807093  6.6596174  7.0984902  10.001059  0.0000000


                        eudis_2
                        *******

           Dense Symmetric Matrix (10 by 10)

 S |          1          2          3          4          5
 ---------------------------------------------------------
 1 |   0.0000000
 2 |   5.9135599  0.0000000
 3 |   10.566485  5.3508700  0.0000000
 4 |   6.6365373  6.2318712  10.132854  0.0000000
 5 |   14.788985  10.194800  7.9197796  12.671486  0.0000000
 6 |   13.704587  8.4382427  5.2512397  10.573009  5.8389505
 7 |   7.9080518  4.4822392  6.8526023  4.9812874  7.8457810
 8 |   4.3468595  3.3647800  7.6585561  3.8384410  11.074632
 9 |   8.6863458  4.5453515  7.4888400  6.8377826  12.298703
10 |   8.2577739  5.2962197  8.4694364  3.0972848  10.840219


 S |          6          7          8          9         10
 ---------------------------------------------------------
 6 |   0.0000000
 7 |   7.0806884  0.0000000
 8 |   9.6791208  3.7312506  0.0000000
 9 |   9.1360795  6.8728797  6.4520669  0.0000000
10 |   8.3133128  3.9235426  4.6062928  4.8329621  0.0000000
```

All the results obtained with COSPA, SUMSCAL, and INDSCAL are the same as those obtained with the raw data input. Here we only show the `mds` call for COSPA:

```
optn = [ "data"      "dis" ,
         "meth"    "cospa" ,
         "nobs"         10 ,
         "ndim"          2 ,
         "norm"           ];
< gof,conf,wgt,eval,evec > = mds(eudis,optn);
print "GOF=",gof;
```

```
print "X Configuration=",conf;
print "W Configuration=",wgt;
print "Eigenvalues=",eval;
print "Eigenvectors=",evec;
```

The tensor data `scalp` of scalar products are then obtained using the simple code:

```
/* [6] Weighted Configuration: Y = X * W */
real scalp[p,n,n];
for (j = 1; j <= p; j++) {
  wv = wght[j,];
  xdat = coef .* (none * wv);
  /* print "Xdat=", xdat; */
  scalp[j,,] = xdat * xdat';
}
print "Tensor SCALP=",scalp;
```

```
Tensor SCALP=
                  ******************************
                  Tensor scalp with 3 Dimensions
                  ******************************


                            scalp_1
                            *******

                Dense Symmetric Matrix (10 by 10)

      S |           1          2          3          4          5
      ---------------------------------------------------------------
      1 |   239.35651
      2 |   56.629284  30.584276
      3 |  -80.019400  12.553964  91.339342
      4 |   104.65673 -7.1096775 -104.18964  122.64655
      5 |  -259.82674 -62.342084  108.60925 -150.95050  367.78340
      6 |  -208.29347 -45.183882  80.939938 -104.51143  238.42991
      7 |  -19.625132 -20.055300 -17.588603  12.970976  37.675707
      8 |   101.15092  14.427146 -51.016135  61.504092 -108.55878
      9 |   44.237560  33.032238  10.388312  4.1211227 -107.50639
     10 |   21.733749 -12.535965 -51.017027  60.861770 -63.313758


      S |           6          7          8          9         10
      ---------------------------------------------------------------
```

```
 6 |   184.34477
 7 |   15.570962  18.314592
 8 |  -90.174890  0.3669736  48.008505
 9 |  -41.842453 -34.399568  5.6948546  77.665048
10 |  -29.279458  6.7693928  18.597319  8.6092796  39.574698


                        scalp_2
                        *******

            Dense Symmetric Matrix (10 by 10)

 S |           1          2          3          4          5
---------------------------------------------------------------
 1 |   53.368298
 2 |   12.654620  6.9111339
 3 |  -15.236448  3.0443698  27.809416
 4 |   17.923227 -2.7016504 -24.171767  26.521784
 5 |  -45.452089 -11.290610  19.764048 -29.801599  74.441589
 6 |  -45.209182 -10.131875  22.131477 -20.618844  42.188651
 7 |  -1.8816273 -3.8868014 -6.8715051  3.5571455  9.1455204
 8 |   22.642510  3.2006496 -10.016078  11.300032 -18.696990
 9 |   1.8234916  5.9911013 -1.2710098  2.7489006 -25.542608
10 |  -0.6327998 -3.7909387 -15.182503  15.242772 -14.755914


 S |           6          7          8          9         10
---------------------------------------------------------------
 6 |   44.029056
 7 |  -0.3506802  5.4057317
 8 |  -19.422208  1.1477050  10.811910
 9 |  -6.8538026 -8.0497273 -2.5429844  25.731289
10 |  -5.7625912  1.7842387  1.5754542  7.9653492  13.556932
```

Running the following call of `mds` will generate the same results as reported above for other data input:

```
optn = [ "data"      "scp" ,
         "meth"    "cospa" ,
         "nobs"         10 ,
         "ndim"          2 ,
         "norm"           ];
< gof,conf,wgt,eval,evec > = mds(scalp,optn);
```

3. Test Example from the KYST Manual: Unsymmetric Matrix of Similarity Measures: Regression: Descending

First we use descending isotone (monotone) regression for the relationship among data and model values:

```
simi = [ 9.30  0.57  6.10  2.00  1.50 ,
         1.00  8.50  5.50  3.30  4.00 ,
         2.10  5.30  7.30  3.90  1.10 ,
         5.00  6.20  1.60  8.90  2.80 ,
         1.30  4.10  1.00  2.50  8.80 ];


print "KYST MDS without initial values: Descending";
optn = [ "data"       "simi" ,
         "unsym"             ,
         "meth"       "kyst" ,
         "regres"     "desc" ,
         "pre-it"          3 ,
         "coord"     "rotat" ,
         "dimmax"          3 ,
         "dimmin"          1 ,
         "phis"              ,
         "norm"             ];
< gof,conf > = mds(simi,optn);
```

```
              *********************************
              NonMetric Multidimensional Scaling
              *********************************

             Similarity Input Data (Unsymmetric)
 Number Rows of Matrix . . . . . .          5
 Maximum Dimension . . . . . .              3
 Minimum Dimension . . . . . .              1
 Amount Input Information. . . . . .       25
 Number of Parameters. . . . . .           15
 Degrees of Freedom. . . . . .             10
 Stress Formula. . . . . .                  1
 Diagonal  . . . . . .                 Included
 Isotone Regression  . . . . . .     Descending
 Tie Treatment . . . . . .              Primary
 Maximum Number Iterations . . . . . .     5000
 TORSCA Preiterations. . . . . .            3


Similarity Matrix: 1
--------------------

  1|        9.3       0.57        6.1          2        1.5
```

```
 2|          1        8.5        5.5        3.3          4
 3|        2.1        5.3        7.3        3.9        1.1
 4|          5        6.2        1.6        8.9        2.8
 5|        1.3        4.1          1        2.5        8.8

          ******************************************
          Compute Kruskal-Young-Shepard-Torgerson KYST
          ******************************************
         There are 25 data values, split into 1 lists.

Torgerson-Young Quasi-Nonmetric Procedure for Initial Configuration

                     Iteration      Stress

                             0  0.022638560
                             1  0.002207875
                             2  0.000459813
                             3  0.000153818
 [note] file tmds1.inp, line 63: Maximum number of pre-iterations 3
           reached.
The best initial configuration by Torgerson-Young quasi-nonmetric
method has Stress=0.000153818 Formula 1

             History of Iteration for 3 Dimensions
             *************************************

   Iter STRESS   SRAT SRATAV CAGRCL  COSAV  ACSAV    SFGR    STEP

      0  0.010  0.800  0.800  0.000  0.000  0.000  0.0010  0.0002
        Satisfactory Stress was reached after 0 iterations.


       Final Configuration (Stress= 0.00989601 Formula 1)

                   D_3_1        D_3_2        D_3_3
          O_1   1.075123487 -0.513110454 -0.160950519
          O_2  -0.696452481  0.652971078 -0.001235367
          O_3   0.568985138  0.719528505 -0.156694064
          O_4   0.008885202 -0.233080220  0.586641998
          O_5  -0.956541346 -0.626308909 -0.267762048


              Group Count      Stress Regression

                  1    25  0.00989601 Descending
```

History of Iteration for 2 Dimensions

Iter	STRESS	SRAT	SRATAV	CAGRCL	COSAV	ACSAV	SFGR	STEP
0	0.059	0.800	0.800	0.000	0.000	0.000	0.0039	0.0057
1	0.057	0.968	0.852	0.999	0.659	0.659	0.0037	0.0155
2	0.052	0.916	0.873	0.988	0.876	0.876	0.0031	0.0530
3	0.041	0.785	0.843	0.357	0.533	0.533	0.0014	0.1104
	0.060			-0.658				0.0110
4	0.039	0.966	0.882	0.757	0.681	0.681	0.0010	0.0291
5	0.037	0.948	0.903	0.216	0.374	0.374	0.0005	0.0476
	0.051			-0.940				0.0048
6	0.037	0.996	0.933	-0.490	-0.196	0.451	0.0002	0.0027
7	0.037	0.999	0.955	-0.283	-0.253	0.340	0.0002	0.0016
8	0.037	0.999	0.969	0.007	-0.081	0.120	0.0001	0.0012
9	0.037	1.000	0.979	0.219	0.117	0.185	0.0001	0.0011
10	0.037	1.000	0.986	-0.272	-0.140	0.242	0.0001	0.0007
11	0.037	1.000	0.991	-0.381	-0.299	0.334	0.0000	0.0003
12	0.037	1.000	0.994	0.111	-0.028	0.187	0.0000	0.0002
13	0.037	1.000	0.996	0.415	0.264	0.337	0.0000	0.0003
14	0.037	1.000	0.997	0.008	0.095	0.120	0.0000	0.0002
15	0.037	1.000	0.998	-0.704	-0.432	0.505	0.0000	0.0001
16	0.037	1.000	0.999	0.850	0.414	0.733	0.0000	0.0001
17	0.037	1.000	0.999	-0.496	-0.187	0.577	0.0000	0.0000

Minimum was achieved after 17 iterations.

Final Configuration (Stress= 0.0371116 Formula 1)

	D_2_1	D_2_2
O_1	1.147921385	-0.496658734
O_2	-0.646499738	0.672365556
O_3	0.649763151	0.672655794
O_4	-0.001631430	-0.354985561
O_5	-1.149553368	-0.493377055

Group	Count	Stress	Regression
1	25	0.03711159	Descending

History of Iteration for 1 Dimensions

Iter	STRESS	SRAT	SRATAV	CAGRCL	COSAV	ACSAV	SFGR	STEP

```
 0  0.212  0.800  0.800  0.000  0.000  0.000  0.0109  0.0575
 1  0.204  0.963  0.851  0.850  0.561  0.561  0.0006  0.1358
 2  0.231  1.134  0.937 -0.946 -0.434  0.815  0.0187  0.0501
 3  0.215  0.928  0.934  0.989  0.505  0.930  0.0110  0.0639
 4  0.205  0.956  0.941  0.501  0.502  0.647  0.0029  0.1008
    0.213               -0.979                        0.0101
 5  0.205  0.997  0.959  0.997  0.829  0.878  0.0022  0.0267
 6  0.204  0.997  0.972 -0.903 -0.314  0.895  0.0008  0.0094
    0.204               -0.977                        0.0009
 7  0.204  1.000  0.981  1.000  0.553  0.964  0.0006  0.0012
 8  0.204  1.000  0.987  0.999  0.847  0.987  0.0004  0.0028
 9  0.204  1.000  0.991 -0.649 -0.140  0.764  0.0001  0.0012
    0.204               -1.000                        0.0001
10  0.204  1.000  0.994  1.000  0.612  0.920  0.0000  0.0002
11  0.204  1.000  0.996  1.000  0.868  0.973  0.0000  0.0004
    0.204               -1.000                        0.0000
12  0.204  1.000  0.997  1.000  0.955  0.991  0.0000  0.0001
    0.204               -1.000                        0.0000
13  0.204  1.000  0.998  1.000  0.985  0.997  0.0000  0.0000
    0.204               -0.990                        0.0000
14  0.204  1.000  0.999  1.000  0.995  0.999  0.0000  0.0000
15  0.204  1.000  0.999  1.000  0.998  1.000  0.0000  0.0000
       Minimum was achieved after 15 iterations.


    Final Configuration (Stress= 0.203946 Formula 1)

                          D_1_1
                O_1   1.394282445
                O_2  -0.663840798
                O_3   0.746266448
                O_4  -0.042636844
                O_5  -1.434071249


          Group Count      Stress Regression

              1    25  0.20394609 Descending
```

4. Test Example from the KYST Manual: Unsymmetric Matrix of Similarity Measures: Regression: Third Order Polynomial

```
simi = [ 9.30  0.57  6.10  2.00  1.50 ,
         1.00  8.50  5.50  3.30  4.00 ,
         2.10  5.30  7.30  3.90  1.10 ,
```

```
         5.00  6.20  1.60  8.90  2.80 ,
         1.30  4.10  1.00  2.50  8.80 ];

print "KYST MDS without initial values: Polynomial Degree=3";
optn = [ "data"       "simi" ,
         "unsym"             ,
         "meth"       "kyst" ,
         "regres"     "poly" ,
         "poldeg"          3 ,
         "pre-it"          3 ,
         "coord"     "rotat" ,
         "dimmax"          3 ,
         "dimmin"          1 ,
         "phis"              ,
         "norm"             ];
< gof,conf > = mds(simi,optn);
```

```
              ******************************
              Metric Multidimensional Scaling
              ******************************

            Similarity Input Data (Unsymmetric)
 Number Rows of Matrix . . . . . .          5
 Maximum Dimension . . . . . .              3
 Minimum Dimension . . . . . .              1
 Amount Input Information. . . . . .       25
 Number of Parameters. . . . . .           15
 Degrees of Freedom. . . . . .             10
 Stress Formula. . . . . .                  1
 Diagonal  . . . . . .                Included
 Polynomial Regression Degree. . . . .      3
 Maximum Number Iterations . . . . . .   5000
 TORSCA Preiterations. . . . . .            3

         *******************************************
         Compute Kruskal-Young-Shepard-Torgerson KYST
         *******************************************
        There are 25 data values, split into 1 lists.

Torgerson-Young Quasi-Nonmetric Procedure for Initial Configuration

                    Iteration      Stress

                            0  0.022638560
```

```
                                1  0.002207875
                                2  0.000459813
                                3  0.000153818
 [note] file tmds1.inp, line 42: Maximum number of pre-iterations 3
           reached.
The best initial configuration by Torgerson-Young quasi-nonmetric
method has Stress=0.000153818 Formula 1


              History of Iteration for 3 Dimensions
              *************************************

   Iter STRESS   SRAT SRATAV CAGRCL  COSAV  ACSAV    SFGR    STEP

      0  0.169  0.800  0.800  0.000  0.000  0.000  0.0069  0.0293
      1  0.163  0.968  0.852  0.974  0.643  0.643  0.0056  0.0776
      2  0.153  0.939  0.880  0.738  0.706  0.706  0.0034  0.2093
      3  0.155  1.008  0.921 -0.703 -0.224  0.704  0.0057  0.0997
      4  0.147  0.948  0.930 -0.463 -0.382  0.545  0.0022  0.0464
      5  0.146  0.998  0.952 -0.837 -0.682  0.738  0.0021  0.0153
      6  0.146  0.995  0.966  0.934  0.384  0.867  0.0006  0.0152
      7  0.146  1.000  0.977 -0.897 -0.462  0.887  0.0008  0.0048
      8  0.146  0.999  0.985  0.983  0.492  0.950  0.0003  0.0054
      9  0.146  1.000  0.990 -0.819 -0.374  0.864  0.0002  0.0018
     10  0.146  1.000  0.993  0.401  0.138  0.559  0.0000  0.0012
     11  0.146  1.000  0.995  0.574  0.426  0.569  0.0000  0.0016
     12  0.146  1.000  0.997 -0.448 -0.151  0.489  0.0001  0.0008
     13  0.146  1.000  0.998 -0.553 -0.416  0.531  0.0000  0.0003
     14  0.146  1.000  0.999 -0.631 -0.558  0.597  0.0000  0.0001
     15  0.146  1.000  0.999  0.901  0.405  0.798  0.0000  0.0001
           Minimum was achieved after 15 iterations.


         Final Configuration (Stress= 0.145597 Formula 1)

                    D_3_1        D_3_2        D_3_3
          O_1   0.966987925 -0.512572882 -0.077858169
          O_2  -0.609811159  0.628542479  0.045316107
          O_3   0.480826435  0.706684142 -0.399330276
          O_4  -0.075197068 -0.128523217  0.908024094
          O_5  -0.762806134 -0.694130522 -0.476151756
```

The regression coefficients start with the intercept and it is of no surprise that the coefficient of the order 3 term is negative indicating an descending behavior:

```
          Group Count      Stress Regression Coefficients
```

```
     1    25  0.14559717   2.003  -0.206   0.030  -0.004
```

History of Iteration for 2 Dimensions

****

Iter	STRESS	SRAT	SRATAV	CAGRCL	COSAV	ACSAV	SFGR	STEP
0	0.209	0.800	0.800	0.000	0.000	0.000	0.0054	0.0281
1	0.206	0.986	0.858	0.929	0.613	0.613	0.0036	0.0714
2	0.203	0.983	0.898	0.149	0.307	0.307	0.0028	0.1094
3	0.205	1.012	0.934	-0.635	-0.315	0.524	0.0073	0.0547
4	0.200	0.973	0.947	0.532	0.244	0.529	0.0015	0.0535
5	0.200	1.002	0.965	-0.631	-0.333	0.596	0.0035	0.0232
6	0.199	0.994	0.975	0.775	0.398	0.714	0.0010	0.0260
7	0.199	1.000	0.983	-0.610	-0.267	0.645	0.0014	0.0109
8	0.199	0.999	0.988	0.742	0.399	0.709	0.0005	0.0119
9	0.199	1.000	0.992	-0.035	0.113	0.264	0.0005	0.0098
10	0.199	1.000	0.995	-0.483	-0.281	0.409	0.0005	0.0048
11	0.198	1.000	0.996	0.293	0.098	0.332	0.0002	0.0036
12	0.198	1.000	0.998	0.805	0.564	0.644	0.0002	0.0059
13	0.198	1.000	0.998	0.068	0.237	0.264	0.0002	0.0064
14	0.198	1.000	0.999	-0.851	-0.481	0.652	0.0005	0.0022
15	0.198	1.000	0.999	0.985	0.486	0.872	0.0003	0.0025

Minimum was achieved after 15 iterations.

Final Configuration (Stress= 0.198448 Formula 1)

	D_2_1	D_2_2
O_1	1.144281569	-0.497908299
O_2	-0.620555127	0.658039311
O_3	0.572111824	0.801728943
O_4	-0.004133515	-0.487920895
O_5	-1.091704751	-0.473939059

```
Group Count      Stress Regression Coefficients

    1    25  0.19844843   2.754  -0.894   0.176  -0.012
```

History of Iteration for 1 Dimensions

****

Iter	STRESS	SRAT	SRATAV	CAGRCL	COSAV	ACSAV	SFGR	STEP

```
 0  0.332  0.800  0.800  0.000  0.000  0.000  0.0078  0.0647
    0.330                -0.967                        0.0065
 1  0.332  0.998  0.861  1.000  0.660  0.660  0.0067  0.0175
 2  0.330  0.996  0.904  0.998  0.883  0.883  0.0037  0.0593
    0.331                -0.989                        0.0059
 3  0.330  0.999  0.935  1.000  0.960  0.960  0.0027  0.0210
 4  0.330  0.999  0.956 -0.929 -0.287  0.939  0.0009  0.0076
    0.330                -0.998                        0.0008
 5  0.330  1.000  0.970  1.000  0.563  0.979  0.0007  0.0010
 6  0.330  1.000  0.980  1.000  0.851  0.993  0.0005  0.0024
 7  0.330  1.000  0.987  0.995  0.946  0.994  0.0001  0.0071
    0.330                -0.999                        0.0007
 8  0.330  1.000  0.991 -0.430  0.038  0.622  0.0000  0.0004
    0.330                -0.992                        0.0000
 9  0.330  1.000  0.994  0.466  0.321  0.519  0.0000  0.0000
    0.330                -0.995                        0.0000
10  0.330  1.000  0.996  0.953  0.738  0.805  0.0000  0.0000
    0.330                -1.000                        0.0000
11  0.330  1.000  0.997  0.998  0.910  0.933  0.0000  0.0000
    0.330                -1.000                        0.0000
12  0.330  1.000  0.998  1.000  0.969  0.977  0.0000  0.0000
    0.330                -1.000                        0.0000
13  0.330  1.000  0.999  1.000  0.990  0.992  0.0000  0.0000
    0.330                -1.000                        0.0000
14  0.330  1.000  0.999  1.000  0.996  0.997  0.0000  0.0000
       Minimum was achieved after 14 iterations.


    Final Configuration (Stress= 0.329676 Formula 1)

                          D_1_1
                O_1   1.452973688
                O_2  -0.710018727
                O_3   0.707880939
                O_4  -0.080754068
                O_5  -1.370081832


    Group Count      Stress Regression Coefficients

        1    25  0.32967640   3.139  -1.212   0.223  -0.014
```

5. Example HANGUL by Yoshio Takane: 3-way Rectangular Data: m=3, n=14

```
print "HANGUL: 3-way Rectangular Data: Nsub=3, Nobs=14";
```

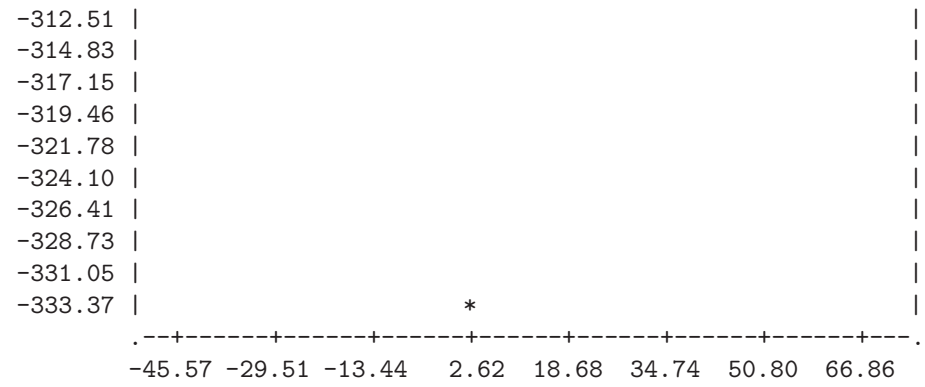
```
n = 14; nw = 3; ndim = 3;
order = [ 24 31  8 42 21 91 25 55 85 15 50 51 27 30 39 26  2 88 87 29 68
          53 49 72 69 67 22 75 57 44 16 19 36 48 10 78 33  7 35  4 61 46
          83 38 70 54 77 74 60 28  3 11 80 64 66 40 43 52 79 56 76 47 71
          90 14 32  6 12 81 23 65  9 63 45 41 73 37 59 58 89 84  5 20 62
          18 13 34 17  1 82 86 ];
dis1 = [ 4 6 3 4 3 1 3 3 2 6 6 7 3 4 6 4 1 2 3 6 6 7 6 1 4 6 6 4 7 4 6 7 6 1 3
         1 4 4 2 4 1 1 6 7 4 6 6 4 7 4 6 6 7 6 3 6 1 4 6 7 7 8 6 2 4 6 6 7 7 3
         3 6 4 2 6 6 7 5 6 3 1 4 7 6 8 6 7 7 4 4 3 ];
dis2 = [ 2 3 1 6 7 1 6 3 5 5 6 5 2 3 6 6 1 2 4 6 3 7 8 3 6 5 6 7 5 6 6 7 6 2 7
         1 6 4 2 5 7 3 5 6 7 7 6 6 7 8 4 7 8 5 4 7 2 4 8 4 3 7 7 4 3 5 4 6 7 6
         2 7 7 2 7 5 6 6 3 6 2 5 6 2 7 3 7 6 5 8 4 ];
dis3 = [ 3 5 2 9 4 1 5 4 8 9 8 8 5 9 8 7 1 6 6 6 8 8 9 2 5 5 6 8 9 8 7 8 4 1 6
         1 9 7 6 8 2 2 7 8 9 6 6 3 8 7 6 5 6 4 3 8 7 6 6 6 7 9 8 2 8 8 6 7 6 4
         6 7 6 1 9 7 7 9 1 7 2 9 7 7 8 2 6 9 6 9 3 ];
dis1[order] = dis1; dis2[order] = dis2; dis3[order] = dis3;

/* get tensor diss[nw,n,n] */
real diss[nw,n,n];
diss[1,.,.] = vec2tri(dis1,n,"ssym"); diss[2,.,.] = vec2tri(dis2,n,"ssym");
diss[3,.,.] = vec2tri(dis3,n,"ssym");
print "Tensor diss=",diss;


rnam = [" GA NA DA LA MA BA SA AA JA CHA KA TA PA HA "];
wnam = [" NAOMI ISABELLE OLEX "];
list tnam[3];
tnam[1] = wnam; tnam[2] = tnam[3] = rnam;
diss = dimname(diss,tnam);
print "Tensor diss=",diss;


 print "HANGUL: Symmetric: nsub=3, nobs=14, ndim=3";
 optn = [ "data"       "diss" ,
          "symm"              ,
          "meth"       "mult" ,
          "model"       "dia" ,
          "regres"    "power" ,
          "ndim"            3 ,
          "ptab"              ,
          "pdis"              ,
          "pres"              ,
          "phis"              ];
 options LS=80;
 < gof,conf,wgt,res > = mds(diss,optn);
```

```
*****************************
3-Way Multidimensional Scaling
*****************************

          Dissimilarity Input Data (Symmetric)
Number Rows of Matrix . . . . . .          14
Dimension . . . . . .           3
Number Subjects / Replications. . . . . .           3
Amount Input Information. . . . . .         273
Number of Parameters. . . . . .          51
Degrees of Freedom. . . . . .         222
Maximum Number Iterations . . . . . .          50
Convergence Criterion for Main Iterations . . .        0.05
Number of Constraint Equations. . . . . .           0
Individual Metric Matrix. . . . . .    DIAGONAL
Estimated Data Transformation Type. . . . . .       POWER
Stimulus Variance Type. . . . . .    CONSTANT
Subject Variance Type . . . . . .     SUBWISE
Assumed Error Distribution. . . . . .   LogNormal
Maximum Number of Configuration Iterations. . . .           1
Convergence Criterion for Configuration . . . .  0.000366367
Maximum Number of Metric Matrix Iterations. . . .           1
Convergence Criterion for Metric Matrix . . . .    0.00109951
Standard Error Weight Estimates . . . . . .    Post-Hoc


*************************************
Compute Metric MULTISCALE by J. Ramsay
*************************************

  Zero Data: 0   Nonzero Data: 546

     Initial Configuration Matrix

            DIM_1        DIM_2        DIM_3
GA     1.693392140  3.440379391 -0.058330375
NA     3.821119948 -0.865510991 -1.535791450
DA     1.242735081  2.824937623  2.057309005
LA     2.926946574 -0.817396710  0.605060613
MA     3.431472988 -0.370735784 -2.321494491
BA    -0.457316347 -1.568108691  2.953564288
SA    -2.857053380 -0.876447666 -2.282018660
AA     1.546868870 -2.013375035  1.268114706
JA    -0.740375144 -0.671116637 -2.832730182
CHA   -3.171367453 -0.345144311 -2.079653150
KA    -2.081242024  4.070858224 -0.393830538
```

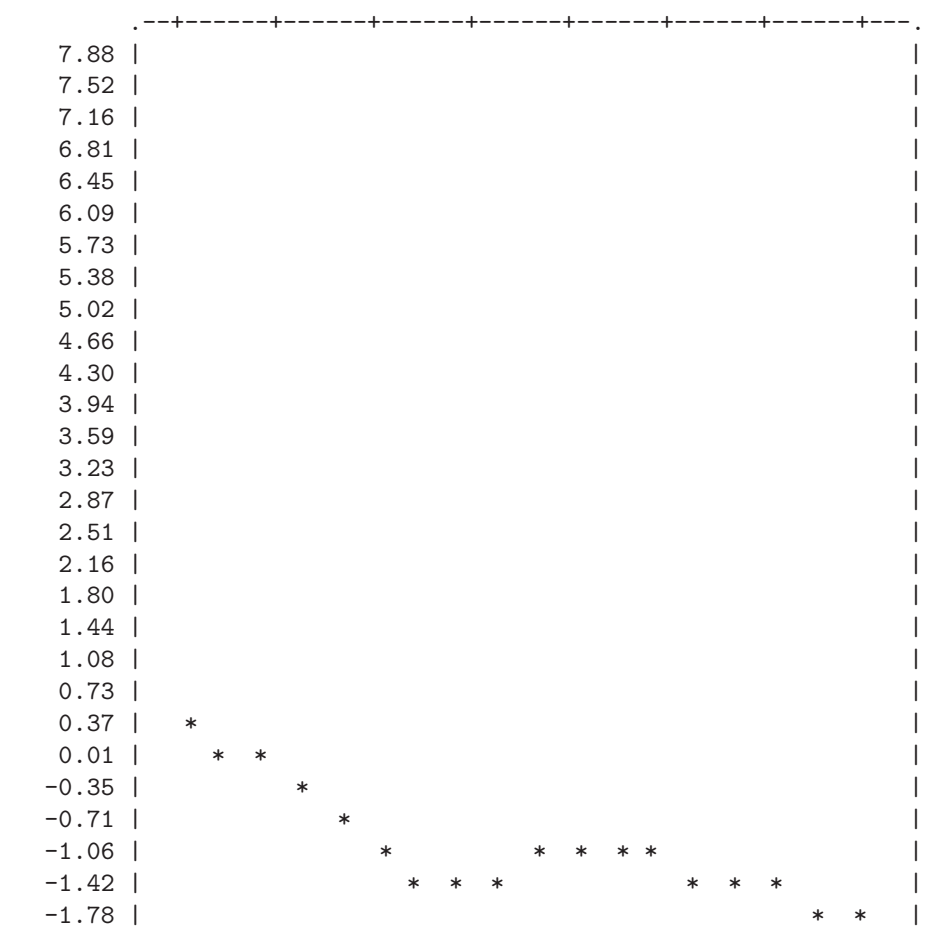
```
        TA   -2.381454586  0.530835880  1.677753277
        PA   -0.755927749 -1.184727728  2.382583253
        HA   -2.217798916 -2.154447564  0.559463706

 MainIter=0 ErrSS=34.7658 LogLik=-332.275 Std.Err=0.504672
                      Grad=3.37034

******************** Main Iteration 1 ********************


                  ************************
                  Configuration Iterations
                  ************************

               Iter  ErrorSS  Penalty    Grad

             0 34.76577380 0.000000000 3.242322213

   Line Search Convergence achieved after 4 Iterations.


             0 20.19175731 0.000000000 0.808229728

                  ************************
                  Weight Matrix Iterations
                  ************************

              Subj  Iter     ErrorSS     GradLgn
   Line Search Convergence achieved after 3 Iterations.
                1     0 5.098514191 0.018974946
   Line Search Convergence achieved after 3 Iterations.
                2     0 8.235164576 0.028990435
   Line Search Convergence achieved after 4 Iterations.
                3     0 6.210575504 0.067967736

 MainIter=1 ErrSS=18.5833 LogLik=-247.913 Std.Err=0.368973
                      Grad=0.781247


******************** Main Iteration 17 ********************


                  ************************
                  Configuration Iterations
                  ************************
```

```
  Iter  ErrorSS  Penalty    Grad

 0 16.07777032 0.000000000 0.028864305

Line Search Convergence achieved after 3 Iterations.


 0 16.07443345 0.000000000 0.016972150
Configuration matrix convergence achieved.

        ************************
        Weight Matrix Iterations
        ************************

  Subj  Iter     ErrorSS     GradLgn


MainIter=17 ErrSS=16.0703 LogLik=-222.141 Std.Err=0.34312
                  Grad=0.0169858
Main Convergence achieved after 17 Iterations.

        ********************************
        Post-Mortem Display of Iterations
        ********************************

 Iter   LogLikel.    ErrorSSQ Grad_Length

    0 -332.275034  34.7657738  3.37034133
    1 -247.912682  18.5832600  0.78124681
    2 -237.995998  17.2037229  0.68795646
    3 -228.132661  16.1328279  0.43250699
    4 -224.694652  16.0185771  0.16987826
    5 -224.111344  16.1412531  0.07203409
    6 -223.947027  16.2305851  0.04191853
    7 -223.834008  16.2993201  0.03925094
    8 -223.718201  16.3456821  0.04228601
    9 -223.562014  16.3497568  0.05768708
   10 -223.325721  16.3119669  0.06698664
   11 -222.986955  16.2402661  0.07156129
   12 -222.696816  16.1705122  0.06266364
   13 -222.459718  16.1037396  0.04617941
   14 -222.320727  16.0801308  0.03924391
   15 -222.224806  16.0631223  0.02864440
   16 -222.170350  16.0777703  0.02346187
```

```
                17 -222.140745  16.0703107  0.01698578

          Log Likelihood Versus Iterations (Dimension=3)
          **********************************************

         .--+------+------+------+------+------+------+------+---.
-222.14 |                             MMM                          |
-224.46 |                           MMM                            |
-226.78 |                                                          |
-229.09 |                          *                               |
-231.41 |                                                          |
-233.73 |                                                          |
-236.04 |                                                          |
-238.36 |                          *                               |
-240.68 |                                                          |
-243.00 |                                                          |
-245.31 |                                                          |
-247.63 |                         *                                |
-249.95 |                                                          |
-252.26 |                                                          |
-254.58 |                                                          |
-256.90 |                                                          |
-259.22 |                                                          |
-261.53 |                                                          |
-263.85 |                                                          |
-266.17 |                                                          |
-268.48 |                                                          |
-270.80 |                                                          |
-273.12 |                                                          |
-275.44 |                                                          |
-277.75 |                                                          |
-280.07 |                                                          |
-282.39 |                                                          |
-284.70 |                                                          |
-287.02 |                                                          |
-289.34 |                                                          |
-291.66 |                                                          |
-293.97 |                                                          |
-296.29 |                                                          |
-298.61 |                                                          |
-300.92 |                                                          |
-303.24 |                                                          |
-305.56 |                                                          |
-307.88 |                                                          |
-310.19 |                                                          |
```



Plot of LOG of Gradient Length vs. Iterations (Dim=3)



```
.--+------+------+------+------+------+------+------+---.
  1.00   3.48   5.96   8.44  10.92  13.40  15.88  18.35
```

****
ML Results for 3 Dimensions
****

Log Likelihood.	-222.141
AIC Statistic	544.281
BIC Statistic	724.755
Unbiased Standard Error Estimate.	0.379642
Number of Parameters.	50
Number of Degrees of Freedom for Error.	223
Maximum Likelihood Standard Error Estimate. . .	0.34312

****
ML Results for Zero Dimensions
****

Log Likelihood.	-397.547
AIC Statistic	801.093
BIC Statistic	811.922
Unbiased Standard Error Estimate.	0.572895
Number of Parameters.	3
Number of Degrees of Freedom for Error.	270

Chi Square for Comparing two Fits	350.812
Number of Degrees of Freedom for Chi Square . . .	47
Probability of Exceeding this Chi Square. . . .	0

Final Configuration Matrix

	DIM_1	DIM_2	DIM_3
GA	-5.833769549	42.95111349	17.49054187
NA	33.71451814	23.01910644	-46.62252275
DA	-6.972180399	32.30519044	26.21958755
LA	74.17208531	4.518121476	3.582671946
MA	40.00171750	38.78290758	-26.40205602
BA	14.43745091	-27.51251267	22.23081108
SA	-61.36680439	-19.85491655	-14.00618563
AA	32.53634008	-18.94987224	-12.58251102
JA	-34.72490670	-2.986183306	-33.73231834
CHA	-35.02008594	-20.98091483	-21.01765981
KA	-26.82689028	31.78583605	21.54301983
TA	-14.13338676	-18.90323856	28.52173339

```
      PA   -6.551859311 -31.25637806  24.62244759
      HA   -3.432228613 -32.91825927  10.15244032
The decimal point has been moved 1 places to the right.
```

Results in Polar Coordinate Form

	1	2	3
GA	46.741310	98.000000	108.00000
NA	61.969409	34.000000	-54.000000
DA	42.186531	102.00000	105.00000
LA	74.395882	3.0000000	3.0000000
MA	61.654845	44.000000	-33.000000
BA	38.204546	-62.000000	57.000000
SA	66.002088	198.00000	193.00000
AA	39.699253	-30.000000	-21.000000
JA	48.503667	185.00000	224.00000
CHA	45.916742	211.00000	211.00000
KA	46.841468	130.00000	141.00000
TA	37.021269	233.00000	116.00000
PA	40.325587	258.00000	105.00000
HA	34.618839	264.00000	109.00000

First Value is Distance from the Origin
Other Values are Angles in Degrees from First Dimension

****
Standard Deviations and Variances for each Dimension
****

Dim	Std.Dev.	Variance	Percent
1	34.60001437	1197.160994	47
2	27.16411058	737.8889034	29
3	24.33720052	592.2993292	23

Interpoint Distances

	GA	NA	DA	LA
GA	0	77	13	89
NA	77	0	83	67
DA	13	83	0	88
LA	89	67	88	0
MA	63	26	70	56
BA	73	87	63	70
SA	89	109	85	138

AA	78	54	75	50
JA	74	74	74	115
CHA	80	85	76	114
KA	24	91	20	106
TA	63	98	51	94
PA	74	98	63	90
HA	76	87	67	86

.....

Final Weight Matrix

	DIM_1	DIM_2	DIM_3
NAOMI	0.412642095	1.155887660	1.222149835
ISABELLE	1.218159780	1.158094466	0.418215205
OLEX	1.160933994	0.567936658	1.153117607

Results in Polar Coordinate Form

	1	2	3
NAOMI	1.7320508	70.000000	45.000000
ISABELLE	1.7320508	44.000000	43.000000
OLEX	1.7320508	26.000000	0.0000000

First Value is Distance from the Origin
Other Values are Angles in Degrees from First Dimension

****
Post-Hoc Standard Error Weight Estimates
****

	Input Order			Size Order	
1	GA	0.00000000	1	GA	0.00000000
2	NA	0.61133448	4	LA	0.16585465
3	DA	2.29960410	5	MA	0.25344875
4	LA	0.16585465	2	NA	0.61133448
5	MA	0.25344875	13	PA	0.64146629
6	BA	1.26767980	11	KA	0.81903743
7	SA	1.66273222	14	HA	0.87274233
8	AA	1.25955433	9	JA	1.06895864
9	JA	1.06895864	8	AA	1.25955433
10	CHA	1.49133855	6	BA	1.26767980
11	KA	0.81903743	10	CHA	1.49133855
12	TA	2.00985566	7	SA	1.66273222
13	PA	0.64146629	12	TA	2.00985566
14	HA	0.87274233	3	DA	2.29960410

```
      ***************************************************
      Final Regression Coefficient and Exponent Estimates
      ***************************************************


   Subject Regression   Exponent Reciprocal     Subject   Exponent
          Coefficient                                     by Size
 1   NAOMI 1.40680746 0.93303742 1.07176838   1   NAOMI 0.93303742
 2ISABELLE 0.72030562 1.31213620 0.76211601   3    OLEX 0.99164164
 3    OLEX 0.98684407 0.99164164 1.00842881   2ISABELLE 1.31213620
           Mean= 1.07894 Standard Deviation= 0.20407


   ***********************************************************
   Within-Subj. Std.Error Estimates and Multiple Correlations
   ***********************************************************

        Standard Error                 Multiple Correlation
Subj  Unordered Subj    Ordered   Subj  Unordered Subj    Ordered
   1       0.22    1       0.22      1       0.91    2       0.67
   2       0.45    3       0.32      2       0.67    3       0.75
   3       0.32    2       0.45      3       0.75    1       0.91


       ***************************************************
       Asympt. Standard Errors of Estimate of Coordinates
       ***************************************************

          1          GA    -0.5834     4.2951     1.7491
                            0.7183     0.8379     0.6427

        Correlation Matrix 1 for Coordinate Estimates

                     DIM_1        DIM_2        DIM_3
        DIM_1   1.000000000 -0.024978736  0.440441259
        DIM_2  -0.024978736  1.000000000 -0.092418119
        DIM_3   0.440441259 -0.092418119  1.000000000


          2          NA     3.3715     2.3019    -4.6623
                            0.4957     0.7825     0.4466

        Correlation Matrix 2 for Coordinate Estimates

                     DIM_1        DIM_2        DIM_3
```

```
DIM_1   1.000000000  0.033762208 -0.100446586
DIM_2   0.033762208  1.000000000 -0.455694562
DIM_3  -0.100446586 -0.455694562  1.000000000

  3          DA    -0.6972     3.2305     2.6220
                    0.4882     0.5092     0.6462

Correlation Matrix 3 for Coordinate Estimates

              DIM_1        DIM_2        DIM_3
DIM_1   1.000000000  0.133052899  0.175968019
DIM_2   0.133052899  1.000000000  0.077459569
DIM_3   0.175968019  0.077459569  1.000000000

  4          LA     7.4172     0.4518     0.3583
                    0.5384     0.5615     0.5177

Correlation Matrix 4 for Coordinate Estimates

              DIM_1        DIM_2        DIM_3
DIM_1   1.000000000 -0.485018869 -0.046880810
DIM_2  -0.485018869  1.000000000 -0.209008714
DIM_3  -0.046880810 -0.209008714  1.000000000

  5          MA     4.0002     3.8783    -2.6402
                    0.5040     0.5151     0.4546

Correlation Matrix 5 for Coordinate Estimates

              DIM_1        DIM_2        DIM_3
DIM_1   1.000000000  0.491034736 -0.057189755
DIM_2   0.491034736  1.000000000 -0.071323172
DIM_3  -0.057189755 -0.071323172  1.000000000

  6          BA     1.4437    -2.7513     2.2231
                    0.9316     0.5191     0.8535

Correlation Matrix 6 for Coordinate Estimates

              DIM_1        DIM_2        DIM_3
DIM_1   1.000000000 -0.588478790  0.025758511
DIM_2  -0.588478790  1.000000000 -0.107072269
DIM_3   0.025758511 -0.107072269  1.000000000
```

```
  7          SA    -6.1367    -1.9855    -1.4006
                    0.7397     0.5875     0.9220

Correlation Matrix 7 for Coordinate Estimates

              DIM_1        DIM_2        DIM_3
DIM_1   1.000000000 -0.304544151 -0.114603685
DIM_2  -0.304544151  1.000000000 -0.361201975
DIM_3  -0.114603685 -0.361201975  1.000000000


  8          AA     3.2536    -1.8950    -1.2583
                    0.6351     0.7670     0.6686

Correlation Matrix 8 for Coordinate Estimates

              DIM_1        DIM_2        DIM_3
DIM_1   1.000000000 -0.063057167 -0.127590439
DIM_2  -0.063057167  1.000000000  0.671252254
DIM_3  -0.127590439  0.671252254  1.000000000


  9          JA    -3.4725    -0.2986    -3.3732
                    0.5475     0.5658     0.5126

Correlation Matrix 9 for Coordinate Estimates

              DIM_1        DIM_2        DIM_3
DIM_1   1.000000000  0.513133620  0.456933319
DIM_2   0.513133620  1.000000000  0.781066209
DIM_3   0.456933319  0.781066209  1.000000000


 10         CHA    -3.5020    -2.0981    -2.1018
                    0.4702     0.7437     0.5956

Correlation Matrix 10 for Coordinate Estimates

              DIM_1        DIM_2        DIM_3
DIM_1   1.000000000 -0.264506346 -0.196169208
DIM_2  -0.264506346  1.000000000  0.319587558
DIM_3  -0.196169208  0.319587558  1.000000000


 11          KA    -2.6827     3.1786     2.1543
                    0.6329     0.8756     0.7605
```

Correlation Matrix 11 for Coordinate Estimates

	DIM_1	DIM_2	DIM_3
DIM_1	1.000000000	-0.080332951	0.401870991
DIM_2	-0.080332951	1.000000000	0.213916668
DIM_3	0.401870991	0.213916668	1.000000000

12	TA	-1.4133	-1.8903	2.8522
		0.5556	0.6714	0.5545

Correlation Matrix 12 for Coordinate Estimates

	DIM_1	DIM_2	DIM_3
DIM_1	1.000000000	0.139738633	0.404038865
DIM_2	0.139738633	1.000000000	-0.109471051
DIM_3	0.404038865	-0.109471051	1.000000000

13	PA	-0.6552	-3.1256	2.4622
		0.5078	0.5252	0.7352

Correlation Matrix 13 for Coordinate Estimates

	DIM_1	DIM_2	DIM_3
DIM_1	1.000000000	0.336710058	-0.014465145
DIM_2	0.336710058	1.000000000	-0.240615577
DIM_3	-0.014465145	-0.240615577	1.000000000

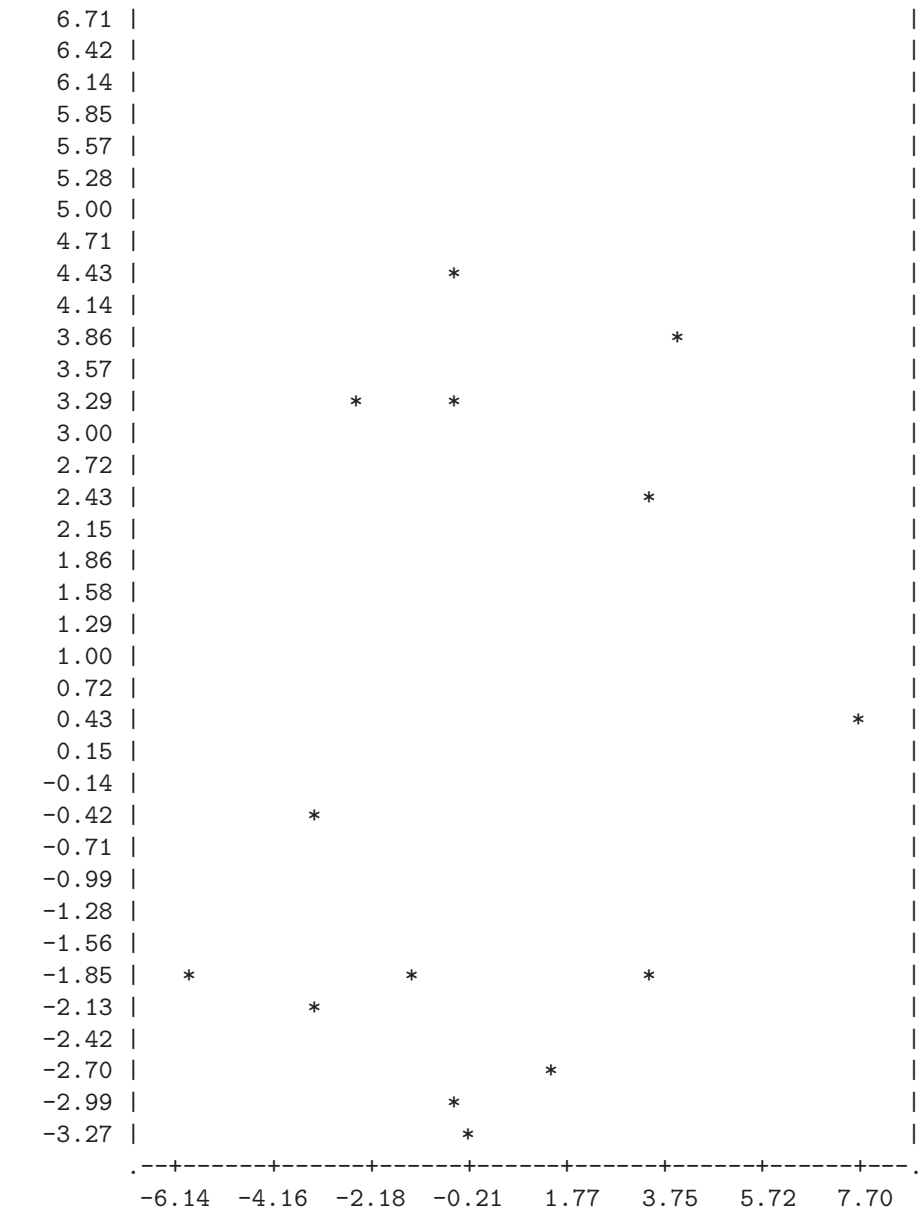
14	HA	-0.3432	-3.2918	1.0152
		0.4934	0.5487	0.5469

Correlation Matrix 14 for Coordinate Estimates

	DIM_1	DIM_2	DIM_3
DIM_1	1.000000000	0.505301215	0.531299385
DIM_2	0.505301215	1.000000000	0.877935810
DIM_3	0.531299385	0.877935810	1.000000000

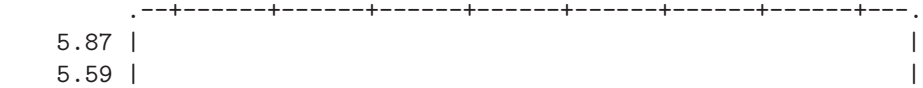
Configuration: Dimension 2 (Y-Axis) vs. Dimension 1 (X-Axis)

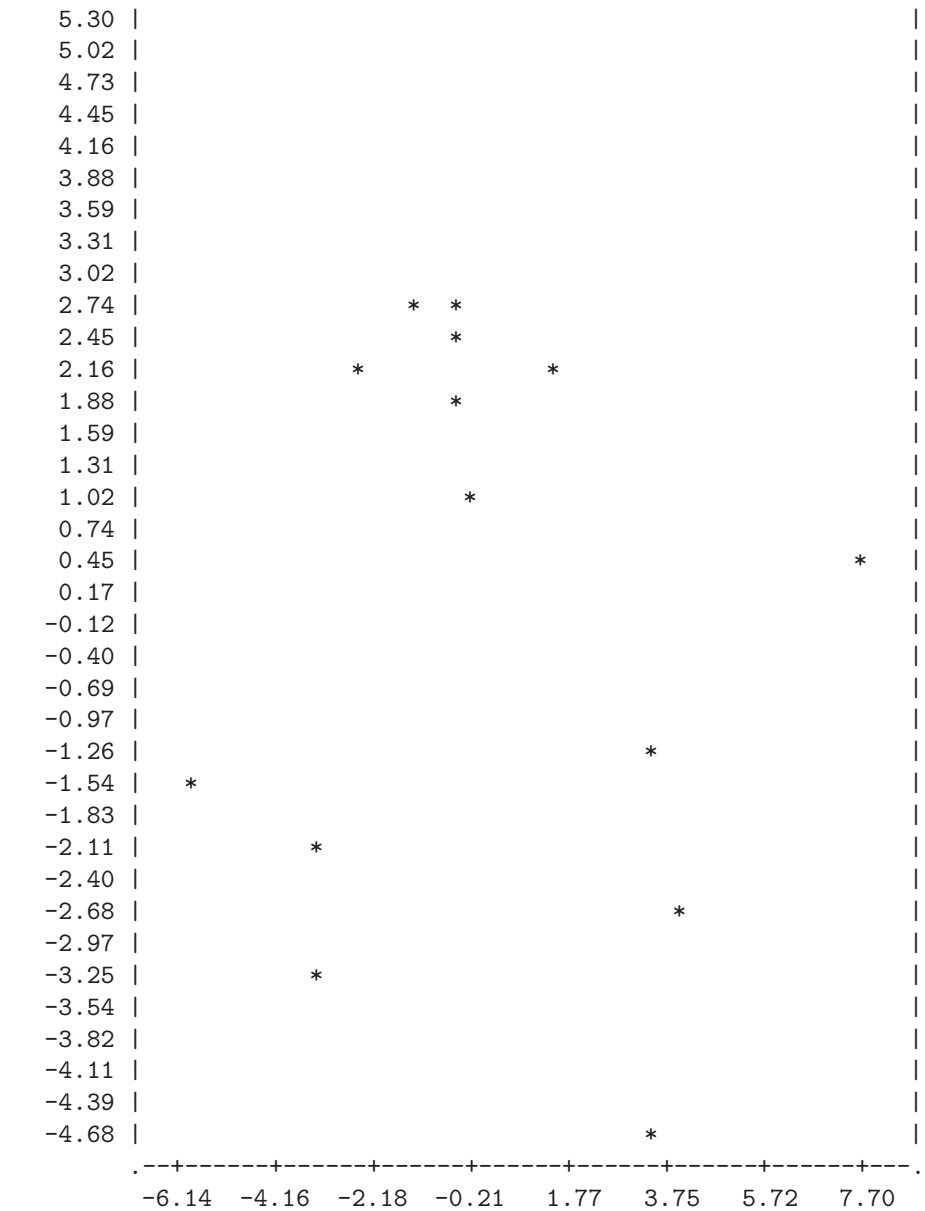
```
     .--+------+------+------+------+------+------+------+---.
7.28 |                                                        |
6.99 |                                                        |
```



Configuration: Dimension 3 (Y-Axis) vs. Dimension 1 (X-Axis)

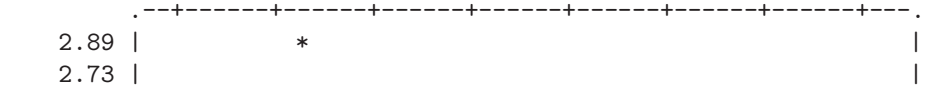
****

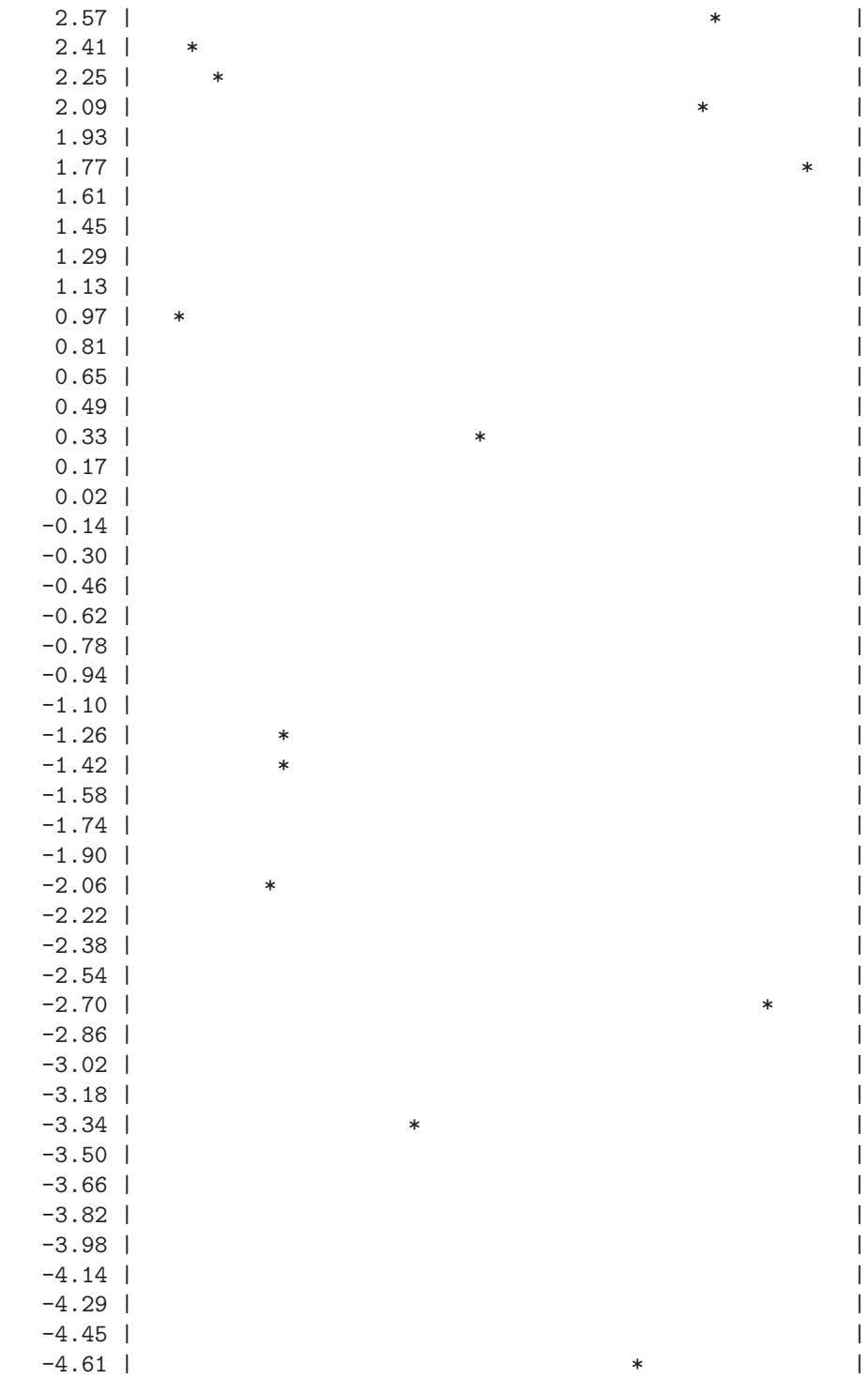


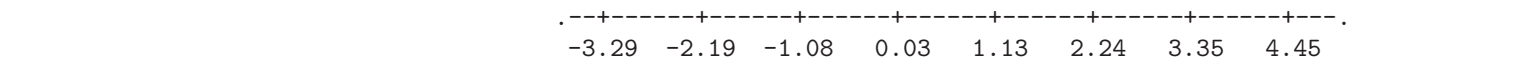


Configuration: Dimension 3 (Y-Axis) vs. Dimension 2 (X-Axis)

****




2.57 |
2.41 |
2.25 |
2.09 |
1.93 |
1.77 |
1.61 |
1.45 |
1.29 |
1.13 |
0.97 |
0.81 |
0.65 |
0.49 |
0.33 |
0.17 |
0.02 |
-0.14 |
-0.30 |
-0.46 |
-0.62 |
-0.78 |
-0.94 |
-1.10 |
-1.26 |
-1.42 |
-1.58 |
-1.74 |
-1.90 |
-2.06 |
-2.22 |
-2.38 |
-2.54 |
-2.70 |
-2.86 |
-3.02 |
-3.18 |
-3.34 |
-3.50 |
-3.66 |
-3.82 |
-3.98 |
-4.14 |
-4.29 |
-4.45 |
-4.61 |



Transformed Observations: Subject=1

	GA	NA	DA	LA
GA	0.000000000	5.128367782	1.406807460	5.128367782
NA	5.128367782	0.000000000	7.486501175	5.128367782
DA	1.406807460	7.486501175	0.000000000	7.486501175
LA	5.128367782	5.128367782	7.486501175	0.000000000
MA	5.128367782	3.921088520	7.486501175	3.921088520
BA	7.486501175	8.644557291	7.486501175	5.128367782
SA	7.486501175	8.644557291	9.791549267	8.644557291
AA	7.486501175	3.921088520	5.128367782	3.921088520
JA	7.486501175	5.128367782	7.486501175	7.486501175
CHA	8.644557291	8.644557291	7.486501175	7.486501175
KA	1.406807460	9.791549267	1.406807460	7.486501175
TA	8.644557291	8.644557291	7.486501175	6.315385005
PA	7.486501175	7.486501175	5.128367782	5.128367782
HA	7.486501175	8.644557291	8.644557291	5.128367782

Smallest Value: 1.40681 Largest Value: 9.79155

Interpoint Distances: Subject=1

	GA	NA	DA	LA
GA	0.000000000	7.158159131	1.370558004	6.191635096
NA	7.158159131	0.000000000	7.795151562	5.891682281
DA	1.370558004	7.795151562	0.000000000	5.951115483
LA	6.191635096	5.891682281	5.951115483	0.000000000
MA	5.206060788	2.591533808	6.026122181	4.956083155
BA	7.045076335	8.625782950	6.027053099	5.077400153
SA	7.674641926	7.734798475	7.282667820	8.502045660
AA	7.166111207	5.372933335	6.794128139	3.738176556
JA	7.077124791	4.939050764	7.170939395	7.462352230
CHA	7.589068483	6.458338546	7.277167683	7.321478936
KA	1.700929113	7.801900058	1.259435926	6.758443997
TA	6.201400534	9.087634037	5.057124107	6.209752037
PA	7.330885611	9.269960481	6.250776957	6.274938090
HA	7.496727933	8.242553984	6.617793875	5.896113597

Smallest Value: 1.25944 Largest Value: 9.26996

Normalized Residuals: Subject=1

```
              MA           BA           SA           AA
GA   -0.067030618  0.270923841 -0.110647805  0.194985653
NA    1.846142233  0.009692450  0.495731186 -1.404289434
DA    0.967375305  0.966686685  1.319666588 -1.253906046
LA   -1.044269497  0.044526847  0.074105732  0.212964481
MA    0.000000000 -0.460180256  0.137649537 -0.595626465
BA   -0.460180256  0.000000000 -1.756529498  0.173533385
SA    0.137649537 -1.756529498  0.000000000 -0.326683485
AA   -0.595626465  0.173533385 -0.326683485  0.000000000
JA   -0.741271493  1.074014716 -0.535849169  2.020291872
CHA   0.073331919 -0.136792760 -0.858941556  1.037428300
KA    0.787188369  1.400725162 -1.085591432  0.943633307
TA    0.049187159 -1.533094294  1.696078236  0.132744744
PA   -0.834288953  0.320711097  1.638671557  0.514433155
HA   -0.490682645 -0.832249088 -2.180979586  0.622017989
.....................................................
    Smallest Value: -2.42579 Largest Value: 2.43163


      Quantiles of Normalized Residuals: Subject=1

              GA           NA           DA           LA
GA    0.000000000 -1.359962536  0.054375453 -0.744341114
NA   -1.359962536  0.000000000 -0.332563025 -0.574739567
DA    0.054375453 -0.332563025  0.000000000  1.027144882
LA   -0.744341114 -0.574739567  1.027144882  0.000000000
MA   -0.247313727  1.844177266  0.938731970 -0.981979846
BA    0.390753455 -0.191527695  0.897170710 -0.136356872
SA   -0.275495595  0.511531310  1.294223527 -0.054375453
AA    0.275495595 -1.294223527 -1.233639904  0.332563025
JA    0.361503163  0.163876553  0.247313727 -0.163876553
CHA   0.640341274  1.233639904  0.081626883  0.027173016
KA   -0.780818419  0.981979846  0.480696247  0.450314414
TA    1.432159890 -0.361503163  1.712054734 -0.027173016
PA    1.0101e-007 -0.938731970 -0.857103039 -0.897170710
HA   -0.219332424  0.303903190  1.177268879 -0.607212923
.....................................................
    Smallest Value: -2.29533 Largest Value: 2.29533
```

3.13 Function `selc`

< nxtgen,nxtforb > = selc(inides,forbid,nlev,optn)

Purpose: Sequential elimination of level combinations algorithm: a method using the cross over and mutation operations of an genetic algorithm for finding nearly optimal experimental designs. For a given data set containing categorical data describing the level combinations for experiments with an outcome value y and a matrix of forbidden designs a new matrix of promising designs is computed. The new designs then must be used in experiments and corresponding outcome values determined. Depending on the outcome of the experiments the designs are then either added to the new initial design or used in adding new rows to the forbidden array and the computation is repeated.

Input: **inides** this is a $N \times n + 1$ matrix containing the integer level measurements (levels) of n categorical effects x_j in its first columns and an integer outcome y in its last column which can be positive or negative.

forbid this is a $M \times n$ matrix containing the M forbidden level combinations of the n effects.

nlev this is an n vector of positive integers specifying the upper most level for each effect. (Specifying this information is necessary since the data may have unobserved levels.)

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

2. P the integer number of offsprings, i.e. the number promising level combinations to be computed which is the number of rows of the output argument `nxtdes`. P must be a positive integer, default is 10.
3. s the integer strength applied to the forbidden array. s should be a positive integer, default is 2.
4. o the order applied to the forbidden array, default is 2.
5. should be either 1 for maximization (default) or 0 for for minimization. (Depends on the orientation of the values of the outcome y.)
6. specifies the seed value for the random generator (default is time of day).

Output: **nxtdes** This is an $P \times n$ matrix with the P promising level combinations found by the algorithm. Note, P is specified by `optn[2]`.

nxtforb During the computation the input forbidden array is extended by additional rows. This output argument shows the input forbidden array in the first rows and for the last rows the newly added forbidden patterns as used in the algorithm.

Restrictions: 1. The level measurements of the data `inides` and `forbid` array must be integers in $[1, K_j]$ for the effect j.

2. The outcome variable y may consist of integers which can be positive or negative.

Relationships:

Examples: 1. Small example without significant interactions: Note, that level 5 in effect 1 is not contained in the data of the initial design.

```
options NOECHO;
ides = [
%inc "..\\tdata\\selc.dat";
       ];
options ECHO;


forbid = [ 3  10   3,  1  23  10,  1  23  23,  1  25  10,
           2  21  57,  2  23  57,  2  29  32,  2  29  57,
           2  30  57,  2  32  57,  2  33  57,  2  34  17,
           2  34  32,  2  34  34,  2  34  36,  2  34  37,
           2  34  45,  2  34  54,  2  34  56,  2  34  57,
           2  34 179,  3  23  10,  3  23  23,  3  25  21,
           3  25  23,  3  25  24,  4  21  57,  4  23  10,
           4  23  21,  4  23  29,  4  23 139,  4  23 171,
           4  23 185,  4  23 214,  4  25  21,  4  25  24,
           4  25  52,  4  25 125,  4  25 139,  4  25 171,
           4  25 185,  4  25 214,  4  25 222,  4  25 224,
           4  25 229,  4  29  32,  4  29  57,  4  30  57,
           4  33  57,  4  34  32,  4  34  34,  4  34  36,
           4  34  37,  4  34  45,  4  34  54,  4  34  56,
           4  34  57,  4  34 179,  5  18 119,  5  18 136,
           5  18 148,  5  18 153,  5  18 156,  5  18 181,
           5  18 182,  5  18 195,  5  18 197,  5  19 148,
           5  20 148,  5  20 181,  5  20 182 ];


inides = shape(ides,.,4);
print "nrow=",nrow(inides);
nlev = [ 5  34  241 ];


print "Option like MATLAB";
optn = [   2 ,  /* ipri */
          10 ,  /* noff */
           2 ,  /* istr */
           2 ,  /* ordr */
           1 ,  /* imax */
```

```
        123 ]; /* seed */
optn[10] = 1;
< nxtdes,nxtforb > = selc(inides,forbid,nlev,optn);
```

```
 ************************************************************
 Sequential Elimination of Level Combinations (Maximization)
 ************************************************************

  Number of Observations . . . . . .        173
  Rows of Forbidden Array. . . . . .         71
  Number Offspring . . . . . .               10
  Strength of Forbidden Array  . . . .        2
  Order of Forbidden Array . . . . . .        2

              Input: Level Numbers for Effects

                          1
                Eff_1     5
                Eff_2    34
                Eff_3   241
```

Frequencies of Effect Levels in Initial Design

1	1	26	3	1	11
	2	24		3	6
	3	69		6	9
	4	54		8	12
2	1	10		9	7
	2	8		10	4
	3	14		11	4
	4	6		13	12
	5	6		14	4
	6	13		20	10
	7	8		22	2
	8	8		27	7
	9	8		28	12
	10	9		30	10
	12	8		32	2
	13	9		34	7
	14	10		35	7
	15	11		37	4
	16	5		40	7
	17	2		46	15
	22	11		54	1
	23	4		58	3
	24	8		172	9
	25	1		227	8
	27	3			
	28	11			

There are 4 observations added to the forbidden array.

Output: Next Design
****

Dense Matrix (10 by 3)

	Eff_1	Eff_2	Eff_3
1	1	6	226
2	2	18	30
3	2	6	6
4	1	32	28
5	3	6	127

6	3	22	64
7	4	26	46
8	1	6	89
9	2	15	158
10	3	24	40

New Forbidden Array
****

Dense Matrix (75 by 3)

	1	2	3
1	3	10	3
2	1	23	10
3	1	23	23
4	1	25	10
5	2	21	57
6	2	23	57
7	2	29	32
8	2	29	57
9	2	30	57
10	2	32	57
11	2	33	57
12	2	34	17
13	2	34	32
14	2	34	34
15	2	34	36
16	2	34	37
17	2	34	45
18	2	34	54
19	2	34	56
20	2	34	57
21	2	34	179
22	3	23	10
23	3	23	23
24	3	25	21
25	3	25	23
26	3	25	24
27	4	21	57
28	4	23	10
29	4	23	21
30	4	23	29
31	4	23	139
32	4	23	171

33	4	23	185
34	4	23	214
35	4	25	21
36	4	25	24
37	4	25	52
38	4	25	125
39	4	25	139
40	4	25	171
41	4	25	185
42	4	25	214
43	4	25	222
44	4	25	224
45	4	25	229
46	4	29	32
47	4	29	57
48	4	30	57
49	4	33	57
50	4	34	32

	1	2	3
51	4	34	34
52	4	34	36
53	4	34	37
54	4	34	45
55	4	34	54
56	4	34	56
57	4	34	57
58	4	34	179
59	5	18	119
60	5	18	136
61	5	18	148
62	5	18	153
63	5	18	156
64	5	18	181
65	5	18	182
66	5	18	195
67	5	18	197
68	5	19	148
69	5	20	148
70	5	20	181
71	5	20	182
72	1	13	9
73	3	8	3
74	3	10	3

```
                    75 |       3     13     34
```

```
print "Option like SAS/IML version";
optn = [   2 ,  /* ipri */
           5 ,  /* noff */
          10 ,  /* istr */
           2 ,  /* ordr */
           1 ,  /* imax */
           1 ]; /* seed */
nxtdes = selc(inides,forbid,nlev,optn);
print "Nextdes=",nxtdes;
```

2. Example containing significant interactions:

```
print "Johnson, Mandal, Ding: JSS 317: Revised Example 2";
options NOECHO;
ides = [
%inc "..\\tdata\\selc2.dat";
       ];
options ECHO;

forbid = .;
inides2 = shape(ides,.,5);
print "nrow=",nrow(inides2);
nlev = [ 11 11 11 11 ];


optn = [   2 ,  /* ipri */
           5 ,  /* noff */
           2 ,  /* istr */
           2 ,  /* ordr */
           1 ,  /* imax */
           1 ]; /* seed */
optn[10] = 1;
< nxtdes,nxtforb > = selc(inides2,forbid,nlev,optn);
print "Nextdes=",nxtdes;
print "NextForb=",nxtforb;
```

```
 ************************************************************
 Sequential Elimination of Level Combinations (Maximization)
 ************************************************************

  Number of Observations . . . . . .       242
  Rows of Forbidden Array. . . . . .         0
  Number Offspring . . . . . .               5
```

```
Strength of Forbidden Array  . . . . . .        2
Order of Forbidden Array . . . . . .        2
```

Input: Level Numbers for Effects

	1
Eff_1	11
Eff_2	11
Eff_3	11
Eff_4	11

Frequencies of Effect Levels in Initial Design

1	1	22	3	1	22
	2	22		2	22
	3	22		3	22
	4	22		4	22
	5	22		5	22
	6	22		6	22
	7	22		7	22
	8	22		8	22
	9	22		9	22
	10	22		10	22
	11	22		11	22
2	1	22	4	1	22
	2	22		2	22
	3	22		3	22
	4	22		4	22
	5	22		5	22
	6	22		6	22
	7	22		7	22
	8	22		8	22
	9	22		9	22
	10	22		10	22
	11	22		11	22

There are 2 observations added to the forbidden array.

Output: Next Design

****

Dense Matrix (5 by 4)

	Eff_1	Eff_2	Eff_3	Eff_4
1	10	11	11	5
2	8	8	8	1
3	3	8	11	10
4	11	3	11	6
5	9	10	11	3

New Forbidden Array

****

Dense Matrix (2 by 4)

	1	2	3	4
1	4	2	3	3
2	7	2	4	2

3.14 Function urd1out

< effrep,outrep,cordat > = urd1out(X,y<,optn>)

Purpose: The function judges significant effects in unreplicated 2^k and 2^{k-p} factorial designs. The method makes it possible to detect a single outlier with the method by Daniel (1960).

Input: **X** is an $N \times n$, $N > n$, matrix of N designs for n effects coded in integers.

y is an N vector of real response values.

optn

Output: **effrep** a table of effects indicating which is found significant

outrep a table describing the outlier anlysis

cordat a table of the original and corrected response data indicating which of the y is found an outlier

Restrictions: 1. The input data X and y may not contain any missing values or string data.

2. The number of rows of matrix X and vector y must be the same.

Relationships:

Examples: 1. Example 1 of submission:

```
nobs = 16; neff = 15;
xdat = cons(nobs,neff,.);
xrow = cons(1,neff,.);
i = 1;
for (d = -1; d <= 1; d+=2)
for (c = -1; c <= 1; c+=2)
for (b = -1; b <= 1; b+=2)
for (a = -1; a <= 1; a+=2) {
  xrow[1] = a; xrow[2] = b; xrow[3] = c; xrow[4] = d;
  xrow[5] = a*b;  xrow[6] = a*c;  xrow[ 7] = a*d;
  xrow[8] = b*c;  xrow[9] = b*d;  xrow[10] = c*d;
  xrow[11] = a*b*c; xrow[12] = a*b*d; xrow[13] = a*c*d;
  xrow[14] = b*c*d; xrow[15] = a*b*c*d;
  xdat[i,] = xrow; i++;
}
cnam = [" a b c d ab ac ad bc bd cd abc abd acd bcd abcd "];
xdat = cname(xdat,cnam);
```

```
print "Example1 of Submission: nobs=16=2^4, neff=11";
y1 = [ 1.68 1.98 3.28 3.44 4.98 5.70 9.97 9.07 2.07
       2.44 4.09 4.53 7.77 9.43 11.75 16.3 ];
y1 = log10(y1); /* print y1; */
optn = [   2 ];  /* ipri */
< effrep,outrep,cordat > = urd1out(xdat,y1,optn);


*****************************************************************
Analysis of Unreplicated 2^k and 2^(k-p) Designs (Lawson, 2007)
*****************************************************************

   Number of Observations . . . . . .        16
   Number of Effects. . . . . .        15
```

Input Design Matrix

	a	b	c	d	ab	ac	ad	bc	bd
1	-1.000	-1.000	-1.000	-1.000	1.000	1.000	1.000	1.000	1.000
2	1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	1.000	1.000
3	-1.000	1.000	-1.000	-1.000	-1.000	1.000	1.000	-1.000	-1.000
4	1.000	1.000	-1.000	-1.000	1.000	-1.000	-1.000	-1.000	-1.000
5	-1.000	-1.000	1.000	-1.000	1.000	-1.000	1.000	-1.000	1.000
6	1.000	-1.000	1.000	-1.000	-1.000	1.000	-1.000	-1.000	1.000
7	-1.000	1.000	1.000	-1.000	-1.000	-1.000	1.000	1.000	-1.000
8	1.000	1.000	1.000	-1.000	1.000	1.000	-1.000	1.000	-1.000
9	-1.000	-1.000	-1.000	1.000	1.000	1.000	-1.000	1.000	-1.000
10	1.000	-1.000	-1.000	1.000	-1.000	-1.000	1.000	1.000	-1.000
11	-1.000	1.000	-1.000	1.000	-1.000	1.000	-1.000	-1.000	1.000
12	1.000	1.000	-1.000	1.000	1.000	-1.000	1.000	-1.000	1.000
13	-1.000	-1.000	1.000	1.000	1.000	-1.000	-1.000	-1.000	-1.000
14	1.000	-1.000	1.000	1.000	-1.000	1.000	1.000	-1.000	-1.000
15	-1.000	1.000	1.000	1.000	-1.000	-1.000	-1.000	1.000	1.000
16	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000

Input Design Matrix

	cd	abc	abd	acd	bcd	abcd
1	1.000	-1.000	-1.000	-1.000	-1.000	1.000
2	1.000	1.000	1.000	1.000	-1.000	-1.000
3	1.000	1.000	1.000	-1.000	1.000	-1.000
4	1.000	-1.000	-1.000	1.000	1.000	1.000
5	-1.000	1.000	-1.000	1.000	1.000	-1.000
6	-1.000	-1.000	1.000	-1.000	1.000	1.000
7	-1.000	-1.000	1.000	1.000	-1.000	1.000

8	-1.000	1.000	-1.000	-1.000	-1.000	-1.000
9	-1.000	-1.000	1.000	1.000	1.000	-1.000
10	-1.000	1.000	-1.000	-1.000	1.000	1.000
11	-1.000	1.000	-1.000	1.000	-1.000	1.000
12	-1.000	-1.000	1.000	-1.000	-1.000	-1.000
13	1.000	1.000	1.000	-1.000	-1.000	1.000
14	1.000	-1.000	-1.000	1.000	-1.000	-1.000
15	1.000	-1.000	-1.000	-1.000	1.000	-1.000
16	1.000	1.000	1.000	1.000	1.000	1.000

Input Response Vector

	1
1	0.225
2	0.297
3	0.516
4	0.537
5	0.697
6	0.756
7	0.999
8	0.958
9	0.316
10	0.387
11	0.612
12	0.656
13	0.890
14	0.975
15	1.070
16	1.212

****
Significance of Effect Report
****

Label	Half_Effect	Abs_Effect	CL_Upper	Rank	95%Sig
a	0.028227058	0.02822706	0.02898070	12	no
b	0.125963135	0.12596313	0.03833962	14	yes
c	0.250686024	0.25068602	0.04843945	15	yes
d	0.070907904	0.07090790	0.03291497	13	yes
ab	-0.007461889	0.00746189	0.01093157	4	no
ac	0.002247698	0.00224770	0.00592203	1	no
ad	0.014527290	0.01452729	0.02307974	10	no
bc	-0.010901759	0.01090176	0.01642879	7	no
bd	-0.003243873	0.00324387	0.00922863	3	no
cd	0.021311038	0.02131104	0.02580127	11	no

```
       abc  0.002252599  0.00225260  0.00756310     2     no
       abd  0.011338698  0.01133870  0.01846654     8     no
       acd  0.011558056  0.01155806  0.02066446     9     no
       bcd -0.007494313  0.00749431  0.01268673     5     no
      abcd  0.008385358  0.00838536  0.01451179     6     no

Lawson,Grimshaw,and Burt Rn=3.35745 with 1.201 95% Percent.


                    **********
                    Corrected Data Report
                    **********

                N     Response   Corr.Resp. Outlier
                1  0.225309282  0.225309282      no
                2  0.296665190  0.296665190      no
                3  0.515873844  0.515873844      no
                4  0.536558443  0.536558443      no
                5  0.697229343  0.697229343      no
                6  0.755874856  0.755874856      no
                7  0.998695158  0.998695158      no
                8  0.957607287  0.957607287      no
                9  0.315970345  0.315970345      no
               10  0.387389826  0.387389826      no
               11  0.611723308  0.611723308      no
               12  0.656098202  0.656098202      no
               13  0.890421019  0.890421019      no
               14  0.974511693  0.974511693      no
               15  1.070037867  1.176690239     yes
               16  1.212187604  1.212187604      no


  Perc.        Gap        PSE  Stand.Gap   Critical Signif.
   50th  0.0054916  0.0144653  1.9104971  1.7880000    yes
   99th  0.0054916  0.0066234  4.1724537  5.1000000     no
```

2. Example 2 of submission:

```
print "Example 2 of Submission: nobs=16=2^4, neff=11";
y2 = [ 47.46 49.62 43.13 46.31 51.47 48.49 49.34 46.10
       46.76 48.56 44.83 44.45 59.15 51.33 47.02 47.90 ];
optn = [   2 ];  /* ipri */
< effrep,outrep,cordat > = urd1out(xdat,y2,optn);


                    Input Response Vector
```

	1
1	47.46
2	49.62
3	43.13
4	46.31
5	51.47
6	48.49
7	49.34
8	46.10
9	46.76
10	48.56
11	44.83
12	44.45
13	59.15
14	51.33
15	47.02
16	47.90

****
Significance of Effect Report
****

Label	Half_Effect	Abs_Effect	CL_Upper	Rank	95%Sig
a	-0.400000000	0.40000000	0.72215383	5	no
b	-2.110000000	2.11000000	2.40116840	15	no
c	1.855000000	1.85500000	1.92935935	14	no
d	0.505000000	0.50500000	0.89968325	7	no
ab	0.455000000	0.45500000	0.80882073	6	no
ac	-1.245000000	1.24500000	1.67551509	13	no
ad	-0.290000000	0.29000000	0.47792967	2	no
bc	-0.400000000	0.40000000	0.63864843	4	no
bd	-0.590000000	0.59000000	0.99608816	8	no
cd	0.745000000	0.74500000	1.21372274	10	no
abc	0.600000000	0.60000000	1.09987834	9	no
abd	0.360000000	0.36000000	0.55747331	3	no
acd	0.200000000	0.20000000	0.39940279	1	no
bcd	-0.790000000	0.79000000	1.49108302	12	no
abcd	0.760000000	0.76000000	1.34176715	11	no

Lawson,Grimshaw,and Burt Rn=1 with 1.201 95% Percent.

****
Corrected Data Report
****

N	Response	Corr.Resp.	Outlier
1	47.46000000	47.46000000	no
2	49.62000000	49.62000000	no
3	43.13000000	43.13000000	no
4	46.31000000	46.31000000	no
5	51.47000000	51.47000000	no
6	48.49000000	48.49000000	no
7	49.34000000	49.34000000	no
8	46.10000000	46.10000000	no
9	46.76000000	46.76000000	no
10	48.56000000	48.56000000	no
11	44.83000000	44.83000000	no
12	44.45000000	44.45000000	no
13	59.15000000	52.75000000	yes
14	51.33000000	51.33000000	no
15	47.02000000	47.02000000	no
16	47.90000000	47.90000000	no

Perc.	Gap	PSE	Stand.Gap	Critical	Signif.
50th	0.4900000	0.8850000	3.3532267	1.7880000	yes
99th	0.4900000	0.2250000	13.189358	5.1000000	yes

3. Example 3 of submission:

```
print "Defines Linear and Quadratic Orthogonal Main Effects Contrats";
nobs = 18; neff = 17;
dat1 = [ 1 1 1 1 1 1 1 1  97.94 ,
         1 1 2 2 2 2 2 2  83.40 ,
         1 1 3 3 3 3 3 3  95.88 ,
         1 2 1 1 2 2 3 3  88.86 ,
         1 2 2 2 3 3 1 1 106.58 ,
         1 2 3 3 1 1 2 2  89.57 ,
         1 3 1 2 1 3 2 3  91.98 ,
         1 3 2 3 2 1 3 1  98.41 ,
         1 3 3 1 3 2 1 2  87.56 ,
         2 1 1 3 3 2 2 1  88.11 ,
         2 1 2 1 1 3 3 2  83.81 ,
         2 1 3 2 2 1 1 3  98.27 ,
         2 2 1 2 3 1 3 2 115.52 ,
         2 2 2 3 1 2 1 3  94.89 ,
         2 2 3 1 2 3 2 1  94.70 ,
         2 3 1 3 2 3 1 2 121.62 ,
         2 3 2 1 3 1 2 3  93.86 ,
         2 3 3 2 1 2 3 1  96.10 ];
 cnam = [" A G B C D E F H Y "];
```

```
 dat1 = cname(dat1,cnam);
 yvec = dat1[,9]; print "Dat1=",dat1;


 effnam = [" Al Gl Gq Bl Bq Cl Cq Dl Dq El
            Eq Fl Fq Hl Hq AlGl AlGq "];
 dat2 = cons(nobs,neff,.);
 print "This will make the X'X matrix diagonal";
 for (i = 1; i <= nobs; i++) {
  dat2[i, 1] = 2*dat1[i,1] - 3;             /* Al */
  dat2[i, 2] = dat1[i,2] - 2;               /* Gl */
  dat2[i, 3] = (dat1[i,2] == 2) ? -2 : 1;   /* Gq */
  dat2[i, 4] = dat1[i,3] - 2;               /* Bl */
  dat2[i, 5] = (dat1[i,3] == 2) ? -2 : 1;   /* Bq */
  dat2[i, 6] = dat1[i,4] - 2;               /* Cl */
  dat2[i, 7] = (dat1[i,4] == 2) ? -2 : 1;   /* Cq */
  dat2[i, 8] = dat1[i,5] - 2;               /* Dl */
  dat2[i, 9] = (dat1[i,5] == 2) ? -2 : 1;   /* Dq */
  dat2[i,10] = dat1[i,6] - 2;               /* El */
  dat2[i,11] = (dat1[i,6] == 2) ? -2 : 1;   /* Eq */
  dat2[i,12] = dat1[i,7] - 2;               /* Fl */
  dat2[i,13] = (dat1[i,7] == 2) ? -2 : 1;   /* Fq */
  dat2[i,14] = dat1[i,8] - 2;               /* Hl */
  dat2[i,15] = (dat1[i,8] == 2) ? -2 : 1;   /* Hq */
  /* Defines Interactions to Make X1 Full Rank; */
  dat2[i,16] = dat2[i,1] * dat2[i,2];  /* ALGL=AL*GL; */
  dat2[i,17] = dat2[i,1] * dat2[i,3];  /* ALGQ=AL*GQ; */
}
print "Dat2=", dat2;


ps = "X'X matrix will be equal";
print "Standardize Contrasts so that diagonals of", ps;
fac1 = 1. / sqrt(2. * 9.);
fac2 = 1. / sqrt(2. * 6.);
fac3 = 1. / sqrt(6. * 6.);
xdat = cons(nobs,neff,.);
xdat[,1] = fac1 * dat2[,1];
for (i = 2; i < 17; i+=2) {
  xdat[,i  ] = fac2 * dat2[,i  ];
  xdat[,i+1] = fac3 * dat2[,i+1];
}
xdat = cname(xdat,effnam);
print "Standardized Xdat=",xdat;


optn = [   2 ];  /* ipri */
```

```
< effrep,outrep,cordat > = urd1out(xdat,yvec,optn);

 ****************************************************************
 Analysis of Unreplicated 2^k and 2^(k-p) Designs (Lawson, 2007)
 ****************************************************************

    Number of Observations . . . . . .        18
    Number of Effects. . . . . .        17
```

Input Design Matrix

	Al	Gl	Gq	Bl	Bq	Cl	Cq	Dl	Dq
1	-0.236	-0.289	0.167	-0.289	0.167	-0.289	0.167	-0.289	0.167
2	-0.236	-0.289	0.167	0.000	-0.333	0.000	-0.333	0.000	-0.333
3	-0.236	-0.289	0.167	0.289	0.167	0.289	0.167	0.289	0.167
4	-0.236	0.000	-0.333	-0.289	0.167	-0.289	0.167	0.000	-0.333
5	-0.236	0.000	-0.333	0.000	-0.333	0.000	-0.333	0.289	0.167
6	-0.236	0.000	-0.333	0.289	0.167	0.289	0.167	-0.289	0.167
7	-0.236	0.289	0.167	-0.289	0.167	0.000	-0.333	-0.289	0.167
8	-0.236	0.289	0.167	0.000	-0.333	0.289	0.167	0.000	-0.333
9	-0.236	0.289	0.167	0.289	0.167	-0.289	0.167	0.289	0.167
10	0.236	-0.289	0.167	-0.289	0.167	0.289	0.167	0.289	0.167
11	0.236	-0.289	0.167	0.000	-0.333	-0.289	0.167	-0.289	0.167
12	0.236	-0.289	0.167	0.289	0.167	0.000	-0.333	0.000	-0.333
13	0.236	0.000	-0.333	-0.289	0.167	0.000	-0.333	0.289	0.167
14	0.236	0.000	-0.333	0.000	-0.333	0.289	0.167	-0.289	0.167
15	0.236	0.000	-0.333	0.289	0.167	-0.289	0.167	0.000	-0.333
16	0.236	0.289	0.167	-0.289	0.167	0.289	0.167	0.000	-0.333
17	0.236	0.289	0.167	0.000	-0.333	-0.289	0.167	0.289	0.167
18	0.236	0.289	0.167	0.289	0.167	0.000	-0.333	-0.289	0.167

Input Design Matrix

	El	Eq	Fl	Fq	Hl	Hq	AlGl	AlGq
1	-0.289	0.167	-0.289	0.167	-0.289	0.167	0.289	-0.167
2	0.000	-0.333	0.000	-0.333	0.000	-0.333	0.289	-0.167
3	0.289	0.167	0.289	0.167	0.289	0.167	0.289	-0.167
4	0.000	-0.333	0.289	0.167	0.289	0.167	0.000	0.333
5	0.289	0.167	-0.289	0.167	-0.289	0.167	0.000	0.333
6	-0.289	0.167	0.000	-0.333	0.000	-0.333	0.000	0.333
7	0.289	0.167	0.000	-0.333	0.289	0.167	-0.289	-0.167
8	-0.289	0.167	0.289	0.167	-0.289	0.167	-0.289	-0.167
9	0.000	-0.333	-0.289	0.167	0.000	-0.333	-0.289	-0.167
10	0.000	-0.333	0.000	-0.333	-0.289	0.167	-0.289	0.167

11	0.289	0.167	0.289	0.167	0.000	-0.333	-0.289	0.167
12	-0.289	0.167	-0.289	0.167	0.289	0.167	-0.289	0.167
13	-0.289	0.167	0.289	0.167	0.000	-0.333	0.000	-0.333
14	0.000	-0.333	-0.289	0.167	0.289	0.167	0.000	-0.333
15	0.289	0.167	0.000	-0.333	-0.289	0.167	0.000	-0.333
16	0.289	0.167	-0.289	0.167	0.000	-0.333	0.289	0.167
17	-0.289	0.167	0.000	-0.333	0.289	0.167	0.289	0.167
18	0.000	-0.333	0.289	0.167	-0.289	0.167	0.289	0.167

Input Response Vector

	1
1	97.94
2	83.40
3	95.88
4	88.86
5	106.6
6	89.57
7	91.98
8	98.41
9	87.56
10	88.11
11	83.81
12	98.27
13	115.5
14	94.89
15	94.70
16	121.6
17	93.86
18	96.10

****
Significance of Effect Report
****

Label	Half_Effect	Abs_Effect	CL_Upper	Rank	95%Sig
Al	11.00729556	11.0072956	13.0848080	11	no
Gl	12.15899667	12.1589967	19.0494817	15	no
Gq	-7.216666667	7.21666667	8.41923908	6	no
Bl	-12.10992190	12.1099219	17.1266856	14	no
Bq	7.368333333	7.36833333	9.25291187	7	no
Cl	12.05218687	12.0521869	15.5773027	13	no
Cq	-8.081666667	8.08166667	10.1239829	8	no
Dl	9.589787971	9.58978797	12.0234517	10	no
Dq	-4.786666667	4.78666667	6.83284397	4	no

El	0.288675135	0.28867513	4.56943981	1	no
Eq	18.38333333	18.3833333	26.6818495	17	no
Fl	-8.163732806	8.16373281	11.0428885	9	no
Fq	17.03333333	17.0333333	21.7090453	16	no
Hl	-5.225019936	5.22501994	7.61477213	5	no
Hq	-2.896666667	2.89666667	6.06783619	3	no
AlGl	11.73753097	11.7375310	14.2548883	12	no
AlGq	-2.266666667	2.26666667	5.31484600	2	no

Lawson,Grimshaw,and Burt Rn=1 with 1.178 95% Percent.

****
Corrected Data Report
****

N	Response	Corr.Resp.	Outlier
1	97.94000000	97.94000000	no
2	83.40000000	83.40000000	no
3	95.88000000	95.88000000	no
4	88.86000000	88.86000000	no
5	106.5800000	106.5800000	no
6	89.57000000	89.57000000	no
7	91.98000000	91.98000000	no
8	98.41000000	98.41000000	no
9	87.56000000	87.56000000	no
10	88.11000000	88.11000000	no
11	83.81000000	83.81000000	no
12	98.27000000	98.27000000	no
13	115.5200000	115.5200000	no
14	94.89000000	94.89000000	no
15	94.70000000	94.70000000	no
16	121.6200000	29.03181091	yes
17	93.86000000	93.86000000	no
18	96.10000000	96.10000000	no

Perc.	Gap	PSE	Stand.Gap	Critical	Signif.
50th	2.5553418	12.245599	1.4309717	1.7880000	no
99th	2.5553418	22.013582	0.7960134	5.1000000	no