CMAT© Newsletter: December 2013

Wolfgang M. Hartmann

December 2013

Contents

1 General Remarks **2**
1.1 New Functions 2
1.2 Fixed Bugs 2

2 Modifications of Features **3**

3 Extensions to the Language **3**

4 Extensions to Various Functions **4**
4.1 Extensions of `cdfmv()` Function 4
4.2 Extensions of `glmod()` Function 10
4.3 Extensions of `ttest()` Function 64

5 New Developments **65**
5.1 The `icdfmv()` Function 65
5.2 The `mahalanobis()` Function 67
5.3 The `multcomp()` Function 72
5.4 The `padjust()` Function 89
5.5 The `pdfmv()` Function 93
5.6 The `wilcox()` Function 96

6 The Bibliography **102**

1 General Remarks

A number of bugs were fixed. Multiple testing (comparison of means) has been added, especially for 1-way ANOVA (new function `multcomp()`). Most of the examples shown in the book by Bretz et al. (2011) can now be computed using the `multcomp()` and `glmod()` functions.
The output arguments of functions `reg()`, `glmod()`, `lrforw()`, and `lrallv()` were made more similar. The options input for function `glmod()` has been changed from vector input to the common 2-column matrix input. Whenever multiple testing is required with the `glmod()` function, a list of two option matrices is required. One entry of the list must be the common 2-column options matrix of the `glmod()` function, the other should be the 2-column options matrix as it is specified with the `multcomp()` function. Additional output was added to `glmod()` too.

1.1 New Functions

icdfmv computes the inverse cumulative distribution (percent point) function of the multivariate normal and t distribution

mahalanobis compute the Mahalanobis matrix or only its diagonal of an $m \times n$ matrix $\mathbf{X}$

multcomp performs a number of parametric and nonparametric tests for linear hypotheses among means of $K > 2$ samples, e.g. Tukey, Dunnett, Kruskal-Wallis

padjust computes adjusted multivariate p values (Bonferroni, Holm, Hommel, Hochberg, Benjamini & Hochberg, Benjamini & Yekutieli, Sidak, Blanchard & Roquain)

pdfmv computes density of multivariate normal and t distribution

wilcox performs the one and two sample *Wilcoxon Signed Rank Test* (*Mann-Whitney Test*) and the *Wilcoxon Rank Sum Test*.

1.2 Fixed Bugs

A large number of bugs were fixed. The most important concerns all functions with model specifications including class variables. For effects with class variables I had changed the parametrization without changing the output correspondingly. Normally, like in SAS PROC GLM or in the *lm* function of R the coefficient of the last level is set to zero. I had changed that so that the first level of each class variable was set to zero. Internally I did obtain with that the correct predicted values $\hat{y}$ and goodness of fit indices, but the coefficients had different values and were also printed in the wrong locations.
Bugs in reporting the AIC, BIC, and SBC fit criteria in functions `reg()`, `glmod()`, `lrforw()`, and `lrallv()` have been fixed.

2 Modifications of Features

3 Extensions to the Language

4 Extensions to Various Functions

4.1 Extensions of `cdfmv()` Function

The function was extended by an additional input argument (mean vector μ for normal and vector δ of noncentralities for the t distribution. The newer Genz-Bretz method ([15]) was added (which acts now as the default method) which is in its implementation very similar to that which is available in CRAN. Due to the many changes we simply show here the new documentation:

---

p = cdfmv(dist,ilim<,mu<,cov<,par>>>)	dist="norm"|"t"

Purpose: Computes multivariate integral of probability function (multivariate cumulative density function of a specific distribution function). Various algorithms by A. Genz (1992, [8], 2000, [10], Genz & Bretz, 1999, [12]) and the newer developments by Genz & Bretz (2009, [15]) are used. Note, that the newer Genz-Bretz algorithm greatly outperforms the older algorithms.

Input: **dist** The first input argument is a string specifying the distribution. Until now, only the multivariate normal and t distributions are supported.

ilim The second argument is a $n \times 2$ matrix specifying the integration limits. The first column specifies the lower range, a missing value means $-\infty$. The second column specifies the upper range, a missing value here means $+\infty$.

mu The third argument specifies the n vector

- of means for the normal distribution
- of noncentralities for the t distribution

cov The fourth argument specifies the $n \times n$ covariance matrix of the $n$variate distribution. The input covariance matrix must be nonsingular (positive definite).

par The (optional) fifth argument may be used to set additional parameters and it's contents depends on the distribution specified in the first argument.

For `dist="norm"` the content of the parameter vector `par` is:

par[1,...,4] currently not used; since `par[1]` specifies the degrees of freedom for the t distribution, `par[1]` should be missing or zero for the normal distribution.

par[5] integration method:

par[5]=0 specific algorithm for only two or three dimensions
par[5]=1 Genz-Bretz algorithm (2009, [15]) (default)
par[5]=10 RANMVN: using Monte-Carlo algorithm
par[5]=11 SADMVN: uses adaptive integration

par[5]=12 KROMVN: uses lattice rule integration
par[5]=13 SPHMVN: uses Deak's (1980) method (this method does not work well and should not be used)

par[6] maximum number of function evaluations, default is

par[5]=1 par[6]=25000
par[5]>10 par[6]=10000

par[7] absolute error tolerance, default is

par[5]=1 par[7]=0.001
par[5]>10 par[7]=0.

par[8] relative error tolerance, default is

par[5]=1 par[8]=0.
par[5]>10 par[8]=0.005

For `dist="t"` the content of the parameter vector `par` is:

par[1] the degrees of freedom

par[2,3,4] currently not used;

par[5] integration method:

par[5]=0 specific algorithm for only two or three dimensions
par[5]=1 Genz-Bretz algorithm (2009, [15]) (default)
par[5]=10 RANMVT: using Monte-Carlo algorithm
par[5]=11 SADMVT: uses adaptive integration
par[5]=12 KROMVT: uses lattice rule integration

par[6] maximum number of function evaluations, default is

par[5]=1 par[6]=25000
par[5]>10 par[6]=10000

par[7] absolute error tolerance, default is

par[5]=1 par[7]=0.001
par[5]>10 par[7]=0.

par[8] relative error tolerance, default is

par[5]=1 par[8]=0.
par[5]>10 par[8]=0.005

Note, that for `sopt="t"` the degrees of freedom must be specified. In this case, the fourth input argument `par` must be set, at least as a scalar.

Output: The returned 2 vector `p` contains the estimated value of the integral (probability) and the estimated error (with 99 % confidence level).

Restrictions: 1. A missing value for p is returned if the inputs contains string or complex data.

2. A missing value is returned also if the matrix `corr` contains any missing values, is not symmetric or not positive definte.

Relationships: cdf(), cdf23(), binorm(), pdf(), pdfmv(), icdfmv(), erf(), beta(), gamma(), betamis(), binomis(), chimis(), fmis(), gammamis(), nbinmis(), normis(), poismis(), tmis()

Examples: 1. Example by Genz (JCGS, 1992):

```
lim = [ .   .  .,
        1. 4. 2. ];
r1= 3. / 5.; r2= 1. / 3.; r3= 11. / 15.;
mu   = cons(3);
corr = [ 1.  r1  r2,
         r1  1.  r3,
         r2  r3  1.];


par = [ . . . . 0 25000 5.e-5 5.e-5 ];
p1 = cdfmv("norm",lim,mu,corr,par);
par[5] = 1;
p2 = cdfmv("norm",lim,mu,corr,par);
par[5] = 10;
p4 = cdfmv("norm",lim,mu,corr,par);
par[5] = 11;
p5 = cdfmv("norm",lim,mu,corr,par);
par[5] = 12;
p6 = cdfmv("norm",lim,mu,corr,par);
all = p1 -> p2 -> p4 -> p5 -> p6;
cnam = [" BITRI GENZB RANMVN SADMVN KROMVN "];
all = cname(all,cnam);
print all;
```

```
  |     BITRI     GENZB    RANMVN    SADMVN    KROMVN
-------------------------------------------------------
1 |  -0.82798   0.82798   0.82802   0.82798   0.82799
2 |     .         3e-007   0.00004   0.00000    4e-007
```

2. Chen Multivariate Normal Problem:

```
print "Genz MVN: MultiVariate Normal CDF: Normal Distribution";
lim = [ 0.  0.      1.7817  1.4755   .  ,
          .  1.5198   .       .      1.5949 ];
mu  = cons(5);
corr= [    1.0 -.707107   0.0   0.0   0.0 ,
      -.707107      1.0   0.5   0.5   0.5 ,
```

```
            0.0      0.5   1.0   0.5   0.5 ,
            0.0      0.5   0.5   1.0   0.5 ,
            0.0      0.5   0.5   0.5   1.0 ];
 k = 5000 * 125;
 par = [ . . . . 1 k 5.e-5 0. ];
 vol = cdfmv("norm",lim,corr,par);


 for (i = 10; i <= 13; i++) {
   par[5] = i;
   v2 = cdfmv("norm",lim,corr,par);
   vol = vol -> v2;
 }
cnam = [" GENZB RANMVN SADMVN KROMVN  SPHMVN "];
p = cname(vol,cnam);
print "MVN Results for Chen Problem:",p;


MVN Results for Chen Problem:
   |     GENZB    RANMVN    SADMVN    KROMVN    SPHMVN
---------------------------------------------------------
 1 |   0.00286   0.00288   0.00287   0.00286   0.00285
 2 |   0.00001   0.00004   0.00003   0.00001   0.00007
```

3. Example by Genz for Multivariate t Density Estimation:

```
print "Genz MVT: MultiVariate Normal CDF: T Distribution";
print "Try: ndf=10, 20, 30, 40";
lim = [  .   .   .   .   . ,
        2.  2.  2.  2.  2. ];
mu  = cons(5);;
corr = cons(5,5,.75);
for (j = 1; j <= 5; j++) corr[j,j] = 1.;
k = 25000;
par = [ 10 . . . 1  k  1.e-3 5.e-3 ];
vol = cdfmv("t",lim,mu,corr,par);


for (df = 10; df <= 40; df += 10) {
  par[1] = df; par[5] = 1;
  vol = cdfmv("t",lim,mu,corr,par);
  for (i = 10; i <= 12; i++) {
    par[5] = i;
    v2 = cdfmv("t",lim,mu,corr,par);
    vol = vol -> v2;
  }
  cnam = [" GENZB RANMVT SADMVT KROMVT "];
```

```
  p = cname(vol,cnam);
  print "MVT Result for df=", df, p;
}
```

```
MVT Result for df= 10
   |     GENZB    RANMVT    SADMVT    KROMVT
-------------------------------------------
 1 |   0.90896   0.90724   0.90811   0.90900
 2 |   0.00059   0.00447   0.00201   0.00065
```

```
MVT Result for df= 20
   |     GENZB    RANMVT    SADMVT    KROMVT
-------------------------------------------
 1 |   0.92221   0.92051   0.92175   0.92252
 2 |   0.00044   0.00445   0.00184   0.00067
```

```
MVT Result for df= 30
   |     GENZB    RANMVT    SADMVT    KROMVT
-------------------------------------------
 1 |   0.92701   0.92798   0.92642   0.92692
 2 |   0.00086   0.00351   0.00178   0.00082
```

```
MVT Result for df= 40
   |     GENZB    RANMVT    SADMVT    KROMVT
-------------------------------------------
 1 |   0.92947   0.93100   0.92878   0.92944
 2 |   0.00051   0.00429   0.00175   0.00065
```

4. Example by Edwards and Berry (1987), see CRAN document:

```
print "Example by Edwards and Berry, 1987";
n = [ 26 24 20 33 32 ];
V = diag(1. / n);
C = [  1  1  1  0  0,
      -1  0  0  1  0,
       0 -1  0  0  1,
       0  0  0 -1 -1,
       0  0 -1  0  0 ];
/* scale matrix */
cv = C * V * C‘; /* print "CV=", cv; */
dv = 1. / sqrt(diag(cv)); /* print "DV=", dv; */
cr = dv‘* cv * dv; print "CR=", cr;
```

```
function myfct(q) global(cr) {
  n = ncol(cr); ndf = 130; alfa = .95;
  dlta = cons(n);
  lim = cons(n,1,-q) -> cons(n,1,q); /* print "Lim=", lim; */
  par = [ ndf . . . 1 25000 1.e-3 1.e-3 ]; /* GenzBretz */
  prob = cdfmv("t",lim,dlta,cr,par); /* print "prob=", prob; */
  res = prob[1] - alfa;
  return(res);
}


q0 = 2.5; ab = [ 1., 5.];
f0 = myfct(q0);
z1 = fzero(myfct,q0,ab);
print "CRAN: 2.560704, or 2.561037, Zero=",z1;
print "CRAN takes 11 iterations, error=.00023146, est.prec=6.1035e-5";


 Compute Zero of Univariate Function
 Method by Kahane, Moler, and Nash (1989)
 Max Function Calls      1000
 Relative Error   1.0000e-006       Absolute Error   1.0000e-020
 Successful Termination
 Left Bracket     1.000000000       Right Bracket    5.000000000
 Zero Estimate    2.571363753       N Function Calls          17


CRAN: 2.560704, or 2.561037, Zero= 2.5714
```

4.2 Extensions of `glmod()` Function

A large number of bugs were fixed. For effects with class variables I had badly changed the parametrization without changing the output correspondingly. In addition, some bugs in reporting the AIC, BIC, and SBC fit criteria have been fixed. The options input was changed from vector input to the common 2-column matrix input. Additional output was added to `glmod()` too. Multiple testing (comparison of means) has been addedd, especially for 1-way ANOVA with the new function `multcomp()`, and were also added to the already existing `glmod()` function which is related to some features of SAS PROC GLM.
Due to the many changes we simply show here the new documentation:

---

gof = glmod(data,model<,optn,<,class<,cont>>>)

<gof,parm,ase,conf,cov,typ1,typ3,resi> = glmod(data,model<,optn,<,class<,cont>>>)

Purpose: This function solves the general linear model

$$\|y - \mathbf{X}\beta\|_2 \longrightarrow \min$$

i.e. computes the least-squares parameter estimates (L_2 fit) β where the given data $\mathbf{X}$ and y can contain interval or nominal (CLASS) scaled variables. If there are more than one, i.e. $ny > 1$ responses, separate analyses are performed for each column of y.

When the response of the model specification is interval scaled, the multiple test (comparison) methods from function `multcomp()` can be applied to the levels of all class effects in the model. That, however, requires the specification of a list of two 2-column options matrices, one for the `glmod()` and the other for the (implicitly run) `multcomp()` function. All types of contrast matrices, including input, und all types of p value adjustment are permitted as can be used with the `multcomp()` function. See the document for the `multcomp()` function for more details on the types of contrasts and p value adjustments.

As the examples show, a wide range of problems (like ANOVA or linear and polynomial regression models) can be solved using this function.

Input: **data** : It is assumed that the rows of the $N \times n$ input data matrix correspond to observations (cases) and the columns correspond to interval or categorical variables. Categorical variables may be of integer or string data type. Using the second and third input argument, specific columns of these data are selected for

- a number of independend (right-hand side) variables x_i
- at least one dependend (left-hand side) response variable y
- a WEIGHT, FREQ (frequency), or OFFSET variable.

The data may contain missing values.

model : The analysis model is specified in form of a string containing column numbers for variables. The model string specifies which variables (columns) are independend (covariates) and which are dependend (response) and is of the form

$$Y_1 \ldots Y_r = X_{effect_1} \ldots X_{effect_n}$$

where X_{effect_i} may be one of the following:

- a single variable x_i is an effect
- effects that are interactions among multiple x variables, separated by the $*$ operator, $x_i * \ldots * x_l$
- nested effects: that are single variables or interactions followed by a list of variables inside parentheses, $(x_i \ldots x_l)$. Note that only categorical variables without interactions can be listed inside parentheses.

The parameters of the effects are estimated for each response variable Y_i separately. There are a number of additional features that can make it easier to use such kind of model specifications:

- You may use the | (bar) operator, for the shorter notation of `model = "y = a b a*b"` by `model="y = a | b"`. Note, that using more than two colon operators in one term may create a large number of effects, e.g. `"a|b|c|d"` creates the following 15 effects: `"a b a*b c a*c b*c a*b*c d a*d b*d c*d a*b*d a*c*d b*c*d a*b*c*d"` For limiting the maximum number of variables in the effects, the @ operator with an integer can be used, e.g. `"a|b|c|d@2"` creates only 10 effects: `"a b a*b c a*c b*c d a*d b*d c*d"`.
- You may use the colon operator to express a successive number single variable effects, e.g. `"y= 1 : 3 * 5"` is the same as `"y= 1 2 3*5"`.
- Instead of specifying the entire model in one long string variable you may use a vector of strings. However, each effect must be entirely contained in one vector entry and cannot be continued with the next entry. E.g. the above example could be split into the vector `v` as `v[1]="y="; v[2]="1 2"; v[3]="3*5";` . Note, that the order of the effect listing is important for stepwise regression and Type I estimates.

The syntax of the `model` string argument is the same as for the `glim()` function.

optn This argument must be specified in one of the following two alternative forms:

1. In form of one 2-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.

2. In form of a list with two entries. One must be the common 2-column matrix specifying the options for `glmod()` and the other specifying the options for the `multcomp()` part. The name of the list entry which contains the `glmod()` options must start with an `opt` prefix. The name of the list entry which contains the `multcomp()` options must start with an `mean` prefix (in accordance to the MEANS statement of SAS/STAT PROC GLMOD).

See the two tables below for the content of the two matrices. If there are multiple tests of the level estimates of CLASS effects specified, i.e. a list input of options, then the two 2-colum matrices may contain a few more spcifications than normally needed. The tables below indicate those additional options.

class : This optional argument should be either missing value, or an integer scalar or a vector of integers naming the number of columns in the data set which are considered categorical (nominal scaled) variables.

contr : This optional argument permits the specification of an $K \times C$ contrast matrix, where C is the number of contrasts and K depends on the `"eff"` option,

- for `"parm"`: the number of all model parameters
- otherwise: the number of levels of the specified CLASS effect

See also the `multcomp()` function.

Options Matrix Argument for the common GLMOD part: The option argument is specified in form of a 2-column matrix:

Option Name	Second Column	Meaning
"alpha"	real	the significance level for confidence intervals; default is $\alpha = 0.05$
"desi"	char	form of design coding: **"ful"** the whole rank-deficient design (GLM) **"ran"** the reduced full-rank design (this is default)
"freq"	int	specifies the column number for a FREQ variable
"incl"	int	valid only for stepwise model selection: an integer n that specifies that the first n effects must be included in any model (default is 0)
"maxstep"	int	maximum number of steps in a stepwise process
"nobstat"		do not print observationwise statistics
"noint"		decides whether an intercept is included (default) in the model or not. A nonzero value means that the intercept is not icluded
"nomis"		controls the treatment of missing values in the data; a nonzero value means that an observations is dropped from the analysis if it contains a missing value in one of the variables used in the analysis. For more information on the treatment of missing values see the *Restrictions* section.
"nopr"		no printed output
"notype1"		do not compute Type I estimates
"notype3"		do not compute Type III estimates
"outf"		path of output file of mult. comparison
"pall"		sets "print" to largest amount
"pcor"		print asymptotic correlation matrix of estimates
"pcov"		print asymptotic covariance matrix of estimates
"print"	int	controls the amount of printed output; a zero means, that no printed output is created; for larger values stepwise more output is added. Default is 2.

Option Name	Second Column	Meaning
"rule"	char	valid only for stepwise model selection: **'n'** : there is NORULE for inclusion or removing effects into/from the model. **'s'** : a SINGLE effect is included in the model only if it's lower order effects are already in the model; it is removed only if it's higher order effects are not in the model. **'m'** : MULTIPLE effects may be included but not violating the hierarchical rule is preserved (e.g. if a and b and a*b are not in the model, only a alone, b alone, or a, b, and a*b together can be included).
"selec"	char	perform a stepwise model selection: **'n'** : full (this is the default) **'f'** : forward selection **'b'** : backward selection **'s'** : stepwise selection
"seq"		valid only for stepwise model selection: specifies that effects are included into or removed from the model only in the sequential order of their listing in the MODEL formulation.
"slentry"	real	real criterion for entering an effect into the model used in stepwise forward process (default is SLENTRY=.05); determines the end of the forward selection process
"slstay"	real	real criterion for keeping an effect into the model used in stepwise forward process (default is SLSTAY=.05); determines the end of the backard selection process
"start"	int	valid only for stepwise model selection: an integer n specifying that the first n effects should be included in the model at the start of the stepwise process
"stop"	int	valid only for stepwise model selection: an integer n specifying the maximum (for forward) or minimum (for backward) number of effects to be included into the final model
"seed"	int	seed of random generator, default is time of day
"sing"	real	singularity threshold, default=1.e-8
"weight" | "wgt"	int	specifies the column number for a WEIGHT variable

Options Matrix Argument for the MULTCOMP part: The option argument is specified in form of a 2-column matrix:

Option Name	Second Column	Meaning
"alpha"	real	level for confidence intervals, def=.05
"conf"	real	confidence level, def=0.95 = $1-\alpha$
"contpop"	int	number of control sample for Dunnett's contrast
"ctype"	string or int	name of the method (contrast matrix)
	"inp" 0	input contrast matrix
	"dun" 10	Dunnett (1955) ($K-1$ mult. treatments vs control)
	"tuk" 11	Tukey (Bofinger, 1985; Hayter, 1990) (($K*(K-1))/2$ all pairwise)
	"seq" 12	Sequence (Liu et al, 1999)
	"ave" 13	AVE (Bretz, Hothorn, & Westfall, 2011)
	"cha" 14	Changepoint (Hirotsu, 1997, 1999)
	"wil" 15	Williams (Bretz, 1999)
	"mar" 16	Marcus (Bretz, 1999)
	"mcd" 17	McDermott and Mudholker (1993)
	"umb" 18	Umbrella-Williams (Bretz and Hothorn)
	"gme" 19	Grand Mean (Bretz, Hothorn, & Westfall, 2011)
	"snk" 40	Student, Newman, and Keul test
	"rgw" 41	Ryan-Einot and Gabriel-Welsch test
	"gao" 42	(nonparametric) Gao et al. (2008)
	"dtk" 43	Dunnett-Tukey-Kramer test
	"hsd" 44	Tukey Honest Significant Difference
"eff"	"parm"	compute multiple comparison with contrast matrices relating to all parameters
	"class"	compute multiple comparison for all CLASS effects of the model
	int	compute multiple comparison only for effect with that number
"fixcon"	int	
"ndf"	int	
"ncon"	int	
"noadj"		do not compute adjusted p's
"noci"		do not compute CIs
"noprint"		no printed output
"pall"		print all of: pcon,pcov,pcor,pest,pglob,ppval
"pcon"		print contrast matrix
"pcor"		print correlation matrix of estimates
"pcov"		print covariance matrix of estimates
"peakpop"	int	

Option Name	Second Column	Meaning
"pest"		print coefficition table with CI's
"pglob"		print global F and χ^2 fit
"poplo"	int	
"pophi"	int	
"ppval"		print p value table
"print"	int	amount of printed output, def=2
"psum"		print only the summary output for each run
"ptype"	string	name of p adjustment, def="adj"
	"adj"	"single-step" method, see Bretz et al. (2011), pp. 62
	"bon"	Bonferroni adjustment
	"hol"	Holm method (1979)
	"hom"	Hommel method (1983)
	"hoc"	Hochberg method (1988)
	"bho"	Benjamini & Hochberg (1995)
	"fdr"	false-discovery-rate, same as Benjamini & Hochberg
	"bye"	Benjamini & Yekutieli (2001)
	"bla"	Blanchard & Roquain (2008)
	"bli"	Benjamini & Liu (1999)
	"sid"	Sidak StepDown (1967)
	"fre"	"free" method, step-down Dunnett "see Bretz et al. (2011), p.79
	"sha"	Shaffer method (1986) "see Bretz et al. (2011), pp.93
	"wes"	Westfall method (1997) "see Bretz et al. (2011), pp.93
"seed"	int	seed for random generator
"sing"	real	singularity threshold, def=1.e-8
"tail"	char	test alternative: "both", "grea", "less", def="both"

Only the "ctype" and the "eff" options are new. All other options are treated the same as with the multcomp function. The specification of "eff" equal to "parm" requires an input contrast matrix

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

parm : $p \times ny$ matrix (or vector) of parameter estimates

ase : $p \times ny$ matrix (or vector) of asymptotic standard errors of parameter estimates

conf : $p \times 4 * ny$ matrix of lower and upper Wald confidence levels, t values and its probabilities

cov : $p \times p$ symmetric covariance matrix of parameter estimates

typ1 : $neff \times 5 * ny$ matrix of Type I statistics

typ3 : $neff \times 5 * ny$ matrix of Type III statistics

resi : $N \times 6 * ny$ containing:

1. input data of response y
2. input data of frequency times weight variable
3. residual $y - \hat{y}$
4. predicted value $\hat{y}$
5. lower Wald CI of predicted value
6. lower Wald CI of predicted value

In addition to those return arguments, the results of multiple comparison can be written into a file specified by the `"outf"` option.

Restrictions:

1. String data in the input matrix *data* are permitted only if the column is specified as a CLASS variable or if the column is not included in the model at all.
2. Missing values are returned if the input matrix *data* contains complex data.
3. The treatment of missing values in the `data` input matrix depends on whether option `optn[3]` is specified or not and on the type of variable where the missing value occurs:
   (a) If the column in which the missing value occurs is not included as a variable in the model, the entire column and therefore any missing value in it are ignored.
   (b) If the NOMISS option `optn[3]=1` is specified and the variable is used in the model (either as an independend y, dependend x, a WEIGHT, FREQ, or OFFSET variable) then the corresponding entire row (observation) is skipped in the analysis reducing the number of observations (degrees of freedom).
   (c) If the NOMISS option `optn[3]=0` is not specified, the type of variable decides how the value is treated:
   - if the column corresponds to a independend (x) CLASS variable, an additional category is assigned to the missing values.
   - if the column corresponds to a independend (x) non-CLASS variable, the mean of the variable is imputed for the missing value.
   - if the column corresponds to a dependend (y) CLASS variable, an additional category is assigned to the missing values.
   - if the column corresponds to a dependend (y) non-CLASS variable, the entire row (observation) is skipped in the analysis reducing the number of observations (degrees of freedom) in the analysis.

Relationships: glm(), reg(), lrforw(), lrallv(), multcomp(), glim()

Examples: To save space we only show the specification of some examples here. For more information, especially the printed output you are referred to the *Details* chapter, see page **??**.

1. Analysis of Unbalanced 2-by-2 Factorial: (see [**?**], PROC GLM, p. 898) This is an ANOVA design which has only one observation in cell (a_2, b_2) but all other cells have 2 observations:

```
a = [ 'a1' 'b1'  12,
      'a1' 'b1'  14,
      'a1' 'b2'  11,
      'a1' 'b2'   9,
      'a2' 'b1'  20,
      'a2' 'b1'  18,
      'a2' 'b2'  17 ];


clas = [ 1 2 ];
model = "3 = 1 2 1*2";
optn = [ "print"       5 ];
< gof,parm,ase,conf,cov,typ1,typ3,resi > = glmod(a,model,optn,clas);
```

```
                    **********
                    Parameter Information
                    **********

          Parameter | Meaning
          ----------------------------------------
                  1 | Intercept
                  2 | C1a1
          No Param. | C1a2
                  4 | C2b1
          No Param. | C2b2
                  6 | C1_a1*C2_b1
          No Param. | C1_a1*C2_b2
          No Param. | C1_a2*C2_b1
          No Param. | C1_a2*C2_b2


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

          Variable       Value    Nobs  Proportion
          C[1]              a1       4   57.142857
                            a2       3   42.857143
```

```
          C[2]              b1       4   57.142857
                            b2       3   42.857143


                     ********************
                     Analysis of Variance
                        Full Analysis
                     ********************

     Source      DF         SSQ  MeanSquare   F_Value  Prob>F

     Model        3 91.71428571 30.57142857 15.285714  0.0253
     Error        3 6.000000000 2.000000000
     C Total      6 97.71428571


                         Goodness of Model Fit

   R-square   0.938596491           Adj R-sq   0.877192982
   MSE        2.000000000           Root MSE   1.414213562
   AIC        6.920945241           BIC        16.03205635
   SBC        6.704585837           C(p)       4.000000000


              ****************************************
              Analysis of Effects and Parameter Estimates
              ****************************************

Variable          DF  Estimate Std_Error TypeII_SS F_Value Prob>F

Intercept          1  17.00000  1.414214 289.00000 144.500 0.0012
Effect     1       1    .         .      32.666667 16.3333 0.0273
C1a1               1 -7.000000  1.732051 32.666667 16.3333 0.0273
C1a2               0         0    .         .        .       .
Effect     2       1    .         .      2.6666667 1.33333 0.3318
C2b1               1  2.000000  1.732051 2.6666667 1.33333 0.3318
C2b2               0         0    .         .        .       .
Effect     3       1    .         .      0.4000000 0.20000 0.6850
C1_a1*C2_b1        1  1.000000  2.236068 0.4000000 0.20000 0.6850
C1_a1*C2_b2        0         0    .         .        .       .
C1_a2*C2_b1        0         0    .         .        .       .
C1_a2*C2_b2        0         0    .         .        .       .


             ********************************************
             Parameter Estimates and Wald Confidence Limits
```

****

Variable	DF	Estimate	Lower_CL	UpperCI	Std_Error
Intercept	1	17.000000	14.228192	19.771808	1.4142136
Effect	1				
C1a1	1	-7.0000000	-10.394757	-3.6052428	1.7320508
C1a2	0	0	.	.	.
Effect	2				
C2b1	1	2.0000000	-1.3947572	5.3947572	1.7320508
C2b2	0	0	.	.	.
Effect	3				
C1_a1*C2_b1	1	1.0000000	-3.3826127	5.3826127	2.2360680
C1_a1*C2_b2	0	0	.	.	.
C1_a2*C2_b1	0	0	.	.	.
C1_a2*C2_b2	0	0	.	.	.

Estimated Covariance Matrix of Estimates

	Intercept	C1a1	C1a2	C2b1
Intercept	1.000000000	-1.000000000	0	-1.000000000
C1a1	-1.000000000	1.500000000	0	1.000000000
C1a2	0	0	0	0
C2b1	-1.000000000	1.000000000	0	1.500000000
C2b2	0	0	0	0
C1_b1*C2_b1	1.000000000	-1.500000000	0	-1.500000000
C1_b1*C2_b1	0	0	0	0
C1_b1*C2_b1	0	0	0	0
C1_b1*C2_b1	0	0	0	0

Estimated Covariance Matrix of Estimates

	C2b2	C1_b1*C2_b1	C1_b1*C2_b1	C1_b1*C2_b1
Intercept	0	1.000000000	0	0
C1a1	0	-1.500000000	0	0
C1a2	0	0	0	0
C2b1	0	-1.500000000	0	0
C2b2	0	0	0	0
C1_b1*C2_b1	0	2.500000000	0	0
C1_b1*C2_b1	0	0	0	0
C1_b1*C2_b1	0	0	0	0
C1_b1*C2_b1	0	0	0	0

Estimated Covariance Matrix of Estimates

```
                              C1_b1*C2_b1
  Intercept                             0
  C1a1                                  0
  C1a2                                  0
  C2b1                                  0
  C2b2                                  0
  C1_b1*C2_b1                           0
  C1_b1*C2_b1                           0
  C1_b1*C2_b1                           0
  C1_b1*C2_b1                           0


 ****************************
 Predicted Values and Residuals
 ****************************

 N      Ydat   Wgt*Frq  Residual     Ypred     Y_LCI     Y_UCI
 1  12.00000  1.000000 -1.000000  13.00000  9.817554  16.18245
 2  14.00000  1.000000  1.000000  13.00000  9.817554  16.18245
 3  11.00000  1.000000  1.000000  10.00000  6.817554  13.18245
 4  9.000000  1.000000 -1.000000  10.00000  6.817554  13.18245
 5  20.00000  1.000000  1.000000  19.00000  15.81755  22.18245
 6  18.00000  1.000000 -1.000000  19.00000  15.81755  22.18245
 7  17.00000  1.000000  1.4e-014  17.00000  12.49934  21.50066
```

The model can be expressed in a shorter form using the *bar* operator yielding the same results:

```
clas = [ 1 2 ];
model = "3 = 1|2";
optn = [ "print"       5 ];
< gof,parm,ase,conf,cov,typ1,typ3,resi >  = glmod(a,model,optn,clas);


                    **********************
                    Parameter Information
                    **********************

          Parameter | Meaning
          ----------------------------------------
                  1 | Intercept
                  2 | C1a1
          No Param. | C1a2
                  4 | C2b1
          No Param. | C2b2
```

```
        6 | C1_a1*C2_b1
No Param. | C1_a1*C2_b2
No Param. | C1_a2*C2_b1
No Param. | C1_a2*C2_b2


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

Variable       Value    Nobs  Proportion
C[1]              a1       4   57.142857
                  a2       3   42.857143

C[2]              b1       4   57.142857
                  b2       3   42.857143


                  ********************
                  Analysis of Variance
                     Full Analysis
                  ********************

  Source      DF         SSQ  MeanSquare   F_Value  Prob>F

  Model        3 91.71428571 30.57142857 15.285714  0.0253
  Error        3 6.000000000 2.000000000
  C Total      6 97.71428571


                      Goodness of Model Fit

 R-square   0.938596491           Adj R-sq   0.877192982
 MSE        2.000000000           Root MSE   1.414213562
 AIC        6.920945241           BIC        16.03205635
 SBC        6.704585837           C(p)       4.000000000


          ******************************************
          Analysis of Effects and Parameter Estimates
          ******************************************

Variable          DF  Estimate Std_Error TypeII_SS F_Value Prob>F

Intercept          1  17.00000  1.414214 289.00000 144.500 0.0012
Effect     1       1    .         .      32.666667 16.3333 0.0273
C1a1               1 -7.000000  1.732051 32.666667 16.3333 0.0273
```

C1a2		0	0	.	.	.	.
Effect	2	1	.	.	2.6666667	1.33333	0.3318
C2b1		1	2.000000	1.732051	2.6666667	1.33333	0.3318
C2b2		0	0	.	.	.	.
Effect	3	1	.	.	0.4000000	0.20000	0.6850
C1_a1*C2_b1		1	1.000000	2.236068	0.4000000	0.20000	0.6850
C1_a1*C2_b2		0	0	.	.	.	.
C1_a2*C2_b1		0	0	.	.	.	.
C1_a2*C2_b2		0	0	.	.	.	.

****
Parameter Estimates and Wald Confidence Limits
****

Variable		DF	Estimate	Lower_CL	UpperCI	Std_Error
Intercept		1	17.000000	14.228192	19.771808	1.4142136
Effect	1					
C1a1		1	-7.0000000	-10.394757	-3.6052428	1.7320508
C1a2		0	0	.	.	.
Effect	2					
C2b1		1	2.0000000	-1.3947572	5.3947572	1.7320508
C2b2		0	0	.	.	.
Effect	3					
C1_a1*C2_b1		1	1.0000000	-3.3826127	5.3826127	2.2360680
C1_a1*C2_b2		0	0	.	.	.
C1_a2*C2_b1		0	0	.	.	.
C1_a2*C2_b2		0	0	.	.	.

Estimated Covariance Matrix of Estimates

	Intercept	C1a1	C1a2	C2b1
Intercept	1.000000000	-1.000000000	0	-1.000000000
C1a1	-1.000000000	1.500000000	0	1.000000000
C1a2	0	0	0	0
C2b1	-1.000000000	1.000000000	0	1.500000000
C2b2	0	0	0	0
C1_b1*C2_b1	1.000000000	-1.500000000	0	-1.500000000
C1_b1*C2_b1	0	0	0	0
C1_b1*C2_b1	0	0	0	0
C1_b1*C2_b1	0	0	0	0

Estimated Covariance Matrix of Estimates

	C2b2	C1_b1*C2_b1	C1_b1*C2_b1	C1_b1*C2_b1
Intercept	0	1.000000000	0	0
C1a1	0	-1.500000000	0	0
C1a2	0	0	0	0
C2b1	0	-1.500000000	0	0
C2b2	0	0	0	0
C1_b1*C2_b1	0	2.500000000	0	0
C1_b1*C2_b1	0	0	0	0
C1_b1*C2_b1	0	0	0	0
C1_b1*C2_b1	0	0	0	0

Estimated Covariance Matrix of Estimates

	C1_b1*C2_b1
Intercept	0
C1a1	0
C1a2	0
C2b1	0
C2b2	0
C1_b1*C2_b1	0
C1_b1*C2_b1	0
C1_b1*C2_b1	0
C1_b1*C2_b1	0

```
*****************************
Predicted Values and Residuals
*****************************
```

N	Ydat	Wgt*Frq	Residual	Ypred	Y_LCI	Y_UCI
1	12.00000	1.000000	-1.000000	13.00000	9.817554	16.18245
2	14.00000	1.000000	1.000000	13.00000	9.817554	16.18245
3	11.00000	1.000000	1.000000	10.00000	6.817554	13.18245
4	9.000000	1.000000	-1.000000	10.00000	6.817554	13.18245
5	20.00000	1.000000	1.000000	19.00000	15.81755	22.18245
6	18.00000	1.000000	-1.000000	19.00000	15.81755	22.18245
7	17.00000	1.000000	1.4e-014	17.00000	12.49934	21.50066

2. 3 by 2 Design with Interactions: (see [**?**], PROC GLM, p. 932)

```
aby = [ 1  1  23.5,
        1  1  23.7,
        1  2  28.7,
        2  1   8.9,
        2  2   5.6,
```

```
        2  2   8.9,
        3  1  10.3,
        3  1  12.5,
        3  2  13.6,
        3  2  14.6 ];


clas = [ 1 2 ];
model = "3 = 1|2";
optn = cons(10,1,.);
optn = [ "print"       5 ];
< gof,parm,ase,conf,cov,typ1,typ3,resi >  = glmod(aby,model,optn,clas);
print "GLMOD: GOF=",gof;
```

3. Snapdragon Experiment: (see [**?**], PROC GLM: Ex. 1, pp. 965)

```
x = [ "clarion"   1  32.7,  "clarion"   2  32.3,
      "clarion"   3  31.5,  "clinton"   1  32.1,
      "clinton"   2  29.7,  "clinton"   3  29.1,
      "knox"      1  35.7,  "knox"      2  35.9,
      "knox"      3  33.1,  "o\'neill"  1  36.0,
      "o\'neill"  2  34.2,  "o\'neill"  3  31.2,
      "compost"   1  31.8,  "compost"   2  28.0,
      "compost"   3  29.2,  "wabash"    1  38.2,
      "wabash"    2  37.8,  "wabash"    3  31.9,
      "webster"   1  32.5,  "webster"   2  31.1,
      "webster"   3  29.7 ];


 model = "3 = 1 2";
 clas = [ 1 2 ];
 /* GLM Design Coding */
 optn = [ "print"       5 ,
          "desi"    "ful" ];
 < gof,parm,ase,conf,cov,typ1,typ3,resi > = glmod(x,"3 = 1 2",optn,clas);
 print "GLMOD: Parameters=",parm;
```

4. Regression with Mileage Data: (see [**?**], PROC GLM: Ex. 2, p. 969)
   Here, a quadratic model is used to determine at which speed a car achieves the highest mileage:

```
x = [  20  15.4,
       30  20.2,
       40  25.7,
       50  26.2,
       50  26.6,
```

```
      50  27.4,
      55    . ,
      60  24.8 ];


model = "2 = 1 1*1";
optn = cons(4,1,.);
optn = [ "print"       5 ];
< gof,parm,ase,conf,cov,typ1,typ3,resi > = glmod(x,model,optn);
print "GLMOD: GOF=",gof;
print "GLMOD: Parameters=",parm;
```

****
Model Information
****

Number Valid Observations 7
Response Variable Y[2]
N Independent Variables 1

Significance Level: 0.0500000
Design Coding: Full-Rank
No Variable Selection Process

****
Model Effects
****

Intercept + X1 + X1 * X1

****
Simple Statistics
****

Column	Nobs	Mean	Std Dev	Skewness	Kurtosis
Y[2]	7	23.757143	4.3718254	-1.4920839	1.4095657
X[1]	8	44.375000	13.479482	-0.9323480	-0.0041244

****
Parameter Information
****

Parameter	Meaning

```
1 | Intercept
2 | X1
3 | X1*X1
```

****
Analysis of Variance
Full Analysis
****

Source	DF	SSQ	MeanSquare	F_Value	Prob>F
Model	2	111.8086183	55.90430913	77.955489	0.0006
Error	4	2.868524590	0.717131148		
C Total	6	114.6771429			

Goodness of Model Fit

R-square	0.974986083	Adj R-sq	0.962479124
MSE	0.717131148	Root MSE	0.846835963
AIC	-0.244786321	BIC	5.130213679
SBC	-0.407055874	C(p)	3.000000000

****
Analysis of Effects and Parameter Estimates
****

Variable	DF	Estimate	Std_Error	TypeII_SS	F_Value	Prob>F
Intercept	1	-5.985246	3.185222	2.5321127	3.53089	0.1334
X1	1	1.305246	0.172599	41.011712	57.1886	0.0016
X1*X1	1	-0.013098	2.2e-003	26.163975	36.4842	0.0038

****
Parameter Estimates and Wald Confidence Limits
****

Variable	DF	Estimate	Lower_CL	UpperCI	Std_Error
Intercept	1	-5.9852459	-12.228167	0.2576755	3.1852225
X1	1	1.3052459	0.9669585	1.6435333	0.17259876
X1*X1	1	-0.0130984	-0.0173486	-0.0088481	0.00216852

Estimated Covariance Matrix of Estimates

```
                 Intercept           X1       X1*X1
Intercept     14.14754098 -0.747540984  0.009016393
X1           -0.747540984  0.041540984 -0.000516393
X1*X1         0.009016393 -0.000516393  6.5574e-006


              ******************************
              Predicted Values and Residuals
              ******************************

N      Ydat   Wgt*Frq  Residual     Ypred     Y_LCI     Y_UCI
1  15.40000  1.000000  0.519672  14.88033  12.69701  17.06364
2  20.20000  1.000000 -1.183607  21.38361  20.01727  22.74994
3  25.70000  1.000000  0.432787  25.26721  23.87460  26.65983
4  26.20000  1.000000 -0.331148  26.53115  25.44573  27.61656
5  26.60000  1.000000  0.068852  26.53115  25.44573  27.61656
6  27.40000  1.000000  0.868852  26.53115  25.44573  27.61656
7     .             0     .         .         .         .
8  24.80000  1.000000 -0.375410  25.17541  23.05955  27.29127
```

Here, we can also use the "nomis" option:

```
model = "2 = 1 1*1";
optn = cons(4,1,.);
optn = [ "print"       5 ,
         "nomis"        ];
< gof,parm,ase,conf,cov,typ1,typ3,resi > = glmod(x,model,optn);


                  *****************
                  Model Information
                  *****************

              Number Valid Observations    7
              Response Variable          Y[2]
              N Independent Variables      1

              Skip Cases with Missing Values
              Significance Level:  0.0500000
              Design Coding:       Full-Rank
              No Variable Selection Process


                 *************
                 Model Effects
                 *************
```

Intercept + X1 + X1 * X1

****
Simple Statistics
****

Column	Nobs	Mean	Std Dev	Skewness	Kurtosis
Y[2]	7	23.757143	4.3718254	-1.4920839	1.4095657
X[1]	8	44.375000	13.479482	-0.9323480	-0.0041244

****
Parameter Information
****

Parameter	Meaning
1	Intercept
2	X1
3	X1*X1

****
Analysis of Variance
Full Analysis
****

Source	DF	SSQ	MeanSquare	F_Value	Prob>F
Model	2	111.8086183	55.90430913	77.955489	0.0006
Error	4	2.868524590	0.717131148		
C Total	6	114.6771429			

Goodness of Model Fit

R-square	0.974986083	Adj R-sq	0.962479124
MSE	0.717131148	Root MSE	0.846835963
AIC	-0.244786321	BIC	5.130213679
SBC	-0.407055874	C(p)	3.000000000

****
Analysis of Effects and Parameter Estimates
****

```
Variable          DF  Estimate Std_Error TypeII_SS F_Value Prob>F

Intercept          1 -5.985246  3.185222 2.5321127 3.53089 0.1334
X1                 1  1.305246  0.172599 41.011712 57.1886 0.0016
X1*X1              1 -0.013098  2.2e-003 26.163975 36.4842 0.0038


          ********************************************
          Parameter Estimates and Wald Confidence Limits
          ********************************************

 Variable          DF   Estimate   Lower_CL    UpperCI  Std_Error

 Intercept          1 -5.9852459 -12.228167  0.2576755  3.1852225
 X1                 1  1.3052459  0.9669585  1.6435333  0.17259876
 X1*X1              1 -0.0130984 -0.0173486 -0.0088481  0.00216852


             Estimated Covariance Matrix of Estimates

                      Intercept           X1       X1*X1
        Intercept   14.14754098 -0.747540984  0.009016393
        X1         -0.747540984  0.041540984 -0.000516393
        X1*X1       0.009016393 -0.000516393  6.5574e-006


                    ******************************
                    Predicted Values and Residuals
                    ******************************

     N      Ydat   Wgt*Frq  Residual     Ypred     Y_LCI     Y_UCI
     1  15.40000  1.000000  0.519672  14.88033  12.69701  17.06364
     2  20.20000  1.000000 -1.183607  21.38361  20.01727  22.74994
     3  25.70000  1.000000  0.432787  25.26721  23.87460  26.65983
     4  26.20000  1.000000 -0.331148  26.53115  25.44573  27.61656
     5  26.60000  1.000000  0.068852  26.53115  25.44573  27.61656
     6  27.40000  1.000000  0.868852  26.53115  25.44573  27.61656
     7     .            0     .         .         .         .
     8  24.80000  1.000000 -0.375410  25.17541  23.05955  27.29127
```

5. Unbalanced ANOVA for Two-Way Design with Interaction: (see [**?**], PROC GLM: Ex. 3, p. 972) The original data sourse is Afifi & Azen (1972).

```
a = [ 1 1 42 44 36 13 19 22,
      1 2 33  . 26  . 33 21,
      1 3 31 -3  . 25 25 24,
```

```
      2 1 28  . 23 34 42 13,
      2 2  . 34 33 31  . 36,
      2 3  3 26 28 32  4 16,
      3 1  .  .  1 29  . 19,
      3 2  . 11  9  7  1 -6,
      3 3 21  1  .  9  3  .,
      4 1 24  .  9 22 -2 15,
      4 2 27 12 12 -5 16 15,
      4 3 22  7 25  5 12  . ];
x = a[,1:3]; aa = a[ ,1:2];
for (j = 4; j <= 8; j++) x = x |> (aa -> a[,j]);
```

First we report the results for the full-rank design coding:

```
/* use bar operator */
clas = [ 1 2 ];
model = "3 = 1|2";
optn = [ "print"       5 ];
< gof,parm,ase,conf,cov,typ1,typ3,resi > = glmod(x,model,optn,clas);
print "GLMOD: GOF=",gof;
print "GLMOD: Parameters=",parm;
```

```
                 *****************
                 Model Information
                 *****************

            Number Valid Observations   58
            Response Variable          Y[3]
            N Independent Variables      2

            Significance Level:  0.0500000
            Design Coding:       Full-Rank
            No Variable Selection Process


                  *************
                  Model Effects
                  *************

            Intercept + C1 + C2 + C1 * C2


               ***********************
               Class Level Information
               ***********************
```

Class	Level	Value				
C[1]	4	1	2	3	4	
C[2]	3	1	2	3		

****
Parameter Information
****

Parameter	Meaning
1	Intercept
2	C1[1]
3	C1[2]
4	C1[3]
No Param.	C1[4]
6	C2[1]
7	C2[2]
No Param.	C2[3]
9	C1_1*C2_1
10	C1_1*C2_2
No Param.	C1_1*C2_3
11	C1_2*C2_1
12	C1_2*C2_2
No Param.	C1_2*C2_3
13	C1_3*C2_1
14	C1_3*C2_2
No Param.	C1_3*C2_3
No Param.	C1_4*C2_1
No Param.	C1_4*C2_2
No Param.	C1_4*C2_3

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

Variable	Value	Nobs	Proportion
C[1]	1	15	25.862069
	2	15	25.862069
	3	12	20.689655
	4	16	27.586207
C[2]	1	19	32.758621
	2	19	32.758621
	3	20	34.482759

****
Analysis of Variance
Full Analysis
****

Source	DF	SSQ	MeanSquare	F_Value	Prob>F
Model	11	4259.338506	387.2125914	3.5056922	0.0013
Error	46	5080.816667	110.4525362		
C Total	57	9340.155172			

Goodness of Model Fit

R-square	0.456024384	Adj R-sq	0.325943258
MSE	110.4525362	Root MSE	10.50964016
AIC	283.4214881	BIC	291.5462518
SBC	308.1468043	C(p)	12.00000000

****
Analysis of Effects and Parameter Estimates
****

Variable	DF	Estimate	Std_Error	TypeII_SS	F_Value	Prob>F
Intercept	1	14.20000	4.700054	1008.2000	9.12790	0.0041
Effect 1	3	.	.	365.36667	1.10264	0.3577
C1[1]	1	6.200000	6.646880	96.100000	0.87006	0.3558
C1[2]	1	3.966667	6.363903	42.912121	0.38851	0.5362
C1[3]	1	-5.700000	7.050081	72.200000	0.65367	0.4230
C1[4]	0	0	.	.	.	.
Effect 2	2	.	.	5.1666667	0.02339	0.9769
C2[1]	1	-0.600000	6.646880	0.9000000	8e-003	0.9285
C2[2]	1	-1.366667	6.363903	5.0939394	0.04612	0.8309
C2[3]	0	0	.	.	.	.
Effect 3	6	.	.	707.26626	1.06723	0.3958
C1_1*C2_1	1	9.533333	9.202189	118.54493	1.07327	0.3056
C1_1*C2_2	1	9.216667	9.497521	104.01667	0.94173	0.3369
C1_1*C2_3	0	0	.	.	.	.
C1_2*C2_1	1	10.43333	9.202189	141.98406	1.28548	0.2628
C1_2*C2_2	1	16.70000	9.301675	356.02979	3.22337	0.0792
C1_2*C2_3	0	0	.	.	.	.
C1_3*C2_1	1	8.433333	10.42169	72.326554	0.65482	0.4226
C1_3*C2_2	1	-2.733333	9.497521	9.1482993	0.08283	0.7748

C1_3*C2_3	0	0	.	.	.	.
C1_4*C2_1	0	0	.	.	.	.
C1_4*C2_2	0	0	.	.	.	.
C1_4*C2_3	0	0	.	.	.	.

****
Parameter Estimates and Wald Confidence Limits
****

Variable	DF	Estimate	Lower_CL	UpperCI	Std_Error
Intercept	1	14.200000	4.9880635	23.411936	4.7000540
Effect	1				
C1[1]	1	6.2000000	-6.8276455	19.227646	6.6468801
C1[2]	1	3.9666667	-8.5063543	16.439688	6.3639031
C1[3]	1	-5.7000000	-19.517905	8.1179047	7.0500809
C1[4]	0	0	.	.	.
Effect	2				
C2[1]	1	-0.6000000	-13.627646	12.427646	6.6468801
C2[2]	1	-1.3666667	-13.839688	11.106354	6.3639031
C2[3]	0	0	.	.	.

Parameter Estimates and Wald Confidence Limits

Variable	DF	Estimate	Lower_CL	UpperCI	Std_Error
Effect	3				
C1_1*C2_1	1	9.5333333	-8.5026252	27.569292	9.2021887
C1_1*C2_2	1	9.2166667	-9.3981324	27.831466	9.4975210
C1_1*C2_3	0	0	.	.	.
C1_2*C2_1	1	10.433333	-7.6026252	28.469292	9.2021887
C1_2*C2_2	1	16.700000	-1.5309475	34.930948	9.3016748
C1_2*C2_3	0	0	.	.	.
C1_3*C2_1	1	8.4333333	-11.992807	28.859474	10.421692
C1_3*C2_2	1	-2.7333333	-21.348132	15.881466	9.4975210
C1_3*C2_3	0	0	.	.	.
C1_4*C2_1	0	0	.	.	.
C1_4*C2_2	0	0	.	.	.
C1_4*C2_3	0	0	.	.	.

****
Predicted Values and Residuals
****

```
 N      Ydat   Wgt*Frq  Residual     Ypred     Y_LCI     Y_UCI
 1  42.00000  1.000000  12.66667  29.33333  20.69692  37.96975
 2  33.00000  1.000000  4.750000  28.25000  17.67260  38.82740
 3  31.00000  1.000000  10.60000  20.40000  10.93928  29.86072
 4  28.00000  1.000000         0  28.00000  18.53928  37.46072
 5     .            0     .         .         .         .
 6  3.000000  1.000000 -15.16667  18.16667  9.530252  26.80308
.....................................................................
68 -6.000000  1.000000 -10.40000  4.400000 -5.060718  13.86072
69     .            0     .         .         .         .
70  15.00000  1.000000  1.400000  13.60000  4.139282  23.06072
71  15.00000  1.000000  2.166667  12.83333  4.196919  21.46975
72     .            0     .         .         .         .
```

Now we report the results for the rankdeficient (GLM) design coding:

```
/* GLM Design Coding */
clas = [ 1 2 ];
model = "3 = 1 2 1*2";
optn = [ "print"       5 ,
         "desi"    "ful" ];
< gof,parm,ase,conf,cov,typ1,typ3,resi > = glmod(x,model,optn,clas);
print "GLMOD: Parameters=",parm;
```

6. Analysis of Covariance: (see [**?**], PROC GLM: Ex. 4, p. 974) Here the features of regression and ANOVA are combined when using both, categorical variables and interval scaled covariates:

```
x = [ 'a' 11  6, 'a'  8  0, 'a'  5  2, 'a' 14  8, 'a' 19 11,
      'a'  6  4, 'a' 10 13, 'a'  6  1, 'a' 11  8, 'a'  3  0,
      'd'  6  0, 'd'  6  2, 'd'  7  3, 'd'  8  1, 'd' 18 18,
      'd'  8  4, 'd' 19 14, 'd'  8  9, 'd'  5  1, 'd' 15  9,
      'f' 16 13, 'f' 13 10, 'f' 11 18, 'f'  9  5, 'f' 21 23,
      'f' 16 12, 'f' 12  5, 'f' 12 16, 'f'  7  1, 'f' 12 20];
```

We first report the results for the rankdeficient (GLM) design coding:

```
/* GLM Design Coding */
clas = 1;
model = "3 = 1 2";
optn = [ "print"       5 ,
         "desi"    "ful" ];
< gof,parm,ase,conf,cov,typ1,typ3,resi > = glmod(x,model,optn,clas);
print "GLMOD: GOF=",gof;
print "GLMOD: Parameters=",parm;
```

****
Model Information
****

Number Valid Observations 30
Response Variable Y[3]
N Independent Variables 2

Significance Level: 0.0500000
Design Coding: Full-Rank
No Variable Selection Process

****
Model Effects
****

Intercept + C1 + X2

****
Class Level Information
****

Class	Level	Value
C[1]	3	a d f

****
Simple Statistics
****

Column	Nobs	Mean	Std Dev	Skewness	Kurtosis
Y[3]	30	7.9000000	6.6661781	0.6128269	-0.6297199
X[2]	30	10.733333	4.7917554	0.5249442	-0.6319484

****
Parameter Information
****

Parameter	Meaning
1	Intercept
2	C1a
3	C1d
No Param.	C1f

Number of Observations for Class Levels

Variable	Value	Nobs	Proportion
C[1]	a	10	33.333333
	d	10	33.333333
	f	10	33.333333

Analysis of Variance
Full Analysis

Source	DF	SSQ	MeanSquare	F_Value	Prob>F
Model	3	871.4974030	290.4991343	18.103860	0.0000
Error	26	417.2025970	16.04625373		
C Total	29	1288.700000			

Goodness of Model Fit

R-square	0.676260885	Adj R-sq	0.638906372
MSE	16.04625373	Root MSE	4.005777544
AIC	86.97123699	BIC	90.15466894
SBC	92.57602651	C(p)	4.000000000

Analysis of Effects and Parameter Estimates

Variable		DF	Estimate	Std_Error	TypeII_SS	F_Value	Prob>F
Intercept		1	-0.434671	2.471354	0.4963929	0.03094	0.8617
Effect	1	2	.	.	68.553711	2.13613	0.1384
C1a		1	-3.446138	1.886781	53.529875	3.33597	0.0793
C1d		1	-3.337167	1.853866	51.996324	3.24040	0.0835
C1f		0	0	.	.	.	.
X2		1	0.987184	0.164498	577.89740	36.0145	0.0000

Parameter Estimates and Wald Confidence Limits

Variable	DF	Estimate	Lower_CL	UpperCI	Std_Error
Intercept	1	-0.4346712	-5.2784351	4.4090928	2.4713536
Effect 1					
C1a	1	-3.4461383	-7.1441604	0.2518838	1.8867806
C1d	1	-3.3371669	-6.9706784	0.2963445	1.8538664
C1f	0	0	.	.	.
X2	1	0.9871838	0.6647745	1.3095931	0.16449757

Effect	Type_I_SS	MeanSquare	F_Value	DF	Prob>F
C1	293.6000000	146.8000000	9.1485528	2	0.0010
X2	577.8974030	577.8974030	36.014475	1	0.0000

Effect	Type_III_SS	MeanSquare	F_Value	DF	Prob>F
C1	68.55371060	34.27685530	2.1361282	2	0.1384
X2	577.8974030	577.8974030	36.014475	1	0.0000

Estimated Covariance Matrix of Estimates

	Intercept	C1a	C1d	C1f
Intercept	0.380623946	-0.178313659	-0.163086003	0
C1a	-0.178313659	0.221854975	0.117605396	0
C1d	-0.163086003	0.117605396	0.214182125	0
C1f	0	0	0	0
X2	-0.021753794	0.006070826	0.004890388	0

Estimated Covariance Matrix of Estimates

	X2
Intercept	-0.021753794
C1a	0.006070826
C1d	0.004890388
C1f	0
X2	0.001686341

Predicted Values and Residuals

```
 N      Ydat   Wgt*Frq  Residual     Ypred     Y_LCI     Y_UCI
 1  6.000000  1.000000 -0.978212  6.978212  4.311701  9.644724
 2         0  1.000000 -4.016661  4.016661  1.376001  6.657321
 3  2.000000  1.000000  0.944890  1.055110 -1.927147  4.037367
 4  8.000000  1.000000 -1.939764  9.939764  6.889280  12.99025
.............................................................
27  5.000000  1.000000 -6.411535  11.41153  8.789994  14.03307
28  16.00000  1.000000  4.588465  11.41153  8.789994  14.03307
29  1.000000  1.000000 -5.475616  6.475616  3.195410  9.755821
30  20.00000  1.000000  8.588465  11.41153  8.789994  14.03307
```

Now, we solve the same problem however using the (default) full-rank design coding:

```
/* DMREG Design Coding */
clas = 1;
model = "3 = 1 2";
optn = [ "print"       5 ];
< gof,parm,ase,conf,cov,typ1,typ3,resi > = glmod(x,model,optn,clas);
print "GLMOD: Parameters=",parm;
```

7. 2-Way AOV with "immer" Data of MASS package in CRAN:

```
print "Analyze immer data in MASS package of CRAN";
immer = [" UF  M  81.0  80.7, UF  S 105.4  82.3, UF  V 119.7  80.4, UF  T 109.7
          UF  P  98.3  84.2,  W  M 146.6 100.4,  W  S 142.0 115.5,  W  V 150.7 1
           W  T 191.5 147.7,  W  P 145.7 108.1,  M  M  82.3 103.1,  M  S  77.3 1
           M  V  78.4 116.5,  M  T 131.3 139.9,  M  P  89.6 129.6,  C  M 119.8
           C  S 121.4  61.9,  C  V 124.0  96.2,  C  T 140.8 125.5,  C  P 124.8
          GR  M  98.9  66.4, GR  S  89.0  49.9, GR  V  69.1  96.7, GR  T  89.3
          GR  P 104.1  80.3,  D  M  86.9  67.7,  D  S  77.1  66.7,  D  V  78.9
           D  T 101.8  91.8,  D  P  96.0  94.1 "];
nr = nrow(immer);
loc = immer[,1]; /* print "LOC Before", loc; */
/* Crookston, Duluth, GrandRapids, Morris, UniversityFarm, Waseca */
old = [" C D GR M UF W "]; new = [ 1:6 ];
loc = replace(loc,old,new); /* print "LOC After", loc; */
```

We change the levels of the two class variables to be consistent with the results by Bretz et al. (2011), page 99:

```
var = immer[,2];   /* print "VAR Before", var; */
/* Manchuria, Peatland, Svanasota, Trebi, Velvet */
old = [" M P S T V "]; new = [ 1:5 ];
```

```
var = replace(var,old,new); /* print "VAR After", var; */

imm = cons(nr,3);
y12 = .5 * (immer[,3] + immer[,4]);
imm[,1] = var; imm[,2] = loc; imm[,3] = y12;
cnam = [" Var Loc Y12 "];
imm = cname(imm,cnam);
print "IMMER Data Set", imm;
```

For computing multiple comparison of the means of the effect 1 ("var") we specify the following list of two options matrices:

```
model = "3 = 1 2"; class = [ 1 2 ];
optns = [ "print"        5 ,
          "pcov"           ,
          "pcorr"          ,
          "desi"     "ful" ];
means = [ "ctyp"     "tuk" ,
          "eff"          1 ,
          "print"        2 ,
          "pall"           ,
          "ptyp"     "wes" ];
list lstopt[2];
lstopt[1] = optns;
lstopt[2] = means; print lstopt;
< gof,parm,ase,conf,cov,typ1,typ3,resi > = glmod(imm,model,lstopt,class);
```

To be sure we print the list of 2-column options matrices:

```
****************************
lstopt (List with 2 Entries)
****************************

     lstopt[1]: optns
     ****************

  Mixed Type Matrix optns

              1   2
         1 print   5
         2  pcov   0
         3 pcorr   0
         4  desi ful


     lstopt[2]: means
```

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

Mixed Type Matrix means

      1   2
1  ctyp tuk
2   eff   1
3 print   2
4  pall   0
5  ptyp wes
```

First we report the common GLMOD output:

```
*****************
Model Information
*****************

Number Valid Observations   30
Response Variable          Y[3]
N Independent Variables     2

Significance Level:  0.0500000
Design Coding:       Full-Rank
No Variable Selection Process


*************
Model Effects
*************

Intercept + C1 + C2


***********************
Class Level Information
***********************

Class  Level    Value

 C[1]      5       1       2       3       4       5
 C[2]      6       1       2       3       4       5
                   6


*********************
Parameter Information
*********************
```

```
Parameter | Meaning
-----------------------------------------
        1 | Intercept
        2 | Var[1]
        3 | Var[2]
        4 | Var[3]
        5 | Var[4]
No Param. | Var[5]
        7 | Loc[1]
        8 | Loc[2]
        9 | Loc[3]
       10 | Loc[4]
       11 | Loc[5]
No Param. | Loc[6]
```

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

Variable	Value	Nobs	Proportion
C[1]	1	6	20.000000
	2	6	20.000000
	3	6	20.000000
	4	6	20.000000
	5	6	20.000000
C[2]	1	5	16.666667
	2	5	16.666667
	3	5	16.666667
	4	5	16.666667
	5	5	16.666667
	6	5	16.666667

```
********************
Analysis of Variance
   Full Analysis
********************
```

Source	DF	SSQ	MeanSquare	F_Value	Prob>F
Model	9	13265.43817	1473.937574	13.299632	0.0000
Error	20	2216.508833	110.8254417		
C Total	29	15481.94700			

Goodness of Model Fit

R-square	0.856832682	Adj R-sq	0.792407389
MSE	110.8254417	Root MSE	10.52736632
AIC	149.0747377	BIC	160.5747377
SBC	163.0867115	C(p)	10.00000000

****
Analysis of Effects and Parameter Estimates
****

Variable		DF	Estimate	Std_Error	TypeII_SS	F_Value	Prob>F
Intercept		1	134.1333	6.077978	53975.253	487.029	0.0000
Effect	1	4	.	.	2654.9862	5.98912	0.0025
Var[1]		1	-4.791667	6.077978	68.880208	0.62152	0.4397
Var[2]		1	3.358333	6.077978	33.835208	0.30530	0.5867
Var[3]		1	-8.050000	6.077978	194.40750	1.75418	0.2003
Var[4]		1	19.01667	6.077978	1084.9008	9.78928	0.0053
Var[5]		0	0	.	.	.	.
Effect	2	5	.	.	10610.452	19.1480	0.0000
Loc[1]		1	-27.14000	6.658091	1841.4490	16.6158	0.0006
Loc[2]		1	-53.20000	6.658091	7075.6000	63.8445	0.0000
Loc[3]		1	-55.48000	6.658091	7695.0760	69.4342	0.0000
Loc[4]		1	-30.73000	6.658091	2360.8323	21.3023	0.0002
Loc[5]		1	-43.15000	6.658091	4654.8063	42.0012	0.0000
Loc[6]		0	0	.	.	.	.

****
Parameter Estimates and Wald Confidence Limits
****

Variable		DF	Estimate	Lower_CL	UpperCI	Std_Error
Intercept		1	134.13333	122.22072	146.04595	6.0779778
Effect	1					
Var[1]		1	-4.7916667	-16.704284	7.1209509	6.0779778
Var[2]		1	3.3583333	-8.5542842	15.270951	6.0779778
Var[3]		1	-8.0500000	-19.962618	3.8626175	6.0779778
Var[4]		1	19.016667	7.1040491	30.929284	6.0779778
Var[5]		0	0	.	.	.
Effect	2					
Loc[1]		1	-27.140000	-40.189619	-14.090381	6.6580911
Loc[2]		1	-53.200000	-66.249619	-40.150381	6.6580911
Loc[3]		1	-55.480000	-68.529619	-42.430381	6.6580911

```
Loc[4]             1 -30.730000 -43.779619 -17.680381  6.6580911
Loc[5]             1 -43.150000 -56.199619 -30.100381  6.6580911
Loc[6]             0           0     .          .          .
```

```
                  ******************************
                  Predicted Values and Residuals
                  ******************************

     N      Ydat   Wgt*Frq  Residual     Ypred     Y_LCI     Y_UCI
     1  80.85000  1.000000 -5.341667  86.19167  73.51323  98.87011
     2  93.85000  1.000000  10.91667  82.93333  70.25489  95.61177
     3  100.0500  1.000000  9.066667  90.98333  78.30489  103.6618
     4  98.45000  1.000000 -11.55000  110.0000  97.32156  122.6784
     5  91.25000  1.000000 -3.091667  94.34167  81.66323  107.0201
.....................................................................
    26  77.30000  1.000000  1.158333  76.14167  63.46323  88.82011
    27  71.90000  1.000000 -0.983333  72.88333  60.20489  85.56177
    28  73.15000  1.000000 -7.783333  80.93333  68.25489  93.61177
    29  96.80000  1.000000 -3.150000  99.95000  87.27156  112.6284
    30  95.05000  1.000000  10.75833  84.29167  71.61323  96.97011
```

After computing all common GLMOD results we report the result of interfacing with the `multcomp()` function. For 5 levels of the `"var"` effect we obtain $10 = 5 * 4/2$ Tukey contrasts of all pairwise comparisons:

```
                Parameter Estimates and CIs (Run=1)
                ***********************************

   Contrast  Coefficient      LowerCI      UpperCI   AsymStdErr

  C_1_2       8.150000000 -10.03320621  26.33320621  6.077977780
  C_1_3      -3.258333333 -21.44153955  14.92487288  6.077977780
  C_2_3      -11.40833333 -29.59153955  6.774872882  6.077977780
  C_1_4       23.80833333  5.625127118  41.99153955  6.077977780
  C_2_4       15.65833333 -2.524872882  33.84153955  6.077977780
  C_3_4       27.06666667  8.883460452  45.24987288  6.077977780
  C_1_5       4.791666667 -13.39153955  22.97487288  6.077977780
  C_2_5      -3.358333333 -21.54153955  14.82487288  6.077977780
  C_3_5       8.050000000 -10.13320621  26.23320621  6.077977780
  C_4_5      -19.01666667 -37.19987288 -0.833460452  6.077977780
```

The Westfall adjusted probabilities agree with those reported by Bretz et al. (2011) at page 99:

Parameter Estimates and p Values (Run=1)

Adjusted Probabilities: Westfall (1997)

Contrast	Coefficient	t Value	RawProb	AdjProb
C_1_2	8.150000000	1.340906515	0.194982573	0.389942042
C_1_3	-3.258333333	-0.536088392	0.597809911	0.825663654
C_2_3	-11.40833333	-1.876994906	0.075185023	0.268938903
C_1_4	23.80833333	3.917147149	0.000853865	0.004381320
C_2_4	15.65833333	2.576240635	0.018028753	0.061765380
C_3_4	27.06666667	4.453235541	0.000243963	0.002023003
C_1_5	4.791666667	0.788365282	0.439728141	0.439728186
C_2_5	-3.358333333	-0.552541232	0.586700499	0.825663654
C_3_5	8.050000000	1.324453674	0.200288832	0.398601706
C_4_5	-19.01666667	-3.128781867	0.005287884	0.018968513

The following is the matrix of the 10 Tukey contrasts:

Contrast Matrix: Tukey (Bofinger, Hayter)

	1	2	3	4	5
C_1_2	0	0	-1.0000000	1.0000000	0
C_1_3	-1.0000000	0	0	1.0000000	0
C_2_3	0	0	0	-1.0000000	1.0000000
C_1_4	0	-1.0000000	0	1.0000000	0
C_2_4	0	-1.0000000	1.0000000	0	0
C_3_4	-1.0000000	1.0000000	0	0	0
C_1_5	0	0	-1.0000000	0	1.0000000
C_2_5	-1.0000000	0	0	0	1.0000000
C_3_5	0	-1.0000000	0	0	1.0000000
C_4_5	-1.0000000	0	1.0000000	0	0

Estimated Variance-Covariance Matrix

	C_1_2	C_1_3	C_2_3	C_1_4
C_1_2	36.94181389			
C_1_3	18.47090694	36.94181389		
C_2_3	-18.47090694	18.47090694	36.94181389	
C_1_4	18.47090694	18.47090694	0	36.94181389
C_2_4	-18.47090694	-3.5527e-015	18.47090694	18.47090694
C_3_4	-3.5527e-015	-18.47090694	-18.47090694	18.47090694
C_1_5	18.47090694	18.47090694	0	18.47090694
C_2_5	-18.47090694	3.5527e-015	18.47090694	3.5527e-015

```
C_3_5   3.5527e-015 -18.47090694 -18.47090694  3.5527e-015
C_4_5   7.1054e-015  7.1054e-015            0 -18.47090694


           Estimated Variance-Covariance Matrix

              C_2_4        C_3_4        C_1_5        C_2_5
C_2_4   36.94181389
C_3_4   18.47090694  36.94181389
C_1_5  -3.5527e-015 -3.5527e-015  36.94181389
C_2_5   18.47090694            0  18.47090694  36.94181389
C_3_5             0  18.47090694  18.47090694  18.47090694
C_4_5  -18.47090694 -18.47090694  18.47090694  18.47090694


           Estimated Variance-Covariance Matrix

                         C_3_5        C_4_5
       C_3_5   36.94181389
       C_4_5   18.47090694  36.94181389


                Global Test for Run=1
     ChiSquared=23.9565 Probability=  0.0001 DF=4
    F=5.98912 Probability=  0.0025 DFnum=4 DFden=20
```

8. 2-Way AOV with "litter" Data by Westfall and Young(1993): Dunnett contrasts for comparing with control level:

```
litter = [ 1     0  28.05     22.5     15,   2    0  33.33     22.5     14,
           3     0  36.37     22.0     14,   4    0  35.52     22.0     13,
           5     0  36.77     21.5     15,   6    0  29.60     23.0      5,
.....................................................................
          69   500  33.20     22.0     12,  70  500  32.97     22.5     14,
          71   500  38.75     23.0     16,  72  500  33.15     22.5     12,
          73   500  30.70     21.5     13,  74  500  35.32     22.0     17 ];


  cnam = [" N dose weight gesttime number "];
  litt = cname(litter,cnam);


  print "Dunnett, ptyp=single step, p.101, 102";
  model = "3 = 2 4 5"; class = 2;
  optns = [ "print"       5 ];
  means = [ "ctyp"      "dun" ,
            "contpop"       1 ,
```

```
          "tail"     "less" ,
          "eff"           1 ,
          "print"         2 ,
          "pall"            ,
          "ptyp"      "adj" ];
list lstopt[2];
lstopt[1] = optns;
lstopt[2] = means; print lstopt;
 < gof,parm,ase,conf,cov,typ1,typ3,resi > = glmod(litt,model,lstopt,class);
```

```
                  ***************************
                  lstopt (List with 2 Entries)
                  ***************************

                       lstopt[1]: optns
                       ****************

                  Mixed Type Row Vector optns

                      R |        1       2
                             print       5


                       lstopt[2]: means
                       ****************

                    Mixed Type Matrix means

                                1    2
                          1    ctyp  dun
                          2 contpop    4
                          3    tail less
                          4     eff    1
                          5   print    2
                          6    pall    0
                          7    ptyp  adj


                        *****************
                        Model Information
                        *****************

                  Number Valid Observations   74
                  Response Variable          Y[3]
                  N Independent Variables     3
```

```
Significance Level:  0.0500000
Design Coding:       Full-Rank
No Variable Selection Process


             *************
             Model Effects
             *************

          Intercept + C2 + X4 + X5


          ***********************
          Class Level Information
          ***********************

Class  Level    Value

 C[2]      4        0       5      50     500


            *****************
            Simple Statistics
            *****************

 Column  Nobs        Mean     Std Dev    Skewness    Kurtosis
 Y[3]      74   30.332568   4.4146373  -0.5707132   0.0628907
 X[4]      74   22.087838   0.4407194   0.0145519  -1.0594586
 X[5]      74   13.432432   2.5646762  -1.2042429   1.4062660


            **********************
            Parameter Information
            **********************

          Parameter | Meaning
          ----------------------------------------
                  1 | Intercept
                  2 | dose[0]
                  3 | dose[5]
                  4 | dose[50]
          No Param. | dose[500]
                  6 | gesttime
                  7 | number


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

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

Variable       Value    Nobs  Proportion
C[2]               0      20   27.027027
                   5      19   25.675676
                  50      18   24.324324
                 500      17   22.972973


                    ********************
                    Analysis of Variance
                       Full Analysis
                    ********************

Source      DF         SSQ  MeanSquare   F_Value  Prob>F

Model        5 336.2758434 67.25516868 4.2095505  0.0022
Error       68 1086.422769 15.97680542
C Total     73 1422.698612


                    Goodness of Model Fit

R-square   0.236364779           Adj R-sq   0.180215131
MSE        15.97680542           Root MSE   3.997099626
AIC        210.8069660           BIC        213.8502186
SBC        224.6313565           C(p)       6.000000000


             ****************************************
             Analysis of Effects and Parameter Estimates
             ****************************************

Variable          DF  Estimate Std_Error TypeII_SS F_Value Prob>F

Intercept          1 -51.42979  24.60907 69.779704 4.36756 0.0404
Effect     1       3    .          .      122.80085 2.56207 0.0620
dose[0]            1  2.675243  1.334312 64.224609 4.01987 0.0490
dose[5]            1 -0.677157  1.339389 4.0837152 0.25560 0.6148
dose[50]           1  0.384364  1.451045 1.1210181 0.07017 0.7919
dose[500]          0         0    .         .        .       .
gesttime           1  3.438128  1.107456 153.98590 9.63809 0.0028
number             1  0.385550  0.191370 64.849055 4.05895 0.0479


            ********************************************
            Parameter Estimates and Wald Confidence Limits
```

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

Variable          DF   Estimate   Lower_CL    UpperCI  Std_Error

Intercept          1 -51.429794 -99.662685 -3.1969039  24.609070
Effect     1
dose[0]            1  2.6752426  0.0600390  5.2904462  1.3343121
dose[5]            1 -0.6771572 -3.3023117  1.9479972  1.3393891
dose[50]           1  0.3843637 -2.4596331  3.2283605  1.4510455
dose[500]          0           0     .          .          .
gesttime           1  3.4381280  1.2675538  5.6087022  1.10745615
number             1  0.3855504  0.0104717  0.7606291  0.19137021
```

The common `glmod` output ends here, after that the `multcomp` output starts:

```
                  ******************************
                  Predicted Values and Residuals
                  ******************************

      N      Ydat   Wgt*Frq  Residual     Ypred     Y_LCI     Y_UCI
      1  28.05000  1.000000 -6.336585  34.38659  32.29018  36.48299
      2  33.33000  1.000000 -0.671035  34.00103  31.97620  36.02587
      3  36.37000  1.000000  4.088029  32.28197  30.47542  34.08852
      4  35.52000  1.000000  3.623580  31.89642  30.09924  33.69361
      5  36.77000  1.000000  5.821543  30.94846  28.66265  33.23426
.....................................................................
     71  38.75000  1.000000  4.934043  33.81596  30.87713  36.75479
     72  33.15000  1.000000  2.595309  30.55469  28.47991  32.62947
     73  30.70000  1.000000  3.197886  27.50211  25.04677  29.95746
     74  35.32000  1.000000  4.556621  30.76338  28.14481  33.38195


               Parameter Estimates and CIs (Run=1)
               ***********************************

    Contrast  Coefficient      LowerCI      UpperCI   AsymStdErr

   C_2_1      -3.352399859 -1.7977e+308 -0.627420156  1.290753274
   C_3_1      -2.290878926 -1.7977e+308  0.534668416  1.338389596
   C_4_1      -2.675242626 -1.7977e+308  0.141696432  1.334312072
```

The following results agree with those reported by Bretz et al. (2011), p. 102:

```
               Parameter Estimates and p Values (Run=1)
```

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

          Adjusted Probabilities: SingleStep

  Contrast  Coefficient      t Value      RawProb      AdjProb

C_2_1      -3.352399859 -2.597242963  0.005754630  0.015768066
C_3_1      -2.290878926 -1.711668212  0.045757812  0.112229318
C_4_1      -2.675242626 -2.004960221  0.024475129  0.062691512


          Contrast Matrix: Dunnett (Hsu, MCB)

                    1          2          3          4
      C_2_1  -1.0000000  1.0000000          0          0
      C_3_1  -1.0000000          0  1.0000000          0
      C_4_1  -1.0000000          0          0  1.0000000


               Global Test for Run=1
        ChiSquared=7.6862 Probability=  0.0530 DF=3
      F=2.56207 Probability=  0.0620 DFnum=3 DFden=68
```

9. 2-Way AOV with "litter" Data by Westfall and Young(1993): Williams and Marcus contrasts for trend tests, see Bretz et al. (2011), pp. 103:

```
litter = [ 1     0  28.05     22.5     15,   2    0  33.33     22.5     14,
           3     0  36.37     22.0     14,   4    0  35.52     22.0     13,
           5     0  36.77     21.5     15,   6    0  29.60     23.0      5,
.....................................................................................
          69   500  33.20     22.0     12,  70  500  32.97     22.5     14,
          71   500  38.75     23.0     16,  72  500  33.15     22.5     12,
          73   500  30.70     21.5     13,  74  500  35.32     22.0     17 ];


  cnam = [" N dose weight gesttime number "];
  litt = cname(litter,cnam);
```

Here the dose levels are treated in an ordinal order for trend test using Williams and Marcus contrasts:

```
  print "Full Rank Model: Williams, ptyp=single-step, p. 107";
  model = "3 = 2 4 5"; class = 2;
  optns = [ "print"       5 ];
  means = [ "ctyp"      "wil" ,
            "tail"     "less" ,
```

```
          "eff"           1 ,
          "print"         2 ,
          "pall"            ,
          "ptyp"      "adj" ];
list lstopt[2];
lstopt[1] = optns;
lstopt[2] = means; print lstopt;
 < gof,parm,ase,conf,cov,typ1,typ3,resi > = glmod(litt,model,lstopt,class);
```

The common `glmod` results are the same as above.

```
              Parameter Estimates and CIs (Run=1)
              ***********************************

   Contrast  Coefficient      LowerCI      UpperCI   AsymStdErr

 C_1        -2.675242626 -1.7977e+308 -0.084350706  1.334312072
 C_2        -2.477569866 -1.7977e+308 -0.297933603  1.122514974
 C_3        -2.785380419 -1.7977e+308 -0.753359887  1.046492717
```

The results here agree with those by Bretz et al. (2011), page 107:

```
             Parameter Estimates and p Values (Run=1)
             ****************************************

               Adjusted Probabilities: SingleStep

   Contrast  Coefficient      t Value      RawProb      AdjProb

 C_1        -2.675242626 -2.004960221  0.024475129  0.043619094
 C_2        -2.477569866 -2.207159746  0.015340733  0.028199752
 C_3        -2.785380419 -2.661633830  0.004847511  0.009715389
```

There is a trend in the size of dose values, the adjusted probabilities are clearly descending.

```
             Contrast Matrix: Williams (Bretz, 1999)

                       1          2          3          4
        C_1  -1.0000000          0          0  1.0000000
        C_2  -1.0000000          0  0.5142857  0.4857143
        C_3  -1.0000000  0.3518519  0.3333333  0.3148148
```

We may also compute the "free" (step-down Dunnett) adjusted probabilities as shown at Bretz et al. (2011), p. 108:

```
model = "3 = 2 4 5"; class = 2;
optns = [ "print"       5 ];
means = [ "ctyp"      "wil" ,
          "tail"     "less" ,
          "eff"           1 ,
          "print"         2 ,
          "pall"            ,
          "ptyp"      "fre" ];
list lstopt[2];
lstopt[1] = optns;
lstopt[2] = means; print lstopt;
 < gof,parm,ase,conf,cov,typ1,typ3,resi > = glmod(litt,model,lstopt,class);
```

Parameter Estimates and CIs (Run=1)

Contrast	Coefficient	LowerCI	UpperCI	AsymStdErr
C_1	-2.675242626	-1.7977e+308	-0.085850352	1.334312072
C_2	-2.477569866	-1.7977e+308	-0.299195208	1.122514974
C_3	-2.785380419	-1.7977e+308	-0.754536050	1.046492717

Parameter Estimates and p Values (Run=1)

Adjusted Probabilities: Free (stepDown)

Contrast	Coefficient	t Value	RawProb	AdjProb
C_1	-2.675242626	-2.004960221	0.024475129	0.024475131
C_2	-2.477569866	-2.207159746	0.015340733	0.023741920
C_3	-2.785380419	-2.661633830	0.004847511	0.009553689

Again, there is a trend, the adjusted probabilities are clearly descending.

Contrast Matrix: Williams (Bretz, 1999)

	1	2	3	4
C_1	-1.0000000	0	0	1.0000000
C_2	-1.0000000	0	0.5142857	0.4857143
C_3	-1.0000000	0.3518519	0.3333333	0.3148148

Finally we test the levels for trend with Marcus (modified Williams) contrasts:

```
model = "3 = 2 4 5"; class = 2;
optns = [ "print"       5 ];
means = [ "ctyp"      "mar" ,
          "tail"     "less" ,
          "eff"           1 ,
          "print"         2 ,
          "pall"            ,
          "ptyp"      "adj" ];
list lstopt[2];
lstopt[1] = optns;
lstopt[2] = means; print lstopt;
 < gof,parm,ase,conf,cov,typ1,typ3,resi > = glmod(litt,model,lstopt,class);
```

Parameter Estimates and CIs (Run=1)

Contrast	Coefficient	LowerCI	UpperCI	AsymStdErr
C_1	-2.785380419	-1.7977e+308	-0.531292015	1.046492717
C_2	-2.477569866	-1.7977e+308	-0.059733654	1.122514974
C_3	-0.844349422	-1.7977e+308	1.184667482	0.942000060
C_4	-2.675242626	-1.7977e+308	0.198793016	1.334312072
C_5	-1.042022182	-1.7977e+308	1.479810123	1.170796645
C_6	-0.834340556	-1.7977e+308	1.610878998	1.135228082

Parameter Estimates and p Values (Run=1)

Adjusted Probabilities: SingleStep

Contrast	Coefficient	t Value	RawProb	AdjProb
C_1	-2.785380419	-2.661633830	0.004847511	0.015689010
C_2	-2.477569866	-2.207159746	0.015340733	0.044625251
C_3	-0.844349422	-0.896336909	0.186617613	0.377830828
C_4	-2.675242626	-2.004960221	0.024475129	0.068058452
C_5	-1.042022182	-0.890011247	0.188298911	0.380492324
C_6	-0.834340556	-0.734954120	0.232447326	0.446890546

Contrast Matrix: Marcus (Bretz, 1999)

	1	2	3	4

```
C_1  -1.0000000  0.3518519  0.3333333  0.3148148
C_2  -1.0000000          0  0.5142857  0.4857143
C_3  -0.5128205 -0.4871795  0.5142857  0.4857143
C_4  -1.0000000          0          0  1.0000000
C_5  -0.5128205 -0.4871795          0  1.0000000
C_6  -0.3508772 -0.3333333 -0.3157895  1.0000000
```

```
Estimated Variance-Covariance Matrix

              C_1          C_2          C_3          C_4
C_1   1.095147007
C_2   1.094062334  1.260039868
C_3   0.559555751  0.875989338  0.887364114
C_4   1.095562046  1.241768302  0.839243689  1.780388705
C_5   0.561055464  0.857717772  0.850618465  1.377864091
C_6   0.384800930  0.575640146  0.559438989  1.273480941
```

```
Estimated Variance-Covariance Matrix

                    C_5          C_6
C_5   1.370764784
C_6   1.257279785  1.288742797
```

```
Global Test for Run=1
  ChiSquared=7.6862 Probability=  0.0530 DF=3
F=2.56207 Probability=  0.0620 DFnum=3 DFden=68
```

10. Variable selection in regression models using multiple testing, see Bretz et al. (2011), p. 108:

```
  options NOECHO;
#include "..\\tdata\\bodyfat.dat"
  options ECHO;
  nr = nrow(bodyfat);


  contr = cons(9,10);
  ind = [ 1:9, 2:10 ]‘; print "Ind=",ind;
  contr[ind] = 1;
  rnam = [" age waistcirc hipcirc elbowbreadth
            kneebreadth anthro3a anthro3b anthro3c anthro4 "];
  contr = rname(contr,rnam); print "Contrasts=", contr;
```

The contrasts are specified for testing the problem of variable selection. The following 9 contrasts specify the nine tests whether each of the parameters, except the intercept, can be considered zero:

```
Contrasts=
           UPP |   1   2   3   4   5   6   7   8   9  10
---------------------------------------------------------
           age |   0   1   0   0   0   0   0   0   0   0
     waistcirc |   0   0   1   0   0   0   0   0   0   0
       hipcirc |   0   0   0   1   0   0   0   0   0   0
  elbowbreadth |   0   0   0   0   1   0   0   0   0   0
   kneebreadth |   0   0   0   0   0   1   0   0   0   0
      anthro3a |   0   0   0   0   0   0   1   0   0   0
      anthro3b |   0   0   0   0   0   0   0   1   0   0
      anthro3c |   0   0   0   0   0   0   0   0   1   0
       anthro4 |   0   0   0   0   0   0   0   0   0   1
```

```
print "Variable selection: see Bretz et al., p. 108";
model = "3 = 2 4:11"; print "Model=",modl;
optns = [ "print"             5 ,
          "outf"   "bodyfact.mc" ];
means = [ "ctyp"           "inp" ,
          "eff"            "par" ,
          "print"              2 ,
          "pall"                 ,
          "ptyp"           "adj" ];
list lstopt[2];
lstopt[1] = optns;
lstopt[2] = means; print lstopt;
< gof,parm,ase,conf,cov,typ1,typ3,resi > = glmod(bodyfat,model,lstopt,.,contr);
```

```
                    *****************
                    Simple Statistics
                    *****************

 Column   Nobs        Mean     Std Dev    Skewness    Kurtosis
 Y[ 3]      71   30.782817   11.042764   0.4637425   0.0164438
 X[ 2]      71   50.859155   13.375132  -0.7604724  -0.6516231
 X[ 4]      71   87.383099   14.028389   0.3960147  -0.7654479
 X[ 5]      71   105.28028   11.040492   0.6092225  -0.4810871
 X[ 6]      71   6.5084507   0.4563039  -0.2354151  -0.1025140
 X[ 7]      71   9.3014085   0.9136259   0.6323933   0.1680129
 X[ 8]      71   3.8691549   0.4730622  -0.8818444   1.0687020
 X[ 9]      71   4.2908451   0.4856240  -1.3587987   2.7383753
 X[10]      71   3.8856338   0.5647066  -1.0698216   1.4070791
```

```
X[11]      71   5.3980282   0.6373710  -1.2877863   2.5416125


                    **********************
                    Parameter Information
                    **********************

                  Parameter | Meaning
                  ----------------------------------------
                          1 | Intercept
                          2 | age
                          3 | waistcirc
                          4 | hipcirc
                          5 | elbowbreadth
                          6 | kneebreadth
                          7 | anthro3a
                          8 | anthro3b
                          9 | anthro3c
                         10 | anthro4


                     ********************
                     Analysis of Variance
                        Full Analysis
                     ********************

     Source      DF         SSQ  MeanSquare   F_Value  Prob>F

     Model        9 7879.465317 875.4961463 81.346167  0.0000
     Error       61 656.5185195 10.76259868
     C Total     70 8535.983837


                       Goodness of Model Fit

    R-square   0.923088125           Adj R-sq   0.911740472
    MSE        10.76259868           Root MSE   3.280639980
    AIC        177.9232428           BIC        183.1481824
    SBC        200.5500416           C(p)       10.00000000


            ****************************************
            Analysis of Effects and Parameter Estimates
            ****************************************

Variable          DF  Estimate Std_Error TypeII_SS F_Value Prob>F
```

```
Intercept          1 -69.02828  7.516860 907.60792 84.3298 0.0000
age                1  0.019962  0.032213 4.1330554 0.38402 0.5378
waistcirc          1  0.210487  0.067145 105.76540 9.82712 0.0026
hipcirc            1  0.343513  0.080373 196.60010 18.2670 0.0001
elbowbreadth       1 -0.412369  1.022907 1.7491076 0.16252 0.6883
kneebreadth        1  1.757984  0.724952 63.289022 5.88046 0.0183
anthro3a           1  5.742295  5.207524 13.086565 1.21593 0.2745
anthro3b           1  9.866431  5.657864 32.728967 3.04099 0.0862
anthro3c           1  0.387430  2.087463 0.3707365 0.03445 0.8534
anthro4            1 -6.574395  6.489177 11.047131 1.02644 0.3150
```

```
          ********************************************
          Parameter Estimates and Wald Confidence Limits
          ********************************************
```

```
 Variable          DF   Estimate   Lower_CL    UpperCI  Std_Error

 Intercept          1 -69.028276 -83.761051 -54.295501  7.5168600
 age                1  0.0199623 -0.0431743  0.0830989  0.03221314
 waistcirc          1  0.2104872  0.0788858  0.3420885  0.06714480
 hipcirc            1  0.3435131  0.1859850  0.5010413  0.08037298
 elbowbreadth       1 -0.4123687 -2.4172287  1.5924914  1.02290656
 kneebreadth        1  1.7579845  0.3371041  3.1788648  0.72495227
 anthro3a           1  5.7422954 -4.4642639  15.948855  5.20752388
 anthro3b           1  9.8664310 -1.2227788  20.955641  5.65786408
 anthro3c           1  0.3874297 -3.7039232  4.4787827  2.08746335
 anthro4            1 -6.5743949 -19.292948  6.1441577  6.48917669
```

The results agree with those by Bretz et al. (2011).

```
              Parameter Estimates and CIs (Run=1)
              ***********************************
```

```
    Contrast  Coefficient      LowerCI      UpperCI   AsymStdErr

 age         0.019962276 -0.070912780  0.110837331  0.032213136
 waistcirc   0.210487152  0.021067931  0.399906372  0.067144796
 hipcirc     0.343513140  0.116776482  0.570249798  0.080372978
 elbowbread -0.412368664 -3.298045198  2.473307871  1.022906562
 kneebreadt  1.757984466 -0.287146370  3.803115301  0.724952269
 anthro3a    5.742295360 -8.948420308  20.43301103  5.207523876
 anthro3b    9.866431037 -6.094719503  25.82758158  5.657864083
 anthro3c    0.387429743 -5.501420943  6.276280430  2.087463351
 anthro4    -6.574394903 -24.88072474  11.73193493  6.489176691
```

From size of the adjusted probabilities we can see, that only waist and hip circumference are relevant for the regression with amount of bodyfat:

```
            Parameter Estimates and p Values (Run=1)
            ****************************************
               Adjusted Probabilities: SingleStep

    Contrast  Coefficient      t Value      RawProb      AdjProb

age          0.019962276  0.619693645  0.537767410  0.995928190
waistcirc    0.210487152  3.134824482  0.002644229  0.021265627
hipcirc      0.343513140  4.273987975  6.8522e-005  0.000550312
elbowbread -0.412368664 -0.403134244  0.688259006  0.999793932
kneebreadt   1.757984466  2.424965809  0.018286021  0.131734167
anthro3a     5.742295360  1.102692085  0.274492472  0.894596277
anthro3b     9.866431037  1.743843771  0.086224057  0.477904207
anthro3c     0.387429743  0.185598345  0.853375554  0.999999487
anthro4     -6.574394903 -1.013132361  0.314998835  0.929517564


                    Global Test for Run=1
          ChiSquared=732.116 Probability=  0.0000 DF=9
         F=81.3462 Probability=  0.0000 DFnum=9 DFden=61
```

See file `cmat/test/bodyfat.mc` for the file output of the multiple computation results.

11. Computing simultaneous confidence bands of two linear regression models, see Bretz et al. (2011), p. 111:

    There is a nice description at Bretz et al. (2011), p. 112, why it sometimes is better to test the equality of linear regression models not as it is commonly done with an F test but with computing a two-sided confidence band in a restricted region of the covariates x. The F test would require much more that the regression models are the same/different over the entire region of the covariate. This approach refers to a paper by Liu, Jamshidian, Zhang, Bretz & Han (2007). Using the `multcomp()` function in R Bretz et al. (2011) obtain results that are reported to be close to those obtained by Liu et al. (2007) when applied to the SBP data by Kleinbaum, et al. (1998).

```
print "SBP data from multcomp package";
sbp = [" 1  female 144  39, 2  female 138  45, 3  female 145  47, 4  female 162
         5  female 142  46, 6  female 170  67, 7  female 124  42, 8  female 158
         ..................................................................
         61   male 172  62, 62   male 160  51, 63   male 157  48, 64   male 170
         65   male 153  40, 66   male 148  35, 67   male 140  33, 68   male 132
```

```
          69   male 169  61 "];
  cnam = [" N gender sbp age "];
  sbp = cname(sbp,cnam);


  old = [" female male "]; new = [ 1 0 ];
  sbp = replace(sbp,old,new);  print "SBP After", sbp;
```

The contrast matrix specifies that the confidence band is computed only at 31 points for the region $17 \leq age \leq 47$.

```
  age = [ 17:47 ]‘;
  contr = cons(31,1,0.) -> cons(31,1,1.) -> cons(31,1,0.) -> age;
  rnam = [ "age_17": 47 ]; contr = rname(contr,rnam);
  print "Contrasts=", contr;
```

```
Contrasts=
         |         1         2         3         4
-------------------------------------------------
  age_17 |   0.00000   1.00000   0.00000    17.000
  age_18 |   0.00000   1.00000   0.00000    18.000
  age_19 |   0.00000   1.00000   0.00000    19.000
  age_20 |   0.00000   1.00000   0.00000    20.000
.................................................
  age_45 |   0.00000   1.00000   0.00000    45.000
  age_46 |   0.00000   1.00000   0.00000    46.000
  age_47 |   0.00000   1.00000   0.00000    47.000
```

```
  print "Confidence Interval for Regression: see Bretz et al., p. 108";
  print "These are 99 percent confidence intervals";
  /* Class var is eliminated:  class = 2; */
  model = "3 = 2|4"; print "Model=",model;
  optns = [ "print"         5 ,
            "outf"   "sbp.mc" ];
  means = [ "alpha"      0.01 ,
            "ctyp"      "inp" ,
            "eff"       "par" ,
            "print"         2 ,
            "pall"            ,
            "ptyp"      "adj" ];
  list lstopt[2];
  lstopt[1] = optns;
  lstopt[2] = means; print lstopt;
  < gof,parm,ase,conf,cov,typ1,typ3,resi > = glmod(sbp,model,lstopt,class,contr);
```

****
Model Information
****

Number Valid Observations 69
Response Variable Y[3]
N Independent Variables 2

Significance Level: 0.0500000
Design Coding: Full-Rank
No Variable Selection Process

****
Model Effects
****

Intercept + X2 + X4 + X2 * X4

****
Simple Statistics
****

Column	Nobs	Mean	Std Dev	Skewness	Kurtosis
Y[3]	69	148.72464	18.475647	-0.0167297	-0.8070568
X[2]	69	0.4202899	0.4972216	0.3301930	-1.9483227
X[4]	69	46.144928	15.079471	-0.3108232	-0.8447614

****
Parameter Information
****

Parameter	Meaning
1	Intercept
2	gender
3	age
4	gender*age

****
Analysis of Variance
Full Analysis
****

Source	DF	SSQ	MeanSquare	F_Value	Prob>F
Model	3	18010.32871	6003.442904	75.022270	0.0000
Error	65	5201.439404	80.02214468		
C Total	68	23211.76812			

Goodness of Model Fit

R-square	0.775913693	Adj R-sq	0.765571248
MSE	80.02214468	Root MSE	8.945509749
AIC	306.2583077	BIC	308.7430415
SBC	315.1947338	C(p)	4.000000000

****
Analysis of Effects and Parameter Estimates
****

Variable	DF	Estimate	Std_Error	TypeII_SS	F_Value	Prob>F
Intercept	1	110.0385	4.736103	43197.376	539.818	0.0000
gender	1	-12.96144	7.011725	273.44330	3.41710	0.0691
age	1	0.961353	0.096323	7971.0071	99.6100	0.0000
gender*age	1	-0.012030	0.145193	0.5493562	7e-003	0.9342

****
Parameter Estimates and Wald Confidence Limits
****

Variable	DF	Estimate	Lower_CL	UpperCI	Std_Error
Intercept	1	110.03853	100.75594	119.32112	4.7361035
gender	1	-12.961444	-26.704172	0.7812834	7.01172459
age	1	0.9613526	0.7725625	1.1501428	0.09632327
gender*age	1	-0.0120301	-0.2966037	0.2725435	0.14519328

Parameter Estimates and CIs (Run=1)
****

Contrast	Coefficient	LowerCI	UpperCI	AsymStdErr
age_17	-13.16595560	-27.22528867	0.893377476	4.729452317
age_18	-13.17798567	-26.85601831	0.500046965	4.601185762
age_19	-13.19001575	-26.48982391	0.109792403	4.473953933

age_20	-13.20204583	-26.12697549	-0.277116172	4.347847668
age_21	-13.21407591	-25.76777281	-0.660379002	4.222967804
age_22	-13.22610598	-25.41254906	-1.039662905	4.099426421
age_23	-13.23813606	-25.06167502	-1.414597104	3.977348245
age_24	-13.25016614	-24.71556372	-1.784768558	3.856872220
age_25	-13.26219622	-24.37467567	-2.149716767	3.738153254
age_26	-13.27422629	-24.03952457	-2.508928019	3.621364158
age_27	-13.28625637	-23.71068365	-2.861829096	3.506697756
age_28	-13.29828645	-23.38879242	-3.207780479	3.394369178
age_29	-13.31031653	-23.07456392	-3.546069139	3.284618281
age_30	-13.32234661	-22.76879213	-3.875901082	3.177712160
age_31	-13.33437668	-22.47235943	-4.196393941	3.073947635
age_32	-13.34640676	-22.18624351	-4.506570009	2.973653600
age_33	-13.35843684	-21.91152334	-4.805350342	2.877193003
age_34	-13.37046692	-21.64938309	-5.091550739	2.784964201
age_35	-13.38249699	-21.40111328	-5.363880708	2.697401305
age_36	-13.39452707	-21.16810738	-5.620946761	2.614973073
age_37	-13.40655715	-20.95185265	-5.861261650	2.538179805
age_38	-13.41858723	-20.75391320	-6.083261255	2.467547659
age_39	-13.43061730	-20.57590384	-6.285330768	2.403619844
age_40	-13.44264738	-20.41945338	-6.465841387	2.346944276
age_41	-13.45467746	-20.28615707	-6.623197846	2.298057591
age_42	-13.46670754	-20.17751935	-6.755895726	2.257465863
age_43	-13.47873762	-20.09488954	-6.862585694	2.225622999
age_44	-13.49076769	-20.03939556	-6.942139826	2.202908422
age_45	-13.50279777	-20.01188198	-6.993713559	2.189606238
age_46	-13.51482785	-20.01285964	-7.016796052	2.185888290
age_47	-13.52685793	-20.04247354	-7.011242314	2.191803352

Bretz et al.: "...H is rejected (at $\alpha = 0.01$) because the $y = 0$ line is not included in the band for any $20 \leq x \leq 47$. Furthermore, one can infer from the band that females tend to have significantly lower blood pressure than males between the ages of 20 and 47."

Parameter Estimates and p Values (Run=1)

****

Adjusted Probabilities: SingleStep

Contrast	Coefficient	t Value	RawProb	AdjProb
age_17	-13.16595560	-2.783822463	0.007027845	0.016540525
age_18	-13.17798567	-2.864041218	0.005626009	0.013434963
age_19	-13.19001575	-2.948178714	0.004436582	0.010718740
age_20	-13.20204583	-3.036455469	0.003442462	0.008401615
age_21	-13.21407591	-3.129097005	0.002625054	0.006497359

```
age_22     -13.22610598 -3.226330864  0.001964753  0.004923652
age_23     -13.23813606 -3.328382442  0.001441501  0.003660715
age_24     -13.25016614 -3.435469309  0.001035349  0.002658970
age_25     -13.26219622 -3.547793607  0.000727035  0.001924717
age_26     -13.27422629 -3.665532025  0.000498515  0.001330214
age_27     -13.28625637 -3.788822789  0.000333394  0.000882708
age_28     -13.29828645 -3.917749000  0.000217260  0.000584183
age_29     -13.31031653 -4.052317617  0.000137862  0.000354775
age_30     -13.32234661 -4.192433403  8.5160e-005  0.000246006
age_31     -13.33437668 -4.337867220  5.1223e-005  0.000138747
age_32     -13.34640676 -4.488218386  3.0031e-005  7.8129e-005
age_33     -13.35843684 -4.642871307  1.7195e-005  5.6580e-005
age_34     -13.37046692 -4.800947499  9.6441e-006  4.1271e-005
age_35     -13.38249699 -4.961255476  5.3228e-006  3.5622e-005
age_36     -13.39452707 -5.122242829  2.9086e-006  8.2267e-006
age_37     -13.40655715 -5.281957221  1.5861e-006  5.1387e-006
age_38     -13.41858723 -5.438025554  8.7171e-007  2.0730e-006
age_39     -13.43061730 -5.587662849  4.8852e-007  1.1781e-006
age_40     -13.44264738 -5.727723287  2.8293e-007  7.7816e-007
age_41     -13.45467746 -5.854804298  1.7181e-007  3.8618e-007
age_42     -13.46670754 -5.965409160  1.1104e-007  1.1058e-006
age_43     -13.47873762 -6.056163880  7.7504e-008  1.8371e-007
age_44     -13.49076769 -6.124071049  5.9171e-008  1.2073e-007
age_45     -13.50279777 -6.166769867  4.9919e-008  1.1967e-006
age_46     -13.51482785 -6.182762362  4.6835e-008  1.6291e-007
age_47     -13.52685793 -6.171565490  4.8974e-008  1.5988e-007


                  Global Test for Run=1
        ChiSquared=38.2278 Probability=  0.0000 DF=2
      F=19.1139 Probability=  0.0000 DFnum=2 DFden=65
```

4.3 Extensions of `ttest()` Function

The new `ttest()` function is exactly the same as the older `hbttest()` function. The documentation now also mentions that the Scatterthwaite (name used in SAS/STAT manual) modification is also known as Welch's t test (name used in SPSS).

5 New Developments

5.1 The `icdfmv()` Function

quant = icdfmv(dist,prob,mu<,sigma<,par>>)	dist="norm"|"t"

Purpose: Computes the inverse cumulative distribution (percent point) function of the multivariate normal and t distribution.

Input: **dist** The first input argument is a string specifying the distribution. Until now, only the multivariate normal and t distributions are supported.

prob must be either a real scalar or a m vector of probabilities w.r.t. of the normal or t distribution.

mu can be either a missing value, a real scalar ($n = 1$) or an n vector specifying the mean or noncentralities

sigma can be either a missing value, a real scalar (for $n = 1$) or an $n \times n$ positive definite (symmetric) specifying the covariance matrix

optn is either a scalar or a vector specifying:

1. the number *df* of degrees of freedom for the t distribution. Default is $df = 1$ for which the t distribution becomes the normal distribution.
2. not used
3. tail: =0: both-sided, =1: lower, =2: upper
4. if nonmissing and different from zero, the logarithm of the density is returned, default is zero.
5. not used
6. maximum function calls for zero finding (default: 25000)
7. absolute error bound for zero finding (default 0.001)
8. relative error bound for zero finding (default 0)

Note, that either `mu` or `sigma` must be specified nonmissing defining the dimension n.

Output: is either a scalar (for $m = 1$) or an $m > 1$ vector of quantiles (or their logarithms) of the normal or t distribution.

Restrictions:
1. No missing values for `prob` are permitted.
2. The dimensions of `mu`, and `sigma` must be compatible.

Relationships: cdfmv(), pdfmv(), cdf(), pdf()

Examples: 1. Normal distribution:

```
print "Normal: both sided tail";
n = 5; prob = .95;
mu = cons(n); cov = ide(n);
par = cons(8,1,.);
par[3] = 0;
qu11 = icdfmv("norm",prob,mu,cov,par);
print "QU11=",qu11;
```

```
print "Normal: Lower tail";
n = 5; prob = .95;
mu = cons(n); cov = ide(n);
par = cons(8,1,.);
par[3] = 1;
qu12 = icdfmv("norm",prob,mu,cov,par);
print "QU12=",qu12;
```

```
Normal: both sided tail
QU11= 2.5689
```

```
Normal: Lower tail
QU12= 2.3181
```

2. t distribution:

```
print "t Dist: both sided tail";
n = 5; prob = .95;
mu = cons(n); cov = ide(n);
par = cons(8,1,.);
par[1] = ndf = 16; par[3] = 0.;
qu13 = icdfmv("t",prob,mu,cov,par);
print "QU13=",qu13;
```

```
print "t Dist: Lower tail";
n = 5; prob = .95;
mu = cons(n); cov = ide(n);
par = cons(8,1,.);
par[1] = ndf = 16; par[3] = 1;
qu14 = icdfmv("t",prob,mu,cov,par);
print "QU14=",qu14;
```

```
t Dist: both sided tail
QU13= 2.8862
```

```
t Dist: Lower tail
QU14= 2.5578
```

5.2 The `mahalanobis()` Function

< dist,ldet,rmu,rcov > = mahalanobis(x<,mu<,cov<,optn>>>)

Purpose: For an input $m \times n$ matrix $\mathbf{X}$ this function computes either the whole Mahalanobis matrix or only its diagonal. If `mu` is specified as a missing value, it will internally be set by the n vector of means, and if `cov` is specified as a missing value, it will internally be set by the $n \times n$ covariance matrix of the $m \times n$ data set `x`.

$$D = (\mathbf{X} - \mu) inv(\Sigma)(\mathbf{X} - \mu)^T$$

(Compared to the CRAN function with the same name, this function it may also compute the whole $m \times m$ matrix and not only its diagonal. It will also work with missing values for $\hat{\mu}$ and `cov` and not require any input of those.)

Input: **x** must be a real or integer $m \times n$ matrix $\mathbf{X}$ without missing values.

mu can be either a missing value or a n vector of real or integer values without missing values.

cov can be either a missing value or a $n \times n$ matrix of real or integer values without missing values. If `cov` is specified as an n *timesn* matrix it must bei either a symmetric (if optn[3]=0) or a lower or upper triangular matrix (if optn[3]=1).

optn if specified, this must be a numeric scalar or vector of a maximum of five values meaning:

1. amount of printed output (default is 0, no printed output)
2. =0: return matrix (def), !=0: return diag as vector
3. =1: input `cov` is Cholesky factor, otherwise (=0) either missing value or Cholesky factor of `cov` is computed internally
4. if `cov` is a missing value and the $n \times n$ covariance matrix is computed internally, a value different from zero means that biased variances (NOINT option, divisor m) are computed, default is 0, i.e. variances are computed with divisor $m - 1$.
5. specifies the singularity criterion for computing the Cholesky factor of matrix `cov`, default is the value of the runtime option (prespecified as 1.e-8).

Output: **dist** depending on the value of `optn[2]` this is either the entire symmetric m *timesm* matrix of Mahalanobis distances or only its diagonal.

ldet the logarithm of the (positive definite) covariance matrix `cov`

rmu the n vector used in the computations (either input or computed internally)

rcov the $n \times n$ matrix used in the computations (either input or computed internally)

Restrictions: 1. A missing values is returned if the input matrix `x` contains any string, complex, or missing data.

2. The second and third arguments `mu` and `cov` can be mssing. But, if they are specified their dimensions must be compatible with those of the first input argument x.

3. If `cov` is specified as an n *timesn* matrix it must bei either a symmetric (if optn[3]=0) or a lower or upper triangular matrix (if optn[3]=1).

Relationships: dist(), bivar()

Examples: 1. Example 1 of the CRAN function:

```
y = [ 1    1,
      2    2,
      3    3,
      4    1,
      5    2,
      6    3 ];
cov = bivar(y,"cov"); ldet1 = log(det(cov));
print "Covariance Matrix",cov," logdet=", ldet1;
```

```
Covariance Matrix
 S |         1         2
-----------------------
 1 |    3.5000
 2 |   0.80000   0.80000
 logdet= 0.7701
```

```
x  = [ 0 0 ]; mu = [ 1 2 ];
optn[1] = 2;
< dist,ldet2 > = mahalanobis(x,mu,cov,optn);
print "Dist=", dist;
print "Ldet=", ldet2;
```

```
Dist= 5.3704
Ldet= 0.7701
```

2. Example 2 of the CRAN function:

```
x = [ 0.031001961 -0.77632289 -0.669440196,  1.337766211  2.37474210  0.15701477
      0.415779450 -0.18242812 -1.223588321,  0.041308880 -0.76507455 -1.37205056
```

```
      -1.438278750 -0.41824984 -0.994801307,  1.076134627  1.01818171  0.58391582
      ..........................................................................
       0.393589960  1.41327170  0.917855279, -1.608365032 -0.82967274 -0.67828234
      -0.079093570  0.55241070  0.248497380, -1.042356862  0.99240119 -0.02181026
   nc = ncol(x);


   print "Compare with row sums: only compute and return diagonal, optn[2]=1";
   mu = cons(1,nc,0.); cov = ide(nc);
   ldet1 = log(det(cov)); print "Logdet=", ldet1;


Logdet= 0.0000


   optn = cons(5,1,.); optn[1] = 2; optn[2] = 1;
   < dist,ldet2 > = mahalanobis(x,mu,cov,optn);
   rowsum = x[,**]; res = dist -> rowsum; sse = ssq(dist - rowsum);
   print "Dist and rowsum=", res;
   print "SSQerror=", sse, " Ldet=", ldet2;


Dist and rowsum=
    |         1         2
------------------------
  1 |    1.0518    1.0518
  2 |    7.4537    7.4537
  3 |    1.7033    1.7033
  ......................
 98 |    3.7353    3.7353
 99 |   0.37316   0.37316
100 |    2.0718    2.0718


SSQerror= 0.0000 Ldet= 0.0000


   print "specified mu and cov:";
   mu = univar(x,"ari"); cov = bivar(x,"cov");
   print "mu=", mu, " cov=",cov;


mu=
   |     Var_1     Var_2     Var_3
---------------------------------
   |   0.13871   0.04742  -0.08309
```

```
 cov=
 S |         1         2         3
-----------------------------------
 1 |    1.1528
 2 |   0.02141    1.0392
 3 |   0.17933   0.04515   0.80283


   optn = cons(.,5); optn[1] = 2;
   < dist1,ldet3 > = mahalanobis(x,mu,cov,optn);
   print "Dist[1:10,1:10]=", dist1[1:10,1:10];
   print "Ldet=",ldet3;


Dist[1:10,1:10]=
 S |         1         2         3         4         5
-----------------------------------------------------------
 1 |    1.0324
 2 |   -1.9083    6.3641
 3 |   0.97469  -0.17374    1.9026
 4 |    1.5116   -1.8373    2.0447    2.6332
 5 |   0.94274   -2.5756   0.70495    1.5153    2.8849
 6 |   -1.1767    3.0660  -0.72349   -1.5982   -2.0879
 7 |   0.49995   -1.1540   0.65514    1.1018    2.4375
 8 |  -0.65942   0.50868   -1.2450   -1.5101   -1.6986
 9 |   -1.4677    2.2171   -1.4285   -1.9731   0.27306
10 |   0.58115  -0.06965    1.4084    1.5918    1.9560


 S |         6         7         8         9        10
-----------------------------------------------------------
 6 |    1.9387
 7 |   -1.4504    2.3184
 8 |    1.0799   -1.6651    1.5560
 9 |   0.89716   0.90771  -0.01480    3.3837
10 |   -1.0737    2.0500   -1.8843   0.41722    2.3461


Ldet=-0.07688


   print "missing mu and cov: that should be the same";
   optn = cons(.,5); optn[1] = 2;
   < dist2,ldet4,rmu,rcov > = mahalanobis(x,.,.,optn);
   print "rmu=", rmu, " rcov=",rcov;


mu=
```

```
   |         1
-------------
 1 |   0.13871
 2 |   0.04742
 3 |  -0.08309


cov=
 S |         1         2         3
-----------------------------------
 1 |   0.89874
 2 |  -0.00982   0.96477
 3 |  -0.20020  -0.05206    1.2932


   sse = ssq(dist1 - dist2);
   print "Ldet=", ldet4, " sse=", sse;


Ldet=-0.07688 sse= 0.0000
```

5.3 The `multcomp()` Function

< gof,stat,parm > = multcomp(imet,gdat,xdat<,optn<,con<,nams>>>)

Purpose: Performs a set of parameteric an nonparametric test for linear hypotheses among the means of $K > 2$ samples. Currently this function only deals with one-way ANOVA models. For such applications this function is very similar to the `multcomp()` function in CRAN, but also includes some nonparametric tests like that of Kruskal-Wallis.

The following table shows the type of contrasts which are available in functions `multcomp()` and `glmod()`, the number of contrasts C for a specific number K of categories and whether the values of the contrast matrix depend on the Frequencies of a CLASS variable:

Name	**C**	**needs F**
Dunnett (1955) : $K-1$ mult. treatments versus control	$K-1$	n
Tukey (Bofinger, 1985; Hayter, 1990): all pairs	$(K*(K-1))/2$	n
Sequence (Liu et al, 1999)	$K-1$	n
AVE (Bretz, Hothorn, & Westfall, 2011)	K	y
Changepoint (Hirotsu, 1997, 1999)	$K-1$	y
Williams (Bretz, 1999)	$K-1$	y
Marcus (Bretz, 1999)	$(K*(K-1))/2$	y
McDermott and Mudholker (1993)	$K-1$	y
Umbrella-Williams (Bretz and Hothorn)	$(K*(K-1))/2$	y
Grand Mean (Bretz, Hothorn, & Westfall, 2011)	K	y

In addition, both functions, `multcomp()` and `glmod()`, permit the input of an $C \times K$ contrast matrix and in that case the number of rows defines the number of contrasts. The number of columns must then be compatible with the input data.

The more imnportant contrasts are:

Tukey: compare all $(K*(K-1))/2$ pairs of levels with each other

Dunnett: compare one control level with the remaing $K-1$ levels

Williams: testing an order among all K ordinal levels

Marcus: testing an order among all K ordinal levels

All types of p value adjustments for the multiple comparison methods and some additional types can be specified as with the `padjust()` function:

Spec.	padjust()	Method
"adj"	y	"single-step" method, see Bretz et al. (2011), pp. 62
"bon"	y	Bonferroni adjustment
"hol"	y	Holm method (1979)
"hom"	y	Hommel method (1983)
"hoc"	y	Hochberg method (1988)
"bho"	y	Benjamini & Hochberg (1995)
"fdr"	y	false-discovery-rate, same as Benjamini & Hochberg
"bye"	y	Benjamini & Yekutieli (2001)
"bla"	y	Blanchard & Roquain (2008)
"bli"	y	Benjamini & Liu (1999)
"sid"	y	Sidak StepDown (1967)
"fre"	n	"free", step-down Dunnett, see Bretz et al. (2011), p.79
"sha"	n	Shaffer method (1986), see Bretz et al. (2011), pp.93
"wes"	n	Westfall method (1997), see Bretz et al. (2011), pp.93

The number of runs R depends on the number of columns ng of `gdat` and the number of columns nx of `xdat`. If $ng > 1$ then there must be either $nx = 1$ or $nx = ng$ and there will be ng runs treating pairs of corresponding columns. If $nx > 1$ then there must be either $ng = 1$ or $nx = ng$ and there will be nx runs treating pairs of corresponding columns. The following table shows the only four alternatives when $R > 1$:

ng	nx	number runs
1	1	1
R	1	R
1	R	R
R	R	R

Some notes on computer resources: The computation time increases nonlinearly ($O(C^3)$) for a large number C of contrasts with methods 0,10,...19. This could be remarkably evident specially for the computation of the confidence intervals of the estimates which requires the quadrature of the cdf of a C dimensional t distribution (with a method by Genz and Bretz). However, the time necessary to compute the CIs can be even small compared to the time for computing Westfall's method of p value adjustment. All other p adjustment methods, even Shaffer's, need less computer time than the computation of CI's for large C. There should be no problems with the size of the needed memory.

Input: imet must be a string scalar specifying the name of the method (contrast matrix):

string	number	Meaning
"inp"	0	input contrast matrix
"com"	1	common t test
"wel"	2	Welch t test
"dun"	10	Dunnett (1955)
"tuk"	11	Tukey (Bofinger, 1985; Hayter, 1990)
"seq"	12	Sequence (Liu et al, 1999)
"ave"	13	AVE (Bretz, Hothorn, & Westfall, 2011)
"cha"	14	Changepoint (Hirotsu, 1997, 1999)
"wil"	15	Williams (Bretz, 1999)
"mar"	16	Marcus (Bretz, 1999)
"mcd"	17	McDermott and Mudholker (1993)
"umb"	18	Umbrella-Williams (Bretz and Hothorn)
"gme"	19	Grand Mean (Bretz, Hothorn, & Westfall, 2011)
"snk"	40	Student, Newman, and Keul test
"rgw"	41	Ryan-Einot and Gabriel-Welsch test
"gao"	42	(nonparametric) Gao et al. (2008)
"kru"	50	(nonparametric) Kruskal-Wallis test (1952)

For the names of contrast matrices see also the `multst()` function.

gdat must be either an n vector ($mg = 1$) or an $n \times mg$ matrix of integer values.

xdat must be either an n vector or an $n \times mx$ matrix of real or complex values.

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

con This optional argument specifies a $C \times K$ contrast matrix, where C is the number of contrasts. Note, that the first input argument must be specified with `"inp"`.

nams This optional argument specifies a vector of strings

- either for names of the K levels of the contrast matrix
- or for the names of the C contrasts of the contrast matrix, in this case, the `"conam"` option must be specified

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

Option Name	Second Column	Meaning
`"alpha"`	real	level for confidence intervals, def=.05
`"conam"`		`nams` input argument specifies names for the C contrasts and not the levels of `gdat`
`"conf"`	real	confidence level, def=$0.95 = 1 - \alpha$
`"contpop"`	int	number of control sample for Dunnett's contrast

Option Name	Second Column	Meaning
"fixcon"	int	
"ndf"	int	
"ncon"	int	
"noadj"		do not compute adjusted p's
"noci"		do not compute CIs
"noprint"		no printed output
"pall"		print all of: pcon,pcov,pcor,pest,pglob,ppval
"pcon"		print contrast matrix
"pcor"		print correlation matrix of estimates
"pcov"		print covariance matrix of estimates
"peakpop"	int	
"pest"		print coefficition table with CI's
"pglob"		print global F and χ^2 fit
"poplo"	int	
"pophi"	int	
"ppval"		print p value table
"print"	int	amount of printed output, def=2
"psum"		print only the summary output for each run
"ptype"	string	name of p adjustment, def="adj"
	"adj"	"single-step" method, see Bretz et al. (2011), pp. 62
	"bon"	Bonferroni adjustment
	"hol"	Holm method (1979)
	"hom"	Hommel method (1983)
	"hoc"	Hochberg method (1988)
	"bho"	Benjamini & Hochberg (1995)
	"fdr"	false-discovery-rate, same as Benjamini & Hochberg
	"bye"	Benjamini & Yekutieli (2001)
	"bla"	Blanchard & Roquain (2008)
	"bli"	Benjamini & Liu (1999)
	"sid"	Sidak StepDown (1967)
	"fre"	"free" method, step-down Dunnett "see Bretz et al. (2011), p.79
	"sha"	Shaffer method (1986) "see Bretz et al. (2011), pp.93
	"wes"	Westfall method (1997) "see Bretz et al. (2011), pp.93
"seed"	int	seed for random generator
"sing"	real	singularity threshold, def=1.e-8
"tail"	char	alternative: "both", "grea", "less", def="both"

Note, for a large number C of contrasts the Shaffer and especially the Westfall method for p value adjustments can be quite expensive in computer time.

Output: **gof** returns a vector with some scalar results

stat is an $R \times P$, where R is the number of runs, P is either 3 or 4, matrix with the global χ^2 or F test statistics, the global p value, and the degrees of freedom. If there is only one run, $R = 1$, this is a row vector.

parm Only for the methods that use contrast matrices returns an $R*C \times 8$ matrix containing for each run a block of C rows (one row for each contrast) containing in its columns:

1. number of run
2. estimate
3. for REGWQ: the adjusted α, for GAO: the estimated DF, otherwise the lower CI
4. for REGWQ or SNK: the T quantile, otherwise the upper CI
5. asymptotic standard error
6. t value
7. raw p value
8. adjusted p value

Restrictions:

1. The second and third input argument must have the same number of rows.
2. The second input argument must contain integers defining a group level and cannot contain missing values.
3. The third input argument may contain missing values.

Relationships: ttest(), wilcox(), mucomp(), multst()

Examples:

1. Data warpbreaks (Tukey, 1977):

   This example is discussed in detail in chapter 3.3 of Bretz, Hothorn, & Westfall (2011). Some wool samples are subjected to three different levels of tension (light, medium, and high) and the frequency of breaks is reported.

```
print "warpbreaks data: wool levels A, B, tension levels L M H";
print "The Number of Breaks in Yarn during Weaving";
breaks = [ 26 30 54 25 70 52 51 26 67 18 21 29
           17 12 18 35 30 36 36 21 24 18 10 43
           28 15 26 27 14 29 19 29 31 41 20 44
           42 26 19 16 39 28 21 39 29 20 21 24
           17 13 15 15 16 28 ]';
wool  =   [ 1 1 1 1 1 1 1 1 1 1 1 1  1 1 1 1 1 1 1 1 1 1 1 1
            1 1 1 2 2 2 2 2 2 2 2 2  2 2 2 2 2 2 2 2 2 2 2 2
            2 2 2 2 2 2 ]';
tension = [ 1 1 1 1 1 1 1 1 1 2 2 2  2 2 2 2 2 2 3 3 3 3 3 3
            3 3 3 1 1 1 1 1 1 1 1 1  2 2 2 2 2 2 2 2 2 3 3 3
```

```
          3 3 3 3 3 3 ]';
cnam = [" breaks wool tension "];
warpb = breaks -> wool -> tension;
```

We specify Tukey's (see also Bofinger, Hayter) contrast matrix which tests all $(K \times (K-1))/2 = 3$ sample pairs $(1,2),(1,3),(2,3)$ for equality and require raw and *single-step* adjusted p values.

```
imet = "tuk";
optn = [ "print"      2 ,
         "pall"         ,
         "tail"  "both" ,
         "ptyp"   "adj" ];
< gof,pval,parm > = multcomp(imet,tension,breaks,optn);
```

The following results agree with those reported by Bretz, Hothorn, & Westfall (2011) at pages 55 to 66.

```
             Parameter Estimates and CIs (Run=1)
             ***********************************

  Contrast  Coefficient      LowerCI      UpperCI   AsymStdErr

C_1_2      -10.00000000 -19.56021889 -0.439781113  3.960192871
C_1_3      -14.72222222 -24.28244111 -5.162003336  3.960192871
C_2_3      -4.722222222 -14.28244111  4.837996664  3.960192871


           Parameter Estimates and p Values (Run=1)
           ****************************************

              Adjusted Probabilities: SingleStep

  Contrast  Coefficient      t Value      RawProb      AdjProb

C_1_2      -10.00000000 -2.525129540  0.014716971  0.038441143
C_1_3      -14.72222222 -3.717551822  0.000500860  0.001437139
C_2_3      -4.722222222 -1.192422283  0.238614440  0.463067125


          Contrast Matrix: Tukey (Bofinger, Hayter)

                      LEV_1      LEV_2      LEV_3
           C_1_2  -1.0000000          0  1.0000000
           C_1_3  -1.0000000  1.0000000          0
           C_2_3           0 -1.0000000  1.0000000
```

```
                Estimated Variance-Covariance Matrix

                    C_1_2        C_1_3        C_2_3
          C_1_2   15.68312757
          C_1_3   7.841563786  15.68312757
          C_2_3  -7.841563786  7.841563786  15.68312757


                 Estimated Correlation Matrix

                    C_1_2        C_1_3        C_2_3
          C_1_2   1.000000000
          C_1_3   0.500000000  1.000000000
          C_2_3  -0.500000000  0.500000000  1.000000000


                   Global Test for Run=1
          ChiSquared=14.4122 Probability=  0.0007 DF=2
         F=7.20611 Probability=  0.0018 DFnum=2 DFden=51
```

We also print the return arguments:

```
print "Tukey Contrasts: Both-sided, single-step adjusted, p.61,63";
print "GOF=", gof;
print "PVAL=", pval;
print "PARM=", parm;
```

```
GOF=
              |         1
--------------------------
   ReturnCode |   0.00000
     CompTime |   0.00000
   NumberRuns |   1.00000
  GroupColumn |   1.00000
  DataColumns |   1.00000
    MaxLevels |    3.0000
  MaxContrast |    3.0000
  SumContrast |    3.0000
       unused |    .
       unused |    .


Tukey Contrasts: Both-sided, single-step adjusted, p.61,63

PVAL=
   |  Probability       F_stat       DF_num       DF_den
```

```
-----------------------------------------------------
 1 |       0.0018       7.2061       2.0000       51.000


PARM=
       |       Run  Estimate    CI_low    CI_upp
-----------------------------------------------
 C_1_2 |   1.00000  -10.0000   -19.560  -0.43978
 C_1_1 |   1.00000   -14.722   -24.282   -5.1620
 C_1_1 |   1.00000   -4.7222   -14.282    4.8380

       |  AsStdErr   t_value     p_raw  P_adjust
-----------------------------------------------
 C_1_2 |    3.9602   -2.5251    0.0147    0.0384
 C_1_1 |    3.9602   -3.7176    0.0005    0.0014
 C_1_1 |    3.9602   -1.1924   0.23861   0.46307
```

On page 61 Bretz, Hothorn, & Westfall (2011) also report p values adjusted with Bonferroni's method:

```
imet = "tuk";
optn = [ "print"      2 ,
         "pall"         ,
         "tail"  "both" ,
         "ptyp"   "bon" ];
< gof,pval,parm > = multcomp(imet,tension,breaks,optn);


            Parameter Estimates and p Values (Run=1)
            ****************************************

               Adjusted Probabilities: Bonferroni

  Contrast  Coefficient      t Value      RawProb      AdjProb

C_1_2      -10.00000000 -2.525129540  0.014716971  0.044150912
C_1_3      -14.72222222 -3.717551822  0.000500860  0.001502581
C_2_3      -4.722222222 -1.192422283  0.238614440  0.715843320
```

2. Recovery Data by Westfall et al. (1999): Dunnett contrasts

   The following examples show the results of different p value adjustment methods with the recovery data. Compare with Bretz et al. (2011), pp. 73.

```
recovery = [   0      15,  0      13,  0      12,  0      16,
               0      16,  0      17,  0      13,  0      13,
```

```
                0      16,  0      17,  0      17,  0      19,
                0      17,  0      15,  0      13,  0      12,
                0      16,  0      10,  0      17,  0      12,
                1      13,  1      16,  1       9,  2       5,
                2       8,  2       9,  3      14,  3      16,
                3      16,  3      12,  3       7,  3      12,
                3      13,  3      13,  3       9,  3      16,
                3      13,  3      18,  3      13,  3      12,
                3      13 ];
cnam = [" blanket minutes "];
recov = cname(recovery,cnam);
blank = recov[,1]; minut = recov[,2];
```

First we report the results for the "single-step" adjustment, as it is also shown in table 4.1 (the column named "Dunnett"), on page 82:

```
optn = [ "print"       2 ,
         "pall"          ,
         "contpop"     1 ,
         "tail"   "less" ,
         "ptyp"    "adj" ];
< gof,pval,parm > = multcomp("dun",blank,minut,optn);
print "Dunnett Contrasts: Less, single-step adjusted";
print "GOF=", gof;
print "PVAL=", pval;
print "PARM=", parm;
```

```
             Parameter Estimates and CIs (Run=1)
             ***********************************

  Contrast  Coefficient      LowerCI      UpperCI   AsymStdErr

C_2_1      -2.133333333 -1.7977e+308  1.367197874  1.603786810
C_3_1      -7.466666667 -1.7977e+308 -3.966135459  1.603786810
C_4_1      -1.666666667 -1.7977e+308  0.264496884  0.884772752


           Parameter Estimates and p Values (Run=1)
           ****************************************

              Adjusted Probabilities: SingleStep

  Contrast  Coefficient      t Value      RawProb      AdjProb

C_2_1      -2.133333333 -1.330185109  0.095799357  0.241183296
```

```
C_3_1      -7.466666667 -4.655647883  2.0329e-005  5.7670e-005
C_4_1      -1.666666667 -1.883722869  0.033739869  0.092492625


            Contrast Matrix: Dunnett (Hsu, MCB)

                LEV_1      LEV_2      LEV_3      LEV_4
      C_2_1  -1.0000000  1.0000000          0          0
      C_3_1  -1.0000000          0  1.0000000          0
      C_4_1  -1.0000000          0          0  1.0000000


           Estimated Variance-Covariance Matrix

                    C_2_1        C_3_1        C_4_1
        C_2_1   2.572132132
        C_3_1   0.335495495  2.572132132
        C_4_1   0.335495495  0.335495495  0.782822823


            Estimated Correlation Matrix

                    C_2_1        C_3_1        C_4_1
        C_2_1   1.000000000
        C_3_1   0.130434783  1.000000000
        C_4_1   0.236433122  0.236433122  1.000000000


                Global Test for Run=1
        ChiSquared=22.6497 Probability=  0.0000 DF=3
       F=7.54989 Probability=  0.0005 DFnum=3 DFden=37
```

The following is applying the "Bonferroni" adjustment (see also Table 4.1):

```
imet = "dun";
optn = [ "print"       2 ,
         "pall"          ,
         "contpop"     1 ,
         "tail"   "less" ,
         "ptyp"    "bon" ];
< gof,pval,parm > = multcomp("dun",blank,minut,optn);


           Parameter Estimates and p Values (Run=1)
           ****************************************
```

```
              Adjusted Probabilities: Bonferroni

  Contrast  Coefficient      t Value      RawProb      AdjProb

C_2_1      -2.133333333 -1.330185109  0.095799357  0.287398070
C_3_1      -7.466666667 -4.655647883  2.0329e-005  6.0986e-005
C_4_1      -1.666666667 -1.883722869  0.033739869  0.101219606
```

The following are the results for the "free" adjustment, see column "step-down Dunnett" in table 4.1 on page 82:

```
optn = [ "print"       2 ,
         "pall"          ,
         "contpop"     1 ,
         "tail"   "less" ,
         "ptyp"    "fre" ];
< gof,pval,parm > = multcomp("dun",blank,minut,optn);


           Parameter Estimates and p Values (Run=1)
           ****************************************
           Adjusted Probabilities: Free (stepDown)

  Contrast  Coefficient      t Value      RawProb      AdjProb

C_2_1      -2.133333333 -1.330185109  0.095799357  0.095799365
C_3_1      -7.466666667 -4.655647883  2.0329e-005  5.4437e-005
C_4_1      -1.666666667 -1.883722869  0.033739869  0.064035883
```

The following are the results for the more "Shaffer" and "Westfall" adjustments:

```
optn = [ "print"       2 ,
         "pall"          ,
         "contpop"     1 ,
         "tail"   "less" ,
         "ptyp"    "sha" ];
< gof,pval,parm > = multcomp("dun",blank,minut,optn);


           Parameter Estimates and p Values (Run=1)
           ****************************************
            Adjusted Probabilities: Shaffer (1986)

  Contrast  Coefficient      t Value      RawProb      AdjProb
```

```
C_2_1      -2.133333333 -1.330185109  0.095799357  0.095799357
C_3_1      -7.466666667 -4.655647883  2.0329e-005  6.0986e-005
C_4_1      -1.666666667 -1.883722869  0.033739869  0.067479737


optn = [ "print"       2 ,
         "pall"          ,
         "contpop"     1 ,
         "tail"   "less" ,
         "ptyp"    "wes" ];
< gof,pval,parm > = multcomp("dun",blank,minut,optn);


           Parameter Estimates and p Values (Run=1)
           ****************************************
           Adjusted Probabilities: Westfall (1997)

  Contrast  Coefficient      t Value      RawProb      AdjProb

C_2_1      -2.133333333 -1.330185109  0.095799357  0.095799365
C_3_1      -7.466666667 -4.655647883  2.0329e-005  7.7254e-005
C_4_1      -1.666666667 -1.883722869  0.033739869  0.064035883
```

3. Recovery Data by Westfall et al. (1999): Input contrasts
   The following example shows that we can also specify input contrasts as in Bretz et al. (2011), page 76:

```
print "Testing Input Contrast Matrix: Bretz et al., 2011, p. 76";
contr1 = [  -1  1  0  0 ,
            -1  0  1  0 ,
            -1  0  0  1 ];
rnam = [ "b1-b0" "b2-b0" "b3-b0" ];
contr1 = rname(contr1,rnam);


optn = [ "print"       2 ,
         "pall"          ,
         "tail"   "less" ,
         "ptyp"    "adj" ];
< gof,pval,parm > = multcomp("inp",blank,minut,optn,contr1);


             Parameter Estimates and CIs (Run=1)
             ***********************************

  Contrast  Coefficient      LowerCI      UpperCI   AsymStdErr
```

```
b1-b0      -2.133333333 -1.7977e+308  1.366752251  1.603786810
b2-b0      -7.466666667 -1.7977e+308 -3.966581083  1.603786810
b3-b0      -1.666666667 -1.7977e+308  0.264251043  0.884772752


          Parameter Estimates and p Values (Run=1)
          ****************************************
             Adjusted Probabilities: SingleStep

  Contrast  Coefficient      t Value      RawProb      AdjProb

b1-b0      -2.133333333 -1.330185109  0.095799357  0.241195153
b2-b0      -7.466666667 -4.655647883  2.0329e-005  6.7404e-005
b3-b0      -1.666666667 -1.883722869  0.033739869  0.092449900


                  Contrast Matrix: Input

              LEV_1      LEV_2      LEV_3      LEV_4
     b1-b0  -1.0000000  1.0000000          0          0
     b2-b0  -1.0000000          0  1.0000000          0
     b3-b0  -1.0000000          0          0  1.0000000


           Estimated Variance-Covariance Matrix

                  b1-b0        b2-b0        b3-b0
     b1-b0   2.572132132
     b2-b0   0.335495495  2.572132132
     b3-b0   0.335495495  0.335495495  0.782822823


            Estimated Correlation Matrix

                  b1-b0        b2-b0        b3-b0
     b1-b0   1.000000000
     b2-b0   0.130434783  1.000000000
     b3-b0   0.236433122  0.236433122  1.000000000


              Global Test for Run=1
     ChiSquared=22.6497 Probability=  0.0000 DF=3
    F=7.54989 Probability=  0.0005 DFnum=3 DFden=37
```

The following input tests the specification of the contrast matrix as in Bretz et al. (2011), p. 77:

```
print "Testing Input Contrast Matrix: Bretz et al., 2011, p. 77";
contr2 = [  -1  0  1  0 ,
             0 -1  1  0 ,
            -1  0  0  1 ,
             0 -1  0  1 ];
rnam = [ "b2-b0" "b2-b1" "b3-b0" "b3-b1" ];
contr2 = rname(contr2,rnam);


optn = [ "print"       2 ,
         "pall"          ,
         "tail"   "less" ,
         "ptyp"    "adj" ];
< gof,pval,parm > = multcomp("inp",blank,minut,optn,contr2);
```

Parameter Estimates and CIs (Run=1)

Contrast	Coefficient	LowerCI	UpperCI	AsymStdErr
b2-b0	-7.466666667	-1.7977e+308	-3.851610860	1.603786810
b2-b1	-5.333333333	-1.7977e+308	-0.565938915	2.115011412
b3-b0	-1.666666667	-1.7977e+308	0.327677504	0.884772752
b3-b1	0.466666667	-1.7977e+308	4.159474504	1.638280795

Parameter Estimates and p Values (Run=1)

Adjusted Probabilities: SingleStep

Contrast	Coefficient	t Value	RawProb	AdjProb
b2-b0	-7.466666667	-4.655647883	2.0329e-005	6.0671e-005
b2-b1	-5.333333333	-2.521657000	0.008058102	0.027580643
b3-b0	-1.666666667	-1.883722869	0.033739869	0.105537755
b3-b1	0.466666667	0.284851454	0.611326844	0.914992399

Contrast Matrix: Input

	LEV_1	LEV_2	LEV_3	LEV_4
b2-b0	-1.0000000	0	1.0000000	0
b2-b1	0	-1.0000000	1.0000000	0
b3-b0	-1.0000000	0	0	1.0000000
b3-b1	0	-1.0000000	0	1.0000000

```
                Estimated Variance-Covariance Matrix

                 b2-b0        b2-b1        b3-b0        b3-b1
    b2-b0   2.572132132
    b2-b1   2.236636637  4.473273273
    b3-b0   0.335495495            0  0.782822823
    b3-b1             0  2.236636637  0.447327327  2.683963964


                 Estimated Correlation Matrix

                 b2-b0        b2-b1        b3-b0        b3-b1
    b2-b0   1.000000000
    b2-b1   0.659380473  1.000000000
    b3-b0   0.236433122            0  1.000000000
    b3-b1             0  0.645497224  0.308606700  1.000000000


                    Global Test for Run=1
          ChiSquared=22.6497 Probability=  0.0000 DF=3
         F=7.54989 Probability=  0.0005 DFnum=3 DFden=37
```

4. Plant Growth Data from CRAN: (N,count,spr) :

```
  options NOECHO;
#include "..\\tdata\\plantgrowth.dat"
  options ECHO;
  nr = nrow(plantgr); nc = ncol(plantgr);
  print "nr,nc=",nr,nc;


  plantgr = plantgr[,2:3];
  str = [" ctrl trt1 trt2 "];
  ind = [ 0:2 ];
  plant = replace(plantgr,str,ind);
  print plant;
  weight = plant[,1]; grps = plant[,2];


  optn = 2;
  < gof,pval > = multcomp("welch",grps,weight,optn);
  print "Results of Plant Growth data: Welch t test=", pval;


          ****************************************
          Welch t Test (Not assuming equal Variances)
          ****************************************
```

```
          Run      F_stat     df1     df2    Prob  SampleSize
            1 5.180972408       2      17  0.0174 10 10 10


Results of Plant Growth data=
   |  Probability       F_stat       DF_num       DF_den
-------------------------------------------------------
 1 |       0.0174       5.1810       2.0000       17.128


  imet = "commt"; optn = 2;
  < gof,pval > = multcomp(imet,grps,weight,optn);
  print "Results of Plant Growth data: Common t test=", pval;


                     **********************
                     Common t Test (Anova)
                     **********************

          Run      F_stat     df1     df2    Prob  SampleSize
            1 4.846087862       2      27  0.0159 10 10 10


Results of Plant Growth data=
   |  Probability       F_stat       DF_num       DF_den
-------------------------------------------------------
 1 |       0.0159       4.8461       2.0000       27.000
```

5. Data airquality from CRAN: group: (month, x: ozone):

```
  options NOECHO;
#include "..\\tdata\\airqual.dat"
  options ECHO;
  nr = nrow(air); nc = ncol(air);
  print "nr,nc=",nr,nc;


  grps = air[,5]; /* month */
  xdat = air[,1]; /* ozone */


   imet = "krusk"; optn = 2;
   < gof,pval > = multcomp(imet,grps,xdat,optn);
   print "Results of Airquality data=", pval;
```

```
Data airquality from CRAN: group: month, x: ozone
nr,nc= 153 6


                  ***************************
                  Kruskal-Wallis Rank Sum Test
                  ***************************

         Run  Chisquared      df    Prob  SampleSize
           1 29.26657631       4  0.0000  26   9  26  26  29


Results of Airquality data=
   |  Probability   Chisquared          NDF
-------------------------------------------
 1 |       7e-006       29.267       4.0000
```

5.4 The `padjust()` Function

pout = padjust(imet,pin<,par<,covm|pi>>)

Purpose: Performs two kinds of adjustments of multivariate p values:

1. only for the `meth="adj"` adjustment: computes adjusted probabilities from a specified n vector `pin` of t values;
2. for all other `meth` adjustments: computes adjusted probabilities from a specified n vector `pin` of p values.

The `meth="adj"` approach is identical with computing the p value for a given t value using the `cdfmv()` function.

Input: **meth** must be a string scalar specifying the name of the adjustment method :

string	Meaning
"adj"	single-step (covariance matrix must be specified) see Bretz et al. (2011), pp. 62
"bon"	Bonferroni adjustment
"hol"	Holm (1979)
"hom"	Hommel (1983)
"hoc"	Hochberg (1988)
"bho"	Benjamini & Hochberg (1995)
"fdr"	False-Discovery-Rate, same as same as Benjamini & Hochberg
"bye"	Benjamini & Yekutieli (2001)
"bla"	Blanchard & Roquain (2008)
"bli"	Benjamini & Liu (1999)
"sid"	Sidak Step-Down (1967)

For the names of adjustments see also the `multcomp()` function.

pin must be either an n vector or a $m \times n$ matrix

1. only for the `meth="adj"` adjustment: of t values for which the covariance or correlation matrix `covm` and the degrees of feedom (in `par[]`) are specified.
2. for all other `imet` adjustments: of p values, i.e. real values inside the interval $[0, 1]$.

par an optional vector of options

cov|pi optional depending on method `meth`:

- only for the `meth="adj"` adjustment: an optional covariance or correlation matrix used by the t distribution ("adj") adjustment.
- only for the `meth="bla"` adjustment: an n vector of priors (values in $[0, 1]$).

Output: The only output argument `pout` is a vector or matrix of adjusted probabilities of the same size as the input argument `pin`.

Restrictions: 1. The second input argument cannot contain missing values.

2. For any adjustment other than `imet="adj"` the second input argument must contain p values in the range $[0, 1]$.

Relationships: multcomp()

Examples: 1. Example by Bretz, Hothorn, & Westfall (2011, p.39):

```
print "Bretz, Hothorn, Westfall, p. 39";
pin = [ 0.01  0.02  0.022 0.09 ];
pbon = padjust("bon",pin);
phol = padjust("hol",pin);
phom = padjust("hom",pin);
phoc = padjust("hoc",pin);
pbho = padjust("bho",pin);
pbye = padjust("bye",pin);
pfdr = padjust("fdr",pin);


pall = pin‘  -> pbon‘ -> phol‘ -> phom‘ ->
       phoc‘ -> pbho‘ -> pbye‘ -> pfdr‘;
cnam = [" None Bonf Holm Homm Hoch BenH BenY FDR "];
pall = cname(pall,cnam);
print "All adjusted p=", pall;
```

```
All adjusted p=
   |      None      Bonf      Holm      Homm
-------------------------------------------
 1 |   0.01000   0.04000   0.04000   0.03000
 2 |   0.02000   0.08000   0.06000   0.04000
 3 |   0.02200   0.08800   0.06000   0.04400
 4 |   0.09000   0.36000   0.09000   0.09000

   |      Hoch      BenH      BenY       FDR
-------------------------------------------
 1 |   0.04000   0.02933   0.06111   0.02933
 2 |   0.04400   0.02933   0.06111   0.02933
 3 |   0.04400   0.02933   0.06111   0.02933
 4 |   0.09000   0.09000   0.18750   0.09000
```

2. Using the raw p values from the warpbreak example with function `multcomp()`:

```
pin = [ 0.014716971  0.000500860  0.238614440 ];
pnon = padjust("non",pin);
pbon = padjust("bon",pin);
phol = padjust("hol",pin);
phom = padjust("hom",pin);
phoc = padjust("hoc",pin);
pbho = padjust("bho",pin);
pbye = padjust("bye",pin);
pfdr = padjust("fdr",pin);


pall = pnon‘ -> pbon‘ -> phol‘ -> phom‘ ->
       phoc‘ -> pbho‘ -> pbye‘ -> pfdr‘;
cnam = [" None Bonf Holm Homm Hoch BenH BenY FDR "];
pall = cname(pall,cnam);
print "All adjusted p=", pall;
```

```
All adjusted p=
   |      None      Bonf      Holm      Homm
-------------------------------------------
 1 |   0.01472   0.04415   0.02943   0.02943
 2 |   0.00050   0.00150   0.00150   0.00150
 3 |   0.23861   0.71584   0.23861   0.23861


   |      Hoch      BenH      BenY       FDR
-------------------------------------------
 1 |   0.02943   0.02208   0.04047   0.02208
 2 |   0.00150   0.00150   0.00275   0.00150
 3 |   0.23861   0.23861   0.43746   0.23861
```

3. Using randomly generated raw p values for Blanchard & Roquain, Benjamini & Liu, and Sidak Step Down:

```
prb = [ 0.57515505 0.81795385 0.11906543 0.94378902 0.89712998
        0.08633331 0.64485872 0.20584938 0.49217546 0.65584144 ];

pii = [ 0.22092137 0.71898636 0.68858840 0.91186795 0.57831946
        0.07395153 0.61858943 0.04922736 0.48308094 0.63636200 ];


optn = .05 ;  /* alpha */
pbla = padjust("bla",prb,optn,pii);
print "Blanchard-Roquain adjusted p values=",pbla;
```

```
Blanchard-Roquain adjusted p values=
   |         1         2         3         4         5
---------------------------------------------------------
 1 |   1.00000   1.00000   1.00000   1.00000   1.00000

   |         6         7         8         9        10
---------------------------------------------------------
 1 |   1.00000   1.00000   1.00000   1.00000   1.00000


   pbli = padjust("bli",prb);
   print "Benjamini-Liu adjusted p values=",pbli;


Benjamini-Liu adjusted p values=
   |         1         2         3         4         5
---------------------------------------------------------
 1 |   0.69390   0.69390   0.61244   0.69390   0.69390

   |         6         7         8         9        10
---------------------------------------------------------
 1 |   0.59461   0.69390   0.67344   0.69390   0.69390


   psid = padjust("sid",prb);
   print "Sidak Step-Down adjusted p values=",psid;


Sidak Step-Down adjusted p values=
   |         1         2         3         4         5
---------------------------------------------------------
 1 |   0.59461   0.68048   0.84179   0.99129   0.99412

   |         6         7         8         9        10
---------------------------------------------------------
 1 |   0.99435   0.99435   0.99435   0.99435   0.99435
```

5.5 The `pdfmv()` Function

The `pdfmv()` function is similar to the CRAN functions `dmvnorm()` and `dmvt()` except for the fact that the latter returns the logarithm of the density by default.

dens = pdfmv(dist,q<,mu<,sigma<,optn>>>)

Purpose: For given quantile $q \in \mathcal{R}^n$ computes the density for the $n > 1$ multivariate normal and t distribution.

Input: **dist** The first input argument is a string specifying the distribution. Until now, only the multivariate normal and t distributions are supported.

q must be either an m vector or an $m \times n$ matrix of quantiles of the normal or t distribution.

mu can be either a missing value, a real scalar (for $n = 1$) or an n vector specifiyng the mean

sigma can be either a missing value, a real scalar (for $n = 1$) or an $n \times n$ positive definite (symmetric) specifiyng the covariance matrix

optn is either a scalar or a vector specifying:

1. the number *df* of degrees of freedom for the t distribution. Default is $df = 1$ for which the t distribution becomes the normal distribution.
2. not used
3. not used
4. for not missing and different from zero the logarithm of the density is returned, default is zero.

Output: is either a scalar (for $m = 1$) or an m vector of densities (or their logarithms) of the normal or t distribution

Restrictions:
1. No missing values for `q` are permitted.
2. The dimensions of `q`, `mu`, and `sigma` must be compatible.

Relationships: cdfmv(), icdfmv(), cdf(), pdf()

Examples: 1. Normal distribution:

```
x = [ 0. 0. ];
print "d1=", d1 = pdfmv("norm",x);
mu = [ 1. 1. ];
print "d2=", d2 = pdfmv("norm",x,mu);
```

```
d1= 0.1592
d2= 0.05855
```

```
  x = [ 0. 0. ];
  mu = [ 1. 2. ];
  y = [ 2.35225409  0.5264301,
        2.10212008  1.0550063,
       -0.43572431  1.4055291,
       ......................
       -0.40942441  0.7858499,
       -0.04968314  1.5492625,
        0.83930289  1.3704203,
       -0.28400203 -0.2356617 ];
  sigma = [ 4. 2., 2. 3. ];
  print "d3=", d3 = pdfmv("norm",y,mu,sigma);
```

```
d3=
   |         1
--------------
 1 |   0.01410
 2 |   0.02763
 3 |   0.04332
...............
18 |   0.04896
19 |   0.05201
20 |   0.02427
```

2. t distribution: For comparison with the CRAN function `dmvt()` we specify that the logarithms of the densities are returned:

```
  optn = cons(8,1,.);
  optn[1] = ndf = 3; optn[5] = 1;
  x = [ 0. 0. ];
  mu = sigm = .;
  print "d1=", d1 = pdfmv("t",x,mu,sigm,optn);
  mu = [ 1. 1. ];
  print "d2=", d2 = pdfmv("t",x,mu,sigm,optn);
```

```
d1=-1.8379
d2=-3.1149
```

```
  x = [ 0. 0. ];
  mu = [ 1. 2. ];
  y = [ 13.37395234  2.9930656,
        11.52278180  5.7830223,
        -0.35611236  1.1487225,
        ......................
```

```
        -0.03456753  1.0779144,
         0.63907166  1.0434812,
        -0.22580744 -0.1873725 ];
  sigma = [ 4. 2., 2. 3. ];
  print "d3=", d3 = pdfmv("t",y,mu,sigma,optn);
```

```
d3=
   |         1
--------------
 1 |   -10.139
 2 |   -8.7778
 3 |   -3.2449
...............
18 |   -3.1531
19 |   -3.1413
20 |   -3.9544
```

5.6 The `wilcox()` Function

< pval,cint,kwal > = wilcox(xdat<,ydat<,optn>>)

Purpose: Performs the one sample *Wilcoxon Signed Rank Test* and the two sample *Wilcoxon Rank Sum Test*, which is also known as *Mann-Whitney Test*. This function is similar to the `wilcox.test()` function in CRAN, but the computation of approximate confidence interval was improved. In comparison with the SAS/STAT PROC NPAR1WAY the values of the statistics differs, but the probability should agree with that of the normal approximation. The decision between the Signed Rank Test and the Rank Sum Test is done based on input data and the specified options:

1. If `ydat` is specified as a missing value, i.e. $ny = my = 0$, then the Signed-Rank-Test is performed. Here is assumed that `xdat` data contain the pairwise differences of two vectors x and y drawn from one sample.
2. If `ydat` is specified with $ny = nx$ and the `"pair"` option is specified, then the Signed-Rank-Test is performed. In that case, `xdat` and `ydat` are treated row pairwise. That means, observations with missing values either in `xdat` or `ydat` are deleted pairwise. Here, the `wilcox()` function computes the pairwise differences between x and y.
3. In all other situations, $nx > 1$ and $ny > 1$, the Rank-Sum-Test (Mann-Whitney test) is performed and missing values in `xdat` and `ydat` are deleted listwise.

Note, that the two sample Kruskal-Wallis test is computed only in addition to the rank sum (Mann-Whitney) test and not to the signed rank test. For $nx < 50$ and if there are no ties in the data, exact probabilities are computed for both, the signed rank and the rank sum test by default. The `multnorm()` function should be used for computing the Kruskal-Wallis test for $K > 2$ samples.

Input: **xdat** must be either an nx vector ($mx = 1$) or an $nx \times mx$ matrix of real values.

ydat can be either a missing value or an ny vector or an $ny \times my$ matrix of real values.

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

If $mx > 1$ or $my > 1$ then $max(mx, my)$ tests are being performed. If $mx > 1$ and $my > 1$ then $mx = my$ is required and the tests are performed pairwise among corresponding columns of $\mathbf{X}$ and $\mathbf{Y}$.

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

Option Name	Second Column	Meaning
"alpha"	real	significance level for confidence limits, numeric value in the range $(0, 1)$. The default is 0.05.
"corr"		use correction (is default)
"ctype"	string	comparison between two means, default is "eq"
	"gt"	greater
	"gr"	greater
	"le"	less
	"eq"	equal: two sided test
"exact"		exact conf. intervals (not possible for ties in data)
"ftol"	real	an absolute termination criterion for the zero finding algorithm. The default is 10^{-4}.
"maxf"	int	upper range of the number of function evaluations for the zero finding algorithm. The default is 1000.
"mean"	real	μ value of the null hypotheses. The default is $\mu = 0$.
"nocor"		do not use correction
"noex"		exact approx. intervals
"noprint"		no printed output
"pair"		apply pairwise Signed-Rank-Test (nx=ny)
"pkrwal"		print Kruskal-Wallis test
"print"	int	amount of printed output

If neither the `exact` nor the `noex` option is specified: for $nx < 50$ and there are not ties in the data, the exact p value is computed otherwise the normal approximation.

Output: Assuming $N = max(mx, my)$ then N tests are being performed and the three return arguments contain the test results in its rows.

pval is an $N \times 5$ matrix or for $N = 1$ row vector of the results of the Wilcoxon signed rank or rank sum tests with the following values in its columns:

1. p value of test
2. value of the V or W statistic
3. the number of ties in $\mathbf{X}$
4. the value of `nx` used in the test (input size maybe modified by missing values)
5. the value of `ny` used in the test (input size maybe modified by missing values)

cint is an $N \times 3$ matrix or for $N = 1$ row vector with the following values in its columns:

1. difference in location

2. the lower confidence limit
3. the upper confidence limit

kwal is an $N \times 4$ matrix or for $N = 1$ row vector of the results of the Kruskal-Wallis tests with the following values in its columns:

1. p value of test
2. value of the χ^2 statistic
3. the value of `nx` used in the test (input size maybe modified by missing values)
4. the value of `ny` used in the test (input size maybe modified by missing values)

Restrictions: 1. If $mx > 1$ and $my > 1$ the data are processed in a column-pairwise manner and there must be $mx = my$.

2. If `ydat` is not a missing value for the *Wilcoxon Signed Rank Test* the data are processed in a row-pairwise manner and there must be $nx = ny$. For that, the `"pair"` option must be specified.

Relationships: ttest(), multcomp(), hbanova(), mucomp(), multst()

Examples: 1. Wilcoxon Signed Rank Test: Exact CI

```
print "Data by Hollander and Wolfe (1973), p. 29";
xdat = [ 1.83  0.50  1.62  2.48  1.68  1.88 1.55 3.06 1.30 ];
ydat = [ 0.878 0.647 0.598 2.05  1.06  1.29 1.06 3.14 1.29 ];


optn = [ "print"       2 ,
         "pair"          ,
         "ctyp"     "gt" ];
< pval,cint > = wilcox(xdat,ydat,optn);


         ****************************************
         Wilcoxon Signed Rank Test (Mu=0, alfa=0.05)
         ****************************************

    Run    Prob       Stat     Median      LowCI      UppCI
      1  0.0195E 40.000000  0.4600000  0.1750000  1.80e+308
```

The trailing letter 'E' indicates that the exact method for computing the probability and CIs has been used. Note, that even when exact probabilities are specified they may not be computable due to tied data.

2. Wilcoxon Signed Rank Test: Approximate Normal CI

```
print "Data by Hollander and Wolfe (1973), p. 29";
xdat = [ 1.83  0.50  1.62  2.48  1.68  1.88 1.55 3.06 1.30 ];
ydat = [ 0.878 0.647 0.598 2.05  1.06  1.29 1.06 3.14 1.29 ];


optn = [ "print"       2 ,
         "pair"          ,
         "ctyp"     "gt" ,
         "nocorr"        ,
         "noexact"       ];
< pval,cint > = wilcox(xdat,ydat,optn);


         ****************************************
         Wilcoxon Signed Rank Test (Mu=0, alfa=0.05)
         ****************************************

    Run    Prob       Stat     Median      LowCI      UppCI
      1  0.0191A 40.000000  0.4600000  0.1750000  1.80e+308
```

The trailing letter 'A' indicates that the normal approximation method for computing the probability and CIs has been used.

3. Wilcoxon Rank Sum Test: Exact CI

```
print " *** Test Wilcoxon Rank Sum Test: Exact";
print "Data by Hollander and Wolfe (1973), p. 69";
xdat = [ 0.80 0.83 1.89 1.04 1.45 1.38 1.91 1.64 0.73 1.46 ];
ydat = [ 1.15 0.88 0.90 0.74 1.21 ];


optn = [ "print"       2 ,
         "ctyp"     "gt" ];
< pval,cint > = wilcox(xdat,ydat,optn);


           ****************************************
           Wilcoxon Rank Sum Test (Mu=0, alfa=0.05)
           ****************************************

    Run    Prob       Stat    LocDiff      LowCI      UppCI
      1  0.1272E 35.000000  0.3050000 -0.0800000  1.80e+308
```

4. Wilcoxon Rank Sum Test: Approximate Normal CI

```
print " *** Test Wilcoxon Rank Sum Test: Approx";
print "Data by Hollander and Wolfe (1973), p. 69";
```

```
xdat = [ 0.80 0.83 1.89 1.04 1.45 1.38 1.91 1.64 0.73 1.46 ];
ydat = [ 1.15 0.88 0.90 0.74 1.21 ];


optn = [ "print"      2 ,
         "ctyp"    "gt" ,
         "nocorr"       ,
         "noexact"      ];
< pval,cint > = wilcox(xdat,ydat,optn);


           ****************************************
           Wilcoxon Rank Sum Test (Mu=0, alfa=0.05)
           ****************************************

    Run    Prob       Stat    LocDiff      LowCI      UppCI
      1  0.1103A 35.000000  0.3063281 -0.0800000  1.80e+308
```

5. Wilcoxon Rank Sum Test and Kruskal-Wallis Test:

```
print "Data by SAS STAT Manual, p1204";
xdat = [ 228 229 218 216 224 208 235 229 233 219 224 220 232 200 208 232 ];
ydat = [ 186 229 220 208 228 198 222 273 216 198 213 ];

optn = [ "print"       2 ,
         "ctyp"     "eq" ,
         "pkrwal"        ,
         "noexact"       ];
< pval,cint,kwal > = wilcox(xdat,ydat,optn);


           ****************************************
           Wilcoxon Rank Sum Test (Mu=0, alfa=0.05)
           ****************************************

    Run    Prob       Stat    LocDiff      LowCI      UppCI
      1  0.1515A 117.50000  7.0000000 -4.0000000  20.000000
    K-W  0.1446  2.1282102


print "Man-Whitney: pval=", pval;
print "cint=", cint;
print "Kruskal-Wallis: kwal=", kwal;


Man-Whitney: pval=
   |   Probability        W_stat            Mu
```

```
-------------------------------------------
 1 |       0.15154        117.50       0.00000

   |            NX            NY
-------------------------------
 1 |        16.000        11.000


cint=
   |    Median   LowerCI   UpperCI
---------------------------------
 1 |    7.0000   -4.0000    20.000


Kruskal-Wallis: kwal=
   |   Probability    Chisquared           NX            NY
-------------------------------------------------------------
 1 |       0.14461        2.1282        16.000        11.000

   print "Kruskal-Wallis test";
   nx = ncol(xdat); ny = ncol(ydat);
   grps = [ nx#1 ny#2 ];
   xydat = xdat ->ydat;
   imet = "krusk"; optn = [ "print"  2 ];
   < gof,pval > = multcomp(imet,grps,xydat,optn);
   print "Kruskal-Wallis: pval=", pval;


                  ****************************
                  Kruskal-Wallis Rank Sum Test
                  ****************************

               Run  Chisquared      df    Prob  SampleSize
                 1 2.128210175       1  0.1446 16 11


Kruskal-Wallis: pval=
   |  Probability   Chisquared          NDF
-----------------------------------------
 1 |      0.14461       2.1282      1.00000
```

6 The Bibliography

References

[1] Benjamini, Y. & Hochberg, Y. (1995), "Controlling the false discovery rate: a practical and powerful approach to multiple testing", *Journal of the Royal Statistical Society B*, **57**, 289-300

[2] Benjamini, Y. & Liu, W. (1999), "A step-down multiple hypotheses testing procedure that controls the false discovery rate under interdependence", *Journal of Statistical Planning and Inference*, **82**, 163-170.

[3] Benjamini, Y. & Yekutieli, D. (2001), "The control of the false discovery rate in multiple testing under dependency", *Annals of Statistics*, **29**1165-1188.

[4] Blanchard, G. & Roquain, E. (2008), "Two simple sufficient conditions for FDR control", *Electronic Journal of Statistics*, **2**, 963-992.

[5] Bretz, F., Hothorn, T., & Westfall, P. (2011), *Multiple Comparisons using R*, Boca Raton, London, New York: CRC Press.

[6] Dunnett, C. W. (1980), "Pairwise multiple comparisons in the unequal variance case", *Journal of the American Statistical Association*, **75**, 796-800.

[7] Gao, X., et al. (2008), "Nonparametric multiple comparison procedures for unbalanced one-way factorial designs", *Journal of Planning and Inference*, **77**, 2574-2591.

[8] Genz, A. (1992), "Numerical Computation of Multivariate Normal Probabilities", *J. of Computational and Graphical Stat.*, **1**, 141-149.

[9] Genz, A. (1999), "Numerical computation of critical values for multiple comparison problems", Technical Report.

[10] Genz, A. (2000), "Numerical computation of bivariate and trivariate normal probabilities", Technical Report.

[11] Genz, A. (2004), "Numerical computation of rectangular bivariate and trivariate normal and t-probabilities", *Statistics and Computing*, **14**, 251-260.

[12] Genz, A. & Bretz, F. (1999a), "Methods for the computation of multivariate t-probabilities", Technical Report.

[13] Genz, A. & Bretz, F. (1999b), "Numerical computation of multivariate t-probabilities with application to power calculation of multiple contrasts", Technical Report.

[14] Genz, A. & Bretz, F. (2002), "Methods for the computation of multivariate t-probabilities", *Journal of Computational and Graphical Statistics*, **11**, 950-971.

[15] Genz, A. & Bretz, F. (2009), *Computation of multivariate normal and t-probabilities*, Lecture Notes in Statistics, Heidelberg: Springer Verlag.

[16] Hochberg, Y. (1988), "A sharper Bonferroni procedure for multiple tests of significance", *Biometrika*, **75**, 800-803.

[17] Hochberg, Y. & Tamhane, A. C. (1987), *Multiple Comparison Procedures* New York: Wiley.

[18] Holm, S. (1979), "A simple sequentielle rejective multiple test procedure", *Scandinavian Journal of Statistics*, **6**, 65-70.

[19] Hommel, G. (1988), "A stagewise rejective multiple test procedure based on a modified Bonferroni test", *Biometrika*, **75**, 383-386.

[20] Keuls, M. (1952), "The use of the studentized range in connection with an analysis of variance", *Euphytica*, **37**, 112-122.

[21] Kleinbaum, D.G. (1998), Kupper, L.L., Muller, K.E., & Nizam, A. (1998), *Applied Regression Analysis and Other Multivariate Methods*, North Scituate, MA: Duxbury Press.

[22] Liu, W., Jamshidian, M., Zhang, Y., Bretz, F., & Han, X. (2007), "Some new methods for the comparison of two regression models", *Journal of Statistical Planning and Inference*, **137**, p. 57-67.

[23] Miha, A., Hayter, J., & Kuriki, S. (2003), "The evaluation of general non-centerd orthant probabilities", *British Journal of the Royal Statistical Society*, Ser. B, **65**, 223-234.

[24] Sachs, L. (1974), *Angewandte Statistik*, Berlin, Heidelberg, New York: Springer Verlag.

[25] Shaffer, J.P. (1995), "Multiple hypothesis testing", *Annual Review of Psychology*, **46**, 561-576.

[26] Somerville, P.N. (1997), "Multiple testing and simultaneous confidence intervals: calculation of constants", *Computational Statistics and Data Analysis*, **25**, 217-233.

[27] Somerville, P.N. (1998), "Numerical computation of multivariate normal and multivariate-t probabilities over convex regions", *Journal of Computational and Graphical Statistics*.

[28] Somerville, P.N. & Bretz, F. (2001), "FORTRAN 90 and SAS-IML programs for computation of critical values for multiple testing and simultaneous confidence intervals", *Journal of Statistical Software*.

[29] Weber, E. (1972), *Grundriss der biologischen Statistik*, Jena: VEB Gustav Fischer Verlag, 1972.

[30] Wright, S.P. (1992), "Adjusted p-values for simultaneous inference", *Biometrics*, **48**, 1005-1013.