THE UNIVERSITY OF MINNESOTA

Statistics 5401 September 7, 2005

Example of the Use of MacAnova

This document is primarily an example of how MacAnova is used. It illustrates inputting data and macros, both by typing in directly and reading from a file, reading a macro from a terminal and retrieving a macro from a file, computing a mean and a covariance matrix, computing eigenvalues and eigenvectors, and computing one-way ANOVA and MANOVA. It should be read in conjunction with *An Introduction to MacAnova*.

There are many explanatory comments. Read them carefully to understand what is happening. Some of them suggest using `help()` to get more information. Regularly using `help()` or **Help** on the **Help** menu which provides HTML based help is a useful habit to acquire.

MacAnova is available for Windows (and DOS), Macintosh and Linux computers. File `Readme.txt` spells out the special features available. In addition, once you are in MacAnova, `help()` topics `launching`, `macintosh`, `dos_windows` and `wx` contain some of the same material.

The version of MacAnova available for downloading is a new generation (except the version for Classic Macintosh) based on the Carapace framework written by Gary Oehlert in the School of Statistics. It is referred to below as Carapace MacAnova. The documentation has not completely caught up to the new version, at least that which describes the user interface.

There is a comprehensive *Users' Guide* to MacAnova 4.07 which can be downloaded in PDF format from URL `http://www.stat.umn.edu/macanova/documentationug.html`, but this is not essential. It is several versions out of date but may still be useful, especially the sections on writing macros.

Launching MacAnova for Macintosh and Windows

Start MacAnova on a Macintosh by double clicking on the MacAnova icon. You may want to put an alias or shortcut to MacAnova on the desktop or someother convenient place.

Typographical conventions

In the sample session below, *`italicized`* items are typed by the user. **`Bold face`** items are comments added to output lines. This is a convention that will be used in many handouts. There are also many comments starting with "#" on the same lines as MacAnova commands. Anything after "#" is ignored by MacAnova.

Lines in the output starting `Cmd>` are command lines input by the user.

In the Carapace version of MacAnova, you type commands (what is to the right of `Cmd>`) in the *lower* panel of the MacAnova window, followed by Return or Enter. What you typed immediately appears in the upper panel, preceded by `Cmd>` and followed by output produced, if any. In the Macintosh Classic MacAnova and earlier Windows or Linux versions, MacAnova prints `Cmd>` at the end of an output window as a prompt, and you type the command after `Cmd>`, followed by Return or Enter.

Getting help
Carapace MacAnova has a **Help** menu with menu item **Help**. Selecting that brings up a new window similar to a browser that displays help with clickable cross references. When you first select it, you have an index window listing all commands, grouped by help file. There are also clickable Search Key items that provide you with lists of related commands.

Alternatively, you can get help using commands `help()` and `usage()` typed in the command panel. For example, to get help on command `vector()`, type `help(vector)`. You can get a list of help topics in a specific area, say regression, by `help(key:"regression")`. Type `help(key:"?")` for a list of all possible values for `key`.

You can also get very brief information on how a command is used by, say, `usage(vector)`. This doesn't tell what the command does but shows how it is used.

Most help topics have subtopics. This makes it easier to zero in on the specific information you need. See the examples below.

I have often found that when someone is having trouble doing something in MacAnova, they have made no attempt to use the browser based help or `help()` or `usage()`.

Sample MacAnova session

```
                     M A C A N O V A   5.05

An Interactive Program for Statistical Analysis and Matrix Algebra
     For information on major features, type 'help(macanova)'
   For information on linear models and GLM's, type 'help(glm)'
       For latest information on changes, type 'help(news)'
        For information on Unix version, type 'help(unix)'
        Version of August 28, 2005 (Carapace Mac OSX [gcc])
      Type 'help(copyright)' for copyright and warranty info
  Copyright (C) 1994 - 2005 Gary W. Oehlert and Christopher Bingham
       MacAnova home page: http://www.stat.umn.edu/macanova

 This release of MacAnova has a completely rewritten user interface.
You won't see a lot of new capabilities in this first release, but new
capabilities are on the way. The computational core has not changed.

   The changes that you will notice now are that you type commands
into the lower pane of the window, and graphics look somewhat different.
Also, you will find that you can do many things from menus.  We are still
    working on the menus, and future versions will be more complete.

       MacAnova now has some dynamic graphics. However, they're
   still a little buggy (we're working on it!), so you may with to
          make sure that you have saved before using them.

    Please let us know if you find any problems.  Happy computing!
              gary@stat.umn.edu and kb@stat.umn.edu

Cmd> # Anything after a '#' is ignored may be used as a comment
Cmd> spool("") # record session in file;
```

In the Mac, Windows and Linux versions, using"" as a file name brings up a file navigation dialog box in which you can select a file. On a Classic Macintosh you can also select **Spool Output To File** on the **File** menu and select the file in a dialog box.

```
Cmd> x <- matrix(vector(42,52,48,58, 4,5,4,3),4)#create 4 by 2 matrix

Cmd> # vector(...) creates a vector made up of all the arguments

Cmd> # matrix(vec, n) makes a matrix with n rows from vec

Cmd> x   # typing a name prints the value
(1,1)          42            4
(2,1)          52            5
(3,1)          48            4
(4,1)          58            3

Cmd> x + 5   # as does typing an expression
(1,1)          47            9
(2,1)          57           10
(3,1)          53            9
(4,1)          63            8

Cmd> x[1,]    # row 1, that is all columns of row 1
(1,1)          42            4

Cmd> x[vector(2,4),]  # rows 2 and 4 of all columns
(1,1)          52            5
(2,1)          58            3

Cmd> # You can enter a matrix by rows, as follows:

Cmd> matrix(vector(42,4 ,52,5 ,48,4 ,58,3),2)'
(1,1)          42            4
(2,1)          52            5
(3,1)          48            4
(4,1)          58            3

Cmd> # The matrix() command creates a matrix with 2 rows which is

Cmd> # then transposed by the ' operator

Cmd> # You can enter data without typing commas using enter()

Cmd> matrix(enter(42 4  52 5  48 4  58 3),2)'
(1,1)          42            4
(2,1)          52            5
(3,1)          48            4
(4,1)          58            3

Cmd> dim(x) # dim(x) yields dimensions of x
(1)           4            2    4 by 2 matrix

Cmd> n <- nrows(x); print(n) # ncols(x) yields no of rows of x
n:
(1)           4                scalar (single number)

Cmd> # Note that print() prints the name of what is being output

Cmd> xbar <- sum(x)/n; print(xbar) # sum operates on columns
xbar:
(1,1)          50            4
```

```
Cmd> # The (1,1) indicates xbar is a matrix; it has 1 row and hence

Cmd> # is a row vector

Cmd> help(sum,subtopics:"?")  # get help subtopics of topic sum
Available subtopics for topic 'sum' are:
  usage
  dimensions_keyword
  margins_keyword
  structure_argument
  examples
  see_also
Type help(sum:vector("subtopicA","subtopicB",...))
```

NOTE: This command crashed my computer in Macintosh OS X 10.3.8, but will be fixed soon.

```
Cmd> help(sum:"usage") # or help(sum,subtopic:"usage")
Usage (subtopic of 'sum')
sum(x) computes the sum of the elements of a REAL or LOGICAL vector x.

If x is LOGICAL, True is interpreted as 1.0 and False as 0.0 and sum(x)
is the number of elements of x that are True.

If x is a m by n matrix, sum(x) computes a row vector (1 by n matrix)
consisting of the sum of the elements in each column of x.

If x is an array with dimensions n1, n2, n3, ..., y <- sum(x) computes
an array with dimensions 1, n2, n3, ... such that y[1,j,k,...] =
sum(x[i,j,k,...], i=1,...,n1).  This is consistent with what happens
when x is a matrix.  Note: MacAnova3.35 and earlier produced a result
with dimensions n2, n3, ... .

sum(x, squeeze:T) does the same, except the first dimension of the
result (of length 1) is squeezed out unless the result is a scalar.  In
particular, if x is a matrix, sum(x,squeeze:T) will be identical to
vector(sum(x)), and if x is an array, sum(x,squeeze:T) will be identical
to array(sum(x),dim(x)[-1]).

sum(NULL) is NULL.  See topic 'NULL'.

sum(a,b,c,...) is equivalent to sum(vector(a,b,c,...)) if a, b, c, ...
are all vectors.  They must all have the same type, REAL or LOGICAL or
be NULL.  sum(NULL, NULL, ..., NULL) is NULL.

sum(x, silent:T) or sum(a,b,c,...,silent:T) does the same but suppresses
warning messages about MISSING values or overflows.

If all the elements of a vector x are MISSING, sum(x) is 0.0.

sum(x, undefval:U), where U is a REAL scalar does the same, except the
returned value is U when all the elements of x are MISSING.

Cmd> help(sum:"examples")
Subtopic 'examples' of help on 'sum'
Examples:
Examples:
If x is a n by m matrix

  Cmd> r <- x - sum(x)/sum(!ismissing(x))

computes the matrix of the residuals of x[i,j] from the column means.
```

```
When there are no MISSING values, divide by nrows(x)

If x is a n by 4 by 5 array,

  Cmd> r <- x - sum(x)/sum(!ismissing(x))

computes an array with r[i,j,k] = the residual of x[i,j,k] from the mean
of all x[i,j,k] with the same values for j and k.  That is, it treats x
analogously to a 4 by 5 array of vectors of length n.  See topic
'arithmetic'.  When there are no MISSING values, divide by dim(x)[1].

If z is a vector of integers,

  Cmd> sum(z == run(min(z),max(z))')

computes a row vector giving the frequency distribution of the values in
z.

  Cmd> a # 2 by 2 by 3 array with labels
                 C1          C2          C3
  A1 B1           9           5           7
     B2           9          12          11
  A2 B1           4          11          10
     B2          11          15           9

  Cmd> sum(a,dimensions:2) # sum over dimension 2; 2 by 1 by 3 result
                  C1          C2          C3
  A1 (1)          18          17          18
  A2 (1)          15          26          19

  Cmd> sum(a,margins:vector(1,3),squeeze:F) # same as preceding
                  C1          C2          C3
  A1 (1)          18          17          18
  A2 (1)          15          26          19

  Cmd> sum(a,dimensions:2,squeeze:T) # sum over dim 2; 2 by 3 result
              C1          C2          C3
  A1          18          17          18
  A2          15          26          19

  Cmd> sum(a,margins:vector(1,3)) # same as preceding
              C1          C2          C3
  A1          18          17          18
  A2          15          26          19
```

Now back to actually doing stuff instead of reading help information.

```
Cmd> dim(xbar) # xbar is a 1 by 2 matrix (a row vector)
(1)           1           2

Cmd> # Compute column sums using matrix operations instead of sum()

Cmd> rep(1,4)' %*% x # 1'X ,  rep(1,4) is same as vector(1,1,1,1)
(1,1)         200         16

Cmd> # The vector rep(1,4) is equivalent to a 4 by 1 matrix of 1's
```

```
Cmd> sum(x) # A more direct way to compute columns sums
(1,1)         200          16

Cmd> describe(x)# gives summary info about variables in columns of y
component: n                          The value of describe is a
(1)            4            4         structure with components 'n',
component: min                        'min', etc.
(1)           42            3
component: q1
(1)           45          3.5          Lower quartiles
component: median
(1)           50            4
component: q3
(1)           55          4.5          Upper quartiles
component: max
(1)           58            5
component: mean                        Note this checks with value above
(1)           50            4          but it's a vector, not a row vector
component: var
(1)       45.333      0.66667

Cmd> describe(x)$mean   # access component 'mean' of describe(x)
(1)           50            4

Cmd> describe(x)[7]    # another way to access it as component 7
(1)           50            4

Cmd> describe(x, mean:T) # yet another way, computing only the mean
(1)           50            4

Cmd> res <- x - xbar  # subtracts the row vector from every row of y

Cmd> # This works only when xbar is a row vector.

Cmd> res <- x - describe(x,mean:T) # gives an error message
ERROR: Dimension mismatch for - near
res <- x - describe(x,mean:T) # gives an error message

Cmd> # This didn't work since describe(x,mean:T) is a column vector

Cmd> # res <- x - describe(x,mean:T)' would work

Cmd> print(res,sum_of_res:sum(res))# sum down columns of res are zero
res:
(1,1)          -8            0  Residuals from column means
(2,1)           2            1
(3,1)          -2            0
(4,1)           8           -1
sum_of_res:                    A 1 by 2 matrix or row vector
(1,1)           0            0  Value of sum(res)

Cmd> # Now compute sample covariance matrix using matrix algebra

Cmd> ssp <- res' %*% res; ssp  # (x-xbar)' %*% (x-xbar)
(1,1)         136           -6
(2,1)          -6            2

Cmd> # You could use res %c% res instead of res' %*% res

Cmd> sn <- ssp/n# sample (biased) covariance matrix, divisor of n
```

```
Cmd> sn
(1,1)          34        -1.5
(2,1)        -1.5         0.5

Cmd> df <- n - 1 # Degrees of freedom in variance estimate

Cmd> df
(1)            3

Cmd> s <- ssp/df # Sample (unbiased) covar. matrix; divisor is n-1

Cmd> s
(1,1)      45.333          -2      Diagonal elements check with
(2,1)          -2     0.66667      variances from describe() above

Cmd> tabs(x,covar:T) # "black box way to do the preceding"
(1,1)      45.333          -2
(2,1)          -2     0.66667

Cmd> eigen(s)  # Compute (ordinary) eigenvalues & eigenvectors of s
component: values                  Note that this yields a
(1)      45.423     0.57729        structure with two components.
component: vectors                 values: a vector of eigenvalues.
(1,1)      -0.999   -0.044642      vectors: a matrix whose columns
(2,1)    0.044642      -0.999      are the eigenvectors

Cmd> eigenvals(s)  # Compute just the eigenvalues
(1)      45.423     0.57729

Cmd> u <- eigen(s)$vectors # Save the eigenvectors as u

Cmd> u
(1,1)      -0.999   -0.044642      The columns of u are the eigen-
(2,1)    0.044642      -0.999      vectors

Cmd> diag(s) # diagonal elements of s, a vector
(1)      45.333     0.66667        Sample variances

Cmd> trace(s) # Trace is sum of diagonals
(1)          46                    45.333 + 0.66667

Cmd> sum(diag(s)) # same as preceding
(1)          46

Cmd> sum(eigenvals(s)) # trace(s) is also the sum of the eigenvalues
(1)          46                    45.423 + 0.57729

Cmd> det(s) # Determinant
(1)      26.222                    (45.333)*(.66667) - (-2)*(-2)

Cmd> prod(eigenvals(s)) # Determinant is product of eigenvalues
(1)      26.222                    45.423*0.57729

Cmd> # Create a diagonal matrix of eigenvalues using dmat()

Cmd> d <- dmat(eigenvals(s)) # create diagonal matrix

Cmd> d
(1,1)      45.423           0
(2,1)           0     0.57729
```

```
Cmd> u %*% d %*% u'  # Equivalent to u %*% d %C% u  (U D U')
(1,1)      45.333         -2
(2,1)          -2    0.66667

Cmd> # Note that this is s, except possibly for rounding error

Cmd> u' %*% s %*% u  # Equivalent to u %c% s %*% u  (U' S U)
(1,1)      45.423 -9.3171e-17
(2,1) -3.7537e-17    0.57729

Cmd> # Note that this is d, except for rounding error
```

The off diagonal elements should be mathematically zero. Because of rounding error they are very small non-zero numbers.

```
Cmd> listbrief() # See all data and macros in memory
adddatapath  CONSOLE    fromclip     MACROFILE  regs         toclip
addmacrofile DATAFILE   getdata      MACROFILES res          twotailt
alltrue      DATAPATH   getmacros    makecols   resid        u
anovapred    DATAPATHS  GRAPHWINDOWS makefactor resvsindex   vboxplot
anytrue      DEGPERRAD  haslabels    model      resvsrankits VERSION
boxcox       DELTAT     hasnotes     n          resvsyhat    x
breakif      df         hist         PI         rowplot      xbar
CLIPBOARD    E          HOME         readcols   s            yhat
colplot      enter      l            redo       sn
console      enterchars LASTLINE     regcoefs   ssp
```

Note: Your list almost certainly will not match this one exactly.

Items are listed in alphabetical order, ignoring case. Some are variables you created; others are predefined constants and macros, or items created by commands in file `MacAnova.ini`, which, it exists, is automatically excuted when you start up MacAnova. Type `help(launching)` and `help(customize)` for information about `MacAnova.ini`.

```
Cmd> listbrief(real:T)  # numerical objects only
DEGPERRAD    E            PI           sn           x
DELTAT       l            res          ssp          xbar
df           n            s            u

Cmd> listbrief(macro:T) # only macros
adddatapath  breakif    getdata   makecols   regs         toclip
addmacrofile colplot    getmacros makefactor resid        twotailt
alltrue      console    haslabels model      resvsindex   vboxplot
anovapred    enter      hasnotes  readcols   resvsrankits yhat
anytrue      enterchars hist      redo       resvsyhat
boxcox       fromclip   LASTLINE  regcoefs   rowplot

Cmd> # Now read in data from file JWdata5.txt, from Table 11.5 of J&W
```

```
Cmd> jw115 <- read("","t11_05")#find file JWdata5.txt
T11_05    150    5 format   Descriptive comments on file
) Data from Table 11.5 p. 657-658 in
) Applied Mulivariate Statistical Analysis, 5th Edition
) by Richard A. Johnson and Dean W. Wichern, Prentice Hall, 2002
) These data were edited from file T11-5.DAT on disk from book
) The variety number was moved to column 1
) Measurements on petals of 4 varieties of Iris.  Originally published
) in R. A. Fisher, The use of mltiple measurements in taxonomic,
) problems Annals of Eugenics, 7 (1936) 179-198
) Col. 1: variety number (1 = I. setosa, 2 = I. versicolor,
)                         3 = I. virginica)
) Col. 2: x1 = sepal length
) Col. 3: x2 = sepal width
) Col. 4: x3 = petal length
) Col. 5: x4 = petal width
) Rows 1-50:    group 1 = Iris setosa
) Rows 51-100:  group 2 = Iris versicolor in
) Rows 101-150: group 3 = Iris virginica in
Read from file "TP1:Stat5401:Data:JWData5.txt"
```

Note: Because of a bug, Carapace MacAnova does not print the name of the file read from.

Alternatively, you can set `CHARACTER` variable `DATAFILE` to the full path name of the file and use `getdata()` to retrieve a data set. Here's what it looks like on my Macintosh.

```
Cmd> DATAFILE <- getfilename() # find JWData5.txt

Cmd> DATAFILE # full name of data file
(1) "TP1:Stat5401:Data:JWData5.txt"

Cmd> jw115 <- getdata(t11_05, quiet:T) #quiet:T suppresses comments
Read from file "TP1:Stat5401:Data:JWData5.txt"
```

Now that the data has been read in, here are some things you can do with then.

```
Cmd> dim(jw115) # find the number of rows and columns
(1)        150          5       150 rows and 5 columns

Cmd> jw115[vector(49,50,51,52),]  # Look at rows 49 through 52
(1,1)            1        5.3         3.7         1.5         0.2
(2,1)            1          5         3.3         1.4         0.2
(3,1)            2          7         3.2         4.7         1.4
(4,1)            2        6.4         3.2         4.5         1.5

Cmd> varieties <- jw115[,1] # make column with variety numbers

Cmd> # These work because 1st subscript indexes rows or cases, 2nd

Cmd> # indexes columns or variables

Cmd> y <- jw115[varieties==2,-1] # extract rows with versicolor data

Cmd> # Subscript of -1 omits column 1 and takes the rest

Cmd> # varieties == 2 is True only for versicolor cases and using

Cmd> # it as subscript 1 selects the rows for which it is True

Cmd> dim(y)    # n = 50 cases with p = 4 variables
(1)         50          4
```

```
Cmd> # Enter a simple macro to compute covariance matrix

Cmd> mycovar <- macro("@y <- $1
 @n <- nrows(@y)
 @res <- @y - sum(@y)/@n
 @res' %*% @res / (@n-1)", dollars:T)#text of macro is inside "..."

Cmd> mycovar # look at text of macro
(1) "@y$$ <- $1
 @n$$ <- nrows(@y$$)
 @res$$ <- @y$$ - sum(@y$$)/@n$$
 @res$$' %*% @res$$ / (@n$$-1)"
```

Variables that start with `@` are temporary and will be deleted automatically. Because I used `dollars:T` on the `macro()` command, `$$` gets added to every temporary variable name. This ensures variables used in different macros don't "collide," that is, end up with the same name. The first macro argument will be substituted for `$1`. Because `@ans' %*% @ans / (@n-1)` is the last expression in the macro, `mycovar(y)` returns its value, the covariance matrix. `%*%` is the MacAnova matrix multiplication operator.

```
Cmd> mycovar(y)  # Use it just like a function
(1,1)     0.26643    0.085184      0.1829     0.05578
(2,1)    0.085184    0.098469    0.082653    0.041204
(3,1)      0.1829    0.082653     0.22082    0.073102
(4,1)     0.05578    0.041204    0.073102    0.039106

Cmd> # This is equivalent to substituting 'y' for '$1' in macro text

Cmd> s <- mycovar(y) # save covariance matrix as s

Cmd> write(s) # write() gives more decimals
s:
(1,1)    0.266432653   0.0851836735    0.182897959   0.0557795918
(2,1)   0.0851836735   0.0984693878   0.0826530612   0.0412040816
(3,1)    0.182897959   0.0826530612    0.220816327   0.0731020408
(4,1)   0.0557795918   0.0412040816   0.0731020408   0.0391061224
```

As the Perl saying goes, *There's more than one way to do it*. Here I used `tabs()` to compute the covariance matrix:

```
Cmd> tabs(y,covar:T)
(1,1)     0.26643    0.085184      0.1829     0.05578
(2,1)    0.085184    0.098469    0.082653    0.041204
(3,1)      0.1829    0.082653     0.22082    0.073102
(4,1)     0.05578    0.041204    0.073102    0.039106
```

`covar()` is a more elaborate macro in file MacAnova.mac.txt which computes a covariance and mean vector. It is automatically retrieved when you use it.

```
Cmd> cov <- covar(y) # use covar()
WARNING: searching for unrecognized macro covar near cov <- covar(

Cmd> cov # the result is a structure with 3 components
component: n            Sample size
(1)           50
component: mean         Sample mean as row vector
(1,1)        5.936        2.77         4.26        1.326
component: covariance  Variance/Covariance matrix
(1,1)      0.26643    0.085184       0.1829      0.05578
(2,1)     0.085184    0.098469     0.082653     0.041204
(3,1)       0.1829    0.082653      0.22082     0.073102
(4,1)      0.05578    0.041204     0.073102     0.039106

Cmd> compnames(cov) # displays names of components
(1) "n"
(2) "mean"
(3) "covariance"

Cmd> cov$mean # extract component mean
(1,1)        5.936        2.77         4.26        1.326

Cmd> cov[2] # extract component 2 ; same as cov$mean
(1,1)        5.936        2.77         4.26        1.326

Cmd> # Change vector varieties into a factor (categorical variable)

Cmd> varieties <- factor(varieties) # essential for anova(), manova()

Cmd> y <- jw115[,-1] # 150 by 4 matrix = everything but column 1

Cmd> makecols(y,sepal_len,sepal_wid,petal_len,petal_wid)

Cmd> # makecols creates 4 vectors from the columns of y

Cmd> listbrief(real:T)
d            jw115        PI           sepal_wid    varieties
DEGPERRAD    n            res          sn           x
DELTAT       petal_len    s            ssp          xbar
df           petal_wid    sepal_len    u            y

Cmd> anova("petal_wid=varieties")  # typical one-way ANOVA
Model used is petal_wid=varieties
                DF          SS          MS
CONSTANT         1      215.76      215.76
varieties        2      80.413      40.207
ERROR1         147      6.1566    0.041882

Cmd> anova("petal_wid=varieties",fstats:T) # f-stats and P values too
Model used is petal_wid=varieties
                DF          SS          MS           F     P-value
CONSTANT         1      215.76      215.76  5151.66322           0
varieties        2      80.413      40.207   960.00715           0
ERROR1         147      6.1566    0.041882

Cmd> secoefs() # get coefficients and standard errors.
component: CONSTANT
  component: coefs                             value of CONSTANT coefficient
(1)      1.1993
```

```
  component: se                               value of its standard error
(1)     0.01671
component: varieties
  component: coefs                            values of variety effects
(1)    -0.95333     0.12667     0.82667    they sum to zero
  component: se                               values of their standard errors
(1)    0.023631    0.023631    0.023631
```

```
Cmd> listbrief (real:T,char:T) # list REAL & CHARACTER variables
CLIPBOARD     DEPVNAME      jw115         RESIDUALS     TERMNAMES
CONSOLE       DF            MACROFILE     s             u
d             df            MACROFILES    sepal_len     varieties
DATAFILE      dpdata        n             sepal_wid     VERSION
DATAPATH      HELPFILES     petal_len     sn            x
DATAPATHS     HELPINDICES   petal_wid     SS            xbar
DEGPERRAD     HII           PI            ssp           y
DELTAT        HOME          res           STRMODEL
```

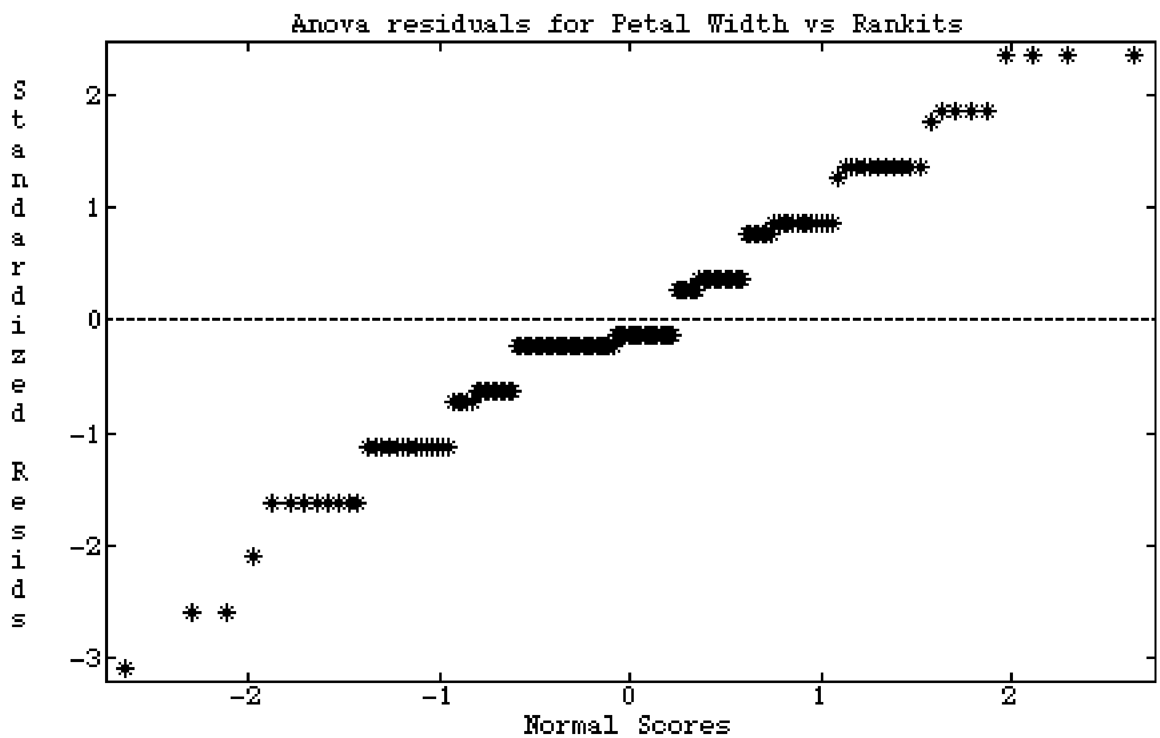
```
Cmd> # DEPVNAME, DF, HII, RESIDUALS, STRMODEL and TERMNAMES are
```

```
Cmd> # so-called "side-effects" variables created by anova()
```

```
Cmd> hconcat(DF,SS,SS/DF) # Type help(hconcat)
                  (1)         (2)         (3)
CONSTANT            1      215.76      215.76
varieties           2      80.413      40.207
ERROR1            147      6.1566    0.041882
```

```
Cmd> # hconcat() puts vectors or matrices side by side
```

```
Cmd> resvsrankits(title:\
"Anova residuals for Petal Width vs Rankits")
```



The graph was copied from a graphics window and pasted into a word processor document on a Macintosh. `resvsrankits()` is a predefined macro. Such a plot is often used as a diagnostic technique to assess whether the residuals are approximately normal; if there is a lot of curvature in the plot, it would suggest non-normality.

Multivariate Analysis of Variance (MANOVA) is a generalization of univariate ANOVA. In place of each ANOVA sum of squares, MANOVA computes a matrix whose elements are sums of squares (on the diagonal) or sums of products (off the diagonal). The basic MacAnova function for doing this is `manova()`, used much like `anova()`.

```
Cmd> manova("y=varieties")  # full-fledged multivariate ANOVA
Model used is y=varieties
WARNING: summaries are sequential
                     SS and SP Matrices
                DF        [4,4] elements of matrices are
CONSTANT         1        ANOVA SS computed above
            SepLen      SepWid      PetLen      PetWid
SepLen      5121.7      2679.8      3293.9      1051.2
SepWid      2679.8      1402.1      1723.4      550.01
PetLen      3293.9      1723.4      2118.4      676.06
PetWid      1051.2      550.01      676.06      215.76
varieties        2
            SepLen      SepWid      PetLen      PetWid
SepLen      63.212     -19.953      165.25      71.279
SepWid     -19.953      11.345      -57.24     -22.933
PetLen      165.25      -57.24       437.1      186.77
PetWid      71.279     -22.933      186.77      80.413
ERROR1         147
            SepLen      SepWid      PetLen      PetWid
SepLen      38.956       13.63      24.625       5.645
SepWid       13.63      16.962      8.1208      4.8084
PetLen      24.625      8.1208      27.223      6.2718
PetWid       5.645      4.8084      6.2718      6.1566
```

```
Cmd> secoefs() # get coefficients and their standard errors
component: CONSTANT
  component: coefs  Constant terms for all 4 variables
         SepLen      SepWid      PetLen      PetWid
(1)      5.8433      3.0573       3.758      1.1993
  component: se     Their standard errors
         SepLen      SepWid      PetLen      PetWid
(1)    0.042032    0.027735    0.035137     0.01671
component: varieties
  component: coefs  Variety effects for all 4 variables
         SepLen      SepWid      PetLen      PetWid
(1)    -0.83733     0.37067      -2.296    -0.95333
(2)    0.092667    -0.28733       0.502     0.12667
(3)     0.74467   -0.083333       1.794     0.82667
  component: se     Their standard errors
         SepLen      SepWid      PetLen      PetWid
(1)    0.059443    0.039224    0.049691    0.023631
(2)    0.059443    0.039224    0.049691    0.023631
(3)    0.059443    0.039224    0.049691    0.023631
```

```
Cmd> predtable(seest:T,sepred:T)# Cell means & se's for all 4 vars
component: estimate    Estimated values of group mean vectors
(1,1)       5.006       3.428       1.462       0.246
(2,1)       5.936        2.77        4.26       1.326
(3,1)       6.588       2.974       5.552       2.026
component: SEest       Their standard errors
(1,1)    0.072802    0.048039    0.060858    0.028942
(2,1)    0.072802    0.048039    0.060858    0.028942
(3,1)    0.072802    0.048039    0.060858    0.028942
component: SEpred      Standard errors of prediction
(1,1)     0.51991     0.34307     0.43462     0.20669
(2,1)     0.51991     0.34307     0.43462     0.20669
(3,1)     0.51991     0.34307     0.43462     0.20669

Cmd> # Without seest:T and sepred:T predtable() just computes means

Cmd> contrast("varieties",vector(1,-1,0)) # compare 1-st two groups
component: estimate        Estimated contrast for each variable
(1,1)       -0.93       0.658      -2.798       -1.08
component: ss              Sums Squares and products for contrast
(1,1,1)      21.623     -15.299      65.054       25.11
(1,2,1)     -15.299      10.824     -46.027     -17.766
(1,3,1)      65.054     -46.027      195.72      75.546
(1,4,1)       25.11     -17.766      75.546       29.16
component: se              Std Error of contrast for each variable
(1,1)     0.10296    0.067938    0.086067     0.04093

Cmd> # save everything in a file so that we could restart at same

Cmd> # place on a future run, using restore()

Cmd> save("") # on Mac or Windows, dialog box lets you name file
Workspace saved on file Example.sav Name selected in dialog
```

On a future run, or later in the same run, `restore("")` would restore things just as they are now. Type `help(save, restore)` to get full information.

In windowed versions you can also use **Save Workspace** on the **File** menu.

```
Cmd> quit # terminate MacAnova
```

In windowed versions, you can select **Quit** on the **File** menu. You will be asked if you want to save your workspace (all the variables and macros) and the command/output window. To bypass these queries, you can type `quit(F)` at the prompt.