Displays for Statistics 5401

Lecture 2

September 9, 2005

Christopher Bingham, Instructor

612-625-1024, kb@umn.edu
372 Ford Hall

Class Web Page
http://www.stat.umn.edu/~kb/classes/5401

Some ways of looking at multivariate data

A **scatterplot matrix** combines scatter plots of every variable against every other in a single display. You can make one using `plotmatrix()`

I will illustrate it on the famous Fisher iris data in J&W Table 11.5 (data set `T11_05` in file `JWData5.txt`.

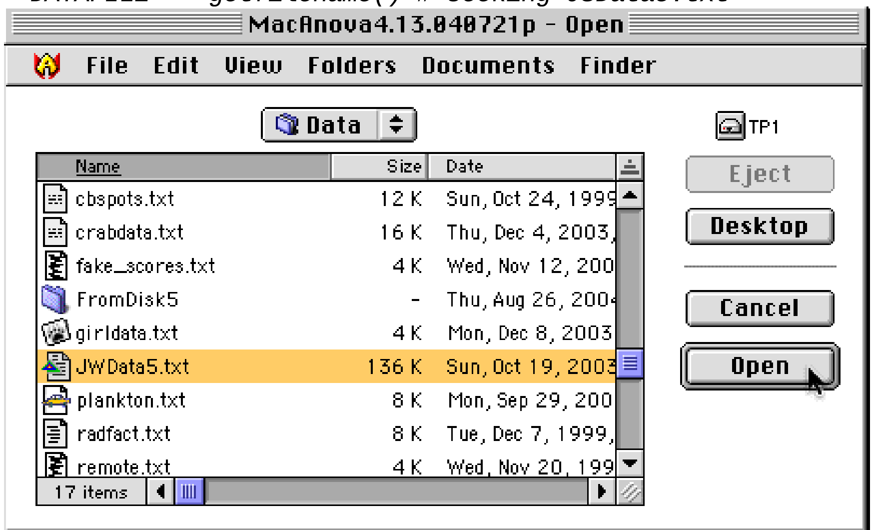
You can read this directly using `read()` (or `matread()`).

Or, provided `CHARACTER` variable `DATAFILE` contains the file name, upi can read it using `getdata()` .

You can ensure this is the case by using `getfilename()`. This brings up a file navigation dialog box in which you select `JWData5.txt`

This is what it looks like in the Classic Macintosh MacAnova.

```
Cmd> DATAFILE <- getfilename() # seeking JSData5.txt
```



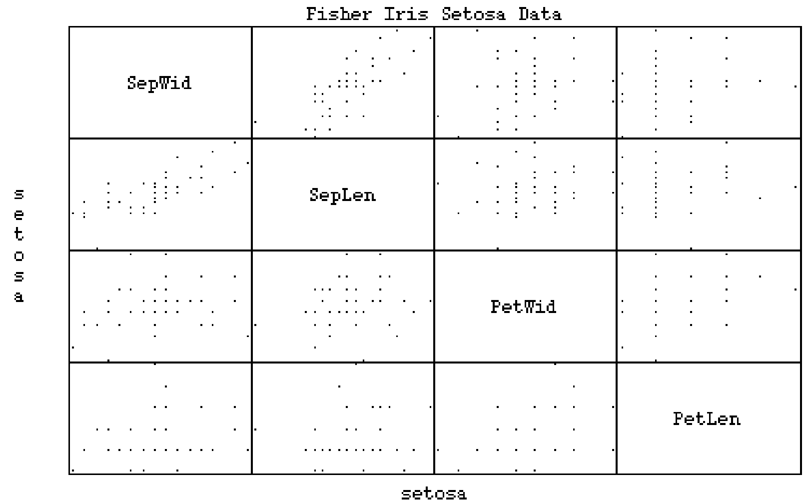

```
Cmd> DATAFILE # now contains the full "path name"
(1) "TP1:Stat5401:Stat5401F04:Data:JWData5.txt"

Cmd> irisdata <- getdata(t11_05,quiet:T) # get T11_05
Read from file "TP1:Stat5401:Data:JWData5.txt"

Cmd> list(irisdata) # dimensions of the matrix
irisdata        REAL    150    5

Cmd> head(irisdata,5) #same as irisdata[run(5),]
          Variety      SepLen      SepWid      PetLen      PetWid
(1)             1         5.1         3.5         1.4         0.2
(2)             1         4.9           3         1.4         0.2
(3)             1         4.7         3.2         1.3         0.2
(4)             1         4.6         3.1         1.5         0.2
(5)             1           5         3.6         1.4         0.2

Cmd> variety <- irisdata[,1] # column 1 contains 1, 2 or 3

Cmd> setosa <- irisdata[variety == 1,-1]#group 1 w/o col. 1
```

`variety == 1` selects variety 1 (setosa).

Statistics 5401 Lecture 2 September 9, 2005

Now make scatterplot matrix using **`plotmatrix()`**.

```
Cmd> plotmatrix(setosa,title:"Fisher Iris Setosa Data")
```



Vocabulary

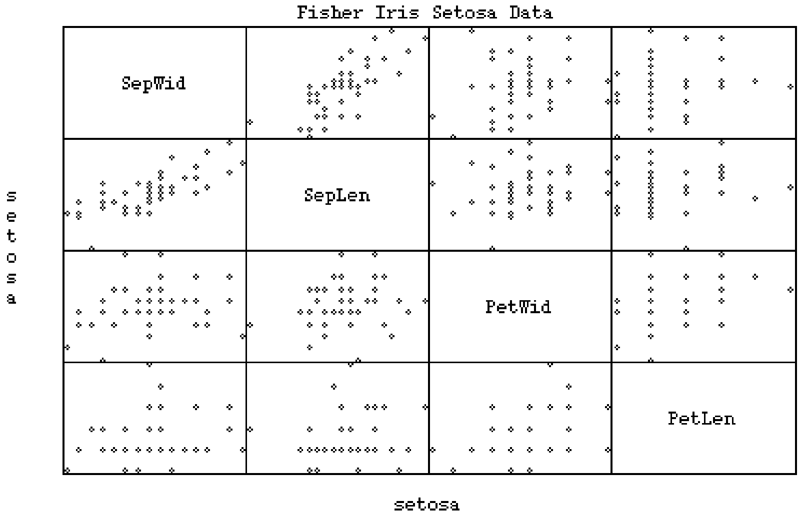
`title:"Fisher Iris Setosa Data"` in the command is a **keyword phrase**. Keyword phrases have the form **`name:value`** and are often used to control or change what a command does behavior of an operation.

Particularly common are **`name:T`** (true or yes) and **`name:F`** (false or no). For example, **`quiet:T`** is sometimes used to suppress the usual printed output.

If you think default symbols in the plot might are too small you can change them.

Symbol code `"\21"` specifies small diamonds using keyword **`symbols`**.

```
Cmd> plotmatrix(setosa,title:"Fisher Iris Setosa Data",\
        symbols:"\21")
```



These are easier to see.

Limitation of scatter plot matrix

It provides information only on relationships of pairs of variables but no information about 3 and 4 variable relationships.

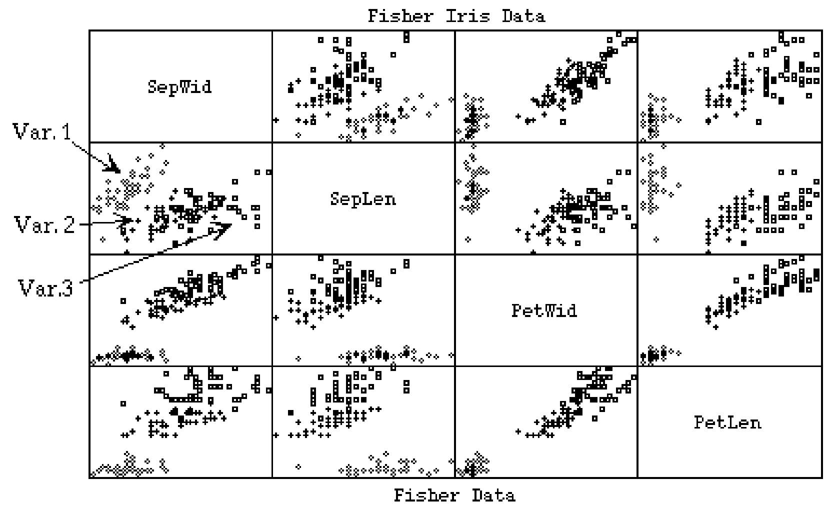
You can make a plot of *all* the iris data, distinguishing the three varieties of iris (1, 2 or 3) by differing symbols.

```
Cmd> y <- irisdata[,-1] # everything but column 1, 150 by 4
Cmd> symbols <- vector("\21","\22","\23")[variety] #150 by 1
```

`"\21","\22","\23"` are codes for small diamond (◇), cross (+) and square (□).

`[variety]` is a *subscript*, selecting elements from `vector("\21","\22","\23")` All variety 1 cases get `"\21"`, all variety 2 cases get `"\22"` and all variety 3 cases get `"\23"`.

```
Cmd> plotmatrix(y,title:"Fisher Iris Data",xlab:"Fisher Data",\
ylab:" ",symbols:symbols)
```



Chernoff Faces

This is a very clever idea that can sometimes be useful. It exploits the fact that people have a lot of experience in recognizing and differentiating *faces*.

A caricature of a face is drawn with the size and shape of features derived from a multivariate observation **x** = $(x_1, x_2, \ldots, x_p)$.

You can use macro **faces()** to draw Chernoff faces. Unfortunately, there is a bug in one of the standard macro files **Graphics.mac** and you need a new version to use **faces()**.

```
Cmd> addmacrofile("") # Get macrofile containg faces()

Cmd> addmacrofile("") # Get new graphics.mac

Cmd> MACROFILES
 (1) "TP1:Macros:graphics:Graphics.mac"   New Graphics.mac
 (2) "TP1:Macros:Mulvar:mvgraphics.mac"   File with faces()
 (3) "Graphics.mac"
 (4) "Regress.mac"
 (5) "Design.mac"
 (6) "Tser.mac"
 (7) "Arima.mac"
 (8) "Mulvar.mac"
 (9) "Math.mac"
(10) "MacAnova.mac"
```


You always need to use **addmacrofile("")** before you can use a non-standard macro.

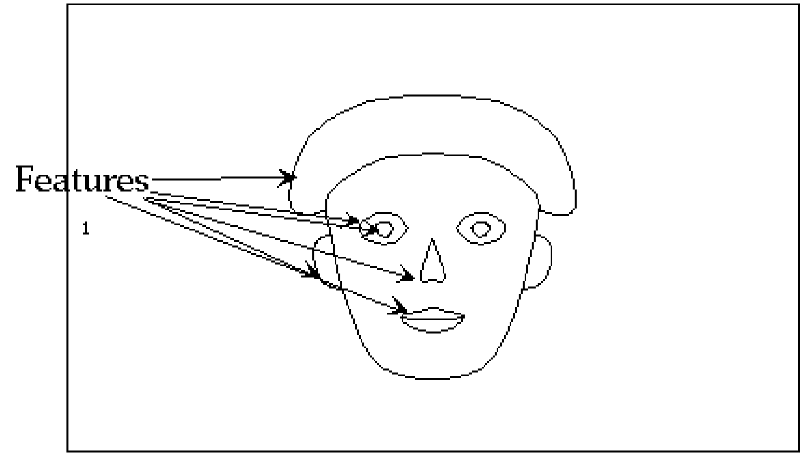
Here is a boring vector of p = 10 values:

```
Cmd> x0 <- rep(1,10)'; x0 # repeat 1 ten times
(1,1)          1          1          1          1          1
(1,6)          1          1          1          1          1
```

"**'**" indicates *transpose*, turning the column vector **rep(1,10)** into a row vector. **faces()** treats rows as cases.

```
Cmd> faces(x0) # draw generic face for featureless data
WARNING: searching for unrecognized macro faces near faces(
```

Faces plot of x0



Cases in order left to right by rows

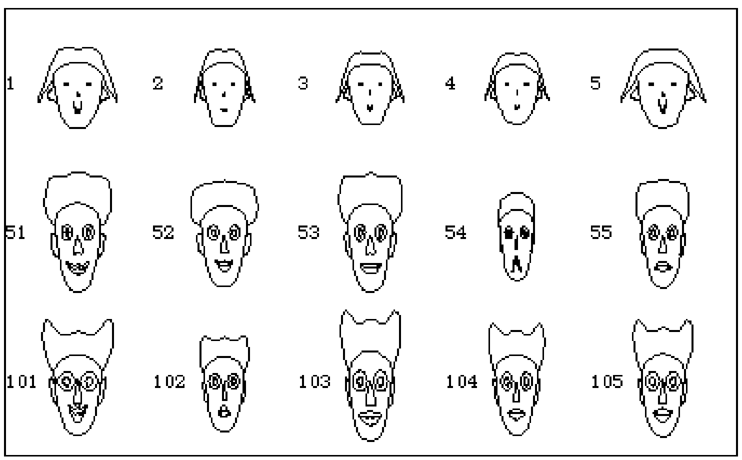
This is a generic face when all x's are the same.

Cases 1 - 50 of iris data are variety 1, cases 51-100 are variety 2, and cases 101-150 are variety 3.

Here is a faces plot of the first 5 cases in each group (cases 1-5, 510-55, 101-105) as selected by vector `J`.

```
Cmd> J <- vector(run(5), 50+run(5), 100+run(5)); J
 (1)          1          2          3          4          5
 (6)         51         52         53         54         55
(11)        101        102        103        104        105

Cmd> faces(y[J,],nrows:3,ncols:5,\
        title:"Faces for 15 cases of Fisher Iris Data",\
        labels:paste(J,multiline:T))
```

Faces for 15 cases of Fisher Iris Data



Cases in order left to right by rows

The backslashes (\) mean the command is continued on the following line.

`nrows:3,ncols:5` says there should be 3 rows of faces, 5 faces per row.

The face labels to their left come from `labels:paste(J,multiline:T)`. `paste(J,multiline:T)` is the same as `vector("1","2","3","4",...,"103", "104","105")`.

`paste()` is the way to turn numbers into character information. `multiline:T` directs that each number goes in a separate element.

There are several options. The help is in the same file as the macro.

```
Cmd> addhelpfile("") # locate file mvgraphics.mac

Cmd> usage(faces)
faces(Y [, whichvars:varnos] [,byrow:T] [,mods:Mods] [,scale:F]\
  [,min:Max] [,max:Max] [,keep:T] [,draw:T] [,nrows:nr] \
  [,ncols:nc] [,panel:F] [,labels:facelabs] [,labelplace:lp] \
  [,gridlines:T]  [,npoints:npts] [graph keywords]), Y REAL
  nonMISSING matrix, Mods a structure with positive scalar
  components, nr > 0, nc > 0 integers, CHARACTER vector
  facelabs, CHARACTER scalar lp, integer npts > 0
```

You can get more information by `help(faces)`, although the information is fairly rudimentary.

Andrews plots

Each case $\mathbf{x} = (x_1, x_2, \ldots, x_p)$ is used to compute a curve f(t) which gets plotted.

Specifically, for Andrews plots the curve is

$$f(t) = \tilde{x}_1/\sqrt{2} + \tilde{x}_2\cos(2\pi t) + \tilde{x}_3\sin(2\pi t) + + \tilde{x}_4\cos(4\pi t) + \tilde{x}_5\sin(4\pi t) + \ldots,$$

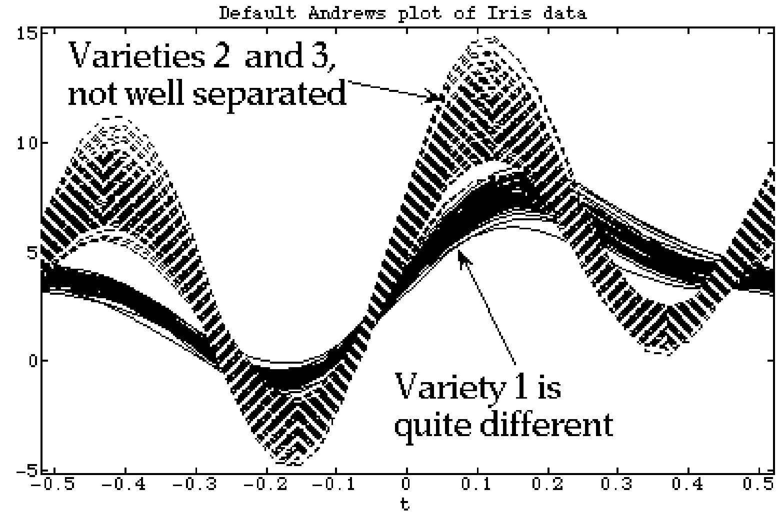
$$-.5 \le t \le +.5$$

where $\tilde{x}_j$ is a rescaled version of x_j, usually mapping the range of x_j into the interval from -1 to 1.

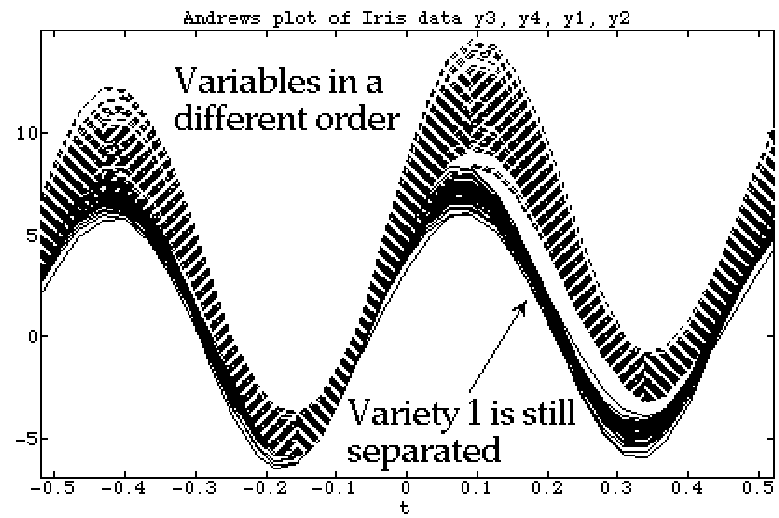
For each t, f(t) is a different linear combination of the variables.

You can make Andrews plots using macro **andrewsplot()**. Argument 1 is a data matrix, with each case in a row. You can use an optional second argument to specify groups so that different line types are used for different groups.


```
Cmd> andrewsplot(y,variety,\
     title:"Default Andrews plot of Iris data")
```




```
Cmd> andrewsplot(y[,vector(3,4,1,2)],variety,\
     title:"Andrews plot of Iris data y3, y4, y1, y2")
```



More on MacAnova

Entering data

- Single number: Just type it, using "scientific" notation if desired

```
Cmd> 3.54e3  # 3.54 x 10^3
(1)        3540

Cmd> -3.676e-3  # -3.676 x 10^-3
(1)   -0.003676
```

- Several numbers, creating a *vector*

```
Cmd> y <- vector(3, 2, 5, 4, 9, -1)

Cmd> y # typing name of variable, prints it
(1)          3          2          5          4          9
(6)         -1
```

`(1)` and `(6)` identify the first numbers in the rows as being elements `y[1]` and `y[6]`.

```
Cmd> z <- enter(1 2 3 4 5) # commas not needed

Cmd> z
(1)          1          2          3          4          5

Cmd> 1/z # reciprocals; expression you type is printed
(1)          1        0.5    0.33333       0.25        0.2

Cmd> sqrt(z) # square roots
(1)          1     1.4142     1.7321          2     2.2361

Cmd> z^3 # cubes; "^" designates power
(1)          1          8         27         64        125

Cmd> 2*z + enter(10 0 0 0 0) # or 2*z + vector(10,0,0,0,0)
(1)         12          4          6          8         10
```


Matrix and vector notation

A *matrix* is like a *rectangular table*, having rows and columns.

Here is a $\boxed{4 \times 3}$ matrix, that is a matrix with 4 <u>rows</u> and 3 <u>columns</u>

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & -3 \\ 0 & 11 & 4 \\ 7 & 17 & 12 \\ -3 & 4 & 5 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix}$$

A lot of statistical data and analysis, especially multivariate statistics, are best expressed using matrices.

One way you create a matrix in MacAnova is to use function **`matrix()`** with usage

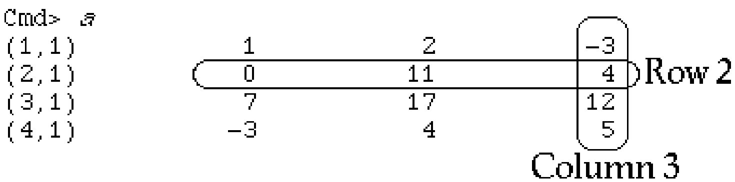
```
Cmd> a <- matrix(data,nrows)
```

`data`, usually a vector, provides the contents of the matrix, put into **`a`** column by column. **`nrows`** = the number of rows = length of each column.

Create **A** in MacAnova.

Matrix is "built" column by column, *not* row by row.

```
Cmd> a <- matrix(vector(1,0,7,-3, 2,11,17,4, -3,4,12,5), 4)

Cmd> # the final 4 is number of rows
```



(1,1), (2,1), (3,1) and (4,1) identify the first numbers in each line as being elements in rows 1 through 4 and in column 1 of x.

For example 7 is in row 3 and column 1.

You extract elements using **subscripts**.

```
Cmd> a[2,3] # element in row 2 of column 3
(1,1)           4

Cmd> a[3,2] # element in row 3 of column 2
(1,1)          17
```

Each are these are essentially 1 by 1 matrices as the (1,1) suggests.