ORIGIN ≡ 1

Multivariate Data: Geometric Interpretations

In addition to matrix algebra formulae, operations on multivariate data often have highly useful geometric interpetations that provide part of the underlying rationale for different methods. A Multivariate data matrix consisting of cases (typically rows) measured over several variables (columns) may be interpreted as consisting of a set of vectors usually considered column vectors in matrix algebra. Each vector may alternately or simultaneously be considered to have two interpretations: directed line segments (i.e., geometric vectors) and/or points in multidimensional space. If the number of variables is two, the entire multidimensional space described occurs on a plane with either vectors or points residing on that plane. If there are three variables, vecors or point lie within 3-D space. Both situations may be graphed normally in 2-D or 3-D respectively. Typically, however, the number of variables in the dataset are greater than three, and although all matrix algebra formulae continue to work properly, the vectors or points occur in "hyperspace" and cannot be graphed directly. Nevertheless, discussion will often center around geometric properties, such as angle, projection, or distance in this hyperspace as if it were possible to do so. The purpose of this worksheet is to present some commonly discussed properties with their definition and examples. For more detail, see RA Johnson & DW Wichern *Applied Multivariate Statistical Analysis 4th Edition* 1998, or AC. Rencher *Methods of Multivariate Analysis* 1995.

Read Data:

M := READPRN("c:/DATA/Multivariate/Isetosastand.txt")

Data represent the standardized values of all 50 objects in Anderson's Iris dataset for species *Iris setosa* only.

M =

	1	2	3	4
1	0.267	0.19	-0.357	-0.436
2	-0.301	-1.129	-0.357	-0.436
3	-0.868	-0.601	-0.933	-0.436
4	-1.152	-0.865	0.219	-0.436
5	-0.017	0.454	-0.357	-0.436
6	1.118	1.245	1.37	1.461
7	-1.152	-0.074	-0.357	0.512
8	-0.017	-0.074	0.219	-0.436
9	-1.719	-1.393	-0.357	-0.436
10	-0.301	-0.865	0.219	-1.385
11	1.118	0.718	0.219	-0.436
12	-0.584	-0.074	0.795	-0.436
13	-0.584	-1.129	-0.357	-1.385
14	-2.003	-1.129	-2.084	-1.385
15	2.253	1.509	-1.509	-0.436
16	1.969	2.564	0.219	1.461
17	1.118	1.245	-0.933	1.461
18	0.267	0.19	-0.357	0.512

X := M^T

Prototype in R:

```
> #MULTIVARIATE DATA: GEOMETRIC INTERPRETATIONS
> #READ DATA:
> M=read.table("c:/DATA/Multivariate/Isetosastand.txt",header=F)

> M
            V1          V2          V3          V4
1   0.26667447  0.18994136 -0.3570112 -0.4364923
2  -0.30071802 -1.12909583 -0.3570112 -0.4364923
3  -0.86811050 -0.60148096 -0.9328358 -0.4364923
4  -1.15180675 -0.86528840  0.2188133 -0.4364923
5  -0.01702177  0.45374879 -0.3570112 -0.4364923
6   1.11776320  1.24517111  1.3704625  1.4613004
7  -1.15180675 -0.07386608 -0.3570112  0.5124040
8  -0.01702177 -0.07386608  0.2188133 -0.4364923
9  -1.71919923 -1.39290327 -0.3570112 -0.4364923
10 -0.30071802 -0.86528840  0.2188133 -1.3853887
11  1.11776320  0.71755623  0.2188133 -0.4364923
12 -0.58441426 -0.07386608  0.7946379 -0.4364923
13 -0.58441426 -1.12909583 -0.3570112 -1.3853887
14 -2.00289548 -1.12909583 -2.0844850 -1.3853887
15  2.25254817  1.50897854 -1.5086604 -0.4364923
16  1.96885193  2.56420830  0.2188133  1.4613004
17  1.11776320  1.24517111 -0.9328358  1.4613004
18  0.26667447  0.18994136 -0.3570112  0.5124040
19  1.96885193  0.98136367  1.3704625  0.5124040
20  0.26667447  0.98136367  0.2188133  0.5124040
21  1.11776320 -0.07386608  1.3704625 -0.4364923
22  0.26667447  0.71755623  0.2188133  1.4613004
...
50 -0.01702177 -0.33767352 -0.3570112 -0.4364923
```

Data as vectors:

In 2-D graphing only two variables:

In 4-D describing all variables:

$$X^{\langle 6\rangle} = \begin{pmatrix} 1.118 \\ 1.245 \\ 1.37 \\ 1.461 \end{pmatrix} \quad X^{\langle 20\rangle} = \begin{pmatrix} 0.267 \\ 0.981 \\ 0.219 \\ 0.512 \end{pmatrix}$$

Vector Length:

$$L_{X6} := \left| \sqrt{X^{\langle 6\rangle T} \cdot X^{\langle 6\rangle}} \right| \qquad L_{X6} = 2.61$$

$$L_{X20} := \left| \sqrt{X^{\langle 20\rangle T} \cdot X^{\langle 20\rangle}} \right| \qquad L_{X20} = 1.16$$

Angle between Vectors:

$$\cos\theta := \left| \frac{(X)^{\langle 6\rangle T} \cdot X^{\langle 20\rangle}}{\left|X^{\langle 6\rangle}\right| \cdot \left|X^{\langle 20\rangle}\right|} \right| \qquad \cos\theta = 0.8486517$$

$$\theta_r := \operatorname{acos}(\cos\theta) \qquad \theta_r = 0.557365 \text{ rad}$$

$$\theta := \left(\frac{180}{\pi}\right) \cdot \theta_r \qquad \theta = 31.935$$

$$\pi = 3.141593$$

Projection of One Vector onto Another:

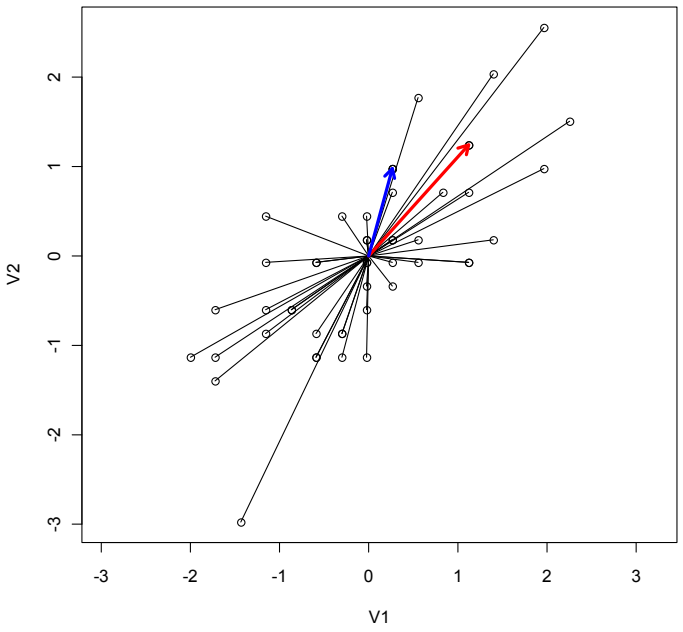
Projection of X^{20} onto X^{6}:

$$proj := \frac{\left|X^{\langle 20\rangle T} \cdot (X)^{\langle 6\rangle}\right|}{\left|X^{\langle 6\rangle T} \cdot X^{\langle 6\rangle}\right|} \cdot X^{\langle 6\rangle} \qquad proj = \begin{pmatrix} 0.4214033 \\ 0.4694368 \\ 0.5166724 \\ 0.5509188 \end{pmatrix}$$

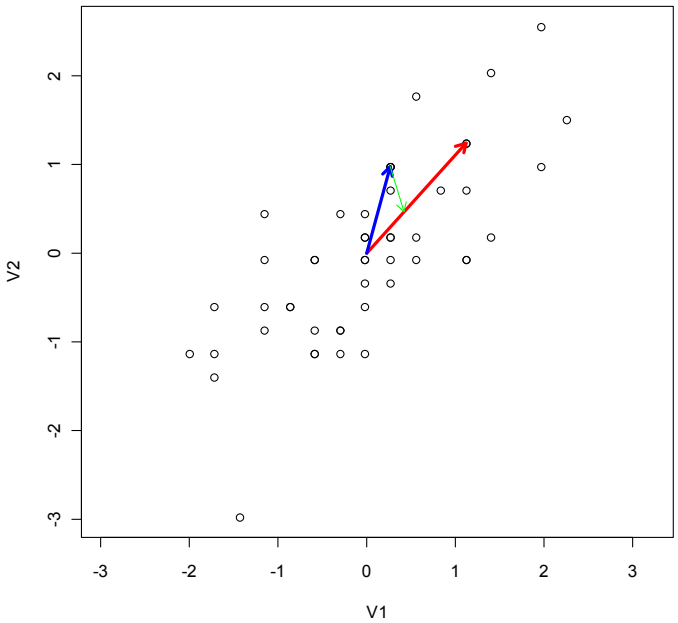
A projection vector (green) by definition occurs at 90 degrees ($\cos\theta = 1$) to the vector projected onto (red) although it may not appear to be so since the graph is in 2-D not 4-D!

$$\cos\theta := \left| \frac{proj^T \cdot X^{\langle 6\rangle}}{|proj| \cdot \left|X^{\langle 6\rangle}\right|} \right| \qquad \cos\theta = 1$$

```
#PLOT OF VECTORS
plot(V1,V2,asp=1)
segments(0,0,V1,V2)

#TWO SPECIFIC VECTORS:
arrows(0,0,V1[6],V2[6],col='red',lwd=3,code=2,length=0.1)
arrows(0,0,V1[20],V2[20],col='blue',lwd=3,code=2,length=0.1)
```




```
> score
[1] 0.4214033 0.4694368 0.5166724 0.5509188
```



Data as points in space:

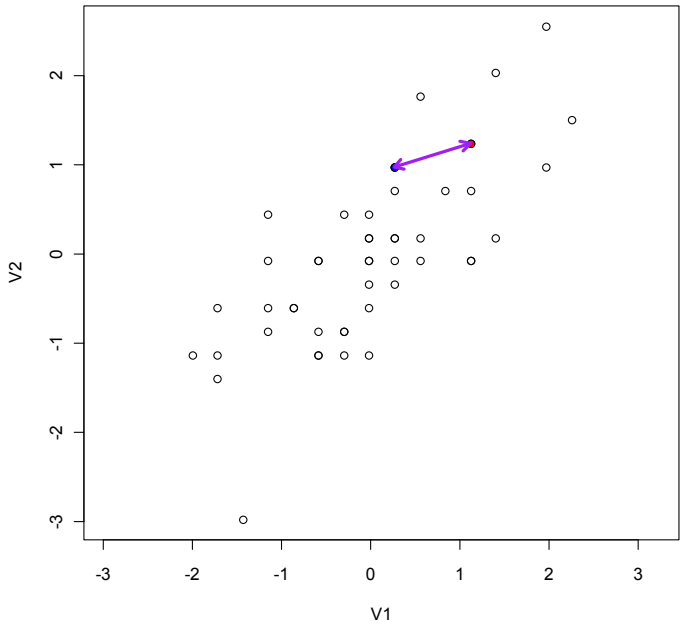
In 2-D graphing only two variables:

Squared Euclidean Distance between points:

$$\left(X^{\langle 20\rangle} - X^{\langle 6\rangle}\right)^T \cdot \left(X^{\langle 20\rangle} - X^{\langle 6\rangle}\right) = (3.021)$$

^ squared Euclidean distance

```
#PLOT OF POINTS:
plot(V1,V2,asp=1)
points(V1[6],V2[6],col='red',pch=20)
points(V1[20],V2[20],col='blue',pch=20)
arrows(V1[20],V2[20],V1[6],V2[6],col='purple',
  lwd=3,code=3,length=0.1)
```



Squared Mahalanobis distance:

$n := \text{rows}(M) = (50)$

$p := \text{cols}(M) = (4)$

$i := 1 .. n \qquad j := 1 .. p$ < index variables

$l_{vec_i} := 1$

$I := \text{identity}(n)$

$$S := \frac{1}{n-1} \cdot M^T \cdot \left(I - \frac{1}{n} \cdot l_{vec} \cdot l_{vec}^T\right) \cdot M$$

$$S = \begin{pmatrix} 1 & 0.743 & 0.267 & 0.278 \\ 0.743 & 1 & 0.178 & 0.233 \\ 0.267 & 0.178 & 1 & 0.332 \\ 0.278 & 0.233 & 0.332 & 1 \end{pmatrix}$$

< covariance matrix

Mahalanobis distance, also called "statistical distance" takes into account the covariance between variables as indicated in matrix S. The role of matrix S will be considered further below.

$$\left(X^{\langle 20\rangle} - X^{\langle 6\rangle}\right)^T \cdot S^{-1} \cdot \left(X^{\langle 20\rangle} - X^{\langle 6\rangle}\right) = (2.186)$$

^ squared Mahalanobis distance

Linear Transformations:

Linear transformation involve multiplying vectors, columns of matrix X (here visualized as representing points in space) by some matrix M. Matrix X was made by using function make.grid() in the accompaning R script.

$X := \text{READPRN}(\text{"c:\DATA\Multivariate\grid.txt"})$

$$M := \begin{pmatrix} 1 & 0.3 \\ 0.3 & 1 \end{pmatrix}$$ < matrix of the linear transformation

$Z := M \cdot X^T$

^ Linear Tranformation of vectors in X into vectors in Z

(function make.grid() defined in R script:)

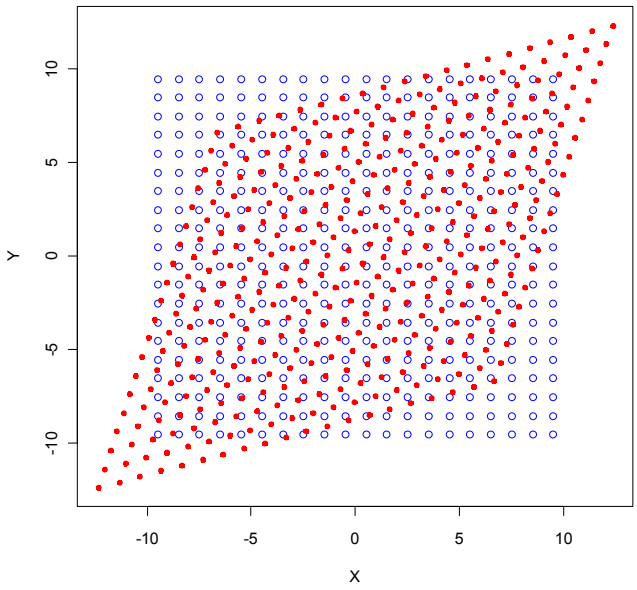
```
X=make.grid(20,20)

#LINEAR TRANSFORMATION:
M=matrix(c(1.0,0.3,0.3,1.0),nrow=2,ncol=2,byrow=T)
Z=M%*%t(X)
Z=t(Z)

#PLOT UNTRANSFORMED AND TRANSFORMED POINTS:
plot(Z,type='n',xlab='X',ylab='Y')
points(X,col='blue')
points(Z,col='red',pch=20)
```

$X =$

	1	2
1	-9.5	-8.5
2	-9.5	-7.5
3	-9.5	-6.5
4	-9.5	-5.5
5	-9.5	-4.5
6	-9.5	-3.5
7	-9.5	-2.5
8	-9.5	-1.5
9	-9.5	-0.5
10	-9.5	0.5
11	-9.5	1.5
12	-9.5	2.5
13	-9.5	3.5

$Z^T =$

	1	2
1	-12.05	-11.35
2	-11.75	-10.35
3	-11.45	-9.35
4	-11.15	-8.35
5	-10.85	-7.35
6	-10.55	-6.35
7	-10.25	-5.35
8	-9.95	-4.35
9	-9.65	-3.35
10	-9.35	-2.35
11	-9.05	-1.35
12	-8.75	-0.35
13	-8.45	0.65



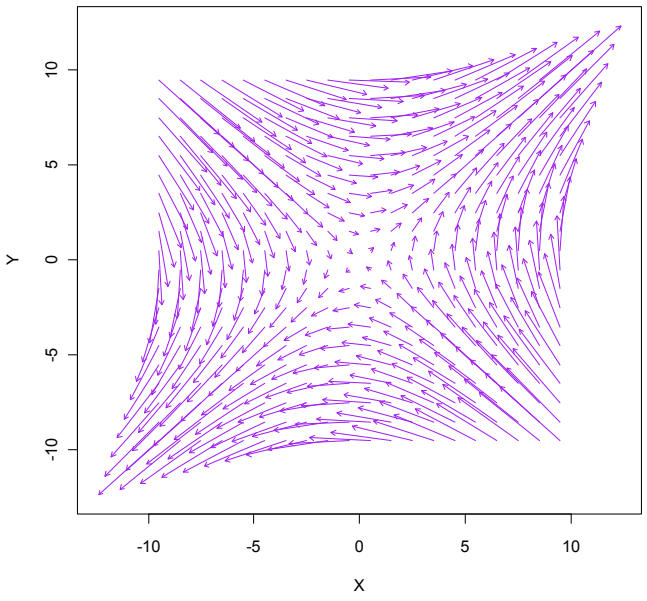
^ original vectors in X (rows) **^ transformed vectors in Z (rows)**

```
#PLOT TRANSFORMATION VECTORS:
plot(Z,type='n',xlab='X',ylab='Y')
arrows(X[,1],X[,2],Z[,1],Z[,2],col='purple',lwd=1,code=2,length=0.05)
```

Note how the points in X transformed to Z in the graph above have been moved by "warping" the underlying two-dimensional space. In the graph to the right the same warp is shown as vectors with the foot of each vector representing a point in X, and the head a point in Z.

Every unique transformation matrix M specifies a different warping of the space.

In general, symmetric matrices yield vector fields with Principal Directions (where orientations of vectors doesn't change) that are mutually perpendicular (i.e., dot product is 0). These transformations are commonly encountered in multivariate statistics.



From this, it is easy to see that Linear Transformations in two dimension have two Principal Directions in the fabric of the warp that involve lengthening or shortening vector lengths only - but not changing their directions. In general there are n - many such Principal Directions in an n-dimensional space.

The Principal Directions are called "Eigenvector" directions (after German eigen = innate, peculiar, own)
The amount of streatch or shrink in the Eigenvector directions are recorded by numbers called "Eigenvalues".

Distances to the Center for each point:

It is also useful to visualize linear transformations as above in terms of Euclidean and Mahalanobis distances. Here we look at 200 points on a circle with each point a set distance (4 units) from then center of the reference system (0,0). Points of the circle, shown blue in the graph below, were constructed using function make.grid() in the accompaning R script.

Squared Euclidean Distances of Circle:

$C := \text{READPRN}(\text{"c:/DATA/Multivariate/circle.txt"})$

$n := \text{rows}(C) \qquad n = 200$

$i := 1 .. n$

$D_{C_i} := \left(C \cdot C^T\right)_{i,i}$ **< squared Euclidean distance calculated for each of 200 points**

$C =$

	1	2
1	1.999	0.063
2	1.996	0.126
3	1.991	0.188
4	1.984	0.251
5	1.975	0.313
6	1.965	0.375
7	1.952	0.436
8	1.937	0.497
9	1.921	0.558

^ original points of the circle (blue)

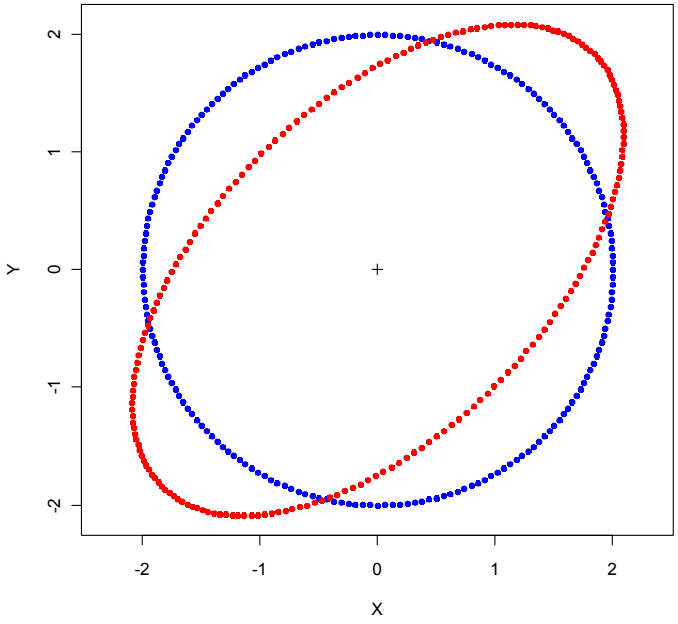
$D_C =$

	1
1	4
2	4
3	4
4	4
5	4
6	4
7	4
8	4
9	4

^ squared Euclidean distance of each point to the center

make.grid() function in R script constructs 200 points equidistant from the center at (0,0) shown in blue.

```
C=make.circle(200,2)
C

#LINEAR TRANSFORMATION:
E=M%*%t(C)
E=t(E)

#PLOT UNTRANSFORMED AND TRANSFORMED CIRCLE:
plot(E,asp=1,type='n',xlab='X',ylab='Y')
points(0,0,col='black',pch=3)
points(C,col='blue',pch=20)
points(E,col='red',pch=20)
```



Linear Transformation of the Circle:

$M = \begin{pmatrix} 1 & 0.3 \\ 0.3 & 1 \end{pmatrix}$ **< matrix of the linear transformation (same as above)**

$E := M \cdot C^T$ **< matrix E of tranformed points (red)**

$E := E^T$

$D_{E_i} := \left(E \cdot E^T\right)_{i,i}$ **< Euclidean distances of transformed points to the center**

The linear transformation converts the circle of points (blue) into an ellipse of transformed points (red). This is an equally valid way to view the geometry of the transformation.

$E =$

	1	2
1	2.018	0.663
2	2.034	0.724
3	2.048	0.786
4	2.059	0.846
5	2.069	0.905
6	2.077	0.964
7	2.083	1.022
8	2.086	1.079
9	2.088	1.134

^ matrix E of ellipse points (red)

Varying Euclidean distance of ellipse points to the center:

$D_E =$

	1
1	4.511
2	4.661
3	4.81
4	4.957
5	5.102
6	5.243
7	5.382
8	5.516
9	5.646

Squared Mahalanobis distance:

$l_{vec_i} := 1$

$I := \text{identity}(n)$

$$S := \frac{1}{n-1}\cdot E^T\cdot\left(I - \frac{1}{n}\cdot l_{vec}\cdot l_{vec}^T\right)\cdot E \qquad S = \begin{pmatrix} 2.191 & 1.206 \\ 1.206 & 2.191 \end{pmatrix}$$

^ covariance matrix calculation using matrix algebra

$D_{E_i} := \left(E\cdot S^{-1}\cdot E^T\right)_{i,i}$

^squared Mahalanobis distance

$D_E =$

	1
1	1.99
2	1.99
3	1.99
4	1.99
5	1.99
6	1.99
7	1.99
8	1.99
9	1.99

Mahalanobis distances use the inverse of the calculated covariance matrix (S^{-1}) to "correct" for the ellipse caused by the linear transformation specified by matrix M. The result are squared distances from the center for all 200 points that are all the same.

Although scaled differently, the important thing to note is that squared Mahalanobis distances are identical for all of the points on the ellipse whereas squared Euclidean Distances are not.

Back-transforming the ellipse:

$CC := M^{-1}\cdot E^T$

^ transforming points in the ellipse through the inverse of linear tranformation matrix M

$C =$

	1	2
1	1.999	0.063
2	1.996	0.126
3	1.991	0.188
4	1.984	0.251
5	1.975	0.313
6	1.965	0.375
7	1.952	0.436
8	1.937	0.497
9	1.921	0.558

^ original circle (blue)

$E =$

	1	2
1	2.018	0.663
2	2.034	0.724
3	2.048	0.786
4	2.059	0.846
5	2.069	0.905
6	2.077	0.964
7	2.083	1.022
8	2.086	1.079
9	2.088	1.134

^ellipse (red)

$CC^T =$

	1	2
1	1.999	0.063
2	1.996	0.126
3	1.991	0.188
4	1.984	0.251
5	1.975	0.313
6	1.965	0.375
7	1.952	0.436
8	1.937	0.497
9	1.921	0.558

^ back-transformed points (same as original circle)

$D_{I_i} := \left(CC^T\cdot CC\right)_{i,i}$

^ Euclidean distances of back transformed points

Using the inverse of transformation matrix M (M^{-1}) allows one to recover the original scale of points...

$D_I =$

	1
1	4
2	4
3	4
4	4
5	4
6	4
7	4
8	4
9	4

However, the problem is that in real situations involving elliptically scattered data, we lack prior knowledge of the original Linear Transformation represented by M and must use instead covariance matrix S^{-1}

Until some statistician tells me otherwise, it would seem that recovering the "original scale" of the data is not possible from the variance/covariance matrix alone.

Such a concept of "original scale" in prior real data probabally doesn't mean much anyway, and statistically this limitation is unimportant.

Plotting Mahalanobis corrected data points:

Multiplying each data vector by the inverse square root matrix of the covariance matrix X converts data into the "Mahalanobis corrected" space.

See Multivariate Worksheet MTB 060 for information on matrix square root.

The important thing to note in the graph below is that the original data (blue) originally showing covariance between variables V1 & V2 (as indicated by elliptical scatter of the points), now have been corrected (red) into a more cicular scatter.

(Function matrix.square.root() for square root of a matrix given in R script)

```
#MAHALANOBIS CORRECTED PLOT:
M=read.table("c:/DATA/Multivariate/Isetosastand.txt",header=F)
attach(M)
S=cov(M)
S

S.sqr=matrix.square.root(S)
S.sqr

M.mah=solve(S.sqr)%*%t(M)
M.mah=t(M.mah)

#PLOT OF POINTS:
plot(M.mah[,1],M.mah[,2],type='n',asp=1,xlab="V1",ylab="V2")
points(V1,V2,col='blue',pch=20)
points(M.mah[,1],M.mah[,2],col='red',pch=20)
```