Statistics 5401
18. Principal Components

Gary W. Oehlert
School of Statistics
313B Ford Hall
612-625-1557
gary@stat.umn.edu

Singular Value Decomposition

SVD is a matrix algebra decomposition that is very helpful for principal components analysis, so let's talk about it now, and then use it when the time is ripe.

Let $\mathbf{X}$ be an $(n \times p)$ matrix with $n \geq p$. (This can be made to work when $p > n$ — just work with $\mathbf{X}'$.)

We decompose as

$$\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}'$$

where

$\mathbf{U}$ is $(n \times p)$ and $\mathbf{U}'\mathbf{U} = \mathbf{I}_p$

$\mathbf{V}$ is $(p \times p)$ and $\mathbf{V}'\mathbf{V} = \mathbf{I}_p$

$\mathbf{D}$ is $(p \times p)$ diagonal with elements $d_1 \geq d_2 \geq \ldots \geq d_p \geq 0$.

The d_is are the singular values, the columns of $\mathbf{U}$ are the left singular vectors, and the columns of $\mathbf{V}$ are the right singular vectors. ($\mathbf{U}$ and $\mathbf{V}$ both have orthonomal columns.)

If the singular values are all different, then $\mathbf{U}$ and $\mathbf{V}$ are essentially unique; you can multiply them both by –1 and not change the product. If the are repeated singular values, then the singular vectors are not unique.

The rank of a matrix is the number of nonzero singular values.

```
Cmd> setseeds(906984,33287)

Cmd> X <- matrix(floor(100*rnorm(21)),7);X
(1,1)           62          151           -6
(2,1)           89          126          -44
(3,1)          -91          115          172
(4,1)         -220          -34            2
(5,1)         -144           28           -4
(6,1)          -60           -7          -19
(7,1)          -98          117           63

Cmd> svd(X)
(1)       333.29       275.21       138.02

Cmd> svd(X,left:T)
component: values
(1)       333.29       275.21       138.02
component: leftvectors
(1,1)     -0.13175       0.5456     -0.34034
(2,1)     -0.25405      0.43841     -0.45858
(3,1)      0.46659      0.51683      0.59196
```

```
(4,1)      0.60835     -0.29063     -0.32815
(5,1)      0.40887    -0.029554     -0.39012
(6,1)      0.14681    -0.095438     -0.22048
(7,1)      0.37626      0.38541     -0.12234

Cmd> svd(X,right:T)
component: values
(1)       333.29       275.21       138.02
component: rightvectors
(1,1)     -0.93502      0.22516      0.27392
(2,1)      0.10655      0.91523     -0.38859
(3,1)       0.3382      0.33416      0.87975

Cmd> compnames(svd(X,all:T))
(1) "values"
(2) "leftvectors"
(3) "rightvectors"

Cmd> D <- dmat(svd(X))

Cmd> U <- svd(X,left:T)$leftvectors

Cmd> V <- svd(X,right:T)$rightvectors

Cmd> U%*%D%*%V'
(1,1)           62          151           -6
(2,1)           89          126          -44
(3,1)          -91          115          172
(4,1)         -220          -34            2
(5,1)         -144           28           -4
(6,1)          -60           -7          -19
(7,1)          -98          117           63
```

$$\mathbf{X}'\mathbf{X} = \mathbf{V}\mathbf{D}\mathbf{U}'\mathbf{U}\mathbf{D}\mathbf{V}' = \mathbf{V}\mathbf{D}^2\mathbf{V}'$$

The eigenvalues of $\mathbf{X}'\mathbf{X}$ are the d_i^2, and the eigenvectors are $\mathbf{V}$.

$$\mathbf{X}\mathbf{X}' = \mathbf{U}\mathbf{D}\mathbf{V}'\mathbf{V}\mathbf{D}\mathbf{U}' = \mathbf{U}\mathbf{D}^2\mathbf{U}'$$

$\mathbf{X}\mathbf{X}'$ has at most p nonzero eigenvalues; these are d_i^2 with corresponding eigenvectors in $\mathbf{U}$. The other eigenvalues are all zero, and their eigenvectors are arbitrary except that they are all orthogonal to the columns of $\mathbf{U}$.

```
Cmd> eigen(X'%*%X)
component: values
(1)   1.1108e+05        75739        19049
```

```
component: vectors
(1,1)     -0.93502      0.22516     -0.27392
(2,1)      0.10655      0.91523      0.38859
(3,1)       0.3382      0.33416     -0.87975

Cmd> svd(X)^2
(1)   1.1108e+05        75739        19049

Cmd> eigenvals(X%*%X')
(1)   1.1108e+05  75739  19049      0      0
(6)            0      0

Cmd> eigen(X%*%X')$vectors[,run(3)]
(1,1)     -0.13175       0.5456     -0.34034
(2,1)     -0.25405      0.43841     -0.45858
(3,1)      0.46659      0.51683      0.59196
(4,1)      0.60835     -0.29063     -0.32815
(5,1)      0.40887    -0.029554     -0.39012
(6,1)      0.14681    -0.095438     -0.22048
(7,1)      0.37626      0.38541     -0.12234
```

The norm of a matrix $\mathbf{A}$ is

$$||\mathbf{A}||^2 = \text{trace}(\mathbf{A}'\mathbf{A}) = \sum_{ij} a_{ij}^2$$

A rank one matrix is $\mathbf{Z} = ab'$, where a is $(n \times 1)$ and b is $(p \times 1)$. Suppose you want to make the best rank one approximation to $\mathbf{X}$; that is, we want $\mathbf{Z}$ to minimize $||\mathbf{X} - \mathbf{Z}||$.
Then

$$\mathbf{Z} = \check{\mathbf{U}}_1 d_1 \check{\mathbf{V}}_1'.$$

One way to look at this rank-one approximation is you want to find a predictor vector such that when you regress all the columns of $\mathbf{X}$ on that predictor (no intercept) and add the error sum of squares, you minimze the total error sum of squares.
Note also that

$$\mathbf{X}\check{\mathbf{V}}_1\check{\mathbf{V}}_1' = \check{\mathbf{U}}_1 d_1 \check{\mathbf{V}}_1' = \mathbf{Z}$$

In general, if you want the best rank k approximation to $\mathbf{X}$, then use

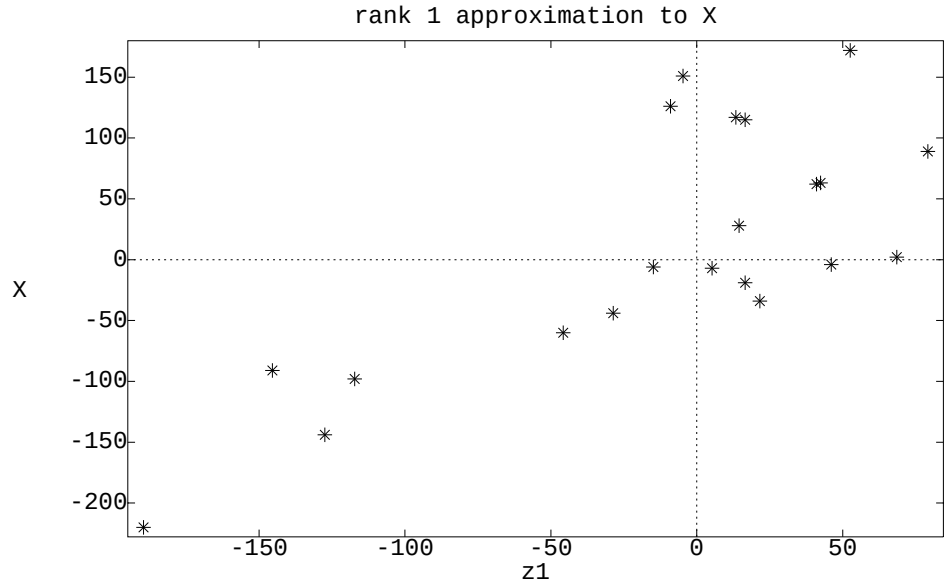
$$\mathbf{Z} = \check{\mathbf{U}}_1 d_1 \check{\mathbf{V}}_1' + \check{\mathbf{U}}_2 d_2 \check{\mathbf{V}}_2' + \ldots + \check{\mathbf{U}}_k d_k \check{\mathbf{V}}_k'$$

For the rank k approximation,

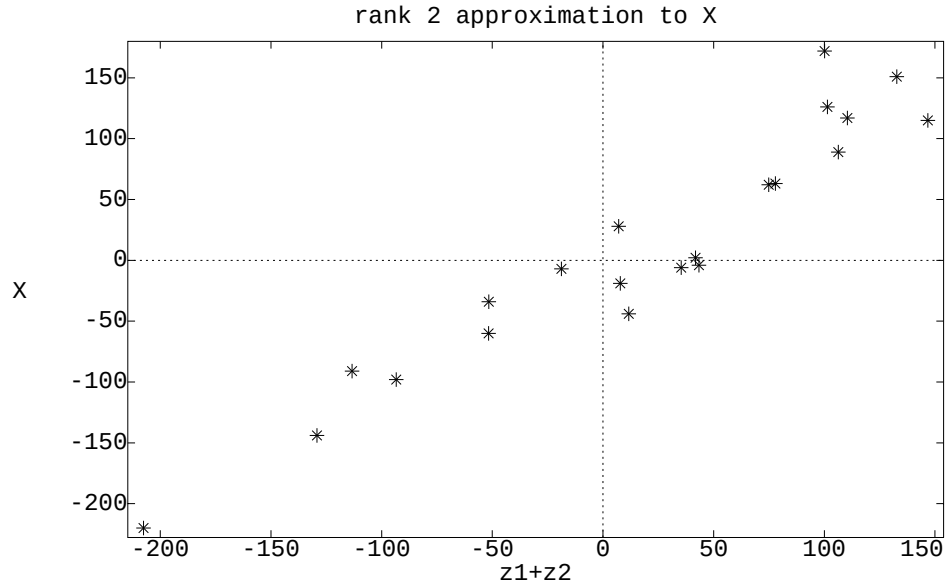
$$\mathbf{Z} = \mathbf{X} \sum_{i=1}^{k} \check{\mathbf{V}}_i \check{\mathbf{V}}_i'$$

```
Cmd> z1 <- U[,1]%*%V[,1]'*D[1,1]

Cmd> z1
(1,1)       41.058      -4.6789      -14.851
```

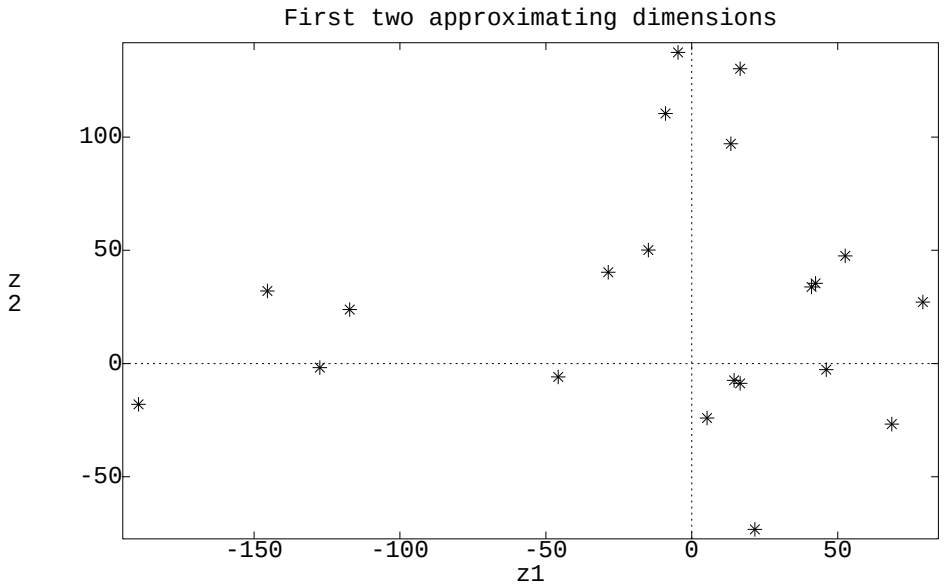
```
(2,1)        79.17       -9.022      -28.636
(3,1)      -145.41        16.57       52.593
(4,1)      -189.58       21.604       68.573
(5,1)      -127.42        14.52       46.088
(6,1)       -45.75       5.2136       16.548
(7,1)      -117.26       13.362       42.412

Cmd> e <- X-z1

Cmd> trace(e'%*%e)
(1)        94788

Cmd> 75739+19049
(1)        94788

Cmd> z2 <- U[,2]%*%V[,2]'*D[2,2]

Cmd> z2
(1,1)       33.809       137.43       50.175
(2,1)       27.167       110.43       40.318
(3,1)       32.026       130.18       47.529
(4,1)       -18.01      -73.204      -26.728
(5,1)      -1.8313       -7.444      -2.7179
(6,1)      -5.9139      -24.039      -8.7767
(7,1)       23.882       97.076       35.443

Cmd> e <- X-z1-z2

Cmd> trace(e'%*%e)
(1)        19049

Cmd> z1+z2
(1,1)       74.867       132.75       35.325
(2,1)       106.34        101.4       11.682
(3,1)      -113.38       146.75       100.12
(4,1)      -207.59        -51.6       41.845
(5,1)      -129.25       7.0764        43.37
(6,1)      -51.664      -18.825       7.7712
(7,1)      -93.375       110.44       77.855

Cmd> sum(z1*z2)
(1,1)   2.7285e-12  -6.8212e-13   -1.819e-12

Cmd> plot(vector(z1),vector(X),xlab:"z1",\
ylab:"X",title:"rank 1 approximation to X")
```




```
Cmd> plot(vector(z1+z2),vector(X),xlab:"z1+z2",\
ylab:"X",title:"rank 2 approximation to X")
```




```
Cmd> plot(z1:vector(z1),z2:vector(z2),\
title:"First two approximating dimensions")
```



OK, that's all good clean fun. Now what about principal components?

Principal components are new variables computed as linear combinations of the original variables. The components are chosen so that the first few lose as little information as possible from the entire data set.

More specifically

• Principal components are the linear combinations of the original variables that have the most variance (subject to some restrictions on the coefficients),

or

• Principal components are the linear combinations of the original variables that provide the best reconstruction of the original data in a least squares sense.

Doesn't that last one just scream SVD?

The first thing we do is center the data by removing variable means:

$$\tilde{\mathbf{X}} = \mathbf{X} - \mathbf{1}_n \overline{\mathbf{x}}'$$

We'll drop the tilde and just assume that the variables have zero mean.

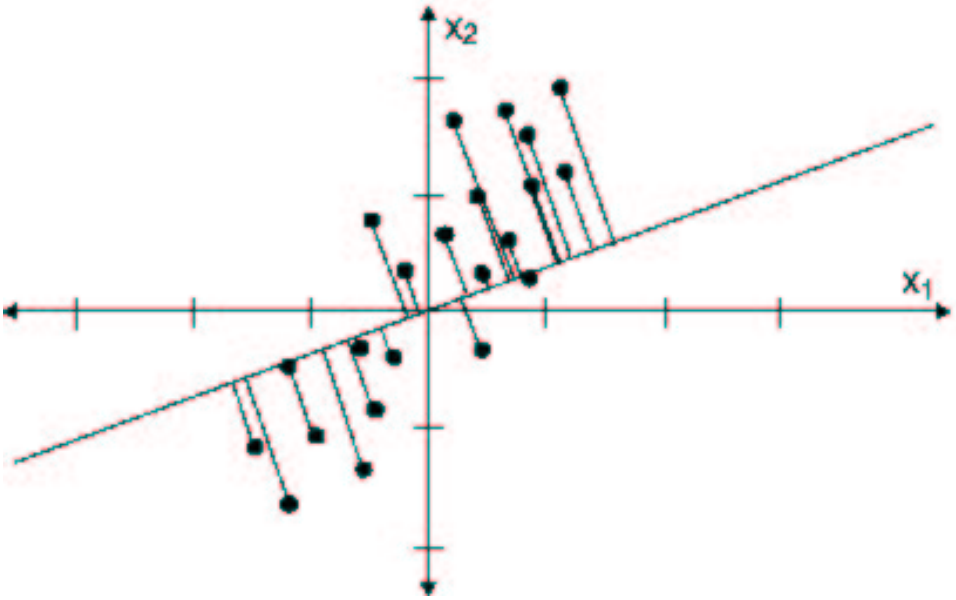
If at some later time you need to work with the original uncentered variables, you can put the mean back in.

The rank one approximation idea says to find a linear combination of the original variables, such that when we reexpress along that linear combination, we come closest to the data.

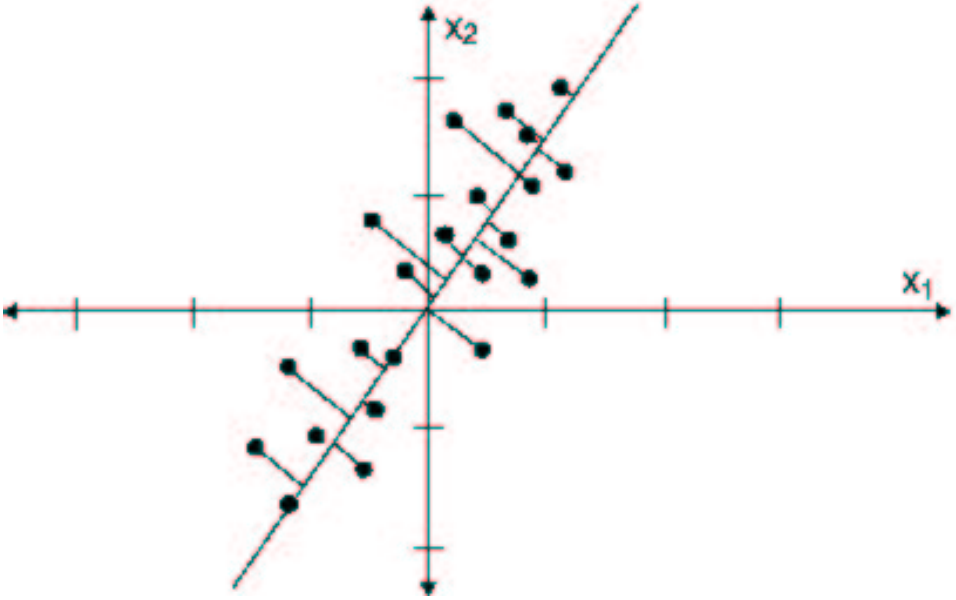
The (squared) distance between the data and the approximation is minimized when we choose the linear combination forming the variable to be the first right singular vector and the coordinates the first singular value times the first left singular vector.

Distance is perpendicular distance from the line, not vertical distance as in linear regression.

This direction is not so good.



This direction is better.



Note that the squared distance of a point from the origin (the mean) is the sum of the squared distances of the point to the line, and the coordinate on the line to the origin (Pythagorean Theorem).

Thus to get the projections onto the line as close as possible to the data, you must spread out the projected coordinates as much as possible.

So the first principal component is the direction in which the data spread out the most.

Algebraically, we want to maximize $a'\mathbf{S}a$ subject to $a'a = 1$. This is maximized when a is the first eigenvector of $\mathbf{S}$.

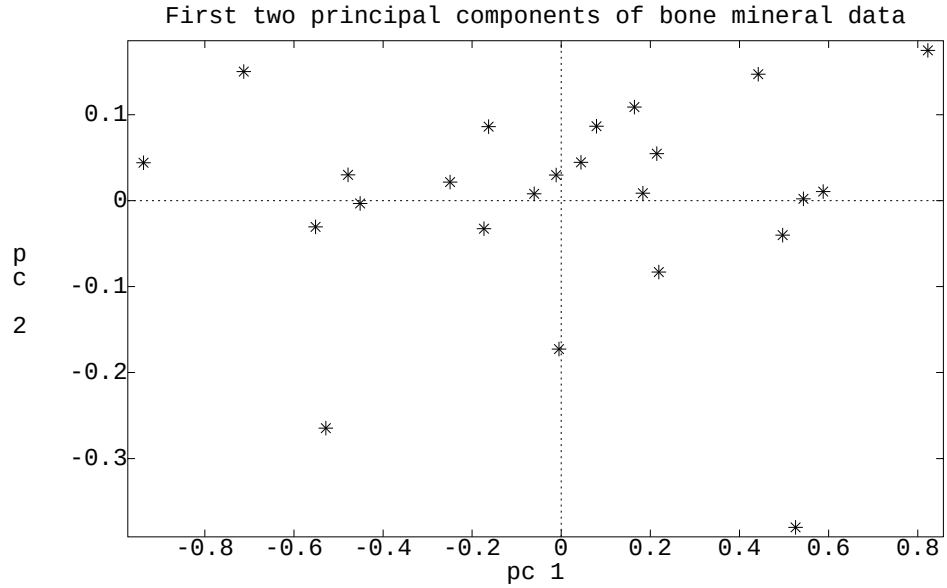
For the second princpal component, we want to maximize $b'\mathbf{S}b$ subject to $b'b = 1$ and $b'\mathbf{S}a = 0$. This is maximized when b is the second eigenvector of $\mathbf{S}$.

Of course, $\mathbf{S} = \mathbf{X}'\mathbf{X}/(n-1)$, so the eigenvectors of $\mathbf{S}$ are the eigenvectors of $\mathbf{X}'\mathbf{X}$, which are the right singular vectors of $\mathbf{X}$.

So the maximal variance and best approximant criteria both lead the the same procedure.

From either point of view, the coordinates of different principal components form uncorrelated (or orthogonal, remember everything has mean 0) vectors.
Thinking via variance matrices, the coordinates are $\mathbf{X}a$ and $\mathbf{X}b$, which have covariance $a'\mathbf{S}b = 0$.
Thinking via SVD, the coordinates are $\mathbf{UD}$; rescaling the columns of $\mathbf{U}$ doesn't remove their orthogonality
Use the bone mineral data of Table 6-16 (p 349).

```
Cmd> X <- readdata("",x1,x2,x3,x4,x5,x6)
Read from file "~/5401/JW5data/T6-16.DAT"
Column 1 saved as REAL vector x1
Column 2 saved as REAL vector x2
Column 3 saved as REAL vector x3
Column 4 saved as REAL vector x4
Column 5 saved as REAL vector x5
Column 6 saved as REAL vector x6

Cmd> X <- hconcat(x1,x2,x3,x4,x5,x6)

Cmd> S <- tabs(X,covar:T)

Cmd> print(eigen(S),format:"f6.3")
component: values
(1)  0.205  0.016  0.011  0.004  0.003  0.001
component: vectors
(1,1)  0.213  0.360 -0.464  0.458  0.454 -0.439
(2,1)  0.187  0.466 -0.057  0.072  0.216  0.832
(3,1)  0.694 -0.472 -0.454 -0.014 -0.261  0.146
(4,1)  0.618  0.035  0.738  0.074  0.200 -0.167
(5,1)  0.178  0.276 -0.177 -0.881  0.223 -0.188
(6,1)  0.156  0.594  0.004  0.059 -0.768 -0.173

Cmd> Xc <- X - tabs(X,mean:T)'

Cmd> P <- Xc %*% eigen(S)$vectors

Cmd> plot(P[,1],P[,2],xlab:"pc 1",ylab:"pc 2",\
title:"First two principal components of bone mineral data")
```




```
Cmd> print(svd(Xc,right:T),format:"f6.3")
STRUCTURE:
component: values
(1)  2.174  0.605  0.495  0.311  0.273  0.171
component: rightvectors
(1,1) -0.213  0.360 -0.464  0.458  0.454  0.439
(2,1) -0.187  0.466 -0.057  0.072  0.216 -0.832
(3,1) -0.694 -0.472 -0.454 -0.014 -0.261 -0.146
(4,1) -0.618  0.035  0.738  0.074  0.200  0.167
(5,1) -0.178  0.276 -0.177 -0.881  0.223  0.188
(6,1) -0.156  0.594  0.004  0.059 -0.768  0.173
```

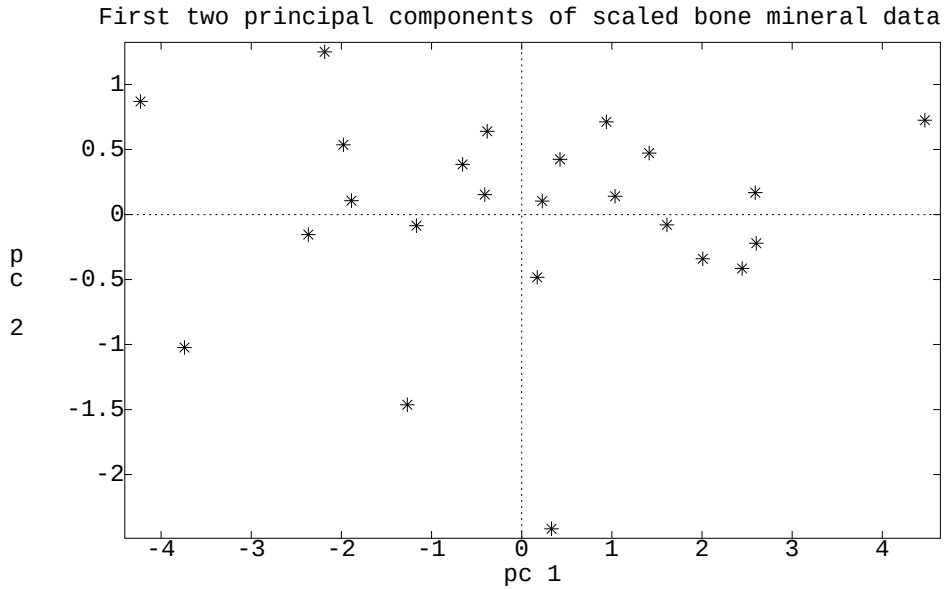
Note the sign changes in the vectors.

```
Cmd> ev <- eigenvals(S)

Cmd> sv <- svd(Xc)

Cmd> print(ev/sum(ev),format:"f6.3")
(1)  0.853  0.066  0.044  0.018  0.013  0.005

Cmd> print(sv^2/sum(sv^2),format:"f6.3")
(1)  0.853  0.066  0.044  0.018  0.013  0.005

Cmd> Xcs <- Xc/tabs(Xc,stddev:T)'

Cmd> print(svd(Xcs,right:T),format:"f6.3")
component: values
(1) 10.191  3.785  2.799  2.684  1.863  1.147
```

```
component: rightvectors
(1,1) -0.413  0.108  0.383  0.709 -0.209 -0.353
(2,1) -0.434  0.296  0.097  0.071  0.736  0.410
(3,1) -0.407 -0.577  0.129 -0.026 -0.355  0.598
(4,1) -0.417 -0.425  0.140 -0.462  0.275 -0.581
(5,1) -0.399  0.010 -0.891  0.171 -0.077 -0.108
(6,1) -0.378  0.622  0.117 -0.499 -0.455  0.026
```

Scaling affects principal components. Here the first two components are more or less average and humerus versus others.

```
Cmd> R <- tabs(Xcs,covar:T)

Cmd> ev <- eigenvals(R)

Cmd> print(ev/sum(ev),format:"f6.3")
VECTOR:
(1)  0.753  0.104  0.057  0.052  0.025  0.010

Cmd> P <- Xcs %*% eigen(R)$vectors

Cmd> plot(P[,1],P[,2],xlab:"pc 1",ylab:"pc 2",\
title:"First two principal components of scaled bone mineral data")
```



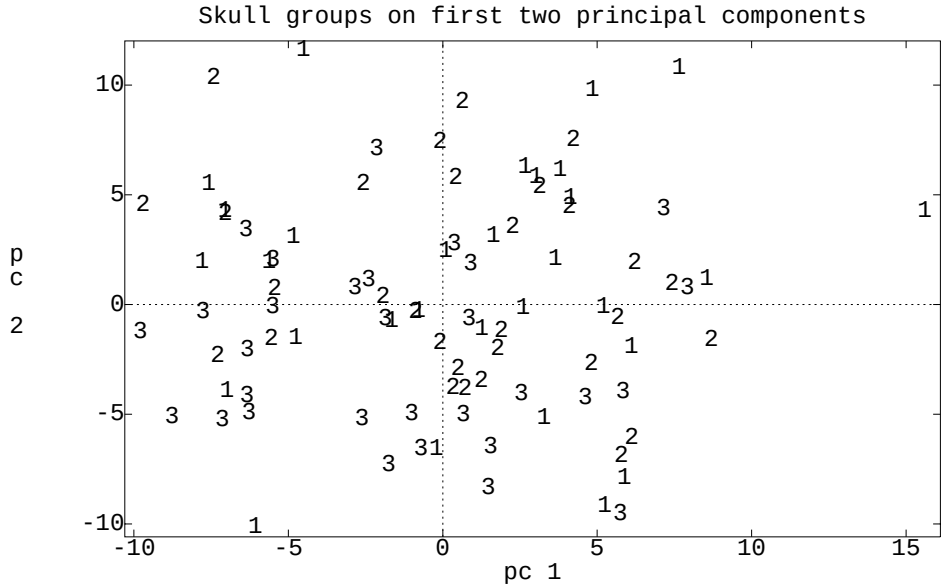
Change one of the units from milliX to to microX; this changes the principal components radically.

```
Cmd> Xb <- X;Xb[,1] <- Xb[,1]*1000

Cmd> Xbc <- Xb - tabs(Xb,mean:T)'
```

```
Cmd> print(svd(Xbc,right:T),format:"f6.3")
component: values
(1) 597.372  1.422  0.560  0.374  0.288  0.185
component: rightvectors
(1,1) -1.000  0.003 -0.000  0.001  0.000 -0.000
(2,1) -0.001 -0.075  0.366 -0.167  0.031  0.912
(3,1) -0.002 -0.705 -0.572 -0.313 -0.251  0.123
(4,1) -0.002 -0.692  0.465  0.493  0.190 -0.160
(5,1) -0.001 -0.124  0.191 -0.683  0.653 -0.234
(6,1) -0.001 -0.057  0.535 -0.406 -0.688 -0.270
```

The first few principal components are not always so dominant. Here are the skulls data:

```
Cmd> readdata("",x1,x2,x3,x4,x5)
Read from file "~/5401/JW5data/T6-13.DAT"
Column 1 saved as REAL vector x1
Column 2 saved as REAL vector x2
Column 3 saved as REAL vector x3
Column 4 saved as REAL vector x4
Column 5 saved as REAL vector x5

Cmd> X <- hconcat(x1,x2,x3,x4)

Cmd> Xc <- X - tabs(X,mean:T)'

Cmd> svd(Xc,right:T)
component: values
(1)    48.923    46.814     41.56    27.456
component: rightvectors
(1,1)    0.1247  -0.72861   0.63545   0.22311
(2,1)   0.36432  -0.55131  -0.74239   0.11039
(3,1)   0.92106   0.33155   0.20389 -0.012764
(4,1)  0.058116  -0.23507   0.05909  -0.96844

Cmd> V <- svd(Xc,right:T)$rightvectors

Cmd> UD <- Xc %*% V

Cmd> chplot(UD[,1],UD[,2],x5,xlab:"pc 1",ylab:"pc 2",\
title:"Skull groups on first two principal components")
```



Scaling makes the components more equal in scale. This suggests that the (scaled) point cloud is rather spherical in shape.

```
Cmd> Xcs <- Xc/tabs(Xc,stddev:T)'

Cmd> svd(Xcs,right:T)
component: values
(1)   10.843    9.5994    8.8462    8.2481
component: rightvectors
(1,1)   0.60372   0.22912  -0.40156  -0.64944
(2,1)   0.47444  -0.39811   0.76584  -0.17294
(3,1)  0.050172  -0.87585  -0.47878  0.033682
(4,1)   0.63868   0.14796   -0.1517   0.73972
```