Z-scores, Matrix Inverses & Least Squares Regression (use R as you proceed)

```
X  #we define X as:
[1,]   3    3  #plot these two columns!
[2,]   5    2  #to see the scatterplot
[3,]   1    1
> apply(X,2,mean)
[1] 3 2
> apply(X,2,sd) #so, var→4 1
[1] 2 1
> X.d<-sweep(X,2,colMeans(X)) #see ?sweep
> X.d  #of course this is our L%*%X (see below)
[1,]   0    1  #note: column sums now zero
[2,]   2    0
[3,]  -2   -1
Alternatively, given
L <- function(n)diag(n) - matrix(1/n,n,n)
So we can use L(3)%*%X as our X.d  #TRY IT!
> t(X.d)%*%X.d #same as t(X.d)%*%X (or X'L X)
     [,1] [,2] #square, symmetric
[1,]   8    2 #NB: SSqr/(n-1)=8/2 for column.1
[2,]   2    2
> Z.scrs <-scale(X) #i.e., 'zscores'
[1,]  0.0   1 #each z score=(dev.score/sd)
[2,]  0.5   0 #sum(sqd.col.z-scores=sqrt(n-1))
[3,] -0.5  -1
> Z<-Z.scrs/sqrt(2) #because n = 3 here
> Z<-Z.scrs/sqrt(2) #i.e. sqrt(n-1)
> Z       #Note that col. sums of sqrs = 1.00
[1,]  0.000  0.707
[2,]  0.707  0.000
[3,] -0.707 -0.707
> t(Z)%*%Z #equals cor(X)(same as cor(Z))
     [,1] [,2]  #call this matrix R
[1,]  1.0  0.5
[2,]  0.5  1.0
>R-inverse<-solve(R) #diag values always=or > 1
       [,1]   [,2]
[1,]  1.333 -0.667  #So R %*% R-Inverse = I
[2,] -0.667  1.333
>S.min.2<-diag(solve(R)) #a vector here (1.33)
> S.sqrd<-diag(1/S.min.2) #diag matrix
     [,1] [,2]  #Be sure to see ?diag (and try)
[1,] 0.75 0.00 #complements of smc's! (1-smc's)
[2,] 0.00 0.75 #next, subtract these from 1's.
> D.smc=diag(2)-S.sqrd
    #NB: D.smc = I - S^2 [diag(2)=I,of order 2]
     [,1] [,2] #squared multiple correlations
[1,] 0.25 0.00  #in the diagonals
[2,] 0.00 0.25
```

Begin from ANY X for which columns are *no interdependent.*
(I'd recommend a small matrix, not necessarily the one I show at left; make sure #rows >#cols)
Compute vector of means and subtract to get deviation scores, say X.d (← see example). We see that X'L X is always square, symmetric. & X'LX=(n-1)*var(X) . Be sure to do all the Operations. (L is known to be *idempotent*; look it up)

Z scores entail rescaling columns of X.d so that each entry in Z of form [Z.scrs] =(X[i,j]-mean)/sd. Divide by square root of (n-1) to get what I call matrix Z, where, Z'Z = R, the matrix of product-moment correlations,i.e.,cor(X) ← example shows that matrix R is computed this way. Examine those columns of Z.
Now compute the <u>inverse</u> of R (w/solve) to get another square, symmetric matrix. Define S^{-2} = diag(R^{-1}), (see S.min.2 on left; also S.sqrd on left). S^2=inverse(S^{-2})[a diag] Compute D.smc=I-S^2, a diag matrix of squared multiple correlations when predicting each column of Z (or X) from all other columns.

```
>sqrt(diag(2)-S.sqrd)
     [,1] [,2]#square roots=multiple correlatns
[1,]  0.5  0.0 #but with only one predictor,
[2,]  0.0  0.5 #these=original product-moment
                correlations (say, p-m r's)

>Z.resds<-Z%*%solve(R)%*%S.sqrd
       [,1]   [,2] #residuals for predicting
[1,] -0.354  0.707 #col.vectors of matrix Z
[2,]  0.707 -0.354 #a DIRECT product operation
[3,] -0.354 -0.354
```

VERIFY ABOVE USING STANDARD R functions!

```
> residuals(lsfit(Z[,2],Z[,1]))
[1] -0.354  0.707 -0.354 [now as row of course]
> residuals(lsfit(Z[,1],Z[,2]))
[1]  0.707 -0.354 -0.354 #for the two columns
>B.wts <-diag(2) - solve(R)%*%S.sqrd
#could also get Predicted Z columns by
#subtraction
>Z%*%B.wts #examine this carefully
       #l.s. predicted Z's in each column
       [,1]   [,2]
[1,]  0.354  0.000 #obtained by subtracting
[2,]  0.000  0.354 #residuals from original Z's
[3,] -0.354 -0.354
```

The geometry for these ops is especially simple when there are only two columns in X.

Satisfy yourself that you can describe all geometric relations using (this?) example.

Next we see that Z.resds=$ZR^{-1}S^2$,the residuals, predicting each variable from all others. So by subtraction we obtain ALL predicted values: Z.pred = Z - $ZR^{-1}S^2$, Or Z.pred = $Z(I- R^{-1}S^2)$ where matrix in ( )'s is B = $I - R^{-1}S^2$ with columns as vectors of l.s. regression coefs for predicting each variable from all others [B.wts on left]

***<u>Note the generality</u>*: All the preceding algebra works for any (data) matrix X, regardless of how many variables there are in (as long as X's cols are not mutually inter-dependent).**

The matrix $I - SR^{-1}S$ contains PARTIAL correlations between each pair of variables while 'holding other variables constant'. (Recall <u>dihedral</u> angles in the geometry hndout)

```
 #Note that the function below gets all the key results available from
input correlation matrix (if X itself not available)[copy/paste to R]
allr.wts <-function(rr){
# rr assumed to be correlation matrix of full column rank; function
# generates all sqrd multiple correlations (D.smc), predicting each variable
# from all p-1 others; also, all vectors of regression coefficients (B.wts)
# for same predictions; also all p-2 order partial correlations (R.part2)
pp<-ncol(rr) #usually p denotes number of variables/
r.inv<-solve(rr) #inverse of rr
S.min.2<-diag(r.inv) # a vector
S.sqrd<-1/S.min.2 #also a vector
S<-sqrt(S.sqrd) #still a vector
S.sqrd<-diag(S.sqrd) #now diag matrix
S<-diag(S) #now diag matrix
D.smc<-diag(pp)-S.sqrd
B.wts<-diag(pp) - r.inv%*%S.sqrd #Note: diagonal values always zero here
R.part2<-round(diag(pp) - S%*%r.inv%*%S,3) #a square,symmetric matrix
#special virtue of R.part2 wts is that these are regression wts (for post-
```

```
#multipling the matrix Z%*%S.minus1) that always fall in interval [0,1]
list(D.smc=diag(D.smc),B.wts=B.wts,R.part2)  } #So we try the function:


>allr.wts(cor(X))
$D.smc   [1] 0.25 0.25
$B.wts
[1,]  0.0  0.5
[2,]  0.5  0.0
```

$R.part2 #study the code in the function; this is an example of **I - SR^{-1}S**

```
[1,]  0.0  0.5
[2,]  0.5  0.0  #Of course we will appreciate this more when p is larger
```

<u>Exercise:</u>

Try these ops for (first few rows?) of trees data, or *   *most quantitative data matrices, but use no more than say 10 rows; w/ say, 3 columns in your trials. You can of course do this multiple times, and if you do study what you get with reference to my notes given here.