# APPENDIX A

**Koopmans Data**

Distance traveled by field mice

| FALL | | WINTER | | SPRING | SUMMER | |
|---|---|---|---|---|---|---|
| 0 | 15 | 0 | 30 | 15 | 60 | 0 |
| 0 | 8 | 34 | 15 | 0 | 21 | 0 |
| 21 | 29 | 0 | 15 | 15 | 15 | 21 |
| 0 | 15 | 15 | 8 | 18 | 15 | 17 |
| 15 | 46 | 15 | 21 | 109 | 15 | 0 |
| 0 | 39 | 87 | 15 | 15 | 33 | 15 |
| 15 | 30 | 15 | 15 | 0 | 24 | 106 |
| 15 | 15 | 0 | 15 | 15 | 33 | 17 |
| 0 | 11 | 5 | 22 | 47 | 42 | 21 |
| 8 | 0 | 0 | 0 | 30 | 54 | 21 |
| 0 | | 0 | 15 | 15 | 11 | |
| 0 | | 15 | 15 | 34 | 32 | |
| 15 | | 8 | 33 | 47 | 8 | |
| 21 | | 0 | 21 | 42 | 71 | |
| 0 | | 15 | 15 | 0 | 150 | |
| 34 | | 47 | 0 | 22 | 18 | |
| 0 | | 0 | 0 | 34 | 12 | |

# APPENDIX B

**Splus program for ZIP model**

Splus program written to fit the zero inflated Poisson model in Chapter Three.
Special thanks to Dr. Mark X. Norleans for providing the GEE procedure version 2.6 in the GEE-ZIP model (through STATLIB).

```
#PROGRAM STARTS HERE

 fitzipA  <- function(Amat, ww, re, phase, ph){


#This function fits the Zip model to the data Amat
# ww  is the column of responses
# phase is the column of the phase indicator, ph is the nmuber of phase
# if phase = 0 and ph< 0  means no phase indicator
# re is the column(s) of the covariate in data Amat
# For example in the egg data re is from column 9 -12, ph is column13
# y is column 1
```

#SET ARRAY FOR THE ESTIMATED REGRESSION COEFFICIENTS

```
upl <- 25
upper <- upl + 1
wid <- length(re) + 1
apsbeta.all <- matrix(0, upper, wid)
abigamma.all <- matrix(0, upper, wid)
if(phase <= 0) {
        print("your phase column is <= 0, this means no phase indicator"
              )
        amat0 <- Amat
}
else amat0 <- fphase(Amat, phase, ph)
```

#OBTAIN INITIAL VALUES FOR THE ESTIMATES

```
        anonz <- fnz(amat0, ww)
        azro <- fzro(amat0, ww)
        aresp <- amat0[, ww]
        aone1 <- as.vector(rep(1, length(aresp)))
        ax <- cbind(aone1, amat0[, re])
        appois.glm <- glm(anonz[, ww] ~ anonz[, re], poisson)
        apsbeta <- as.vector(appois.glm$coefficients)
        pmatx <- fpmatx2(amat0, 20, re)
        px <- ncol(pmatx) - 1
        apsqeta <- pmatx %*% apsbeta
        pnot <- (sum(fobs01(amat0, 20, ww)) - n * sum(exp( - exp(apsqeta))))/(
                nrow(pmatx) * n)
        pzero <- rep(0, px)
        abigamma <- c(log(abs(pnot)), pzero)
        apsbeta.all[1,  ] <- apsbeta
        abigamma.all[1,  ] <- abigamma
        i <- 0
        if(length(re) == 1)
                xstar <- as.matrix(c(amat0[, re], azro[, re]))
        else xstar <- as.matrix(rbind(amat0[, re], azro[, re]))
        yo <- as.vector(c(rep(0, length(aresp))))
        numyo <- length(yo)
#SET THE RESPONSE FOR THE  MORTALITY (LOGISTIC) PART FOR
ESTIMATING GAMMA

        k <- 0
        conditions <- T
        while(conditions) {
                k <- k + 1
                if(aresp[k] == 0)
                        yo[k] <- 1
                else yo[k] <- 0
                if(k == numyo)
                        conditions <- F
        }
        ystar <- c(yo, rep(1, nrow(azro)))
        numay <- length(aresp)
```

---

#SET UP THE WEIGHTS FOR THE MORTALITY (LOGISTIC) PART OF THE MODEL

---

```
z <- rep(0, numay)
```

```
        condition <- T
        while(condition) {
                j <- 0
                conds <- T
                while(conds) {
                        j <- j + 1
                        if(aresp[j] == 0)
                                z[j] <- 1/(1 + exp(( - (ax[j,  ] %*% abigamma)) -
                                  exp(ax[j,  ] %*% apsbeta)))
                        else z[j] <- 0
                        if(j == numay)
                                conds <- F
                }
                priorweights <- as.double(1 - z)
pweights <- as.double(c(1 - z, z[z != 0]))
```

#ITERATIVELY ESTIMATING THE REGRESSION COEFFECIENTS

```
                i <- i + 1
                apsbeta.old <- apsbeta
                abigamma.old <- abigamma
                appois.glm <- glm(aresp ~ amat0[, re], poisson, weights =
                        priorweights)
                alogis.glm <- glm(ystar ~ xstar, binomial(logit), weights =
                        pweights)
                apsbeta <- as.double(appois.glm$coefficients)
                abigamma <- as.double(alogis.glm$coefficients)
                torl <- max(max(abs((apsbeta - apsbeta.old)/apsbeta.old)), max(
                        abs((abigamma - abigamma.old)/abigamma.old)))
                apsbeta.all[i + 1,  ] <- apsbeta
                abigamma.all[i + 1,  ] <- abigamma
                if((ceiling(i/3) == 1) & (i == 1)) {
                        print("number of iterations for this fit:")
                        print(i)
                        print("the maximum ratio of change in the parameter:")
                        print(torl)
                        print("beta")
                        print(apsbeta)
                        print("gamma")
                        print(abigamma)
                }
```

```
            if((torl < e-15) | (i > upl))
                    condition <- F}
        alamhat <- exp(ax %*% apsbeta)
        aphat <- 1/(exp( - ax %*% abigamma) + 1)
        apredict <- (1 - aphat) * alamhat
        aresi <- aresp - apredict
        sqaresi <- aresi * aresi
        assr <- sum(sqaresi)
        aout <- cbind(aresp, apredict, alamhat, aphat, aresi, ax)
        print("the X matrix is")
        print(pmatx)
        print("the sum squared residual is")
        print(assr)
        alhat <- exp(pmatx %*% apsbeta)
        aponehat <- 1/(exp( - pmatx %*% abigamma) + 1)
        apredmn <- (1 - aponehat) * alhat
        resiamn <- (fmna1(amat0, 20, ww) - apredmn)
        amnout1 <- cbind(fmna1(amat0, 20, ww), fobs01(amat0, 20, ww), fmna1pos(
                amat0, 20, ww), alhat, aponehat, apredmn, resiamn, pmatx)
        ssramn <- sum((resiamn * resiamn))
        print("1. Mean of obs, 2. Number of zeros, 3. Mean of positive obs, 4. lambda hat,
5. p hat, 6. Prediction = lambda hat*p hat, 7. Residual, 8. Covariates")
```

#PRINT THE RESULTS

```
        print(amnout1)
        print("sum squared residual of the mean")
        print(ssramn)
        print(appois.glm)
        print(summary(appois.glm))
        print(alogis.glm)
        print(summary(alogis.glm))
        dev <- deviance(appois.glm) + deviance(alogis.glm)
        nulldev <- appois.glm$null.deviance + alogis.glm$null.deviance
        print("deviance is")
        print(dev)
        print(" null deviance is")
        print(nulldev)
        amnout1
}
```

# APPENDIX C

**Splus program for GEE-ZIP model**

Splus program written to fit the GEE-ZIP model in Chapter four.
Special thanks to Dr. Steve Gregorich for providing the macro em_covar in the SAS macro.

```
#PROGRAM STARTS HERE

 fzgeecom  <- function(Amat, ww,m, re, id, repeated, phase, ph){
cat("n#########################################################################
", fill=T)

cat("\nThis function fits the Zip model to the data Amat using the gee function
as a extension to repeated measures",fill=T)
cat("\n THE covariance structure is compoundsymmetric", fill=T)

cat("\n#########################################################################
", fill=T)

#This function fits the Zip model to the data Amat
# ww  is the column of responses
# phase is the column of the phase indicator, ph is the nmuber of phase
# if phase = 0 and ph< 0  means no phase indicator
# re is the column(s) of the covariate in data Amat
#repeated is the variable  of repeated measures eg. days, visit etc..
# id is the identification of subject
#repeated is the number of repeated measures of each subject

#SET UP THE ARRAY FOR THE REGRESSION COEFFICENTS

        upl <- 12
        upper <- upl + 2
        wid <- length(re) + 1
        apsbeta.all <- matrix(0, upper, wid)
        abigamma.all <- matrix(0, upper, wid)
        if(phase <= 0) {
```

```
cat("\nYour phase column is",phase," <= 0, this means no phase indicator",fill=T)

                amat0 <- as.matrix(Amat)
        }
        else amat0 <- fphase(Amat, phase, ph)

cat("\nColumn",ww, "is the column of responses", fill=T)
cat("Column(s)",re, "is the column of covariate",fill=T)
cat("Column",m, "is the column of mortality information",fill=T)
cat( "Column", id," is the identification of each subject",fill=T)
cat("Column",repeated,"is the indicator for the repeated measures for each
subject",fill=T)
```

---

#CALCULATE INITIAL VALUES

---

```
        almat <-  rbind(Amat,findzero(Amat, ww, id))
      almresp <- almat[,ww]
      aresp <- amat0[,ww]
      aresp1 <- findzero(Amat,ww,id)[,ww]

        aone1 <- as.vector(rep(1, nrow(amat0))

        ax <- cbind(aone1, amat0[, re])
        appois.glm <- glm(amat0[, ww] ~ amat0[, re], poisson )
        apsbeta <- as.vector(appois.glm$coefficients)

        pmatx <- fpmatx3(amat0, 80,4 ,re[-1])
        px <- ncol(pmatx) - 1
        apsqeta <- pmatx %*% apsbeta

        pnot <- (sum(fobs01(amat0, 80, ww)) - n * sum(exp( -
exp(apsqeta))))/(nrow(pmatx) * n)
      print(pnot)
        pzero <- rep(0, px)

        abigamma <- c(log(abs(pnot)), pzero)

        apsbeta.all[1,  ] <- apsbeta
        abigamma.all[1,  ] <- abigamma
```

---

#SET UP THE RESPONSES FOR THE MORTALITY  PART OF THE MODEL

---

```
        i <- 0
```

```
            xstar <- as.matrix(almat[, re])
        yo <- as.vector(c(rep(0, length(almresp))))
        numyo <- length(yo)
cat("NUMYO is number of zero response =",numyo,fill=T)

        k <- 0
        conditions <- T
        while(conditions) {
            k <- k + 1
            if(almresp[k] == 0)
                yo[k] <- 1
            else yo[k] <- 0
            if(k == numyo)
                conditions <- F
        }
        ystar <- yo
        numay <-   nrow(amat0)
    numay1 <- nrow(almat)-nrow(amat0)
```

#SET UP WEIGHTS FOR THE MORTALITY PART OF THE MODEL

```
        z <- rep(0, numay)
     v <- rep(0, numay1)

        condition <- T
        while(condition) {
            j <- 0
            conds <- T
            while(conds) {
                j <- j + 1
                if(aresp[j] == 0)
                    z[j] <- 1/(1 + exp(( - (ax[j,  ] %*% abigamma)) -
                      exp(ax[j,  ] %*% apsbeta)))
                else z[j] <- 0
                if(j == numay)
                    conds <- F
            }


        cond1s <- T
            j <- 0
            while(cond1s) {
                j <- j + 1
```

```
                    if(aresp1[j] == 0)
                    {       v[j] <- 1/(1 + exp(( - (ax[j,  ] %*% abigamma)) -
                              exp(ax[j,  ] %*% apsbeta)))}
                    else v[j] <- 0
                    if(j == numay1)
                            cond1s <- F
                }

                prws <- as.double(1 - z)
lpw <- length(prws)
                pws <- as.double(c(1 - z, v))
lp <- length(pws)
```

#START THE ITERATION TO ESTIMATE THE COEFFECIENTS

```
               i <- i + 1
                apsbeta.old <- apsbeta
                abigamma.old <- abigamma

                appois.gee <- gee(amat0[,ww] ~ amat0[, re], poisson, subject =amat0[,id],
repeated= amat0[,repeated], weights = prws,wc="com")

                alogis.gee <- gee(ystar ~ xstar, binomial(logit), weights = pws,
subject=almat[,id], repeated=almat[,repeated],wc="com")
                apsbeta <- as.double(appois.gee$coefficients)
                abigamma <- as.double(alogis.gee$coefficients)

                torl <- max(max(abs((apsbeta - apsbeta.old)/apsbeta.old)), max(
                        abs((abigamma - abigamma.old)/abigamma.old)))

                apsbeta.all[i + 1,  ] <- apsbeta
                abigamma.all[i + 1,  ] <- abigamma
                if((ceiling(i/4) == i/4) | (i == 1)) {
cat("\nNumber of iterations for this fit:",i, fill=T)
cat("The maximum ratio of change in the parameter:",round(torl,9), fill=T)
                        cat("beta:",round(apsbeta,4),"gamma:",round(abigamma,4),fill=T)
                }
                if((max(torl) < e-7) | (i > upl))
                        condition <- F
        }
        alamhat <- exp(ax %*% apsbeta)
        aphat <- 1/(exp( - ax %*% abigamma) + 1)
```

```
        apredict <- (1 - aphat) * alamhat
        aresi <- aresp - apredict
        sqaresi <- aresi * aresi
        assr <- sum(sqaresi)
        aout <- cbind(aresp, apredict, alamhat, aphat, aresi, ax)

        cat("\nThe sum squared residual is",round(assr,4),fill=T)

        alhat <- exp(pmatx %*% apsbeta)
        aponehat <- 1/(exp( - pmatx %*% abigamma) + 1)
        apredmn <- (1 - aponehat) * alhat
        resiamn <- (fmna1(amat0, 80, ww) - apredmn)
        amnout1 <- cbind(fmna1(amat0, 80, ww), fobs01(amat0, 80, ww), fmna1pos(
                amat0, 80, ww), alhat, aponehat, apredmn, resiamn, pmatx)
        ssramn <- sum((resiamn * resiamn))
```

---

#PRINT THE RESULTS

---

```
cat("\n1. Mean of obs",fill=T)
cat(" 2. Number of zeros",fill=T)
cat(" 3. Mean of positive obs",fill=T)
cat(" 4. lambda hat",fill=T)
cat(" 5. p hat",fill=T)
cat(" 6. Prediction = lambda hat*p hat",fill=T)
cat(" 7. Residual",fill=T)
cat(" 8. Covariates",fill=T)

        #print(amnout1)
        cat("\nSum squared residual of the mean",round(ssramn,4),fill=T)
        print(appois.gee)
        print(summary(appois.gee))
betacov(appois.gee)
print(betacov(appois.gee))
wc(appois.gee)
print(wc(appois.gee))
        print(alogis.gee)
        print(summary(alogis.gee))

betacov(alogis.gee)
print(betacov(alogis.gee))
wc(alogis.gee)
print(wc(alogis.gee))
amnout1}
```

# APPENDIX D

**SAS macro for Principal Component Analysis**

A sample SAS macro for the simulation study of principal component analysis on heavy zero data. Special thanks to Mr. Steve Gregorich for providing the EM_COV.SAS macro

```
 ************************************************************;
*PROGRAM STARTS HERE;
************************************************************;
options ls =80 nodate;


************************************************************;
/* SAS MACRO PCMISS TO RECOVER CORRELATION STRUCTURE         */
************************************************************;

%macro pcmiss;
%include "/usr1/home/grad/swang/em-cov.sas";
%do ivar = 1 %to 101;

************************************************************;
/*    GENERATE NORMAL DATA
      WITH VARYING MEAN AND VARIANCE=V     */
************************************************************;


data nor;
v  = 1;
do i= 1 to 100;
x1 = 4 + sqrt(v)*rannor(453451*&ivar);
x2 = 5 + sqrt(v)*rannor(597970*&ivar);
x3 = 6 + sqrt(v)*rannor(786700*&ivar);
x4 = 7 + sqrt(v)*rannor(176800*&ivar);
x5 = 8 + sqrt(v)*rannor(156000*&ivar);
x6 = 9 + sqrt(v)*rannor(297546*&ivar);
x7 = 10 + sqrt(v)*rannor(234540*&ivar);
x8 = 11 + sqrt(v)*rannor(467700*&ivar);
x9 = 12 + sqrt(v)*rannor(823400*&ivar);
```

```
x10 =14 + sqrt(v)*rannor(245000*&ivar);
output;
end;
data new;set nor;
keep x1-x10;

*************************************************************;
/*    GENERATE MORTALITY DATA
      WITH PROBABILITY VARYING MORTALITY PROBABILITY
       WHICH IS DIRECTLY PROPORTIONAL TO THE MEAN OF X      */
*************************************************************;

data bin2;
p1 = 0.5;
p2 = 0.4;
p3 = 0.35;
p4 = 0.28;
p5 = 0.25;
p6 = 0.22;
p7 = 0.2;
p8 = 0.18;
p9 = 0.165;
p10 = 0.15;
do i= 1 to 100;
m1  = ranuni(109766*&ivar);
m2  = ranuni(798850*&ivar);
m3  = ranuni(7456400*&ivar);
m4  = ranuni(1434300*&ivar);
m5  = ranuni(1032300*&ivar);
m6  = ranuni(2245656*&ivar);
m7  = ranuni(2234750*&ivar);
m8  = ranuni(4567700*&ivar);
m9  = ranuni(8464400*&ivar);
m10  = ranuni(2034300*&ivar);
if (m1 < p10) then m1= 0; else m1 =1;
if (m2 < p9) then m2= 0; else m2 =1;
if (m3 < p8) then m3= 0; else m3 =1;
if (m4 < p7) then m4= 0; else m4 =1;
if (m5 < p6) then m5= 0; else m5 =1;
if (m6 < p5) then m6= 0; else m6 =1;
if (m7 < p4) then m7= 0; else m7 =1;
if (m8 < p3) then m8= 0; else m8 =1;
if (m9 < p2) then m9= 0; else m9 =1;
if (m10 < p1) then m10= 0; else m10 =1;
```

```
output;
end;


data newm2;set bin2;
det0 = ranuni(896043*&ivar);

    if (m1=0 and m2=0 and m3=0 and m4=0 and m5=0 and m6=0 and m7=0 and m8 =0
and m9=0 and m10=0 and  det0<0.1) then m1 =1;
else if (m1=0 and m2=0 and m3=0 and m4=0 and m5=0 and m6=0 and m7=0 and m8 =0
and m9=0 and m10=0 and  det0<0.2) then m2 =1;
else if (m1=0 and m2=0 and m3=0 and m4=0 and m5=0 and m6=0 and m7=0 and m8 =0
and m9=0 and m10=0 and det0<0.3) then m3 =1;
else if (m1=0 and m2=0 and m3=0 and m4=0 and m5=0 and m6=0 and m7=0 and m8 =0
and m9=0 and m10=0 and  det0<0.4) then m4 =1;
else if (m1=0 and m2=0 and m3=0 and m4=0 and m5=0 and m6=0 and m7=0 and m8 =0
and m9=0 and m10=0 and  det0<0.5) then m5 =1;
else if (m1=0 and m2=0 and m3=0 and m4=0 and m5=0 and m6=0 and m7=0 and m8 =0
and m9=0 and m10=0 and  det0<0.6) then m6 =1;
else if (m1=0 and m2=0 and m3=0 and m4=0 and m5=0 and m6=0 and m7=0 and m8 =0
and m9=0 and m10=0 and  det0<0.7) then m7 =1;
else if (m1=0 and m2=0 and m3=0 and m4=0 and m5=0 and m6=0 and m7=0 and m8 =0
and m9=0 and m10=0 and  det0<0.8) then m8 =1;
else if (m1=0 and m2=0 and m3=0 and m4=0 and m5=0 and m6=0 and m7=0 and m8 =0
and m9=0 and m10=0 and  det0<0.9) then m9 =1;
else if (m1=0 and m2=0 and m3=0 and m4=0 and m5=0 and m6=0 and m7=0 and m8 =0
and m9=0 and m10=0 and  det0<0.9) then m10=1;
keep m1-m10;


Proc iml;
use new;
read all into x;
close new;

use newm2;
read all into z2;
close newm2;

*************************************************************;
/*    THE TRUE CORRELATION MATRIX      10 by 10 HIGH NOISE   */
*************************************************************;
```

```
SIGMA ={ 1.0   0.9 0.9    0.9   0.9  0.9  0.40  0.4 0.4 0.4,
        0.9  1.0  0.95   0.9   0.9  0.9  0.4  0.4 0.4 0.4,
        0.9   0.95  1.0   0.85  0.9  0.96 0.4 0.4 0.4 0.4,
        0.9   0.9  0.85  1.0    0.8  0.9  0.4 0.4 0.4 0.4,
        0.9   0.9  0.9   0.8    1.0  0.9  0.4 0.4 0.4 0.4,
        0.9   0.9  0.96   0.9   0.9  1.0  0.4  0.4 0.4 0.4,
        0.40  0.40   0.4  0.4 0.4  0.4 1.0  0.95 0.89  0.9,
        0.40  0.40   0.4  0.4 0.4  0.4 0.95 1.0  0.89  0.9,
        0.40  0.40  0.4  0.4 0.4  0.4 0.89 0.89 1.0   0.7,
        0.40  0.40   0.4  0.4 0.4  0.4 0.9  0.9  0.7  1.0};
c={v1 v2 v3 v4 v5 v6 v7 v8 v9 v10};
r={s1 s2 s3 s4 s5};
create sigma from sigma [colname=c];
append from sigma;
close sigma;
M = EIGVAL(SIGMA);
e={eigen};

create eigen from m [colname=e];
append from m;
close eigen;

*************************************************************;
/*    GENERATE MULTIVARIATE NORMAL DATA (RN matrix) WITH MEAN=MN
AND
      WITH THE TRUE CORRELATION
     STORAGE IT IN  DATA SET MVN
*/
*************************************************************;

S = ROOT(SIGMA);
RN =X*S;
c={v1 v2 v3 v4 v5 v6 v7 v8 v9 v10};
r={s1 s2 s3 s4 s5};
create mvn FROM rn [colname=c];
append from rn; close mvn;

*************************************************************;
/*    GENERATE ZERO MIXED MULTIVARIATE NORMAL DATA - Y1 (LOW p0)
Y2 (hi p0)
      MATRIX*/
/*      STORAGE IT IN  DATA SET MIXl MIXH                          */
*************************************************************;
```

```
Y2 = RN#z2;
c={mv21 mv22 mv23 mv24 mv25 mv26 mv27 mv28 mv29 mv210};
r={s1 s2 s3 s4 s5};
create mix2 from Y2 [colname=c];
append from Y2;
close mix2;
quit;

*************************************************************;
/* PRINCIPAL COMPONENT ANALYSIS ON
MVN (full data X) and MIXH (Observed X)data                    */
*************************************************************;
proc princomp data = mvn outstat =stat0 noprint;
proc princomp data =mix2  outstat = stat2 noprint;
 *************************************************************;
/*   RE-CONSTRUCT MULTIVARIATE NORMAL DATA USING MEAN
SUBSTUTION
      STORAGE IT IN  DATA SET NEWX2
    THEN DO PRINCIPAL COMPONENT ANALYSIS ON NEWX2
*/

*************************************************************;
proc means data= mix2 noprint;
output out = inter mean = m1 m2 m3 m4 m5 m6 m7 m8 m9 m10;
data inter1; set inter;
do i = 1 to 100;
output;
end;
data inter2; merge mix2 inter1;
keep mv21-mv29 mv210 m1-m10;
data newx2; set inter2;
if mv21= 0 then mv21= m1; else mv21 =mv21;
if mv22= 0 then mv22 =m2; else mv22= mv22;
if mv23= 0 then mv23 =m3 ;else mv23=mv23;
if mv24= 0 then mv24 =m4; else mv24=mv24;
if mv25= 0 then mv25 =m5 ; else mv25 =mv25;
if mv26= 0 then mv26 =m6; else mv26=mv26;
if mv27= 0 then mv27 =m7; else mv27 = mv27;
if mv28= 0 then mv28 =m8 ;else mv28=mv28;
if mv29= 0 then mv29 =m9;else mv29=mv29;
if mv210= 0 then mv210 = m10; else mv210=mv210;
drop m1-m10;
proc princomp data = newx2 outstat= statx2 noprint;
```

```
************************************************************;
/*   TREAT ZEROES AS MISSING VALUES
     USE EM_COVAR* (EM ALGORITHM) TO ESTIMATE THE CORRELATION
     THEN DO DECOMPOSITION ON THE CORRELATION (PCA) TO OBTAIN
     EIGENVALUES
     STORAGE DATA IN DATA SET EMC2
*/
************************************************************;
data newms2; set mix2;
if mv21= 0  then mv21=.;  else mv21=mv21;
if mv22= 0  then mv22=.;  else mv22=mv22;
if mv23= 0  then mv23=.;  else mv23=mv23;
if mv24= 0  then mv24=.;  else mv24=mv24;
if mv25= 0  then mv25=.;  else mv25=mv25;
if mv26= 0  then mv26=.;  else mv26=mv26;
if mv27= 0  then mv27=.;  else mv27=mv27;
if mv28= 0  then mv28=.;  else mv28=mv28;
if mv29= 0  then mv29=.;  else mv29=mv29;
if mv210= 0  then mv210=.;  else mv210=mv210;

%em_covar( data     = newms2,
        nomiss   =     ,
        var      = mv21 mv22 mv23 mv24 mv25 mv26 mv27 mv28 mv29 mv210,
        out_file = /usr1/home/grad/swang/emc10hv.out2,
        mat_type = corr,
        max_iter = 200,
        conv     = 0.01,
        df       = n,
        bs       = 1)


data emc2;

infile "/usr1/home/grad/swang/emc10hv.out2";
input x21 x22 x23 x24 x25 x26 x27 x28 x29 x210;

proc iml;
use stat0;
read all into ss0;
close stat0;
gen = ss0[14,];

c={v1 v2 v3 v4 v5 v6 v7 v8 v9 v10};
r={s1 s2 s3 s4 s5};
```

```
create gen FROM gen [colname=c];
append from gen;
close gen;

use stat2;
read all into ss2;
close stat2;
mixh=ss2[14,];
c={mv21 mv22 mv23 mv24 mv25 mv26 mv27 mv28 mv29 mv210};
r={s1 s2 s3 s4 s5};
create mixh    From  mixh   [colname=c];
append from   mixh     ;
close mixh   ;

use statx2;
read all into ssx2;
close statx2;
meanh = ssx2[14,];
c={mnv21 mnv22 mnv23 mnv24 mnv25 mnv26 mnv27 mnv28 mnv29 mnv210};
r={s1 s2 s3 s4 s5};
create meanh    From meanh    [colname=c];
append from meanh       ;
close meanh      ;

use emc2;
read all into aa2;
close emc2;
a2 = aa2[ {1 2 3 4 5 6 7 8 9 10},{1 2 3 4 5 6 7 8 9 10}];
eval2 = eigval(a2);
evec2 = eigvec(a2);
teval2 = eval2`;
tevec2 = evec2`;
c={emv21 emv22 emv23 emv24 emv25 emv26 emv27 emv28 emv29 emv210};
r={s1 s2 s3 s4 s5};
create em2    From  teval2   [colname=c];
append from  teval2      ;
close  em2     ;
quit;
run;
data all; merge gen  mixh  meanh  em2;
run;

data stats2;
set all stats2;
```

```
run;
%end;
run;
%mend;
*************************************************;
*START THE MACRO PCMISS    ;
*GET THE FINAL RESULTS BY PROC MEANS;
*************************************************;
%pcmiss;
proc print data =stats2;
proc means data =stats2;
proc print data = eigen;
run;
```