R Code for Full/Partial MI, and Pooling Parameter Estimates

```
# In this illustration, y1 and y2 represent the two dependent variables (DV)
in the VAR model, x1 and x2 represent the covariates (COV).
# Each of them has T=15 and N=100, with around 30% of missing data.
# We also simulated two fully observed variables: ax1 and ax2.
# Data generating model we used follows the VAR model described in the paper.

#Read in simulated data set (provided in the attached .zip file).
data=read.table("Simulate Data.txt")
n=100
nt=15

# The first step is to define an imputation model, which includes both DVs,
COVs, lagged DVs, lagged COVs if necessary, and all other auxiliary variables.

#Following steps will create a long/tall format data set with lag one DVs and COVs.
#The lagged variables can also be created using R functions of choice, such as lag().
#The number of lags to be included in the imputation model depends on
#the order of the assumed true model.  For example, if the true model is a VAR
#at order one, only lag one variables are necessary in the imputation model.

y1=data[,1:15]
y2=data[,16:30]
x1=data[,31:45]
x2=data[,46:60]
ax1=data[,61:75]
ax2=data[,76:90]

y1.lag1=cbind(rep(NA,n),y1[,c(1:(nt-1))])
y2.lag1=cbind(rep(NA,n),y2[,c(1:(nt-1))])
x1.lag1=cbind(rep(NA,n),x1[,c(1:(nt-1))])
x2.lag1=cbind(rep(NA,n),x2[,c(1:(nt-1))])

y1.temp=reshape(y1, direction="long",varying=list(1:15))
y1.temp=y1.temp[order(y1.temp$id),][,2]

y2.temp=reshape(y2, direction="long",varying=list(1:15))
y2.temp=y2.temp[order(y2.temp$id),][,2]

x1.temp=reshape(x1, direction="long",varying=list(1:15))
x1.temp=x1.temp[order(x1.temp$id),][,2]

x2.temp=reshape(x2, direction="long",varying=list(1:15))
x2.temp=x2.temp[order(x2.temp$id),][,2]
```

```
y1.lag1.temp=reshape(y1.lag1, direction="long",varying=list(1:15))
y1.lag1.temp=y1.lag1.temp[order(y1.lag1.temp$id),][,2]

y2.lag1.temp=reshape(y2.lag1, direction="long",varying=list(1:15))
y2.lag1.temp=y2.lag1.temp[order(y2.lag1.temp$id),][,2]

x1.lag1.temp=reshape(x1.lag1, direction="long",varying=list(1:15))
x1.lag1.temp=x1.lag1.temp[order(x1.lag1.temp$id),][,2]

x2.lag1.temp=reshape(x2.lag1, direction="long",varying=list(1:15))
x2.lag1.temp=x2.lag1.temp[order(x2.lag1.temp$id),][,2]

ax1.temp=reshape(ax1, direction="long",varying=list(1:15))
ax1.temp=ax1.temp[order(ax1.temp$id),][,2]

ax2.temp=reshape(ax2, direction="long",varying=list(1:15))
ax2.temp=ax2.temp[order(ax2.temp$id),][,2]

# Note all categorical variables need to be specified using the as.factor function.
# In this simulated data set, COV x1 is a categorial variable.
MImodel=data.frame(cbind(y1.temp,y2.temp, x1.temp,x2.temp,
                   y1.lag1.temp,y2.lag1.temp,x1.lag1.temp,x2.lag1.temp,
                   ax1.temp,ax2.temp))
MImodel[,3]=as.factor(MImodel[,3]) #specify x1 to be categorical
MImodel[,7]=as.factor(MImodel[,7]) #specify x1.lag1 to be categorical

# The next step is to perform imputation using the specified imputation model.
# Number of imputation can be specified with the argument "m=", which by default
# is 5.
library(mice)
m=5
imp=mice(MImodel,m=m)

# Initial list to store outputs
# Number of parameters estimated in this illustration is 11.
k=11

# Parameter estimates from each imputation will be stored in matrix qhat.
qhat=matrix(NA, nrow=m,ncol=k)

# Variance covariance matrix of parameter estimates from each imputation
will be stored in a list u.
u=array(NA,dim=c(k,k,m))
```

```
# Perform model fitting with the m imputed data sets.
for (i in 1:m) {
        # Retrieve the ith imputed data sets
        data.impute=complete(imp,action=i)

        # Arrange data for model fitting procedures as necessary.
        # With Full MI, imputed data are used for all DVs and COVs.
        # For Partial MI, we keep the missingness in DVs and use imputed data
        for COVs.
        # Extract DVs and COVs from imputed data sets.
        y1.imp=matrix(data.impute[,1],nrow=n,byrow=TRUE)
        y2.imp=matrix(data.impute[,2],nrow=n,byrow=TRUE)
        x1.imp=matrix(data.impute[,3],nrow=n,byrow=TRUE)
        x2.imp=matrix(data.impute[,4],nrow=n,byrow=TRUE)

        # Perform modeling fitting procedures with full or partially imputed
        variables with time series model fitting program of choice.  In this paper,
        mkfm6(Dolan,2002) was used to fit the VAR model.

        # We also provided:
        #  - a R function to write out the data set in mkfm6 format,writemkfm.R;
        #  - a R function to write mkfm6 model script,compileKFscript.R; and
        #  - R functions to call mkfm6 through R (for PC users)
        # To run the functions, please make sure all files are saved in the same
        # directory.

        # create a name for new data set
        datafile=sprintf(paste("data%i",".txt",sep=""),i)
        # create a name for model script
        fileKF =sprintf(paste("mk%i",".txt",sep="",collapse=""),i)
        filebat=sprintf(paste("run%i",".bat",sep="",collapse=""),i)
        ne=2 #number of DVs
        temp=cbind(y1.imp, y2.imp, x1.imp, x2.imp)
        source("writemkfm.R") #function to write a data set in mkfm6 format
        writemkfm(temp,ne,nt,datafile)
        source(paste("compileKFscript.R",sep="")) #function to write mkfm6 model script
        source(paste("compilebat.R",sep=""))
        system(sprintf(paste("run%i",".bat",sep="",colapse=""),i),
               wait = TRUE,intern=TRUE)

        # Store model fitting results in qhat and u.

        pars=scan("pars.out")
        # program generated parameter point estimates.
        qhat[i,]=pars[seq((k*k+1),length(pars),2)]
        # program estimated parameter variance covariance matrix.
        u[, ,i]=matrix(pars[1:(k*k)],ncol=k)
}
# Finally, pool results from m sets of estimations.
# Calculate average parameter point estimates across m sets of model fitting results
qbar <- apply(qhat, 2, mean)

# Calculate pooled standard error estimates
ubar <- apply(u, 1:2, mean)
e <- qhat - matrix(qbar, nrow = m, ncol = k, byrow = TRUE)
b <- (t(e) %*% e)/(m - 1)
vcov <- ubar + (1 + 1/m) * b #vcov is the pooled variance covariance matrix
for parameter estimates
se=sqrt(diag(vcov))
```