Multivariate Multiple Imputation

Utrecht University Winter School: Missing Data in R

Kyle M. Lang

Department of Methodology & Statistics
Utrecht University

2022-02-03

Outline

Joint Modeling vs. Fully Conditional Specification

When imputing with *Joint Modeling* (JM) approaches, the missing data are replaced by samples from the joint posterior predictive distribution.

- To impute X, Y, and Z, we draw:

$$X, Y, Z \sim P(X, Y, Z | \theta)$$

With *Fully Conditional Specification* (FCS), the missing data are replaced with samples from the conditional posterior predictive distribution of each incomplete variable.

- To impute X, Y, and Z, we draw:

$$\begin{aligned} X &\sim P(X | Y, Z, \theta_X) \\ Y &\sim P(Y | X, Z, \theta_Y) \\ Z &\sim P(Z | Y, X, \theta_Z) \end{aligned}$$

Joint Modeling: Strengths

When correctly implemented, JM approaches are guaranteed to produce *Bayesianly proper* imputations.

- A sufficient condition for *properness* is that the imputations are randomly sampled from the correctly specified joint posterior predictive distribution of the missing data.
    - This is the defining characteristic of JM methods.

When using the correct distribution, imputations produced by JM methods will be the best possible imputations.

- Unbiased parameter estimates
- Well-calibrated sampling variability

Joint Modeling: Weaknesses

JM approaches don’t scale well.

- The computational burden increases with the number of incomplete variables.

JM approaches are only applicable when the joint distribution of all incomplete variables follows a known form.

- Mixes of continuous and categorical variables are difficult to accommodate.

Software Implementations

In R, MI via JM is available from several packages.

- **Amelia** (Honaker, King, & Blackwell, 2011)
  - Bootstrapped EM algorithm
- **norm** (Schafer, 2013)
  - Classic data augmentation.
- **mice** (Van Buuren & Groothuis-Oudshoorn, 2011)
  - Data augmentation for block updating.

JM imputation is also available in SAS, Stata, SPSS, and Mplus.

Fully Conditional Specification: Strengths

FCS scales much better than JM.

- FCS only samples from a series of univariate distributions, not large joint distributions.

FCS approaches can create imputations for variables that don't have a sensible joint distribution.

- FCS can easily treat mixes of continuous and categorical variables.

Fully Conditional Specification: Weaknesses

FCS will usually be slower than JM.

- Each variable gets its own fully parameterized distribution, even if that granularity is unnecessary.

When the incomplete variables don't have a known joint distribution, FCS doesn't have theoretical support.

- There is, however, a large degree of empirical support for the tenability of the FCS approach.
- In practice, we usually choose FCS since real data rarely arise from a known joint distribution.

Software Implementations

The **mice** package is the most popular R implementation of FCS.
(Van Buuren & Groothuis-Oudshoorn, 2011).

- Mature implementation
- Well integrated into the larger R ecosystem
- Very active development

The **mi** package (Su, Yajima, Gelman, & Hill, 2011) offers another option.

- More focus on diagnostics
- Object oriented flavor
- Not very actively developed

FCS imputation is also available in SAS, SPSS, Stata, and Mplus.

Aside: Gibbs Sampling

Up to this point, most of the models we've explored could be approximated by sampling directly from their posterior distributions.

- This won't be true with arbitrary, multivariate missing data.

To make inference regarding a multivariate distribution with multiple, interrelated, unknown parameters, we can use *Gibbs sampling*.

- Sample from the conditional distribution of each parameter, conditioning on the current best guesses of all other parameters.

Aside: Gibbs Sampling

Suppose the following:

1. I want to make some inference about the tri-variate mean of $X, Y, Z = \mu_X, \mu_Y, \mu_Z \sim P(\mu|\theta)$
2. $P(\mu|\theta)$ is super hairy and difficult to sample
3. I can easily sample from the conditional distributions: $P(\mu_X|\hat{\mu}_Y, \hat{\mu}_Z, \theta)$, $P(\mu_Y|\hat{\mu}_X, \hat{\mu}_Z, \theta)$, and $P(\mu_Z|\hat{\mu}_X, \hat{\mu}_Y, \theta)$.

Then, I can approximate the full joint distribution $P(\mu|\theta)$ by sequentially sampling from $P(\mu_X|\hat{\mu}_Y, \hat{\mu}_Z, \theta)$, $P(\mu_Y|\hat{\mu}_X, \hat{\mu}_Z, \theta)$, and $P(\mu_Z|\hat{\mu}_X, \hat{\mu}_Y, \theta)$.

Aside: Gibbs Sampling

Starting with initial guesses of μ_Y, $\hat{\mu}_Y^{(0)}$, and μ_Z, $\hat{\mu}_Z^{(0)}$, and assuming θ is known, Gibbs sampling proceeds as follows:

$$\begin{aligned}
\hat{\mu}_X^{(1)} &\sim P(\mu_X | \hat{\mu}_Y^{(0)}, \hat{\mu}_Z^{(0)}, \theta) \\
\hat{\mu}_Y^{(1)} &\sim P(\mu_Y | \hat{\mu}_X^{(1)}, \hat{\mu}_Z^{(0)}, \theta) \\
\hat{\mu}_Z^{(1)} &\sim P(\mu_Z | \hat{\mu}_Y^{(1)}, \hat{\mu}_X^{(1)}, \theta)
\end{aligned}$$

$$\begin{aligned}
\hat{\mu}_X^{(2)} &\sim P(\mu_X | \hat{\mu}_Y^{(1)}, \hat{\mu}_Z^{(1)}, \theta) \\
\hat{\mu}_Y^{(2)} &\sim P(\mu_Y | \hat{\mu}_X^{(2)}, \hat{\mu}_Z^{(1)}, \theta) \\
\hat{\mu}_Z^{(2)} &\sim P(\mu_Z | \hat{\mu}_Y^{(2)}, \hat{\mu}_X^{(2)}, \theta) \\
&\vdots
\end{aligned}$$

Why do we care?

Multivariate MI employs the same logic as Gibbs sampling.

- The imputations are created by conditioning on the current estimates of the imputation model parameters.
- The imputation model parameters are updated by conditioning on the most recent imputations.
- With FCS, each variable is imputed by conditioning on the most recent imputations of all other variables.

Fully Conditional Specification

Procedure: Fully Conditional Specification

1. Fill the missing data with reasonable guesses.
2. For each incomplete variable, do a single iteration of univariate Bayesian MI (e.g., as seen in the last set of slides).
    - After each variable on the data set is so treated, we've completed one iteration.
3. Repeat Step 2 many times.
4. After the imputation model parameters stabilize, save M imputed data sets.

Example: Data Generation

First we'll simulate some synthetic data.

```
## Simulate some data:
simData <-
    simCovData(nObs = 1000, sigma = 0.25, nVars = 4)

head(simData, 10)

            x1         x2          x3          x4
1   0.06313632 -1.4057704 -0.01709217  1.47929405
2  -1.31592547  1.2970920 -0.83500777 -0.44528158
3  -0.30997023  0.9782580  0.02731853  0.35507390
4   0.06927787  0.1836032  0.68794409  0.08049987
5  -0.99354894 -0.3038956  0.80918329 -1.72143555
6   0.36828016 -0.9423245  1.05155348 -0.11078496
7   1.33333163  2.3089780  1.47203000  0.85877495
8  -0.02759718  0.1714383  0.18927909  0.28627771
9   2.37929433  2.5080935  2.18344726  1.56980951
10  0.31841502  0.7886025  0.73658136  0.39445970
```

Example: Data Generation

Next, we impose some missing values on the simulated data.

```r
targets  <- paste0("x", 1:3)
missData <- imposeMissData(data    = simData,
                           targets = targets,
                           preds   = "x4",
                           pm      = 0.3,
                           types = c("low", "center", "high")
                           )
```

Example: Data Visualization

Check the results.

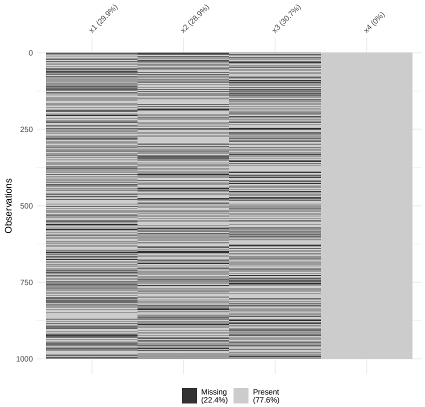
```
head(missData, 10)

            x1        x2          x3          x4
1   0.06313632        NA -0.01709217  1.47929405
2           NA 1.2970920          NA -0.44528158
3  -0.30997023        NA  0.02731853  0.35507390
4           NA        NA  0.68794409  0.08049987
5           NA        NA  0.80918329 -1.72143555
6   0.36828016        NA          NA -0.11078496
7   1.33333163 2.3089780          NA  0.85877495
8           NA        NA  0.18927909  0.28627771
9   2.37929433 2.5080935          NA  1.56980951
10  0.31841502 0.7886025  0.73658136  0.39445970
```

Example: Data Visualization

Use the `naniar::vis_missing()` function to visualize the pattern of missing values.

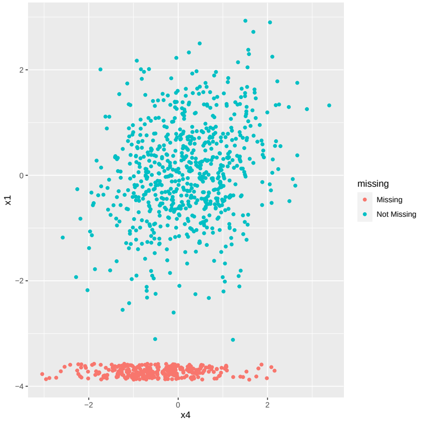
```r
vis_miss(missData)
```



Example: Data Visualization

Use `naniar::geom_miss_point()` to visualize the response pattern.

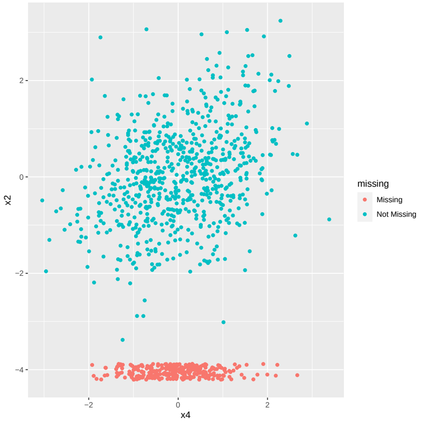
```
ggplot(missData, aes(x4, x1)) +
    geom_miss_point()
```



Example: Data Visualization

Use `naniar::geom_miss_point()` to visualize the response pattern.

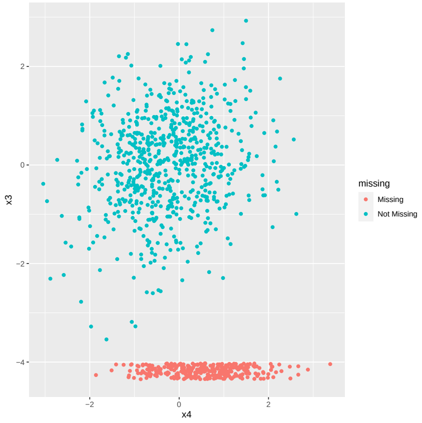
```
ggplot(missData, aes(x4, x2)) +
    geom_miss_point()
```



Example: Data Visualization

Use `naniar::geom_miss_point()` to visualize the response pattern.

```
ggplot(missData, aes(x4, x3)) +
    geom_miss_point()
```



Example: Fully Conditional Specification

```r
## Define iteration numbers:
nImps <- 100
nBurn <- 500
nSams <- nBurn + nImps

## Summarize missingness:
rMat <- !is.na(missData)
nObs <- colSums(rMat)
nMis <- colSums(!rMat)

## Fill the missingness with initial (bad) guesses:
mean0  <- colMeans(missData, na.rm = TRUE)
sigma0 <- cov(missData, use = "pairwise")
draws0 <- rmvnorm(nrow(missData), mean0, sigma0)

impData        <- missData
impData[!rMat] <- draws0[!rMat]
```

Example: Fully Conditional Specification

Define an elementary imputation function:

```r
eif <- function(data, rVec, v) {
    ## Get the expected model parameters:
    fit  <- lm(paste(v, "~ ."), data = data[rVec, ])
    beta <- coef(fit)
    s2   <- summary(fit)$sigma^2

    ## Partition data:
    vars <- setdiff(colnames(data), v)
    data <- as.matrix(data)
    xObs <- cbind(1, data[rVec, vars])
    xMis <- cbind(1, data[!rVec, vars])

    ## Sample sigma:
    sigmaSam <- rinvchisq(1, df = fit$df, scale = s2)

    ## Sample beta:
    betaVar <- sigmaSam * solve(crossprod(xObs))
    betaSam <- rmvnorm(1, mean = beta, sigma = betaVar)

    ## Return a randomly sampled imputation:
    xMis %*% t(betaSam) + rnorm(nrow(xMis), 0, sqrt(sigmaSam))
}
```

Example: Fully Conditional Specification

Apply the elementary imputation function to each incomplete variable:

```r
## Iterate through the FCS algorithm:
impList <- list()
for(s in 1 : nSams) {
    for(v in targets) {
        rVec               <- rMat[ , v]
        impData[!rVec, v] <- eif(impData, rVec, v)
    }

    ## If the chains are burnt-in, save imputed datasets:
    if(s > nBurn) impList[[s - nBurn]] <- impData
}
```

Example: Fully Conditional Specification

Analyze the multiply imputed datasets:

```r
## First, our manual version:
fits1 <- lapply(impList,
                function(x) lm(x1 ~ x2 + x3, data = x)
                )
pool1 <- MIcombine(fits1)

## Do the same analysis with mice::mice():
miceOut <- mice(data      = missData,
                m         = 100,
                method    = "norm",
                printFlag = FALSE)
fits2 <- with(miceOut, lm(x1 ~ x2 + x3))
pool2 <- pool(fits2)
```

Example: Fully Conditional Specification

Compare approaches:

	Est	SE	t	p	FMI
(Intercept)	-0.009	0.038	-0.238	1.188	0.406
x2	0.225	0.041	5.519	<0.001	0.483
x3	0.161	0.049	3.283	0.001	0.637

Manual Version

	Est	SE	t	p	FMI
(Intercept)	-0.012	0.036	-0.342	1.267	0.351
x2	0.225	0.04	5.647	<0.001	0.463
x3	0.167	0.045	3.681	<0.001	0.58

MICE Version

Joint Modeling

Aside: Definition of Regression Parameters

So far, we've been using the least-squares estimates of α, β, and σ^2 to parameterize our posterior distributions.

- We can also define the parameters in terms of sufficient statistics.

Given μ and Σ, we can define all of our regression moments as:

$$\begin{aligned}
\beta &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} \\
&= \text{Cov}(\mathbf{X})^{-1}\text{Cov}(\mathbf{X}, \mathbf{Y}) \\
\alpha &= \mu_Y - \beta^T\mu_X \\
\Sigma_\varepsilon &= \Sigma_Y - \beta^T\Sigma_X\beta
\end{aligned}$$

These definitions are crucial for JM approaches.

- Within the subset of data define by a given response pattern, the outcome variables will be entirely missing.

Multivariate Bayesian Regression

Previously, we saw examples of univariate Bayesian regression which used the following model:

$$\beta \sim \text{MVN}\left(\hat{\beta}_{ls},\ \sigma^2(\mathbf{X}^T\mathbf{X})^{-1}\right)$$
$$\sigma^2 \sim \text{Inv-}\chi^2\,(N-P, MSE)$$

We can directly extend the above to the multivariate case:

$$\Sigma^{(i)} \sim \text{Inv-W}\left(N-1, (N-1)\Sigma^{(i-1)}\right)$$
$$\mu^{(i)} \sim \text{MVN}\left(\mu^{(i-1)}, N^{-1}\Sigma^{(i)}\right)$$

We get α, β, and Σ_ε via the calculations on the preceding slide

Procedure: Joint Modeling

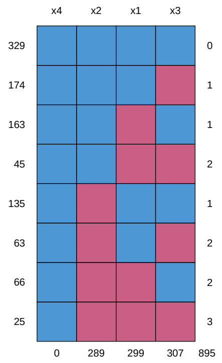
In JM imputation, we estimate the imputation model via the Tanner and Wong (1987) *data augmentation* algorithm.

1. Partition the incomplete data by response pattern.
   - Produce S subsets wherein each row shares the same response pattern.
2. Provide initial guesses for μ and Σ.
3. Within each subset, use the current guesses of μ and Σ to generate imputations via multivariate Bayesian regression.
4. Use the filled-in data matrix to updated the sufficient statistics.
5. Repeat Steps 3 and 4 many times.
6. After the imputation model parameters have stabilized, save M imputed data sets produced in Step 3.

Example: Data Visualization

Use `mice::md.pattern()` to visualize the response patterns.

```r
pats <- md.pattern(missData)
```



Example: Joint Modeling

```r
iStep <- function(data, pats, ind, pars) {
    ## Loop over non-trivial response patterns:
    for(i in 1:nrow(pats)) {
        ## Define the current response pattern:
        p1 <- pats[i, ]

        ## Replace missing data with imputations:
        data[ind == i, !p1] <- getImps(data  = data[ind == i, ],
                                       mu    = pars$mu,
                                       sigma = pars$sigma,
                                       p1    = p1)
    }

    ## Return the imputed data:
    data
}
```

Example: Joint Modeling

```r
getImps <- function(data, mu, sigma, p1) {
    ## Partition the parameter matrices:
    mY  <- matrix(mu[!p1])
    mX  <- matrix(mu[p1])
    sY  <- sigma[!p1, !p1]
    sX  <- sigma[p1, p1]
    cXY <- sigma[p1, !p1]

    ## Compute the imputation model parameters:
    beta  <- solve(sX) %*% cXY
    alpha <- mY - t(beta) %*% mX
    sE    <- sY - t(beta) %*% sX %*% beta

    ## Pull out predictors:
    X <- as.matrix(data[ , p1])

    ## Generate and return the imputations:
    n <- nrow(X)
    matrix(1, n) %*% t(alpha) + X %*% beta + rmvnorm(n, sigma = sE)
}
```

Example: Joint Modeling

```r
pStep <- function(data) {
    ## Update the complete-data sufficient statistics:
    n <- nrow(data)
    m <- colMeans(data)
    s <- (n - 1) * cov(data)

    ## Sample the covariance matrix and mean vector:
    sigma <- riwish((n - 1), s)
    mu    <- rmvnorm(1, m, (sigma / n))

    ## Return the updated parameters:
    list(mu = mu, sigma = sigma)
}
```

Example: Joint Modeling

Now that we've defined the necessary functions, do the imputation:

```r
## Some preliminaries:
impData <- missData
nIter   <- 50
nImps   <- 100

## Summarize response patterns:
rMat <- !is.na(impData)
pats <- uniquecombs(rMat, ordered = TRUE)
ind  <- attr(pats, "index")

## Exclude the fully observed pattern:
if(tail(pats, 1) %>% all())
    pats <- pats[-nrow(pats), ]

## Get starting values for the parameters:
theta0 <- list(mu    = colMeans(impData, na.rm = TRUE),
               sigma = cov(impData, use = "pairwise")
               )
```

Example: Joint Modeling

```r
## Iterate over I- and P-Steps to generate imputations:
impList1 <- list()
for(m in 1:nImps) {
    ## Initialize the parameter vector:
    theta1 <- theta0
    for(rp in 1:nIter) {
        ## Do one I-Step:
        impData <- iStep(data = impData,
                         pats = pats,
                         ind  = ind,
                         pars = theta1)

        ## Do one P-Step:
        theta1 <- pStep(impData)
    }
    ## Save the final imputed dataset:
    impList1[[m]] <- impData
}
```

Example: Joint Modeling

Do the same type of imputation with **norm**:

```r
missData <- as.matrix(missData)

## Pre-process the data and get starting values via EM:
meta   <- prelim.norm(missData)
theta0 <- em.norm(meta, showits = FALSE)

rngseed(235711)

impList2 <- list()
for(m in 1 : nImps) {
    ## Estimate the imputation model via data augmentation:
    theta1 <- da.norm(s = meta, start = theta0, steps = nIter)

    ## Impute missing values via a final I-Step of DA:
    impList2[[m]] <- imp.norm(s = meta, theta = theta1, x = missData)
}
```

Example: Joint Modeling

Analyze the multiply imputed data:

```
## Manual implementation:
fits1 <- lapply(impList1,
                function(x) lm(x1 ~ x2 + x3, data = x)
                )
pool1 <- MIcombine(fits1)

## Imputation using norm:
fits2 <- lapply(impList2,
                function(x) lm(x1 ~ x2 + x3,
                               data = as.data.frame(x)
                               )
                )
pool2 <- MIcombine(fits2)
```

Example: Joint Modeling

Compare approaches:

	Est	SE	t	p	FMI
(Intercept)	-0.01	0.036	-0.276	1.217	0.355
x2	0.229	0.042	5.464	<0.001	0.506
x3	0.17	0.044	3.878	<0.001	0.542

Manual Version

	Est	SE	t	p	FMI
(Intercept)	-0.012	0.036	-0.337	1.264	0.359
x2	0.231	0.041	5.652	<0.001	0.482
x3	0.162	0.046	3.504	0.001	0.591

NORM Version

Mixed Data Types

FCS for Mixed Data Types

FCS imputation can easily accommodate incomplete data that contain both continuous and categorical/non-normal variables.

- Replace the normal-theory elementary imputation model described above with an appropriate model for the distribution of each incomplete variable
    - Logistic regression (various flavors)
    - Donor-based methods
    - Tree-based methods

The FCS framework can essentially accommodate any data for which you can define an appropriate supervised model.

- Many useful methods are already implemented in the **mice** package.

JM for Mixed Data Types

When applying JM to incomplete data with mixed variable types, we have to general options.

1. Impute under the multivariate normal model and round, coarsen, or truncate the continuous imputations to “match” the original data.
    - This was the old-school recommendation from the days before FCS (e.g., Allison, 2002; Schafer, 1997).
    - The **Amelia** package implements this approach.
    - This approach tends to perform poorly in methodological evaluations (e.g., Lang & Wu, 2017; Wu, Jia, & Enders, 2015).
2. Impute under an appropriate joint model for the data.
    - This approach is only available when a suitable joint model exists.
    - The **mix** package (Schafer, 2017) implements this approach for the general location model (Little & Schluchter, 1985).
    - This approach also doesn’t do very well, in practice (Lang & Wu, 2017).

References

Allison, P. D. (2002). *Missing data*. Thousand Oaks, CA: Sage Publictions.

Honaker, J., King, G., & Blackwell, M. (2011). Amelia II: A program for missing data. *Journal of Statistical Software*, *45*(7), 1–47. Retrieved from `https://www.jstatsoft.org/v45/i07/`

Lang, K. M., & Wu, W. (2017). A comparison of methods for creating multiple imputations of nominal variables. *Multivariate Behavioral Research*, *52*(3), 290-304. doi: http://doi.org/10.1080/00273171.2017.1289360

Little, R. J. A., & Schluchter, M. D. (1985). Maximum likelihood estimation for mixed continuous and categorical variables with missing values. *Biometrika*, *72*, 497–512.

Schafer, J. L. (1997). *Analysis of incomplete multivariate data* (Vol. 72). Boca Raton, FL: Chapman & Hall/CRC.

References

Schafer, J. L. (2013). norm: Analysis of multivariate normal datasets with missing values [Computer software manual]. Retrieved from `https://CRAN.R-project.org/package=norm` (R package version 1.0-9.5, Ported to R by Alvaro A. Novo)

Schafer, J. L. (2017). mix: Estimation/multiple imputation for mixed categorical and continuous data [Computer software manual]. Retrieved from `https://CRAN.R-project.org/package=mix` (R package version 1.0-10)

Su, Y.-S., Yajima, M., Gelman, A. E., & Hill, J. (2011). Multiple imputation with diagnostics (mi) in r: opening windows into the black box. *Journal of Statistical Software*, *45*(2), 1–31.

Tanner, M. A., & Wong, W. H. (1987). The calculation of posterior distributions by data augmentation. *Journal of the American Statistical Association*, *82*(398), 528–540.

References

Van Buuren, S., & Groothuis-Oudshoorn, K. (2011). mice: Multivariate imputation by chained equations in R. *Journal of Statistical Software*, *45*(3), 1–67.

Wu, W., Jia, F., & Enders, C. (2015). A comparison of imputation strategies for ordinal missing data on likert scale variables. *Multivariate Behavioral Research*, *50*(5), 484-503. doi: http://doi.org/10.1080/00273171.2015.1022644