Univariate Multiple Imputation

Stats Camp 2018: Missing Data Analysis

Kyle M. Lang

Department of Methodology & Statistics
Tilburg University

19–21 October 2018

Outline

- Build up the basis for MI from linear regression.
- Demonstrate each step with examples in R.
- Show how to manually implement a simple MI in R.

Brief Regression Refresher

Ordinary least squares (OLS) regression estimates the following model:

$$Y = \mathbf{X}\beta + \varepsilon$$

By minimizing the residual sum of squared errors, we get the following estimated regression coefficients:

$$\hat{\beta} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T Y$$

We can predict the values of unobserved outcome data by applying the fitted βs to new predictor data:

$$\hat{Y} = \mathbf{X}_{new}\hat{\beta}$$

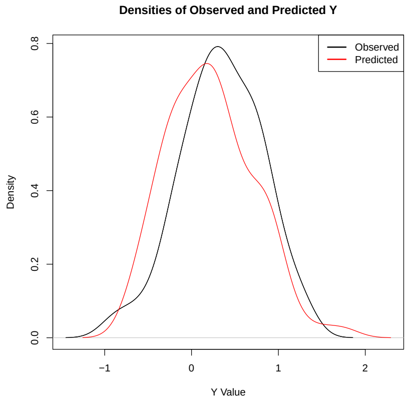
These predicted values are the basis for nearly all imputation methods.

OLS Example

```
## Create some data:
X    <- cbind(1, rnorm(100))
beta <- matrix(c(0.25, 0.5))
y    <- X %*% beta + rnorm(100, 0.0, 0.1)

## R's built-in solution:
rFit <- lm(y ~ X - 1)
coef(rFit) # R's fitted coefficients

##        X1        X2
## 0.2515459 0.4961789

## Least squares by hand:
betaHat <- solve(t(X) %*% X) %*% t(X) %*% y
t(betaHat) # Our hand-fitted coefficients

##           [,1]      [,2]
## [1,] 0.2515459 0.4961789
```

OLS Example

```r
## What about prediction?
X2   <- cbind(1, rnorm(100))
yHat <- X2 %*% betaHat
```



Prediction

Train a model to predict BMI from diet-related and exercise-related features.

- Plug-in new feature values corresponding to an experimental diet program to see the expected BMI for a hypothetical patient treated with the new program.

Predict future gasoline prices based on geo-political events in oil-producing countries.

- If conflict escalates in the Middle East, adjust the appropriate features and project likely changes in gasoline prices.

Prediction Example

To fix ideas, let's consider the *diabetes* data and the following model:

$$Y_{LDL} = \beta_0 + \beta_1 X_{BP} + \beta_2 X_{gluc} + \beta_3 X_{BMI} + \varepsilon$$

Training this model on the first $N = 400$ patients' data produces the following fitted model:

$$Y_{LDL} = 22.135 + 0.089 X_{BP} + 0.498 X_{gluc} + 1.48 X_{BMI}$$

Prediction Example

To fix ideas, let's consider the *diabetes* data and the following model:

$$Y_{LDL} = \beta_0 + \beta_1 X_{BP} + \beta_2 X_{gluc} + \beta_3 X_{BMI} + \varepsilon$$

Training this model on the first $N = 400$ patients' data produces the following fitted model:

$$Y_{LDL} = 22.135 + 0.089 X_{BP} + 0.498 X_{gluc} + 1.48 X_{BMI}$$

Suppose a new patient presents with $BP = 121$, $gluc = 89$, and $BMI = 30.6$. We can predict their LDL score by:

$$\begin{aligned} \hat{Y}_{LDL} &= 22.135 + 0.089(121) + 0.498(89) + 1.48(30.6) \\ &= 122.463 \end{aligned}$$

Missing Data Imputation

Levels of Uncertainty Modeling

Van Buuren (2012) provides a very useful classification of different imputation methods:

1. Simple Prediction
   - The missing data are naively filled with predicted values from some regression equation.
   - All uncertainty is ignored.
2. Prediction + Noise
   - A random residual error is added to each predicted value to create the imputations.
   - Only uncertainty in the predicted values is modeled.
   - The imputation model itself is assumed to be correct and error-free.
3. Prediction + Noise + Model Error
   - Uncertainty in the imputation model itself is also modeled.
   - Only way to get fully proper imputations in the sense of Rubin (1987).

Do we really need to worry?

The arguments against single imputation can seem archaic and petty. Do we really need to worry about this stuff?

Do we really need to worry?

The arguments against single imputation can seem archaic and petty. Do we really need to worry about this stuff?

- YES!!! (At least if you care about inference)

The following are results from a simple Monte Carlo simulation:

	Complete Data	Conditional Mean	Stochastic	MI
cor(X, Y)	0.500	0.563	0.498	0.497
Type I Error	0.052	0.138	0.120	0.054

Table: Mean Correlation Coefficients and Type I Error Rates

Do we really need to worry?

The arguments against single imputation can seem archaic and petty. Do we really need to worry about this stuff?

- YES!!! (At least if you care about inference)

The following are results from a simple Monte Carlo simulation:

	Complete Data	Conditional Mean	Stochastic	MI
cor(X, Y)	0.500	0.563	0.498	0.497
Type I Error	0.052	0.138	0.120	0.054

Table: Mean Correlation Coefficients and Type I Error Rates

- Conditional mean substitution overestimates the correlation effect.
- Both single imputation methods inflate Type I error rates.
- MI provides unbiased point estimates and accurate Type I error rates.

Simulate Some Toy Data

```r
nObs <- 1000 # Sample Size
pm   <- 0.3 # Proportion Missing

sigma <- matrix(c(1.0, 0.5, 0.0,
                  0.5, 1.0, 0.3,
                  0.0, 0.3, 1.0),
                ncol = 3)

simData <- as.data.frame(rmvnorm(nObs, c(0, 0, 0), sigma))
colnames(simData) <- c("y", "x", "z")

## Impose MAR Nonresponse:
misData <- simData
rVec    <- pnorm(misData$x,
                 mean = mean(misData$x),
                 sd   = sd(misData$x)) < pm
misData[rVec, "y"] <- NA

## Subset the data:
yMis <- misData[rVec, ]; yObs <- misData[!rVec, ]
```

Look at the incomplete data.

```
head(misData, n = 5)

##            y            x          z
## 1         NA -0.625895673 -1.2420694
## 2 -0.1448488 -0.001578954  0.4701091
## 3  1.2017766  1.069733846 -0.7550419
## 4  1.0424014 -0.192959605 -1.4458352
## 5  1.5970164 -0.249389277 -0.7206223
```

Expected Imputation Model Parameters

```
## Get the imputation model moments:
lsFit <- lm(y ~ x + z, data = yObs)

beta  <- coef(lsFit)
sigma <- summary(lsFit)$sigma

beta

## (Intercept)           x           z
##  0.03510861  0.55476584 -0.15346474

sigma

## [1] 0.8595465
```

Conditional Mean Substitution

```
## Get deterministic imputations:
imp1 <- beta[1] + beta[2] * yMis[ , "x"] +
    beta[3] * yMis[ , "z"]

## Fill missing cells in Y:
impData1               <- misData
impData1[rVec, "y"] <- imp1

head(impData1, n = 5)

##            y            x          z
## 1 -0.1215031 -0.625895673 -1.2420694
## 2 -0.1448488 -0.001578954  0.4701091
## 3  1.2017766  1.069733846 -0.7550419
## 4  1.0424014 -0.192959605 -1.4458352
## 5  1.5970164 -0.249389277 -0.7206223
```

Stochastic Regression Imputation

```
## Get stochastic imputations:
imp2 <- beta[1] + beta[2] * yMis[ , "x"] +
    beta[3] * yMis[ , "z"] + rnorm(nrow(yMis), 0, sigma)

## Fill missing cells in Y:
impData2               <- misData
impData2[rVec, "y"] <- imp2

head(impData2, n = 5)

##            y            x          z
## 1  1.0827485 -0.625895673 -1.2420694
## 2 -0.1448488 -0.001578954  0.4701091
## 3  1.2017766  1.069733846 -0.7550419
## 4  1.0424014 -0.192959605 -1.4458352
## 5  1.5970164 -0.249389277 -0.7206223
```

Flavors of MI

MI simply repeats a single regression imputation M times.

- The specifics of the underlying regression imputation are important.

Flavors of MI

MI simply repeats a single regression imputation *M* times.

- The specifics of the underlying regression imputation are important.

Simply repeating the stochastic regression imputation procedure described above won't suffice.

- Still produces too many Type I errors

	Complete Data	PN-Type	PNE-Type
cor(X, Y)	0.499	0.499	0.498
Type I Error	0.040	0.066	0.046

Table: Mean Correlation Coefficients and Type I Error Rates

- Type I error rates for PN-Type MI are much better than they were for single stochastic regression imputation, but they're still too high.

Proper MI

The problems on the previous slide arise from using the same regression coefficients to create each of the M imputations.

- Implies that you're using the "correct" coefficients.
- This assumption is plainly ridiculous.
    - If we don't know some values of our outcome variable, how can we know the "correct" coefficients to link the incomplete outcome to the observed predictors?

Proper MI

The problems on the previous slide arise from using the same regression coefficients to create each of the M imputations.

- Implies that you're using the "correct" coefficients.
- This assumption is plainly ridiculous.
    - If we don't know some values of our outcome variable, how can we know the "correct" coefficients to link the incomplete outcome to the observed predictors?
- Proper MI also models uncertainty in the regression coefficients used to create the imputations.
    - A different set of of coefficients is randomly sampled (using Bayesian simulation) to create each of the M imputations.
    - The tricky part about implemented MI is deriving the distributions from which to sample these coefficients.

Setting Up Proper MI

Our imputation model is simply a linear regression model:

$$Y = \mathbf{X}\beta + \varepsilon$$

To fully account for model uncertainty, we need to randomly sample both β and $\text{var}(\varepsilon) = \sigma^2$.

- QUESTION: Why do we only sample σ^2 and not ε?

Setting Up Proper MI

Our imputation model is simply a linear regression model:

$$Y = \mathbf{X}\beta + \varepsilon$$

To fully account for model uncertainty, we need to randomly sample both β and $\text{var}(\varepsilon) = \sigma^2$.

- QUESTION: Why do we only sample σ^2 and not ε?

For a simple imputation model with a normally distributed outcome and uninformative priors, we need to specify two distributions:

1. The marginal posterior distribution of σ^2
2. The conditional posterior distribution of β

Marginal Distribution of σ^2

We first specify the marginal posterior distribution for the noise variance, σ^2.

- This distribution does not depend on any other parameters.

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

- σ^2 follows a scaled inverse χ^2 distribution.

Conditional Distribution of β

We then specify the conditional posterior distribution for β.

- This distribution is conditioned on a specific value of σ^2.

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

- β (conditionally) follows a multivariate normal distribution.

PPD of the Missing Data

Once we've sampled our imputation model parameters, we can construct the posterior predictive distribution of the missing data.

- This is the distribution from which we sample our imputed values.
- In practice, we directly compute the imputations based on the simulated imputation model parameters.

$$Y_{imp} = \mathbf{X}_{mis}\tilde{\beta} + \tilde{\varepsilon} \tag{3}$$
$$\text{with } \varepsilon \sim \mathrm{N}\left(0, \widetilde{\sigma^2}\right)$$

General Steps for Basic MI

With all of the elements in place, we can execute a basic MI by following these steps:

1. Find the least squares estimates of β, $\hat{\beta}_{ls}$, by regressing the observed portion of Y onto the the analogous rows of $\mathbf{X}$.
2. Use $\hat{\beta}_{ls}$ to parameterize the posterior distribution of σ^2, given by Equation 1, and draw M samples of σ^2 from this distribution.
3. For each of the σ^2_m, sample a corresponding value of β from Equation 2.
4. Plug the M samples of β and σ^2 into Equation 3 to create the M imputations.

Manual MI Example

First, we need to sample from the marginal posterior distribution of σ^2.

```r
## Define iteration numbers:
nImps <- 100
nSams <- 5000

## Get the expected betas:
fit0  <- lm(y ~ ., data = yObs)
beta0 <- coef(fit0)

## Sample sigma:
sigScale  <- (1 / fit0$df) * crossprod(resid(fit0))
sigmaSams <-
    rinvchisq(nSams, df = fit0$df, scale = sigScale)
```

Manual MI Example

Then we need to use those samples of σ^2 to parameterize the conditional posterior distribution of β and sample from it.

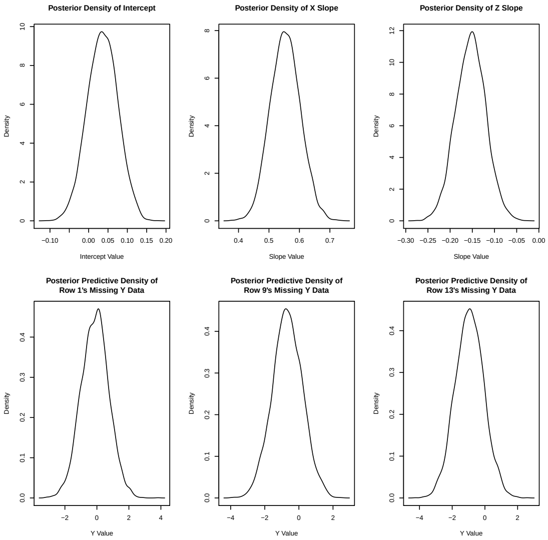
```r
## Partition the predictor matrix:
misX <- as.matrix(cbind(1, yMis[ , c("x", "z")]))
obsX <- as.matrix(cbind(1, yObs[ , c("x", "z")]))

## Sample beta:
betaSams <- matrix(NA, nSams, ncol(obsX))
for(i in 1 : nSams) {
    betaVar       <- sigmaSams[i] * solve(crossprod(obsX))
    betaSams[i, ] <-
        rmvnorm(1, mean = beta0, sigma = betaVar)
}
```

Manual MI Example

Finally, we use the sampled imputation model moments to construct the missing data's posterior predictive distribution:

```r
nMis   <- sum(rVec)
impMat <- matrix(NA, nMis, nSams)
for(i in 1 : nSams) {
    impMat[ , i] <- misX %*% matrix(betaSams[i, ]) +
        rnorm(nMis, 0, sqrt(sigmaSams[i]))
}

## Fill the missing cells with the M imputations:
impList <- list()
ind     <- sample(1 : nSams)
for(m in 1 : nImps) {
    impList[[m]]                <- misData
    impList[[m]][rVec, "y"] <- impMat[ , ind[m]]
}
```

What do we get?



Doing MI-Based Analysis

An MI-based data analysis consists of three phases:

1. The imputation phase
    - Replace missing values with M plausible estimates.
    - Produce M completed datasets.
2. The analysis phase
    - Estimate M replicates of your analysis model.
    - Fit the same model to each of the M datasets from Step 1.
3. The pooling phase
    - Combine the M sets of parameter estimates and standard errors from Step 2 into a single set of MI estimates.
    - Use these pooled parameter estimates and standard errors for inference.

Pooling MI Estimates

Rubin (1987) formulated a simple set of pooling rules for MI estimates.

- The MI point estimate of some interesting quantity, Q^*, is simply the mean of the M estimates, $\{\hat{Q}_m\}$:

$$Q^* = \frac{1}{M} \sum_{m=1}^{M} \hat{Q}_m$$

Pooling MI Estimates

The MI variability estimate, T, is a slightly more complex entity.

- A weighted sum of the *within-imputation* variance, W, and the *between-imputation* variance, B.

$$
\begin{aligned}
W &= \frac{1}{M} \sum_{m=1}^{M} \widehat{SE}_{Q,m}^{2} \\
B &= \frac{1}{M-1} \sum_{m=1}^{M} \left(\hat{Q}_m - Q^* \right)^2 \\
T &= W + \left(1 + M^{-1} \right) B \\
&= W + B + \frac{B}{M}
\end{aligned}
$$

Inference with MI Estimates

After computing Q^* and T, we combine them in the usual way to get test statistics and confidence intervals.

$$t = \frac{Q^* - Q_0}{\sqrt{T}}$$

$$CI = Q^* \pm t_{crit}\sqrt{T}$$

We must take care with our *df*, though.

$$df = (M-1)\left[1 + \frac{W}{(1+M^{-1})\,B}\right]^2$$

Fraction of Missing Information

In Lecture 4, we briefly discussed a very desirable measure of nonresponse: *fraction of missing information* (FMI).

$$FMI = \frac{r + \frac{2}{(df+3)}}{r+1} \approx \frac{(1+M^{-1})B}{(1+M^{-1})B + W} \rightarrow \frac{B}{B+W}$$

where

$$r = \frac{(1+M^{-1})B}{W}$$

The FMI gives us a sense of how much the missing data (and their treatment) have influence our parameter estimates.

- We should report the FMI for an estimated parameter along with other ancillary statistics (e.g., t-tests, p-values, effect sizes, etc.).

Example: Analysis & Pooling

Analyze the multiply imputed datasets and pool results:

```
## Use each dataset to estimate the analysis model:
fits1 <- lapply(impList,
                function(dat) lm(z ~ x + y, data = dat)
                )

## Pool the results:
pool1 <- MIcombine(fits1)
summary(pool1, digits = 3)

## Multiple imputation results:
##       MIcombine.default(fits1)
##                results     se  (lower  upper) missInfo
## (Intercept)  0.000765 0.0304 -0.0587  0.0603      3 %
## x            0.366918 0.0378  0.2928  0.4410     12 %
## y           -0.182703 0.0407 -0.2626 -0.1028     29 %
```

Model-Based vs. Donor-Based Methods

They types of MI we've discussed above are all *model-based.*

- The imputations are randomly sampled from an estimated distribution of the missing values (i.e., a probability *model* of the missing data).

Model-based methods are theoretically ideal when the missing data truly follow the chosen distribution.

- If the missing data do not follow the model, performance suffers.

Sometimes, the solution is to employ a different probability model.

- We'll see this approach when we discuss MI for categorical variables.

Model-Based vs. Donor-Based Methods

If we’re not able to choose a sensible distribution for the missing data, we can use *Donor-Based Methods.*

- Imputations are sampled from a pool of matched observed cases.
- The empirical distribution of the observed data is preserved.

One particularly useful donor-based method is *Predictive Mean Matching* (Little, 1988).

- The cases that make up the donor pool are matched based on their predicted outcome values.

Predictive Mean Matching: Procedure

Suppose we want to generate M imputations for an incomplete variable, Y, using some set of predictors, $\mathbf{X}$.

1. Regress Y_{obs} onto $\mathbf{X}_{obs}$ and compute the conditional mean of Y_{obs}:
    - $\hat{\mu} = \mathbf{X}_{obs}\hat{\beta}$
2. Do a Bayesian linear regression of Y_{obs} onto $\mathbf{X}_{obs}$ and sample M values of the posterior predicted mean of Y_{mis}:
    - $\tilde{\mu}_m = \mathbf{X}_{mis}\tilde{\beta}_m$.
3. Compute M sets of the matching distances:
    - $d(i, j)_m = (\tilde{\mu}_{mi} - \hat{\mu}_j)^2, \quad i = 1, 2, \ldots N_{mis}, \quad j = 1, 2, \ldots, N_{obs}$.

Predictive Mean Matching: Procedure

4. Use each $d(i, j)_m$ to construct N_{mis} donor pools.
   - Find the K (e.g., $K \in \{3, 5, 10\}$) cases with the smallest values of $d(1, j)_m$, $d(2, j)_m$, . . ., $d(N_{mis}, j)_m$.

5. For $m = 1, 2, \ldots, M$, select the final donor cases by randomly sampling a single observation from each of the N_{mis} donor pools defined in Step 4.

6. For each of the M imputations replace the missing values in Y with the donor data selected in Step 5.

Predictive Mean Matching: Example

Compute/sample the appropriate conditional means:

```r
## Define donor pool size:
K <- 5

## Conditional mean of Y_mis:
mu0 <- predict(fit0)

## Posterior predicted means of Y_mis:
mu1 <- as.data.frame(
    misX %*% t(betaSams[sample(1 : nSams, nImps), ])
)
```

Predictive Mean Matching: Example

Define a function to find donor cases:

```r
getDonors <- function(x, y, K) {
    ## Compute distances:
    d <- (x - y)^2

    ## Indices of the K smallest distances:
    ind <- which(order(d) %in% 1 : K)

    ## Return a randomly sampled index:
    sample(ind, 1)
}
```

Predictive Mean Matching: Example

Implement the imputation:

```
impList2 <- list()
for(m in 1 : nImps) {
    ## Find donor cases:
    d0 <- sapply(mu1[ , m], getDonors, y = mu0, K = K)

    ## Impute the missing values:
    impData                  <- misData
    impData[rVec, "y"] <- yObs$y[d0]

    ## Save the imputed dataset:
    impList2[[m]] <- impData
}
```

Predictive Mean Matching: Example

```
## Use each dataset to estimate the analysis model:
fits2 <- lapply(impList2,
                function(dat) lm(z ~ x + y, data = dat)
                )

## Pool the results:
pool2 <- MIcombine(fits2)
summary(pool2, digits = 3)

## Multiple imputation results:
##       MIcombine.default(fits2)
##             results     se  (lower  upper) missInfo
## (Intercept)  0.0277 0.0323 -0.0357  0.0910      4 %
## x            0.2805 0.0313  0.2191  0.3419      1 %
## y           -0.0940 0.0330 -0.1589 -0.0292     32 %
```

Pros and Cons of Predictive Mean Matching

PMM tends to work well with continuous, non-normal variables.

- Relatively robust to misspecification of the imputation model
- Imputed values are always valid

PMM does have some important limitations.

- In small samples, the same donor cases can be re-used many times.
- PMM cannot extrapolate beyond the observed range of the data.
- PMM cannot be used with some variable types.
    - Nominal variables
- PMM may perform poorly when the number of predictor variables is small.

References

Little, R. J. A. (1988). Missing-data adjustments in large surveys. *Journal of Business & Economic Statistics*, *6*(3), 287–296. doi: 10.1080/07350015.1988.10509663

Rubin, D. B. (1987). *Multiple imputation for nonresponse in surveys* (Vol. 519). New York, NY: John Wiley & Sons.

Van Buuren, S. (2012). *Flexible imputation of missing data.* Boca Raton, FL: CRC Press. doi: 10.1201/b11826