Univariate Multiple Imputation

Utrecht University Winter School: Missing Data in R

Utrecht University

Kyle M. Lang

Department of Methodology & Statistics
Utrecht University

2022-02-03

Outline

Prediction

Many of us learn supervised learning from the perspective of estimation and inference.

- Asking questions about how $\mathbf{X}$ is related to Y

We can also supervised learning for *prediction*.

- Given a new observation, X_m, what outcome value, $\hat{Y}_m$, does our model attribute to the mth observation?

Prediction

Train a model to predict employee performance using features extracted from CVs.

- When screening applicants for a new position, use the data in their CVs to predict their expected performance.

Predict recidivism risk based on personal history, criminal history, and in-prison behavior record.

- When evaluating a parole application, calculate the predicted chance of recidivism.

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:

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

Single Imputation

Imputation is Just Prediction*

In Lecture 3, you heard a bit about missing data imputation.

- Multiple imputation is one of the best ways to treat missing data.

Imputation is nothing more than a type of prediction.

1. Train a model on the observed parts of the data, Y_{obs}.
   - Train the imputation model.
2. Predict the missing values, Y_{mis}.
   - Generate imputations.
3. Replace the missing values with these predictions.
   - Impute the missing data.

Imputation can be used to support either prediction or inference.

- Our goals will dictate what type of imputation we need to do.

*Levels of Uncertainty Modeling

van Buuren (2018) 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

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

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)

dat0 <- as.data.frame(rmvnorm(nObs, c(0, 0, 0), sigma))
colnames(dat0) <- c("y", "x", "z")
```

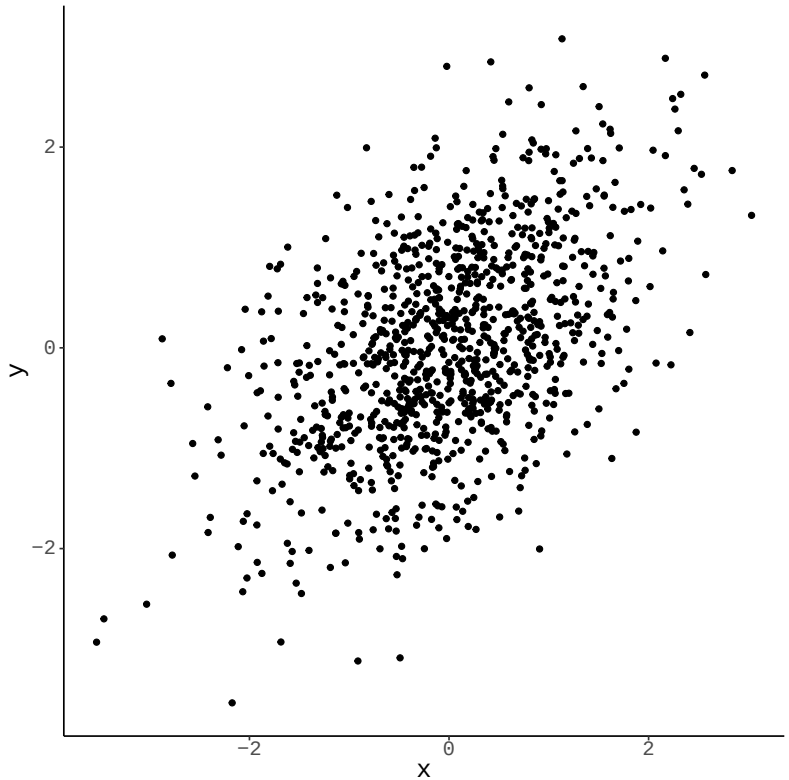
Simulate Some Toy Data

```
## Impose MAR Nonresponse:
dat1 <- dat0
mVec <- with(dat1, x < quantile(x, probs = pm))

dat1[mVec, "y"] <- NA

## Subset the data:
yMis <- dat1[mVec, ]
yObs <- dat1[!mVec, ]
```

Look at the Data

```
round(head(dat0, n = 5), 3)
```

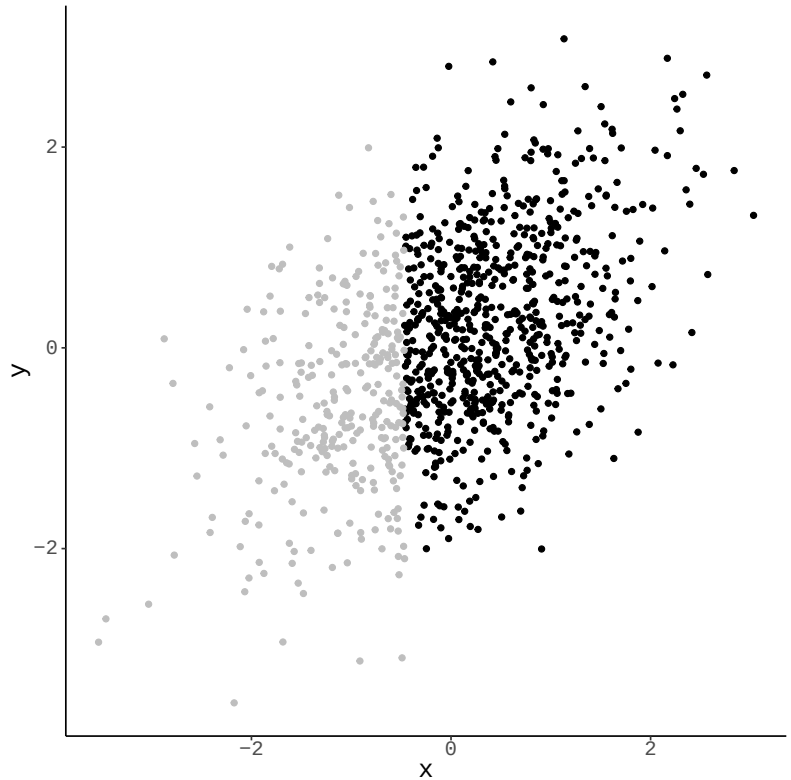
```
       y      x      z
1 -0.370 -1.546 -0.290
2  1.237 -0.621  1.352
3 -0.896 -0.658 -0.500
4  0.963  0.248  0.268
5 -0.007  0.208  0.702
```



Look at the Data

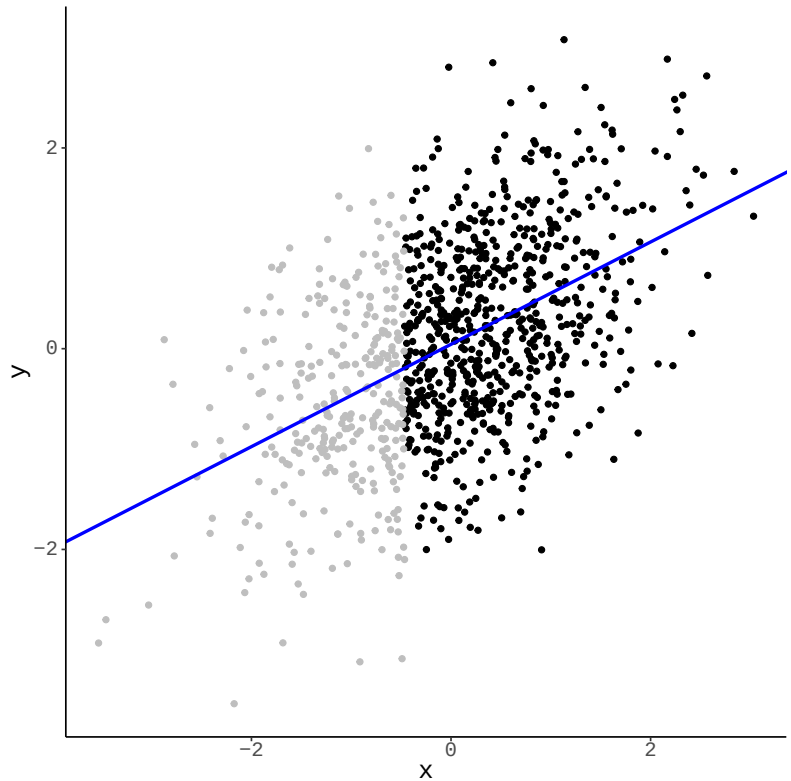
```
round(head(dat1, n = 5), 3)
```

```
       y      x      z
1     NA -1.546 -0.290
2     NA -0.621  1.352
3     NA -0.658 -0.500
4  0.963  0.248  0.268
5 -0.007  0.208  0.702
```



Expected Imputation Model Parameters

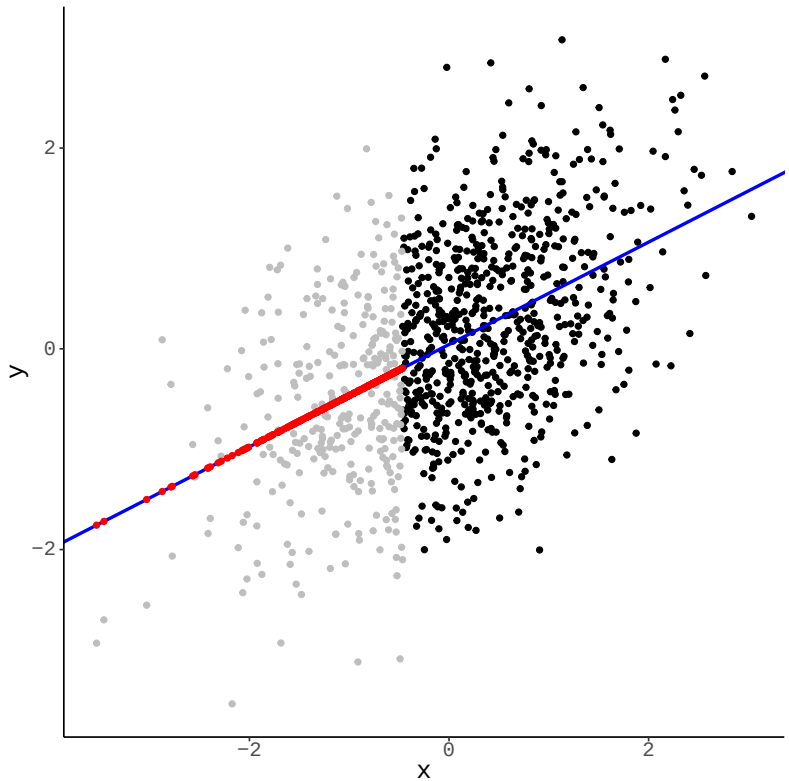
```
lsFit <- lm(y ~ x + z, data = yObs)

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

as.matrix(beta)

                    [,1]
(Intercept)  0.04646748
x            0.54470194
z           -0.14796492

sigma

[1] 0.8417349
```



Conditional Mean Substitution

```
## Generate imputations:
imps <- beta[1] +
    beta[2] * yMis[ , "x"] +
    beta[3] * yMis[ , "z"]

## Fill missing cells in Y:
dat1[mVec, "y"] <- imps

round(head(dat1, n = 5), 3)
```

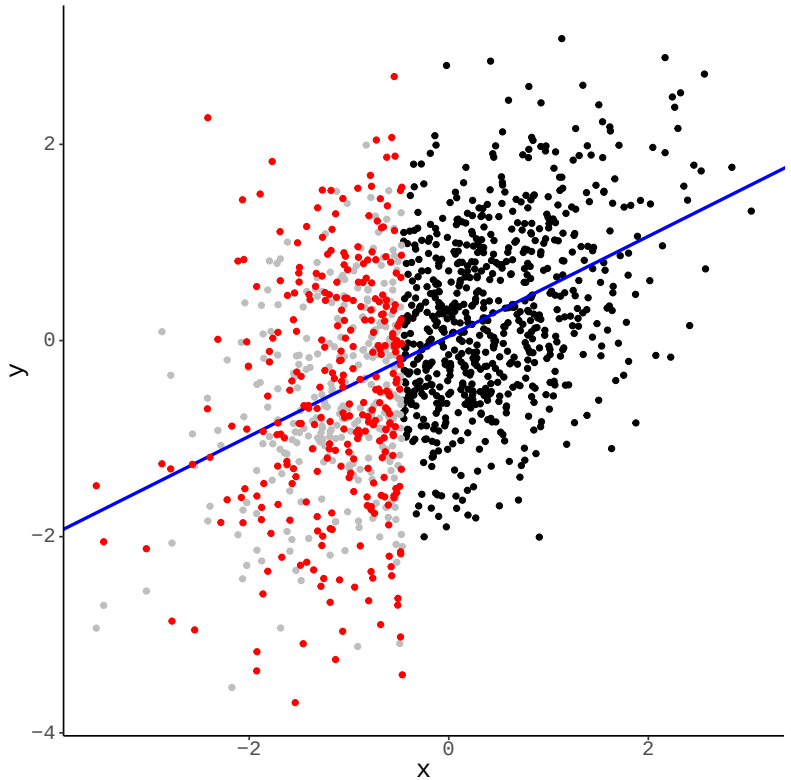
```
       y      x      z
1 -0.752 -1.546 -0.290
2 -0.492 -0.621  1.352
3 -0.238 -0.658 -0.500
4  0.963  0.248  0.268
5 -0.007  0.208  0.702
```



Stochastic Regression Imputation

```
## Generate imputations:
imps <- imps +
    rnorm(nrow(yMis), 0, sigma)

## Fill missing cells in Y:
dat1[mVec, "y"] <- imps

round(head(dat1, n = 5), 3)
```

```
       y      x      z
1 -0.710 -1.546 -0.290
2  0.873 -0.621  1.352
3 -0.134 -0.658 -0.500
4  0.963  0.248  0.268
5 -0.007  0.208  0.702
```



Multiple Imputation

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

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.

$$Y_{imp} = \mathbf{X}_{mis}\tilde{\beta} + \tilde{\varepsilon} \qquad (3)$$

$$\text{with } \varepsilon \sim \mathrm{N}\left(0, \widetilde{\sigma^2}\right)$$

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.

```
## 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.

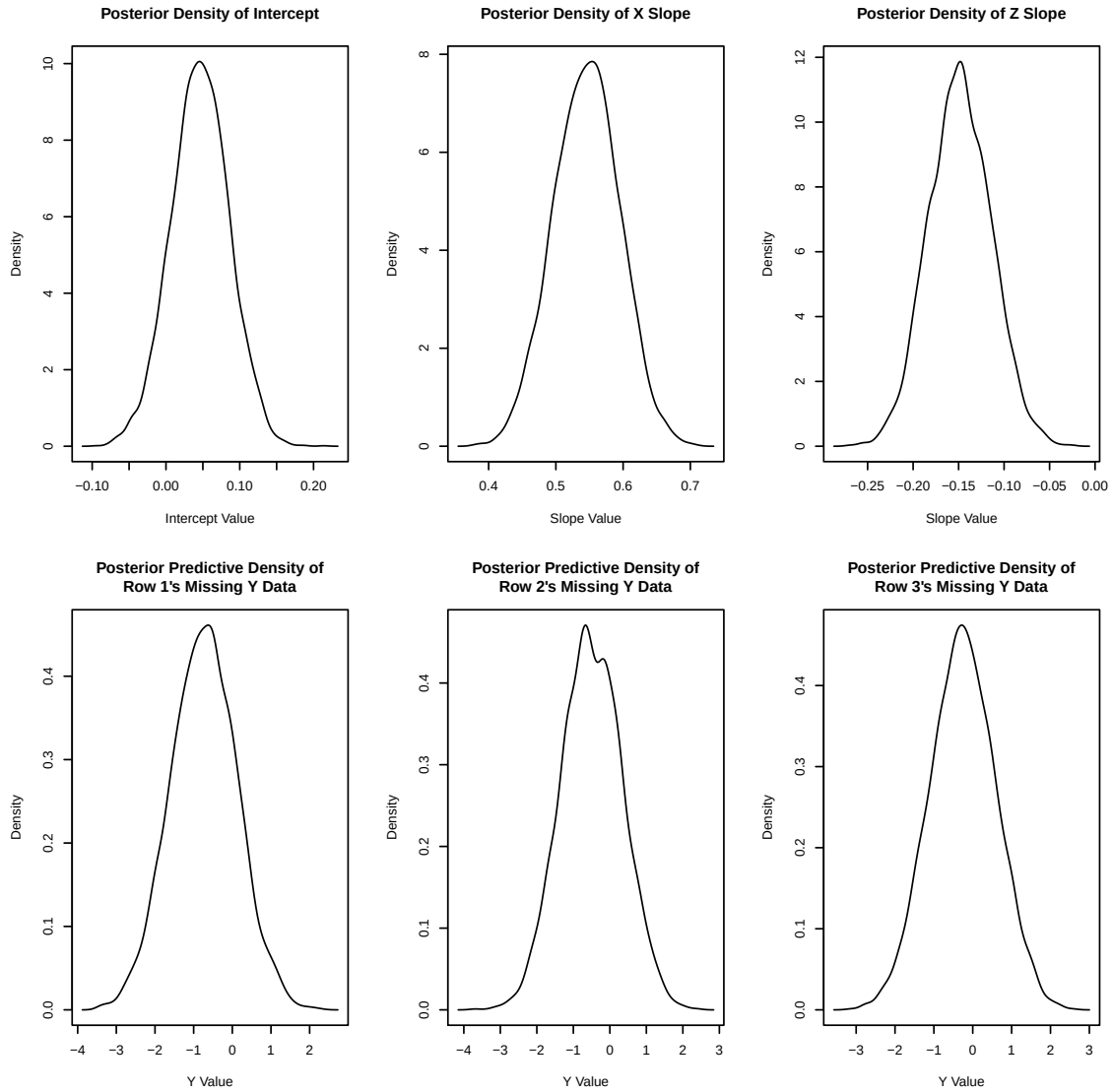
```
## Partition the predictor matrix:
xMis <- as.matrix(cbind(1, yMis[ , c("x", "z")]))
xObs <- as.matrix(cbind(1, yObs[ , c("x", "z")]))

## Sample beta:
betaSams <- matrix(NA, nSams, ncol(xObs))
for(i in 1 : nSams) {
    betaVar       <- sigmaSams[i] * solve(crossprod(xObs))
    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:

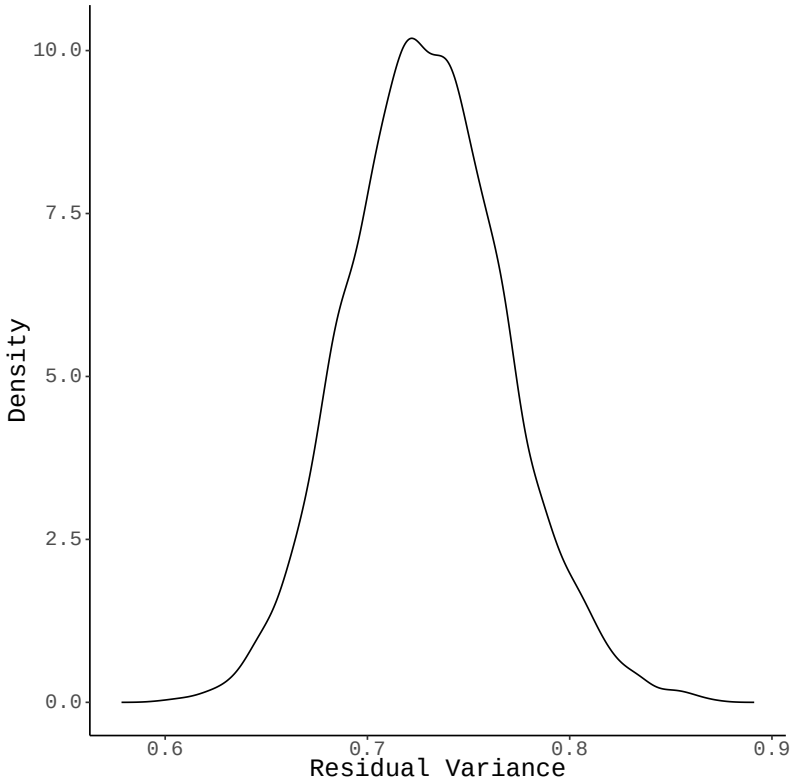
```
nMis   <- sum(mVec)
impMat <- matrix(NA, nMis, nSams)
for(i in 1 : nSams) {
    impMat[ , i] <- xMis %*% matrix(betaSams[i, ]) +
        rnorm(nMis, 0, sqrt(sigmaSams[i]))
}

## Fill the missing cells with the M imputations:
impList <- list()
ind     <- sample(1 : nSams)
for(m in 1 : nImps) {
    impList[[m]]             <- dat1
    impList[[m]][mVec, "y"] <- impMat[ , ind[m]]
}
```

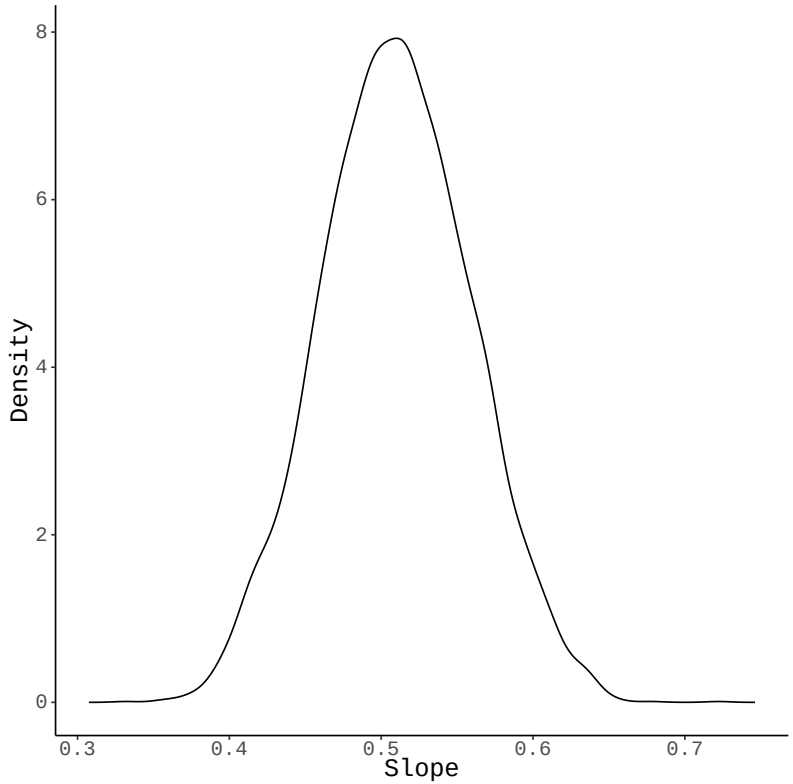
What do we get?



Visualizing MI

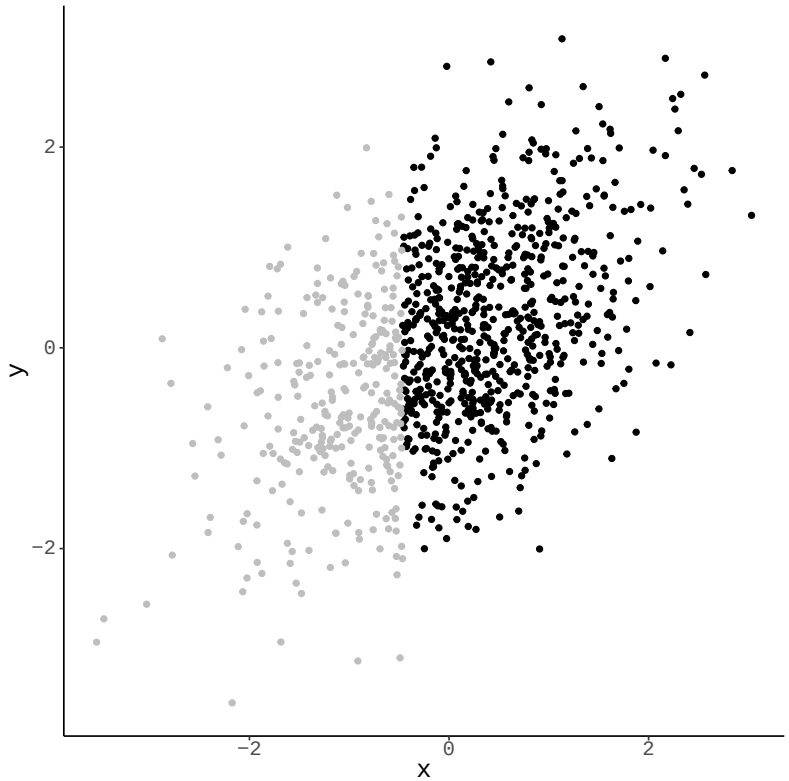
Use Bayesian simulation to estimate posterior distributions for the imputation model parameters:





Visualizing MI

Recall the incomplete data from the single imputation examples.



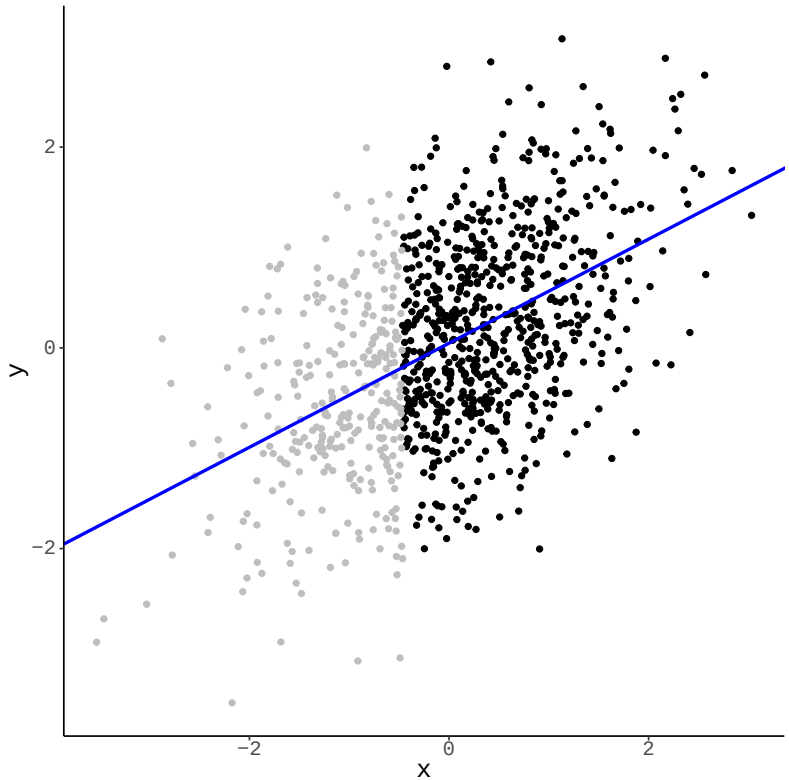
Visualizing MI

Sample values of β_0 and β_1:

- $\beta_0 = 0.046$
- $\beta_1 = 0.518$

Define the predicted best-fit line:

$\hat{Y}_{mis} = 0.046 + 0.518X_{mis}$



Visualizing MI

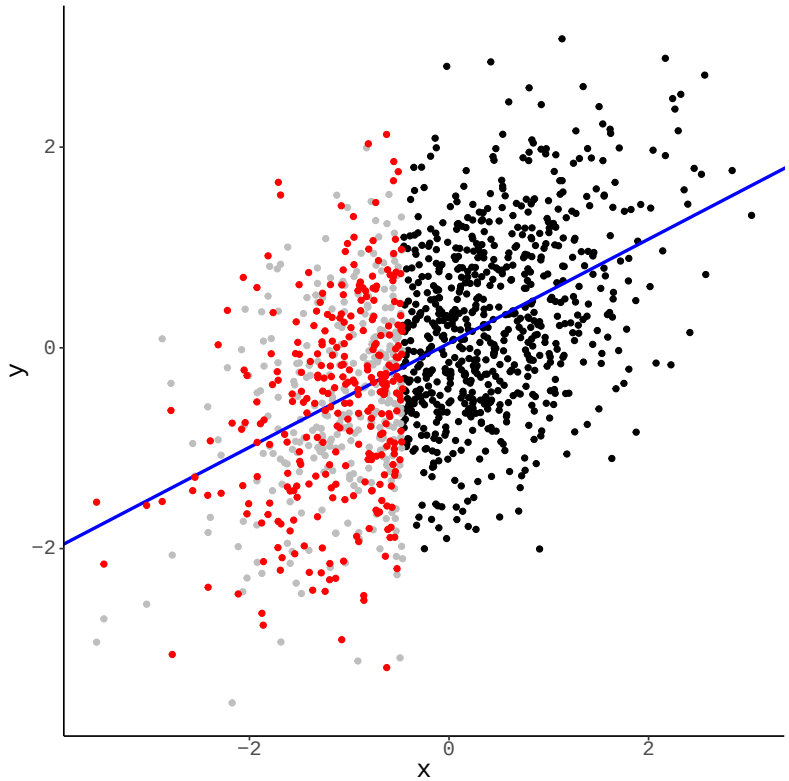
Sample a value of σ^2:

- $\sigma^2 = 0.736$

Generate imputations using the same procedure described in Single Stochastic Regression Imputation:

$$Y_{imp} = \hat{Y}_{mis} + \varepsilon$$

$$\varepsilon \sim \text{N}(0, 0.736)$$



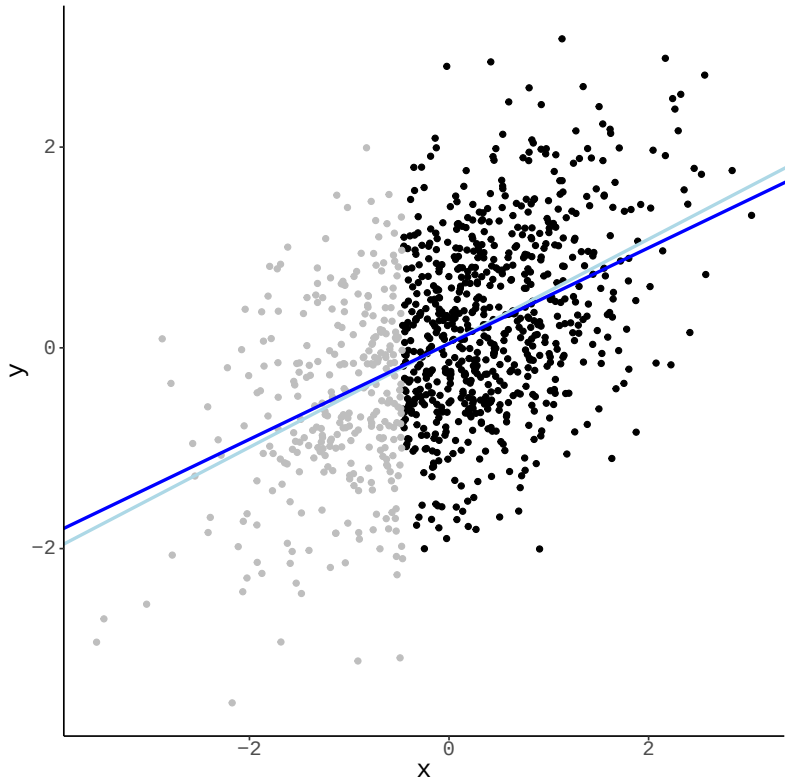
Visualizing MI

Sample values of β_0 and β_1:

- $\beta_0 = 0.042$
- $\beta_1 = 0.477$

Define the predicted best-fit line:

$\hat{Y}_{mis} = 0.042 + 0.477 X_{mis}$



Visualizing MI

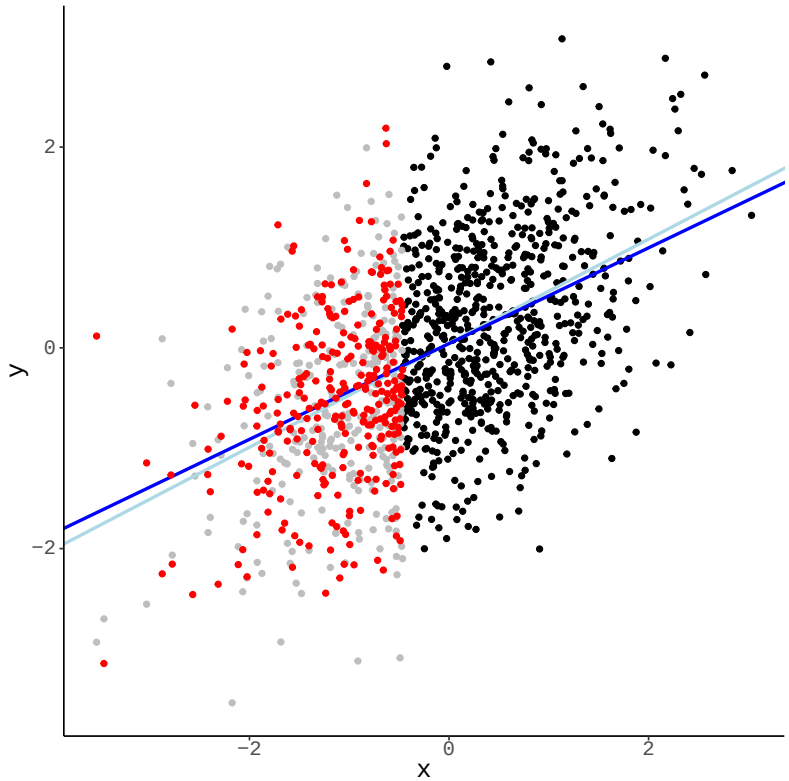
Sample a value of σ^2:

- $\sigma^2 = 0.673$

Generate imputations using the same procedure described in Single Stochastic Regression Imputation:

$$Y_{imp} = \hat{Y}_{mis} + \varepsilon$$

$$\varepsilon \sim \text{N}(0, 0.673)$$



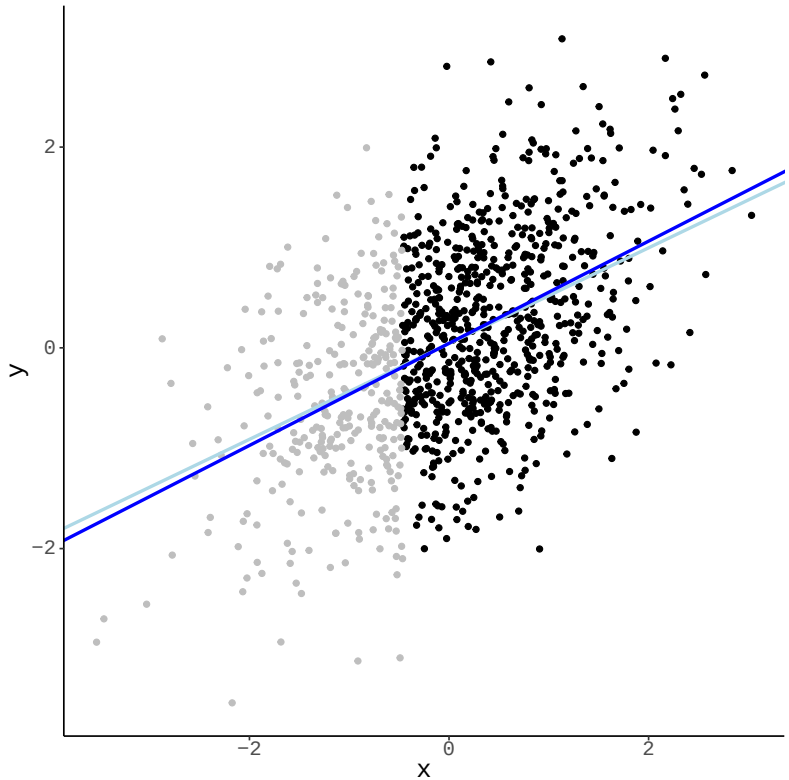
Visualizing MI

Sample values of β_0 and β_1:

- $\beta_0 = 0.045$
- $\beta_1 = 0.509$

Define the predicted best-fit line:

$\hat{Y}_{mis} = 0.045 + 0.509X_{mis}$



Visualizing MI

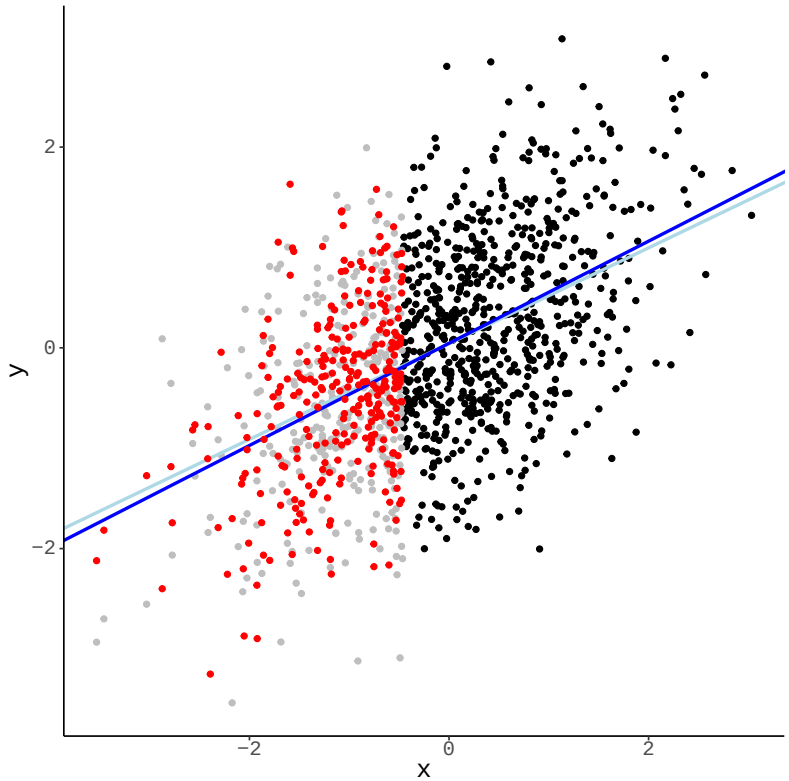
Sample a value of σ^2:

- $\sigma^2 = 0.671$

Generate imputations using the same procedure described in Single Stochastic Regression Imputation:

$$Y_{imp} = \hat{Y}_{mis} + \varepsilon$$

$$\varepsilon \sim \text{N}(0, 0.671)$$



MI-Based Analysis

Doing MI–Based Analysis

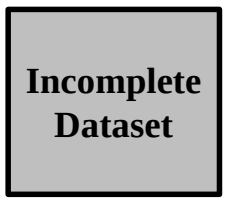
An MI-based data analysis consists of three phases:

1. The imputation phase
   - Replace missing values with M plausible estimates.
   - Produce M completed datasets.

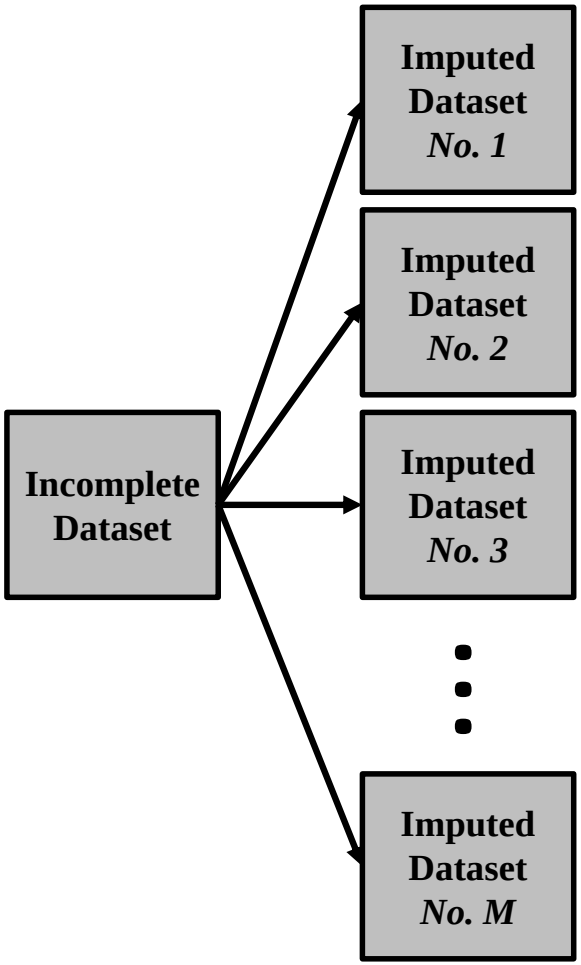
2. The analysis phase
   - Estimate M replicates of your analysis model.
   - Fit the same model to each of the M datasets from Step 1.

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.

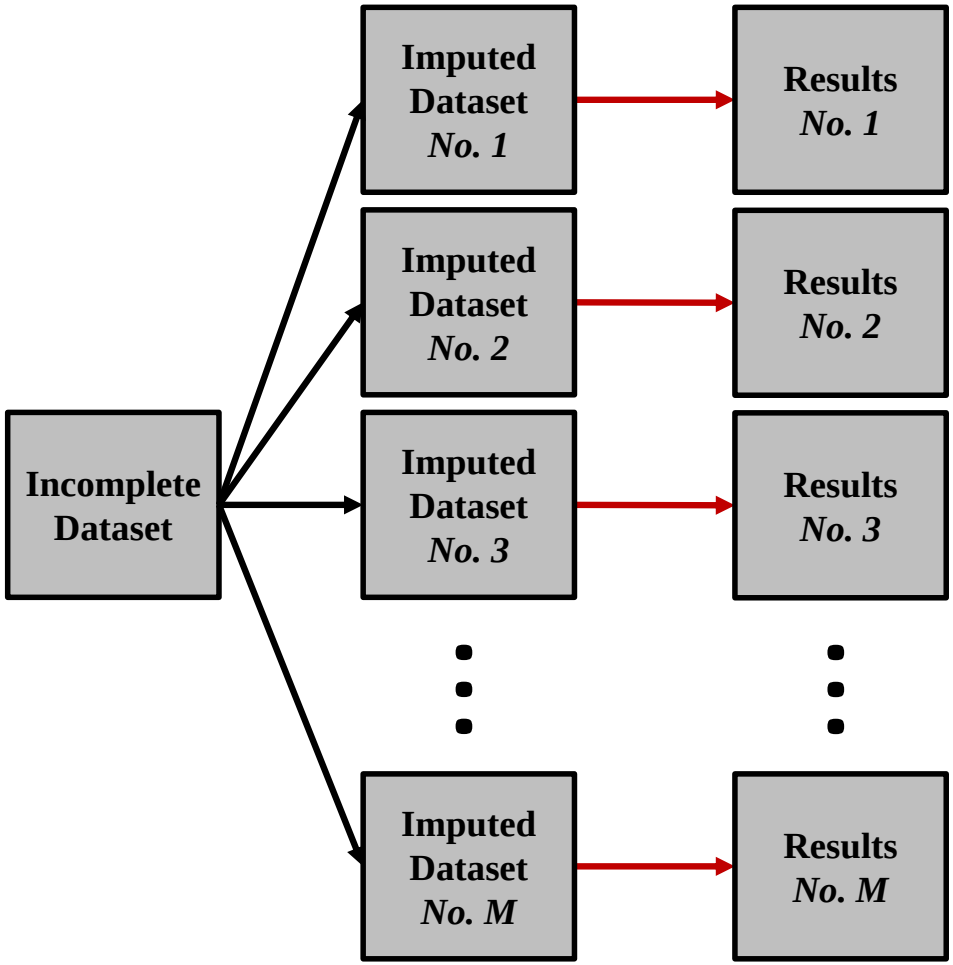
MI-Based Analysis



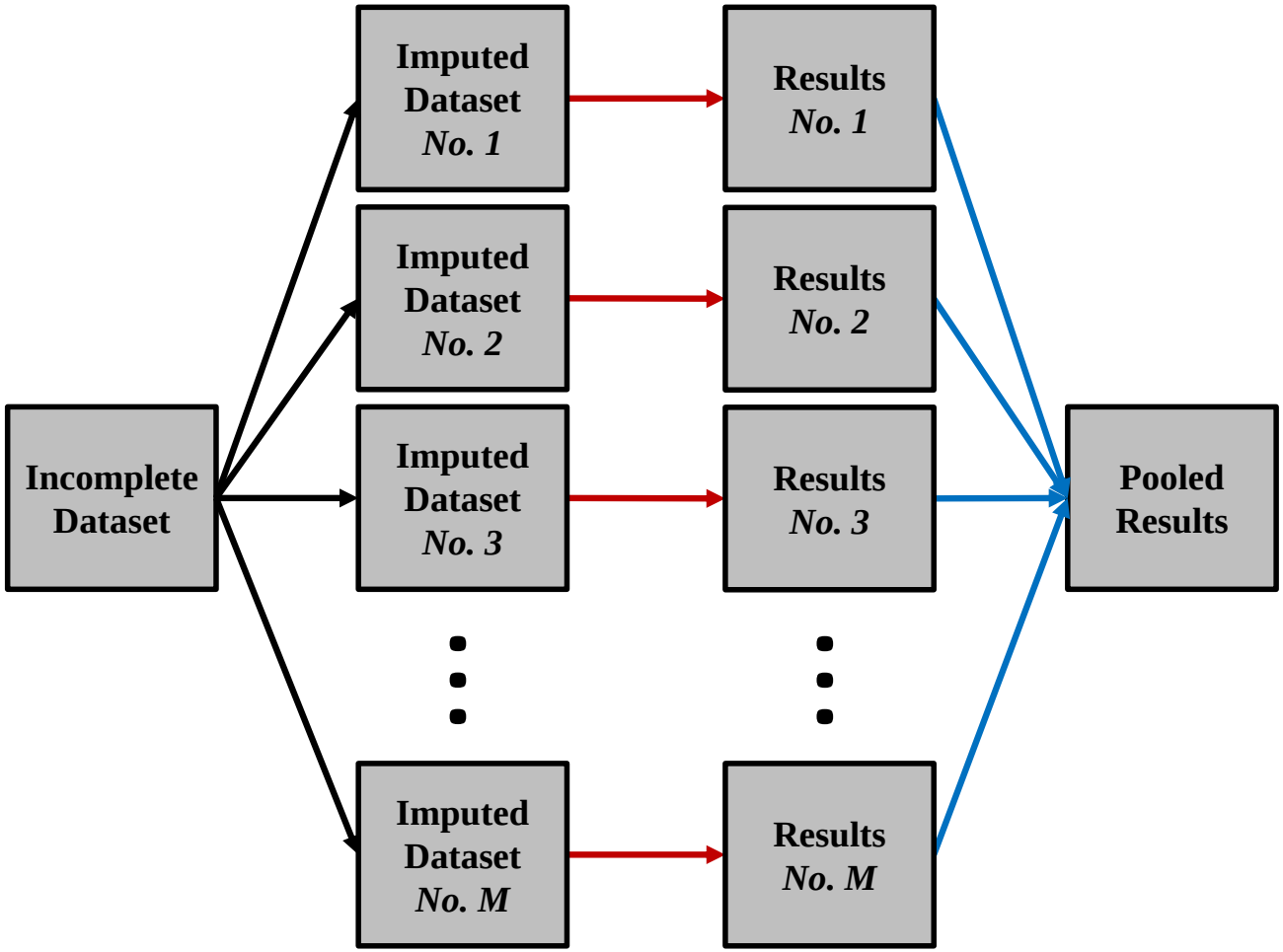
MI-Based Analysis



MI-Based Analysis



MI-Based Analysis



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.

$$W = \frac{1}{M}\sum_{m=1}^{M}\widehat{SE}_{Q,m}^{2}$$

$$B = \frac{1}{M-1}\sum_{m=1}^{M}\left(\hat{Q}_m - Q^*\right)^2$$

$$T = W + \left(1 + M^{-1}\right)B$$

$$= W + B + \frac{B}{M}$$

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:
fits <- lapply(impList,
               function(dat) lm(z ~ x + y, data = dat)
               )

## Pool the results:
MIcombine(fits) %>% summary(digits = 3)

Multiple imputation results:
      MIcombine.default(fits)
             results     se  (lower  upper) missInfo
(Intercept)   0.011 0.0299 -0.0477  0.0697      4 %
x             0.355 0.0378  0.2813  0.4296     14 %
y            -0.175 0.0416 -0.2568 -0.0937     31 %
```

Special Pooling Considerations

The Rubin (1987) pooling rules only hold when the parameter of interest, Q, follows an approximately normal sampling distribution.

- For substantially non-normal parameters, we may want to transform before pooling and back-transform the pooled estimate.

The following table, reproduced from van Buuren (2018), shows some recommended transformations.

Statistic	**Transformation**	**Source**
Correlation	Fisher's z	Schafer (1997)
Odds ratio	Logarithm	Agresti (2013)
Relative risk	Logarithm	Agresti (2013)
Hazard ratio	Logarithm	Marshall et al. (2009)
R^2	Fisher's z on square root	Harel (2009)
Survival probabilities	Complementary log-log	Marshall et al. (2009)
Survival distribution	Logarithm	Marshall et al. (2009)

Pooling Predictions

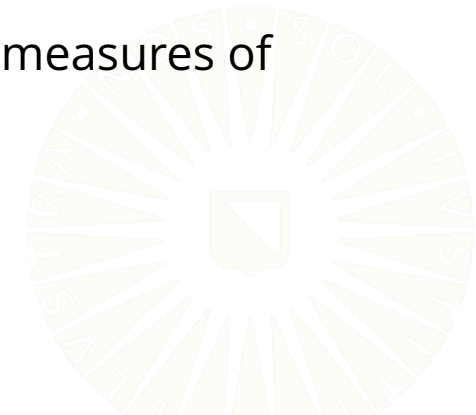
When doing an MI-based analysis, we generally want to pool results as late as possible in the analytic process.

- This pattern also holds when doing prediction with MI data (Wood, Royston, & White, 2015).
- When doing prediction, we pool the M sets of predictions.
  - We don't generate predictions using the pooled parameters.
  - *Caveat:* For GLMs, we pool predictions before applying the inverse link function.
- When pooling fit measures based on predictions (e.g., MSE), we pool the M estimates of fit.
  - We don't generate fit values using pooled predictions or parameters.
- Variability between the M predictions (or any estimates derived therefrom) quantifies uncertainty due to missing data.

Pooling Predictions

According to Wood et al. (2015), the most natural approach also tends to perform best:

1. Train the prediction model on each of the M imputed datasets separately.
2. Generate M sets of predictions by submitting the fully observed future data to the M models from above.
3. Average the M sets of predictions into a single vector of predicted values.
    - When estimating prediction error, calculate M separate measures of error, and pool these estimates.



Pooling Predictions

To cross-validate predictive models with MI data, we have a few options:

1. We can simply impute the entire sample before splitting and run the cross-validation procedure on each of the M imputed datasets.
2. We can split the sample first, train the imputation model on the training set, and also use this imputation model to generate imputations for the validation data.
3. We can train separate imputation models on the training and validation data.
    - When generating the validation-set predictions, we need to cross the training- and validation-set imputations.
    - I.e., for $M_1 = 10$ sets of training-set estimates and $M_2 = 10$ validation-set imputations, we'll have $M_1 \times M_2 = 100$ predictions.

Donor-Based Methods

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:
   - $d(i,j)_m = (\tilde{\mu}_{mi} - \hat{\mu}_j)^2, \;\; i = 1, 2, \ldots N_{mis}, \;\; j = 1, 2, \ldots, N_{obs}.$

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:

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

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

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

Predictive Mean Matching: Example

Define a function to find donor cases:

```
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            <- dat1
    impData[mVec, "y"] <- yObs$y[d0]

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

Predictive Mean Matching: Example

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

## Pool the results:
MIcombine(fits) %>% summary(digits = 3)

Multiple imputation results:
      MIcombine.default(fits)
             results     se  (lower  upper) missInfo
(Intercept)   0.037 0.0315 -0.0246  0.0987      5 %
x             0.291 0.0314  0.2290  0.3522      2 %
y            -0.125 0.0374 -0.1987 -0.0520     29 %
```

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

Agresti, A. (2013). *Categorical data analysis* (3rd ed.). Hoboken, NJ: John Wiley & Sons.

Harel, O. (2009). The estimation of r^2 and adjusted r^2 in incomplete data sets using multiple imputation. *Journal of Applied Statistics*, *36*(10), 1109–1118. doi: http://doi.org/10.1080/02664760802553000

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

Marshall, A., Altman, D. G., Holder, R. L., & Royston, P. (2009). Combining estimates of interest in prognostic modelling studies after multiple imputation: Current practice and guidelines. *BMC Medical Research Methodology*, *9*(57). doi: http://doi.org/10.1186/1471-2288-9-57

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

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

References

van Buuren, S. (2018). *Flexible imputation of missing data* (2nd ed.). Boca Raton, FL: CRC Press.

Wood, A. M., Royston, P., & White, I. R. (2015). The estimation and use of predictions for the assessment of model performance using large samples with multiply imputed data. *Biometrical Journal*, *57*(4), 614–632. doi: http://doi.org/10.1002/bimj.201400004