# Statistical Analyses with Missing Data

## Day 4 MI for Multivariate missing data

Bryan Shepherd and Cindy Chen

Vanderbilt-Nigeria Biostatistics Workshop

June 2–6, 2025

# Overview

# Introduction to Missing Data Patterns

- Definition: A missing data pattern describes the structure of observed and missing values across variables in a dataset.
- Importance: Recognizing these patterns is crucial for selecting appropriate imputation methods and understanding the potential biases in analyses.

# Univariate and multivariate

- A missing data pattern is said to be **univariate** if there is only one variable with missing data (Day 3).
- A missing data pattern is said to be **multivariate** if multiple variables have missing values, which may occur:
  - Simultaneously (e.g., skipped sections in questionnaires)
  - Sequentially (e.g., dropout)

# Monotone and non-monotone (or general)

Monotone missingness:

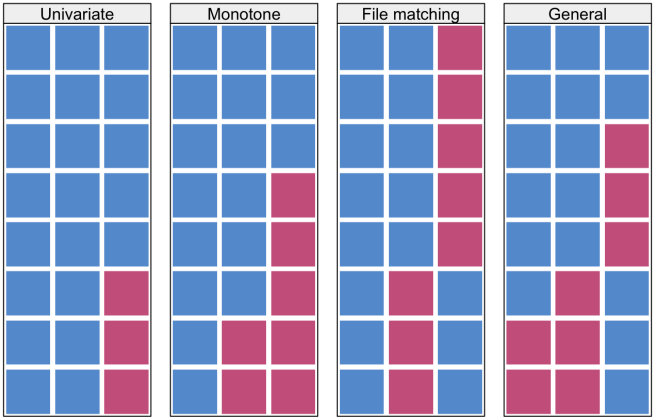
- If a variable is missing, all subsequent variables are also missing. E.g. longitudinal studies with drop-out.
- Can simplify imputation ordering

Arbitrary, non-monotone, or general missingness:

- No clear pattern; any value can be missing
- Requires iterative methods like fully conditional likelihood (FCS) – introduced later

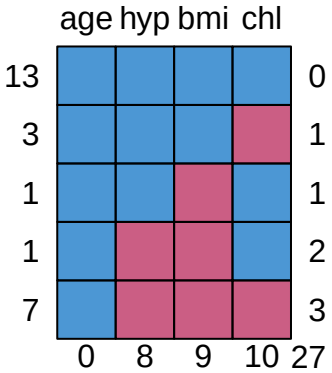
# Types of Missing Data Patterns

Missing data patterns in multivariate data. Blue is observed, red is missing (van Buuren, S. 2018)



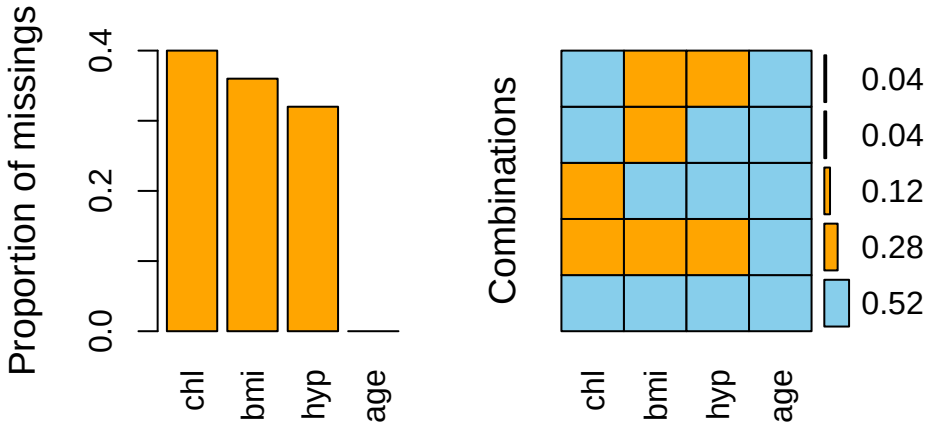
# Visualizing Patterns

```
library(mice)
library(naniar)
data(nhanes)
# Missing data pattern matrix
md.pattern(nhanes)
```



# Visualizing Patterns

```
# Missing data shadow matrix
library(VIM)
aggr(nhanes, col=c("skyblue", "orange"), numbers=TRUE, sortVars=TRUE)
```



# Overview

# Challenges in multivariate imputation

- Most imputation models for $Y_j$ use the remaining columns $Y_{-j}$ as predictors. The rationale is that conditioning on $Y_{-j}$ preserves the relations among the $Y_j$ in the imputed data.
- It could lead to various practical problems:
  - The predictors $Y_{-j}$ themselves can contain missing values;
  - "Circular" dependence can occur, where $Y_j^{mis}$ depends on $Y_h^{mis}$, and $Y_h^{mis}$ depends on $Y_j^{mis}$ with $h \neq j$, because in general $Y_j$ and $Y_h$ are correlated, even given other variables;
  - Variables are often of different types (e.g., binary, unordered, ordered, continuous), thereby making the application of theoretically convenient models, such as the multivariate normal, theoretically inappropriate;
  - Especially with large $p$ and small $n$, collinearity or empty cells can occur;
  - The ordering of the rows and columns can be meaningful, e.g., as in longitudinal data;
  - The relation between $Y_j$ and predictors $Y_{-j}$ can be complex, e.g., nonlinear, or subject to censoring processes;
  - Imputation can create impossible combinations, such as pregnant fathers.
  - ...

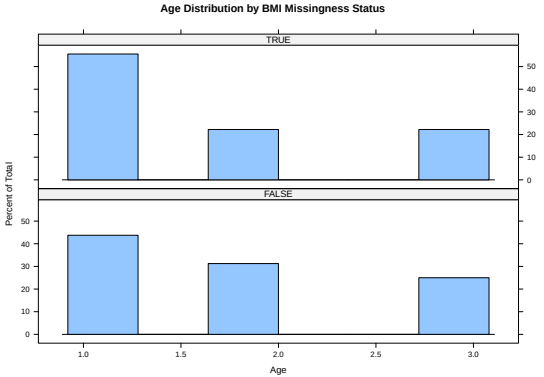
# Missing Data Mechanisms in Multivariate Settings (MAR Focus)

- The validity of standard multiple imputation techniques, including MICE, relies heavily on the Missing At Random (MAR) assumption.
- **Checking MAR**: The MAR assumption is fundamentally untestable from the observed data alone, as it depends on the unobserved values. Sensitivity analyses are needed to assess the potential impact of departures from MAR. For this course, we proceed assuming MAR holds.
- **Inspecting MAR Plausibility**: While MAR cannot be proven, we can look for evidence against it or support its plausibility. The vignette 'Inspecting how the observed data and missingness are related' (Gerko Vink and Stef van Buuren) suggests comparing distributions of observed values conditional on the missingness status of other variables.

# Missing Data Mechanisms in Multivariate Settings (MAR Focus)

```r
require(mice)
require(lattice)
 # Create missingness indicator for bmi
nhanes$bmi_missing <- is.na(nhanes$bmi)

# Plot age distribution conditional on bmi missingness
histogram(~age | bmi_missing, data = nhanes,
          xlab = "Age", layout = c(1, 2),
          main = "Age Distribution by BMI Missingness Status")
```




- **Interpretation**: If the age distributions look substantially different (e.g., younger kids are more likely to have missing `BMI`), it suggests age might be related to the missingness of `bmi`. This supports the MAR assumption (as age is observed) and highlights the importance of including age in the imputation model for `bmi`.

# Overview

# Joint Modeling (JM)

- **Concept**: JM assumes that the partially observed data $Y = (Y_1, ..., Y_p)$ follow some multivariate distribution, $P(Y \mid \theta)$. Imputation involves:
  1. Estimating the parameters $\theta$ of this joint distribution from the observed data.
  2. Drawing imputations for the missing values ($Y_{mis}$) from their conditional distribution given the observed data ($Y_{obs}$), i.e., $P(Y_{mis} \mid Y_{obs}, \theta)$.
- Common Models for continuous data:
  - Multivariate Normal (MVN): Assumes data follow an MVN distribution. Imputation often uses MCMC techniques to estimate the mean vector and covariance matrix and draw imputations. Suitable for continuous, approximately normally distributed data with transformation.
- Common Models for categorical data:
  - MVN with rounding
  - Log-linear Models: Used for categorical data, modeling cell counts in a contingency table.
  - Latent class (or finite mixture) model

# Joint Modeling (JM)

- **Pros**: Theoretically well-grounded if the assumed joint distribution is appropriate. Ensures compatibility among imputed variables by design.
- **Cons**: Can be difficult to specify a suitable joint model for datasets with mixed variable types (continuous, binary, categorical) or complex distributions. Standard `JM` software might lack flexibility for specific variable types or constraints.

# Fully Conditional Specification (FCS)/Chained Equations

- **Concept**: FCS avoids specifying a potentially complex multivariate joint distribution. Instead, it specifies a separate univariate imputation model for each variable with missing data, conditional on all other variables in the dataset. It uses an iterative approach (chained equations) to generate imputations (MICE).
- The Iterative Process (MICE Algorithm):
  1. Initialization: Start with simple initial imputations (e.g., mean imputation) for all missing values.
  2. Iteration: Cycle through the variables with missing data ($Y_1, Y_2, ..., Y_k$). For each variable $Y_j$:
     - Set its current imputed values back to missing.
     - Fit a univariate regression model for $Y_j$ using all other variables ($Y_{-j}$) as predictors (using their current observed or imputed values). The model type (e.g., linear, logistic) is chosen based on the type of $Y_j$.
     - Impute the missing values of $Y_j$ by drawing from the predictive distribution derived from this model.
  3. Repeat: Repeat Step 2 for a specified number of iterations (maxit). The imputations are updated in each cycle.
  4. Multiple Imputations: Repeat the entire process (Steps 1-3) $m$ times, starting from potentially different initial imputations or using different random draws, to generate $m$ completed datasets.

# Fully Conditional Specification (FCS)/Chained Equations

- **Pros**: Highly flexible; allows tailoring the imputation model to each variable's type and distribution. Easily handles mixed data types. Conceptually simpler for practitioners to specify univariate models. Empirically shown to work well in many situations.
- **Cons**: Theoretical foundation is less direct than JM. The specified conditional models may not correspond to a valid joint distribution ("incompatibility"). Convergence needs careful monitoring.

# Comparison

| Approach | Strengths | Weaknesses |
|---|---|---|
| Joint Modeling | Theoretically coherent | Assumes multivariate normality |
| FCS | Flexible with variable types | May lack coherence |

While JM has theoretical appeal, FCS (implemented via MICE) has become the dominant approach in practice due to its flexibility and the availability of robust software like the mice package. It readily handles the mix of variable types common in real datasets. This course will focus on understanding and applying the MICE algorithm.

# Overview

# Assessing Convergence

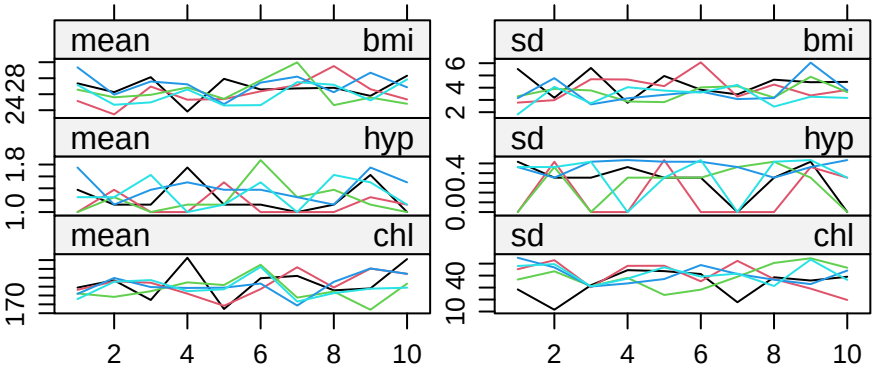
Since MICE is an iterative algorithm, we need to ensure it has run long enough to reach a stable state (convergence). Non-convergence means the imputed values might depend heavily on the starting values or the iteration number, making them unreliable.

- **Trace Plots**: The primary diagnostic tool is the trace plot generated by `plot(mids_object)`. These plots show the mean and standard deviation of the imputed values for each variable across iterations for each of the $m$ imputation chains.

# Assessing Convergence

```r
# Perform imputation (using nhanes data from earlier)
imp <- mice(nhanes, m = 5, maxit = 10, seed = 123, print = FALSE)

# Plot convergence diagnostics for specific variables
plot(imp, c("bmi", "hyp", "chl"))
```



# Assessing Convergence

- **Other Diagnostics**: While trace plots are standard, advanced diagnostics exist (e.g., autocorrelation plots, potential scale reduction factor - PSRF, though PSRF is more common in Bayesian MCMC), but trace plots are the main tool recommended in mice documentation for assessing MICE convergence.

# R Implementation: FCS via `mice()`

```r
nhanes2 <- nhanes %>%
           mutate(hyp.fac=factor(hyp)) %>%
           select(age, bmi, hyp.fac, chl)

# Default methods for nhanes
imp_default <- mice(nhanes2, maxit = 0) # Run 0 iterations to see setup
print(imp_default$method)
```

```
     age      bmi  hyp.fac      chl
      ""    "pmm" "logreg"    "pmm"
```

```r
# Specify methods explicitly
my_methods <- imp_default$method # Start with defaults
my_methods["bmi"] <- "norm"     # Change bmi to Bayesian linear regression
my_methods["hyp.fac"] <- "logreg"   # Explicitly set hyp to logistic regression
my_methods["chl"] <- "pmm"     # Explicitly set chl to pmm

# Check methods
print(my_methods)
```

```
     age      bmi  hyp.fac      chl
      ""   "norm" "logreg"    "pmm"
```

```r
# Run imputation with specified methods
imp_custom_meth <- mice(nhanes2, method = my_methods, m = 5, seed = 123, print = FALSE)
```

# Choosing Imputation Models (`method` argument)

- **Considerations**: The choice of method should match the variable type and distribution. `pmm` is a good general-purpose default for continuous variables as it avoids imputing impossible values and is robust to non-normality. For categorical variables, the defaults (`logreg`, `polyreg`, `polr`) are usually appropriate. Non-parametric methods like `cart` or `rf` can be useful if complex interactions or non-linearities are suspected, but may require more computational time.

# Defining Predictors (`predictorMatrix` argument)

The predictorMatrix controls which variables are used as predictors in the univariate imputation model for each target variable. It's a square matrix where rows represent target variables (to be imputed) and columns represent predictors. A 1 indicates the column variable predicts the row variable; a 0 means it does not.

- **Default Behavior**: By default, mice uses all other variables as predictors for each target variable.
- **Customization**: Users can modify this matrix to exclude irrelevant predictors, prevent multicollinearity, or incorporate substantive knowledge.

# Defining Predictors

```
# Get the default predictor matrix
imp0 <- mice(nhanes2, maxit = 0)
predM <- imp0$predictorMatrix
print(predM)
```

```
        age bmi hyp.fac chl
age       0   1       1   1
bmi       1   0       1   1
hyp.fac   1   1       0   1
chl       1   1       1   0
```

```
# Customize: Prevent 'chl' from predicting 'bmi' and vice-versa
predM_custom <- predM
predM_custom["bmi", "chl"] <- 0
predM_custom["chl", "bmi"] <- 0
print(predM_custom)
```

```
        age bmi hyp.fac chl
age       0   1       1   1
bmi       1   0       1   0
hyp.fac   1   1       0   1
chl       1   0       1   0
```

```
# Run imputation with the custom predictor matrix
imp_custom_pred <- mice(nhanes2, predictorMatrix = predM_custom,
                        m = 5, seed = 123, print = FALSE)

# Quick check using quickpred() for automatic selection
# (based on correlation)
# Note: quickpred is useful but may not capture complex relationships
# Select predictors with cor > 0.1

pred_quick <- quickpred(nhanes2, mincor = 0.1)
print(pred_quick)
```

```
        age bmi hyp.fac chl
age       0   0       0   0
bmi       1   0       1   1
hyp.fac   1   0       0   1
chl       1   1       1   0
```

```
imp_quick <- mice(nhanes2, predictorMatrix = pred_quick,
                  m = 5, seed = 123, print = FALSE)
```

# Exercise Day 4A

For exercise Day 3D, part 2, now remove outcome $y$ from the imputation model for $x$ using `mice` package.

# Practice to define predictors

- **Include relevant variables**: The imputation model should include variables associated with the target variable and variables related to its missingness to support the MAR assumption. It's generally recommended to include the outcome variable from the planned analysis model in the predictor matrix.
- **Avoid multicollinearity**: If predictors are highly collinear, it can cause instability. Removing one of the collinear variables from the predictors for certain models might be necessary.
- **Avoid logical inconsistencies**: Don't use variables as predictors if they are consequences of the target variable or defined circularly (especially important with passive imputation).
- **Balance richness and parsimony**: While richer models are generally preferred, including too many irrelevant predictors can increase noise and computation time. `quickpred` can offer a starting point, but domain knowledge is valuable.

# Controlling Imputation Order (`visitSequence` argument)

The `visitSequence` argument determines the order in which variables are imputed within each iteration.

- **Default Behavior**: variables are visited from left to right as they appear in the data columns.
- **Customization**: Can be specified as a vector of variable names in the desired order.
- **Why Change It?**
  - Convergence: Sometimes changing the order can improve convergence speed, although the final imputations should ideally not depend heavily on the order if convergence is reached.
  - Passive Imputation: Crucial for passive imputation, where variables used in the calculation must be imputed before the passively imputed variable.
  - Monotone Imputation: If data have a monotone missingness pattern, setting visitSequence = "monotone" and maxit = 1 performs efficient non-iterative imputation (though mice doesn't check if the pattern is truly monotone).

# Controlling Imputation Order (visitSequence argument)

```
# Get the default visit sequence
visSeq_default <- imp0$visitSequence
print(visSeq_default)
```

```
[1] "age"     "bmi"     "hyp.fac" "chl"
```

```
# Define a custom sequence (e.g., putting 'bmi' first)
visSeq_custom <- c("bmi", setdiff(visSeq_default, "bmi"))
print(visSeq_custom)
```

```
[1] "bmi"     "age"     "hyp.fac" "chl"
```

```
# Run imputation with the custom sequence
imp_custom_vis <- mice(nhanes2, visitSequence = visSeq_custom, m = 5, seed = 123, print = FALSE)
```

# Advanced Techniques: Passive Imputation and Post-Processing

These techniques allow incorporating specific knowledge or constraints into the MICE algorithm.

- **Passive Imputation** (method = "~ I(...)"):
  - **Purpose**: To ensure that deterministic relationships between variables hold for the imputed values (e.g., $BMI = weight/height^2$, Total Score = Sum of Items, Interaction Term = Var1 * Var2).
  - **How it Works**: Instead of fitting a model for the derived variable (e.g., BMI), its value is calculated within each MICE iteration based on the currently imputed values of its constituent variables (e.g., weight, height). This is specified using a formula in the method vector, like meth["bmi"] <- "~I(wgt/(hgt/100)^2)".
  - Implementation Details:
    1. Specify the formula method: meth["derived_var"] <- "~ I(formula)".
    2. Adjust predictorMatrix: Ensure the derived variable does not predict its source variables to avoid circularity (e.g., predM[c("wgt", "hgt"), "bmi"] <- 0).
    3. Adjust visitSequence: Ensure source variables are imputed before the derived variable.

# Advanced Techniques: Passive Imputation and Post-Processing

```r
data(boys)
# Add BMI column (initially NA)
boys$bmi <- boys$wgt / (boys$hgt/100)^2

# Initial setup
ini <- mice(boys, maxit = 0)
meth <- ini$method
pred <- ini$predictorMatrix
vis <- ini$visitSequence

# Specify passive imputation for bmi
meth["bmi"] <- "~I(wgt / (hgt/100)^2)"

# Prevent bmi from predicting its sources
pred[c("hgt", "wgt"), "bmi"] <- 0

# Ensure hgt and wgt are visited before bmi
# Find positions
hgt_pos <- which(vis == "hgt")
wgt_pos <- which(vis == "wgt")
bmi_pos <- which(vis == "bmi")
# Simple fix: move bmi to the end if needed (more robust fixes might be needed)
if (bmi_pos < hgt_pos | bmi_pos < wgt_pos) {
  vis <- c(setdiff(vis, "bmi"), "bmi")
}
print(vis) # Check the new sequence
```

```
[1] "age" "hgt" "wgt" "bmi" "hc"  "gen" "phb" "tv"  "reg"
```

# Advanced Techniques: Passive Imputation and Post-Processing

```r
# Run imputation
    imp_passive <- mice(boys, method = meth, predictorMatrix = pred,
                        visitSequence = vis, m = 5, seed = 123, print = FALSE)

    # Check consistency in one imputed dataset
    comp1 <- complete(imp_passive, 1)
    head(comp1$bmi)
```

```
[1] 14.54177 11.77395 12.56000 14.37589 15.21361 15.11241
```

```r
    head(comp1$wgt / (comp1$hgt/100)^2) # Should match
```

```
[1] 14.54177 11.77395 12.56000 14.37589 15.21361 15.11241
```

**Benefits:** Ensures logical consistency in imputed data. Essential when derived variables are part of the imputation process or analysis.

# Post-Processing (`post` argument)

- **Purpose**: To apply custom transformations or constraints to imputed values after they are generated by the univariate imputation method, but within the same MICE iteration. Useful for bounding values (e.g., ensuring positivity, logical ranges) or applying other data cleaning steps.
- **How it Works**: The `post` argument takes a vector of character strings, where each string contains R code to be executed. The code can access the imputed values using `imp[[j]][, i]` (imputed values for variable `i` in chain `j`).

# Post-Processing (post argument)

```
# Example: Ensure imputed 'chl' (cholesterol) is non-negative
imp0 <- mice(nhanes2, maxit = 0)
post_cmd <- imp0$post # Get empty post vector
post_cmd["chl"] <- "imp[[j]][, i] <- pmax(1, imp[[j]][, i])" # Set minimum to 1

# Run imputation with post-processing
imp_post <- mice(nhanes2, post = post_cmd, m = 5, seed = 123, print = FALSE)

# Check minimum imputed chl in first imputation
min(complete(imp_post, 1)$chl, na.rm = TRUE)
```

```
[1] 113
```

```
summary(complete(imp_post, 1)$chl)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  113.0   186.0   187.0   191.6   206.0   284.0
```

- **Caution**: Post-processing directly alters the imputed values used in subsequent steps. Use with care, as it can potentially distort distributions if applied inappropriately. It's generally used for enforcing known logical constraints.
- Passive imputation and post-processing offer powerful ways to incorporate substantive knowledge into the MICE algorithm, enhancing the plausibility and consistency of the resulting imputations.

# Brief Mention: MICE Extensions (`blocks`, `formulas`)

The `mice` package offers further flexibility:

- `blocks` Argument: Allows grouping variables to be imputed together using multivariate methods (if available) or in a specific sequence within an iteration. This can be useful for variables with strong dependencies or for implementing specific imputation strategies (e.g., imputing related items as a block). It conceptually bridges FCS and JM approaches.
- `formulas` Argument: Provides an alternative to `predictorMatrix` for specifying imputation models using R's formula syntax. This makes it easier to include interactions, non-linear terms (e.g., `~ age + I(age^2) + bmi`), or other complex relationships directly into the imputation model, potentially improving compatibility with the analysis model.
- These arguments enable more advanced customization but are typically explored after mastering the core method, `predictorMatrix`, and `visitSequence` controls.

# Summary of Multivariate Imputation with MICE

We have explored the MICE algorithm, a flexible and widely used method for multivariate imputation based on Fully Conditional Specification. Key steps involve:

- Specifying univariate imputation models (method) and predictors (predictorMatrix) for each incomplete variable.
- Running the iterative mice() function to generate m imputed datasets.
- Assessing convergence using trace plots (plot()).
- Checking the plausibility of imputed values (stripplot(), densityplot()).

This process yields a mids object containing the *m* completed datasets, ready for analysis.

# Inspecting Imputed Values (Plausibility Checks)

Beyond checking algorithmic convergence, it's crucial to assess whether the imputed values themselves are plausible or reasonable. Do they look like values that could have been observed?

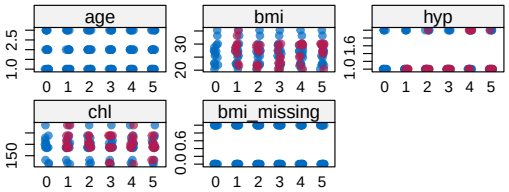
**Visual Comparison**:

- `stripplot(imp)`: Plots observed (blue) vs. imputed (red) values for each variable across the m imputations. Look for systematic differences in location or spread. Under MAR, distributions might differ, but imputed values should generally fall within a plausible range and not look drastically different from observed ones.
- `densityplot(imp)`: Compares the density curves of observed and imputed values, providing another view of distributional similarity.
- `bwplot(imp)`: Uses boxplots for comparison.

# Inspecting Imputed Values (Plausibility Checks)

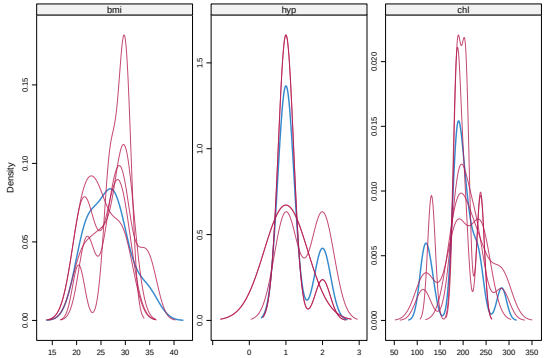
Interpretation: Large deviations between observed and imputed distributions might suggest issues with the imputation model (e.g., wrong method, inappropriate predictors) or potential violations of the MAR assumption.

```
# Assuming 'imp' is our final mids object from previous steps
stripplot(imp, pch = 20, cex = 1.2)
```

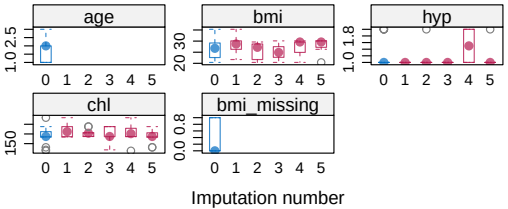


# Inspecting Imputed Values (Plausibility Checks)

```
# Assuming 'imp' is our final mids object from previous steps
densityplot(imp)
```




```
# Assuming 'imp' is our final mids object from previous steps
bwplot(imp)
```



# Inspecting Imputed Values (Plausibility Checks)

Interpretation: Large deviations between observed and imputed distributions might suggest issues with the imputation model (e.g., wrong method, inappropriate predictors) or potential violations of the MAR assumption.

```r
# Summary of original data (with NAs)
summary(nhanes2)
```

```
      age             bmi          hyp.fac        chl       
 Min.   :1.00   Min.   :20.40   1   :13   Min.   :113.0  
 1st Qu.:1.00   1st Qu.:22.65   2   : 4   1st Qu.:185.0  
 Median :2.00   Median :26.75   NA's: 8   Median :187.0  
 Mean   :1.76   Mean   :26.56             Mean   :191.4  
 3rd Qu.:2.00   3rd Qu.:28.93             3rd Qu.:212.0  
 Max.   :3.00   Max.   :35.30             Max.   :284.0  
                NA's   :9                 NA's   :10     
```

```r
# Summary of the first completed dataset
imp <- mice(nhanes2, m = 5, maxit = 10, seed = 123, print = FALSE)
summary(complete(imp, 1))
```

```
      age             bmi          hyp.fac      chl       
 Min.   :1.00   Min.   :20.40   1:17    Min.   :113  
 1st Qu.:1.00   1st Qu.:22.50   2: 8    1st Qu.:184  
 Median :2.00   Median :27.20           Median :187  
 Mean   :1.76   Mean   :26.56           Mean   :187  
 3rd Qu.:2.00   3rd Qu.:28.70           3rd Qu.:218  
 Max.   :3.00   Max.   :35.30           Max.   :284  
```

```r
# Potentially check summaries across all imputations if concerned
ind_summary <- lapply(1:imp$m, function(i) summary(complete(imp, i)))
ind_summary[[1]]
```

```
      age             bmi          hyp.fac      chl       
 Min.   :1.00   Min.   :20.40   1:17    Min.   :113  
 1st Qu.:1.00   1st Qu.:22.50   2: 8    1st Qu.:184  
 Median :2.00   Median :27.20           Median :187  
 Mean   :1.76   Mean   :26.56           Mean   :187  
 3rd Qu.:2.00   3rd Qu.:28.70           3rd Qu.:218  
 Max.   :3.00   Max.   :35.30           Max.   :284  
```

```r
ind_summary[[2]]
```

```
      age             bmi          hyp.fac      chl       
 Min.   :1.00   Min.   :20.40   1:20    Min.   :113.0  
 1st Qu.:1.00   1st Qu.:22.70   2: 5    1st Qu.:187.0  
 Median :2.00   Median :27.20           Median :199.0  
 Mean   :1.76   Mean   :26.98           Mean   :198.5  
 3rd Qu.:2.00   3rd Qu.:29.60           3rd Qu.:206.0  
 Max.   :3.00   Max.   :35.30           Max.   :284.0  
```

# Exercise Day 4B

Generate 1000 iid realizations of $(Y, X, V)$ from a known joint distribution:

$$V \sim Bernoulli(0.5);\ X|V \sim N(V, 1);\ Y|X, V \sim N(X - V, 1)$$

Generate $R_v$ $Bernoulli(0.5)$ and $R_x$ $Bernoulli(p_x)$ where $p_x = expit(V)$

Estimate regression coefficients for the model $E(Y|X, V) = \beta_0 + \beta_1 X + \beta_2 V$ in the following manner:

- Complete case analysis
- Multiple imputation using default settings in `mice`
- Multiple imputation using other settings in `mice`

How do the coefficient estimates and their standard errors differ between the three approaches?

# Summary

- We have built the foundation for handling multivariate missing data using the powerful and flexible MICE approach.
- However, there are scenarios MI is not necessary.

# When NOT to Use Multiple Imputation

Suppose that the complete-data model is a regression with outcome $Y$ and predictors $X$.

- If the missing data occur in Y only, complete-case analysis and multiple imputation are equivalent, so then complete-case analysis is preferred since it is easier, more efficient and more robust (Von Hippel 2007).
- Multiple imputation gains an advantage over complete-case analysis if additional predictors for $Y$ are available that are not part of $X$. The efficiency of complete-case analysis declines if $X$ contains missing values, which may result in inflated type II error rates.
- Complete-case analysis can perform quite badly under MAR and some MNAR cases (Schafer and Graham 2002), but there are two special cases where complete-case outperforms multiple imputation.
  - The probability to be missing covariate does not depend on $Y$: Assume the complete-data model is correct, the regression coefficients are unbiased (Little 1992; King et al. 2001). This holds for any type of regression analysis, and for missing data in both $Y$ and $X$.
  - When the outcome model is Logistic regression (more in next slide).

# When NOT to Use Multiple Imputation

- When the outcome model is logistic regression, complete-case could outperform multiple imputation (Carpenter et al. "Multiple Imputation and its Application", Table 1.9).
- Notice that the bias does not depend on which variable has missing data, but instead on the mechanism which differentiates, or selects, the complete records from the rest of the sample. However, the appropriate approach for handling the bias (e.g. multiple imputation) will depend on the variable that is actually missing.

Table 2: Biased estimation of parameters using complete records

| | Typical Regression | | | Logistic Regression | | |
|---|---|---|---|---|---|---|
| Mechanism depends on | Constant | Coef. of X | Coef. of Z | Constant | Coef. of X | Coef. of Z |
| Y | Yes | Yes | Yes | Yes | No | No |
| X | No | No | No | No | No | No |
| Z | No | No | No | No | No | No |
| X,Z | No | No | No | No | No | No |
| Y,X | Yes | Yes | Yes | Yes | Yes | No |
| Y,Z | Yes | Yes | Yes | Yes | No | Yes |
| Y,X,Z | Yes | Yes | Yes | Yes | Yes | Yes |

*Note:* Outcome Y, Covariates X, Confounders Z

# Overview

# MICE as a Gibbs Sampler: An Intuitive Analogy

The iterative nature of MICE, where each variable is updated conditional on the current state of others, closely resembles a Gibbs sampler, a type of Markov Chain Monte Carlo (MCMC) algorithm.

- **Gibbs Sampling Basics**: Gibbs sampling is used to draw samples from a complex multivariate distribution (the target joint distribution) by iteratively drawing samples from simpler conditional distributions. Imagine trying to understand the joint distribution of height, weight, and age. A Gibbs sampler might:
  1. Start with initial guesses for (height, weight, age).
  2. Sample a new height, conditional on the current weight and age.
  3. Sample a new weight, conditional on the new height and current age.
  4. Sample a new age, conditional on the new height and new weight.
  5. Repeat steps 2-4 many times. Under certain conditions, the sequence of sampled values eventually converges to represent draws from the true joint distribution.

# MICE as a Gibbs Sampler: An Intuitive Analogy

- **MICE Analogy**: In MICE, the "target distribution" is the underlying (unknown) joint distribution of the complete data. Each step in the iteration (imputing variable $Y_j$ given $Y_{-j}$) is analogous to drawing from a conditional distribution $P(Y_j \mid Y_{-j})$. By iteratively updating each variable based on the others, the algorithm aims to converge to a state where the imputed values are plausible draws from the true conditional distributions, thus implicitly reflecting the joint distribution.
- **Key Difference (Compatibility)**: A crucial distinction is that in standard Gibbs sampling, the conditional distributions are derived from a known joint distribution, ensuring compatibility. In MICE/FCS, the conditionals are specified individually, and may not perfectly correspond to a single underlying joint distribution. This is the "compatibility" issue discussed earlier. Despite this theoretical nuance, the Gibbs sampling analogy helps understand the iterative, conditional updating process.

# Compatibility of Conditional Models

A theoretical concern with FCS/MICE is compatibility: do the specified univariate conditional models $P(Y_j \mid Y_{-j})$ mathematically correspond to a single, valid joint distribution $P(Y_1, ..., Y_p)$?

- **The Issue**: If we specify, say, $P(Y_1 \mid Y_2)$ as linear and $P(Y_2 \mid Y_1)$ as quadratic, there might be no joint distribution $P(Y_1, Y_2)$ that satisfies both conditions simultaneously. The conditional models are then "incompatible".
- **Theoretical Implications**: Incompatibility means the MICE algorithm isn't technically drawing from a single, well-defined posterior predictive distribution of the missing data. The theoretical guarantees of Gibbs samplers converging to the target joint distribution don't strictly apply. The validity of Rubin's Rules for pooling might also be questioned.
- **Practical Implications**:
  - Convergence: Severe incompatibility could lead to non-convergence or instability in the MICE iterations.
  - Bias: More commonly, incompatibility between the imputation models and the substantive analysis model can lead to bias. For example, if the analysis model includes $Y_1^2$ but the imputation model for $Y_1$ only assumes a linear relationship with predictors, the imputed $Y_1$ values might not correctly reflect the quadratic effect, biasing the analysis results.

# Compatibility of Conditional Models

- **Does it Matter in Practice?** Despite the theoretical concerns, FCS often performs well empirically. Severe problems seem rare unless there's a strong conflict between imputation and analysis models. Recent work also provides theoretical justification that FCS can identify the correct conditional distributions under MAR, suggesting the approach is fundamentally sound in principle.
- **Mitigation**:
  - Ensure the imputation model is at least as complex as the analysis model (e.g., include interactions/non-linear terms in the imputation if they are in the analysis). The formulas argument can facilitate this.
  - Use passive imputation to handle derived terms consistently.
  - Monitor convergence and plausibility checks carefully.
  - Be aware of the issue, especially when analysis models are complex.

# Congeniality vs. FCS Compatibility:

- FCS Compatibility (Internal): Concerns whether the set of univariate conditional models used in MICE could theoretically arise from a single, coherent multivariate distribution. This is about the internal consistency of the imputation algorithm itself. While MICE's flexibility is a strength, these conditional models might not always be mathematically compatible. However, MICE often performs well in practice despite potential minor incompatibilities.
- Congeniality (External): Refers to whether the overall imputation model (produced by MICE or another method) aligns with the separate analysis model. This is about the external consistency between the imputation stage and the analysis stage. Even an internally sound imputation model must also be congenial with the subsequent analysis model for valid final inferences.