Heart Failure Clinical Records Synthetic Data Project

Linus Chirchir

7 September 2025

Contents

# Load necessary libraries  
# --- Project and Performance Utilities ---  
library(here) # Manage project-relative file paths  
library(doParallel) # Enable parallel processing for faster computations  
library(tidyverse) # Core data science toolkit (dplyr, tidyr, readr, purrr, etc.)  
library(readxl) # Import Excel (.xlsx, .xls) files into R  
  
# --- Data Exploration and Summaries ---  
library(DataExplorer) # Automate exploratory data analysis (EDA) and visual summaries  
library(skimr) # Generate concise data summaries with skim()  
library(psych) # Produce descriptive statistics and psychometric analyses  
library(knitr) # Format dynamic tables and reports in Markdown/Quarto  
  
# --- Visualisation ---  
library(ggplot2) # Build customizable, high-quality data visualizations  
library(patchwork) # Combine multiple ggplot2 plots into unified layouts  
library(corrplot) # Visualize correlation matrices with heatmaps and plots  
library(GGally) # Extend ggplot2 with correlation plots, pairwise plots, ggpairs()  
library(PerformanceAnalytics) # Create financial-style charts (correlations, histograms, scatterplots)  
library(VIM) # Visualize/impute missing values with advanced methods (e.g., matrix plots)  
library(scales)  
  
# --- Data Imputation and Synthesis ---  
library(mice) # Perform multiple imputation by chained equations for missing data  
library(synthpop) # Generate, evaluate, and compare synthetic datasets  
  
# --- Statistical and Information-Theoretic Measures ---  
library(transport) # Compute optimal transport distances (e.g., Wasserstein) for distribution comparison  
library(infotheo) # Calculate information-theoretic metrics (entropy, mutual information)  
  
# --- Modeling and Machine Learning ---  
library(caret) # Streamline machine learning workflows (training, tuning, evaluation)  
library(xgboost) # Implement gradient boosting for classification/regression  
library(SHAPforxgboost) # Explain XGBoost predictions using SHAP values  
library(FNN) # Perform fast k-nearest neighbor searches and distance calculations  
  
# Set up parallel processing  
num\_cores <- detectCores() - 1 # Use one less than the total number of cores  
cl <- makeCluster(num\_cores)  
registerDoParallel(cl)

## 1 Introduction

This project evaluates the quality of synthetic datasets derived from the Heart Failure Clinical Records data (299 patients, 13 variables). Our goal is to determine how well different generation strategies reproduce the statistical properties and analytic value of the original data while protecting privacy. We generate synthetic data using four approaches: (1) parametric imputation (MICE), (2) non-parametric imputation (CART via MICE), (3) distribution-driven synthesis (synthpop), and (4) metadata-guided rules. We then compare each synthetic dataset with the real data across three dimensions:

* Fidelity – univariate and multivariate similarity (distributions, ranges, correlations), histogram similarity, and mutual information.
* Utility – model transportability using XGBoost (TRTR vs. TSTR), feature-importance agreement, and SHAP-based behaviour.
* Privacy / Disclosure risk – exact record matches, neighbour-proximity checks, and membership-inference sensitivity.

Results are presented through aligned tables and plots (structure checks, categorical level retention, density overlays, correlation matrices, SHAP summaries) with concise scores for each method. The report concludes with a comparative summary to guide method selection given different fidelity–utility–privacy trade-offs.

## 2 Heart Failure Clinical Records Dataset and Initial Exploration

The **Heart Failure Clinical Records dataset**, sourced from the [UCI Machine Learning Repository](https://archive.ics.uci.edu/dataset/519/heart+failure+clinical+records), contains records of **299 patients** with heart failure collected during their follow-up period. Each record includes **13 clinical features** covering demographics, clinical conditions, and laboratory measures:

| Variable | Description | Unit / Levels |
| --- | --- | --- |
| **age** | Age of the patient | Years |
| **anaemia** | Reduction in red blood cells | No / Yes |
| **creatinine\_phosphokinase** | Enzyme level in the blood | mcg/L |
| **diabetes** | Diabetes status | No / Yes |
| **ejection\_fraction** | % of blood leaving the heart with each contraction | Percentage (%) |
| **platelets** | Platelet count | Kiloplatelets/mL |
| **serum\_creatinine** | Serum creatinine level | mg/dL |
| **serum\_sodium** | Serum sodium level | mEq/L |
| **sex** | Gender of the patient | Female / Male |
| **smoking** | Smoking status | No / Yes |
| **hypertension** | Hypertension status | No / Yes |
| **deceased** | Survival status during follow-up | No / Yes |
| **follow\_up** | Duration of the follow-up period | Days |

A preview of the first 10 rows of the dataset is shown below.

# Set seed for reproducibility  
set.seed(123)  
  
# Load the heart failure dataset from the local directory  
heart\_failure <- read.csv(here("data", "heart\_failure.csv"))  
  
# Let's preview the heart failure dataset  
knitr::kable(  
 head(heart\_failure, 10),  
 caption = "Preview of the first 10 rows of the Heart Failure dataset",  
 align = rep("l", ncol(heart\_failure)) # left align all columns  
)

Preview of the first 10 rows of the Heart Failure dataset

| age | anaemia | creatinine\_phosphokinase | diabetes | ejection\_fraction | platelets | serum\_creatinine | serum\_sodium | sex | smoking | hypertension | deceased | follow\_up |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 75 | No | 582 | No | 20 | 265000 | 1.9 | 130 | Male | No | Yes | Yes | 4 |
| 55 | No | 7861 | No | NA | 263358 | 1.1 | NA | Male | No | No | Yes | 6 |
| 65 | No | NA | No | 20 | 162000 | NA | 129 | Male | Yes | No | Yes | 7 |
| NA | Yes | 111 | No | NA | NA | 1.9 | NA | Male | No | No | Yes | 7 |
| 65 | Yes | 160 | Yes | 20 | 327000 | 2.7 | 116 | Female | No | No | Yes | 8 |
| 90 | Yes | 47 | No | 40 | 204000 | NA | 132 | Male | Yes | Yes | Yes | 8 |
| 75 | Yes | 246 | No | 15 | NA | 1.2 | NA | Male | No | No | Yes | 10 |
| 60 | Yes | 315 | Yes | 60 | 454000 | 1.1 | 131 | Male | Yes | No | Yes | 10 |
| NA | No | NA | No | 65 | 263358 | NA | NA | Female | No | No | Yes | 10 |
| 80 | Yes | 123 | No | 35 | 388000 | 9.4 | 133 | Male | Yes | Yes | Yes | 10 |

Here’s the preview of the structure of the heart failure dataset.

# Display the structure of the dataset  
data.frame(  
 Variable = names(heart\_failure),  
 Type = sapply(heart\_failure, \(x) paste(class(x), collapse = ", ")),  
 Values = sapply(heart\_failure, \(x) paste(head(unique(x), 10), collapse = ", "))  
) |>  
 knitr::kable(  
 caption = "Structure of the Heart Failure Dataset: Variables, Types, and Values",  
 align = c("l","l","l")  
 )

Structure of the Heart Failure Dataset: Variables, Types, and Values

|  | Variable | Type | Values |
| --- | --- | --- | --- |
| age | age | numeric | 75, 55, 65, NA, 90, 60, 80, 62, 45, 50 |
| anaemia | anaemia | character | No, Yes |
| creatinine\_phosphokinase | creatinine\_phosphokinase | integer | 582, 7861, NA, 111, 160, 47, 246, 315, 123, 81 |
| diabetes | diabetes | character | No, Yes |
| ejection\_fraction | ejection\_fraction | integer | 20, NA, 40, 15, 60, 65, 35, 38, 25, 30 |
| platelets | platelets | numeric | 265000, 263358.03, 162000, NA, 327000, 204000, 454000, 388000, 368000, 253000 |
| serum\_creatinine | serum\_creatinine | numeric | 1.9, 1.1, NA, 2.7, 1.2, 9.4, 4, 0.9, 1, 1.3 |
| serum\_sodium | serum\_sodium | integer | 130, NA, 129, 116, 132, 131, 133, 140, 137, 138 |
| sex | sex | character | Male, Female |
| smoking | smoking | character | No, Yes |
| hypertension | hypertension | character | Yes, No |
| deceased | deceased | character | Yes, No |
| follow\_up | follow\_up | integer | 4, 6, 7, 8, 10, 11, 12, 13, 14, 15 |

Let’s perform some basic data cleaning and preprocessing steps. Convert the following columns to factors: sex, anaemia, diabetes, smoking, hypertension and deceased. Here’s the new structure of the dataset after the basic cleaning.

# Convert the specified columns to factors  
heart\_failure$sex <- as.factor(heart\_failure$sex)  
heart\_failure$anaemia <- as.factor(heart\_failure$anaemia)  
heart\_failure$diabetes <- as.factor(heart\_failure$diabetes)  
heart\_failure$smoking <- as.factor(heart\_failure$smoking)  
heart\_failure$hypertension <- as.factor(heart\_failure$hypertension)  
heart\_failure$deceased <- as.factor(heart\_failure$deceased)

Here are the summary statistics of the heart failure dataset.

# Generate descriptive statistics and display as a single table  
describe(heart\_failure) |>  
 kable(  
 caption = "Summary Statistics of the Heart Failure Dataset",  
 digits = 2, # round to 2 decimal places  
 align = "lrrrrrrrrrr" # left for variable, right for numbers  
 )

Summary Statistics of the Heart Failure Dataset

|  | vars | n | mean | sd | median | trimmed | mad | min | max | range | skew | kurtosis | se |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| age | 1 | 284 | 60.80 | 12.08 | 60.0 | 60.15 | 14.83 | 40.0 | 95.0 | 55.0 | 0.42 | -0.28 | 0.72 |
| anaemia\* | 2 | 299 | 1.43 | 0.50 | 1.0 | 1.41 | 0.00 | 1.0 | 2.0 | 1.0 | 0.28 | -1.93 | 0.03 |
| creatinine\_phosphokinase | 3 | 285 | 578.81 | 981.99 | 249.0 | 358.95 | 268.35 | 23.0 | 7861.0 | 7838.0 | 4.45 | 24.52 | 58.17 |
| diabetes\* | 4 | 299 | 1.42 | 0.49 | 1.0 | 1.40 | 0.00 | 1.0 | 2.0 | 1.0 | 0.33 | -1.90 | 0.03 |
| ejection\_fraction | 5 | 287 | 38.00 | 11.92 | 38.0 | 37.31 | 11.86 | 14.0 | 80.0 | 66.0 | 0.58 | 0.02 | 0.70 |
| platelets | 6 | 285 | 265163.83 | 97508.63 | 263358.0 | 259212.20 | 65765.22 | 25100.0 | 850000.0 | 824900.0 | 1.43 | 6.24 | 5775.91 |
| serum\_creatinine | 7 | 281 | 1.39 | 1.05 | 1.1 | 1.18 | 0.30 | 0.5 | 9.4 | 8.9 | 4.44 | 25.03 | 0.06 |
| serum\_sodium | 8 | 279 | 136.59 | 4.49 | 137.0 | 136.80 | 4.45 | 113.0 | 148.0 | 35.0 | -1.05 | 3.88 | 0.27 |
| sex\* | 9 | 299 | 1.65 | 0.48 | 2.0 | 1.68 | 0.00 | 1.0 | 2.0 | 1.0 | -0.62 | -1.62 | 0.03 |
| smoking\* | 10 | 299 | 1.32 | 0.47 | 1.0 | 1.28 | 0.00 | 1.0 | 2.0 | 1.0 | 0.76 | -1.42 | 0.03 |
| hypertension\* | 11 | 299 | 1.35 | 0.48 | 1.0 | 1.32 | 0.00 | 1.0 | 2.0 | 1.0 | 0.62 | -1.62 | 0.03 |
| deceased\* | 12 | 299 | 1.32 | 0.47 | 1.0 | 1.28 | 0.00 | 1.0 | 2.0 | 1.0 | 0.76 | -1.42 | 0.03 |
| follow\_up | 13 | 299 | 130.26 | 77.61 | 115.0 | 129.28 | 105.26 | 4.0 | 285.0 | 281.0 | 0.13 | -1.22 | 4.49 |

## 3 Generating Synthetic Data

The process of generating synthetic data is essential for tasks such as privacy preservation, testing machine learning models, and conducting simulations. In this section, we employ various techniques to generate synthetic data based on the real heart failure dataset. Each method serves a different purpose and offers distinct advantages, depending on the use case and the type of missing data or privacy concerns.

We explore four key approaches for generating synthetic data:

### 3.1 Parametric Imputation Using MICE

This method uses the Multivariate Imputation by Chained Equations (MICE) framework, with parametric imputation based on a normal distribution. MICE allows for handling missing data through multiple imputations and is widely used in data science and healthcare for its flexibility and statistical rigor. Here’s the preview of the first 10 rows of the synthetic data generated.

# Function to compute the 5th and 95th percentiles for numeric columns  
compute\_percentiles <- function(data, lower\_percentile = 0.05, upper\_percentile = 0.95) {  
 percentiles <- lapply(data, function(column) {  
 if (is.numeric(column)) {  
 lower <- quantile(column, probs = lower\_percentile, na.rm = TRUE)  
 upper <- quantile(column, probs = upper\_percentile, na.rm = TRUE)  
 return(c(lower, upper))  
 }  
 return(c(NA, NA))  
 })  
 names(percentiles) <- names(data)  
 return(percentiles)  
}  
  
# Function to scale numeric values based on the 5th and 95th percentiles  
scale\_to\_percentiles <- function(data, percentiles) {  
 scaled\_data <- data  
 for (col in colnames(data)) {  
 if (is.numeric(data[[col]])) {  
 # Apply scaling only if percentiles are available for this column  
 if (!is.na(percentiles[[col]][1]) && !is.na(percentiles[[col]][2])) {  
 min\_val <- percentiles[[col]][1]  
 max\_val <- percentiles[[col]][2]  
 # Clip values to lie within the 5th and 95th percentile range  
 scaled\_data[[col]] <- pmin(pmax(data[[col]], min\_val), max\_val)  
 }  
 }  
 }  
 return(scaled\_data)  
}  
  
# Function to generate missingness in a dataset based on the real data's pattern  
generate\_missingness\_based\_on\_real <- function(real\_data, synthetic\_data) {  
 # Ensure the synthetic dataset has the same structure as the real  
 if (!all(colnames(real\_data) == colnames(synthetic\_data))) {  
 stop("Column names in real and synthetic data must match")  
 }  
   
 # Copy the real missingness pattern to the synthetic dataset  
 synthetic\_data\_with\_missingness <- synthetic\_data  
 for (col in colnames(real\_data)) {  
 # Apply missingness where it was present in the real data  
 missing\_indices <- is.na(real\_data[[col]])  
 synthetic\_data\_with\_missingness[[col]][missing\_indices] <- NA  
 }  
   
 return(synthetic\_data\_with\_missingness)  
}  
  
# Compute the 5th and 95th percentiles for the real data  
percentiles <- compute\_percentiles(heart\_failure)  
  
# Generate synthetic data using MICE  
where <- make.where(heart\_failure, "all")  
method <- make.method(heart\_failure, where = where)  
method[method == "pmm"] <- "norm"  
  
# Perform multiple imputation (10 datasets)  
syn\_param <- mice(heart\_failure, m = 10, maxit = 1, method = method, where = where, printFlag = FALSE)  
  
# Extract the first synthetic dataset  
syn\_data\_1 <- complete(syn\_param, 1)  
  
# Scale synthetic data to the 5th and 95th percentile ranges  
syn\_data\_1 <- scale\_to\_percentiles(syn\_data\_1, percentiles)  
  
# Apply the real missingness pattern to the synthetic dataset  
syn\_data\_1 <- generate\_missingness\_based\_on\_real(heart\_failure, syn\_data\_1)  
  
# Round off all numeric columns to 0 decimal places  
syn\_data\_1 <- syn\_data\_1 %>%  
 mutate(across(where(is.numeric), round, digits = 0))  
  
# Add a prefix 'synth\_' to all synthetic dataset variable names  
colnames(syn\_data\_1) <- paste("synth", colnames(syn\_data\_1), sep = "\_")  
  
# Display the structure of the dataset  
data.frame(  
 Variable = names(syn\_data\_1),  
 Type = sapply(syn\_data\_1, \(x) paste(class(x), collapse = ", ")),  
 Values = sapply(syn\_data\_1, \(x) paste(head(unique(x), 10), collapse = ", "))  
) |>  
 knitr::kable(  
 caption = "Structure of the Heart Failure Dataset: Variables, Types, and Values",  
 align = c("l","l","l")  
 )

Structure of the Heart Failure Dataset: Variables, Types, and Values

|  | Variable | Type | Values |
| --- | --- | --- | --- |
| synth\_age | synth\_age | numeric | 56, 42, 57, NA, 69, 53, 72, 59, 82, 73 |
| synth\_anaemia | synth\_anaemia | factor | No, Yes |
| synth\_creatinine\_phosphokinase | synth\_creatinine\_phosphokinase | numeric | 184, 1540, NA, 559, 941, 1926, 570, 1380, 59, 1531 |
| synth\_diabetes | synth\_diabetes | factor | No, Yes |
| synth\_ejection\_fraction | synth\_ejection\_fraction | numeric | 40, NA, 52, 48, 49, 27, 33, 51, 30, 39 |
| synth\_platelets | synth\_platelets | numeric | 284572, 235177, 175379, NA, 352540, 132200, 349886, 421200, 246142, 265262 |
| synth\_serum\_creatinine | synth\_serum\_creatinine | numeric | 2, 1, NA, 3 |
| synth\_serum\_sodium | synth\_serum\_sodium | numeric | 138, NA, 131, 130, 137, 135, 142, 134, 141, 144 |
| synth\_sex | synth\_sex | factor | Male, Female |
| synth\_smoking | synth\_smoking | factor | Yes, No |
| synth\_hypertension | synth\_hypertension | factor | Yes, No |
| synth\_deceased | synth\_deceased | factor | Yes, No |
| synth\_follow\_up | synth\_follow\_up | numeric | 90, 13, 88, 40, 31, 129, 91, 51, 29, 118 |

### 3.2 Non-Parametric Imputation Using CART

The Classification and Regression Trees (CART) method, a non-parametric approach, is useful for imputation when the relationships between variables are complex or non-linear. By leveraging decision trees, CART is capable of capturing intricate patterns in the data without making strong assumptions about the underlying distributions. Here’s the preview of the first 10 rows of the synthetic data generated.

# Compute the 5th and 95th percentiles for the real data  
percentiles <- compute\_percentiles(heart\_failure)  
  
# Generate synthetic data using CART imputation with MICE  
where <- make.where(heart\_failure, "all")  
method <- make.method(heart\_failure, where = where)  
method[method == "pmm"] <- "cart" # Set the method to "cart" for non-parametric imputation  
  
# Perform multiple imputation (10 datasets)  
syn\_cart <- mice(heart\_failure, m = 10, maxit = 1, method = method, where = where, printFlag = FALSE)  
  
# Extract the first synthetic dataset  
syn\_cart\_1 <- complete(syn\_cart, 1)  
  
# Scale synthetic data to the 5th and 95th percentile ranges  
syn\_cart\_1 <- scale\_to\_percentiles(syn\_cart\_1, percentiles)  
  
# Apply the real missingness pattern to the synthetic dataset  
syn\_cart\_1 <- generate\_missingness\_based\_on\_real(heart\_failure, syn\_cart\_1)  
  
# Round off all numeric columns to 0 decimal places  
syn\_cart\_1 <- syn\_cart\_1 %>%  
 mutate(across(where(is.numeric), round, digits = 0))  
  
# Add a prefix 'synth\_' to all synthetic dataset variable names  
colnames(syn\_cart\_1) <- paste("synth", colnames(syn\_cart\_1), sep = "\_")  
  
# Display the structure of the dataset  
data.frame(  
 Variable = names(syn\_cart\_1),  
 Type = sapply(syn\_cart\_1, \(x) paste(class(x), collapse = ", ")),  
 Values = sapply(syn\_cart\_1, \(x) paste(head(unique(x), 10), collapse = ", "))  
) |>  
 knitr::kable(  
 caption = "Structure of the Heart Failure Dataset: Variables, Types, and Values",  
 align = c("l","l","l")  
 )

Structure of the Heart Failure Dataset: Variables, Types, and Values

|  | Variable | Type | Values |
| --- | --- | --- | --- |
| synth\_age | synth\_age | numeric | 60, 50, 65, NA, 82, 53, 75, 69, 70, 45 |
| synth\_anaemia | synth\_anaemia | factor | No, Yes |
| synth\_creatinine\_phosphokinase | synth\_creatinine\_phosphokinase | numeric | 582, 161, NA, 203, 112, 371, 246, 60, 59, 2221 |
| synth\_diabetes | synth\_diabetes | factor | No, Yes |
| synth\_ejection\_fraction | synth\_ejection\_fraction | numeric | 30, NA, 38, 25, 50, 60, 35, 20, 45, 40 |
| synth\_platelets | synth\_platelets | numeric | 132200, 263358, NA, 421200, 351000, 219000, 327000, 213000, 395000, 282000 |
| synth\_serum\_creatinine | synth\_serum\_creatinine | numeric | 3, 1, NA, 2 |
| synth\_serum\_sodium | synth\_serum\_sodium | numeric | 133, NA, 130, 140, 138, 139, 137, 136, 134, 135 |
| synth\_sex | synth\_sex | factor | Female, Male |
| synth\_smoking | synth\_smoking | factor | Yes, No |
| synth\_hypertension | synth\_hypertension | factor | No, Yes |
| synth\_deceased | synth\_deceased | factor | Yes, No |
| synth\_follow\_up | synth\_follow\_up | numeric | 59, 30, 154, 24, 66, 13, 28, 65, 129, 72 |

### 3.3 Generating Synthetic Data Using Synthpop

The Synthpop package is designed for creating synthetic data based on the distribution of the real dataset. It is particularly useful for privacy-preserving data sharing, where the aim is to create a synthetic dataset that mimics the real data without exposing sensitive information. Here’s the preview of the first 10 rows of the synthetic data generated.

# Compute the 5th and 95th percentiles for the real data  
percentiles <- compute\_percentiles(heart\_failure)  
  
# Generate low-fidelity synthetic data using random sampling  
syn\_data\_low\_fidelity <- syn(heart\_failure, method = "sample", seed = 123)

Synthesis  
-----------  
 age anaemia creatinine\_phosphokinase diabetes ejection\_fraction platelets serum\_creatinine serum\_sodium sex smoking  
 hypertension deceased follow\_up

# Convert the synthetic dataset to a data frame  
syn\_data\_low\_fidelity\_synthpop <- syn\_data\_low\_fidelity$syn  
  
# Scale the synthetic data to the 5th and 95th percentile ranges  
syn\_data\_low\_fidelity\_synthpop <- scale\_to\_percentiles(syn\_data\_low\_fidelity\_synthpop, percentiles)  
  
# Apply the real missingness pattern to the synthetic dataset  
syn\_data\_low\_fidelity\_synthpop <- generate\_missingness\_based\_on\_real(heart\_failure, syn\_data\_low\_fidelity\_synthpop)  
  
# Round off all numeric columns to 0 decimal places  
syn\_data\_low\_fidelity\_synthpop <- syn\_data\_low\_fidelity\_synthpop %>%  
 mutate(across(where(is.numeric), round, digits = 0))  
  
# Add a prefix 'synth\_' to all synthetic dataset variable names  
colnames(syn\_data\_low\_fidelity\_synthpop) <- paste("synth", colnames(syn\_data\_low\_fidelity\_synthpop), sep = "\_")  
  
# Display the structure of the dataset  
data.frame(  
 Variable = names(syn\_data\_low\_fidelity\_synthpop),  
 Type = sapply(syn\_data\_low\_fidelity\_synthpop, \(x) paste(class(x), collapse = ", ")),  
 Values = sapply(syn\_data\_low\_fidelity\_synthpop, \(x) paste(head(unique(x), 10), collapse = ", "))  
) |>  
 knitr::kable(  
 caption = "Structure of the Heart Failure Dataset: Variables, Types, and Values",  
 align = c("l","l","l")  
 )

Structure of the Heart Failure Dataset: Variables, Types, and Values

|  | Variable | Type | Values |
| --- | --- | --- | --- |
| synth\_age | synth\_age | numeric | 63, 50, 45, NA, 73, 57, 70, 52, 65, 80 |
| synth\_anaemia | synth\_anaemia | factor | Yes, No |
| synth\_creatinine\_phosphokinase | synth\_creatinine\_phosphokinase | numeric | 427, 335, NA, 249, 64, 1610, 2221, 224, 910, 260 |
| synth\_diabetes | synth\_diabetes | factor | No, Yes |
| synth\_ejection\_fraction | synth\_ejection\_fraction | numeric | 25, NA, 20, 40, 60, 30, 35, 38, 45, 50 |
| synth\_platelets | synth\_platelets | numeric | NA, 229000, 385000, 277000, 268000, 271000, 274000, 189000, 236000, 329000 |
| synth\_serum\_creatinine | synth\_serum\_creatinine | numeric | 1, NA, 2, 3 |
| synth\_serum\_sodium | synth\_serum\_sodium | numeric | 142, NA, 139, 136, 137, 134, 140, 130, 144, 135 |
| synth\_sex | synth\_sex | factor | Male, Female |
| synth\_smoking | synth\_smoking | factor | Yes, No |
| synth\_hypertension | synth\_hypertension | factor | Yes, No |
| synth\_deceased | synth\_deceased | factor | Yes, No |
| synth\_follow\_up | synth\_follow\_up | numeric | 250, 13, 91, 121, 20, 113, 186, 85, 74, 50 |

### 3.4 Generating Synthetic Data Using Metadada

This method uses a data dictionary (metadata) to drive the generation of synthetic data. The metadata defines the structure and types of variables, which ensures that the generated synthetic data conforms to expected formats, such as ranges for continuous variables and categories for binary or categorical variables. Here’s the preview of the first 10 rows of the synthetic data generated.

# Compute the 5th and 95th percentiles for the real data  
percentiles <- compute\_percentiles(heart\_failure)  
  
# Generate synthetic data using metadata  
  
# Load the metadata from the Excel file  
heart\_failure\_metadata <- read\_excel(here("data", "heart\_failure\_metadata.xlsx"))  
  
# Create an empty data frame based on the metadata structure  
n\_rows <- nrow(heart\_failure) # Number of synthetic records to generate  
variable\_names <- heart\_failure\_metadata$`Variable Name`  
data\_types <- heart\_failure\_metadata$Type  
  
syn\_data\_metadata <- data.frame(matrix(ncol = length(variable\_names), nrow = n\_rows))  
colnames(syn\_data\_metadata) <- variable\_names  
  
# Function to generate data based on metadata  
generate\_data <- function(variable, real\_data, type, n) {  
 if (type == "Continuous" || type == "Integer") {  
 if (is.numeric(real\_data[[variable]])) {  
 # Compute 5th and 95th percentiles for numeric variables  
 p5 <- quantile(real\_data[[variable]], probs = 0.05, na.rm = TRUE)  
 p95 <- quantile(real\_data[[variable]], probs = 0.95, na.rm = TRUE)  
   
 if (type == "Continuous") {  
 return(runif(n, min = p5, max = p95))  
 } else if (type == "Integer") {  
 return(sample(floor(p5):ceiling(p95), n, replace = TRUE))  
 }  
 }  
 } else if (is.factor(real\_data[[variable]])) {  
 return(sample(levels(real\_data[[variable]]), n, replace = TRUE))  
 } else if (variable %in% c("anaemia", "diabetes", "hypertension", "smoking", "sex")) {  
 return(sample(unique(real\_data[[variable]]), n, replace = TRUE))  
 } else {  
 return(rep(NA, n))  
 }  
}  
  
# Generate synthetic data based on the metadata  
for (i in seq\_along(variable\_names)) {  
 syn\_data\_metadata[[i]] <- generate\_data(variable\_names[i], heart\_failure, data\_types[i], n\_rows)  
}  
  
# Scale the synthetic data to the 5th and 95th percentile ranges  
syn\_data\_metadata <- scale\_to\_percentiles(syn\_data\_metadata, percentiles)  
  
# Apply the real missingness pattern to the synthetic dataset  
syn\_data\_metadata <- generate\_missingness\_based\_on\_real(heart\_failure, syn\_data\_metadata)  
  
# Round off all numeric columns to 0 decimal places  
syn\_data\_metadata <- syn\_data\_metadata %>%  
 mutate(across(where(is.numeric), round, digits = 0))  
  
# Add a prefix 'synth\_' to all synthetic dataset variable names  
colnames(syn\_data\_metadata) <- paste("synth", colnames(syn\_data\_metadata), sep = "\_")  
  
# Display the structure of the dataset  
data.frame(  
 Variable = names(syn\_data\_metadata),  
 Type = sapply(syn\_data\_metadata, \(x) paste(class(x), collapse = ", ")),  
 Values = sapply(syn\_data\_metadata, \(x) paste(head(unique(x), 10), collapse = ", "))  
) |>  
 knitr::kable(  
 caption = "Structure of the Heart Failure Dataset: Variables, Types, and Values",  
 align = c("l","l","l")  
 )

Structure of the Heart Failure Dataset: Variables, Types, and Values

|  | Variable | Type | Values |
| --- | --- | --- | --- |
| synth\_age | synth\_age | numeric | 45, 51, 82, NA, 80, 66, 59, 71, 47, 57 |
| synth\_anaemia | synth\_anaemia | character | Yes, No |
| synth\_creatinine\_phosphokinase | synth\_creatinine\_phosphokinase | numeric | 180, 1663, NA, 191, 571, 1228, 395, 1553, 812, 335 |
| synth\_diabetes | synth\_diabetes | character | Yes, No |
| synth\_ejection\_fraction | synth\_ejection\_fraction | numeric | 31, NA, 58, 60, 29, 26, 37, 33, 24, 28 |
| synth\_platelets | synth\_platelets | numeric | 210137, 203132, 257570, NA, 395609, 331643, 171789, 279683, 413390, 185403 |
| synth\_serum\_creatinine | synth\_serum\_creatinine | numeric | 3, 1, NA, 2 |
| synth\_serum\_sodium | synth\_serum\_sodium | numeric | 136, NA, 138, 131, 130, 135, 134, 137, 143, 142 |
| synth\_sex | synth\_sex | character | Female, Male |
| synth\_smoking | synth\_smoking | character | Yes, No |
| synth\_hypertension | synth\_hypertension | character | Yes, No |
| synth\_deceased | synth\_deceased | character | No, Yes |
| synth\_follow\_up | synth\_follow\_up | numeric | 152, 231, 199, 46, 188, 117, 57, 248, 90, 237 |

## 4 Synthetic Data Identification

In this section, we generate tables to identify which columns in the synthetic datasets are synthetic (prefixed with “synth\_”) and which remain unchanged from the real dataset. This helps ensure clarity on which variables have been altered during the synthetic data generation process.

For each dataset, we produce a table displaying the column names and whether the column is synthetic. A column is marked “Yes” if it was generated synthetically and “No” if it was retained from the real data.

# Function to create column info with dataset name  
create\_column\_info <- function(dataset, dataset\_name) {  
 data.frame(  
 column\_name = colnames(dataset),  
 dataset = dataset\_name,  
 is\_synthetic = ifelse(grepl("^synth\_", colnames(dataset)), "Yes", "No"),  
 stringsAsFactors = FALSE  
 )  
}  
  
# Collect column info for all datasets  
all\_column\_info <- bind\_rows(  
 create\_column\_info(syn\_data\_1, "Parametric MICE"),  
 create\_column\_info(syn\_cart\_1, "CART Imputation"),  
 create\_column\_info(syn\_data\_low\_fidelity\_synthpop, "Synthpop (Low Fidelity)"),  
 create\_column\_info(syn\_data\_metadata, "Metadata-Based")  
)  
  
# Reshape into wide format so variables appear once  
wide\_column\_info <- all\_column\_info %>%  
 pivot\_wider(  
 names\_from = dataset,  
 values\_from = is\_synthetic  
 )  
  
# Display the combined wide table  
kable(  
 wide\_column\_info,  
 caption = "Synthetic Indicators for Each Variable Across Datasets"  
)

Synthetic Indicators for Each Variable Across Datasets

| column\_name | Parametric MICE | CART Imputation | Synthpop (Low Fidelity) | Metadata-Based |
| --- | --- | --- | --- | --- |
| synth\_age | Yes | Yes | Yes | Yes |
| synth\_anaemia | Yes | Yes | Yes | Yes |
| synth\_creatinine\_phosphokinase | Yes | Yes | Yes | Yes |
| synth\_diabetes | Yes | Yes | Yes | Yes |
| synth\_ejection\_fraction | Yes | Yes | Yes | Yes |
| synth\_platelets | Yes | Yes | Yes | Yes |
| synth\_serum\_creatinine | Yes | Yes | Yes | Yes |
| synth\_serum\_sodium | Yes | Yes | Yes | Yes |
| synth\_sex | Yes | Yes | Yes | Yes |
| synth\_smoking | Yes | Yes | Yes | Yes |
| synth\_hypertension | Yes | Yes | Yes | Yes |
| synth\_deceased | Yes | Yes | Yes | Yes |
| synth\_follow\_up | Yes | Yes | Yes | Yes |

## 5 Synthetic Dataset Structure

### 5.1 Dataset Size and Structure Comparison

This step evaluates whether the synthetic datasets match the real dataset in terms of the number of rows, columns, and feature types. Consistency in these dimensions is essential for the synthetic data to be a reliable stand-in for the real.

* **Number of Rows**: Indicates if the synthetic data captures the same number of records as the real. Any mismatch can impact comparability.
* **Number of Columns**: Ensures all variables are retained. Missing or extra columns suggest structural issues.
* **Feature Types**: Confirms that categorical, numerical, and other variable types remain unchanged, preserving analytical consistency.

# Function to extract structure info  
get\_structure\_info <- function(real\_data, synthetic\_data, dataset\_name) {  
 tibble(  
 Dataset = dataset\_name,  
 Rows\_Real = nrow(real\_data),  
 Cols\_Real = ncol(real\_data),  
 Rows\_Synthetic = nrow(synthetic\_data),  
 Cols\_Synthetic = ncol(synthetic\_data),  
 Structure\_Match = ifelse(  
 nrow(real\_data) == nrow(synthetic\_data) & ncol(real\_data) == ncol(synthetic\_data),  
 "✅ Yes", "❌ No"  
 )  
 )  
}  
  
# Collect results  
structure\_comparison <- bind\_rows(  
 get\_structure\_info(heart\_failure, syn\_data\_1, "Parametric MICE"),  
 get\_structure\_info(heart\_failure, syn\_cart\_1, "CART Imputation"),  
 get\_structure\_info(heart\_failure, syn\_data\_low\_fidelity\_synthpop, "Synthpop (Low Fidelity)"),  
 get\_structure\_info(heart\_failure, syn\_data\_metadata, "Metadata-Based")  
)  
  
# Display as a neat table  
kable(  
 structure\_comparison,  
 caption = "Comparison of Real vs Synthetic Dataset Sizes and Structures"  
)

Comparison of Real vs Synthetic Dataset Sizes and Structures

| Dataset | Rows\_Real | Cols\_Real | Rows\_Synthetic | Cols\_Synthetic | Structure\_Match |
| --- | --- | --- | --- | --- | --- |
| Parametric MICE | 299 | 13 | 299 | 13 | ✅ Yes |
| CART Imputation | 299 | 13 | 299 | 13 | ✅ Yes |
| Synthpop (Low Fidelity) | 299 | 13 | 299 | 13 | ✅ Yes |
| Metadata-Based | 299 | 13 | 299 | 13 | ✅ Yes |

### 5.2 Categorical Variables: All Levels Maintained and Comparison of Distribution

This section evaluates how well the synthetic datasets replicate the categorical variables in the real dataset, focusing on the preservation of all levels and their distributions.

* **Level Matching**: For each categorical variable, the synthetic dataset should retain all levels (categories) present in the real dataset. Missing levels suggest that the synthetic data may be incomplete, while additional levels could indicate errors in data generation.
* **Distribution Comparison**: The frequency of each categorical level in the synthetic data should closely mirror that of the real dataset. Significant deviations in category frequencies suggest that the synthetic data does not fully capture the true distribution of the categorical features.

# Function to collect categorical level info  
get\_cat\_levels <- function(real\_data, synthetic\_data, dataset\_name) {  
 cat\_vars <- colnames(real\_data)[sapply(real\_data, is.factor)]  
   
 do.call(rbind, lapply(cat\_vars, function(var) {  
 levels\_real <- unique(real\_data[[var]])  
 levels\_synth <- unique(synthetic\_data[[paste("synth", var, sep = "\_")]])  
   
 data.frame(  
 Variable = var,  
 Dataset = dataset\_name,  
 Real\_Levels = paste(levels\_real, collapse = ", "),  
 Synthetic\_Levels = paste(levels\_synth, collapse = ", "),  
 Match = ifelse(all(levels\_real %in% levels\_synth), "✅ Yes", "❌ No"),  
 stringsAsFactors = FALSE  
 )  
 }))  
}  
  
# Collect results  
cat\_comparison <- bind\_rows(  
 get\_cat\_levels(heart\_failure, syn\_data\_1, "Parametric MICE"),  
 get\_cat\_levels(heart\_failure, syn\_cart\_1, "CART Imputation"),  
 get\_cat\_levels(heart\_failure, syn\_data\_low\_fidelity\_synthpop, "Synthpop (Low Fidelity)"),  
 get\_cat\_levels(heart\_failure, syn\_data\_metadata, "Metadata-Based")  
)  
  
# Display table  
kable(cat\_comparison,  
 caption = "Comparison of Categorical Variable Levels in Real vs Synthetic Datasets")

Comparison of Categorical Variable Levels in Real vs Synthetic Datasets

| Variable | Dataset | Real\_Levels | Synthetic\_Levels | Match |
| --- | --- | --- | --- | --- |
| anaemia | Parametric MICE | No, Yes | No, Yes | ✅ Yes |
| diabetes | Parametric MICE | No, Yes | No, Yes | ✅ Yes |
| sex | Parametric MICE | Male, Female | Male, Female | ✅ Yes |
| smoking | Parametric MICE | No, Yes | Yes, No | ✅ Yes |
| hypertension | Parametric MICE | Yes, No | Yes, No | ✅ Yes |
| deceased | Parametric MICE | Yes, No | Yes, No | ✅ Yes |
| anaemia | CART Imputation | No, Yes | No, Yes | ✅ Yes |
| diabetes | CART Imputation | No, Yes | No, Yes | ✅ Yes |
| sex | CART Imputation | Male, Female | Female, Male | ✅ Yes |
| smoking | CART Imputation | No, Yes | Yes, No | ✅ Yes |
| hypertension | CART Imputation | Yes, No | No, Yes | ✅ Yes |
| deceased | CART Imputation | Yes, No | Yes, No | ✅ Yes |
| anaemia | Synthpop (Low Fidelity) | No, Yes | Yes, No | ✅ Yes |
| diabetes | Synthpop (Low Fidelity) | No, Yes | No, Yes | ✅ Yes |
| sex | Synthpop (Low Fidelity) | Male, Female | Male, Female | ✅ Yes |
| smoking | Synthpop (Low Fidelity) | No, Yes | Yes, No | ✅ Yes |
| hypertension | Synthpop (Low Fidelity) | Yes, No | Yes, No | ✅ Yes |
| deceased | Synthpop (Low Fidelity) | Yes, No | Yes, No | ✅ Yes |
| anaemia | Metadata-Based | No, Yes | Yes, No | ✅ Yes |
| diabetes | Metadata-Based | No, Yes | Yes, No | ✅ Yes |
| sex | Metadata-Based | Male, Female | Female, Male | ✅ Yes |
| smoking | Metadata-Based | No, Yes | Yes, No | ✅ Yes |
| hypertension | Metadata-Based | Yes, No | Yes, No | ✅ Yes |
| deceased | Metadata-Based | Yes, No | No, Yes | ✅ Yes |

### 5.3 Numeric Variables: Range and Distribution Comparison

This analysis evaluates how well the numeric variables in the synthetic datasets replicate the real data in terms of range and distribution.

* **Range**: For each numeric variable, the synthetic data should maintain values within the range of the real dataset. Any deviations, especially outliers, could indicate inaccuracies in the synthetic generation process.
* **Distribution Comparison**: Density plots are used to compare the shape of the distributions between the real and synthetic datasets. Ideally, the synthetic data should closely mirror the real distribution, indicating that the underlying data generation model has captured the true variability of the numeric features.

# Helper: compute per-variable stats for a (real, synthetic) pair  
numeric\_summary <- function(real, synth, dataset\_name) {  
 num\_vars <- names(real)[sapply(real, is.numeric)]  
 # real percentiles  
 pctl <- map(num\_vars, ~quantile(real[[.x]], probs = c(.05, .95), na.rm = TRUE)) |>  
 setNames(num\_vars)  
  
 tibble(  
 Variable = num\_vars,  
 Real\_Min = map\_dbl(num\_vars, ~min(real[[.x]], na.rm = TRUE)),  
 Real\_P5 = map\_dbl(num\_vars, ~pctl[[.x]][1]),  
 Real\_P95 = map\_dbl(num\_vars, ~pctl[[.x]][2]),  
 Real\_Max = map\_dbl(num\_vars, ~max(real[[.x]], na.rm = TRUE)),  
 Synth\_Min = map\_dbl(num\_vars, ~min(synth[[paste0("synth\_", .x)]], na.rm = TRUE)),  
 Synth\_Max = map\_dbl(num\_vars, ~max(synth[[paste0("synth\_", .x)]], na.rm = TRUE))  
 ) |>  
 mutate(  
 Within\_P5\_P95 = ifelse(Synth\_Min >= Real\_P5 & Synth\_Max <= Real\_P95, "✅ Yes", "❌ No"),  
 Dataset = dataset\_name,  
 .before = 1  
 )  
}  
  
# Build one continuous table for all synthetic datasets  
num\_table <- bind\_rows(  
 numeric\_summary(heart\_failure, syn\_data\_1, "Parametric MICE"),  
 numeric\_summary(heart\_failure, syn\_cart\_1, "CART Imputation"),  
 numeric\_summary(heart\_failure, syn\_data\_low\_fidelity\_synthpop, "Synthpop (Low Fidelity)"),  
 numeric\_summary(heart\_failure, syn\_data\_metadata, "Metadata-Based")  
)  
  
kable(  
 num\_table,  
 caption = "Numeric variables: real ranges & percentiles vs. synthetic ranges (with P5–P95 containment check)"  
)

Numeric variables: real ranges & percentiles vs. synthetic ranges (with P5–P95 containment check)

| Within\_P5\_P95 | Dataset | Variable | Real\_Min | Real\_P5 | Real\_P95 | Real\_Max | Synth\_Min | Synth\_Max |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ❌ No | Parametric MICE | age | 40.0 | 42.15 | 82.0 | 95.0 | 42 | 82 |
| ❌ No | Parametric MICE | creatinine\_phosphokinase | 23.0 | 59.20 | 2220.8 | 7861.0 | 59 | 2221 |
| ✅ Yes | Parametric MICE | ejection\_fraction | 14.0 | 20.00 | 60.0 | 80.0 | 20 | 60 |
| ✅ Yes | Parametric MICE | platelets | 25100.0 | 132200.00 | 421200.0 | 850000.0 | 132200 | 421200 |
| ❌ No | Parametric MICE | serum\_creatinine | 0.5 | 0.70 | 2.9 | 9.4 | 1 | 3 |
| ✅ Yes | Parametric MICE | serum\_sodium | 113.0 | 130.00 | 144.0 | 148.0 | 130 | 144 |
| ✅ Yes | Parametric MICE | follow\_up | 4.0 | 12.90 | 250.0 | 285.0 | 13 | 250 |
| ❌ No | CART Imputation | age | 40.0 | 42.15 | 82.0 | 95.0 | 42 | 82 |
| ❌ No | CART Imputation | creatinine\_phosphokinase | 23.0 | 59.20 | 2220.8 | 7861.0 | 59 | 2221 |
| ✅ Yes | CART Imputation | ejection\_fraction | 14.0 | 20.00 | 60.0 | 80.0 | 20 | 60 |
| ✅ Yes | CART Imputation | platelets | 25100.0 | 132200.00 | 421200.0 | 850000.0 | 132200 | 421200 |
| ❌ No | CART Imputation | serum\_creatinine | 0.5 | 0.70 | 2.9 | 9.4 | 1 | 3 |
| ✅ Yes | CART Imputation | serum\_sodium | 113.0 | 130.00 | 144.0 | 148.0 | 130 | 144 |
| ✅ Yes | CART Imputation | follow\_up | 4.0 | 12.90 | 250.0 | 285.0 | 13 | 250 |
| ❌ No | Synthpop (Low Fidelity) | age | 40.0 | 42.15 | 82.0 | 95.0 | 42 | 82 |
| ❌ No | Synthpop (Low Fidelity) | creatinine\_phosphokinase | 23.0 | 59.20 | 2220.8 | 7861.0 | 59 | 2221 |
| ✅ Yes | Synthpop (Low Fidelity) | ejection\_fraction | 14.0 | 20.00 | 60.0 | 80.0 | 20 | 60 |
| ✅ Yes | Synthpop (Low Fidelity) | platelets | 25100.0 | 132200.00 | 421200.0 | 850000.0 | 132200 | 421200 |
| ❌ No | Synthpop (Low Fidelity) | serum\_creatinine | 0.5 | 0.70 | 2.9 | 9.4 | 1 | 3 |
| ✅ Yes | Synthpop (Low Fidelity) | serum\_sodium | 113.0 | 130.00 | 144.0 | 148.0 | 130 | 144 |
| ✅ Yes | Synthpop (Low Fidelity) | follow\_up | 4.0 | 12.90 | 250.0 | 285.0 | 13 | 250 |
| ❌ No | Metadata-Based | age | 40.0 | 42.15 | 82.0 | 95.0 | 42 | 82 |
| ✅ Yes | Metadata-Based | creatinine\_phosphokinase | 23.0 | 59.20 | 2220.8 | 7861.0 | 67 | 2216 |
| ✅ Yes | Metadata-Based | ejection\_fraction | 14.0 | 20.00 | 60.0 | 80.0 | 20 | 60 |
| ✅ Yes | Metadata-Based | platelets | 25100.0 | 132200.00 | 421200.0 | 850000.0 | 134054 | 421118 |
| ❌ No | Metadata-Based | serum\_creatinine | 0.5 | 0.70 | 2.9 | 9.4 | 1 | 3 |
| ✅ Yes | Metadata-Based | serum\_sodium | 113.0 | 130.00 | 144.0 | 148.0 | 130 | 144 |
| ✅ Yes | Metadata-Based | follow\_up | 4.0 | 12.90 | 250.0 | 285.0 | 13 | 249 |

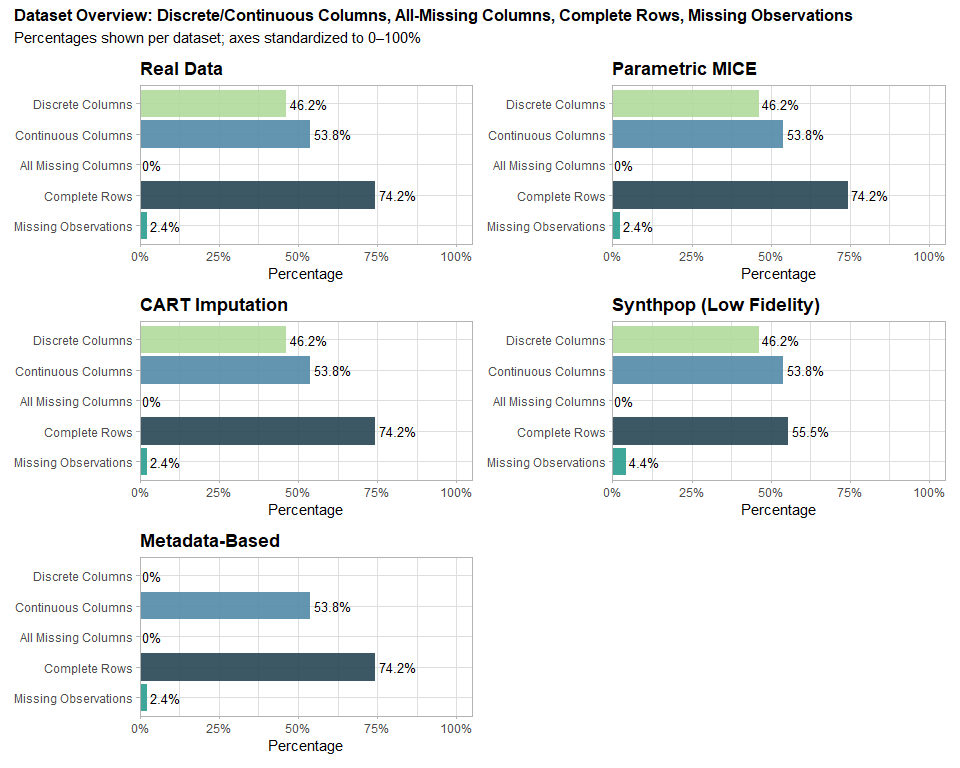
### 5.4 Missingness Comparison

In this section, we compare the proportions of missing values between the real dataset and each synthetic dataset. This comparison helps determine whether the synthetic datasets accurately represent the patterns of missingness in the real data.

* **Missingness Proportions**: For each dataset, we calculate the proportion of missing values for every column and compare the results across the real and synthetic datasets.
* **Comparison Outcome**: Ideally, the synthetic datasets should exhibit similar missingness proportions to the real data. Any deviations may indicate differences in how missing values were handled during the synthetic data generation process.

#### 5.4.1 Introduction Plots

# Custom palette for the 5 metrics  
metric\_palette <- c(  
 "Discrete Columns" = "#B0D99B",  
 "Continuous Columns" = "#528AA8",  
 "All Missing Columns" = "#FFB6DB",  
 "Complete Rows" = "#264653",  
 "Missing Observations" = "#2A9D8F"  
)  
  
# Helper: strip 'synth\_' for fair visuals  
strip\_synth\_prefix <- function(df) {   
 names(df) <- sub("^synth\_", "", names(df))   
 df   
}  
  
# Datasets to compare  
datasets <- list(  
 "Real Data" = heart\_failure,  
 "Parametric MICE" = strip\_synth\_prefix(syn\_data\_1),  
 "CART Imputation" = strip\_synth\_prefix(syn\_cart\_1),  
 "Synthpop (Low Fidelity)" = strip\_synth\_prefix(syn\_data\_low\_fidelity\_synthpop),  
 "Metadata-Based" = strip\_synth\_prefix(syn\_data\_metadata)  
)  
  
# Function: horizontal bar chart for the 5 metrics  
make\_intro\_plot <- function(df, title) {  
 n\_rows <- nrow(df)  
 n\_cols <- ncol(df)  
  
 pct\_discrete <- 100 \* sum(vapply(df, is.factor, logical(1))) / n\_cols  
 pct\_continuous <- 100 \* sum(vapply(df, is.numeric, logical(1))) / n\_cols  
 pct\_allmisscol <- 100 \* sum(colSums(is.na(df)) == n\_rows) / n\_cols  
 pct\_completer <- 100 \* sum(complete.cases(df)) / n\_rows  
 pct\_missobs <- 100 \* sum(is.na(df)) / (n\_rows \* n\_cols)  
  
 plot\_df <- tibble::tibble(  
 Metric = factor(  
 c("Discrete Columns", "Continuous Columns", "All Missing Columns",  
 "Complete Rows", "Missing Observations"),  
 levels = rev(c("Discrete Columns", "Continuous Columns", "All Missing Columns",  
 "Complete Rows", "Missing Observations")) # reverse for nicer order  
 ),  
 Percentage = c(pct\_discrete, pct\_continuous, pct\_allmisscol, pct\_completer, pct\_missobs)  
 )  
  
 ggplot(plot\_df, aes(x = Percentage, y = Metric, fill = Metric)) +  
 geom\_col(alpha = 0.9, show.legend = FALSE) +  
 geom\_text(aes(label = paste0(round(Percentage, 1), "%")),  
 hjust = -0.1, size = 3.5) +  
 scale\_x\_continuous(  
 labels = scales::percent\_format(scale = 1),  
 limits = c(0, 100),  
 expand = expansion(mult = c(0, 0.05)) # headroom for labels  
 ) +  
 scale\_fill\_manual(values = metric\_palette) +  
 labs(title = title, x = "Percentage", y = NULL) +  
 theme\_light() +  
 theme(  
 axis.text.y = element\_text(size = 9),  
 plot.title = element\_text(face = "bold")  
 )  
}  
  
# Build plots and combine in one grid  
wrap\_plots(  
 imap(datasets, ~ make\_intro\_plot(.x, .y)),  
 ncol = 2  
) +  
 plot\_annotation(  
 title = "Dataset Overview: Discrete/Continuous Columns, All-Missing Columns, Complete Rows, Missing Observations",  
 subtitle = "Percentages shown per dataset; axes standardized to 0–100%",  
 theme = theme(plot.title = element\_text(face = "bold", size = 12))  
 )



#### 5.4.2 Missingness Proportions and Comparison Outcome

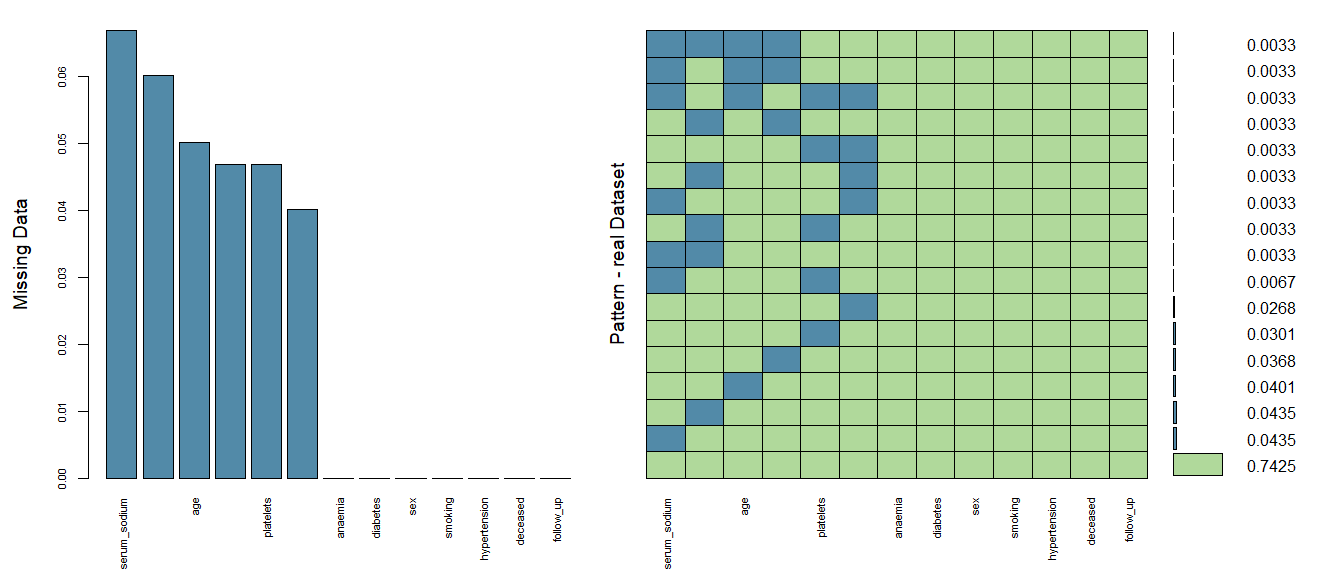
# Helper: remove 'synth\_' prefix so names match the real dataset  
strip\_synth\_prefix <- function(df) {  
 names(df) <- sub("^synth\_", "", names(df))  
 df  
}  
  
# Compare against the real dataset's variable set  
core\_vars <- names(heart\_failure)  
  
# Function to compute missingness proportions (2 d.p.) for one dataset  
get\_missingness <- function(df, label) {  
 df2 <- df %>% strip\_synth\_prefix()  
 # keep only variables present in the real data (prevents mismatches)  
 df2 <- df2 %>% select(any\_of(core\_vars))  
 tibble(  
 Variable = names(df2),  
 Proportion = round(colSums(is.na(df2)) / nrow(df2), 2),  
 Dataset = label  
 )  
}  
  
# Build tidy table  
missing\_tbl <- bind\_rows(  
 get\_missingness(heart\_failure, "Real Data"),  
 get\_missingness(syn\_data\_1, "Parametric MICE"),  
 get\_missingness(syn\_cart\_1, "CART Imputation"),  
 get\_missingness(syn\_data\_low\_fidelity\_synthpop, "Synthpop (Low Fidelity)"),  
 get\_missingness(syn\_data\_metadata, "Metadata-Based")  
)  
  
# Pivot wider for side-by-side comparison  
missing\_tbl\_wide <- missing\_tbl %>%  
 pivot\_wider(names\_from = Dataset, values\_from = Proportion) %>%  
 arrange(Variable)  
  
# Optional: add quick match flags (✅ if identical to Real Data)  
# missing\_tbl\_wide <- missing\_tbl\_wide %>%  
# mutate(  
# `Parametric MICE Match` = ifelse(`Parametric MICE` == `Real Data`, "✅", "❌"),  
# `CART Imputation Match` = ifelse(`CART Imputation` == `Real Data`, "✅", "❌"),  
# `Synthpop (Low Fidelity) Match` = ifelse(`Synthpop (Low Fidelity)` == `Real Data`, "✅", "❌"),  
# `Metadata-Based Match` = ifelse(`Metadata-Based` == `Real Data`, "✅", "❌")  
# )  
  
kable(  
 missing\_tbl\_wide,  
 caption = "Missingness Proportions by Variable (rounded to 2 d.p.)"  
)

Missingness Proportions by Variable (rounded to 2 d.p.)

| Variable | Real Data | Parametric MICE | CART Imputation | Synthpop (Low Fidelity) | Metadata-Based |
| --- | --- | --- | --- | --- | --- |
| age | 0.05 | 0.05 | 0.05 | 0.09 | 0.05 |
| anaemia | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| creatinine\_phosphokinase | 0.05 | 0.05 | 0.05 | 0.10 | 0.05 |
| deceased | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| diabetes | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| ejection\_fraction | 0.04 | 0.04 | 0.04 | 0.05 | 0.04 |
| follow\_up | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| hypertension | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| platelets | 0.05 | 0.05 | 0.05 | 0.10 | 0.05 |
| serum\_creatinine | 0.06 | 0.06 | 0.06 | 0.12 | 0.06 |
| serum\_sodium | 0.07 | 0.07 | 0.07 | 0.11 | 0.07 |
| sex | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| smoking | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |

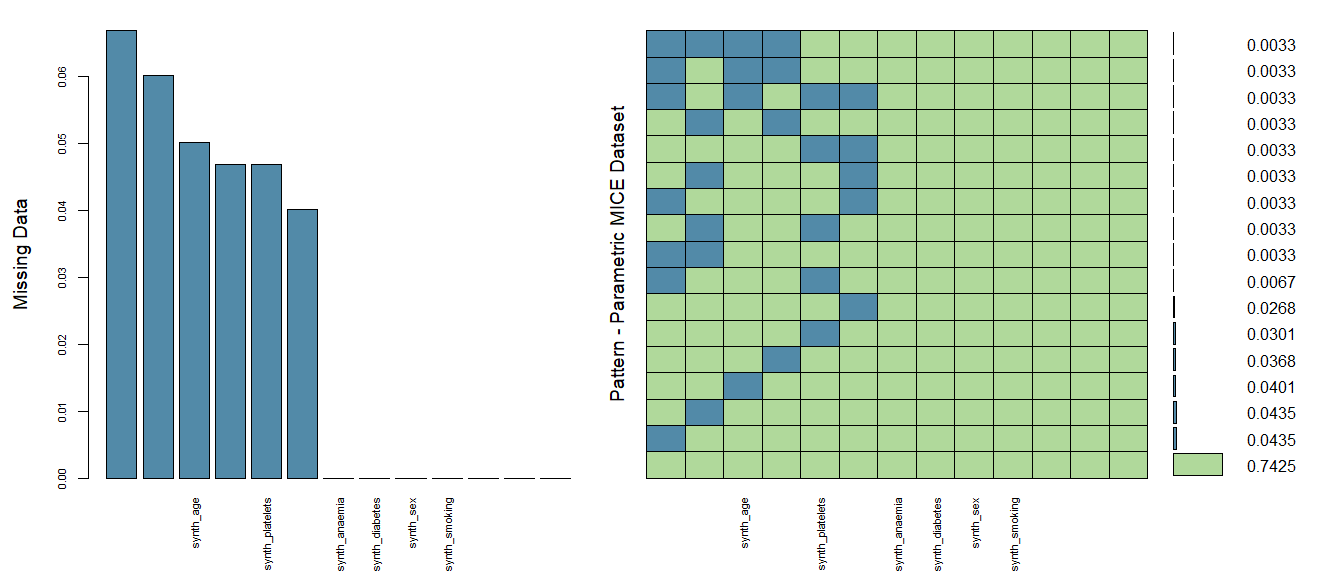
#### 5.4.3 Missingness Maps

# Generate missingness map for the real dataset  
real\_missing\_map <- aggr(heart\_failure, col = c("#B0D99B", "#528AA8"),  
 numbers = TRUE, sortVars = TRUE,  
 labels = names(heart\_failure), cex.axis = .7,  
 gap = 3, ylab = c("Missing Data", "Pattern - real Dataset"))



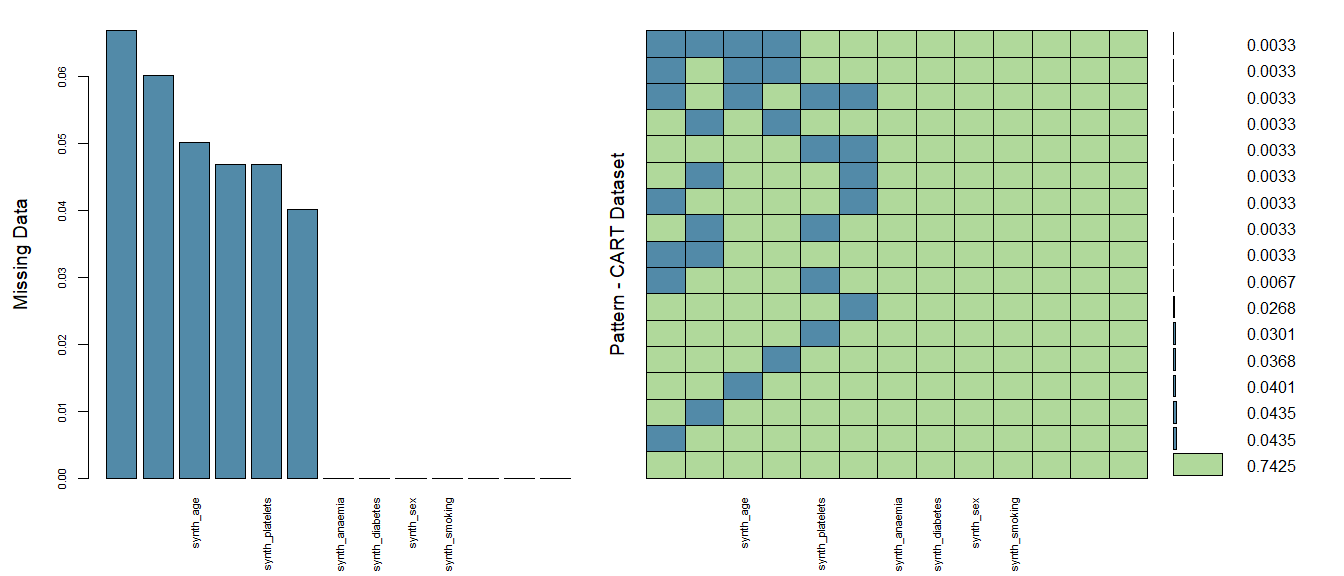
Variables sorted by number of missings:   
 Variable Count  
 serum\_sodium 0.06688963  
 serum\_creatinine 0.06020067  
 age 0.05016722  
 creatinine\_phosphokinase 0.04682274  
 platelets 0.04682274  
 ejection\_fraction 0.04013378  
 anaemia 0.00000000  
 diabetes 0.00000000  
 sex 0.00000000  
 smoking 0.00000000  
 hypertension 0.00000000  
 deceased 0.00000000  
 follow\_up 0.00000000

# Generate missingness map for Parametric MICE synthetic dataset  
parametric\_mice\_missing\_map <- aggr(syn\_data\_1, col = c("#B0D99B", "#528AA8"),  
 numbers = TRUE, sortVars = TRUE,  
 labels = names(syn\_data\_1), cex.axis = .7,  
 gap = 3, ylab = c("Missing Data", "Pattern - Parametric MICE Dataset"))



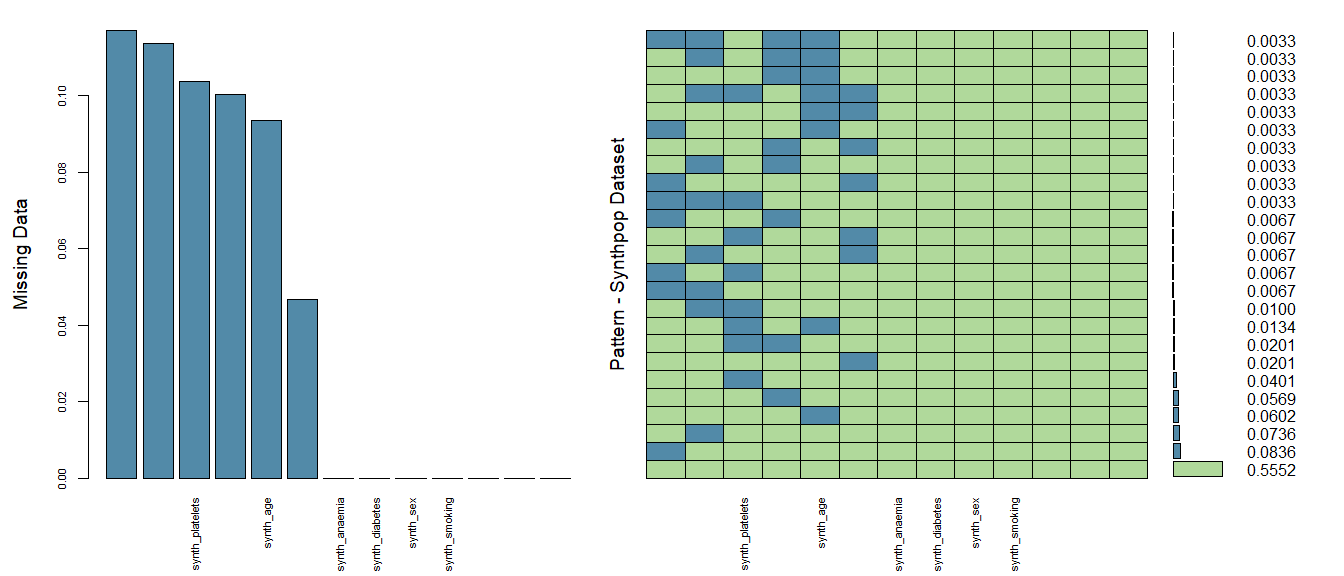
Variables sorted by number of missings:   
 Variable Count  
 synth\_serum\_sodium 0.06688963  
 synth\_serum\_creatinine 0.06020067  
 synth\_age 0.05016722  
 synth\_creatinine\_phosphokinase 0.04682274  
 synth\_platelets 0.04682274  
 synth\_ejection\_fraction 0.04013378  
 synth\_anaemia 0.00000000  
 synth\_diabetes 0.00000000  
 synth\_sex 0.00000000  
 synth\_smoking 0.00000000  
 synth\_hypertension 0.00000000  
 synth\_deceased 0.00000000  
 synth\_follow\_up 0.00000000

# Generate missingness map for CART synthetic dataset  
cart\_missing\_map <- aggr(syn\_cart\_1, col = c("#B0D99B", "#528AA8"),  
 numbers = TRUE, sortVars = TRUE,  
 labels = names(syn\_cart\_1), cex.axis = .7,  
 gap = 3, ylab = c("Missing Data", "Pattern - CART Dataset"))



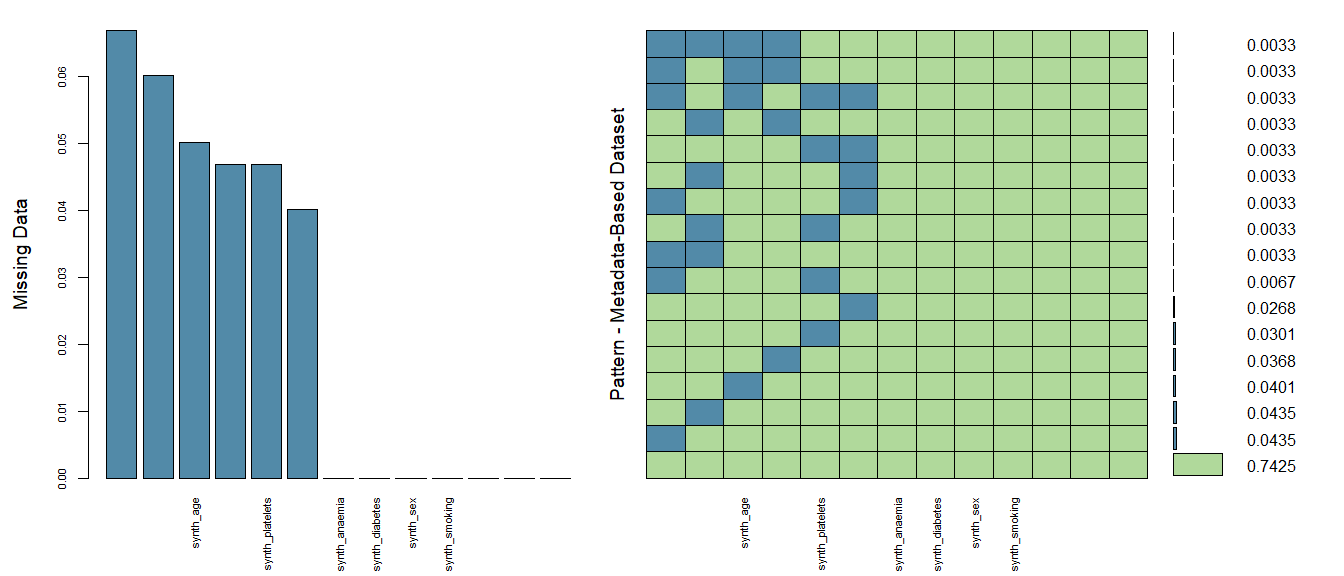
Variables sorted by number of missings:   
 Variable Count  
 synth\_serum\_sodium 0.06688963  
 synth\_serum\_creatinine 0.06020067  
 synth\_age 0.05016722  
 synth\_creatinine\_phosphokinase 0.04682274  
 synth\_platelets 0.04682274  
 synth\_ejection\_fraction 0.04013378  
 synth\_anaemia 0.00000000  
 synth\_diabetes 0.00000000  
 synth\_sex 0.00000000  
 synth\_smoking 0.00000000  
 synth\_hypertension 0.00000000  
 synth\_deceased 0.00000000  
 synth\_follow\_up 0.00000000

# Generate missingness map for Synthpop synthetic dataset  
synthpop\_missing\_map <- aggr(syn\_data\_low\_fidelity\_synthpop, col = c("#B0D99B", "#528AA8"),  
 numbers = TRUE, sortVars = TRUE,  
 labels = names(syn\_data\_low\_fidelity\_synthpop), cex.axis = .7,  
 gap = 3, ylab = c("Missing Data", "Pattern - Synthpop Dataset"))



Variables sorted by number of missings:   
 Variable Count  
 synth\_serum\_creatinine 0.11705686  
 synth\_serum\_sodium 0.11371237  
 synth\_platelets 0.10367893  
 synth\_creatinine\_phosphokinase 0.10033445  
 synth\_age 0.09364548  
 synth\_ejection\_fraction 0.04682274  
 synth\_anaemia 0.00000000  
 synth\_diabetes 0.00000000  
 synth\_sex 0.00000000  
 synth\_smoking 0.00000000  
 synth\_hypertension 0.00000000  
 synth\_deceased 0.00000000  
 synth\_follow\_up 0.00000000

# Generate missingness map for Metadata-based synthetic dataset  
metadata\_missing\_map <- aggr(syn\_data\_metadata, col = c("#B0D99B", "#528AA8"),  
 numbers = TRUE, sortVars = TRUE,  
 labels = names(syn\_data\_metadata), cex.axis = .7,  
 gap = 3, ylab = c("Missing Data", "Pattern - Metadata-Based Dataset"))



Variables sorted by number of missings:   
 Variable Count  
 synth\_serum\_sodium 0.06688963  
 synth\_serum\_creatinine 0.06020067  
 synth\_age 0.05016722  
 synth\_creatinine\_phosphokinase 0.04682274  
 synth\_platelets 0.04682274  
 synth\_ejection\_fraction 0.04013378  
 synth\_anaemia 0.00000000  
 synth\_diabetes 0.00000000  
 synth\_sex 0.00000000  
 synth\_smoking 0.00000000  
 synth\_hypertension 0.00000000  
 synth\_deceased 0.00000000  
 synth\_follow\_up 0.00000000

### 5.5 Correlation Matrices Comparison

In this section, we assess how well the correlations between numeric variables are preserved in the synthetic datasets compared to the real dataset. By comparing the correlation matrices, we evaluate whether the relationships between variables in the real data are reflected in the synthetic versions. This assessment is essential because maintaining the real data’s correlation structure ensures that any statistical or machine learning models trained on synthetic data will behave similarly to those trained on real data.

* **Correlation Matrix**: A correlation matrix quantifies the strength and direction of relationships between pairs of numeric variables. Each cell in the matrix represents the correlation coefficient between two variables, which can range from -1 (perfect negative correlation) to +1 (perfect positive correlation). It is crucial that synthetic data retains these relationships to ensure that any analyses based on these dependencies remain valid.
* **Comparison Process**: The correlation matrices for both the real and synthetic datasets are computed and Visualised. Ideally, the structure and strength of correlations in the synthetic datasets should closely match those in the real dataset. Differences in correlation strength or direction indicate that the synthetic data may not fully capture the underlying relationships, which can affect the validity of any conclusions drawn from analyses using the synthetic data.

Maintaining similar correlation structures is particularly important for use cases where relationships between variables play a crucial role, such as in predictive modeling, feature selection, and understanding variable dependencies. The following sections outline different methods used to compare correlation matrices between the real and synthetic datasets.

#### 5.5.1 Correlation Matrices Comparison using psych

The psych package provides tools for analyzing and visualizing correlation matrices. The corr.test() function is particularly useful for computing correlation coefficients while handling missing data and providing additional statistics, such as confidence intervals.

# Helper: strip 'synth\_' so names line up with real  
strip\_synth\_prefix <- function(df) {  
 names(df) <- sub("^synth\_", "", names(df))  
 df  
}  
  
# Build a tidy table of pairwise correlations for real vs one synthetic dataset  
corr\_compare\_one <- function(real, synth, label) {  
 real\_num <- real |> select(where(is.numeric))  
 synth\_num <- synth |> strip\_synth\_prefix() |> select(where(is.numeric))  
  
 common <- intersect(names(real\_num), names(synth\_num))  
 if (length(common) < 2) {  
 return(tibble(Dataset = label, Var1 = character(), Var2 = character(),  
 Real = numeric(), Synthetic = numeric(), Diff = numeric(), AbsDiff = numeric()))  
 }  
  
 # Compute Pearson correlations with pairwise complete observations  
 rc <- cor(real\_num[common], use = "pairwise.complete.obs", method = "pearson")  
 sc <- cor(synth\_num[common], use = "pairwise.complete.obs", method = "pearson")  
  
 # Keep only upper triangle (unique pairs)  
 ut <- upper.tri(rc, diag = FALSE)  
 pairs <- which(ut, arr.ind = TRUE)  
 tibble(  
 Var1 = colnames(rc)[pairs[, 1]],  
 Var2 = colnames(rc)[pairs[, 2]],  
 Real = rc[pairs],  
 Synthetic = sc[pairs]  
 ) |>  
 mutate(  
 Dataset = label,  
 Diff = Synthetic - Real,  
 AbsDiff = abs(Diff)  
 ) |>  
 select(Dataset, Var1, Var2, Real, Synthetic, Diff, AbsDiff)  
}  
  
# Build one combined table for all synthetic datasets  
corr\_cmp\_tbl <- bind\_rows(  
 corr\_compare\_one(heart\_failure, syn\_data\_1, "Parametric MICE"),  
 corr\_compare\_one(heart\_failure, syn\_cart\_1, "CART Imputation"),  
 corr\_compare\_one(heart\_failure, syn\_data\_low\_fidelity\_synthpop, "Synthpop (Low Fidelity)"),  
 corr\_compare\_one(heart\_failure, syn\_data\_metadata, "Metadata-Based")  
) |>  
 arrange(Dataset, desc(AbsDiff)) |>  
 mutate(  
 Real = round(Real, 2),  
 Synthetic = round(Synthetic, 2),  
 Diff = round(Diff, 2),  
 AbsDiff = round(AbsDiff, 2)  
 )  
  
knitr::kable(  
 corr\_cmp\_tbl,  
 caption = "Pairwise Correlations: Real vs. Synthetic (sorted by largest absolute difference per dataset)",  
 align = "lllrrrr"  
)

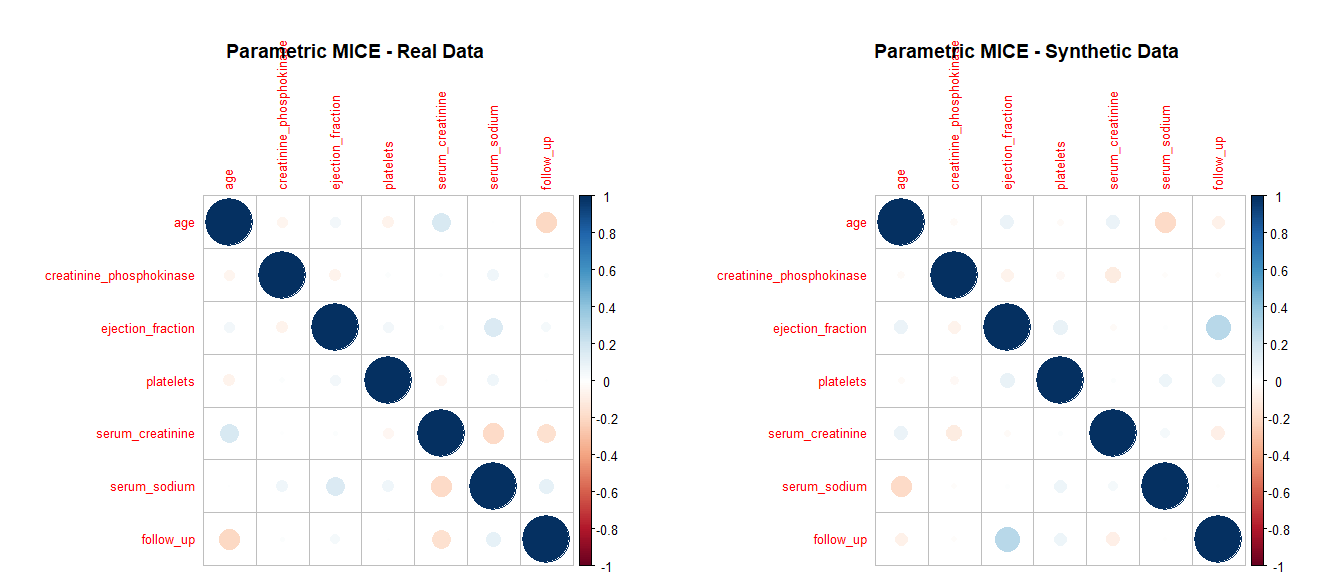
Pairwise Correlations: Real vs. Synthetic (sorted by largest absolute difference per dataset)

| Dataset | Var1 | Var2 | Real | Synthetic | Diff | AbsDiff |
| --- | --- | --- | --- | --- | --- | --- |
| CART Imputation | creatinine\_phosphokinase | platelets | 0.01 | -0.12 | -0.13 | 0.13 |
| CART Imputation | ejection\_fraction | serum\_creatinine | -0.01 | -0.11 | -0.10 | 0.10 |
| CART Imputation | creatinine\_phosphokinase | serum\_sodium | 0.06 | -0.04 | -0.10 | 0.10 |
| CART Imputation | age | follow\_up | -0.24 | -0.15 | 0.09 | 0.09 |
| CART Imputation | ejection\_fraction | platelets | 0.06 | -0.02 | -0.08 | 0.08 |
| CART Imputation | age | serum\_creatinine | 0.17 | 0.25 | 0.08 | 0.08 |
| CART Imputation | platelets | serum\_creatinine | -0.05 | 0.03 | 0.08 | 0.08 |
| CART Imputation | ejection\_fraction | follow\_up | 0.03 | -0.05 | -0.08 | 0.08 |
| CART Imputation | serum\_sodium | follow\_up | 0.11 | 0.03 | -0.07 | 0.07 |
| CART Imputation | age | creatinine\_phosphokinase | -0.08 | -0.15 | -0.07 | 0.07 |
| CART Imputation | platelets | serum\_sodium | 0.06 | -0.01 | -0.07 | 0.07 |
| CART Imputation | creatinine\_phosphokinase | follow\_up | -0.03 | 0.03 | 0.06 | 0.06 |
| CART Imputation | platelets | follow\_up | -0.01 | 0.05 | 0.06 | 0.06 |
| CART Imputation | age | platelets | -0.05 | -0.01 | 0.04 | 0.04 |
| CART Imputation | age | serum\_sodium | -0.04 | -0.08 | -0.04 | 0.04 |
| CART Imputation | age | ejection\_fraction | 0.04 | 0.00 | -0.04 | 0.04 |
| CART Imputation | creatinine\_phosphokinase | ejection\_fraction | -0.05 | -0.02 | 0.03 | 0.03 |
| CART Imputation | creatinine\_phosphokinase | serum\_creatinine | 0.00 | -0.02 | -0.02 | 0.02 |
| CART Imputation | serum\_creatinine | serum\_sodium | -0.21 | -0.23 | -0.02 | 0.02 |
| CART Imputation | ejection\_fraction | serum\_sodium | 0.17 | 0.18 | 0.02 | 0.02 |
| CART Imputation | serum\_creatinine | follow\_up | -0.15 | -0.14 | 0.01 | 0.01 |
| Metadata-Based | age | follow\_up | -0.24 | 0.12 | 0.36 | 0.36 |
| Metadata-Based | platelets | serum\_sodium | 0.06 | -0.16 | -0.23 | 0.23 |
| Metadata-Based | serum\_creatinine | serum\_sodium | -0.21 | 0.01 | 0.21 | 0.21 |
| Metadata-Based | creatinine\_phosphokinase | ejection\_fraction | -0.05 | 0.11 | 0.16 | 0.16 |
| Metadata-Based | ejection\_fraction | serum\_sodium | 0.17 | 0.03 | -0.13 | 0.13 |
| Metadata-Based | creatinine\_phosphokinase | serum\_creatinine | 0.00 | -0.12 | -0.13 | 0.13 |
| Metadata-Based | creatinine\_phosphokinase | follow\_up | -0.03 | 0.09 | 0.12 | 0.12 |
| Metadata-Based | age | serum\_creatinine | 0.17 | 0.07 | -0.10 | 0.10 |
| Metadata-Based | serum\_sodium | follow\_up | 0.11 | 0.00 | -0.10 | 0.10 |
| Metadata-Based | age | serum\_sodium | -0.04 | 0.06 | 0.10 | 0.10 |
| Metadata-Based | serum\_creatinine | follow\_up | -0.15 | -0.05 | 0.09 | 0.09 |
| Metadata-Based | creatinine\_phosphokinase | platelets | 0.01 | -0.08 | -0.09 | 0.09 |
| Metadata-Based | ejection\_fraction | platelets | 0.06 | -0.02 | -0.08 | 0.08 |
| Metadata-Based | age | platelets | -0.05 | 0.02 | 0.07 | 0.07 |
| Metadata-Based | age | ejection\_fraction | 0.04 | -0.03 | -0.07 | 0.07 |
| Metadata-Based | ejection\_fraction | follow\_up | 0.03 | 0.08 | 0.05 | 0.05 |
| Metadata-Based | creatinine\_phosphokinase | serum\_sodium | 0.06 | 0.11 | 0.05 | 0.05 |
| Metadata-Based | age | creatinine\_phosphokinase | -0.08 | -0.05 | 0.03 | 0.03 |
| Metadata-Based | ejection\_fraction | serum\_creatinine | -0.01 | 0.01 | 0.02 | 0.02 |
| Metadata-Based | platelets | follow\_up | -0.01 | -0.02 | -0.01 | 0.01 |
| Metadata-Based | platelets | serum\_creatinine | -0.05 | -0.06 | -0.01 | 0.01 |
| Parametric MICE | serum\_creatinine | serum\_sodium | -0.21 | 0.02 | 0.23 | 0.23 |
| Parametric MICE | ejection\_fraction | follow\_up | 0.03 | 0.20 | 0.17 | 0.17 |
| Parametric MICE | age | follow\_up | -0.24 | -0.08 | 0.16 | 0.16 |
| Parametric MICE | ejection\_fraction | serum\_sodium | 0.17 | 0.03 | -0.13 | 0.13 |
| Parametric MICE | serum\_sodium | follow\_up | 0.11 | -0.01 | -0.11 | 0.11 |
| Parametric MICE | platelets | serum\_creatinine | -0.05 | 0.06 | 0.11 | 0.11 |
| Parametric MICE | creatinine\_phosphokinase | serum\_creatinine | 0.00 | -0.11 | -0.11 | 0.11 |
| Parametric MICE | age | serum\_creatinine | 0.17 | 0.08 | -0.09 | 0.09 |
| Parametric MICE | creatinine\_phosphokinase | platelets | 0.01 | -0.07 | -0.08 | 0.08 |
| Parametric MICE | age | serum\_sodium | -0.04 | -0.11 | -0.07 | 0.07 |
| Parametric MICE | age | creatinine\_phosphokinase | -0.08 | -0.02 | 0.06 | 0.06 |
| Parametric MICE | creatinine\_phosphokinase | follow\_up | -0.03 | 0.03 | 0.06 | 0.06 |
| Parametric MICE | platelets | follow\_up | -0.01 | 0.04 | 0.05 | 0.05 |
| Parametric MICE | creatinine\_phosphokinase | ejection\_fraction | -0.05 | -0.01 | 0.04 | 0.04 |
| Parametric MICE | serum\_creatinine | follow\_up | -0.15 | -0.11 | 0.04 | 0.04 |
| Parametric MICE | platelets | serum\_sodium | 0.06 | 0.03 | -0.03 | 0.03 |
| Parametric MICE | creatinine\_phosphokinase | serum\_sodium | 0.06 | 0.03 | -0.03 | 0.03 |
| Parametric MICE | ejection\_fraction | serum\_creatinine | -0.01 | -0.04 | -0.03 | 0.03 |
| Parametric MICE | age | platelets | -0.05 | -0.03 | 0.02 | 0.02 |
| Parametric MICE | age | ejection\_fraction | 0.04 | 0.05 | 0.01 | 0.01 |
| Parametric MICE | ejection\_fraction | platelets | 0.06 | 0.06 | 0.00 | 0.00 |
| Synthpop (Low Fidelity) | age | follow\_up | -0.24 | 0.04 | 0.28 | 0.28 |
| Synthpop (Low Fidelity) | serum\_creatinine | serum\_sodium | -0.21 | 0.00 | 0.20 | 0.20 |
| Synthpop (Low Fidelity) | age | serum\_creatinine | 0.17 | -0.03 | -0.20 | 0.20 |
| Synthpop (Low Fidelity) | ejection\_fraction | serum\_sodium | 0.17 | -0.02 | -0.19 | 0.19 |
| Synthpop (Low Fidelity) | age | serum\_sodium | -0.04 | 0.11 | 0.15 | 0.15 |
| Synthpop (Low Fidelity) | serum\_creatinine | follow\_up | -0.15 | 0.00 | 0.15 | 0.15 |
| Synthpop (Low Fidelity) | serum\_sodium | follow\_up | 0.11 | -0.03 | -0.14 | 0.14 |
| Synthpop (Low Fidelity) | age | creatinine\_phosphokinase | -0.08 | 0.03 | 0.12 | 0.12 |
| Synthpop (Low Fidelity) | creatinine\_phosphokinase | ejection\_fraction | -0.05 | 0.05 | 0.10 | 0.10 |
| Synthpop (Low Fidelity) | ejection\_fraction | platelets | 0.06 | 0.15 | 0.09 | 0.09 |
| Synthpop (Low Fidelity) | platelets | follow\_up | -0.01 | 0.08 | 0.09 | 0.09 |
| Synthpop (Low Fidelity) | ejection\_fraction | follow\_up | 0.03 | 0.09 | 0.06 | 0.06 |
| Synthpop (Low Fidelity) | creatinine\_phosphokinase | platelets | 0.01 | -0.05 | -0.06 | 0.06 |
| Synthpop (Low Fidelity) | age | ejection\_fraction | 0.04 | -0.02 | -0.06 | 0.06 |
| Synthpop (Low Fidelity) | ejection\_fraction | serum\_creatinine | -0.01 | -0.06 | -0.05 | 0.05 |
| Synthpop (Low Fidelity) | creatinine\_phosphokinase | serum\_sodium | 0.06 | 0.01 | -0.05 | 0.05 |
| Synthpop (Low Fidelity) | creatinine\_phosphokinase | follow\_up | -0.03 | 0.02 | 0.04 | 0.04 |
| Synthpop (Low Fidelity) | platelets | serum\_creatinine | -0.05 | -0.08 | -0.03 | 0.03 |
| Synthpop (Low Fidelity) | creatinine\_phosphokinase | serum\_creatinine | 0.00 | -0.02 | -0.03 | 0.03 |
| Synthpop (Low Fidelity) | platelets | serum\_sodium | 0.06 | 0.08 | 0.02 | 0.02 |
| Synthpop (Low Fidelity) | age | platelets | -0.05 | -0.04 | 0.01 | 0.01 |

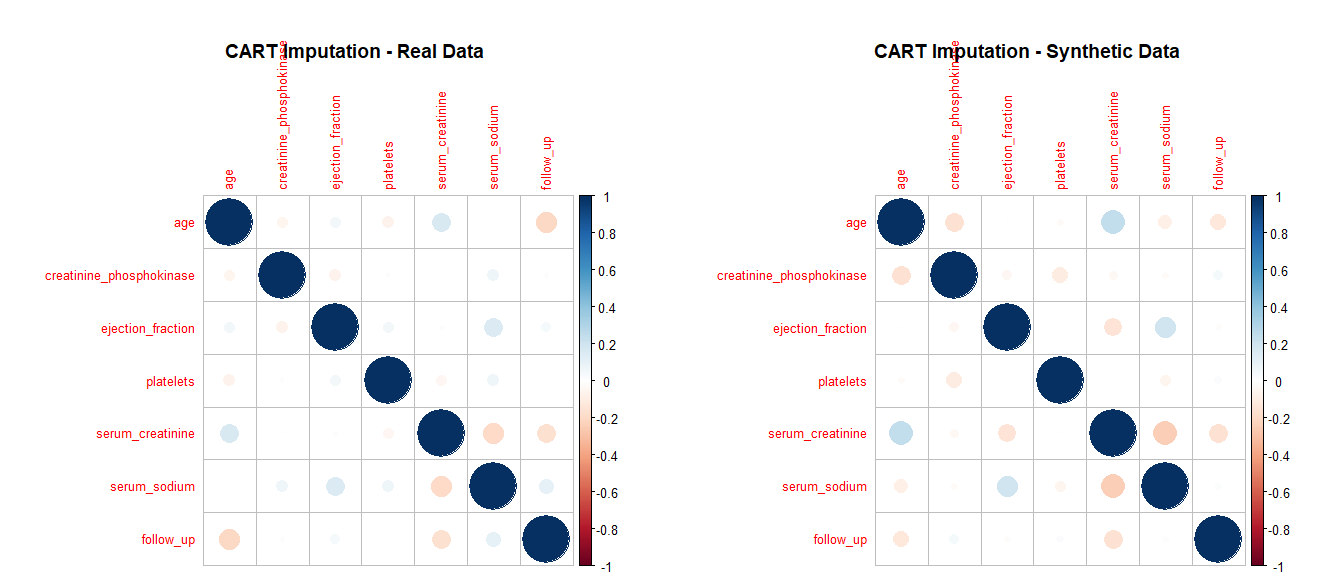
#### 5.5.2 Correlation Matrices Comparison using Corrplot

The corrplot package is widely used for visualizing correlation matrices. It provides a variety of options for customization, making it easy to identify patterns and relationships between variables. This method is particularly effective for visual comparisons because it highlights differences in correlation strength and direction.

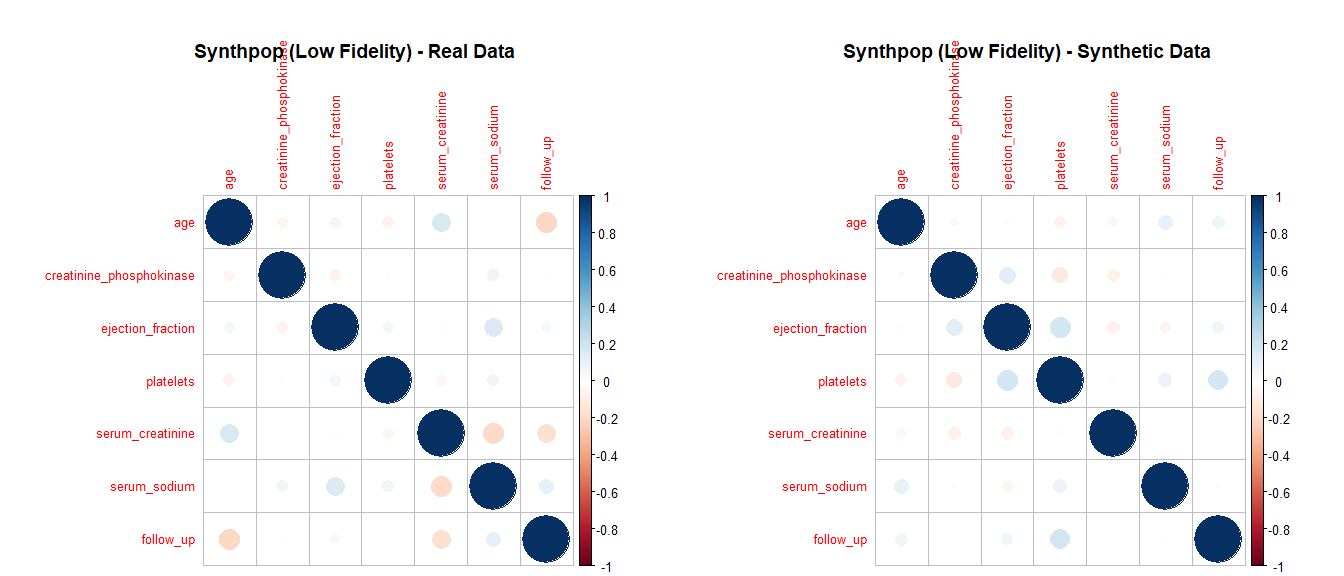
# Helper: remove 'synth\_' so names match the real dataset  
strip\_synth\_prefix <- function(df) { names(df) <- sub("^synth\_", "", names(df)); df }  
  
# Side-by-side correlation matrices (full width)  
compare\_correlation\_matrices <- function(real\_data, synthetic\_data, dataset\_name) {  
 # Numeric-only  
 real\_num <- real\_data[sapply(real\_data, is.numeric)]  
 synth\_num <- strip\_synth\_prefix(synthetic\_data)[sapply(strip\_synth\_prefix(synthetic\_data), is.numeric)]  
  
 # Use common numeric variables in same order  
 common <- intersect(names(real\_num), names(synth\_num))  
 if (length(common) < 2) {  
 message(dataset\_name, ": fewer than 2 common numeric columns — skipping.")  
 return(invisible(NULL))  
 }  
 real\_corr <- cor(real\_num[common], use = "complete.obs")  
 synth\_corr <- cor(synth\_num[common], use = "complete.obs")  
  
 # Make two panels across the page  
 op <- par(no.readonly = TRUE); on.exit(par(op), add = TRUE)  
 par(mfrow = c(1, 2), mar = c(4, 4, 4, 2))  
  
 corrplot(real\_corr, tl.cex = 0.8, mar = c(0,0,2,0))  
 title(paste(dataset\_name, "- Real Data"), line = 1)  
  
 corrplot(synth\_corr, tl.cex = 0.8, mar = c(0,0,2,0))  
 title(paste(dataset\_name, "- Synthetic Data"), line = 1)  
}  
  
# Run for each synthetic dataset  
compare\_correlation\_matrices(heart\_failure, syn\_data\_1, "Parametric MICE")



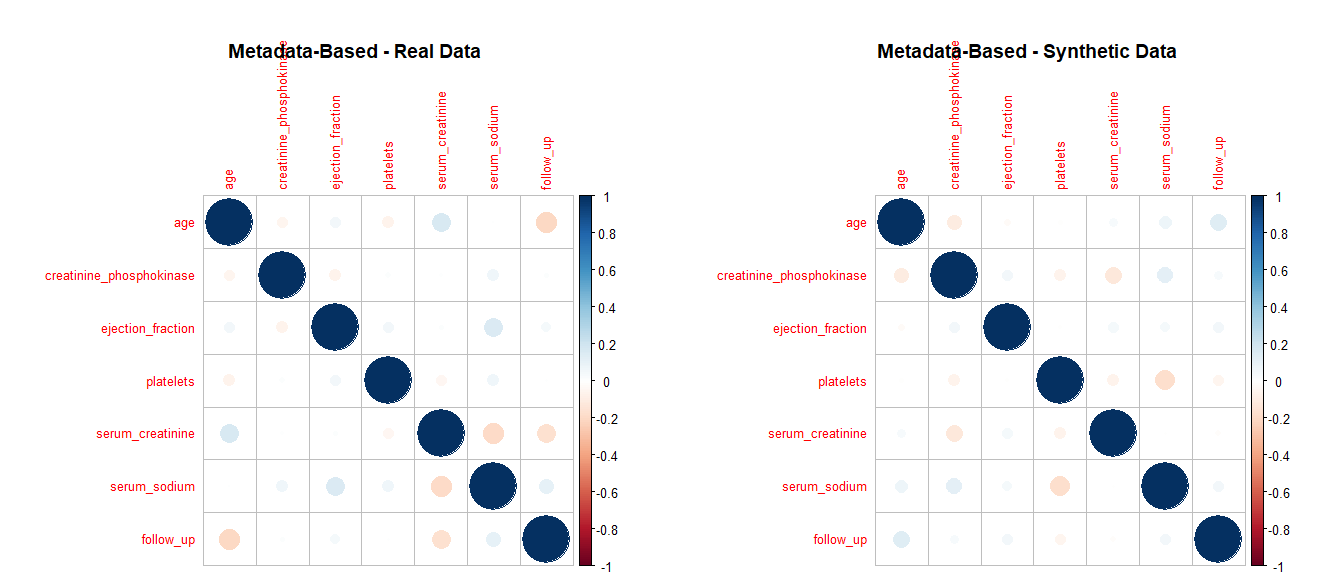
compare\_correlation\_matrices(heart\_failure, syn\_cart\_1, "CART Imputation")



compare\_correlation\_matrices(heart\_failure, syn\_data\_low\_fidelity\_synthpop, "Synthpop (Low Fidelity)")



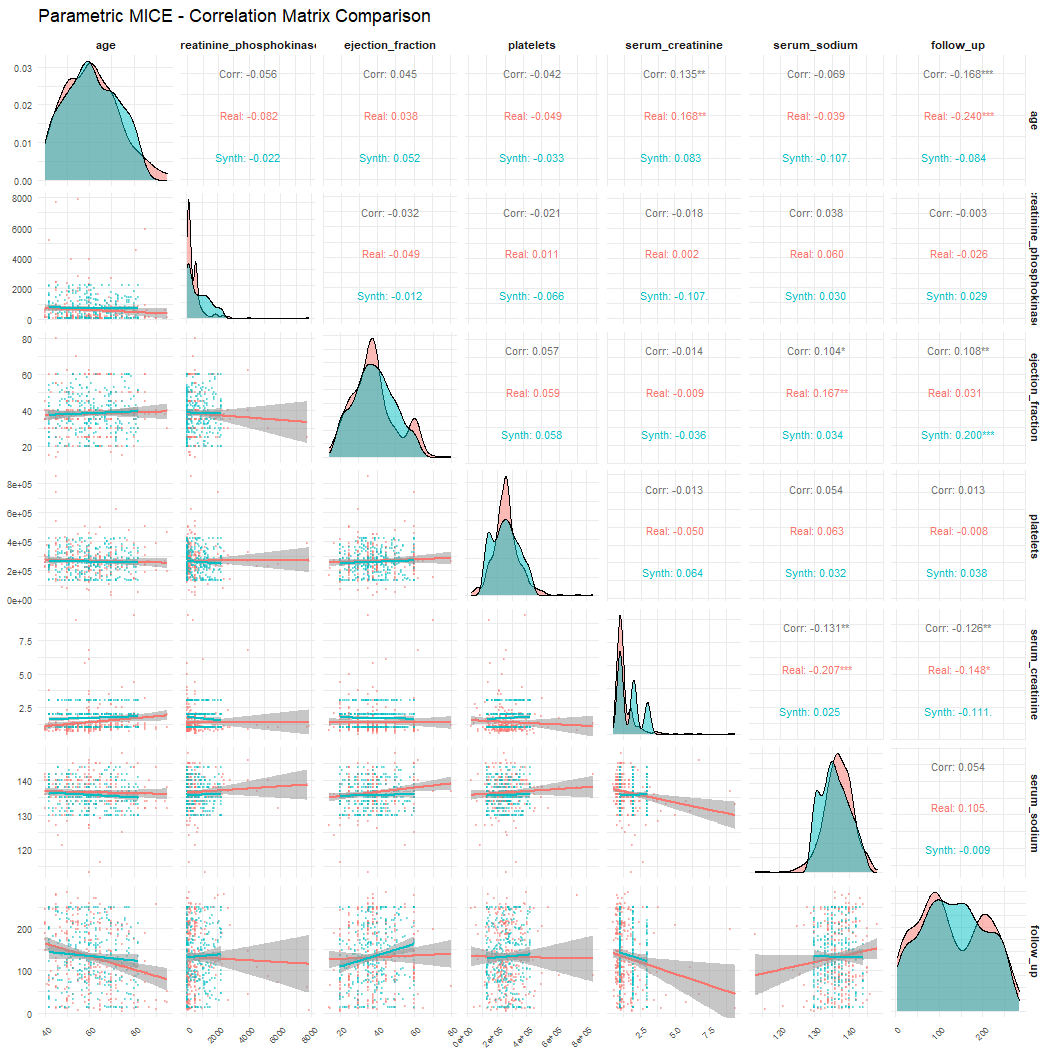
compare\_correlation\_matrices(heart\_failure, syn\_data\_metadata, "Metadata-Based")



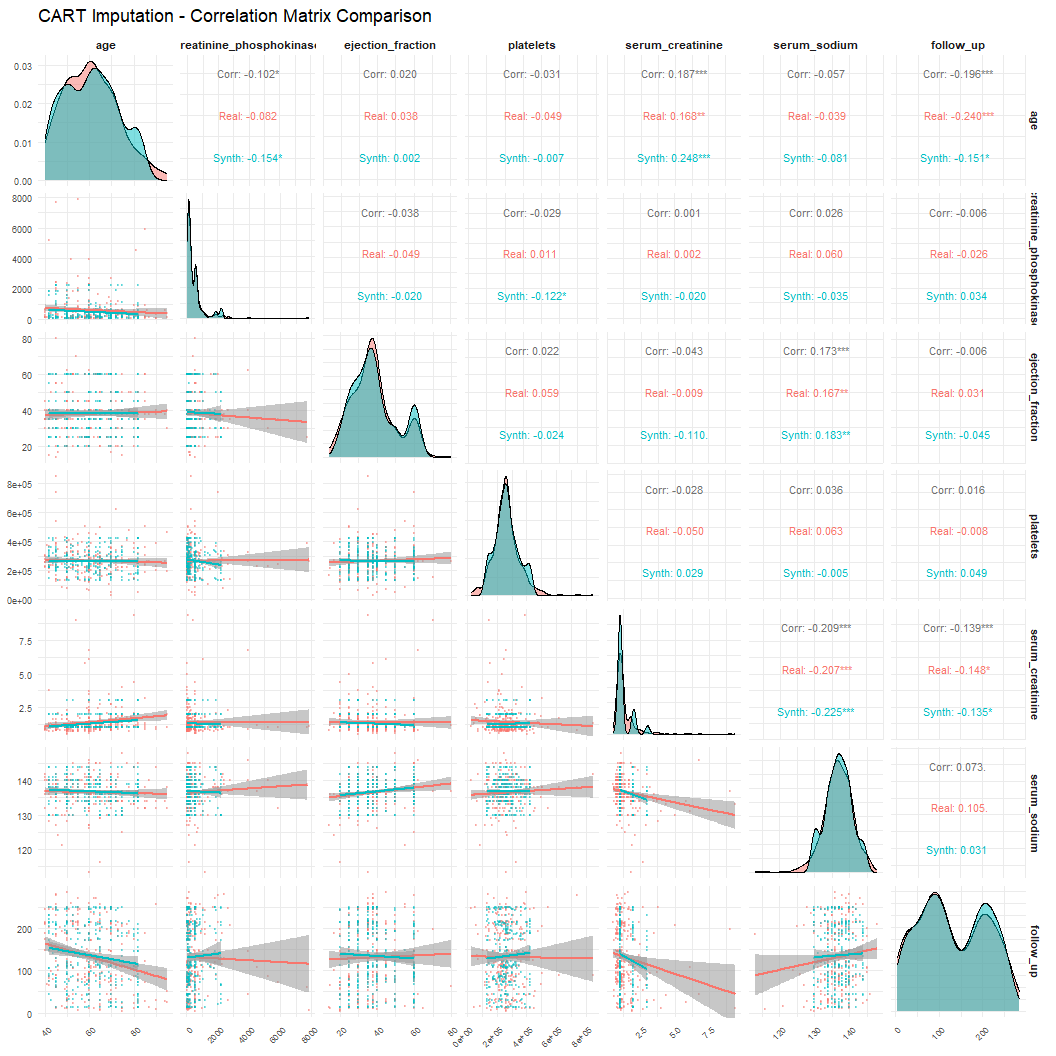
#### 5.5.3 Correlation Matrices Comparison using GGally

The GGally package extends ggplot2 to create pairwise visualizations, including correlation matrix plots. Using ggpairs(), you can generate a matrix of scatterplots, histograms, and correlation coefficients, which provides a detailed view of the relationships between numeric variables in the real and synthetic datasets.

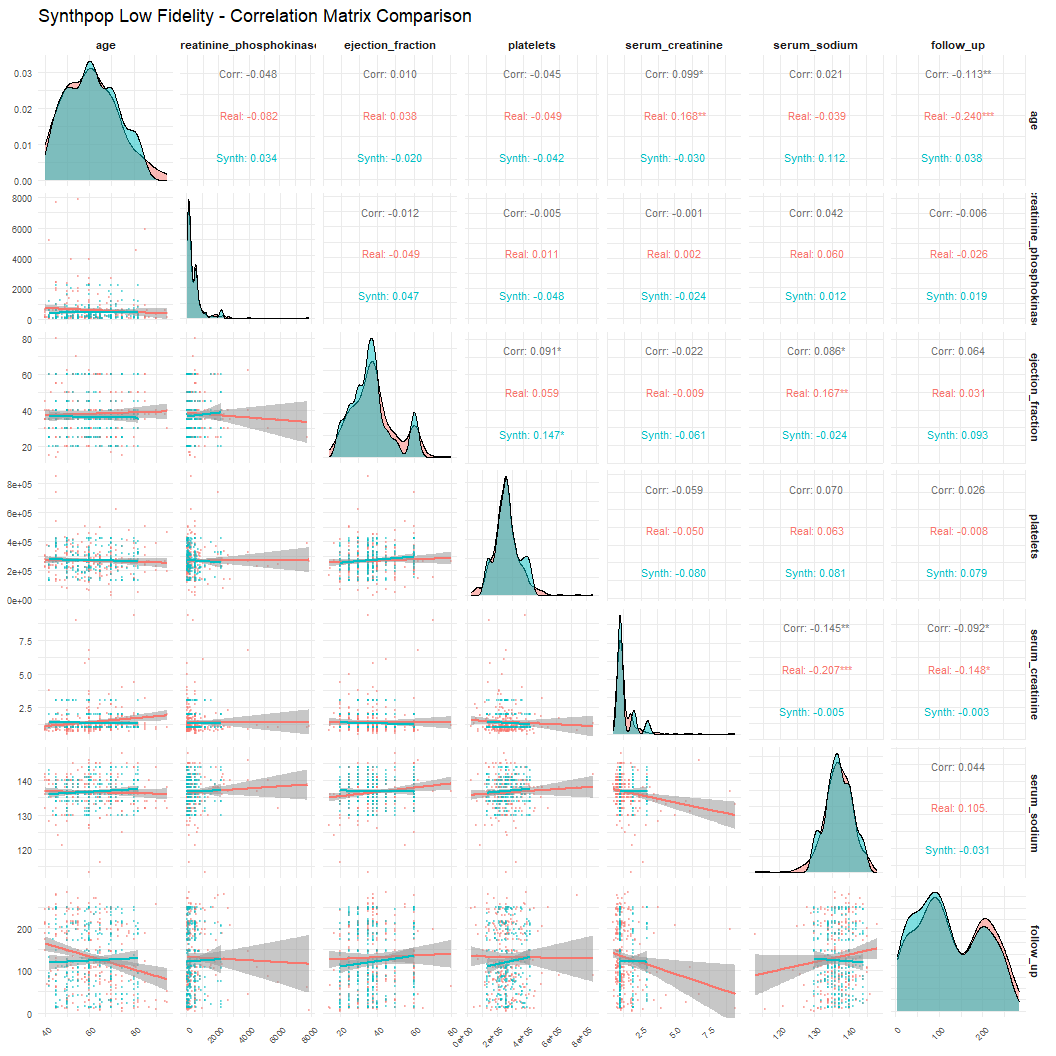
compare\_correlation\_matrices\_ggally <- function(real\_data, synthetic\_data, dataset\_name) {  
  
 # Select only numeric columns from both datasets  
 real\_numeric <- real\_data %>% select(where(is.numeric))  
  
 # Remove prefix 'synth\_' from synthetic data column names  
 names(synthetic\_data) <- gsub("^synth\_", "", names(synthetic\_data))  
 synthetic\_numeric <- synthetic\_data %>% select(where(is.numeric))  
  
 # Ensure that both datasets have the same columns  
 common\_columns <- intersect(names(real\_numeric), names(synthetic\_numeric))  
   
 if (length(common\_columns) == 0) {  
 cat("\nNo common numeric columns found for comparison in", dataset\_name, "\n")  
 return()  
 }  
  
 real\_numeric <- real\_numeric[, common\_columns, drop = FALSE]  
 synthetic\_numeric <- synthetic\_numeric[, common\_columns, drop = FALSE]  
  
 # Add a column to differentiate between real and synthetic data  
 real\_numeric$Dataset <- "Real"  
 synthetic\_numeric$Dataset <- "Synth"  
  
 # Combine the datasets  
 combined\_data <- rbind(real\_numeric, synthetic\_numeric)  
  
 # Create correlation matrix plot using GGally if there are valid columns to plot  
 if (ncol(combined\_data) > 1) {  
 p <- ggpairs(  
 combined\_data,   
 aes(color = Dataset, alpha = 0.5),   
 title = paste(dataset\_name, "- Correlation Matrix Comparison"),  
 columns = 1:(ncol(combined\_data) - 1), # Exclude the 'Dataset' column from the plot  
 upper = list(continuous = wrap("cor", size = 3, alignPercent = 0.5)), # Smaller correlation labels  
 lower = list(continuous = wrap("smooth", alpha = 0.4, size = 0.5)), # Smoother plots for readability  
 diag = list(continuous = wrap("densityDiag", alpha = 0.5)) # Density plot with improved transparency  
 ) +  
 theme\_minimal() + # Use a cleaner theme  
 theme(  
 strip.text = element\_text(size = 8, face = "bold"), # Smaller strip labels  
 axis.text.x = element\_text(angle = 45, hjust = 1, size = 7), # Rotate x-axis labels for better readability  
 axis.text.y = element\_text(size = 7),  
 legend.position = "top", # Move legend to a better location  
 legend.title = element\_blank() # Remove unnecessary legend title  
 )  
 print(p)  
 } else {  
 cat("\nNot enough numeric columns to generate a correlation matrix plot for", dataset\_name, "\n")  
 }  
}  
  
# Apply the function to each synthetic dataset  
compare\_correlation\_matrices\_ggally(heart\_failure, syn\_data\_1, "Parametric MICE")



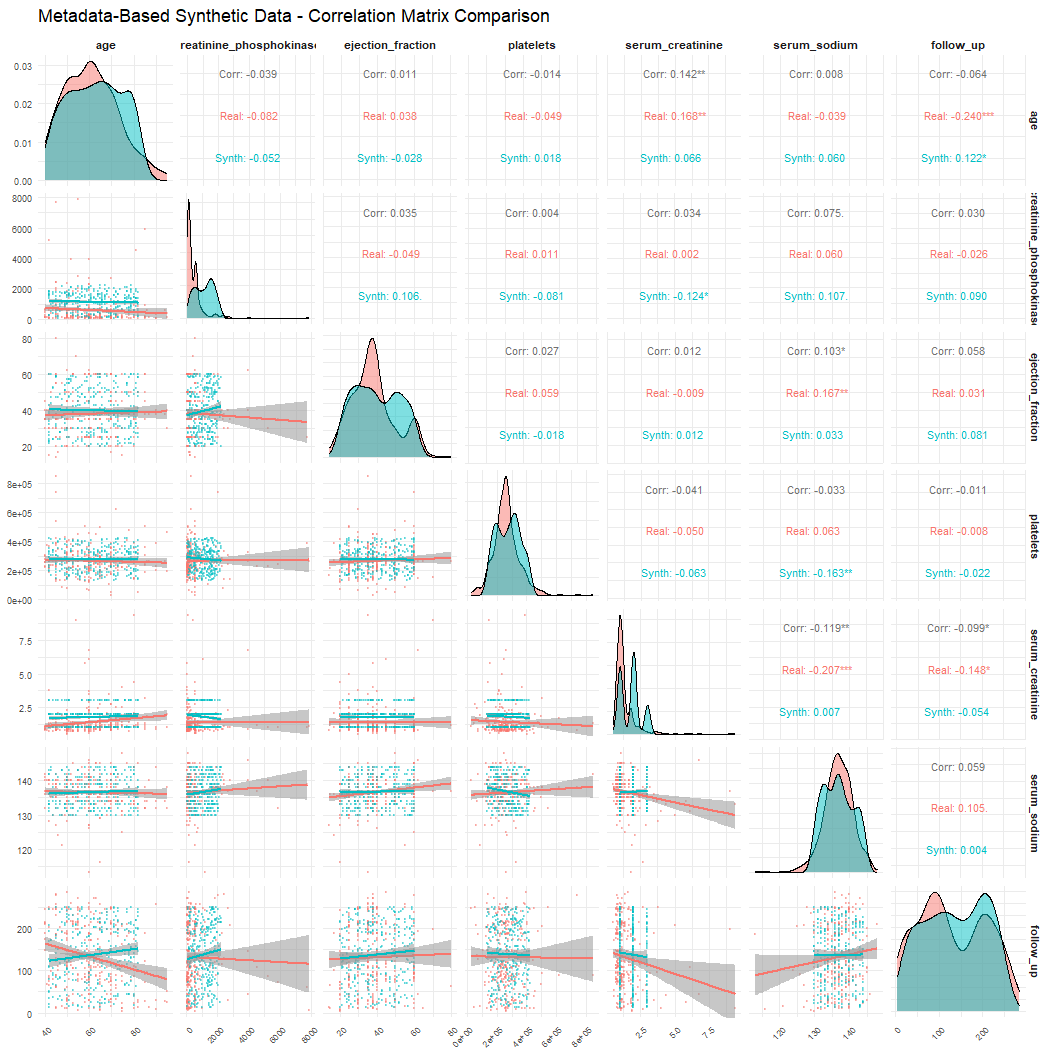
compare\_correlation\_matrices\_ggally(heart\_failure, syn\_cart\_1, "CART Imputation")



compare\_correlation\_matrices\_ggally(heart\_failure, syn\_data\_low\_fidelity\_synthpop, "Synthpop Low Fidelity")



compare\_correlation\_matrices\_ggally(heart\_failure, syn\_data\_metadata, "Metadata-Based Synthetic Data")



## 6 Fidelity Assessment

### 6.1 Descriptive Statistics

This section provides a comparative analysis of the descriptive statistics between the real dataset and the synthetic datasets generated using different methods.

#### 6.1.1 Real Dataset

The descriptive statistics for the real heart failure dataset include metrics such as mean, median, standard deviation, and range. This serves as the baseline for comparison with the synthetic datasets.

# Generate descriptive stats  
desc\_tbl <- describe(heart\_failure) %>%  
 as.data.frame() %>%  
 select(n, mean, sd, min, median, max) %>%  
 round(2) %>%  
 tibble::rownames\_to\_column("Variable")  
  
# Display as a neat table  
kable(  
 desc\_tbl,  
 caption = "Descriptive Statistics for the Heart Failure Dataset",  
 align = "lrrrrrr"  
)

Descriptive Statistics for the Heart Failure Dataset

| Variable | n | mean | sd | min | median | max |
| --- | --- | --- | --- | --- | --- | --- |
| age | 284 | 60.80 | 12.08 | 40.0 | 60.0 | 95.0 |
| anaemia\* | 299 | 1.43 | 0.50 | 1.0 | 1.0 | 2.0 |
| creatinine\_phosphokinase | 285 | 578.81 | 981.99 | 23.0 | 249.0 | 7861.0 |
| diabetes\* | 299 | 1.42 | 0.49 | 1.0 | 1.0 | 2.0 |
| ejection\_fraction | 287 | 38.00 | 11.92 | 14.0 | 38.0 | 80.0 |
| platelets | 285 | 265163.83 | 97508.63 | 25100.0 | 263358.0 | 850000.0 |
| serum\_creatinine | 281 | 1.39 | 1.05 | 0.5 | 1.1 | 9.4 |
| serum\_sodium | 279 | 136.59 | 4.49 | 113.0 | 137.0 | 148.0 |
| sex\* | 299 | 1.65 | 0.48 | 1.0 | 2.0 | 2.0 |
| smoking\* | 299 | 1.32 | 0.47 | 1.0 | 1.0 | 2.0 |
| hypertension\* | 299 | 1.35 | 0.48 | 1.0 | 1.0 | 2.0 |
| deceased\* | 299 | 1.32 | 0.47 | 1.0 | 1.0 | 2.0 |
| follow\_up | 299 | 130.26 | 77.61 | 4.0 | 115.0 | 285.0 |

#### 6.1.2 Parametric Imputation (MICE)

The synthetic data generated using the MICE framework is analyzed to compare its summary statistics with the real dataset. This will help evaluate how well the parametric method captures the key characteristics of the data.

# Descriptive stats for Parametric MICE synthetic dataset  
desc\_mice <- describe(syn\_data\_1) %>%  
 as.data.frame() %>%  
 select(n, mean, sd, min, median, max) %>%  
 round(2) %>%  
 tibble::rownames\_to\_column("Variable")  
  
# Display table  
kable(  
 desc\_mice,  
 caption = "Descriptive Statistics for Parametric MICE Synthetic Dataset",  
 align = "lrrrrrr"  
)

Descriptive Statistics for Parametric MICE Synthetic Dataset

| Variable | n | mean | sd | min | median | max |
| --- | --- | --- | --- | --- | --- | --- |
| synth\_age | 284 | 61.01 | 11.40 | 42 | 60 | 82 |
| synth\_anaemia\* | 299 | 1.45 | 0.50 | 1 | 1 | 2 |
| synth\_creatinine\_phosphokinase | 285 | 735.72 | 687.98 | 59 | 540 | 2221 |
| synth\_diabetes\* | 299 | 1.42 | 0.49 | 1 | 1 | 2 |
| synth\_ejection\_fraction | 287 | 38.44 | 10.70 | 20 | 38 | 60 |
| synth\_platelets | 285 | 257523.78 | 86505.14 | 132200 | 254087 | 421200 |
| synth\_serum\_creatinine | 281 | 1.70 | 0.77 | 1 | 2 | 3 |
| synth\_serum\_sodium | 279 | 135.78 | 3.96 | 130 | 135 | 144 |
| synth\_sex\* | 299 | 1.64 | 0.48 | 1 | 2 | 2 |
| synth\_smoking\* | 299 | 1.29 | 0.45 | 1 | 1 | 2 |
| synth\_hypertension\* | 299 | 1.42 | 0.49 | 1 | 1 | 2 |
| synth\_deceased\* | 299 | 1.31 | 0.47 | 1 | 1 | 2 |
| synth\_follow\_up | 299 | 132.74 | 71.11 | 13 | 132 | 250 |

#### 6.1.3 Non-Parametric Imputation (CART)

The CART method is a non-parametric technique that handles non-linear relationships. The descriptive statistics of the CART-imputed synthetic data provide insight into how this method captures the dataset’s distribution.

# Descriptive stats for CART synthetic dataset  
desc\_cart <- describe(syn\_cart\_1) %>%  
 as.data.frame() %>%  
 select(n, mean, sd, min, median, max) %>%  
 round(2) %>%  
 tibble::rownames\_to\_column("Variable")  
  
# Display table  
kable(  
 desc\_cart,  
 caption = "Descriptive Statistics for CART Synthetic Dataset",  
 align = "lrrrrrr"  
)

Descriptive Statistics for CART Synthetic Dataset

| Variable | n | mean | sd | min | median | max |
| --- | --- | --- | --- | --- | --- | --- |
| synth\_age | 284 | 60.81 | 11.88 | 42 | 60 | 82 |
| synth\_anaemia\* | 299 | 1.42 | 0.49 | 1 | 1 | 2 |
| synth\_creatinine\_phosphokinase | 285 | 469.20 | 547.58 | 59 | 231 | 2221 |
| synth\_diabetes\* | 299 | 1.46 | 0.50 | 1 | 1 | 2 |
| synth\_ejection\_fraction | 287 | 38.56 | 11.85 | 20 | 35 | 60 |
| synth\_platelets | 285 | 265126.32 | 76944.75 | 132200 | 263358 | 421200 |
| synth\_serum\_creatinine | 281 | 1.28 | 0.56 | 1 | 1 | 3 |
| synth\_serum\_sodium | 279 | 136.79 | 3.67 | 130 | 137 | 144 |
| synth\_sex\* | 299 | 1.67 | 0.47 | 1 | 2 | 2 |
| synth\_smoking\* | 299 | 1.37 | 0.48 | 1 | 1 | 2 |
| synth\_hypertension\* | 299 | 1.38 | 0.49 | 1 | 1 | 2 |
| synth\_deceased\* | 299 | 1.33 | 0.47 | 1 | 1 | 2 |
| synth\_follow\_up | 299 | 132.53 | 75.17 | 13 | 117 | 250 |

#### 6.1.4 Synthpop-Based Synthetic Data

The Synthpop package generates synthetic data aimed at preserving data privacy. Descriptive statistics for this dataset help assess how well the key attributes of the real data are preserved.

# Descriptive stats for Synthpop synthetic dataset  
desc\_synthpop <- describe(syn\_data\_low\_fidelity\_synthpop) %>%  
 as.data.frame() %>%  
 select(n, mean, sd, min, median, max) %>%  
 round(2) %>%  
 tibble::rownames\_to\_column("Variable")  
  
# Display as a clean table  
kable(  
 desc\_synthpop,  
 caption = "Descriptive Statistics for Synthpop Synthetic Dataset",  
 align = "lrrrrrr"  
)

Descriptive Statistics for Synthpop Synthetic Dataset

| Variable | n | mean | sd | min | median | max |
| --- | --- | --- | --- | --- | --- | --- |
| synth\_age | 271 | 61.15 | 11.13 | 42 | 60 | 82 |
| synth\_anaemia\* | 299 | 1.47 | 0.50 | 1 | 1 | 2 |
| synth\_creatinine\_phosphokinase | 269 | 435.35 | 498.44 | 59 | 232 | 2221 |
| synth\_diabetes\* | 299 | 1.37 | 0.48 | 1 | 1 | 2 |
| synth\_ejection\_fraction | 285 | 36.73 | 10.93 | 20 | 35 | 60 |
| synth\_platelets | 268 | 269756.34 | 77738.41 | 132200 | 263358 | 421200 |
| synth\_serum\_creatinine | 264 | 1.33 | 0.63 | 1 | 1 | 3 |
| synth\_serum\_sodium | 265 | 136.92 | 3.69 | 130 | 137 | 144 |
| synth\_sex\* | 299 | 1.65 | 0.48 | 1 | 2 | 2 |
| synth\_smoking\* | 299 | 1.28 | 0.45 | 1 | 1 | 2 |
| synth\_hypertension\* | 299 | 1.37 | 0.48 | 1 | 1 | 2 |
| synth\_deceased\* | 299 | 1.31 | 0.46 | 1 | 1 | 2 |
| synth\_follow\_up | 299 | 121.22 | 74.28 | 13 | 108 | 250 |

#### 6.1.5 Metadata-Based Synthetic Data

The synthetic data generated using metadata ensures that variable structures conform to the data dictionary. The summary statistics here show how well the dataset reflects the real’s attributes based on predefined metadata.

# Descriptive stats for Metadata-Based synthetic dataset  
desc\_metadata <- describe(syn\_data\_metadata) %>%  
 as.data.frame() %>%  
 select(n, mean, sd, min, median, max) %>%  
 round(2) %>%  
 tibble::rownames\_to\_column("Variable")  
  
# Display as a clean table  
kable(  
 desc\_metadata,  
 caption = "Descriptive Statistics for Metadata-Based Synthetic Dataset",  
 align = "lrrrrrr"  
)

Descriptive Statistics for Metadata-Based Synthetic Dataset

| Variable | n | mean | sd | min | median | max |
| --- | --- | --- | --- | --- | --- | --- |
| synth\_age | 284 | 62.57 | 11.98 | 42 | 62.5 | 82 |
| synth\_anaemia\* | 299 | 1.52 | 0.50 | 1 | 2.0 | 2 |
| synth\_creatinine\_phosphokinase | 285 | 1129.11 | 594.90 | 67 | 1185.0 | 2216 |
| synth\_diabetes\* | 299 | 1.48 | 0.50 | 1 | 1.0 | 2 |
| synth\_ejection\_fraction | 287 | 39.81 | 12.08 | 20 | 38.0 | 60 |
| synth\_platelets | 285 | 275686.74 | 79876.34 | 134054 | 280442.0 | 421118 |
| synth\_serum\_creatinine | 281 | 1.79 | 0.71 | 1 | 2.0 | 3 |
| synth\_serum\_sodium | 279 | 136.69 | 4.20 | 130 | 137.0 | 144 |
| synth\_sex\* | 299 | 1.49 | 0.50 | 1 | 1.0 | 2 |
| synth\_smoking\* | 299 | 1.51 | 0.50 | 1 | 2.0 | 2 |
| synth\_hypertension\* | 299 | 1.51 | 0.50 | 1 | 2.0 | 2 |
| synth\_deceased\* | 299 | 1.47 | 0.50 | 1 | 1.0 | 2 |
| synth\_follow\_up | 299 | 136.77 | 70.26 | 13 | 139.0 | 249 |

By comparing the descriptive statistics, you can evaluate how closely the synthetic datasets resemble the real dataset across key summary metrics. This is a critical step in assessing the quality and usability of synthetic data. Histogram Similarity Score ### Variable Exploration

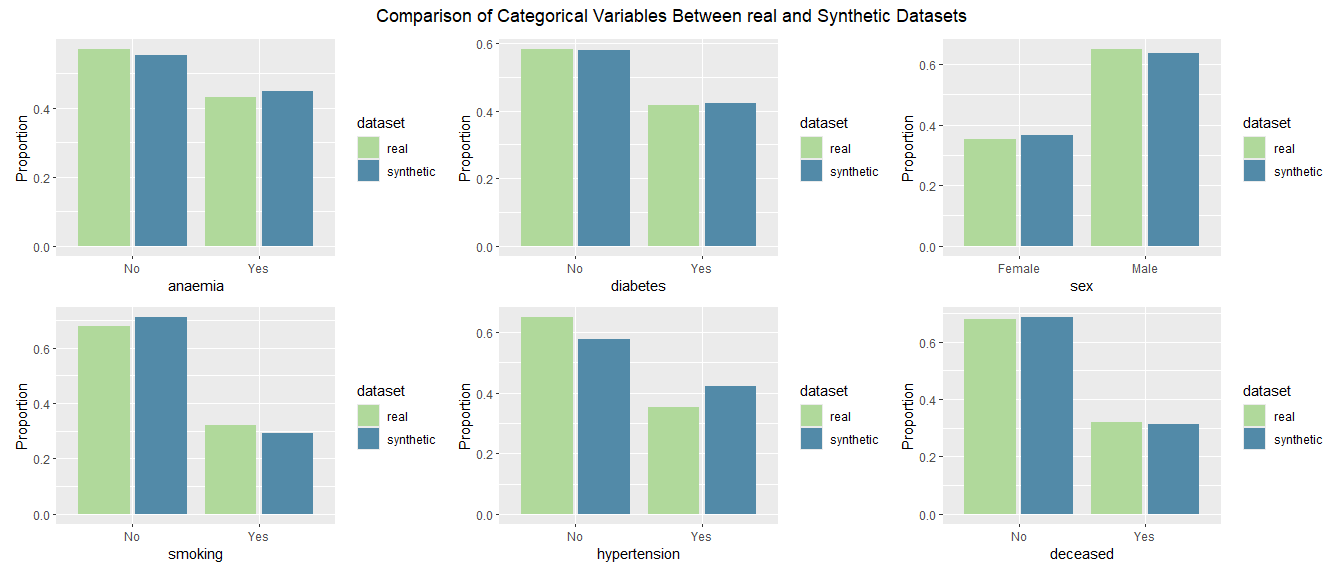
#### 6.1.6 Parametric Imputation (MICE)

The parametric imputation using the MICE framework generates synthetic data by imputing missing values based on multivariate normal distributions.

##### 6.1.6.1 Bar Plots for Categorical Variables

To evaluate the preservation of categorical variable distributions in the synthetic data, we generate bar plots comparing the proportions of categorical variables between the real and synthetic datasets.

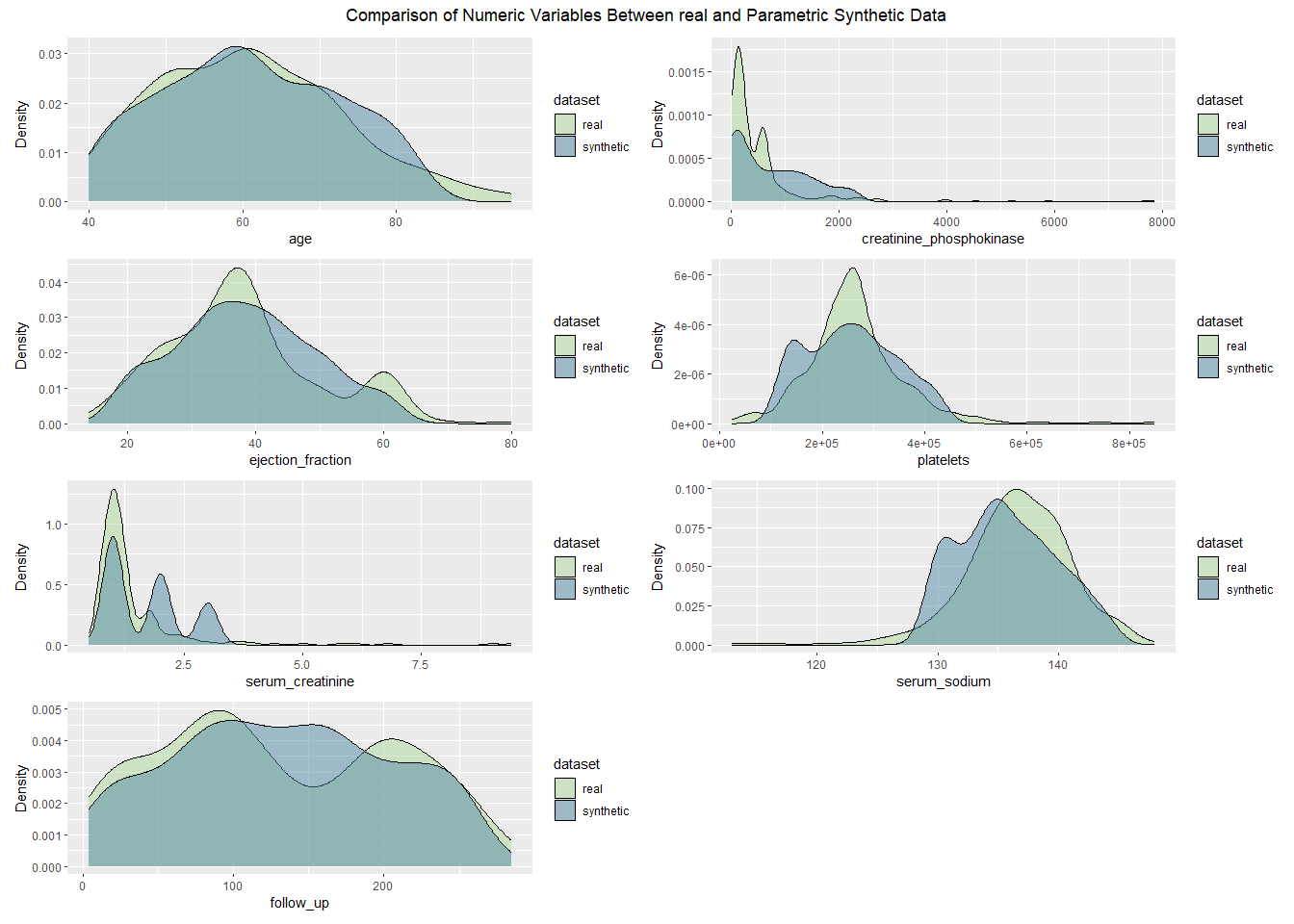
# Combine real and parametric MICE synthetic datasets  
heart\_failure$dataset <- "real"  
syn\_data\_1$dataset <- "synthetic"  
  
# Adjust the column names for synthetic data by removing the 'synth\_' prefix  
colnames(syn\_data\_1) <- gsub("synth\_", "", colnames(syn\_data\_1))  
  
# Combine both datasets (real and synthetic)  
combined\_data\_param <- bind\_rows(heart\_failure, syn\_data\_1)  
  
# Bar plots for categorical variables  
bar\_plots <- colnames(heart\_failure)[map\_lgl(heart\_failure, is.factor)] %>%  
 map(~ ggplot(combined\_data\_param, aes\_string(.x, fill = 'dataset', group = 'dataset')) +  
 geom\_bar(aes(y = ..prop..), position = position\_dodge2(), stat = "count") +  
 scale\_fill\_manual(values = c("real" = "#B0D99B", "synthetic" = "#528AA8")) +  
 labs(x = .x, y = "Proportion")) %>%  
 patchwork::wrap\_plots() +  
 plot\_annotation(title = "Comparison of Categorical Variables Between real and Synthetic Datasets",  
 theme = theme(plot.title = element\_text(hjust = 0.5)))  
  
# Print the combined plot with one main title  
print(bar\_plots)



##### 6.1.6.2 Density Plots for Numeric Variables

The density plots illustrate the distribution of numeric variables between the real dataset and the parametric MICE synthetic data. The goal is to ensure that the synthetic data closely follows the real data’s numeric distribution.

# Density plots for numeric variables (parametric imputed data)  
density\_plots\_param <- colnames(heart\_failure)[map\_lgl(heart\_failure, is.numeric)] %>%  
 map(~ ggplot(combined\_data\_param, aes\_string(.x, fill = 'dataset', group = 'dataset')) +  
 geom\_density(alpha = 0.5) +  
 scale\_fill\_manual(values = c("real" = "#B0D99B", "synthetic" = "#528AA8")) +  
 labs(x = .x, y = "Density")) %>%  
 patchwork::wrap\_plots(ncol = 2) + # Arrange plots in 2 columns  
 plot\_annotation(title = "Comparison of Numeric Variables Between real and Parametric Synthetic Data",  
 theme = theme(plot.title = element\_text(hjust = 0.5)))  
  
# Print the combined density plots with one main title  
print(density\_plots\_param)



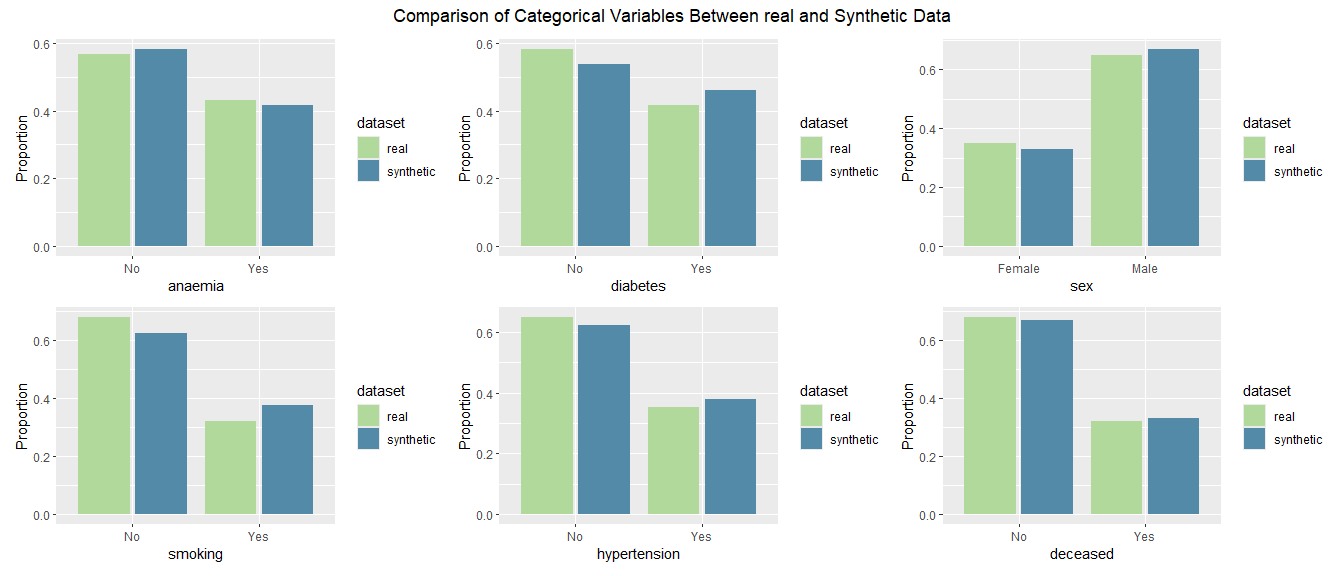
#### 6.1.7 Non-Parametric Imputation (CART)

CART is a non-parametric approach for generating synthetic data that captures complex relationships between variables without making distributional assumptions.

##### 6.1.7.1 Bar Plots for Categorical Variables

The bar plots compare the distribution of categorical variables in the real and synthetic datasets generated using the CART method.

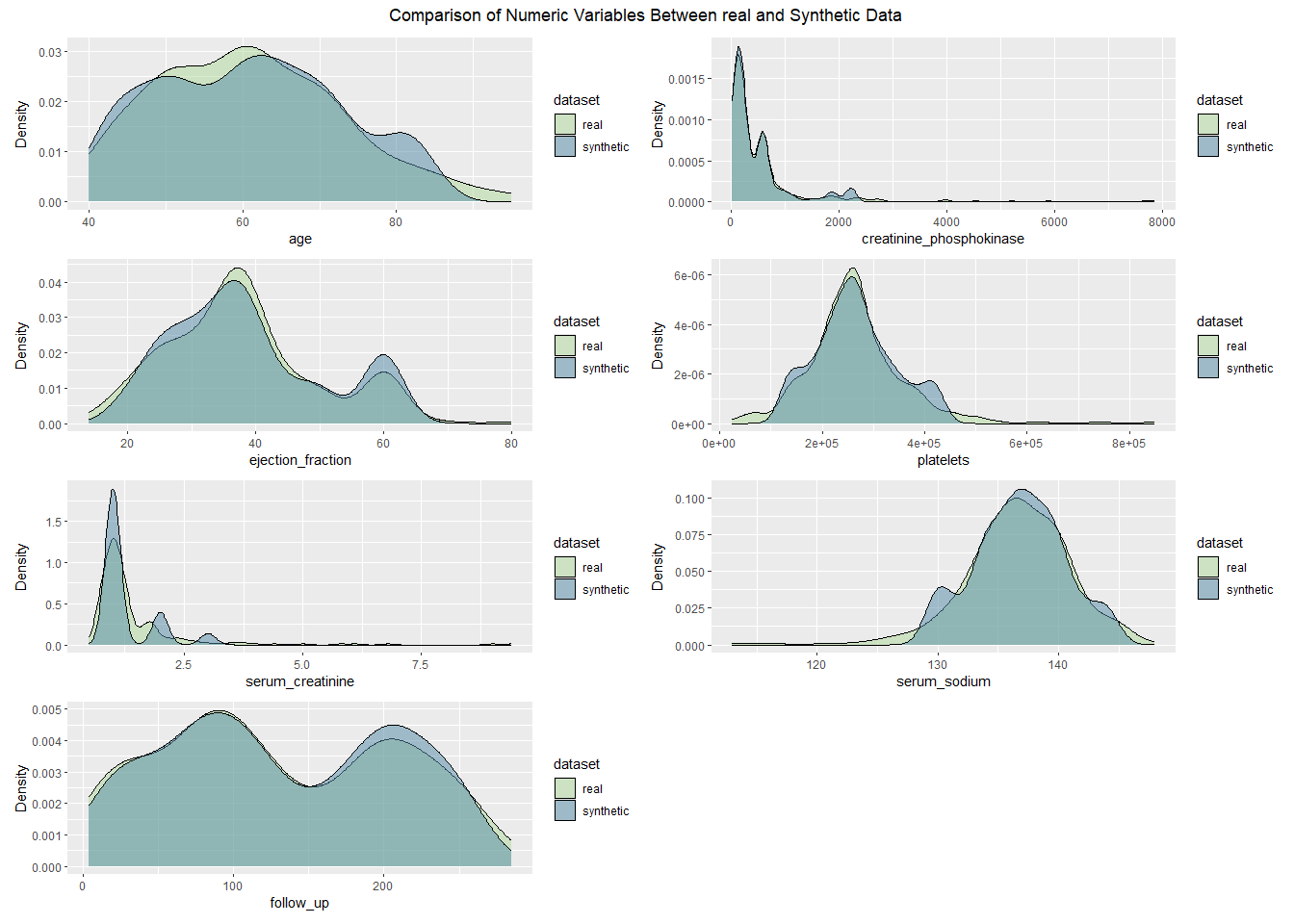
# Combine real and non-parametric CART synthetic datasets  
syn\_cart\_1$dataset <- "synthetic"  
  
# Adjust the column names for synthetic data by removing the 'synth\_' prefix  
colnames(syn\_cart\_1) <- gsub("synth\_", "", colnames(syn\_cart\_1))  
  
# Combine both datasets (real and synthetic)  
combined\_data\_cart <- bind\_rows(heart\_failure, syn\_cart\_1)  
  
# Bar plots for categorical variables (CART-imputed data)  
bar\_plots\_cart <- colnames(heart\_failure)[map\_lgl(heart\_failure, is.factor)] %>%  
 map(~ ggplot(combined\_data\_cart, aes\_string(.x, fill = 'dataset', group = 'dataset')) +  
 geom\_bar(aes(y = ..prop..), position = position\_dodge2(), stat = "count") +  
 scale\_fill\_manual(values = c("real" = "#B0D99B", "synthetic" = "#528AA8")) +  
 labs(x = .x, y = "Proportion")) %>%  
 patchwork::wrap\_plots() +  
 plot\_annotation(title = "Comparison of Categorical Variables Between real and Synthetic Data",  
 theme = theme(plot.title = element\_text(hjust = 0.5)))  
  
# Print the combined plot with one main title  
print(bar\_plots\_cart)



##### 6.1.7.2 Density Plots for Numeric Variables

The density plots display how well the numeric variables in the CART-imputed synthetic data align with the real dataset.

# Density plots for numeric variables (CART-imputed data)  
density\_plots\_cart <- colnames(heart\_failure)[map\_lgl(heart\_failure, is.numeric)] %>%  
 map(~ ggplot(combined\_data\_cart, aes\_string(.x, fill = 'dataset', group = 'dataset')) +  
 geom\_density(alpha = 0.5) +  
 scale\_fill\_manual(values = c("real" = "#B0D99B", "synthetic" = "#528AA8")) +  
 labs(x = .x, y = "Density")) %>%  
 patchwork::wrap\_plots(ncol = 2) + # Arrange plots in 2 columns  
 plot\_annotation(title = "Comparison of Numeric Variables Between real and Synthetic Data",  
 theme = theme(plot.title = element\_text(hjust = 0.5)))  
  
# Print the combined density plots with one main title  
print(density\_plots\_cart)



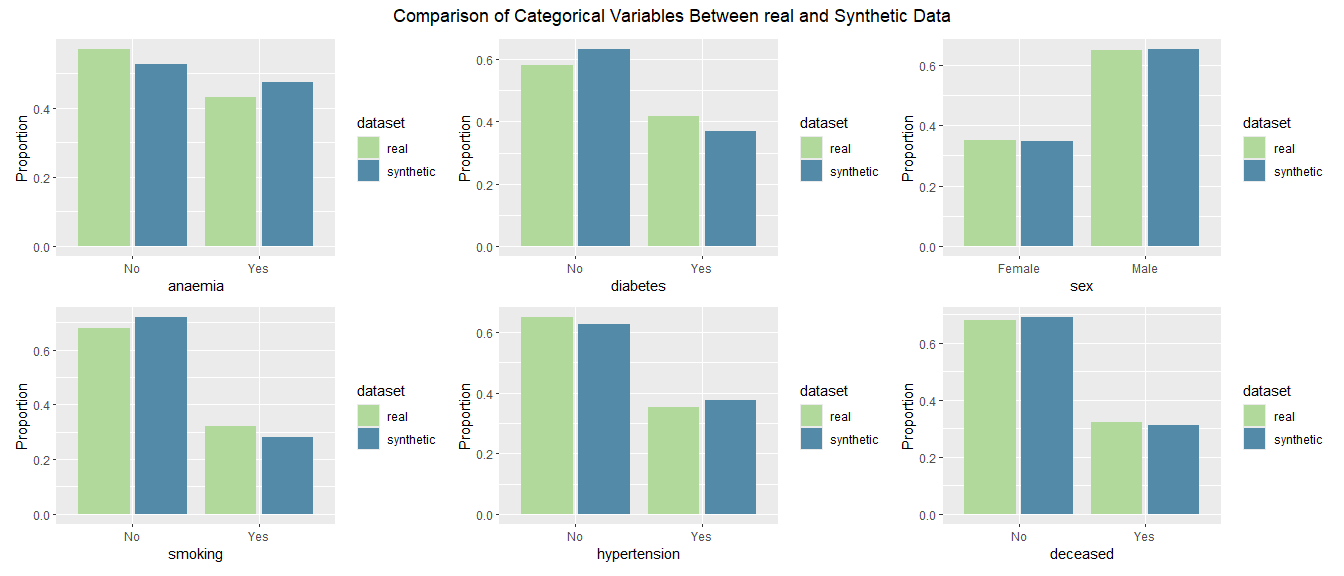
#### 6.1.8 Synthpop-Based Synthetic Data

Synthpop is used to generate synthetic datasets that closely follow the distributions of the real data for privacy-preserving data sharing.

##### 6.1.8.1 Bar Plots for Categorical Variables

These bar plots assess how well Synthpop-based synthetic data maintains the categorical distributions of the real dataset.

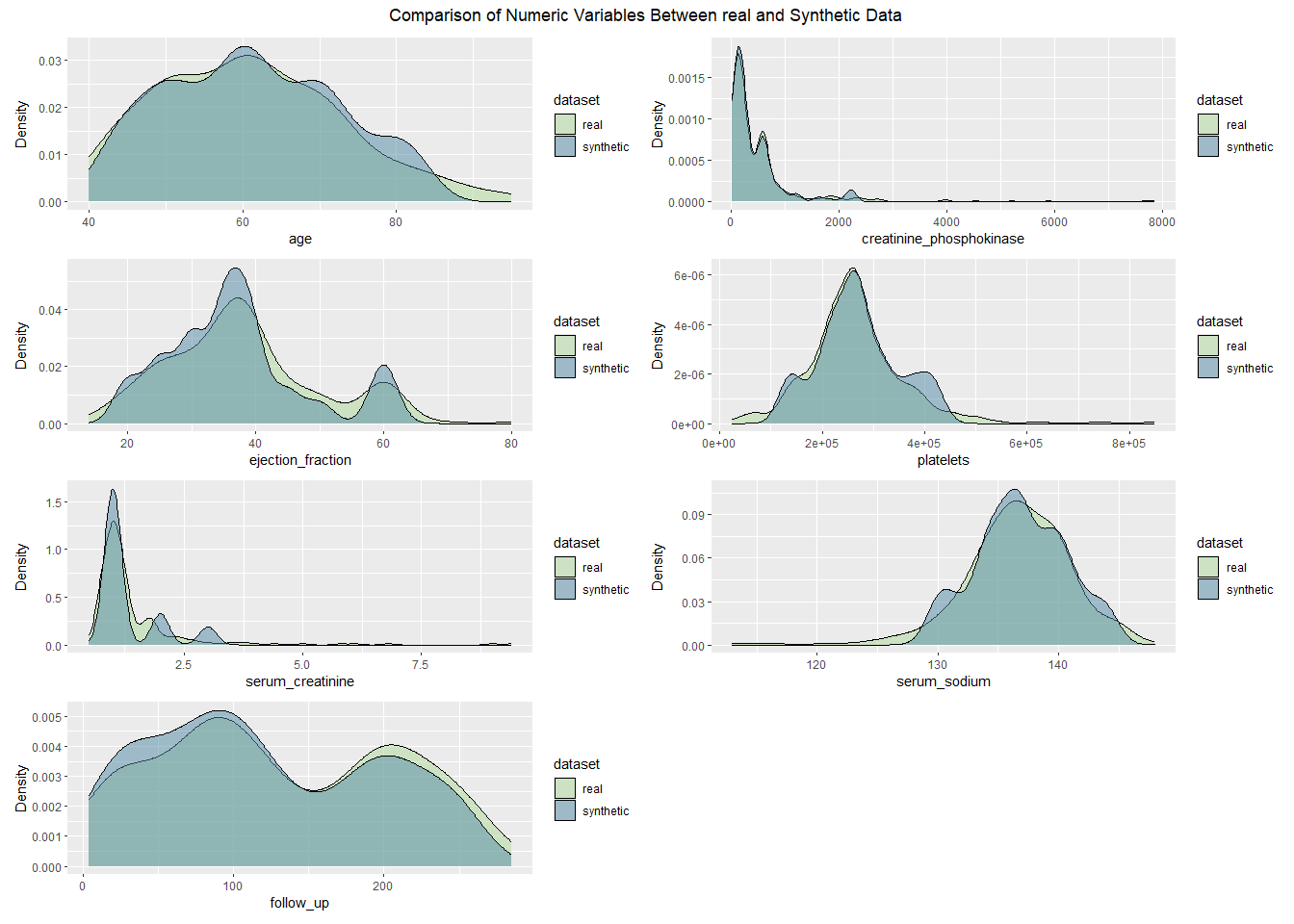
# Combine real and Synthpop-based synthetic datasets  
heart\_failure$dataset <- "real"  
syn\_data\_low\_fidelity\_synthpop$dataset <- "synthetic"  
  
# Adjust the column names for synthetic data by removing the 'synth\_' prefix  
colnames(syn\_data\_low\_fidelity\_synthpop) <- gsub("synth\_", "", colnames(syn\_data\_low\_fidelity\_synthpop))  
  
# Combine both datasets (real and synthetic)  
combined\_data\_synthpop <- bind\_rows(heart\_failure, syn\_data\_low\_fidelity\_synthpop)  
  
# Bar plots for categorical variables (Synthpop-based synthetic data)  
bar\_plots\_synthpop <- colnames(heart\_failure)[map\_lgl(heart\_failure, is.factor)] %>%  
 map(~ ggplot(combined\_data\_synthpop, aes\_string(.x, fill = 'dataset', group = 'dataset')) +  
 geom\_bar(aes(y = ..prop..), position = position\_dodge2(), stat = "count") +  
 scale\_fill\_manual(values = c("real" = "#B0D99B", "synthetic" = "#528AA8")) +  
 labs(x = .x, y = "Proportion")) %>%  
 patchwork::wrap\_plots() +  
 plot\_annotation(title = "Comparison of Categorical Variables Between real and Synthetic Data",  
 theme = theme(plot.title = element\_text(hjust = 0.5)))  
  
# Print the combined plot with one main title  
print(bar\_plots\_synthpop)



##### 6.1.8.2 Density Plots for Numeric Variables

The density plots illustrate how closely Synthpop-based synthetic data matches the real dataset for numeric variables.

# Combine real and Synthpop-based synthetic datasets  
heart\_failure$dataset <- "real"  
syn\_data\_low\_fidelity\_synthpop$dataset <- "synthetic"  
combined\_data\_synthpop <- bind\_rows(heart\_failure, syn\_data\_low\_fidelity\_synthpop)  
  
# Density plots for numeric variables (Synthpop-based synthetic data)  
density\_plots\_synthpop <- colnames(heart\_failure)[map\_lgl(heart\_failure, is.numeric)] %>%  
 map(~ ggplot(combined\_data\_synthpop, aes\_string(.x, fill = 'dataset', group = 'dataset')) +  
 geom\_density(alpha = 0.5) +  
 scale\_fill\_manual(values = c("real" = "#B0D99B", "synthetic" = "#528AA8")) +  
 labs(x = .x, y = "Density")) %>%  
 patchwork::wrap\_plots(ncol = 2) + # Arrange plots in 2 columns  
 plot\_annotation(title = "Comparison of Numeric Variables Between real and Synthetic Data",  
 theme = theme(plot.title = element\_text(hjust = 0.5)))  
  
# Print the combined density plots with one main title  
print(density\_plots\_synthpop)



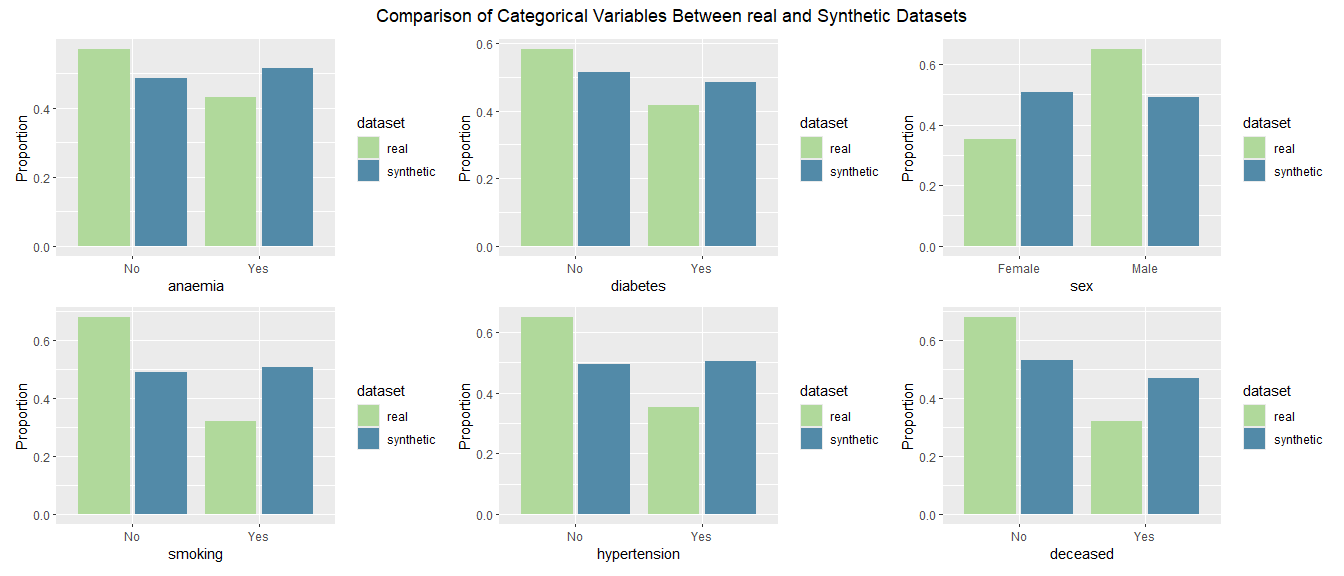
#### 6.1.9 Metadata-Based Synthetic Data

Metadata-based synthetic data generation uses a predefined data dictionary to ensure that the synthetic data follows the correct structure and distribution.

##### 6.1.9.1 Bar Plots for Categorical Variables

We compare the categorical variables of the real and metadata-based synthetic datasets to assess distributional alignment.

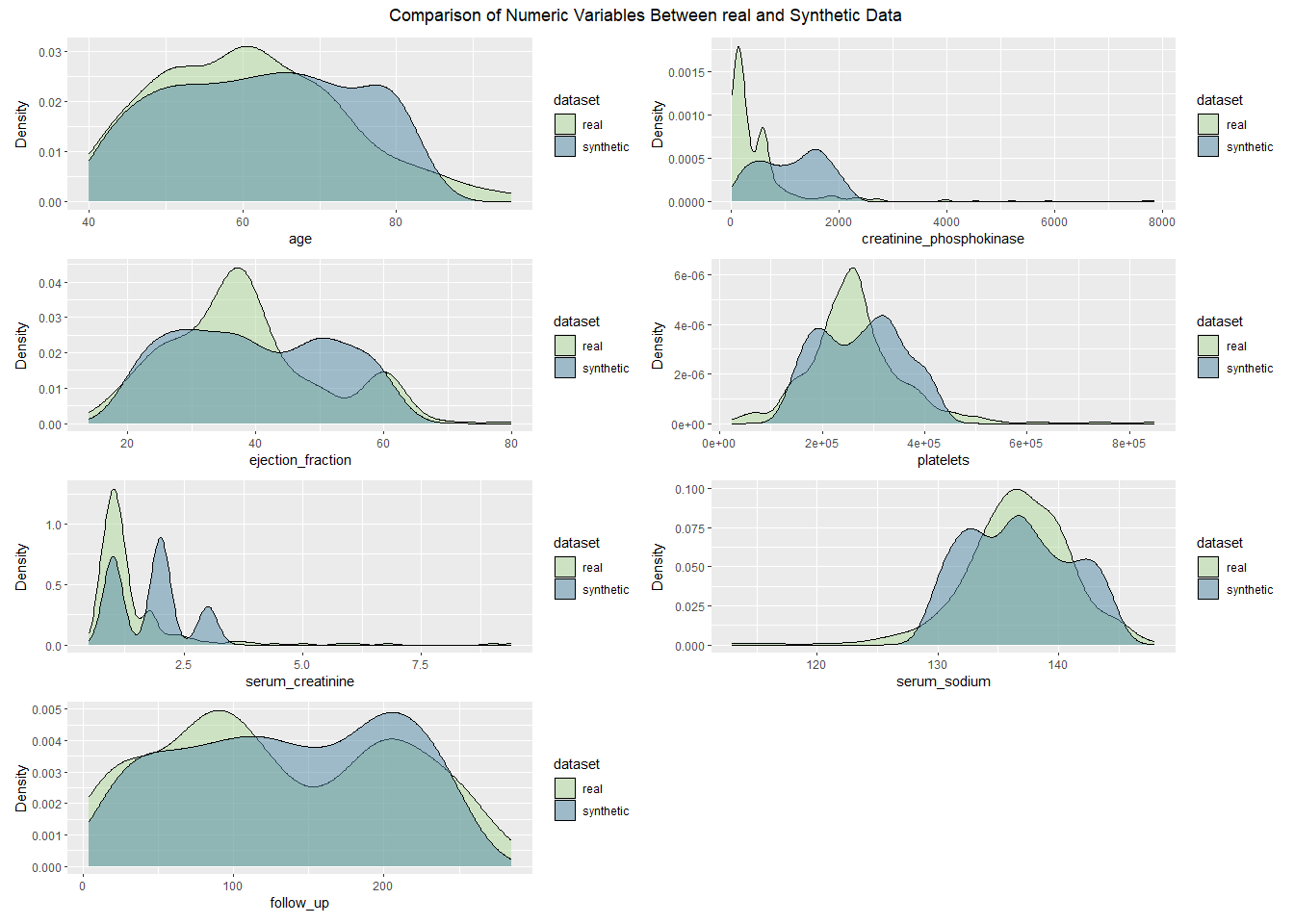
# Combine real and Metadata-based synthetic datasets  
heart\_failure$dataset <- "real"  
syn\_data\_metadata$dataset <- "synthetic"  
  
# Adjust the column names for synthetic data by removing the 'synth\_' prefix  
colnames(syn\_data\_metadata) <- gsub("synth\_", "", colnames(syn\_data\_metadata))  
  
# Combine both datasets (real and synthetic)  
combined\_data\_metadata <- bind\_rows(heart\_failure, syn\_data\_metadata)  
  
# Bar plots for categorical variables (Metadata-based synthetic data)  
bar\_plots\_metadata <- colnames(heart\_failure)[map\_lgl(heart\_failure, is.factor)] %>%  
 map(~ ggplot(combined\_data\_metadata, aes\_string(.x, fill = 'dataset', group = 'dataset')) +  
 geom\_bar(aes(y = ..prop..), position = position\_dodge2(), stat = "count") +  
 scale\_fill\_manual(values = c("real" = "#B0D99B", "synthetic" = "#528AA8")) +  
 labs(x = .x, y = "Proportion")) %>%  
 patchwork::wrap\_plots() +  
 plot\_annotation(title = "Comparison of Categorical Variables Between real and Synthetic Datasets",  
 theme = theme(plot.title = element\_text(hjust = 0.5)))  
  
# Print the combined plot with one main title  
print(bar\_plots\_metadata)



##### 6.1.9.2 Density Plots for Numeric Variables

The density plots for numeric variables compare the distribution of numeric values between the real and metadata-based synthetic datasets.

# Combine real and Metadata-based synthetic datasets  
heart\_failure$dataset <- "real"  
syn\_data\_metadata$dataset <- "synthetic"  
combined\_data\_metadata <- bind\_rows(heart\_failure, syn\_data\_metadata)  
  
# Density plots for numeric variables (Metadata-based synthetic data)  
density\_plots\_metadata <- colnames(heart\_failure)[map\_lgl(heart\_failure, is.numeric)] %>%  
 map(~ ggplot(combined\_data\_metadata, aes\_string(.x, fill = 'dataset', group = 'dataset')) +  
 geom\_density(alpha = 0.5) +  
 scale\_fill\_manual(values = c("real" = "#B0D99B", "synthetic" = "#528AA8")) +  
 labs(x = .x, y = "Density")) %>%  
 patchwork::wrap\_plots(ncol = 2) + # Arrange plots in 2 columns  
 plot\_annotation(title = "Comparison of Numeric Variables Between real and Synthetic Data",  
 theme = theme(plot.title = element\_text(hjust = 0.5)))  
  
# Print the combined density plots with one main title  
print(density\_plots\_metadata)



### 6.2 Histogram Similarity Score

The Histogram Similarity Score measures how closely the marginal distributions of the synthetic data match the real data. Marginal distributions are critical for assessing whether key data characteristics such as spread, central tendency (mean/median), and shape (skewness/kurtosis) are preserved.

Ideal Ranges for Histogram Similarity:

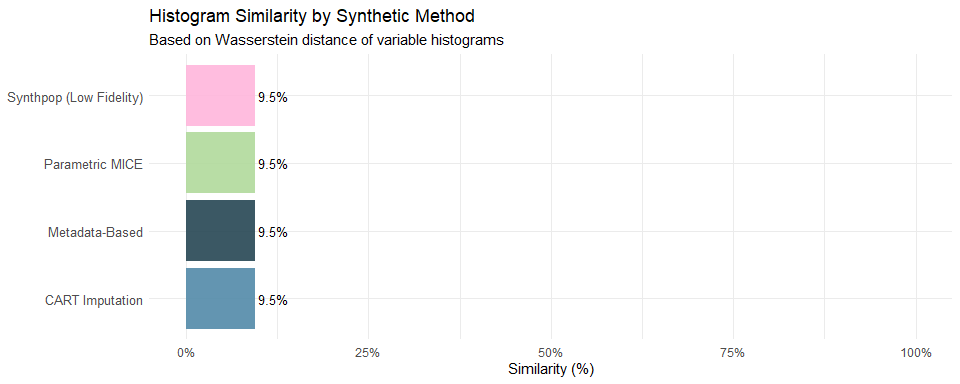
* **Perfect Match:** 1 (100%) indicates a perfect match between the distributions.
* **Good Match:** 0.85–1 (85% to 100%) shows strong similarity.
* **Moderate Match:** 0.65–0.85 suggests some differences that may need addressing.
* **Poor Match:** Below 0.65 indicates significant deviation from the real data.

# Custom palette for datasets  
dataset\_palette <- c(  
 "Parametric MICE" = "#B0D99B",  
 "CART Imputation" = "#528AA8",  
 "Synthpop (Low Fidelity)" = "#FFB6DB",  
 "Metadata-Based" = "#264653"  
)  
  
# Define function once  
calculate\_histogram\_similarity <- function(real\_data, synthetic\_data) {  
 real\_numeric <- real\_data[sapply(real\_data, is.numeric)]  
 synthetic\_numeric <- synthetic\_data[sapply(synthetic\_data, is.numeric)]  
   
 similarity\_scores <- sapply(names(real\_numeric), function(var) {  
 real\_hist <- hist(real\_numeric[[var]], plot = FALSE)  
 synthetic\_hist <- hist(synthetic\_numeric[[var]], plot = FALSE)  
   
 # Wasserstein distance between histogram midpoints  
 wasserstein\_distance <- transport::wasserstein1d(real\_hist$mids, synthetic\_hist$mids)  
 1 / (1 + wasserstein\_distance) # similarity score  
 })  
   
 mean(similarity\_scores)  
}  
  
# Collect results in a tibble (convert to percentages)  
hist\_sim\_tbl <- tibble::tibble(  
 Dataset = c("Parametric MICE", "CART Imputation", "Synthpop (Low Fidelity)", "Metadata-Based"),  
 Histogram\_Similarity = c(  
 calculate\_histogram\_similarity(heart\_failure, syn\_data\_1),  
 calculate\_histogram\_similarity(heart\_failure, syn\_cart\_1),  
 calculate\_histogram\_similarity(heart\_failure, syn\_data\_low\_fidelity\_synthpop),  
 calculate\_histogram\_similarity(heart\_failure, syn\_data\_metadata)  
 )  
) %>%  
 mutate(Histogram\_Similarity = round(Histogram\_Similarity \* 100, 1)) # percentage  
  
# Display as table  
kable(  
 hist\_sim\_tbl,  
 caption = "Histogram Similarity Scores (%): higher = more similar to real data",  
 align = "lr"  
)

Histogram Similarity Scores (%): higher = more similar to real data

| Dataset | Histogram\_Similarity |
| --- | --- |
| Parametric MICE | 9.5 |
| CART Imputation | 9.5 |
| Synthpop (Low Fidelity) | 9.5 |
| Metadata-Based | 9.5 |

# Horizontal bar chart with custom colours  
ggplot(hist\_sim\_tbl, aes(y = Dataset, x = Histogram\_Similarity, fill = Dataset)) +  
 geom\_col(alpha = 0.9, show.legend = FALSE) +  
 geom\_text(aes(label = paste0(Histogram\_Similarity, "%")),   
 hjust = -0.1, size = 3.3) +  
 scale\_fill\_manual(values = dataset\_palette) +  
 scale\_x\_continuous(labels = scales::percent\_format(scale = 1), limits = c(0, 100)) +  
 labs(  
 title = "Histogram Similarity by Synthetic Method",  
 subtitle = "Based on Wasserstein distance of variable histograms",  
 x = "Similarity (%)", y = NULL  
 ) +  
 theme\_minimal() +  
 theme(axis.text.y = element\_text(size = 10))



### 6.3 Mutual Information Score

The Mutual Information (MI) Score measures the shared information between pairs of features in the real and synthetic datasets, assessing how well relationships between variables are preserved.

Ideal Ranges for Mutual Information:

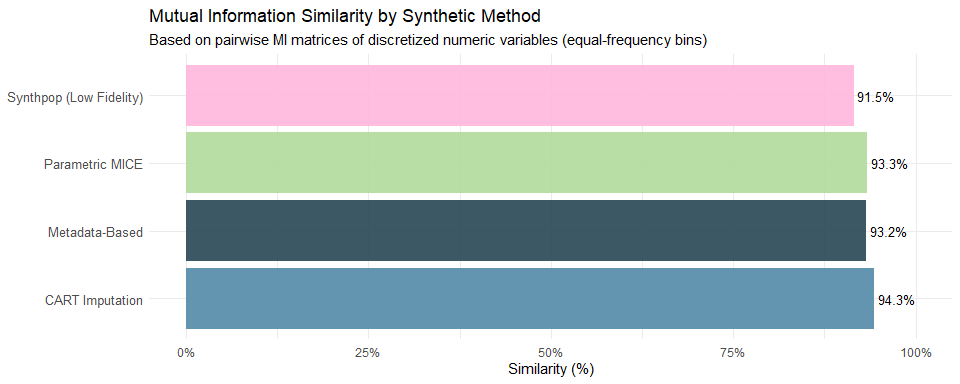
* **Perfect Alignment:** MI scores for synthetic data should ideally match the real within 10-20% for critical feature pairs.
* **Lower MI:** Consistently lower MI scores suggest that noise has been added, weakening relationships.
* **Overfitting Risk:** Higher MI scores than the real may indicate overfitting, increasing re-identification risk.

# Custom palette for datasets  
dataset\_palette <- c(  
 "Parametric MICE" = "#B0D99B",  
 "CART Imputation" = "#528AA8",  
 "Synthpop (Low Fidelity)" = "#FFB6DB",  
 "Metadata-Based" = "#264653"  
)  
  
# ------- Helpers -------  
# Strip 'synth\_' so variable names align with the real dataset  
strip\_synth\_prefix <- function(df) {  
 names(df) <- sub("^synth\_", "", names(df))  
 df  
}  
  
# Compute pairwise MI matrix for a data.frame of discretized columns  
compute\_mi\_matrix <- function(df\_disc) {  
 p <- ncol(df\_disc)  
 if (p < 2) return(matrix(numeric(0), nrow = p, ncol = p,  
 dimnames = list(colnames(df\_disc), colnames(df\_disc))))  
 M <- matrix(0, nrow = p, ncol = p,  
 dimnames = list(colnames(df\_disc), colnames(df\_disc)))  
 for (i in seq\_len(p)) {  
 for (j in i:p) {  
 # safe MI (skip NA rows; zero if not enough variation)  
 ok <- is.finite(df\_disc[[i]]) & is.finite(df\_disc[[j]])  
 xi <- df\_disc[[i]][ok]; xj <- df\_disc[[j]][ok]  
 mi <- if (length(unique(xi)) < 2 || length(unique(xj)) < 2) 0 else  
 suppressWarnings(infotheo::mutinformation(xi, xj))  
 M[i, j] <- mi  
 M[j, i] <- mi  
 }  
 }  
 M  
}  
  
# Compare two MI matrices (upper triangle only) -> 0–1 similarity score (higher = closer)  
mi\_similarity\_score <- function(M\_real, M\_synth) {  
 if (length(M\_real) == 0 || length(M\_synth) == 0) return(NA\_real\_)  
 common <- intersect(colnames(M\_real), colnames(M\_synth))  
 if (length(common) < 2) return(NA\_real\_)  
 R <- M\_real[common, common, drop = FALSE]  
 S <- M\_synth[common, common, drop = FALSE]  
 ut <- upper.tri(R, diag = FALSE)  
 mean\_abs\_diff <- mean(abs(R[ut] - S[ut]))  
 1 / (1 + mean\_abs\_diff)  
}  
  
# Discretize numeric columns (equal-frequency bins); keep only shared numeric vars  
Mutual\_Information <- function(real\_df, synth\_df, nbins = 10) {  
 real\_num <- dplyr::select(real\_df, where(is.numeric))  
 synth\_num <- strip\_synth\_prefix(synth\_df) |> dplyr::select(where(is.numeric))  
  
 common <- intersect(names(real\_num), names(synth\_num))  
 if (length(common) < 2) return(NA\_real\_)  
  
 real\_num <- real\_num[common]  
 synth\_num <- synth\_num[common]  
  
 # Discretize numerics (returns integer codes)  
 real\_disc <- as.data.frame(lapply(real\_num, \(x) infotheo::discretize(x, disc = "equalfreq", nbins = nbins)))  
 synth\_disc <- as.data.frame(lapply(synth\_num, \(x) infotheo::discretize(x, disc = "equalfreq", nbins = nbins)))  
  
 # MI matrices and similarity score  
 M\_real <- compute\_mi\_matrix(real\_disc)  
 M\_synth <- compute\_mi\_matrix(synth\_disc)  
 mi\_similarity\_score(M\_real, M\_synth)  
}  
  
# ------- Compute scores and present nicely (percentages) -------  
mi\_results <- tibble::tibble(  
 Dataset = c("Parametric MICE", "CART Imputation", "Synthpop (Low Fidelity)", "Metadata-Based"),  
 Mutual\_Information = c(  
 Mutual\_Information(heart\_failure, syn\_data\_1),  
 Mutual\_Information(heart\_failure, syn\_cart\_1),  
 Mutual\_Information(heart\_failure, syn\_data\_low\_fidelity\_synthpop),  
 Mutual\_Information(heart\_failure, syn\_data\_metadata)  
 )  
) |>  
 mutate(Mutual\_Information = round(Mutual\_Information \* 100, 1)) # convert to %  
  
# User-friendly table  
knitr::kable(  
 mi\_results,  
 caption = "Mutual Information Similarity (%): higher = synthetic preserves dependency structure more closely",  
 align = "lr"  
)

Mutual Information Similarity (%): higher = synthetic preserves dependency structure more closely

| Dataset | Mutual\_Information |
| --- | --- |
| Parametric MICE | 93.3 |
| CART Imputation | 94.3 |
| Synthpop (Low Fidelity) | 91.5 |
| Metadata-Based | 93.2 |

# Horizontal bar chart with custom colours  
ggplot(mi\_results, aes(y = Dataset, x = Mutual\_Information, fill = Dataset)) +  
 geom\_col(alpha = 0.9, show.legend = FALSE) +  
 geom\_text(aes(label = paste0(Mutual\_Information, "%")),   
 hjust = -0.1, size = 3.3) +  
 scale\_fill\_manual(values = dataset\_palette) +  
 scale\_x\_continuous(labels = scales::percent\_format(scale = 1), limits = c(0, 100)) +  
 labs(  
 title = "Mutual Information Similarity by Synthetic Method",  
 subtitle = "Based on pairwise MI matrices of discretized numeric variables (equal-frequency bins)",  
 x = "Similarity (%)", y = NULL  
 ) +  
 theme\_minimal() +  
 theme(axis.text.y = element\_text(size = 10))



### 6.4 Correlation Score

The Correlation Score measures how well the relationships (correlations) between variables are preserved in the synthetic dataset compared to the real.

Ideal Ranges for Correlation Score:

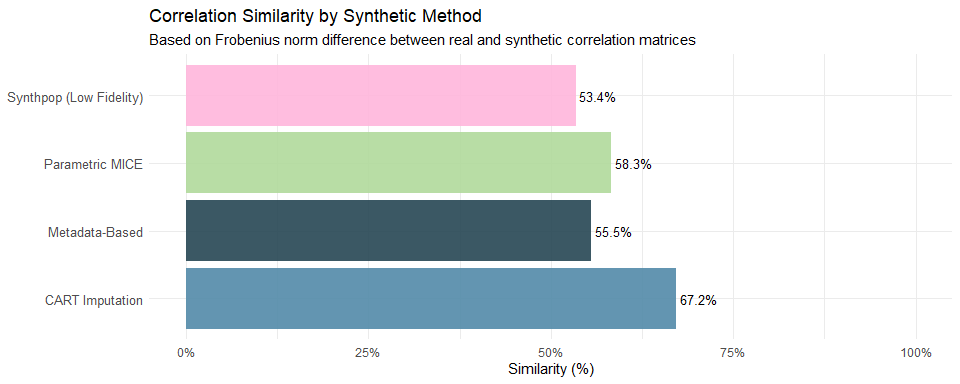
* **Perfect Alignment:** Correlations in the synthetic data should match the real closely, with deviations of 5-10% being acceptable.
* **Weaker Correlations:** If synthetic correlations are weaker, it may indicate loss of fidelity due to noise.
* **Stronger Correlations:** Stronger correlations suggest overfitting, which may increase privacy risks.

# Custom palette for datasets  
dataset\_palette <- c(  
 "Parametric MICE" = "#B0D99B",  
 "CART Imputation" = "#528AA8",  
 "Synthpop (Low Fidelity)" = "#FFB6DB",  
 "Metadata-Based" = "#264653"  
)  
  
# --- Helper: strip 'synth\_' so names align  
strip\_synth\_prefix <- function(df) {  
 names(df) <- sub("^synth\_", "", names(df))  
 df  
}  
  
# --- Function to calculate Correlation Similarity Score  
calculate\_correlation\_score <- function(real\_data, synthetic\_data) {  
 # Keep numeric columns only  
 real\_num <- real\_data[sapply(real\_data, is.numeric)]  
 synth\_num <- strip\_synth\_prefix(synthetic\_data)[sapply(strip\_synth\_prefix(synthetic\_data), is.numeric)]  
   
 # Use only common variables  
 common <- intersect(names(real\_num), names(synth\_num))  
 if (length(common) < 2) return(NA\_real\_)  
   
 real\_corr <- cor(real\_num[common], use = "complete.obs")  
 synth\_corr <- cor(synth\_num[common], use = "complete.obs")  
   
 # Frobenius norm difference → similarity score in [0,1]  
 corr\_diff <- norm(real\_corr - synth\_corr, type = "F")  
 1 / (1 + corr\_diff)  
}  
  
# --- Compute scores for all synthetic datasets  
corr\_scores <- tibble::tibble(  
 Dataset = c("Parametric MICE", "CART Imputation", "Synthpop (Low Fidelity)", "Metadata-Based"),  
 Correlation\_Score = c(  
 calculate\_correlation\_score(heart\_failure, syn\_data\_1),  
 calculate\_correlation\_score(heart\_failure, syn\_cart\_1),  
 calculate\_correlation\_score(heart\_failure, syn\_data\_low\_fidelity\_synthpop),  
 calculate\_correlation\_score(heart\_failure, syn\_data\_metadata)  
 )  
) %>%  
 mutate(Correlation\_Score = round(Correlation\_Score \* 100, 1)) # convert to %  
  
# --- User-friendly table  
kable(  
 corr\_scores,  
 caption = "Correlation Similarity Scores (%): higher = synthetic preserves correlation structure more closely",  
 align = "lr"  
)

Correlation Similarity Scores (%): higher = synthetic preserves correlation structure more closely

| Dataset | Correlation\_Score |
| --- | --- |
| Parametric MICE | 58.3 |
| CART Imputation | 67.2 |
| Synthpop (Low Fidelity) | 53.4 |
| Metadata-Based | 55.5 |

# --- Horizontal bar chart with custom colours  
ggplot(corr\_scores, aes(y = Dataset, x = Correlation\_Score, fill = Dataset)) +  
 geom\_col(alpha = 0.9, show.legend = FALSE) +  
 geom\_text(aes(label = paste0(Correlation\_Score, "%")),   
 hjust = -0.1, size = 3.3) +  
 scale\_fill\_manual(values = dataset\_palette) +  
 scale\_x\_continuous(labels = scales::percent\_format(scale = 1), limits = c(0, 100)) +  
 labs(  
 title = "Correlation Similarity by Synthetic Method",  
 subtitle = "Based on Frobenius norm difference between real and synthetic correlation matrices",  
 x = "Similarity (%)", y = NULL  
 ) +  
 theme\_minimal() +  
 theme(axis.text.y = element\_text(size = 10))



## 7 Utility Assessment

### 7.1 Feature Importance Consistency Assessment

The Feature Importance Consistency Assessment evaluates how well the synthetic datasets retain the predictive significance of features compared to the real dataset. This assessment is essential for ensuring that the synthetic data accurately reflects the underlying relationships that contribute to the model’s predictive performance.

Feature importance measures each variable’s contribution to model predictions. Consistency in feature importance rankings between the real and synthetic datasets serves as a crucial indicator of data quality. Significant discrepancies may indicate deficiencies in the synthetic data generation process, potentially affecting the performance of models trained on synthetic data.

Ideal Ranges for Feature Importance Consistency Assessment:

* **Perfect Match**: Feature importance scores in the synthetic dataset should closely align with those in the real dataset, particularly for highly predictive features.
* **Acceptable Deviation**: A deviation of 10-15% is generally acceptable for most features. For variables identified as highly influential, stricter thresholds may be needed to maintain predictive accuracy.

Types of Deviations

* **Underrepresentation**: If an important feature has lower importance in the synthetic dataset, it may suggest that critical predictive relationships are not well-preserved.
* **Overrepresentation**: If a feature’s importance is higher in the synthetic data, this could indicate noise or overfitting, potentially distorting the model’s perception of feature relationships.

The following visualisation techniques will be used to compare feature importance:

* **Bar Plots for Feature Importance Comparison**: Bar plots will compare feature importance scores between the real and synthetic datasets. This visual comparison can reveal significant deviations.
* **SHAP Summary Plots**: SHAP (SHapley Additive exPlanations) summary plots provide a comprehensive view of the distribution of SHAP values for each feature, allowing for a direct comparison between the real and synthetic datasets.

# Prepare the data for XGBoost  
set.seed(123) # For reproducibility  
train\_indices <- sample(seq\_len(nrow(heart\_failure)), size = 0.7 \* nrow(heart\_failure))  
train\_real <- heart\_failure[train\_indices, ]  
test\_real <- heart\_failure[-train\_indices, ]  
  
train\_real\_matrix <- model.matrix(deceased ~ age + sex + anaemia + creatinine\_phosphokinase + diabetes + ejection\_fraction + platelets + serum\_creatinine + serum\_sodium + smoking + hypertension + follow\_up - 1, data = train\_real)  
train\_real\_label <- train\_real[rownames(train\_real\_matrix), "deceased"]  
train\_real\_label <- as.numeric(train\_real\_label) - 1 # Convert factor to 0 and 1  
  
train\_synth <- syn\_data\_1[train\_indices, ]  
train\_synth\_matrix <- model.matrix(deceased ~ age + sex + anaemia + creatinine\_phosphokinase + diabetes + ejection\_fraction + platelets + serum\_creatinine + serum\_sodium + smoking + hypertension + follow\_up - 1, data = train\_synth)  
train\_synth\_label <- train\_synth[rownames(train\_synth\_matrix), "deceased"]  
train\_synth\_label <- as.numeric(train\_synth\_label) - 1 # Convert factor to 0 and 1  
  
# Convert to DMatrix format, which is required for XGBoost  
dtrain\_real <- xgb.DMatrix(data = train\_real\_matrix, label = train\_real\_label)  
dtrain\_synth <- xgb.DMatrix(data = train\_synth\_matrix, label = train\_synth\_label)  
  
# Set up parameters for XGBoost  
params <- list(  
 objective = "binary:logistic",  
 eval\_metric = "auc", # Use AUC as an evaluation metric  
 max\_depth = 3,  
 eta = 0.1  
)  
  
# Tune hyperparameters using cross-validation (with parallel processing)  
cv\_real <- xgb.cv(  
 params = params,  
 data = dtrain\_real,  
 nrounds = 100,  
 nfold = 5,  
 verbose = 0,  
 early\_stopping\_rounds = 10,  
 nthread = num\_cores # Number of threads for parallel processing  
)  
  
best\_nrounds\_real <- cv\_real$best\_iteration  
  
cv\_synth <- xgb.cv(  
 params = params,  
 data = dtrain\_synth,  
 nrounds = 100,  
 nfold = 5,  
 verbose = 0,  
 early\_stopping\_rounds = 10,  
 nthread = num\_cores # Number of threads for parallel processing  
)  
  
best\_nrounds\_synth <- cv\_synth$best\_iteration  
  
# Train XGBoost on real data (TRTR)  
xgb\_model\_trtr <- xgboost(params = params, data = dtrain\_real, nrounds = best\_nrounds\_real, verbose = 0, nthread = num\_cores)  
  
# Train XGBoost on synthetic data (TSTR)  
xgb\_model\_tstr <- xgboost(params = params, data = dtrain\_synth, nrounds = best\_nrounds\_synth, verbose = 0, nthread = num\_cores)  
  
# Get feature importance for TRTR  
importance\_trtr <- xgb.importance(feature\_names = colnames(train\_real\_matrix), model = xgb\_model\_trtr)  
cat("Feature Importance for TRTR:\n")

Feature Importance for TRTR:

print(importance\_trtr)

Feature Gain Cover Frequency  
 <char> <num> <num> <num>  
1: follow\_up 0.550035467 0.35582680 0.26470588  
2: serum\_creatinine 0.179450673 0.25780255 0.22794118  
3: ejection\_fraction 0.107840968 0.17061583 0.12500000  
4: creatinine\_phosphokinase 0.100830923 0.12847956 0.20588235  
5: serum\_sodium 0.032740748 0.04233723 0.08823529  
6: age 0.022510598 0.03446459 0.04411765  
7: platelets 0.006590623 0.01047345 0.04411765

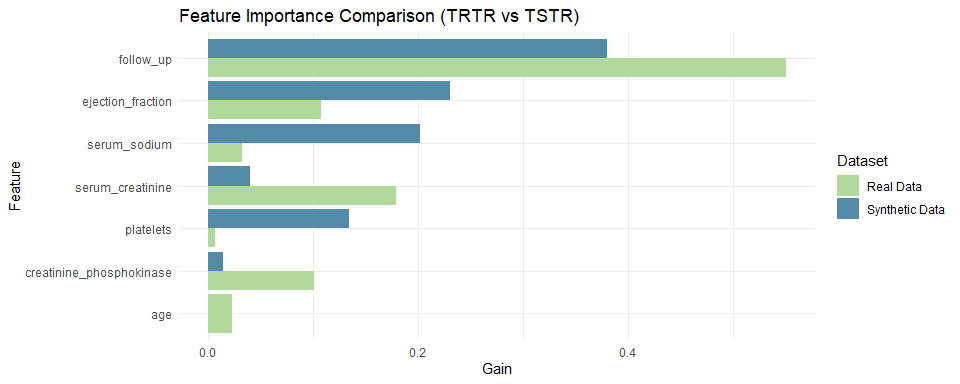
# Get feature importance for TSTR  
importance\_tstr <- xgb.importance(feature\_names = colnames(train\_synth\_matrix), model = xgb\_model\_tstr)  
cat("\nFeature Importance for TSTR:\n")

Feature Importance for TSTR:

print(importance\_tstr)

Feature Gain Cover Frequency  
 <char> <num> <num> <num>  
1: follow\_up 0.37991328 0.37325270 0.20  
2: ejection\_fraction 0.22999025 0.21347069 0.32  
3: serum\_sodium 0.20146359 0.17350782 0.16  
4: platelets 0.13419963 0.10862788 0.12  
5: serum\_creatinine 0.04007483 0.12057396 0.16  
6: creatinine\_phosphokinase 0.01435841 0.01056696 0.04

# Visualise feature importance for TRTR and TSTR  
importance\_trtr$Dataset <- "Real Data"  
importance\_tstr$Dataset <- "Synthetic Data"  
  
importance\_combined <- rbind(importance\_trtr, importance\_tstr)  
  
# Plot feature importance comparison  
ggplot(importance\_combined, aes(x = reorder(Feature, Gain), y = Gain, fill = Dataset)) +  
 geom\_bar(stat = "identity", position = "dodge") +  
 coord\_flip() +  
 labs(title = "Feature Importance Comparison (TRTR vs TSTR)",  
 x = "Feature",  
 y = "Gain") +  
 scale\_fill\_manual(values = c("Real Data" = "#B0D99B", "Synthetic Data" = "#528AA8")) +  
 theme\_minimal()



# Correlation analysis between feature importance rankings  
  
importance\_trtr <- importance\_trtr %>% mutate(rank = rank(-Gain))  
importance\_tstr <- importance\_tstr %>% mutate(rank = rank(-Gain))  
  
importance\_ranking <- inner\_join(importance\_trtr, importance\_tstr, by = "Feature", suffix = c("\_trtr", "\_tstr"))  
  
correlation <- cor(importance\_ranking$rank\_trtr, importance\_ranking$rank\_tstr, method = "spearman")  
cat("\nSpearman correlation Between TRTR and TSTR Feature Importance Rankings:\n")

Spearman correlation Between TRTR and TSTR Feature Importance Rankings:

print(correlation)

[1] 0.3714286

# Calculate SHAP values for better interpretability  
  
# SHAP values for TRTR  
shap\_values\_trtr <- shap.values(xgb\_model\_trtr, dtrain\_real)  
shap\_long\_trtr <- shap.prep(shap\_contrib = shap\_values\_trtr$shap\_score, X\_train = train\_real\_matrix)  
  
cat("\nSHAP Summary for TRTR:\n")

SHAP Summary for TRTR:

shap\_values\_trtr$mean\_shap\_score %>% print()

follow\_up serum\_creatinine ejection\_fraction   
 0.92293409 0.52780225 0.39932806   
creatinine\_phosphokinase serum\_sodium age   
 0.18850937 0.10067811 0.06711965   
 platelets sexFemale sexMale   
 0.02991459 0.00000000 0.00000000   
 anaemiaYes diabetesYes smokingYes   
 0.00000000 0.00000000 0.00000000   
 hypertensionYes   
 0.00000000

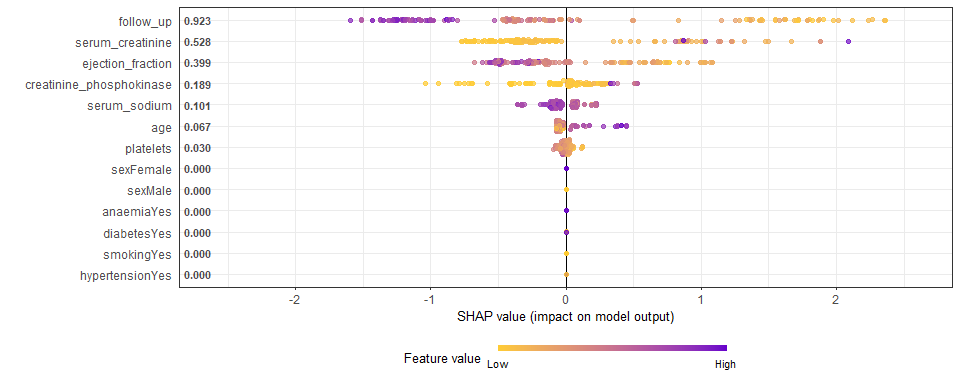
# SHAP values for TSTR  
shap\_values\_tstr <- shap.values(xgb\_model\_tstr, dtrain\_synth)  
shap\_long\_tstr <- shap.prep(shap\_contrib = shap\_values\_tstr$shap\_score, X\_train = train\_synth\_matrix)  
  
cat("\nSHAP Summary for TSTR:\n")

SHAP Summary for TSTR:

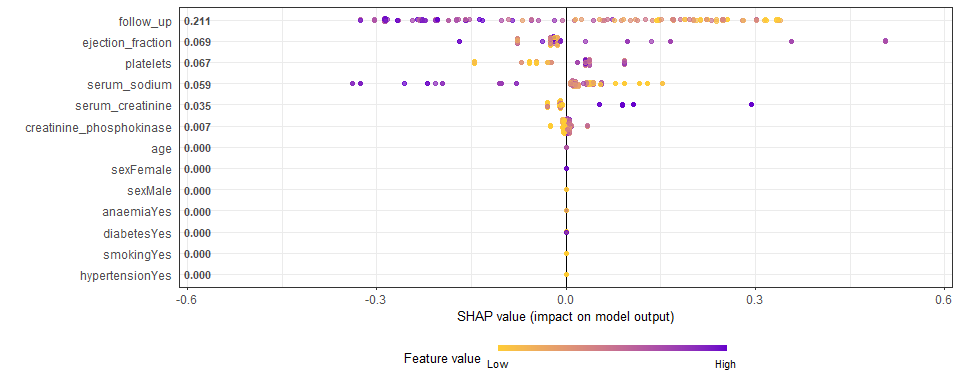
shap\_values\_tstr$mean\_shap\_score %>% print()

follow\_up ejection\_fraction platelets   
 0.210816300 0.069274587 0.067060756   
 serum\_sodium serum\_creatinine creatinine\_phosphokinase   
 0.058853411 0.035087248 0.006947092   
 age sexFemale sexMale   
 0.000000000 0.000000000 0.000000000   
 anaemiaYes diabetesYes smokingYes   
 0.000000000 0.000000000 0.000000000   
 hypertensionYes   
 0.000000000

# SHAP visualisation  
shap.plot.summary(shap\_long\_trtr)



shap.plot.summary(shap\_long\_tstr)



### 7.2 Prediction Score

The Prediction Score assesses synthetic data utility by comparing models trained on real data (Train Real Test Real (TRTR)) versus synthetic data (Train Synthetic Test Real (TSTR)) using two metrics:

* **Accuracy**: The proportion of correct predictions made by the model.
* **pMSE (Prediction Mean Squared Error)**: Measures the average of the squared differences between predicted and actual values.

Ideal Ranges for Prediction Score:

* **Accuracy**: TSTR accuracy should be within 5-10% of TRTR accuracy. Lower accuracy signals that synthetic data isn’t fully representative of the real dataset.
* **pMSE**: TSTR pMSE should be close to TRTR pMSE. A 5-10% deviation is acceptable; a higher pMSE suggests synthetic data may not be accurately reflecting the true relationships in the real data.

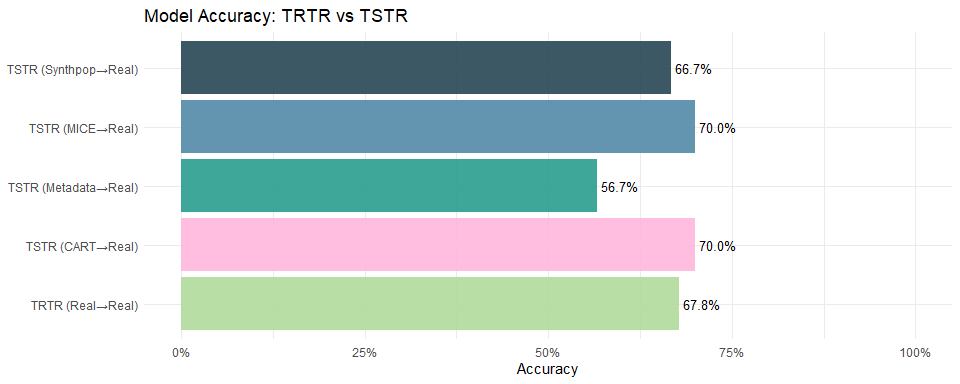
This section assesses the utility of synthetic data generated through various methods, using XGBoost to calculate both Accuracy and pMSE for TRTR and TSTR.

set.seed(123)  
  
# ---- helpers ----  
strip\_synth\_prefix <- function(df) { names(df) <- sub("^synth\_", "", names(df)); df }  
  
align\_target\_levels <- function(df, levels\_ref = NULL, positive = NULL) {  
 if (!is.factor(df$deceased)) {  
 if (is.numeric(df$deceased)) df$deceased <- factor(as.character(df$deceased)) else df$deceased <- factor(df$deceased)  
 }  
 if (!is.null(levels\_ref)) df$deceased <- factor(df$deceased, levels = levels\_ref)  
 lev <- levels(df$deceased)  
 if (length(lev) < 2) stop("Target 'deceased' must have two levels.")  
 pos <- if (!is.null(positive) && positive %in% lev) positive else if ("1" %in% lev) "1" else lev[2]  
 df$deceased <- stats::relevel(df$deceased, ref = pos)  
 list(data = df, levels = levels(df$deceased), positive = pos)  
}  
  
evaluate\_model <- function(train\_data, test\_data, label, tune\_grid, train\_control, seed = 123) {  
 test\_aln <- align\_target\_levels(test\_data)  
 train\_aln <- align\_target\_levels(train\_data, levels\_ref = test\_aln$levels, positive = test\_aln$positive)  
 train\_data <- train\_aln$data  
 test\_data <- test\_aln$data  
  
 set.seed(seed)  
 model <- caret::train(  
 deceased ~ follow\_up + serum\_creatinine + ejection\_fraction +  
 creatinine\_phosphokinase + serum\_sodium + age + platelets,  
 data = train\_data,  
 method = "xgbTree",  
 trControl = train\_control,  
 tuneGrid = tune\_grid,  
 na.action = na.pass  
 )  
  
 pred <- predict(model, test\_data)  
 acc <- mean(pred == test\_data$deceased, na.rm = TRUE)  
 yhat <- as.numeric(pred)  
 y <- as.numeric(test\_data$deceased)  
 pMSE <- mean((yhat - y)^2, na.rm = TRUE)  
  
 tibble::tibble(Dataset = label, Accuracy = acc, pMSE = pMSE)  
}  
  
# ---- CV setup ----  
train\_control <- caret::trainControl(method = "cv", number = 5)  
tune\_grid <- expand.grid(  
 nrounds = 50,  
 max\_depth = 3,  
 eta = 0.1,  
 gamma = 0,  
 colsample\_bytree = 0.8,  
 min\_child\_weight = 1,  
 subsample = 0.7  
)  
  
# ---- prepare synthetic splits ----  
syn\_mice <- strip\_synth\_prefix(syn\_data\_1)  
syn\_cart <- strip\_synth\_prefix(syn\_cart\_1)  
syn\_synthpop <- strip\_synth\_prefix(syn\_data\_low\_fidelity\_synthpop)  
syn\_metadata <- strip\_synth\_prefix(syn\_data\_metadata)  
  
train\_synth\_mice <- syn\_mice[train\_indices, , drop = FALSE]  
train\_synth\_cart <- syn\_cart[train\_indices, , drop = FALSE]  
train\_synth\_synthpop <- syn\_synthpop[train\_indices, , drop = FALSE]  
train\_synth\_metadata <- syn\_metadata[train\_indices, , drop = FALSE]  
  
# ---- results table ----  
results <- dplyr::bind\_rows(  
 evaluate\_model(train\_real, test\_real, "TRTR (Real→Real)", tune\_grid, train\_control),  
 evaluate\_model(train\_synth\_mice, test\_real, "TSTR (MICE→Real)", tune\_grid, train\_control),  
 evaluate\_model(train\_synth\_cart, test\_real, "TSTR (CART→Real)", tune\_grid, train\_control),  
 evaluate\_model(train\_synth\_synthpop, test\_real, "TSTR (Synthpop→Real)", tune\_grid, train\_control),  
 evaluate\_model(train\_synth\_metadata, test\_real, "TSTR (Metadata→Real)", tune\_grid, train\_control)  
) %>%  
 mutate(  
 Accuracy = scales::percent(Accuracy, accuracy = 0.1),  
 pMSE = round(pMSE, 3)  
 )  
  
knitr::kable(  
 results,  
 caption = "Prediction Score Summary (XGBoost, 5-fold CV): TRTR vs TSTR",  
 align = "lrr"  
)

Prediction Score Summary (XGBoost, 5-fold CV): TRTR vs TSTR

| Dataset | Accuracy | pMSE |
| --- | --- | --- |
| TRTR (Real→Real) | 67.8% | 0.322 |
| TSTR (MICE→Real) | 70.0% | 0.300 |
| TSTR (CART→Real) | 70.0% | 0.300 |
| TSTR (Synthpop→Real) | 66.7% | 0.333 |
| TSTR (Metadata→Real) | 56.7% | 0.433 |

# ---- horizontal Accuracy bar chart ----  
palette5 <- c("#B0D99B", "#528AA8", "#FFB6DB", "#264653", "#2A9D8F")  
  
plot\_df <- results %>%  
 mutate(Accuracy\_num = readr::parse\_number(Accuracy) / 100)  
  
ggplot(plot\_df, aes(y = Dataset, x = Accuracy\_num, fill = Dataset)) +  
 geom\_col(alpha = 0.9, show.legend = FALSE) +  
 geom\_text(aes(label = Accuracy), hjust = -0.1, size = 3.5) +  
 scale\_fill\_manual(values = setNames(palette5, unique(plot\_df$Dataset))) +  
 scale\_x\_continuous(labels = scales::percent\_format(accuracy = 1), limits = c(0, 1)) +  
 labs(title = "Model Accuracy: TRTR vs TSTR", x = "Accuracy", y = NULL) +  
 theme\_minimal()



### 7.3 Quality Score (QScore)

The QScore provides a comprehensive evaluation of synthetic data by aggregating multiple metrics, such as distribution similarity, feature importance, and correlation preservation, into a single score. This offers a clear, overall assessment of how well the synthetic data replicates the real dataset.

Ideal Ranges for QScore:

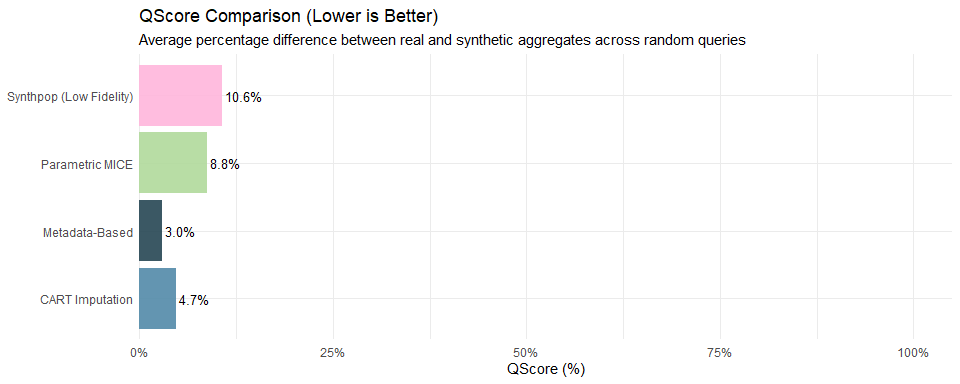
* **Excellent Quality (0.8 - 1.0)**: Indicates the synthetic data closely mirrors the real and can be confidently used for most analyses.
* **Good Quality (0.6 - 0.8)**: Reflects good alignment, with some minor deviations. Suitable for most use cases, but some caution is advised.
* **Moderate Quality (0.4 - 0.6)**: The synthetic data may be useful for exploratory analysis but shows significant deviations in key metrics.
* **Poor Quality (< 0.4)**: The synthetic data poorly aligns with the real and is likely unsuitable for most analytic or modeling tasks.

# ---- QScore helper ----  
calculate\_qscore <- function(real\_data, synthetic\_data, num\_queries = 10) {  
 numeric\_real\_vars <- names(real\_data)[sapply(real\_data, is.numeric)]  
 numeric\_synth\_vars <- names(synthetic\_data)[sapply(synthetic\_data, is.numeric)]  
 common\_vars <- intersect(numeric\_real\_vars, numeric\_synth\_vars)  
 if (length(common\_vars) == 0) return(NA\_real\_)  
  
 qscores <- replicate(num\_queries, {  
 var\_name <- sample(common\_vars, 1)  
 fun\_name <- sample(c("mean", "sum"), 1)  
 fun <- match.fun(fun\_name)  
  
 real\_val <- fun(real\_data[[var\_name]], na.rm = TRUE)  
 synth\_val <- fun(synthetic\_data[[var\_name]], na.rm = TRUE)  
  
 if (!is.na(real\_val) && real\_val != 0 && !is.na(synth\_val)) {  
 abs(real\_val - synth\_val) / abs(real\_val) # proportion difference (0–1)  
 } else {  
 0  
 }  
 })  
  
 mean(qscores, na.rm = TRUE)  
}  
  
# ---- Compute and present ----  
qscore\_tbl <- tibble::tibble(  
 Dataset = c("Parametric MICE", "CART Imputation",  
 "Synthpop (Low Fidelity)", "Metadata-Based"),  
 QScore = c(  
 calculate\_qscore(heart\_failure, syn\_data\_1),  
 calculate\_qscore(heart\_failure, syn\_cart\_1),  
 calculate\_qscore(heart\_failure, syn\_data\_low\_fidelity\_synthpop),  
 calculate\_qscore(heart\_failure, syn\_data\_metadata)  
 )  
) %>%  
 dplyr::mutate(  
 QScore\_num = QScore, # keep numeric 0–1  
 QScore\_pct = scales::percent(QScore\_num, accuracy = 0.1) # pretty %  
 )  
  
# --- User-friendly table (as percentages) ---  
knitr::kable(  
 qscore\_tbl %>% dplyr::select(Dataset, `QScore (%)` = QScore\_pct),  
 caption = "Quality Score (QScore) — average % difference in aggregates (lower is better)",  
 align = "lr"  
)

Quality Score (QScore) — average % difference in aggregates (lower is better)

| Dataset | QScore (%) |
| --- | --- |
| Parametric MICE | 8.8% |
| CART Imputation | 4.7% |
| Synthpop (Low Fidelity) | 10.6% |
| Metadata-Based | 3.0% |

# --- Horizontal bar chart (percent axis) ---  
pal <- c("#B0D99B", "#528AA8", "#FFB6DB", "#264653", "#2A9D8F")  
  
ggplot(qscore\_tbl, aes(y = Dataset, x = QScore\_num, fill = Dataset)) +  
 geom\_col(alpha = 0.9, show.legend = FALSE) +  
 geom\_text(aes(label = QScore\_pct), hjust = -0.1, size = 3.5) +  
 scale\_fill\_manual(values = setNames(pal[seq\_len(nrow(qscore\_tbl))], qscore\_tbl$Dataset)) +  
 scale\_x\_continuous(labels = scales::percent\_format(accuracy = 1), limits = c(0, 1),  
 expand = expansion(mult = c(0, 0.05))) +  
 labs(  
 title = "QScore Comparison (Lower is Better)",  
 subtitle = "Average percentage difference between real and synthetic aggregates across random queries",  
 x = "QScore (%)", y = NULL  
 ) +  
 theme\_minimal()



## 8 Privacy / Disclosure Risk

### 8.1 Check for Replication of Real Value Combinations

This check identifies if any records in the synthetic data are exact duplicates of those in the real dataset. High replication increases privacy risks, so minimal or no duplication is ideal for preserving privacy.

# ---- Helper: replicated rows count ----  
check\_replicated\_rows <- function(real\_data, synthetic\_data) {  
 combined <- rbind(real\_data, synthetic\_data)  
 sum(duplicated(combined))  
}  
  
# ---- Build results table ----  
replication\_tbl <- tibble::tibble(  
 Dataset = c("Parametric MICE", "Non-Parametric CART",   
 "Synthpop (Low Fidelity)", "Metadata-Based"),  
 Replicated\_Rows = c(  
 check\_replicated\_rows(heart\_failure, syn\_data\_1),  
 check\_replicated\_rows(heart\_failure, syn\_cart\_1),  
 check\_replicated\_rows(heart\_failure, syn\_data\_low\_fidelity\_synthpop),  
 check\_replicated\_rows(heart\_failure, syn\_data\_metadata)  
 )  
) %>%  
 mutate(  
 Total\_Rows = nrow(heart\_failure),  
 Replication\_Percent = round((Replicated\_Rows / Total\_Rows) \* 100, 2),  
 Status = ifelse(Replicated\_Rows > 0, "⚠️ Replication Found", "✅ No Replication")  
 )  
  
# ---- User-friendly table ----  
knitr::kable(  
 replication\_tbl,  
 caption = "Replication Check: Exact Matching Rows Between Real and Synthetic Datasets",  
 align = "lrrrl"  
)

Replication Check: Exact Matching Rows Between Real and Synthetic Datasets

| Dataset | Replicated\_Rows | Total\_Rows | Replication\_Percent | Status |
| --- | --- | --- | --- | --- |
| Parametric MICE | 0 | 299 | 0 | ✅ No Replication |
| Non-Parametric CART | 0 | 299 | 0 | ✅ No Replication |
| Synthpop (Low Fidelity) | 0 | 299 | 0 | ✅ No Replication |
| Metadata-Based | 0 | 299 | 0 | ✅ No Replication |

# ---- Optional: bar chart if any replication > 0 ----  
if (any(replication\_tbl$Replicated\_Rows > 0)) {  
 pal <- c("#B0D99B", "#528AA8", "#FFB6DB", "#264653")  
   
 ggplot(replication\_tbl, aes(y = Dataset, x = Replicated\_Rows, fill = Dataset)) +  
 geom\_col(alpha = 0.9, show.legend = FALSE) +  
 geom\_text(aes(label = Replicated\_Rows), hjust = -0.1, size = 3.5) +  
 scale\_fill\_manual(values = pal[seq\_len(nrow(replication\_tbl))]) +  
 labs(  
 title = "Replication of Real Rows in Synthetic Data",  
 subtitle = "Counts of exact duplicated rows across real and synthetic datasets",  
 x = "Number of Replicated Rows", y = NULL  
 ) +  
 theme\_minimal()  
} else {  
 cat("✅ No replicated rows were found across any synthetic dataset. No plot is generated.\n")  
}

✅ No replicated rows were found across any synthetic dataset. No plot is generated.

### 8.2 Check for Unique Value Combinations

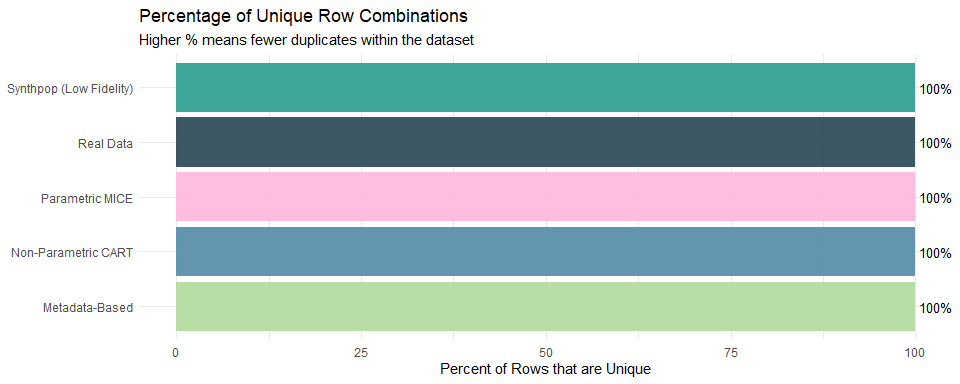
This analysis compares the number of unique row combinations between the real and synthetic datasets. A high number of unique combinations in the synthetic data suggests that it captures diverse patterns while minimising the replication of sensitive information, enhancing privacy protection.

# ---- Helper: count unique rows ----  
unique\_combinations <- function(data) {  
 nrow(unique(data))  
}  
  
# ---- Build results table ----  
unique\_tbl <- tibble::tibble(  
 Dataset = c("Real Data", "Parametric MICE",   
 "Non-Parametric CART", "Synthpop (Low Fidelity)",   
 "Metadata-Based"),  
 Unique\_Combinations = c(  
 unique\_combinations(heart\_failure),  
 unique\_combinations(syn\_data\_1),  
 unique\_combinations(syn\_cart\_1),  
 unique\_combinations(syn\_data\_low\_fidelity\_synthpop),  
 unique\_combinations(syn\_data\_metadata)  
 )  
) %>%  
 mutate(  
 Total\_Rows = c(nrow(heart\_failure), rep(nrow(heart\_failure), 4)),  
 Percent\_Unique = round((Unique\_Combinations / Total\_Rows) \* 100, 2)  
 )  
  
# ---- User-friendly table ----  
knitr::kable(  
 unique\_tbl,  
 caption = "Unique Value Combinations in Real vs. Synthetic Datasets",  
 align = "lrr"  
)

Unique Value Combinations in Real vs. Synthetic Datasets

| Dataset | Unique\_Combinations | Total\_Rows | Percent\_Unique |
| --- | --- | --- | --- |
| Real Data | 299 | 299 | 100 |
| Parametric MICE | 299 | 299 | 100 |
| Non-Parametric CART | 299 | 299 | 100 |
| Synthpop (Low Fidelity) | 299 | 299 | 100 |
| Metadata-Based | 299 | 299 | 100 |

# ---- Plot only if any dataset has > 0 unique rows ----  
if (any(unique\_tbl$Unique\_Combinations > 0, na.rm = TRUE)) {  
 pal <- c("#B0D99B", "#528AA8", "#FFB6DB", "#264653", "#2A9D8F")  
  
 ggplot(unique\_tbl, aes(y = Dataset, x = Percent\_Unique, fill = Dataset)) +  
 geom\_col(alpha = 0.9, show.legend = FALSE) +  
 geom\_text(aes(label = paste0(Percent\_Unique, "%")), hjust = -0.1, size = 3.5) +  
 scale\_fill\_manual(values = pal[seq\_len(nrow(unique\_tbl))]) +  
 scale\_x\_continuous(limits = c(0, 100)) +  
 labs(  
 title = "Percentage of Unique Row Combinations",  
 subtitle = "Higher % means fewer duplicates within the dataset",  
 x = "Percent of Rows that are Unique", y = NULL  
 ) +  
 theme\_minimal()  
} else {  
 cat("✅ No unique combinations detected across datasets. No plot is generated. \n")  
}



### 8.3 Exact Match Score

The Exact Match Score measures how many individual records in the synthetic data are identical to records in the real dataset. Lower scores are ideal for privacy preservation, as high exact matches may pose privacy risks.

Ideal Ranges for Exact Match Score:

* **0 - 0.1**: Strong privacy protection with minimal or no exact matches, ideal for most privacy-preserving synthetic data.
* **0.1 - 0.3**: Moderate exact matches, indicating some privacy risk but generally acceptable for many use cases.
* **0.3 - 0.5**: Higher exact matches, raising privacy concerns. Review is needed to ensure proper data generation.
* **Above 0.5**: Significant privacy risk, as the synthetic data too closely replicates the real.

# ---- Helper: Membership Inference Score ----  
calculate\_membership\_inference\_score <- function(real\_data, synthetic\_data, k = 5, threshold = 0.01) {  
 real\_num <- real\_data[sapply(real\_data, is.numeric)]  
 synth\_num <- synthetic\_data[sapply(synthetic\_data, is.numeric)]  
 if (ncol(real\_num) == 0 || ncol(synth\_num) == 0) return(NA\_real\_)  
 common <- intersect(names(real\_num), names(synth\_num))  
 if (length(common) == 0) return(NA\_real\_)  
 real\_num <- real\_num[common]  
 synth\_num <- synth\_num[common]  
 real\_cc <- real\_num[complete.cases(real\_num), , drop = FALSE]  
 synth\_cc <- synth\_num[complete.cases(synth\_num), , drop = FALSE]  
 if (nrow(real\_cc) < k || nrow(synth\_cc) == 0) return(NA\_real\_)  
 real\_scaled <- scale(real\_cc)  
 synth\_scaled <- scale(synth\_cc)  
 nn <- FNN::get.knnx(real\_scaled, synth\_scaled, k)$nn.dist  
 risky <- sum(rowMeans(nn) < threshold)  
 risky / nrow(synth\_scaled)  
}  
  
# ---- Results table ----  
mis\_tbl <- tibble::tibble(  
 Dataset = c("Parametric MICE", "Non-Parametric CART",  
 "Synthpop (Low Fidelity)", "Metadata-Based"),  
 Membership\_Inference\_Score = c(  
 calculate\_membership\_inference\_score(heart\_failure, syn\_data\_1),  
 calculate\_membership\_inference\_score(heart\_failure, syn\_cart\_1),  
 calculate\_membership\_inference\_score(heart\_failure, syn\_data\_low\_fidelity\_synthpop),  
 calculate\_membership\_inference\_score(heart\_failure, syn\_data\_metadata)  
 )  
) %>%  
 mutate(  
 Membership\_Inference\_Score = round(Membership\_Inference\_Score, 4),  
 Status = dplyr::case\_when(  
 is.na(Membership\_Inference\_Score) ~ "— Insufficient data",  
 Membership\_Inference\_Score == 0 ~ "✅ No risky memberships",  
 Membership\_Inference\_Score < 0.01 ~ "⚠️ Low risk",  
 TRUE ~ "❌ High risk"  
 )  
 )  
  
# ---- User-friendly table ----  
knitr::kable(  
 mis\_tbl,  
 caption = "Membership Inference Score: Proportion of synthetic records that are very close to real records (lower is better for privacy)",  
 align = "lrr"  
)

Membership Inference Score: Proportion of synthetic records that are very close to real records (lower is better for privacy)

| Dataset | Membership\_Inference\_Score | Status |
| --- | --- | --- |
| Parametric MICE | 0 | ✅ No risky memberships |
| Non-Parametric CART | 0 | ✅ No risky memberships |
| Synthpop (Low Fidelity) | 0 | ✅ No risky memberships |
| Metadata-Based | 0 | ✅ No risky memberships |

# ---- Conditional plot ----  
if (any(mis\_tbl$Membership\_Inference\_Score > 0, na.rm = TRUE)) {  
 pal <- c("#B0D99B", "#528AA8", "#FFB6DB", "#264653")  
 ggplot(mis\_tbl, aes(y = Dataset, x = Membership\_Inference\_Score, fill = Dataset)) +  
 geom\_col(alpha = 0.9, show.legend = FALSE, na.rm = TRUE) +  
 geom\_text(  
 aes(label = ifelse(is.na(Membership\_Inference\_Score), "NA",  
 scales::percent(Membership\_Inference\_Score, accuracy = 0.01))),  
 hjust = -0.1, size = 3.5  
 ) +  
 scale\_fill\_manual(values = pal[seq\_len(nrow(mis\_tbl))]) +  
 scale\_x\_continuous(labels = scales::percent\_format(accuracy = 1)) +  
 labs(  
 title = "Membership Inference Score by Synthetic Method",  
 subtitle = "Lower values indicate better privacy protection",  
 x = "Score (Proportion of risky memberships)",  
 y = NULL  
 ) +  
 theme\_minimal()  
} else {  
 cat("✅ All Membership Inference Scores are 0. No privacy risks detected, so no plot is generated\n")  
}

✅ All Membership Inference Scores are 0. No privacy risks detected, so no plot is generated

### 8.4 Neighbors’ Privacy Score

The Neighbors’ Privacy Score identifies real records that are “too similar” to synthetic records by using a nearest-neighbors search. This metric helps evaluate privacy risk by detecting synthetic records that closely resemble real data, which could compromise privacy.

Ideal Ranges for Neighbors’ Privacy Score:

* **0 - 0.1**: Indicates strong privacy protection, with very few synthetic records too similar to real ones.
* **0.1 - 0.3**: Moderate similarity, where privacy risk exists but remains within acceptable limits for many cases.
* **Above 0.3**: Higher similarity, raising privacy concerns and requiring review to ensure the synthetic data is not too closely mimicking real data.

# ---- Helper: Neighbors' Privacy Score ----  
calculate\_neighbors\_privacy\_score <- function(real\_data, synthetic\_data, k = 5, threshold = 0.01) {  
 # Numeric-only columns  
 real\_num <- real\_data[sapply(real\_data, is.numeric)]  
 synth\_num <- synthetic\_data[sapply(synthetic\_data, is.numeric)]  
   
 if (ncol(real\_num) == 0 || ncol(synth\_num) == 0) return(NA\_real\_)  
   
 # Keep only common columns  
 common <- intersect(names(real\_num), names(synth\_num))  
 if (length(common) == 0) return(NA\_real\_)  
 real\_num <- real\_num[common]  
 synth\_num <- synth\_num[common]  
   
 # Complete cases  
 real\_cc <- real\_num[complete.cases(real\_num), , drop = FALSE]  
 synth\_cc <- synth\_num[complete.cases(synth\_num), , drop = FALSE]  
   
 if (nrow(real\_cc) < k || nrow(synth\_cc) == 0) return(NA\_real\_)  
   
 # Standardise  
 real\_scaled <- scale(real\_cc)  
 synth\_scaled <- scale(synth\_cc)  
   
 # k-NN distances from synthetic → real  
 neighbors <- FNN::get.knnx(real\_scaled, synth\_scaled, k)$nn.dist  
   
 # Privacy risk: synthetic records "too close" to real  
 close\_matches <- sum(rowMeans(neighbors) < threshold)  
 score <- close\_matches / nrow(synth\_scaled)  
   
 return(score)  
}  
  
# ---- Build results table ----  
neighbors\_tbl <- tibble::tibble(  
 Dataset = c("Parametric MICE", "Non-Parametric CART",  
 "Synthpop (Low Fidelity)", "Metadata-Based"),  
 Neighbors\_Privacy\_Score = c(  
 calculate\_neighbors\_privacy\_score(heart\_failure, syn\_data\_1),  
 calculate\_neighbors\_privacy\_score(heart\_failure, syn\_cart\_1),  
 calculate\_neighbors\_privacy\_score(heart\_failure, syn\_data\_low\_fidelity\_synthpop),  
 calculate\_neighbors\_privacy\_score(heart\_failure, syn\_data\_metadata)  
 )  
) %>%  
 mutate(  
 Neighbors\_Privacy\_Score = round(Neighbors\_Privacy\_Score, 4),  
 Status = case\_when(  
 Neighbors\_Privacy\_Score == 0 ~ "✅ No risky neighbors",  
 Neighbors\_Privacy\_Score < 0.01 ~ "⚠️ Low risk",  
 TRUE ~ "❌ High risk"  
 )  
 )  
  
# ---- User-friendly table ----  
knitr::kable(  
 neighbors\_tbl,  
 caption = "Neighbors' Privacy Score: Proportion of synthetic records too close to real records (lower is better for privacy)",  
 align = "lrr"  
)

Neighbors’ Privacy Score: Proportion of synthetic records too close to real records (lower is better for privacy)

| Dataset | Neighbors\_Privacy\_Score | Status |
| --- | --- | --- |
| Parametric MICE | 0 | ✅ No risky neighbors |
| Non-Parametric CART | 0 | ✅ No risky neighbors |
| Synthpop (Low Fidelity) | 0 | ✅ No risky neighbors |
| Metadata-Based | 0 | ✅ No risky neighbors |

# ---- Plot only if any score > 0 ----  
if (any(neighbors\_tbl$Neighbors\_Privacy\_Score > 0)) {  
 pal <- c("#B0D99B", "#528AA8", "#FFB6DB", "#264653")  
   
 ggplot(neighbors\_tbl, aes(y = Dataset, x = Neighbors\_Privacy\_Score, fill = Dataset)) +  
 geom\_col(alpha = 0.9, show.legend = FALSE) +  
 geom\_text(aes(label = Neighbors\_Privacy\_Score), hjust = -0.1, size = 3.5) +  
 scale\_fill\_manual(values = pal[seq\_len(nrow(neighbors\_tbl))]) +  
 scale\_x\_continuous(labels = scales::percent\_format(accuracy = 0.1)) +  
 labs(  
 title = "Neighbors' Privacy Score by Synthetic Method",  
 subtitle = "Lower values mean synthetic records are less likely to be close replicas of real individuals",  
 x = "Privacy Score (Proportion of risky neighbors)", y = NULL  
 ) +  
 theme\_minimal()  
} else {  
 cat("✅ All Neighbors' Privacy Scores are 0. No risky neighbors detected. No plot is generated.\n")  
}

✅ All Neighbors' Privacy Scores are 0. No risky neighbors detected. No plot is generated.

### 8.5 Membership Inference Score

The Membership Inference Score evaluates the risk of membership inference attacks, which attempt to determine whether a specific record belongs to the real dataset. This metric helps assess the vulnerability of synthetic data to privacy breaches.

Ideal Ranges for Membership Inference Score:

* **0 - 0.1**: Indicates low risk, where it is unlikely that membership inference attacks can accurately identify real records.
* **0.1 - 0.3**: Moderate risk, where some vulnerability to membership inference exists but may still be acceptable for certain use cases.
* **Above 0.3**: High risk, suggesting significant privacy concerns as the synthetic data could reveal membership information about the real records.

# ---- Helper: Membership Inference Score ----  
# Interpreted as the proportion of synthetic records that are "too close" to the real data.  
# Lower is better (safer).  
calculate\_membership\_inference\_score <- function(real\_data, synthetic\_data, k = 5, threshold = 0.01) {  
 # Numeric-only columns  
 real\_num <- real\_data[sapply(real\_data, is.numeric)]  
 synth\_num <- synthetic\_data[sapply(synthetic\_data, is.numeric)]  
  
 # Guardrails  
 if (ncol(real\_num) == 0 || ncol(synth\_num) == 0) return(NA\_real\_)  
  
 # Keep only common numeric variables  
 common <- intersect(names(real\_num), names(synth\_num))  
 if (length(common) == 0) return(NA\_real\_)  
 real\_num <- real\_num[common]  
 synth\_num <- synth\_num[common]  
  
 # Use complete cases  
 real\_cc <- real\_num[complete.cases(real\_num), , drop = FALSE]  
 synth\_cc <- synth\_num[complete.cases(synth\_num), , drop = FALSE]  
 if (nrow(real\_cc) < k || nrow(synth\_cc) == 0) return(NA\_real\_)  
  
 # Standardize  
 real\_scaled <- scale(real\_cc)  
 synth\_scaled <- scale(synth\_cc)  
  
 # k-NN distances from synthetic → real  
 nn <- FNN::get.knnx(real\_scaled, synth\_scaled, k)$nn.dist  
  
 # Score: proportion of synthetic records whose average distance to k nearest reals is below threshold  
 risky <- sum(rowMeans(nn) < threshold)  
 risky / nrow(synth\_scaled)  
}  
  
# ---- Results table ----  
mis\_tbl <- tibble::tibble(  
 Dataset = c("Parametric MICE", "Non-Parametric CART",  
 "Synthpop (Low Fidelity)", "Metadata-Based"),  
 Membership\_Inference\_Score = c(  
 calculate\_membership\_inference\_score(heart\_failure, syn\_data\_1),  
 calculate\_membership\_inference\_score(heart\_failure, syn\_cart\_1),  
 calculate\_membership\_inference\_score(heart\_failure, syn\_data\_low\_fidelity\_synthpop),  
 calculate\_membership\_inference\_score(heart\_failure, syn\_data\_metadata)  
 )  
) %>%  
 mutate(  
 Membership\_Inference\_Score = round(Membership\_Inference\_Score, 4),  
 Status = dplyr::case\_when(  
 is.na(Membership\_Inference\_Score) ~ "— Insufficient data",  
 Membership\_Inference\_Score == 0 ~ "✅ No risky memberships",  
 Membership\_Inference\_Score < 0.01 ~ "⚠️ Low risk",  
 TRUE ~ "❌ High risk"  
 )  
 )  
  
# ---- User-friendly table ----  
knitr::kable(  
 mis\_tbl,  
 caption = "Membership Inference Score: Proportion of synthetic records that are very close to real records (lower is better for privacy)",  
 align = "lrr"  
)

Membership Inference Score: Proportion of synthetic records that are very close to real records (lower is better for privacy)

| Dataset | Membership\_Inference\_Score | Status |
| --- | --- | --- |
| Parametric MICE | 0 | ✅ No risky memberships |
| Non-Parametric CART | 0 | ✅ No risky memberships |
| Synthpop (Low Fidelity) | 0 | ✅ No risky memberships |
| Metadata-Based | 0 | ✅ No risky memberships |

# ---- Plot only if any risky memberships (> 0) ----  
if (any(mis\_tbl$Membership\_Inference\_Score > 0, na.rm = TRUE)) {  
 pal <- c("#B0D99B", "#528AA8", "#FFB6DB", "#264653")  
   
 ggplot(mis\_tbl, aes(y = Dataset, x = Membership\_Inference\_Score, fill = Dataset)) +  
 geom\_col(alpha = 0.9, show.legend = FALSE, na.rm = TRUE) +  
 geom\_text(  
 aes(label = ifelse(is.na(Membership\_Inference\_Score), "NA",  
 scales::percent(Membership\_Inference\_Score, accuracy = 0.01))),  
 hjust = -0.1, size = 3.5  
 ) +  
 scale\_fill\_manual(values = pal[seq\_len(nrow(mis\_tbl))]) +  
 scale\_x\_continuous(labels = scales::percent\_format(accuracy = 1)) +  
 labs(  
 title = "Membership Inference Score by Synthetic Method",  
 subtitle = "Lower values indicate better privacy protection",  
 x = "Score (Proportion of risky memberships)",  
 y = NULL  
 ) +  
 theme\_minimal()  
} else {  
 cat("✅ All Membership Inference Scores are 0. No risky memberships detected. No plot is generated. \n")  
}

✅ All Membership Inference Scores are 0. No risky memberships detected. No plot is generated.

## 9 Conclusions

This report evaluated the fidelity, utility, privacy, and overall quality of synthetic data generated using parametric MICE, non-parametric CART, Synthpop, and metadata-based methods. Key findings include:

* **Data Structure Preservation**: All methods preserved the structure of the real dataset, though minor variations in variable distributions were observed.
* **Categorical Variables**: Most synthetic datasets retained categorical levels, but frequency distribution discrepancies were particularly notable in Synthpop data.
* **Numeric Variables**: The range and distribution of numeric variables were largely maintained across methods, with slight variations in density plots.
* **Correlation & Mutual Information**: While most methods captured variable relationships, weaker correlations and lower mutual information scores suggested some loss of feature dependencies in certain synthetic datasets.
* **Utility (Prediction Score)**: Synthetic data performed similarly to real data in predictive modeling, though slight reductions in accuracy and increased pMSE indicated less precise representation of the real dataset’s predictive capabilities.
* **Privacy (Exact Match & Neighbors’ Privacy Scores)**: All synthetic datasets demonstrated strong privacy protection, with zero values for both Exact Match and Neighbors’ Privacy Scores. This indicates that no synthetic records were identical to or highly similar to any real records, affirming robust privacy across all methods.

In conclusion, synthetic data generated through these methods provides a balance between utility and privacy, though method selection should align with specific analytic or privacy needs.

# Stop the parallel cluster  
stopCluster(cl)  
registerDoSEQ()