Kaggle Competition 1: Impute Missing Values

Joyce Lin, jlin99@uw.edu
Departments of Statistics, University of Washington

2024-02-07

1 Introduction

The ability to effectively handle missing data is crucial in many data analysis tasks, as missingness can introduce bias and reduce the accuracy of statistical models. In this Kaggle competition, our goal is to develop robust methods for imputing missing values in a data set provided to us, named `Data.csv`. All values in the dataset are numerical. Furthermore, the missing data in this data set is generated according to a specific mechanism, making it an interesting challenge to tackle.

The missing data generation process is defined such that the values of certain elements in the data set are missing if they exceed certain thresholds defined by vectors β and τ. That is, the i, j-th element of the matrix X, X_{ij}, is missing if and only if

$$X_i\beta > \tau_j.$$

This mechanism introduces a complex missingness pattern that requires careful consideration when designing imputation methods.

In this report, we will explore the missingness pattern in the provided data set and conduct preliminary analysis to better understand the characteristics of the missing data. We will then present different methods for imputing missing values. Our objective is to develop effective imputation techniques that can accurately recover missing information in the data set, thereby improving the quality and reliability of downstream analyses and modeling tasks.

2 Data Description

2.1 Missingness Pattern

The data set provided in Data.csv consists of 500 rows and 50 columns, forming a matrix structure. Missing values within the data set are denoted by NA. This data set serves as the basis for our analysis, where understanding and effectively managing these missing values are key objectives. The total number of missing values is 5178 out of 25000, representing 20.7% of the data set. The missingness pattern is illustrated in Figure 1. As shown in Figure 2, we observe that the variable with the highest number of missing values is Variable 1, with 244 missing entries (48.8%), while Variable 32 has the fewest missing values, with 32 entries missing (6.4%).

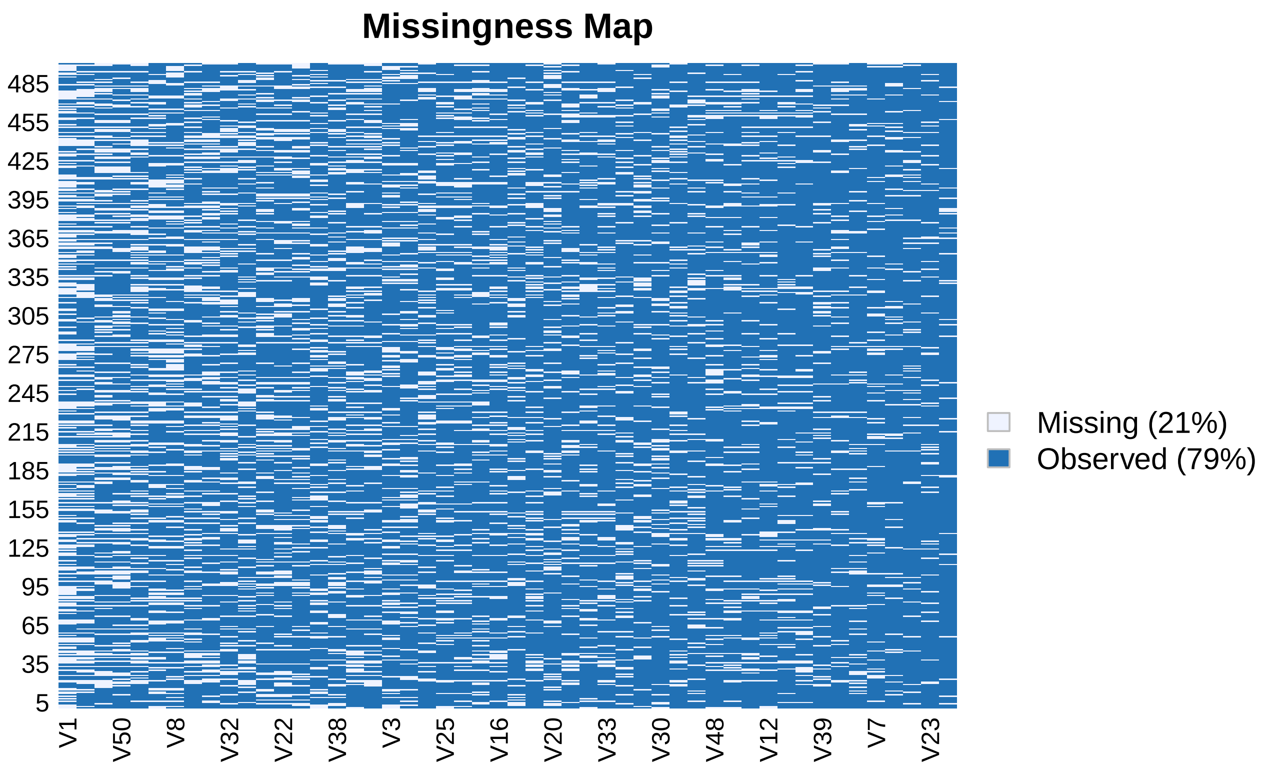

Figure 1: Missingness map of the data.

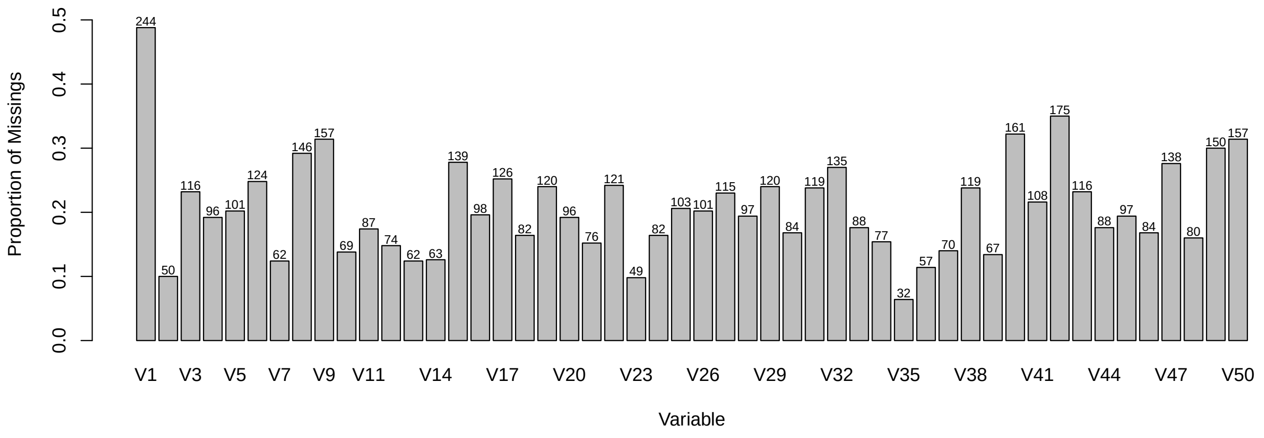

Figure 2: Proportion of missing values by variables.

2.2 Data Visualization

Exploring the distributions of each variable can be instrumental in identifying suitable imputation methods. As depicted in Figure 3, the mean of each variable remains relatively consistent, hovering around 0. However, there are notable variations in the variance across the 50 variables, with some exhibiting increasing and decreasing trends.

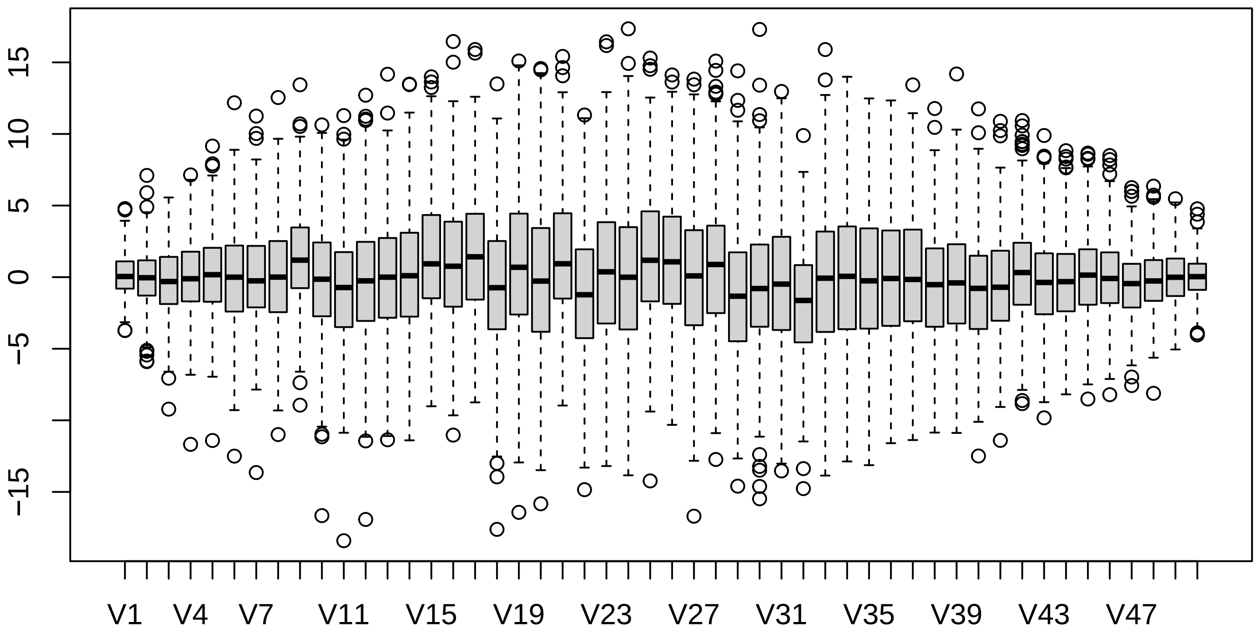

Figure 3: Boxplot for each variable in the data.

Furthermore, to examine the relationships between variables, the correlation between each pair of variables was computed using complete observations. As illustrated in Figure 4, significant correlations are observed among neighboring variables (e.g., V_i and V_{i+1} for $i = 1, \ldots, 50$), while correlations between distant variables (e.g., V_1 and V_{50}) are relatively low. This suggests a potential temporal or sequential structure within the data, with missing values potentially linked to time. Hence, exploring time series imputation methods may be warranted.

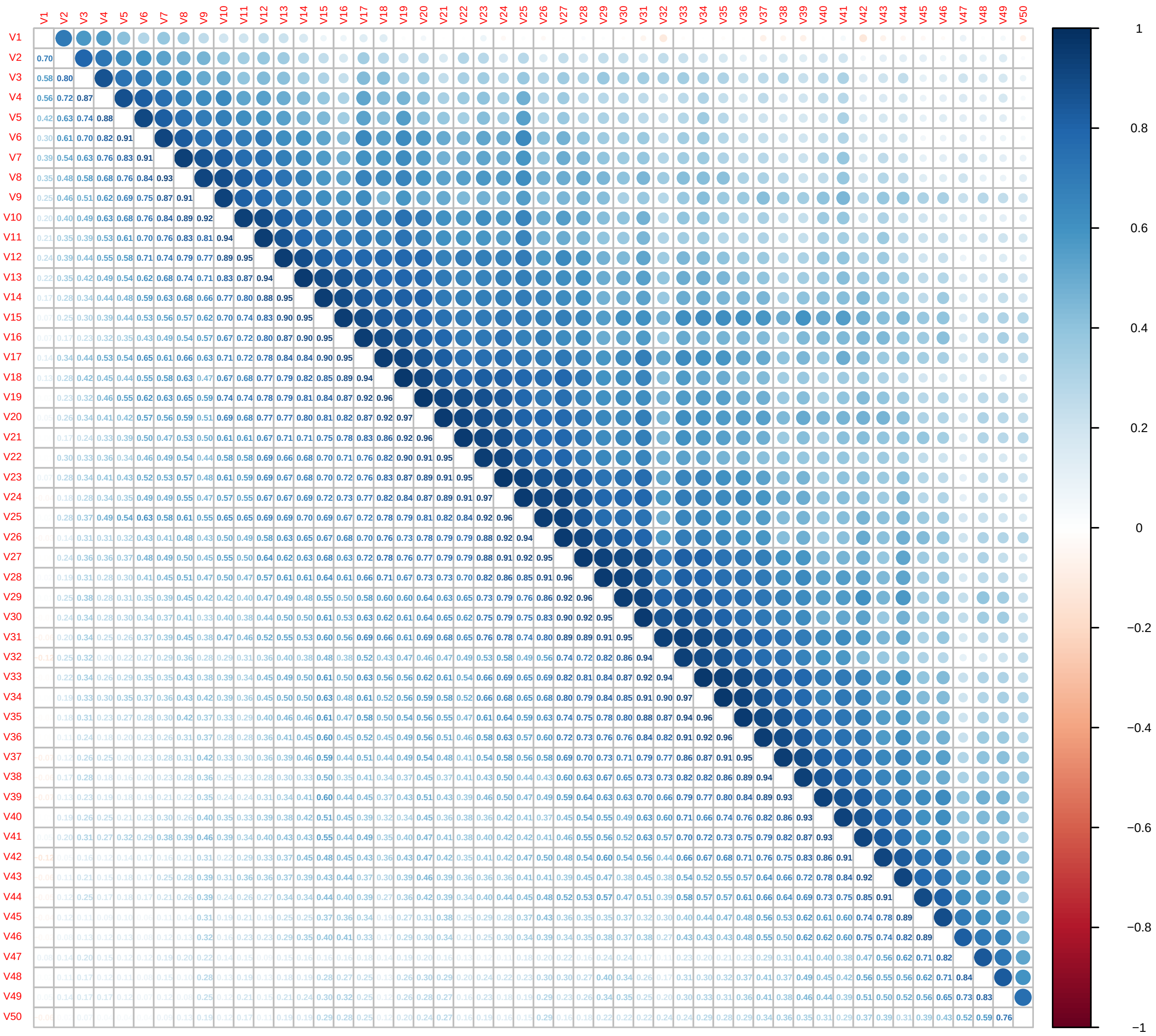

Figure 4: Correlation plot between each pair of variables.

3 Imputation Methods

3.1 `mice` package with `norm.predict` method

In this approach, missing values are imputed using the Multiple Imputation by Chained Equations (MICE) algorithm implemented in the `mice` package in `R`. The MICE algorithm is a flexible and widely used method for handling missing data by generating multiple imputed data sets based on chained equations.

The `norm.predict` method within the `mice` package employs linear regression imputation to estimate missing values. Specifically, it models each variable with missing values as a function of other variables in the data set, using a linear regression model. The estimated coefficients from the linear regression model are then used to predict missing values for each variable.

In this method, we set the argument `m` to 1, implying that only one imputed dataset is generated. The argument `maxit` is set to 100; in this case, increasing the iteration count can potentially improve the quality of the imputation results.

3.2 `CART` imputation by `impute_cart`

In this approach, missing values are imputed using a Classification and Regression Tree (CART). CART is a non-parametric machine learning method that recursively partitions the data into subsets based on the values of predictor variables, ultimately forming a decision tree structure. This decision tree is then used to predict missing values by traversing the tree and assigning values based on the terminal nodes reached by each observation.

We utilize the `impute_cart` function available in the `simputation` package in our study. As illustrated in the code snippet in the Appendix, the formula argument indicates that we are imputing all missing values in all variables based on every variable in the dataset.

3.3 Time Series Missing Value Imputation by `imputeTS`

Through the data visualization section, we observed indications of a potential temporal or sequential structure within the data, suggesting that missing values might be linked to time. As a result, employing time series imputation techniques becomes valuable for estimating these missing values. In this study, we leverage the `imputeTS` package to perform imputation on univariate time series. In essence, this means that the imputation is estimated based on the variables within each observation. I this section, we are going to use three function in `imputeTS` package, that is, `na_interpolation`, `na_locf`, `na_ma`.

Firstly, the `na_interpolation` function implements interpolation techniques such as linear interpolation or spline interpolation to estimate missing values based on the surrounding time series variables. Linear interpolation estimates missing values by drawing a straight line between two adjacent observed values and assigning the missing value as the point on the line corresponding to the time index of the missing value. Additionally, spline interpolation fits a piecewise polynomial function to the observed values and uses this function to estimate missing values. This method often provides smoother imputations compared to linear interpolation, especially when the underlying data exhibits nonlinear patterns.

Secondly, missing values are imputed using the Last Observation Carried Forward (LOCF) method through the `na_locf` function. This method replaces missing values with the most recent observed value that precedes it in time.

Lastly, we utilize `na_ma` to impute missing values using weighted moving average. In this study, we opt for exponential weighted moving average with a window of 4 observations (2 left and 2 right based on the central value) for the moving average window. This implies that observations directly adjacent to a central value i have a weight of $1/2^1$, the observations one further away $(i-2, i+2)$ have weight $1/2^2$, the next $(i-3, i+3)$ have weight $1/2^3$, and so forth.

3.4 Gaussian Conditional Imputation

The method employed for imputation involved estimating the parameters β and τ and utilizing the conditional distribution of Gaussians to impute missing values. Missing values, denoted as X_{ij}, were generated if and only if $X_i\beta > \tau_j$. To quantify missingness, we defined a binary missingness random variable R, where $R_{ij} = 1$ if X_{ij} is missing and 0 otherwise.

To estimate β and τ, we employed a logistic regression model:

$$\text{logit}(p_{ij}) = X_i\beta + \tau_j,$$

where p_{ij} represents the probability of $R_{ij} = 1$, and τ_j is the random intercept for each column. We utilized the imputed data from the `na_interpolation` imputation method to fit a generalized linear mixed-effects model with a random intercept. This allowed us to obtain estimates for both β and τ.

Next, we simulated sample values, denoted as $\tilde{X}_i$, from a multivariate normal distribution using the mean vector $\bar{X}_j$ and covariance matrix $\hat{\Sigma}$ from the previously imputed data. Given missing values where $R_{ij} = 1$, we imputed the sample value $\tilde{X}_i$ if $X_i\beta > \tau_j$. This procedure ensures that missing values are imputed based on the estimated parameters and the observed data structure.

It allows us to estimate β and τ through linear regression and subsequently impute missing values based on the conditional distribution of the Gaussian random variable given that it is missing. This method leverages the observed values of other variables to predict missing values, taking into account the correlations and relationships present in the data set.

4 Results

To evaluate the imputation results, we utilize the mean square error (MSE) metric. The MSE metric is used to evaluate the accuracy of imputation techniques, with lower values indicating better performance. Observing the MSE values presented in Table 1, it is evident that linear interpolation outperforms other methods in terms of imputation accuracy. This suggests that the data may possess a sequential or time-dependent structure, making linear interpolation a suitable choice for imputation. Additionally, methods such as spline interpolation and weighted moving average demonstrate competitive performance in handling missing values, showcasing their effectiveness in capturing underlying patterns within the data. However, it is noteworthy that the MSE of Gaussian conditional imputation stands at 33.2279, which is considerably higher compared to other imputation methods. This discrepancy highlights potential limitations or challenges associated with the Gaussian conditional imputation approach.

Table 1: Mean Square Error (MSE) values for various imputation methods.

Method	MSE
`mice` with `norm.predict` method	3.2673
CART	3.3955
Linear interpolation	1.2496
Spline interpolation	1.9509
Last observation carried forward	2.3640
Weighted moving average	1.4986
Gaussian conditional imputation	33.2279

5 Discussion

The results of our imputation analysis, as evaluated through the mean square error (MSE) metric, provide valuable insights into the effectiveness of different imputation methods in handling missing data. Our findings highlight the superiority of linear interpolation over other methods in terms of imputation accuracy. This suggests that the data set may exhibit a sequential or time-dependent structure, thereby rendering linear interpolation an effective choice for imputation. The robust performance of linear interpolation underscores its ability to capture and leverage the underlying temporal patterns within the data, resulting in accurate imputations.

Additionally, our analysis reveals competitive performance from spline interpolation and weighted moving average methods, both of which utilize time series imputation techniques. These methods demonstrate efficacy in handling missing values by capturing the underlying patterns within the data. Notably, the importance of data visualization cannot be overstated, as it was through techniques such as correlation plots that we observed the high correlation among neighboring variables, prompting further exploration into time series imputation methods.

Furthermore, our analysis included the utilization of the `mice` package with the `norm.predict` method and the machine learning imputation technique CART through the `impute_cart` function. While these methods did not perform as effectively as time series imputation techniques, they still presented viable options for handling missing data. The `mice` package with `norm.predict` method offers a flexible approach to imputation by leveraging multiple imputation techniques, yet its performance was not as strong as time series imputation methods. Similarly, the CART method showed promise in capturing complex relationships within the data, but its imputation accuracy fell short compared to time series techniques. Despite not achieving the same level of accuracy as time series imputation, these methods still provide valuable alternatives for imputing missing data in data sets where temporal patterns are less pronounced or when a more flexible approach to imputation is required.

Despite utilizing a sophisticated approach involving estimating parameters β and τ and leveraging the conditional distribution of Gaussians for imputation, the resulting MSE was substantially higher compared to other imputation methods employed in this study. This discrepancy suggests that while the method may theoretically capture the underlying patterns and relationships in the data, its performance in practice may be limited by the complexity of the model or the assumptions made during imputation. Further investigation into the factors contributing to this higher MSE may be warranted to refine and improve the effectiveness of this method in handling missing data.

Overall, our results underscore the importance of selecting appropriate imputation methods tailored to the characteristics of the data set. While linear interpolation proves to be effective in capturing linear sequential patterns, spline interpolation and weighted moving average methods offer viable alternatives, particularly when the data exhibits nonlinear patterns. Future research may explore the combination of multiple imputation techniques or advanced machine learning algorithms to further enhance imputation accuracy and robustness.

Appendix: R Code

```r
knitr::opts_chunk$set(echo = FALSE)
library(knitr)
library(Amelia)
library(corrplot)

MissingData=read.csv('impute-missing-values/Data.csv')

## 2.1    Missingness Pattern
# missmap
missmap(MissingData)

# barplot
nMiss = apply(MissingData, 2, function(x) sum(is.na(x)))
pMiss = nMiss/nrow(MissingData)
Miss.dt = data.frame(Variable = colnames(MissingData), pMiss = pMiss)
Miss.dt$Variable = factor(Miss.dt$Variable, levels = colnames(MissingData))
pMiss.np = barplot(pMiss ~ Variable, data = Miss.dt, ylim = c(0,0.55),
        xlab = "Variable", ylab = "Proportion of Missings")
text(pMiss.np, pMiss + 0.01, labels = nMiss, cex = 0.7)

## 2.2    Data Visualization
# boxplot
boxplot(MissingData)

# correlation plot
M = cor(MissingData, use = "pairwise.complete.obs")
corrplot.mixed(M, tl.pos = 'lt', tl.cex = 0.45, number.cex = 0.3, cl.cex = 0.5)

## 3.    Imputation Methods
# mice package with norm.predict method
library(mice)
mice.imp = mice(MissingData, m = 1, maxit = 100, method = "norm.predict", seed = 528)
CompleteData = as.matrix(complete(mice.imp, 1))

# CART imputation by impute_cart
library(simputation)
CompleteData = impute_cart(MissingData, formula = . ~ .)
CompleteData = as.matrix(CompleteData)

# Time Series Missing Value Imputation by imputeTS
library(imputeTS)
CompleteData = t(apply(MissingData, 1,
                       function(x) na_interpolation(unlist(as.vector(x)))))
CompleteData = t(apply(MissingData, 1,
                       function(x) na_interpolation(unlist(as.vector(x)),
                                                    option = "spline")))
CompleteData = t(apply(MissingData, 1,
                       function(x) na_locf(unlist(as.vector(x)))))
CompleteData = t(apply(MissingData, 1,
                       function(x) na_ma(unlist(as.vector(x)))))
```

```r
# Gaussian Conditional Imputation
#  - estimate beta and tau
library(lmer)
dt.R = ifelse(is.na(MissingData), 1, 0)
p = ncol(MissingData)
n = nrow(MissingData)
CompleteData = t(apply(MissingData, 1, function(x) na_interpolation(unlist(as.vector(x)))))
rownames(CompleteData) = 1:n
repeated_CompleteData = as.data.frame(cbind(R = as.numeric(dt.R),
                                            CompleteData[rep(rownames(CompleteData), each = p), ]))
repeated_CompleteData$colind = rep(1:50, times = n)
colnames(repeated_CompleteData)[2:51] = paste("V", 1:50, sep = "")
model <- glmer(R ~ -1 + V1 + V2 + V3 + V4 + V5 + V6 + V7 + V8 + V9 + V10 +
                 V11 + V12 + V13 + V14 + V15 + V16 + V17 + V18 + V19 + V20 +
                 V21 + V22 + V23 + V24 + V25 + V26 + V27 + V28 + V29 + V30 +
                 V31 + V32 + V33 + V34 + V35 + V36 + V37 + V38 + V39 + V40 +
                 V41 + V42 + V43 + V44 + V45 + V46 + V47 + V48 + V49 + V50 +
                 (1 | colind), data = repeated_CompleteData, family = binomial)
model.smry = summary(model)
tau = ranef(model)$colind
beta = model.smry$coefficients[,1]

#  - imputation using conditional distribution of Gaussians
library(mvtnorm)
X.cov = cov(CompleteData)
X.mean = apply(CompleteData, 2, mean)

set.seed(528)
CompleteData = MissingData
for (i in 1:n) {
  for (j in 1:p) {
    cat(i, j, "\n")
    if (dt.R[i, j]) {
      cond = TRUE
      while (cond) {
        X.imp = rmvnorm(1, mean = X.mean, sigma = X.cov)
        X.imp[which(!is.na(MissingData[i, ]))] = MissingData[i, which(!is.na(MissingData[i, ]))]
        cond = !(sum(unlist(X.imp) * beta) > tau[j, ])
      }
      CompleteData[i,j] = X.imp[j]
    }
  }
}
CompleteData = as.matrix(CompleteData)
```