Missing Data Random Forest Imputation for Different Missing Data Mechanisms.

Author: Faizan Khalid Mohsin, Course: CHL5250HY - Special Topics in Biostatistics, Professor: Wendy Lou.

January 14, 2021

Contents

1 Overview: **2**

2 Introduction **2**

3 Methods **2**

3.1 Data 2

3.1.1 Types of Missing data 2

3.1.2 Simulating the Data Set. 3

3.1.3 Introducing Missing Data 3

3.2 Missing Data Imputation Methods 3

3.2.1 Mean Imputation 3

3.2.2 Random Forest Imputation 3

3.3 Assessment of Missing Data Methods 5

3.4 Software 6

4 Results **6**

4.1 Imputation Results 6

4.2 Computational Efficiency 7

5 Discussion **7**

6 References **8**

7 Appendix **9**

7.1 All Packages Used. 9

7.2 Helper Functions 9

7.3 Running the imputations 14

7.4 Loading the Results 16

1 Overview:

In this paper we implement and assess the performance of Random Forest Imputation, in terms of accuracy; computational time; and ease of implementation, for two missing data types: Missing Completely at Random and Missing Not at Random. Further, we will asses the performance of Random Forest Imputation for different percentages of missing data: 10%, 20%, 30% and 40%. We will describe the theoretical framework and the algorithm of Random Forest Imputation and implement it on a simulated data set in which we will also simulate missing data. Finally, for comparison, we will also do the same assessment for the mean imputation method. We found that random forest imputation produce good and reliable imputed values and is able to capture and maintain linear as well as nonlinear structure of the data when imputing. Further, we found that Random Forest Imputation outperformed mean imputation for all percentages of missing data as well as for both missing data types, which is in line with what we expect. Further, random forest imputation is very easy to implement, has relatively very fast computational time, and is a flexible non-parametric imputation method, making it a highly recommended missing data imputation method.

2 Introduction

As data has become more ubiquitous so has the problem of missing data. This is also true for studies which are getting bigger and bigger in scope. To overcome this issue several missing data methods have been established over the years. In this paper will be looking at four missing data methods: complete case analysis, mean imputation, multiple imputation with chained equations, and Random Forest Imputation.

3 Methods

3.1 Data

3.1.1 Types of Missing data

We will describe the three types of missing data: missing completely at random (MCAR), missing at random (MAR), and missing not at random (MNAR).

The first type of missing data, missing completely at random (MCAR), as the name suggests, is missing completely at random. That is, there is no pattern in the missing data and the missing data is not in anyway associated with either the variable of interest or any other variable in the data set. MCAR impacts the analysis least negatively. Consider a study on people's blood pressure where blood pressure is the variable of interest and contains several covariates such as age. Then if the missingness in the data is due to people randomly missing a blood pressure measurement due to a random life event, then the missing data would be MCAR. The second type of missing data, missing at random (MAR) has missing data that depends on the covariates present in the data set and not on the variable of interest. For our blood pressure study, if older people are more prone to missing blood pressure measurements due to their old age such as reduced memory, transportation difficulty and need for assistance, then the missing blood pressure measurements would be MAR. The last type of missing data, missing not at random (MNAR), the missing data depends on the variable of interest. Another way to put this is that the missing data depends on the missing values themselves. For example, in the blood pressure study if higher blood pressure causes people to have worse health which causes them to miss blood pressure measurements and the higher the blood pressure the more often they miss their blood pressure measurements, then the missing data is MNAR. Such missing data can have a significant impact on the results and can seriously bias the analysis.

3.1.2 Simulating the Data Set.

We will create a data set with 1000 observations and 91 variables. For simplicity, all variables will be continuous with 90 variables treated as covariates and one as dependent.

3.1.3 Introducing Missing Data

We will introduce missing data using MCAR, MAR, and MNAR at the 10%, 20%, 30% and 40% missingness for each missing data type, hence giving us a total of 12 missing data sets for each of the two data sets. We have included the code to introduce missing data in the Appendix code section.

3.2 Missing Data Imputation Methods

3.2.1 Mean Imputation

One of the easiest methods to implement for imputing missing data is to simply do mean imputation. As the name suggests, the missing values in different variables are replaced with the mean value of the variable using values that are not missing in those variables. One major benefit of mean imputation is that the sample mean for the variables are not changed. Further, it is a very easy method, is simple to implement, and requires very low computational power.

However, mean imputation can have several major drawbacks. One major problem is that any correlations between variables that are imputed is reduced, almost always underestimating the correlation and hence the association between variables on which mean imputation is used. This is because, in cases with imputation, there is guaranteed to be no relationship between the imputed values of one variable with any imputed values of other variables. Another issue is that mean imputation can cause the standard error to be under estimated because replacing missing values with a single estimate will reduce the variability that should be observed in the data. By having low standard errors our p-values will also be low, hence we will be inadvertently making Type I error.

3.2.2 Random Forest Imputation

We will describe random forest imputation, the algorithm, the theoretical framework, the equations, and the formulas. We will use a few helpful diagrams for illustrating the algorithm.

Random forest is an ensemble method that combines many individual classification trees to give a more robust prediction. From the original sample several bootstrap samples are taken, and an unpruned classification tree is fit to each bootstrap sample. The variable selection for each split in the classification tree is conducted only from a small random subset of predictor variables, so that the curse of dimensionality (small n large p) is avoided. The final prediction of the model is the average of the results of all the individual tree models in the case of regression (continuous variables) or the mode (majority vote) in case of classification (categorical variable) (Strobl et al. 2007). Random forest for categorical variables and continuous variables is illustrated beneath in Figures 1 & 2 respectively.

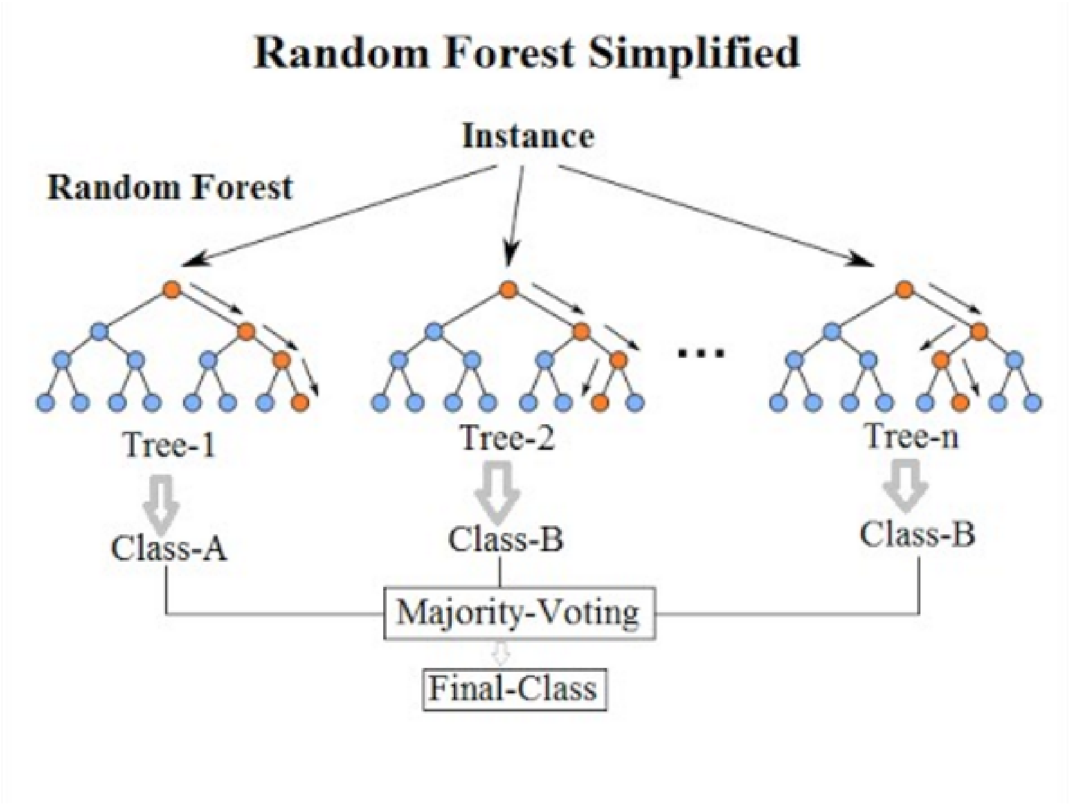

Figure 1: Random Forest for categorical variables.

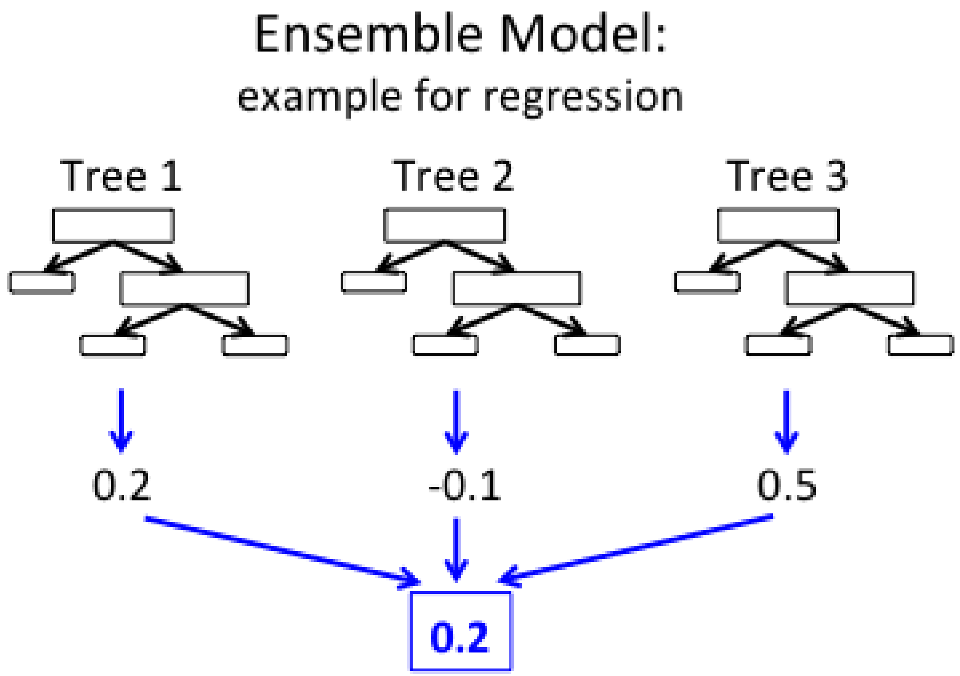

Figure 2: Random Forest for continuous variables.

The random forest imputation for missing data is implemented using the machine learning random forest algorithm which is an ensemble method. For this paper we will be going through the R package "missForest" implementation of RFImp.

Below is the random forest imputation algorithm from the missForest R package paper by Stekhoven and Bühlmann (2011).

Let $\mathbf{X} = (\mathbf{X_1}, \mathbf{X_2}, ..., \mathbf{X_p})$ be a $n \times p$ data matrix with missing values in the covariates where $\mathbf{X_s}$ is the s^{th} covariate. The missing values are predicted using a random forest trained on the non-missing parts of the dataset which we will called the "observed" values. Also, let $\mathbf{X_{\text{-}s}}$ be the data matrix $\mathbf{X}$ without the s^{th} covariate, hence $\mathbf{X_{\text{-}s}}$ has dimensions $n \times (p-1)$. Lastly, let $\mathbf{i}^{(s)}_{mis}$ denote the entries (rows) of the missing values in the $\mathbf{X_s}$ (the s^{th} covariate). Now all the data can be separated into four groups:

1. For missing values in $\mathbf{X_s}$ (the s^{th} covariate) let them be denoted by $\mathbf{y}^{(s)}_{mis}$.
2. The observed values in $\mathbf{X_s}$ let them be denoted by $\mathbf{y}^{(s)}_{obs}$.
3. The $\mathbf{i}^{(s)}_{mis}$ entries in $\mathbf{X_{\text{-}s}}$ let them be denoted by $\mathbf{x}^{(s)}_{mis}$. This is basically the rows of data in $\mathbf{X_{\text{-}s}}$ that correspond to the rows in $\mathbf{X_s}$ that have missing data.
4. Finally, let $\mathbf{x}^{(s)}_{obs}$ be the the observations in the rows of $\mathbf{X_{\text{-}s}}$ that correspond to the rows in $\mathbf{X_s}$ that do not have missing data. That is the $\mathbf{i}^{(s)}_{obs} = \{1, 2, ..., n\}/\mathbf{i}^{(s)}_{mis}$ entries in $\mathbf{X_{\text{-}s}}$.

Start with making an initial guess for the missing values in X (mean values for example). Then, sort the variables $\mathbf{X_s}$, s=1,. . . , p according to the amount of missing values starting with the lowest amount. For each variable $\mathbf{X_s}$, the missing values are imputed by first fitting an RF with response y(s)obs and predictors x(s)obs; then, predicting the missing values y(s)mis by applying the trained RF to x(s)mis. The imputation procedure is repeated until a stopping criterion is met.

Algorithm for imputing missing values with Random Forest.

```
1. Setup: $\mathbf{X}$ an $n \times p$ data matrix, and a stopping criterion $\omega$
2. Make initial guess for missing values;
3. **k** vector of sorted indices of columns in $\mathbf{X}$ w.r.t increasing amount of missing values:
4. **while** not $\omega$ **do**
5. . . $\mathbf{X}^{imp}_{old}$ store previously imputed matrix;
6. . . **for** s in **k** **do**
7. . . . . . Fit a random forest: $\mathbf{y}^{(s)}_{obs} \sim \mathbf{x}^{(s)}_{obs}$ ;
8. . . . . . Predict $\mathbf{y}^{(s)}_{mis} \sim \mathbf{x}^{(s)}_{mis}$ ;
9. . . . . . $\mathbf{X}^{imp}_{new}$ update imputed matrix, using predicted $\mathbf{y}^{(s)}_{mis}$ ;
10. . . **end for**
11. . . update $\omega$
12. **end while**
13. **return** the imputed matrix $\mathbf{X}^{imp}$
```

3.3 Assessment of Missing Data Methods

To assess the missing data methods we will perform a regression analysis on the imputed data set, the mse's from imputed data set will be divided by the mse from the corresponding complete data set. We will call this the "standardized mse" and use it to assess the performance of the imputation method bench marked to the performance of the complete data set.

To assess RFImp and mean imputation we fitted a linear regression on the imputed data sets at 10%, 20%, 30%, and 40% missingness and for the two types of missing data. Giving us 16 mse's in total. The mse's from these regression models were divided by the mse of the linear regression fitted on the original complete data set (data set without any missing data). We call this the "standardized mse" and use it to assess the performance of the imputation method bench marked to the performance of the complete data set. These are plotted in the Results Section in Figure 3.

The linear regression was simply a multiple linear regression with the 90 covariate variables used as the independent variables and the response variable as dependent variable.

3.4 Software

All analyses, as well as simulating the data and introducing the missing values is performed in R version 3.6.0. The "missForest" R package by Stekhoven and Bühlmann (2011) will be used for implementing Random Forest Imputation.

4 Results

4.1 Imputation Results

Below we present the standardized mse's in Tables 1 & 2 and are also plotted in Figure 3.

Table 1: Standardized MSE's of random forest imputation (RFImp) and mean imputation for different percentages of missing data for MCAR.

Missing Data (%)	Mean Imputation Error	RFImp Error
10	1.086	1.043
20	1.441	1.236
30	1.627	1.254
40	2.028	1.200

Table 2: Standardized MSE's of random forest imputation (RFImp) and mean imputation for different percentages of missing data for MNAR.

Missing Data (%)	Mean Imputation Error	RFImp Error
10	2.572	2.042
20	6.655	4.661
30	15.169	10.064
40	34.073	20.767

The standardized mse's tell us how much the mse for an imputed data set increased compared to the mse of the complete data set. Hence, for example in Table 2, the standardized mse of the mean imputation for 40% missing data is 34.073, meaning that the mse of the linear regression increases by 34.073 times when the imputation method is mean imputation for 40% missing data when compared to the complete data set.

Comparing this to the standardized mse for RFImp in Table 2 for 40% missing data, which is 20.767, it can be seen that RFImp performs much better even though from an absolute point of view, the mse increasing about 20 times is quite bad.

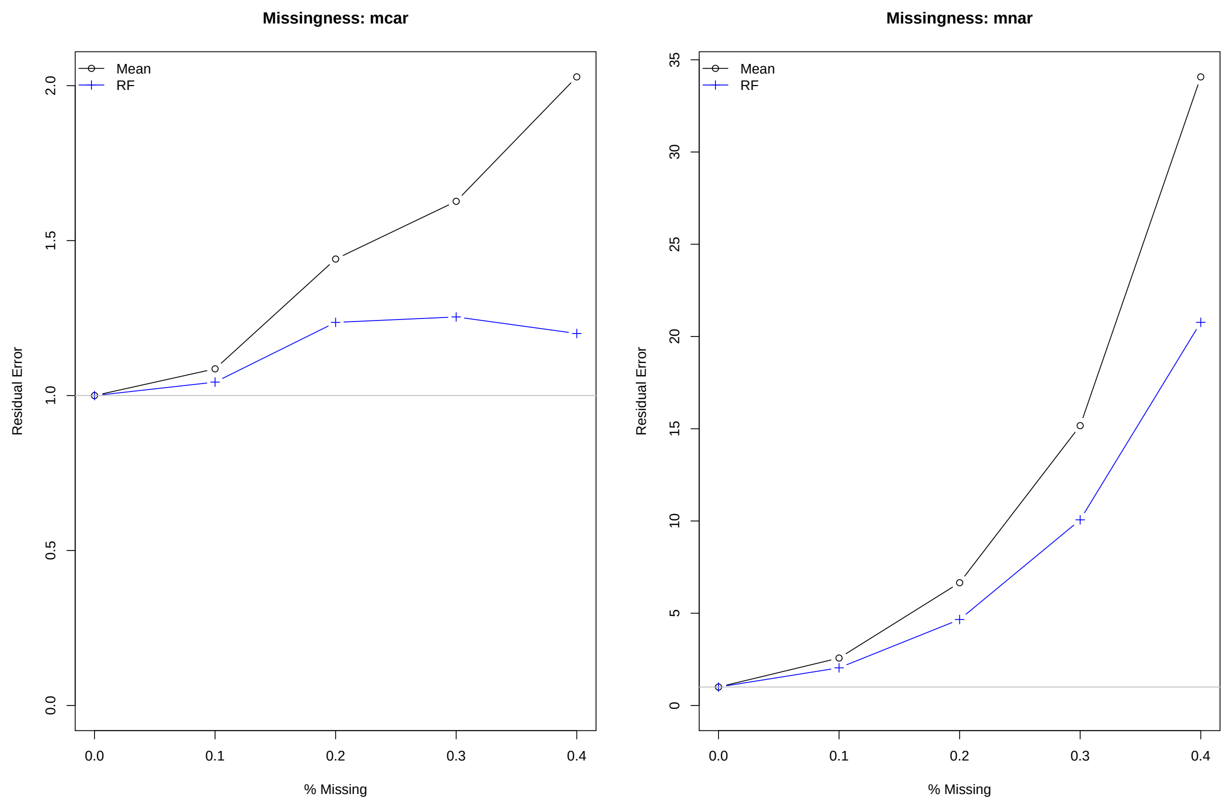

Figure 3: Random Forest Imputation for different percent and types of missing data.

4.2 Computational Efficiency

Below are the computational times for RFImp and mean imputation for missing data MNAR. The computational times for MCAR were very similar.

Table 3: Computational time for random forest imputation (RFImp) and mean imputation for MNAR.

Missing Data (%)	Mean Imputation (min)	RFImp (min)
10	0.012	3.800
20	0.015	2.782
30	0.011	2.349
40	0.011	3.140

5 Discussion

We can see that the RFimp performs pretty well when imputing missing data. Further, we can see from the results in Figure 3 that RFimp clearly out performs mean imputation for MCAR and MNAR types of missing data and also over all different percentages of missing data.

We find that for MCAR the residual error for RFImp very gradually increases as percentage of missing data increases. Also, based on our results on could say that the residual error stays constant for over 20% missing data. For mean imputation, however, its residual error increases as the percentage of missing data increases, and the increase appears to be linear. Further, and most importantly, at each percentage of missing data the residual error for RFImp is smaller compared to mean imputation.

For MNAR, we observe that RFImp and mean imputation residual errors are higher for all the missing data percentages when compared to MCAR. For example, RFImp residual error at 20% for MCAR is approximately 1.25, and for MNAR it is approximately 5. Hence, for the same data set and same percentage of missing data RFImp perform much poorly for MNAR missing data. We also we find that for both RFImp and mean imputation residual error increases as percentage of missing data increases and the increase appears to be exponential, however, RFImp residual error is always lower compared to mean imputation.

Therefore, for MCAR, random forest imputation works well for up to 40% missing data and out performs mean imputation. For MNAR even though it's residual error is high relatively when compared to MCAR, it outperforms mean imputation and hence is a better option to use when data is MNAR when compared to mean imputation.

In conclusion, random forest imputation is very easy to implement, has relatively very fast computational time, and is a flexible non-parametric imputation method, making it a highly recommended missing data imputation method for imputing missing data.

6 References

We will use the papers Strobl et al. (2007), Tang and Ishwaran (2017) and Hong and Lynn (2020) for referring to the theoretical framework of Random Forest Imputation. The papers Azur et al. (2011), Sterne et al. (2009) and Rubin (1996) will be used for referring to the theoretical framework of MICE imputation.

Azur, Melissa J, Elizabeth A Stuart, Constantine Frangakis, and Philip J Leaf. 2011. "Multiple Imputation by Chained Equations: What Is It and How Does It Work?" *International Journal of Methods in Psychiatric Research* 20 (1). Wiley Online Library: 40–49.

Hong, Shangzhi, and Henry S Lynn. 2020. "Accuracy of Random-Forest-Based Imputation of Missing Data in the Presence of Non-Normality, Non-Linearity, and Interaction." *BMC Medical Research Methodology* 20 (1). BioMed Central: 1–12.

Rubin, Donald B. 1996. "Multiple Imputation After 18+ Years." *Journal of the American Statistical Association* 91 (434). Taylor & Francis Group: 473–89.

Stekhoven, Daniel J., and Peter Bühlmann. 2011. "MissForest—non-parametric missing value imputation for mixed-type data." *Bioinformatics* 28 (1): 112–18. doi:10.1093/bioinformatics/btr597.

Sterne, Jonathan A C, Ian R White, John B Carlin, Michael Spratt, Patrick Royston, Michael G Kenward, Angela M Wood, and James R Carpenter. 2009. "Multiple Imputation for Missing Data in Epidemiological and Clinical Research: Potential and Pitfalls." *BMJ* 338. BMJ Publishing Group Ltd. doi:10.1136/bmj.b2393.

Strobl, Carolin, Anne-Laure Boulesteix, Achim Zeileis, and Torsten Hothorn. 2007. "Bias in Random Forest Variable Importance Measures: Illustrations, Sources and a Solution." *BMC Bioinformatics* 8 (1). BioMed Central: 25.

Tang, Fei, and Hemant Ishwaran. 2017. "Random Forest Missing Data Algorithms." *Statistical Analysis and Data Mining: The ASA Data Science Journal* 10 (6). Wiley Online Library: 363–77.

7 Appendix

7.1 All Packages Used.

```
knitr::opts_chunk$set(echo = T, eval = F)


rm(list=ls())

source('a1_functions_train.r')
load('a1_simulated_data.RData')

#library(installr)
#updateR()

#Load all packages
library(missForest)
require(tableone)
require(ggplot2)
require(UsingR)
require(glmnet)
require(knitr)
require(dplyr)
require(epiR)
require(class)
library(rpart)
library(tree)
library(pROC)
library(mice)
library(ISLR)
library(gplots)
library(xtable)
library(cluster)
library(corrplot)
library(doParallel)
require(kableExtra)
#library(doMC)
library(tools)
```

7.2 Helper Functions

```
check.dim <- function(data) {
    if (is.na(dim(data)[1])) {
        data <- matrix(data, length(data), 1)
    }
    return(data)
}

#impute using the mean (assuming it's all numeric)
mean.imp <- function(data) {
    data <- check.dim(data)
```

```r
    #stop('Complete this function.')
    #lapply(data, )
    for (i in 1:ncol(data)){
      data[is.na(data[,i]),i] = mean(data[,i], na.rm=TRUE)
    }
    #for each variable in data, impute the missing values using the mean of the available values
    return(data)
}

#add missingness to a matrix or dataframe
miss.set.df <- function(dat, prop, type.miss='mcar') {
    dat <- check.dim(dat)
    for (i in 1:dim(dat)[2]) {
        dat[,i] <- miss.set.vec(dat[,i], prop, type.miss)
    }
    return(dat)
}
# what is this ?miss.set.vec

#add misingness to a vector; mcar = at random; mnar = not at random (remove higher values)
miss.set.vec <- function(x, prop, type.miss='mcar') {
    n.miss <- rbinom(1, length(x), prop)
    if (type.miss == 'mcar') {
        miss.idx <- sample(1:length(x), n.miss, replace=F)
    } else if (type.miss == 'mnar') {
        miss.idx <- order(x, decreasing=T)[1:n.miss]
    }
    x[miss.idx] <- NA
    return(x)
}

split.data <- function(data, train.prop, set.seed=NA) {
    if (!is.na(set.seed)) {set.seed(set.seed)}
    train.idx <- sample(1:dim(data)[1], round(dim(data)[1]*train.prop), replace=F)
    test.idx <- setdiff(1:dim(data)[1], train.idx)
    train.set <- data[train.idx,]
    test.set <- data[test.idx,]
    return(list(train=train.set, test=test.set))
}


get.resid.err <- function(train.set, test.set) {
    #Train the models and get the MSE
    #stop('Complete this function.')

    #fit a linear model on the training set (train.set) using only the intercept Y ~ 1
  model1 = lm(Y ~1, data =  train.set)

    #fit a linear model on the training set using all covariates Y ~ .
  modelfull = lm(Y ~., train.set)

    #predict from each model on the test set (test.set)
  predict1 = predict(model1, newdata = test.set[,-91])
  predict.full = predict(modelfull, newdata = test.set[,-91])

    #calculate the mse of the null model
  mse.null = mean((predict1 - test.set$Y)^2)

    #calculate the mse of the full model
  mse.full = mean((predict.full - test.set$Y)^2)

    #calculate the residual error of the ratio of the latter over the former
    pred.res <- mse.full/ mse.null

    return(pred.res)
}



# Function to run the three imputations.

missing_function = function(miss_type_idx, miss_prop_idx, data_idx){

  train.prop <- 0.50
  train.split.seed <- 78901
  outcome.var <- "Y"
  rf.iter <- 12
  mice.iter <- 10

  miss.prop.list <- c(0.10, 0.20, 0.30, 0.40) #proportion of missingness to assign
  miss.type.list <- c('mcar', 'mnar') #mcar = missing completely at random; mnar = missing not at rando
  data.idx <- data_idx #the dataset to use
  miss.type.idx <- miss_type_idx #type of missingness to use
  miss.prop.idx <- miss_prop_idx #proportio of missingness to use

  t.st <- Sys.time() #time the run

  data <- data.list[[data.idx]]
  miss.prop <- miss.prop.list[miss.prop.idx]
  miss.type <- miss.type.list[miss.type.idx]

  split <- split.data(data, train.prop, set.seed=train.split.seed)
  train.set <- split$train; test.set <- split$test
  pred.var <- colnames(data)[-which(colnames(data) == outcome.var)]

  ##########################
  ## Set the missing data ##
  train.set.miss <- train.set
  train.set.miss[,pred.var] <- miss.set.df(train.set.miss[,pred.var], miss.prop, miss.type)

  dim(train.set.miss)
  length(pred.var)

  ################
  #MICE Imputation
  mice.data <- train.set.miss
```

```r
dim(mice.data)

mice.data0 = mice.data[,pred.var]
tempdata = mice(mice.data0,m=mice.iter,maxit=25,meth='pmm',seed=train.split.seed)

# Will take the exptectation of the imputated data sets to obtain one data set.
imputed = complete(tempdata, 1)
for (i in 2:mice.iter){
  print(i)
  print(dim(imputed))
  imputed = imputed + complete(tempdata, i)
  print(dim(imputed))
}

imputed = imputed / mice.iter


#run an imputation on the mice.data dataframe using the mice function from the mice package
#use 'mice.iter' datasets and cap the iterations at 25
#DO NOT INCLUDE the outcome 'Y' in the imputation, impute only mice.data[,pred.var]
#try to parallelize the computation by running several interations in parallel

mice.data.comp = data.frame(imputed, Y = mice.data$Y)
#write.csv(mice.data.comp, "mice_data_comp.csv", row.names = FALSE)
#mice.data.comp = read.csv("mice_data_comp.csv")
# same, but now as list, mild object
# dslist <- complete(tempdata, "all")
# length(dslist)
# imputed_list_data = mean(dslist)

# use package miceadds to save the imputed datasets.
# require(miceadds)
# write.mice.imputation(mi.res=tempdata, "tempdata1", include.varnames=TRUE,
#        long=TRUE, mids2spss=TRUE, spss.dec=",", dattype=NULL)

# # Parallalize mice imputation.
# total.cores <- detectCores()
# tempdata_core = parlmice(mice.data0, m = mice.iter, seed = NA, cluster.seed = 500,
#                          n.core = total.cores,  n.imp.core = NULL, cl.type = "PSOCK")
mice.data = mice.data.comp
t.mice = Sys.time() - t.st


t.st.mean = Sys.time()
#####################
#Impute with the mean
mean.data <- train.set.miss

#finish the mean.imp function from the functions file and impute mean.data[,pred.var]
# maybe should only use this mean.data[,pred.var]

head(is.na(mean.data))
is.na(dim(mean.data)[1])
```

```r
check.dim(mean.data)

# RUN the mean.imp function.
mean.data = mean.imp(mean.data)
all(!is.na(mean.data))

# Testing if the mean.imp() function works.
all(sapply(train.set.miss, function(x) mean(x, na.rm  = T)) == sapply(mean.data, mean))
summary(sapply(train.set.miss, function(x) mean(x, na.rm  = T)))

t.mean = Sys.time() - t.st.mean

t.st.rf = Sys.time()
#########################
#Random Forest Imputation
total.cores <- detectCores()
print(total.cores)
cl <- makeCluster(total.cores)
registerDoParallel(cl)

rf.data <- train.set.miss[,pred.var]
dim(rf.data)

#impute using random forest imputation, use 500 trees, and cap the number of iterations at 12 (rf.ite
#try to parallelize the forests using the doParallel package to save time
#?missForest

class(rf.data)
set.seed(train.split.seed)
rf.data.comp = missForest(xmis = rf.data, maxiter = rf.iter, ntree = 500, parallelize = c('forests'))
rf.data.comp.train = data.frame(rf.data.comp$ximp, Y = train.set.miss$Y)
#write.csv(rf.data.comp.train, "rf_data_comp_train_final.csv", row.names = F)
#rf.data.comp.train = read.csv("rf_data_comp_train_final.csv")
#file.remove("rf.data.comp.train.csv")
# if (file.exists(fn))
#   #Delete file if it exists
#   file.remove(fn)
rf.data = rf.data.comp.train

t.rf = Sys.time() - t.st.rf

##############################
#finish the get.resid.err function and calculate the test set errors for each imputed dataset
mean.imp.err <- get.resid.err(mean.data, test.set)
mice.imp.err <- get.resid.err(mice.data, test.set)
rf.imp.err <- get.resid.err(rf.data, test.set)
no.imp.err <- get.resid.err(train.set, test.set)

# mean.imp.err
# mice.imp.err
# rf.imp.err
# no.imp.err
```

```
  t.end = Sys.time() - t.st
  print(t.end)

  return(data.frame(mean.imp.err, mice.imp.err, rf.imp.err, no.imp.err, t.mice, t.mean, t.rf, t.end))

}
```

7.3 Running the imputations

```
#mse_data_1 = data.frame(mean.imp.err, mice.imp.err, rf.imp.err, no.imp.err)
# miss_type_idx <- 1 #type of missingness to use
# miss_prop_idx <- 1 #proportio of missingness to use
# data_idx <- 1

# TesT
# mse_missingtype1_missprop1_data1_v1 = missing_function(miss_type_idx=1, miss_prop_idx=1, data_idx=1)
# mse_missingtype1_missprop1_data1_v1



########

mse_missingtype1_data3 = data.frame()
for (i in 1:4){

  mse = missing_function(miss_type_idx=1, miss_prop_idx=i, data_idx=3)
  mse_missingtype1_data3 = rbind(mse_missingtype1_data3, mse)

}
dim(mse_missingtype1_data3)
View(mse_missingtype1_data3)
write.csv(mse_missingtype1_data3, "mse_missingtype1_data3.csv", row.names = F)

mse_missingtype2_data3 = data.frame()

for (i in 1:4){

  mse1 = missing_function(miss_type_idx=2, miss_prop_idx=i, data_idx=3)
  mse_missingtype2_data3 = rbind(mse_missingtype2_data3, mse1)

}
dim(mse_missingtype2_data3)
View(mse_missingtype2_data3)
write.csv(mse_missingtype2_data3, "mse_missingtype2_data3.csv", row.names = F)



########

mse_missingtype1_missprop1_data1_v1 = missing_function(miss_type_idx=1, miss_prop_idx=1, data_idx=1)
mse_missingtype1_missprop1_data1_v1
```

```r
mse_missingtype1_data3 = data.frame()
for (i in 1:4){

  mse = missing_function(miss_type_idx=1, miss_prop_idx=i, data_idx=3)
  mse_missingtype1_data3 = rbind(mse_missingtype1_data3, mse)

}
dim(mse_missingtype1_data3)
View(mse_missingtype1_data3)
write.csv(mse_missingtype1_data3, "mse_missingtype1_data3.csv", row.names = F)

mse_missingtype2_data3 = data.frame()

for (i in 1:4){

  mse1 = missing_function(miss_type_idx=2, miss_prop_idx=i, data_idx=3)
  mse_missingtype2_data3 = rbind(mse_missingtype2_data3, mse1)

}
dim(mse_missingtype2_data3)
View(mse_missingtype2_data3)
write.csv(mse_missingtype2_data3, "mse_missingtype2_data3.csv", row.names = F)


########

mse_missingtype1_missprop1_data1_v1 = missing_function(miss_type_idx=1, miss_prop_idx=1, data_idx=1)
mse_missingtype1_missprop1_data1_v1

mse_missingtype1_data3 = data.frame()
for (i in 1:4){

  mse = missing_function(miss_type_idx=1, miss_prop_idx=i, data_idx=3)
  mse_missingtype1_data3 = rbind(mse_missingtype1_data3, mse)

}
dim(mse_missingtype1_data3)
View(mse_missingtype1_data3)
write.csv(mse_missingtype1_data3, "mse_missingtype1_data3.csv", row.names = F)

mse_missingtype2_data3 = data.frame()

for (i in 1:4){

  mse1 = missing_function(miss_type_idx=2, miss_prop_idx=i, data_idx=3)
  mse_missingtype2_data3 = rbind(mse_missingtype2_data3, mse1)

}
dim(mse_missingtype2_data3)
View(mse_missingtype2_data3)
write.csv(mse_missingtype2_data3, "mse_missingtype2_data3.csv", row.names = F)
```

7.4 Loading the Results

```
#Repeat the above for every combination of missing proportion, type of missingness and data
#Put the results into a 3x5x2x3 array corresponding to the method (mean, mice, rf)x(missing proportion
#output.array <- array(0, dim=c(3, 5, 2, 3)) # method, missing prop, missing type, dataset

mse_missingtype1_data1 = read.csv("mse_missingtype1_data1.csv")
mse_missingtype2_data1 = read.csv("mse_missingtype2_data1.csv")
mse_missingtype1_data2 = read.csv("mse_missingtype1_data2.csv")
mse_missingtype2_data2 = read.csv("mse_missingtype2_data2.csv")
mse_missingtype1_data3 = read.csv("mse_missingtype1_data3.csv")
mse_missingtype2_data3 = read.csv("mse_missingtype2_data3.csv")


data_matrix_fn = function(data){

  no.miss.da = data$no.imp.err[1:3]

  final_data = as.matrix(cbind(no.miss.da, t(subset(data, select = c(1,2,3))) ))
  class(final_data)
  colnames(final_data) = c("missing0per", "missing10per", "missing20per",
                           "missing30per", "missing40per")
  rownames(final_data) = c("mean.imp.err", "mice.imp.err", "rf.imp.err")
  return(final_data)
}

# Test
#data_matrix_fn(mse_missingtype1_data2)

output.array <- array(0, dim=c(3, 5, 2, 3))

output.array[,, 1, 1] = data_matrix_fn(mse_missingtype1_data1)
output.array[,, 2, 1] = data_matrix_fn(mse_missingtype2_data1)
output.array[,, 1, 2] = data_matrix_fn(mse_missingtype1_data2)
output.array[,, 2, 2] = data_matrix_fn(mse_missingtype2_data2)
output.array[,, 1, 3] = data_matrix_fn(mse_missingtype1_data3)
output.array[,, 2, 3] = data_matrix_fn(mse_missingtype2_data3)

print(output.array)




output.array <- array(0, dim=c(3, 5, 2, 3))

output.array[,, 1, 1] = data_matrix_fn(mse_missingtype1_data1)/0.2429281
output.array[,, 2, 1] = data_matrix_fn(mse_missingtype2_data1)/0.2429281
output.array[,, 1, 2] = data_matrix_fn(mse_missingtype1_data2)/0.1845156
output.array[,, 2, 2] = data_matrix_fn(mse_missingtype2_data2)/0.1845156
output.array[,, 1, 3] = data_matrix_fn(mse_missingtype1_data3)/0.2480059
output.array[,, 2, 3] = data_matrix_fn(mse_missingtype2_data3)/0.2480059

print(output.array)
```

```
# mean.imp.err = rep(1,5)
# mice.imp.err = rep(2, 5)
# rf.imp.err = rep(3, 5)
# no.imp.err = rep(4, 5)
#
#
# dataframe1 = data.frame(mean.imp.err, mice.imp.err, rf.imp.err )
# data_frame1 = t(dataframe1)
# matrix1 = as.matrix(dataframe1)
# matrix1 = t(matrix1)
# print(output.array)
# vec = as.vector(matrix1)
# output.array[ , , 2, data_idx] = data_frame1
# output.array[,, miss_prop_idx, data_idx]
#
#
# output.array[,, miss_prop_idx, data_idx] = matrix1
# print(output.array)
```

```
rownames(mse_missingtype1_data2) = c(1, 2, 3, 4)
time_table = data.frame(MissingPercent = c(10, 20, 30, 40), mse_missingtype1_data2)

kable(time_table, caption = "The Run Times of different imputation methods for MCAR and data 2",format =
```

```
train.prop <- 0.50
train.split.seed <- 78901
outcome.var <- "Y"
rf.iter <- 12
mice.iter <- 10

miss.prop.list <- c(0.10, 0.20, 0.30, 0.40) #proportion of missingness to assign
miss.type.list <- c('mcar', 'mnar') #mcar = missing completely at random; mnar = missing not at random
# data.idx <- data_idx #the dataset to use
# miss.type.idx <- 1 #type of missingness to use
# miss.prop.idx <-1 #proportio of missingness to use

#t.st <- Sys.time() #time the run

# data <- data.list[[data.idx]]
# miss.prop <- miss.prop.list[miss.prop.idx]
# miss.type <- miss.type.list[miss.type.idx]

# split <- split.data(data, train.prop, set.seed=train.split.seed)
# train.set <- split$train; test.set <- split$test
# pred.var <- colnames(data)[-which(colnames(data) == outcome.var)]

###########################################
#Generate and save the plots of the results
miss.type.idx <- 1; data.idx <- 1
col.list <- c('black', 'blue', 'darkgreen')

pdf(paste0('error_ratio_plot_dataset.pdf'), height=15, width=10)
par(mfrow=c(3, 2))
```

```r
for (data.idx in 1:3) {

  for (miss.type.idx in 1:length(miss.type.list)) {

  x.lim=c(0, max(miss.prop.list))
  y.lim <- c(0, max(1, max(output.array[,,miss.type.idx,data.idx])))
  plot(c(0, miss.prop.list), output.array[1,,miss.type.idx, data.idx],
       main=paste('Dataset:', data.idx, 'Missingness:', miss.type.list[miss.type.idx]),
       xlab="% Missing", ylab='Residual Error', pch=NA, ylim=y.lim, xlim=x.lim)

    for (i in 1:3) {

        lines(c(0, miss.prop.list), output.array[i,,miss.type.idx, data.idx],
              lty=1, pch=i, type='b', col=col.list[i])
    }

  abline(h=1, col='gray')
  legend('topleft', legend=c('Mean', 'MICE', 'RF'), pch=1:3, col=col.list, lty=1, bty='n')

  }
}

dev.off()
```