HW1: Predictive Modeling

Faizan Khalid Mohsin (Student# 997157570), Syed M O Hasan (Student# 997068104)

May 14, 2019

Contents

Results

Below is a sample table of the MSE's and run times for dataset 2 for MCAR

Table 1: MSE and run times of different imputation methods with MCAR and data set 2.

Missing Data	**mean.imp.err**	**mice.imp.err**	**rf.imp.err**	**no.imp.err**	**Mean Imputation Time** (min)	**Mice Imputation Time** (min)	**RF Imputation Time** (min)
10%	0.2037	0.1990	0.1803	0.1845	0.011	15.593	2.491
20%	0.2412	0.2276	0.1937	0.1845	0.011	14.853	3.250
30%	0.3325	0.2179	0.1931	0.1845	0.028	15.637	3.235
40%	0.4664	0.2877	0.2029	0.1845	0.011	15.508	2.307

The run times for other data sets and missing data type were similar to this one.

Below are the MSE plots for the different imputation methods, datasets and types of missing data.

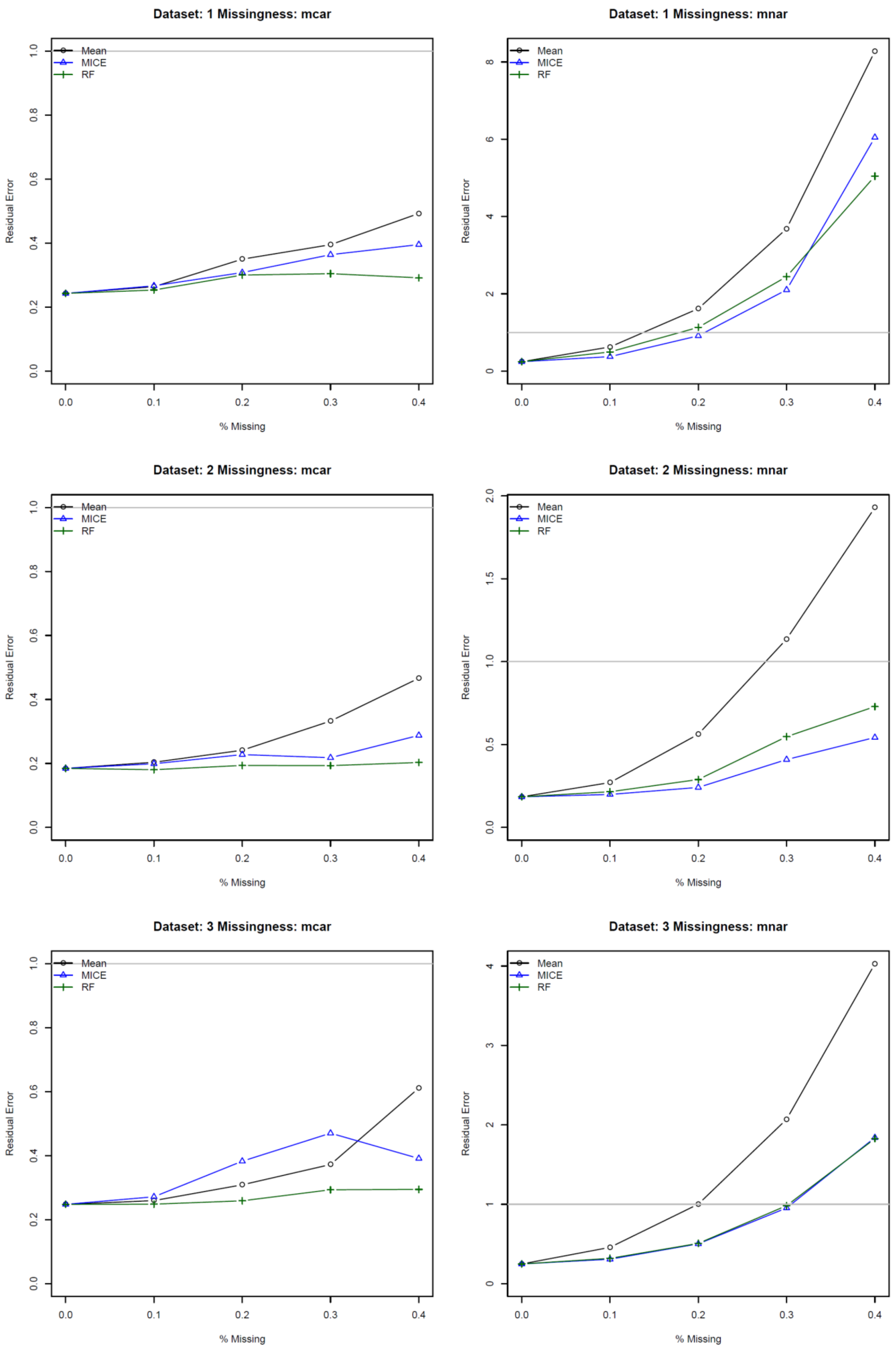

Figure 1. Residual errors versus missingness for all datasets, and the three imputation methods.

Discussion

Q1) Go back to the imputed training sets. Provide a measure of how effective MICE and random forest were relative to using the mean.

For missing data less than or equal to approximately 10%, all three methods work almost equally well. Hence, mean imputation is approximately as effective as MICE and RF. This is true for both types of missing data: missing completely at random (MCAR) and missing not at random (MNAR). Therefore, when dealing with up to 10% missing data, mean imputation would be sufficient to use.

When the missing data is around 20% we do start seeing MICE and Random Forest perform better than mean imputation for both types of missing data. However, for MNAR, MICE and RF perform markedly better than mean imputation and are much more effective.

Further, for 30% missing data or more, in general, we really see a clear improvement in performance with MICE and RF compared to the mean imputation. Hence, they are much more effective than mean imputation. The improvement in performance of MICE and RF compared to the mean imputation is even more pronounced when the data is MNAR. Further, the higher the missing data percentage, the more effective MICE and RF are compared to mean imputation.

Also, note that for 40% missingness, all imputation methods exhibit very high MSE's. In general, for MNAR, as the percent of missing data increases the MSE increases exponentially. For MCAR, the MSE curve increase approximately linearly. Hence, MICE and RF effectiveness compared to mean imputation increases with higher percent of missing data and when there is systematically more missing data (i.e. MCAR -> MNAR).

Q2) Provide a measure of how effective each of the three imputation methods was relative to the true values. Comment on the results for each dataset (imputation with random/non-random missingness, test set prediction error).

We would compare the MSE of the three imputation methods to the MSE of the training set without any missing data. For example, we would take the ratio of the MICE MSE over the MSE of the training set with no missing data. Another, way to do this would be to take the difference of the two MSE's. However, we only look at the ratio of the two MSE in this study and not at the difference in the interest of time.

The closer the ratio is to one (or lower than one) the better the imputation method performs in absolute terms by being closer to the true values. With a ratio smaller than one, it would mean that the imputation method MSE is even lower than that of the dataset without any missing data. This could be because through imputation, stronger relationships between the variables are created, which would not have been the case in the original data set, leading to smaller MSE. This simply would mean that in absolute terms the imputation method performs well. Hence, we have standardized the MSE's by dividing them by the MSE of the dataset without any missing data. These ratios are plotted in Figure 2 below.

For MCAR, all imputation methods seem to perform well in absolute terms since from the graph we can see that the ratios are close to 1, except for mean imputation when the percent of missing data is 40%, the ratio is greater or equal to 2.

For MNAR, when missing data is less than 10%, all imputation methods seem to perform well in absolute terms. However, when the percent of missing data is over 20%, we see very high ratios that are well above 1. For instance, dataset 1 exhibits ratios well above 5 for over 30% missing data.

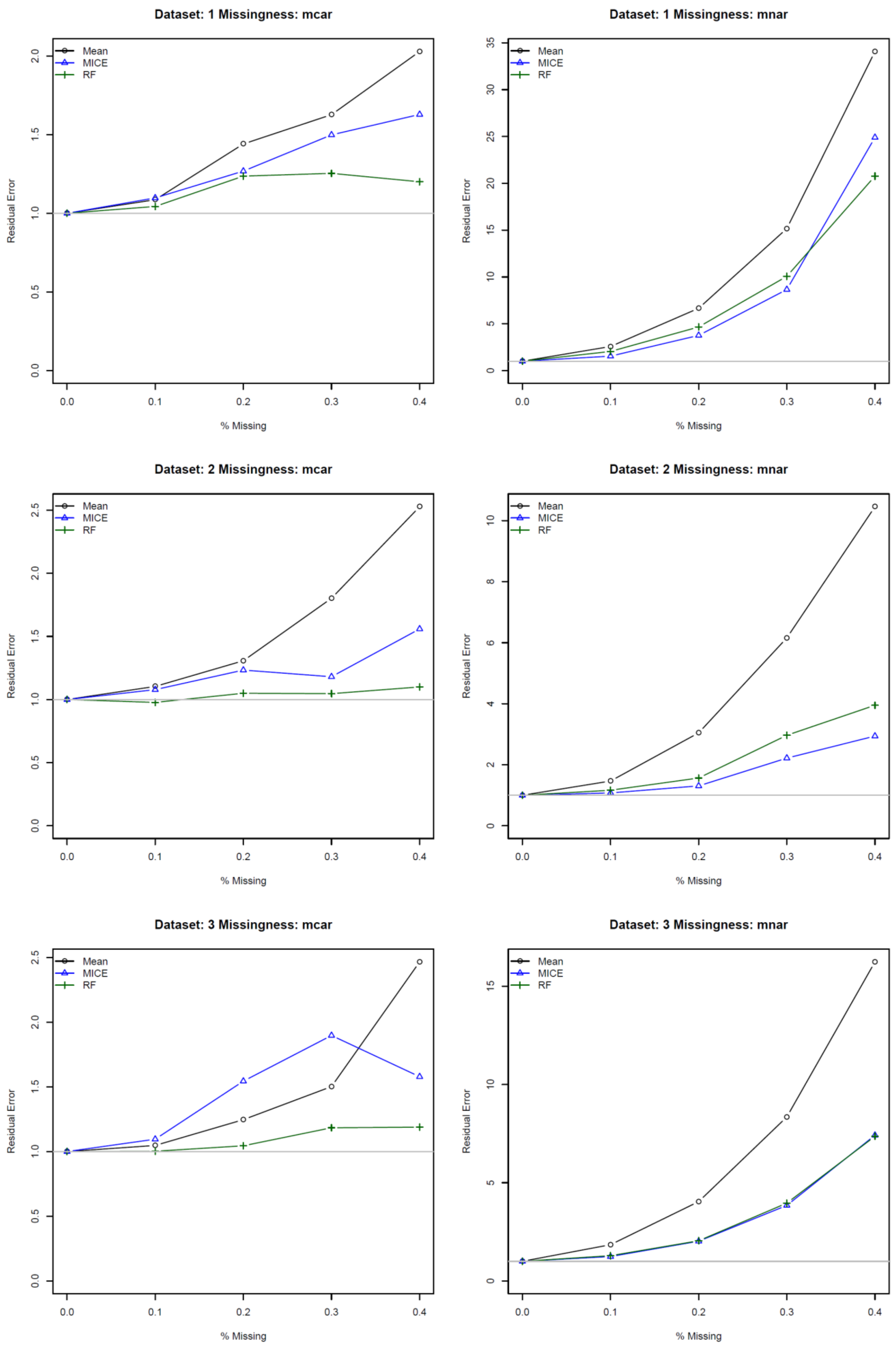

Figure 2: Ratios of imputation method MSEs over MSE from no missing datasets.

Q3) What would you do to improve some of the metrics?

We would standardize the MSE as we did by taking the ratio of the imputation method MSEs to the MSEs obtained from the datasets without any missing data, as shown in Figure 2.

Q4) Comment on the computation times for each method vs. their performance.

Run times for all the data sets were similar to those shown in Table 1 & 2.

For all the runs the RF imputation was close to 3min per run, MICE imputation was close to 11min and mean imputation was close to 0.012min (0.72 sec) per run. Hence, in general, mean imputation is 250 times faster than RF and RF is about 4 times faster than MICE.

Moreover, RF imputations, in general had smaller MSEs overall compared to the MICE and Mean imputation methods over the different percentages of missing data.

Therefore, in conclusion, RF imputation is the most recommended method in general because it is a non-parametric, and non-linear method with relatively fast computational times, especially under the conditions when the missing data percentage increases and the missingness is MNAR. In these conditions, RF is very cost effective in terms of performance and computational time over the improvement it provides.

Mean imputation is a very fast method computational, however, if the missing data is over 10-20% then it's performance starts to decline, especially for MNAR. Hence, it would only be recommended when missingness is less than 10%.

Lastly, MICE's computational time is 4 times more than RF and in general performs worse than RF in terms of MSE and is 1000 times slower than mean imputation. Hence, for missingness less than 10-20% we would recommend using mean imputation and for more than 20% missing data we would recommend RF.

Table 2: MSE and run times of different imputation methods with MNAR and data set 2.

Missing Data	**mean.imp.err**	**mice.imp.err**	**rf.imp.err**	**no.imp.err**	**Mean Imputation Time** (min)	**Mice Imputation Time** (min)	**RF Imputation Time** (min)	**Total Run Time**
10%	0.271	0.199	0.215	0.185	0.012	11.273	3.800	15.074
20%	0.562	0.241	0.289	0.185	0.015	11.142	2.782	13.925
30%	1.136	0.409	0.547	0.185	0.011	11.024	2.349	13.374
40%	1.930	0.542	0.729	0.185	0.011	12.280	3.140	15.421

R Code Used

```r
rm(list=ls())

knitr::opts_chunk$set(echo = TRUE)

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)
```

Helper Functions

```r
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)
    #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
```

```r
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)
```

```r
    #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 random
  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
  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])
  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.iter)
  #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))

}
```

Running the imputations

```
######## dataset 1


### MCAR


mse_missingtype1_data1 = data.frame()
for (i in 1:4){

  mse = missing_function(miss_type_idx=1, miss_prop_idx=i, data_idx=1)
  mse_missingtype1_data1 = rbind(mse_missingtype1_data1, mse)

}
dim(mse_missingtype1_data1)
View(mse_missingtype1_data1)
write.csv(mse_missingtype1_data1, "mse_missingtype1_data1.csv", row.names = F)


### MNAR


mse_missingtype2_data1 = data.frame()

for (i in 1:4){

  mse1 = missing_function(miss_type_idx=2, miss_prop_idx=i, data_idx=1)
  mse_missingtype2_data1 = rbind(mse_missingtype2_data1, mse1)

}
dim(mse_missingtype2_data1)
View(mse_missingtype2_data1)
write.csv(mse_missingtype2_data1, "mse_missingtype2_data1.csv", row.names = F)



########  dataset 2


### MCAR


mse_missingtype1_data2 = data.frame()
for (i in 1:4){

  mse = missing_function(miss_type_idx=1, miss_prop_idx=i, data_idx=2)
  mse_missingtype1_data2 = rbind(mse_missingtype1_data2, mse)
```

```r
}
dim(mse_missingtype1_data2)
View(mse_missingtype1_data2)
write.csv(mse_missingtype1_data2, "mse_missingtype1_data2.csv", row.names = F)


### MNAR


mse_missingtype2_data2 = data.frame()

for (i in 1:4){

  mse1 = missing_function(miss_type_idx=2, miss_prop_idx=i, data_idx=2)
  mse_missingtype2_data2 = rbind(mse_missingtype2_data2, mse1)

}
dim(mse_missingtype2_data2)
View(mse_missingtype2_data2)
write.csv(mse_missingtype2_data2, "mse_missingtype2_data2.csv", row.names = F)


######## dataset 3


### MCAR


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)


### MNAR


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)
```

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 = 0, 0.1, 0.2, 0.3, 0.4)x(random/non-random
missingness)x(dataset)
#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)

## , , 1, 1
##
##           [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 0.2429281 0.2638898 0.3499656 0.3952079 0.4927500
## [2,] 0.2429281 0.2666556 0.3080602 0.3639899 0.3955587
## [3,] 0.2429281 0.2534760 0.3003149 0.3045787 0.2915782
##
## , , 2, 1
##
##           [,1]      [,2]      [,3]     [,4]     [,5]
## [1,] 0.2429281 0.6247241 1.6167804 3.684977 8.277186
## [2,] 0.2429281 0.3746770 0.9131942 2.103041 6.048593
## [3,] 0.2429281 0.4961054 1.1322874 2.444891 5.044907
##
## , , 1, 2
##
##           [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 0.1845156 0.2036723 0.2412122 0.3325233 0.4664111
## [2,] 0.1845156 0.1989914 0.2275556 0.2178520 0.2877130
## [3,] 0.1845156 0.1802983 0.1936946 0.1931198 0.2029362
##
## , , 2, 2
##
##           [,1]     [,2]      [,3]      [,4]      [,5]
## [1,] 0.1845156 0.270549 0.5619149 1.1361373 1.9296712
## [2,] 0.1845156 0.198730 0.2411662 0.4092623 0.5424927
## [3,] 0.1845156 0.214959 0.2889397 0.5472991 0.7289057
##
## , , 1, 3
##
##           [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 0.2480059 0.2597110 0.3091038 0.3725483 0.6115577
## [2,] 0.2480059 0.2716757 0.3829260 0.4704838 0.3914477
## [3,] 0.2480059 0.2486160 0.2590704 0.2936104 0.2948012
##
## , , 2, 3
##
##           [,1]      [,2]      [,3]      [,4]     [,5]
## [1,] 0.2480059 0.4576930 1.0016544 2.0703111 4.028037
## [2,] 0.2480059 0.3081545 0.5017168 0.9536186 1.839333
## [3,] 0.2480059 0.3187547 0.5072816 0.9825349 1.825952

# 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 = "html" )
```

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

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')

  }
}
```