Question 14.1

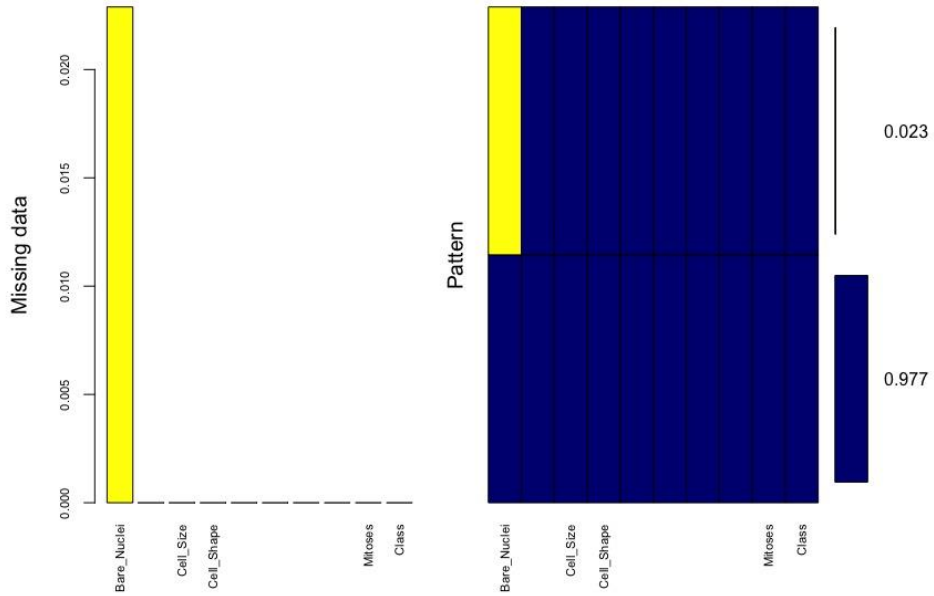
The breast cancer data set breast-cancer-wisconsin.data.txt from
http://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/
(description at
http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29) has missing values.
1. Use the mean/mode imputation method to impute values for the missing data.
2. Use regression to impute values for the missing data.
3. Use regression with perturbation to impute values for the missing data.
4. (Optional) Compare the results and quality of classification models (e.g., SVM, KNN) build using
(1) the data sets from questions 1,2,3;
(2) the data that remains after data points with missing values are removed; and (3) the data set when a binary variable is introduced to indicate missing values.

Summary :

First find the % of observations with missing data

0.0229 which is less than 5% so ok to use data imputation

Bare,Nuclei has 16 missing values



Variables sorted by number of missings:

Variable Count

Bare_Nuclei 0.02288984

Clump_Thickness 0.00000000

Cell_Size 0.00000000

Cell_Shape 0.00000000

Marginal_Adhesion 0.00000000

Single_Epith_Cell_Size 0.00000000

Bland_Chromatin 0.00000000

Normal_Nucleoli 0.00000000

Mitoses 0.00000000

Class 0.00000000

2 > imp regression

Class: mids

Number of multiple imputations: 1

Imputation methods:

Clump_Thickness Cell_Size Cell_Shape Marginal_Adhesion Single_Epith_Cell_Size

"" "" "" "" ""

Bare_Nuclei Bland_Chromatin Normal_Nucleoli Mitoses Class

"norm.predict" "" "" "" ""

PredictorMatrix:

	Clump_Thickness	Cell_Size	Cell_Shape	Marginal_Adhesion	Single_Epith_Cell_Size	Bare_Nuclei	Bland_Chromatin
Clump_Thickness	0	1	1	1	1	1	1
Cell_Size	1	0	1	1	1	1	1
Cell_Shape	1	1	0	1	1	1	1
Marginal_Adhesion	1	1	1	0	1	1	1
Single_Epith_Cell_Size	1	1	1	1	0	1	1
Bare_Nuclei	1	1	1	1	1	0	1

	Normal_Nucleoli	Mitoses	Class
Clump_Thickness	1	1	1
Cell_Size	1	1	1
Cell_Shape	1	1	1
Marginal_Adhesion	1	1	1
Single_Epith_Cell_Size	1	1	1
Bare_Nuclei	1	1	1

3> Imp perturbation

Class: mids

Number of multiple imputations: 1

Imputation methods:

Clump_Thickness Cell_Size Cell_Shape Marginal_Adhesion Single_Epith_Cell_Size

```
          ""           ""           ""           ""           ""
  Bare_Nuclei Bland_Chromatin Normal_Nucleoli      Mitoses        Class
     "norm.nob"           ""           ""           ""           ""
```

PredictorMatrix:

```
                Clump_Thickness Cell_Size Cell_Shape Marginal_Adhesion Single_Epith_Cell_Size
Bare_Nuclei Bland_Chromatin
Clump_Thickness               0         1          1                 1                      1
1              1
Cell_Size                     1         0          1                 1                      1
1              1
Cell_Shape                    1         1          0                 1                      1
1              1
Marginal_Adhesion             1         1          1                 0                      1
1              1
Single_Epith_Cell_Size        1         1          1                 1                      0
1              1
Bare_Nuclei                   1         1          1                 1                      1
0              1
                Normal_Nucleoli Mitoses Class
Clump_Thickness               1       1     1
Cell_Size                     1       1     1
Cell_Shape                    1       1     1
Marginal_Adhesion             1       1     1
Single_Epith_Cell_Size        1       1     1
Bare_Nuclei                   1       1     1
```

KNN Model :

Accuracy of Confusion Matrix

```
#Data_imput_w_regression_w_perturbation
```

98.28

#Data with rows w/ missing values removed

97.66

RSCRIPT

```
# ---------- Code Answer for Question 11.1 ----------

# Clear environment

rm(list = ls())

# Setting the random number generator seed so that our results are reproducible

# (Your solution doesn't need this, but it's usually good practice to do)

set.seed(100)

# ---------- Data manipulation ----------

# First, read in the data

#

c_data <- read.csv("breast-cancer-wisconsin.data.txt",header = FALSE,na.strings = "?")
```

```
#
# Optional check to make sure the data is read correctly
#
str(c_data)
print(summary(c_data))
#
#http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29
#Attribute Information:
#1. Sample code number: id number
#2. Clump Thickness: 1 - 10
#4. Uniformity of Cell Shape: 1 - 10
#5. Marginal Adhesion: 1 - 10
#6. Single Epithelial Cell Size: 1 - 10
#7. Bare Nuclei: 1 - 10
#8. Bland Chromatin: 1 - 10
#9. Normal Nucleoli: 1 - 10
#10. Mitoses: 1 - 10
#11. Class: (2 for benign, 4 for malignant)



# Add Column Name
colnames(c_data) <- c("ID", "Clump_Thickness", "Cell_Size", "Cell_Shape",
                      "Marginal_Adhesion", "Single_Epith_Cell_Size", "Bare_Nuclei",
                      "Bland_Chromatin","Normal_Nucleoli", "Mitoses", "Class")
require(dplyr)
Data <- c_data %>%mutate(Class = ifelse(Class == 4,1,0))
```

```
drops <- c("ID")
Data <- c_data[ , !(names(Data) %in% drops)]
summary(Data)

print(summary(Data))

# From the above data, Bare,Nuclei has 16 missing values
library(VIM)
mice_plot <- aggr(Data, col=c('navyblue','yellow'),numbers=TRUE, sortVars=TRUE,
                  labels=names(Data), cex.axis=.7,gap=3, ylab=c("Missing data","Pattern"))

# There are 97.8% with no missing values.

# Mean Mode
Data_imput_w_mean <- c_data
Data_imput_w_mean$Bare_Nuclei[is.na(Data_imput_w_mean$Bare_Nuclei)] <-
mean(Data_imput_w_mean$Bare_Nuclei, na.rm=TRUE)


Data_imput_w_mode <- c_data
Data_imput_w_mode$Bare_Nuclei[is.na(Data_imput_w_mean$Bare_Nuclei)] <-
mode(Data_imput_w_mean$Bare_Nuclei)


# Use regression to impute missing values
library(mice)
Data_imput_w_regression <- Data
```

```
imp <- mice(Data, method="norm.predict",m=1)
imp

Data_imput_w_regression <- complete(imp)
#Use regression with perturbation impute values

imp_perturbation <- mice(Data, method="norm.nob",m=1)
imp_perturbation

# list the actual imputations for Bare_Nuclei

Data_imput_w_regression_pert <- complete(imp_perturbation)



#4. (Optional) Compare the results and quality of classification models (e.g., SVM, KNN) build
#using
#(1) the data sets from questions 1,2,3;
#(2) the data that remains after data points with missing values are removed; and (3) the
#data set when a binary variable is introduced to indicate missing values.

# Data with Mode Imputation

split_test_train <- function(df) {
  set.seed(1)
train = sample(1:nrow(df), 0.75*nrow(df))
train_df <- df[train,]
test_df <- df[-train,]
```

```
return (list("train" = train_df, "test" = test_df))}

result_w_mean <- split_test_train(Data_imput_w_mean)

summary(result_w_mean$train)


dim(result_w_mean$train)

dim(result_w_mean$test)

data_train_target_category <- result_w_mean$train[,"Class"]

data_test_target_category <- result_w_mean$test[,"Class"]

#run KNN models

library(class)

pr <- knn(result_w_mean$train,result_w_mean$test, cl=data_train_target_category)

pr

##create confusion matrix

cf <- table(pr,data_test_target_category)

cf

##this function divides the correct predictions by total number of predictions that tell us how
#accurate the model is.

accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}

accuracy(cf)



# Data Imputed with regression

result_w_regression_pert <- split_test_train(Data_imput_w_regression_pert)

summary(result_w_regression_pert$train)

data_train_target_category <- result_w_regression_pert$train[,"Class"]

data_test_target_category <- result_w_regression_pert$test[,"Class"]

pr <- knn(result_w_regression_pert$train,result_w_regression_pert$test,
cl=data_train_target_category)
```

```
pr

##create confusion matrix
cf <- table(pr,data_test_target_category)
cf
accuracy(cf)



#Data with rows w/ missing values removed
data_w_no_na = Data[complete.cases(Data), ]
dim(data_w_no_na)
result_w_no_na <- split_test_train(data_w_no_na)
data_train_target_category <- result_w_no_na$train[,"Class"]
data_test_target_category <- result_w_no_na$test[,"Class"]
pr <- knn(result_w_no_na$train,result_w_no_na$test, cl=data_train_target_category)
pr

##create confusion matrix
cf <- table(pr,data_test_target_category)
cf
accuracy(cf)
```