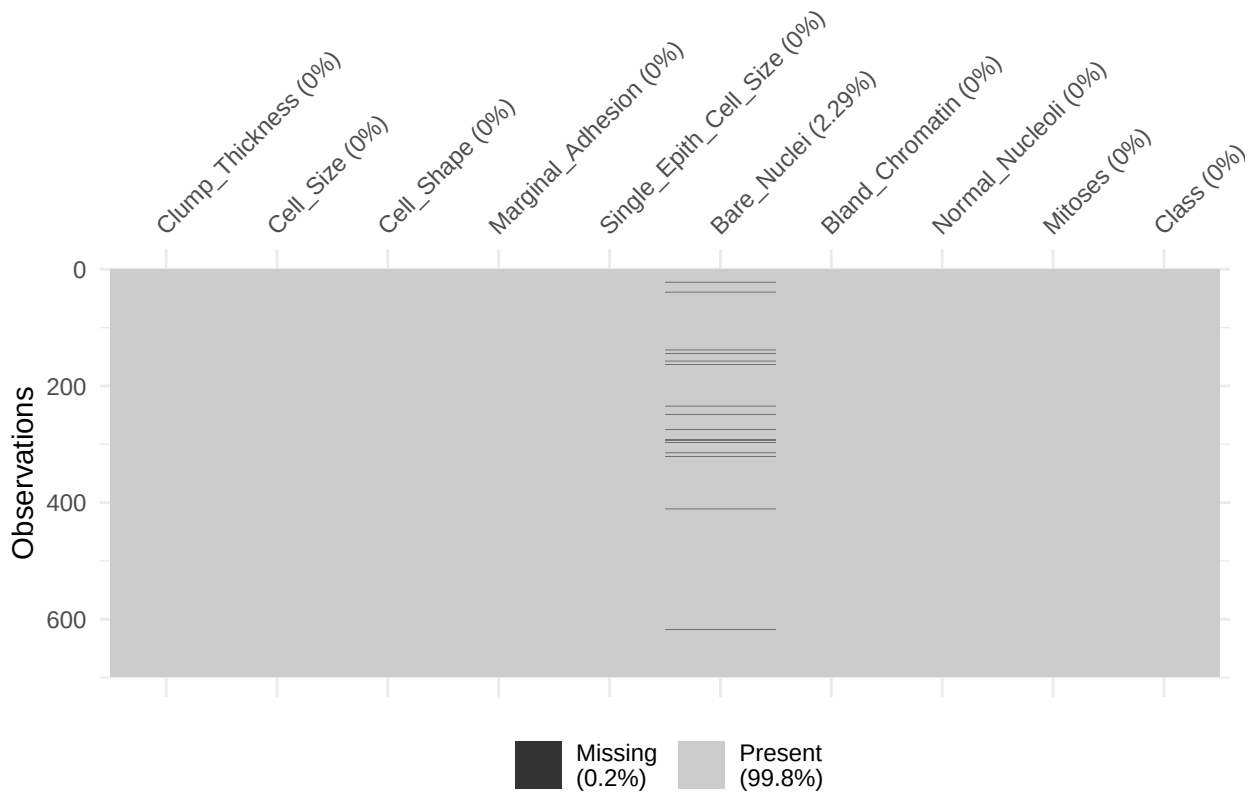
Question 14.1

Breaking down the question so that I won't miss any parts, the way I did in the last 2 assignments.

Dataset has missing data.

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
    - the data sets from questions 1,2,3;
    - the data that remains after data points with missing values are removed;
    - the data set when a binary variable is introduced to indicate missing values.

```
set.seed(123)
data <- read.table("http://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/br
                   , header = TRUE
                   , sep=","
                   , na.strings = "?")
# add column names
colnames(data) <- c("ID", "Clump_Thickness", "Cell_Size", "Cell_Shape",
                "Marginal_Adhesion", "Single_Epith_Cell_Size", "Bare_Nuclei",
                "Bland_Chromatin","Normal_Nucleoli", "Mitoses", "Class")

# drop id column because unneccessary
data$ID <- NULL

data <- data %>%
  mutate(Class = ifelse(Class == 4,1,0))

#summary(data)

# set formula for later use
formula <- Class ~ Clump_Thickness+Cell_Size+Cell_Shape+Marginal_Adhesion+Single_Epith_Cell_Size+Bare_Nu

# check for missing data
vis_miss(data)
```



The dataset had missing values marked with `?`, all of which are located in the `Bare_Nuclei` column (n = 16).

```
# baseline model
fit <- with(data, lm(Class ~ Clump_Thickness+Cell_Size+Cell_Shape+Marginal_Adhesion+Single_Epith_Cell_S

# inspect baseline model
summary(fit)
```

```
##
## Call:
## lm(formula = Class ~ Clump_Thickness + Cell_Size + Cell_Shape +
##     Marginal_Adhesion + Single_Epith_Cell_Size + Bare_Nuclei +
##     Bland_Chromatin + Normal_Nucleoli + Mitoses)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.83997 -0.08295 -0.01225  0.05679  0.76358
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)            -0.2476295  0.0164330 -15.069  < 2e-16 ***
## Clump_Thickness         0.0317829  0.0035685   8.907  < 2e-16 ***
## Cell_Size               0.0218135  0.0063766   3.421 0.000662 ***
## Cell_Shape              0.0155989  0.0062402   2.500 0.012666 *
## Marginal_Adhesion       0.0082314  0.0039947   2.061 0.039728 *
```

```
## Single_Epith_Cell_Size  0.0100750  0.0052405   1.923 0.054959 .
## Bare_Nuclei             0.0453514  0.0032233  14.070  < 2e-16 ***
## Bland_Chromatin         0.0192476  0.0050488   3.812 0.000150 ***
## Normal_Nucleoli         0.0185094  0.0037221   4.973 8.39e-07 ***
## Mitoses                 0.0009794  0.0049697   0.197 0.843824
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1903 on 672 degrees of freedom
##   (16 observations deleted due to missingness)
## Multiple R-squared:  0.8433, Adjusted R-squared:  0.8412
## F-statistic: 401.7 on 9 and 672 DF,  p-value: < 2.2e-16
```

1. **Use the mean/mode imputation method to impute values for the missing data.** *(with `mice`)*

```
# impute data with mice using 5 imputations and unconditional mean imputation
imp_m <- mice(data
              , m = 5
              , method = 'mean')
```

```
##
##  iter imp variable
##   1   1  Bare_Nuclei
##   1   2  Bare_Nuclei
##   1   3  Bare_Nuclei
##   1   4  Bare_Nuclei
##   1   5  Bare_Nuclei
##   2   1  Bare_Nuclei
##   2   2  Bare_Nuclei
##   2   3  Bare_Nuclei
##   2   4  Bare_Nuclei
##   2   5  Bare_Nuclei
##   3   1  Bare_Nuclei
##   3   2  Bare_Nuclei
##   3   3  Bare_Nuclei
##   3   4  Bare_Nuclei
##   3   5  Bare_Nuclei
##   4   1  Bare_Nuclei
##   4   2  Bare_Nuclei
##   4   3  Bare_Nuclei
##   4   4  Bare_Nuclei
##   4   5  Bare_Nuclei
##   5   1  Bare_Nuclei
##   5   2  Bare_Nuclei
##   5   3  Bare_Nuclei
##   5   4  Bare_Nuclei
##   5   5  Bare_Nuclei
```

```
# export imputed data
data_imp_mean <- complete(imp_m)

#data_imp_mean
```

```
# mean of each column
colMeans(data, na.rm = TRUE)
```

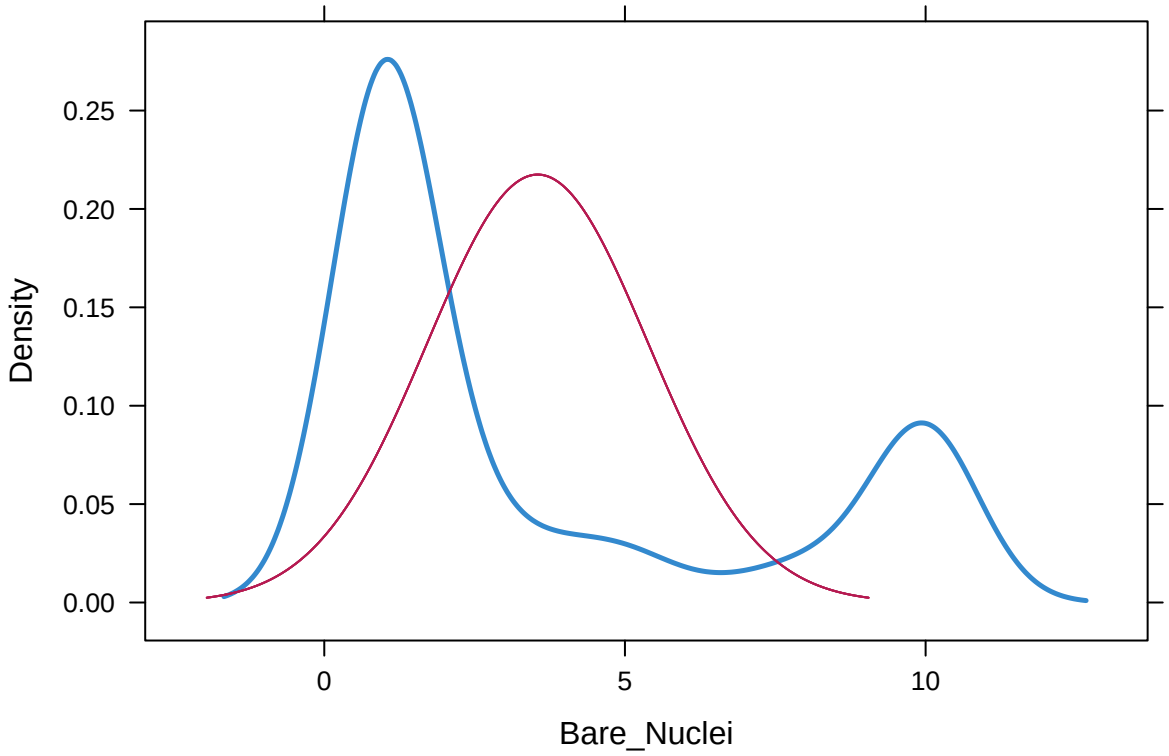
```
##        Clump_Thickness              Cell_Size             Cell_Shape
##              4.4169054              3.1375358              3.2106017
##      Marginal_Adhesion Single_Epith_Cell_Size            Bare_Nuclei
##              2.8094556              3.2177650              3.5483871
##        Bland_Chromatin        Normal_Nucleoli               Mitoses
##              3.4383954              2.8696275              1.5902579
##                  Class
##              0.3452722
```

We see a repetition of `3.548387` in the imputed data which is the mean of the `Bare_Nuclei` column.

```
# train the model
fit_m <- with(imp_m, lm(Class ~ Clump_Thickness+Cell_Size+Cell_Shape+Marginal_Adhesion+Single_Epith_Cel
# inspect regression model with imputed data
summary(fit_m)
```

```
## # A tibble: 50 x 6
##    term                   estimate std.error statistic  p.value  nobs
##    <chr>                     <dbl>     <dbl>     <dbl>    <dbl> <int>
##  1 (Intercept)            -0.252     0.0165    -15.3   8.63e-46   698
##  2 Clump_Thickness         0.0328    0.00360     9.13  7.59e-19   698
##  3 Cell_Size               0.0225    0.00644     3.50  4.97e- 4   698
##  4 Cell_Shape              0.0162    0.00629     2.58  1.02e- 2   698
##  5 Marginal_Adhesion       0.00628   0.00402     1.56  1.18e- 1   698
##  6 Single_Epith_Cell_Size  0.00788   0.00528     1.49  1.36e- 1   698
##  7 Bare_Nuclei             0.0456    0.00324    14.1   1.04e-39   698
##  8 Bland_Chromatin         0.0207    0.00509     4.07  5.29e- 5   698
##  9 Normal_Nucleoli         0.0172    0.00375     4.59  5.33e- 6   698
## 10 Mitoses                 0.00284   0.00503     0.563 5.73e- 1   698
## # ... with 40 more rows
```

```
# visual representation
densityplot(imp_m)
```



Looking at `Bare_Nuclei`, nothing much has changed from the baseline model since we're only just adding weight to the median —> bringing distribution closer to a normal distribution.

2. **Use regression to impute values for the missing data** *(with `mice`)*

```
# impute data with mice using 5 imputations and linear regression through prediction
imp_lm <- mice(data
               , m = 5
               , method = 'norm.predict')
```

```
##
##  iter imp variable
##   1   1  Bare_Nuclei
##   1   2  Bare_Nuclei
##   1   3  Bare_Nuclei
##   1   4  Bare_Nuclei
##   1   5  Bare_Nuclei
##   2   1  Bare_Nuclei
##   2   2  Bare_Nuclei
##   2   3  Bare_Nuclei
##   2   4  Bare_Nuclei
##   2   5  Bare_Nuclei
##   3   1  Bare_Nuclei
##   3   2  Bare_Nuclei
##   3   3  Bare_Nuclei
```

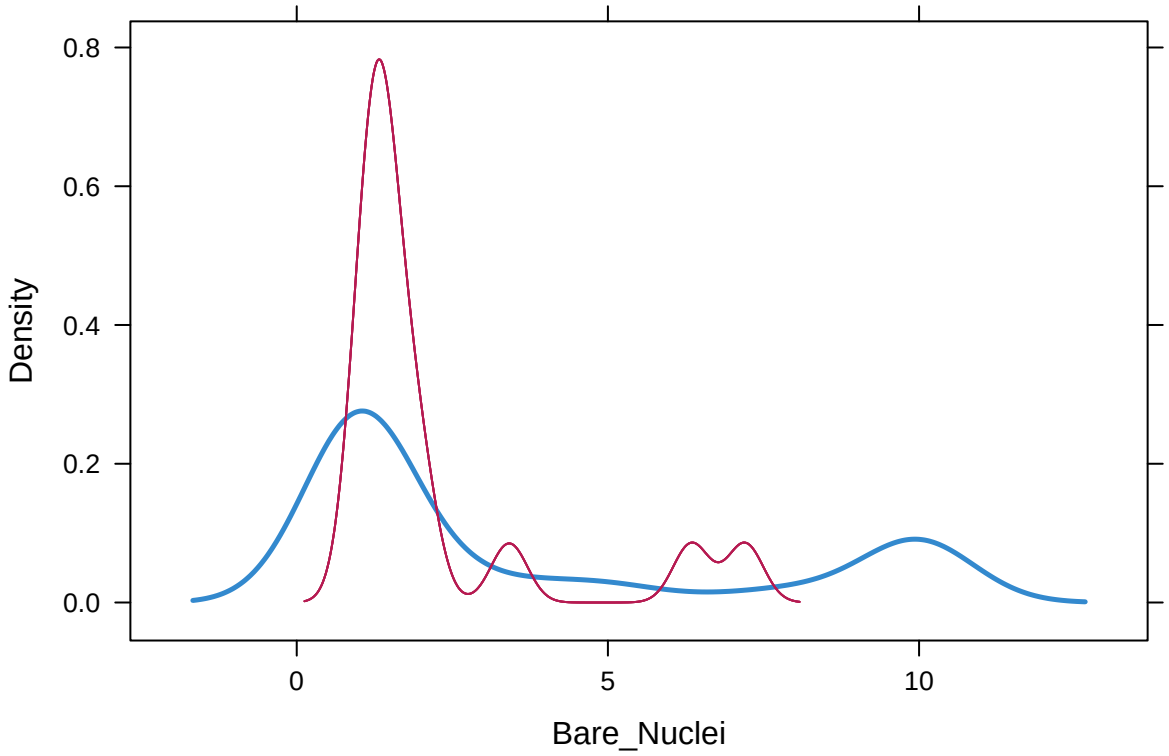
```
##   3   4  Bare_Nuclei
##   3   5  Bare_Nuclei
##   4   1  Bare_Nuclei
##   4   2  Bare_Nuclei
##   4   3  Bare_Nuclei
##   4   4  Bare_Nuclei
##   4   5  Bare_Nuclei
##   5   1  Bare_Nuclei
##   5   2  Bare_Nuclei
##   5   3  Bare_Nuclei
##   5   4  Bare_Nuclei
##   5   5  Bare_Nuclei
```

```
# export imputed data
data_imp_lm <- complete(imp_lm)

#data_imp_lm
```

```
# train the model
fit_lm <- with(imp_lm, lm(Class ~ Clump_Thickness+Cell_Size+Cell_Shape+Marginal_Adhesion+Single_Epith_C
# inspect regression model with imputed data
summary(fit_lm)
```

```
## # A tibble: 50 x 6
##    term                   estimate std.error statistic  p.value  nobs
##    <chr>                     <dbl>     <dbl>     <dbl>    <dbl> <int>
##  1 (Intercept)            -0.247     0.0164    -15.1   9.92e-45   698
##  2 Clump_Thickness         0.0318    0.00357     8.91  4.57e-18   698
##  3 Cell_Size               0.0224    0.00638     3.51  4.86e- 4   698
##  4 Cell_Shape              0.0158    0.00624     2.54  1.14e- 2   698
##  5 Marginal_Adhesion       0.00600   0.00398     1.51  1.32e- 1   698
##  6 Single_Epith_Cell_Size  0.00813   0.00523     1.55  1.21e- 1   698
##  7 Bare_Nuclei             0.0472    0.00323    14.6   2.60e-42   698
##  8 Bland_Chromatin         0.0198    0.00506     3.91  1.03e- 4   698
##  9 Normal_Nucleoli         0.0171    0.00372     4.60  5.05e- 6   698
## 10 Mitoses                 0.00297   0.00499     0.594 5.52e- 1   698
## # ... with 40 more rows
```

```
# visual representation
densityplot(imp_lm)
```



In this case, we can see how distribution of the `Bare_Nuclei` variable is no longer a normal distribution but it's still not close to the observed (blue).

3. **Use regression with perturbation to impute values for the missing data.** *(with `mice`)*

```
# impute data with mice using 5 imputations and regression with perturb
imp_lm_p <- mice(data
                 , m = 5
                 , method = 'norm.nob')
```

```
##
##  iter imp variable
##   1   1  Bare_Nuclei
##   1   2  Bare_Nuclei
##   1   3  Bare_Nuclei
##   1   4  Bare_Nuclei
##   1   5  Bare_Nuclei
##   2   1  Bare_Nuclei
##   2   2  Bare_Nuclei
##   2   3  Bare_Nuclei
##   2   4  Bare_Nuclei
##   2   5  Bare_Nuclei
##   3   1  Bare_Nuclei
##   3   2  Bare_Nuclei
##   3   3  Bare_Nuclei
##   3   4  Bare_Nuclei
```

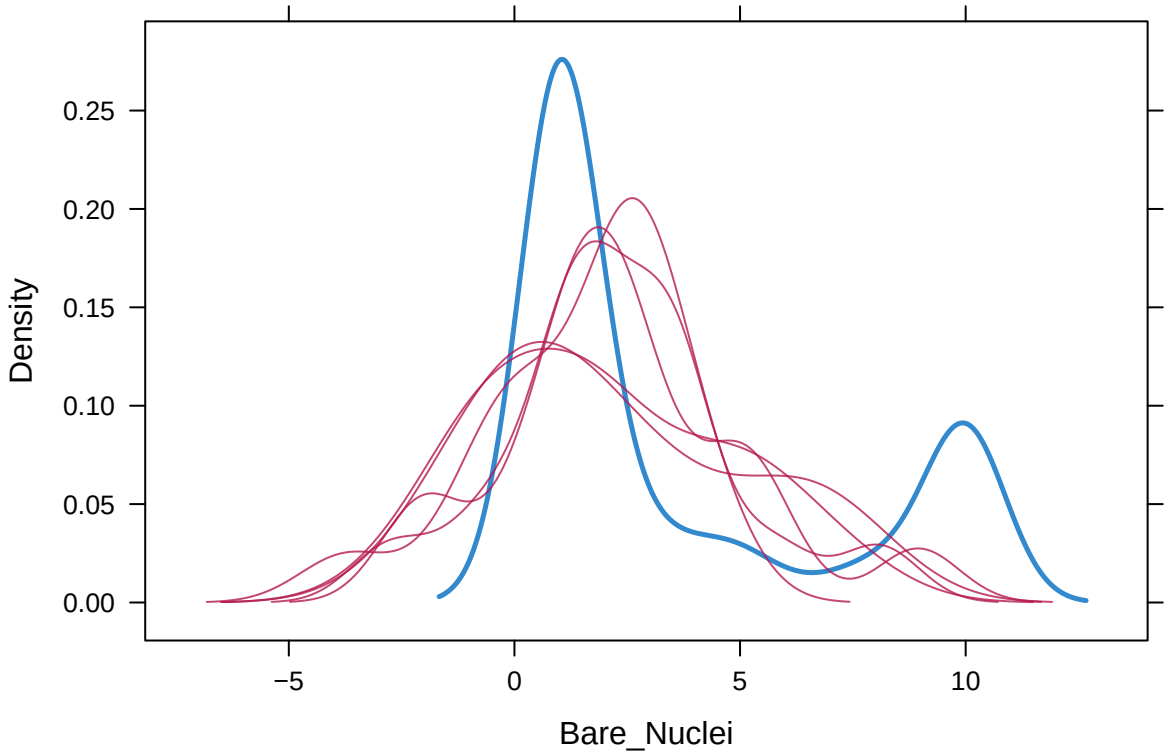
```
##   3   5  Bare_Nuclei
##   4   1  Bare_Nuclei
##   4   2  Bare_Nuclei
##   4   3  Bare_Nuclei
##   4   4  Bare_Nuclei
##   4   5  Bare_Nuclei
##   5   1  Bare_Nuclei
##   5   2  Bare_Nuclei
##   5   3  Bare_Nuclei
##   5   4  Bare_Nuclei
##   5   5  Bare_Nuclei
```

```
# export imputed data
data_imp_lm_p <- complete(imp_lm_p)

#data_imp_lm_p
```

```
# train the data
fit_lm_p <- with(imp_lm_p, lm(Class ~ Clump_Thickness+Cell_Size+Cell_Shape+Marginal_Adhesion+Single_Epi
# inspect regression model with imputed data
summary(fit_lm_p)
```

```
## # A tibble: 50 x 6
##    term                   estimate std.error statistic  p.value  nobs
##    <chr>                     <dbl>     <dbl>     <dbl>    <dbl> <int>
##  1 (Intercept)            -0.248     0.0164    -15.1   6.07e-45   698
##  2 Clump_Thickness         0.0321    0.00358     8.96  3.08e-18   698
##  3 Cell_Size               0.0222    0.00640     3.47  5.54e- 4   698
##  4 Cell_Shape              0.0159    0.00625     2.54  1.15e- 2   698
##  5 Marginal_Adhesion       0.00631   0.00399     1.58  1.14e- 1   698
##  6 Single_Epith_Cell_Size  0.00838   0.00524     1.60  1.10e- 1   698
##  7 Bare_Nuclei             0.0465    0.00322    14.5   1.43e-41   698
##  8 Bland_Chromatin         0.0202    0.00507     4.00  7.10e- 5   698
##  9 Normal_Nucleoli         0.0169    0.00372     4.54  6.55e- 6   698
## 10 Mitoses                 0.00297   0.00500     0.594 5.53e- 1   698
## # ... with 40 more rows
```

```
# visual representation
densityplot(imp_lm_p)
```



In this iteration, we can see the multiple imputations with different perturbations that shifted the distribution of the `Bare_Nuclei` variable.

4. **Dataset that remains after missing values are removed**

```
data_nona <- na.omit(data)

# train
fit_ <- with(data_nona, lm(Class ~ Clump_Thickness+Cell_Size+Cell_Shape+Marginal_Adhesion+Single_Epith_

# inspect model
summary(fit_)
```

```
##
## Call:
## lm(formula = Class ~ Clump_Thickness + Cell_Size + Cell_Shape +
##     Marginal_Adhesion + Single_Epith_Cell_Size + Bare_Nuclei +
##     Bland_Chromatin + Normal_Nucleoli + Mitoses)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.83997 -0.08295 -0.01225  0.05679  0.76358
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)            -0.2476295  0.0164330 -15.069  < 2e-16 ***
## Clump_Thickness         0.0317829  0.0035685   8.907  < 2e-16 ***
## Cell_Size               0.0218135  0.0063766   3.421 0.000662 ***
## Cell_Shape              0.0155989  0.0062402   2.500 0.012666 *
## Marginal_Adhesion       0.0082314  0.0039947   2.061 0.039728 *
## Single_Epith_Cell_Size  0.0100750  0.0052405   1.923 0.054959 .
## Bare_Nuclei             0.0453514  0.0032233  14.070  < 2e-16 ***
## Bland_Chromatin         0.0192476  0.0050488   3.812 0.000150 ***
## Normal_Nucleoli         0.0185094  0.0037221   4.973 8.39e-07 ***
## Mitoses                 0.0009794  0.0049697   0.197 0.843824
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1903 on 672 degrees of freedom
## Multiple R-squared:  0.8433, Adjusted R-squared:  0.8412
## F-statistic: 401.7 on 9 and 672 DF,  p-value: < 2.2e-16
```

5. **Dataset when a binary variable is introduced to indicate missing values**

```
# create new dummy variable
data_new <- mutate(data, Missing = ifelse(is.na(Bare_Nuclei), 1,0))

# train
fit_. <- with(data_new, lm(Class ~ Clump_Thickness+Cell_Size+Cell_Shape+Marginal_Adhesion+Single_Epith_

# inspect model
summary(fit_.)
```

```
##
## Call:
## lm(formula = Class ~ Clump_Thickness + Cell_Size + Cell_Shape +
##     Marginal_Adhesion + Single_Epith_Cell_Size + Bare_Nuclei +
##     Bland_Chromatin + Normal_Nucleoli + Mitoses)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.83997 -0.08295 -0.01225  0.05679  0.76358
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)            -0.2476295  0.0164330 -15.069  < 2e-16 ***
## Clump_Thickness         0.0317829  0.0035685   8.907  < 2e-16 ***
## Cell_Size               0.0218135  0.0063766   3.421 0.000662 ***
## Cell_Shape              0.0155989  0.0062402   2.500 0.012666 *
## Marginal_Adhesion       0.0082314  0.0039947   2.061 0.039728 *
## Single_Epith_Cell_Size  0.0100750  0.0052405   1.923 0.054959 .
## Bare_Nuclei             0.0453514  0.0032233  14.070  < 2e-16 ***
## Bland_Chromatin         0.0192476  0.0050488   3.812 0.000150 ***
## Normal_Nucleoli         0.0185094  0.0037221   4.973 8.39e-07 ***
## Mitoses                 0.0009794  0.0049697   0.197 0.843824
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.1903 on 672 degrees of freedom
##   (16 observations deleted due to missingness)
## Multiple R-squared:  0.8433, Adjusted R-squared:  0.8412
## F-statistic: 401.7 on 9 and 672 DF,  p-value: < 2.2e-16
```

6. **Compare the results and quality of classification models (e.g., SVM, KNN) build**

First, we prep the datasets.

```
data_imp_mean$Class <- as.factor(data_imp_mean$Class)
data_imp_lm$Class <- as.factor(data_imp_lm$Class)
data_imp_lm_p$Class <- as.factor(data_imp_lm_p$Class)
data_nona$Class <- as.factor(data_nona$Class)
data_new$Class <- as.factor(data_new$Class)

# split each dataset so that we use 70% of it for training
## 1 - data with mean imputation
one.index <- createDataPartition(y=data_imp_mean$Class, p=0.7, list=FALSE)
## 2 - data with reg imputation
two.index <- createDataPartition(y=data_imp_lm$Class, p=0.7, list=FALSE)
## 3 - data with reg with perturb imputation
three.index <- createDataPartition(y=data_imp_lm_p$Class, p=0.7, list=FALSE)
## 4 - data with nas removed
four.index <- createDataPartition(y=data_nona$Class, p=0.7, list=FALSE)
## 5 - data with additional variable
five.index <- createDataPartition(y=data_new$Class, p=0.7, list=FALSE)

# split the data
mean.train <- data_imp_mean[one.index, ]
mean.test <- data_imp_mean[-one.index, ]

reg.train <- data_imp_lm[two.index, ]
reg.test <- data_imp_lm[-two.index, ]

regper.train <- data_imp_lm_p[three.index, ]
regper.test <- data_imp_lm_p[-three.index, ]

nona.train <- data_nona[four.index, ]
nona.test <- data_nona[-four.index, ]


new.train <- data_new[five.index, ]
new.test <- data_new[-five.index, ]
```

Now we assess each model with Random Forest CV.

Model 1 - Mean Imputation

```
set.seed(123)

# define repeated cross validation with 5 folds and three repeats
repeat_cv <- trainControl(method='repeatedcv', number=5,repeats=3)
```

```r
# train a random forest model
forest1 <- train(
        Class ~ .,
        data=mean.train,
        method='rf',
        trControl=repeat_cv,
        preProcess = c("center","scale"),
        metric='Accuracy')

# print details about the model
forest1$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 3.07%
## Confusion matrix:
##     0   1 class.error
## 0 311   9  0.02812500
## 1   6 163  0.03550296
```

```r
# predict
y_hats <- predict(
        object=forest1,
        newdata=mean.test[, -10])

## print the accuracy
accuracy1 <- mean(y_hats == mean.test$Class)*100
cat('Accuracy on model with mean Imputation: ', round(accuracy1, 2), '%',  sep='')
```

```
## Accuracy on model with mean Imputation: 97.13%
```

Model 2 - Reg. Imputation

```r
set.seed(123)

# define repeated cross validation with 5 folds and three repeats
repeat_cv <- trainControl(method='repeatedcv', number=5,repeats=3)

# train a random forest model
forest2 <- train(
        Class ~ .,
        data=reg.train,
        method='rf',
        trControl=repeat_cv,
        preProcess = c("center","scale"),
        metric='Accuracy')
```

```
# print details about the model
forest2$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 9
##
##         OOB estimate of  error rate: 2.66%
## Confusion matrix:
##     0   1 class.error
## 0 312   8   0.0250000
## 1   5 164   0.0295858
```

```
# predict
y_hats <- predict(
        object=forest2,
        newdata=reg.test[, -10])

## print the accuracy
accuracy2 <- mean(y_hats == reg.test$Class)*100
```

Model 3 - Reg. with Perturbation Imputation

```
set.seed(123)

# define repeated cross validation with 5 folds and three repeats
repeat_cv <- trainControl(method='repeatedcv', number=5,repeats=3)

# train a random forest model
forest3 <- train(
        Class ~ .,
        data=regper.train,
        method='rf',
        trControl=repeat_cv,
        preProcess = c("center","scale"),
        metric='Accuracy')

# print details about the model
forest3$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 2.86%
## Confusion matrix:
```

```
##     0   1 class.error
## 0 311   9   0.0281250
## 1   5 164   0.0295858
```

```
# predict
y_hats <- predict(
        object=forest3,
        newdata=regper.test[, -10])

## print the accuracy
accuracy3 <- mean(y_hats == regper.test$Class)*100
```

Model 4 - Removed missing data model

```
set.seed(123)

# define repeated cross validation with 5 folds and three repeats
repeat_cv <- trainControl(method='repeatedcv', number=5,repeats=3)

# train a random forest model
forest4 <- train(
        Class ~ .,
        data=nona.train,
        method='rf',
        trControl=repeat_cv,
        preProcess = c("center","scale"),
        metric='Accuracy')

# print details about the model
forest4$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 2.71%
## Confusion matrix:
##     0   1 class.error
## 0 303   8  0.02572347
## 1   5 163  0.02976190
```

```
# predict
y_hats <- predict(
        object=forest4,
        newdata=nona.test[, -10])

## print the accuracy
accuracy4 <- mean(y_hats == nona.test$Class)*100
```

Model 5 - Added variable for missing data
Unavailable because this model can't handle missing data unfortunately.

So, how do the other models fare?

```
cat('Accuracy on model 1 data: ', round(accuracy1, 2), '%',  sep='', '\n')
```

```
## Accuracy on model 1 data: 97.13%
```

```
cat('Accuracy on model 2 data: ', round(accuracy2, 2), '%',  sep='', '\n')
```

```
## Accuracy on model 2 data: 95.22%
```

```
cat('Accuracy on model 3 data: ', round(accuracy3, 2), '%',  sep='', '\n')
```

```
## Accuracy on model 3 data: 96.65%
```

```
cat('Accuracy on model 4 data: ', round(accuracy4, 2), '%',  sep='', '\n')
```

```
## Accuracy on model 4 data: 97.54%
```

We can see that model 2, utilising just regression as imputation performed relatively poorer than the other models. However, the accuracy on all 4 models are suspiciously high.

Qn 15.1

Describe a situation or problem from your job, everyday life, current events, etc., for which optimization would be appropriate. What data would you need?

I used to play Rollercoaster Tycoon obsessively. Optimization would be apt for deciding on how many food stands, toilets, concession stands, and rides I should place to optimise the level of happiness of the park visitors. Data I would possibly need in this situation:

- Hourly amenities usage data
- Weather data
- Hourly ride data
- Calendar data (holidays, semester holidays etc)
- Pricing data