Week 6 Homework

```
#Loading needed packages
library(tidyverse)
```

```
## Loading tidyverse: ggplot2
## Loading tidyverse: tibble
## Loading tidyverse: tidyr
## Loading tidyverse: readr
## Loading tidyverse: purrr
## Loading tidyverse: dplyr
```

```
## Conflicts with tidy packages ----------------------------------------------
```

```
## filter(): dplyr, stats
## lag():    dplyr, stats
```

```
library(Amelia)
```

```
## Loading required package: Rcpp
```

```
## ##
## ## Amelia II: Multiple Imputation
## ## (Version 1.7.4, built: 2015-12-05)
## ## Copyright (C) 2005-2017 James Honaker, Gary King and Matthew Blackwell
## ## Refer to http://gking.harvard.edu/amelia/ for more information
## ##
```

```
library(kernlab)
```

```
##
## Attaching package: 'kernlab'
```

```
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
# Loading data
cancerData <- read.csv("http://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsi
                       header = FALSE, na.strings = "?")
#Adding column names
colnames(cancerData) <- c("SampleNo",
                          "Thickness",
                          "SizeUniform",
                          "ShapeUniform",
                          "Adhesion",
                          "SE_CellSize",
                          "BareNuclei",
                          "BlandChromatin",
                          "NormalNucleoli",
                          "Mitoses",
                          "Class")
```

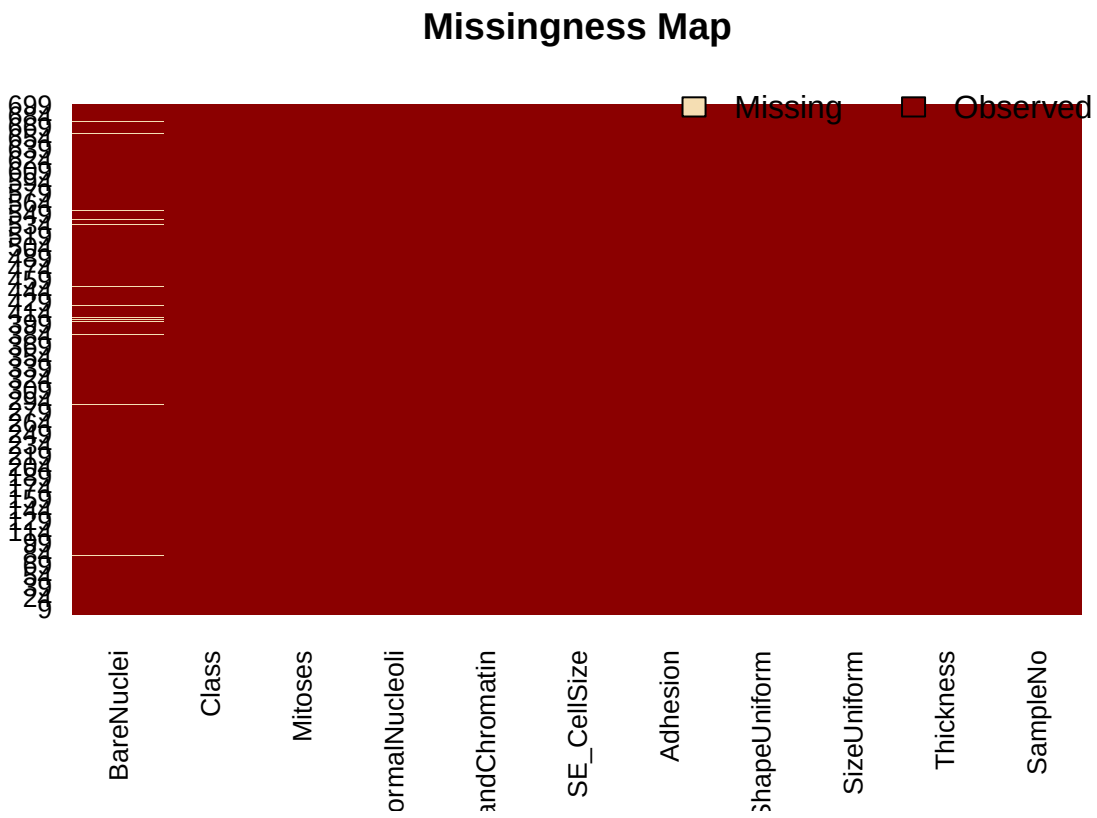
Question 2

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

```
#Creating 3 data sets to answers questions 1-3
mmData <- cancerData
regData <- cancerData
pertData <- cancerData

#visualizing missing data. Looks like there are only a few missing values for BareNuclei
Amelia::missmap(cancerData)
```




```
summary(cancerData$BareNuclei)
```

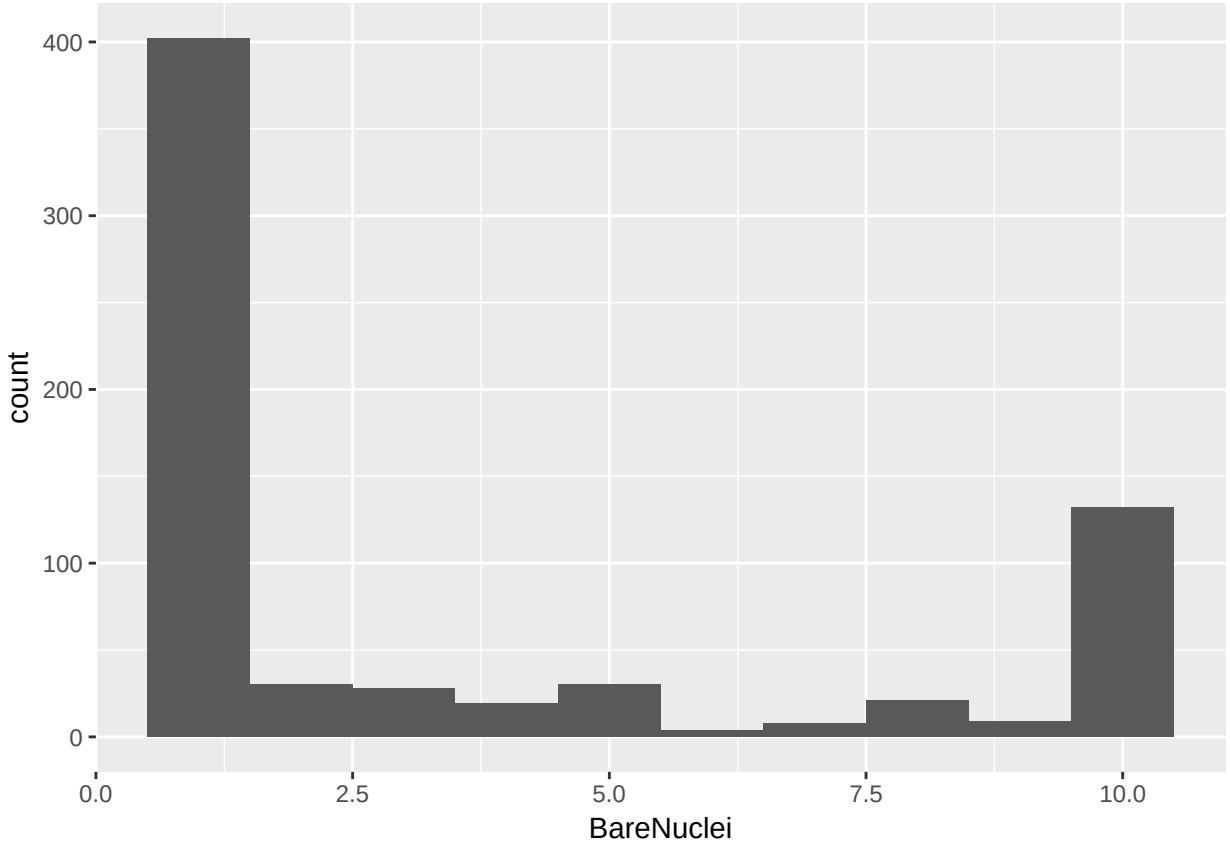
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   1.000   1.000   1.000   3.545   6.000  10.000      16
```

Question 2.1

Mean/Mode imputation - The assignement seems to make it optional on which method to choose. I chose the mode method as there is a large amount of 1's in the dataset which makes my chances of imputating an accurate value higher.

```
#Visualizing the distribution of values from column with missing values. Most datapoints
# seem to be either 1 or 10 with scattering in between. Based on this I am choosing the mode imputation
ggplot(mmData, aes(BareNuclei)) +
  geom_histogram(binwidth = 1)
```

```
## Warning: Removed 16 rows containing non-finite values (stat_bin).
```




```
#since we know only one of the columns has missing data I will focus on this alone for mean/mode imputa
mean(mmData$BareNuclei,na.rm = TRUE)
```

```
## [1] 3.544656
```

```
#Mean of vector: 3.54

#function for finding the mode, the number which appears most often
Mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}

Mode(mmData$BareNuclei)
```

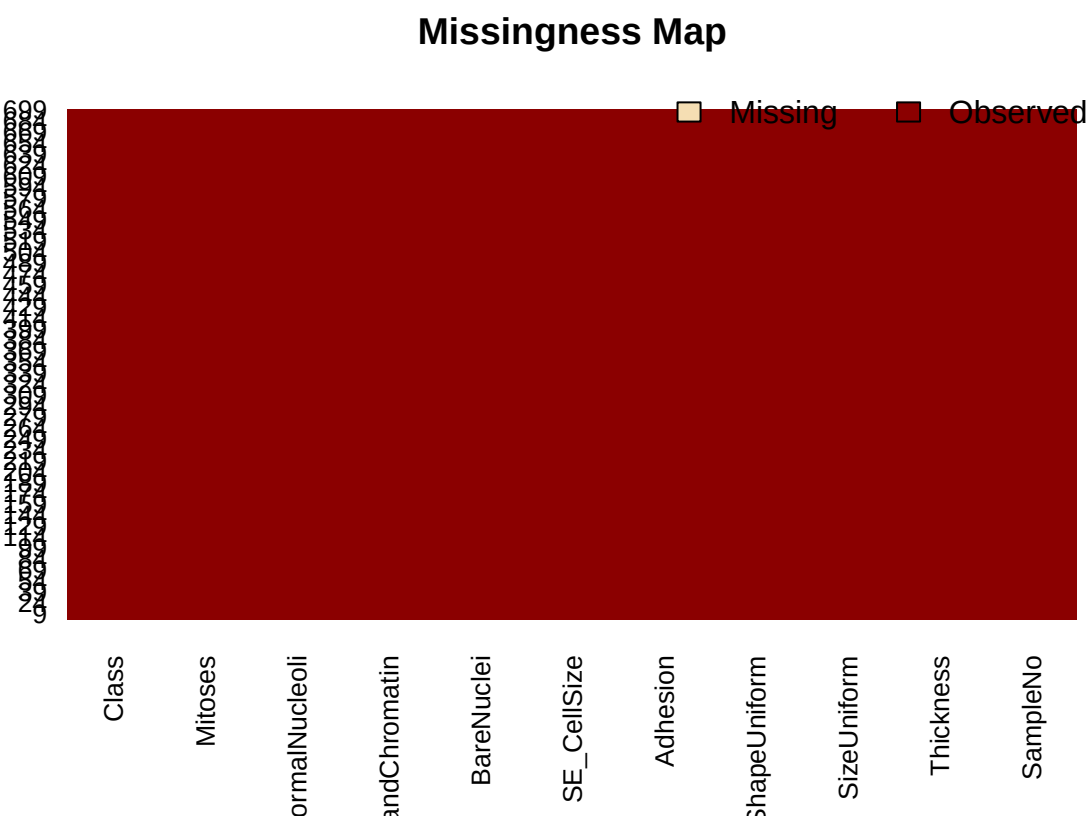
```
## [1] 1
```

```
#Mode of BareNuclei vector: 1

#imputing mising values with mean - I will not use this as my final result. Just demonstrating for home
# mmData$BareNuclei <- ifelse(is.na(mmData$BareNuclei),round(mean(mmData$BareNuclei,na.rm = TRUE),0),mm

#imputing missing values with mode - these are the results I will use.
mmData$BareNuclei <- ifelse(is.na(mmData$BareNuclei),
                            Mode(mmData$BareNuclei),mmData$BareNuclei)

#confirming results: No more missing values
Amelia::missmap(mmData)
```




```
summary(mmData$BareNuclei)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   1.000   3.486   5.000  10.000
```

```
#imputation completed, checking mean and mode with imputations, Mean is slightly lowered to 3.48 from 3
mean(mmData$BareNuclei)
```

```
## [1] 3.486409
```

```
Mode(mmData$BareNuclei)
```

```
## [1] 1
```

Question 2.2

Use regression to impute values for the missing data.

```
#selecting columns for new df which is used for the linear model. SampleNo is not important to results.
lmModData <- regData %>%
            dplyr::select(Thickness,SizeUniform,ShapeUniform,Adhesion,SE_CellSize,BareNuclei
                          ,BlandChromatin,NormalNucleoli,Mitoses,Class)

#building basic linear model
lmMod <- lm(BareNuclei ~ ., lmModData)
lmMod
```

```
##
## Call:
## lm(formula = BareNuclei ~ ., data = lmModData)
##
## Coefficients:
##    (Intercept)       Thickness     SizeUniform    ShapeUniform
##       -4.25273         0.01853        -0.16215         0.18437
##       Adhesion     SE_CellSize  BlandChromatin  NormalNucleoli
##        0.22093         0.01922         0.15128        -0.08738
##        Mitoses           Class
##       -0.06300         2.50988
```

```
summary(lmMod)
```

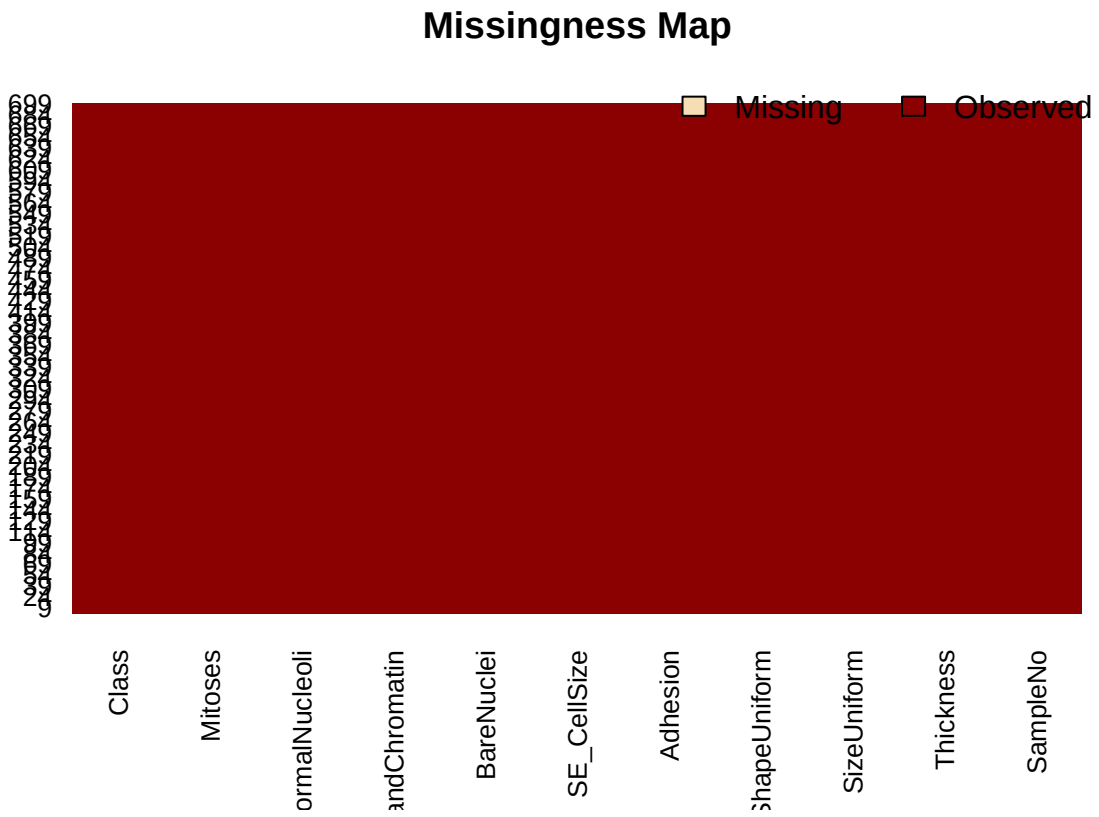
```
##
## Call:
## lm(formula = BareNuclei ~ ., data = lmModData)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.6030 -0.4262 -0.2194  0.8696  8.6294
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -4.25273    0.30981 -13.727  < 2e-16 ***
## Thickness       0.01853    0.03962   0.468  0.64019
## SizeUniform    -0.16215    0.06731  -2.409  0.01627 *
## ShapeUniform    0.18437    0.06551   2.815  0.00503 **
## Adhesion        0.22093    0.04125   5.356 1.17e-07 ***
## SE_CellSize     0.01922    0.05523   0.348  0.72790
## BlandChromatin  0.15128    0.05330   2.839  0.00467 **
## NormalNucleoli -0.08738    0.03969  -2.201  0.02804 *
## Mitoses        -0.06300    0.05218  -1.207  0.22770
## Class           2.50988    0.17811  14.091  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2 on 673 degrees of freedom
##   (16 observations deleted due to missingness)
## Multiple R-squared:  0.7027, Adjusted R-squared:  0.6987
## F-statistic: 176.7 on 9 and 673 DF,  p-value: < 2.2e-16
```

```
#generating rounded predictions for missing data points - rounding is required as all observations are
round(predict(lmMod, regData[is.na(regData$BareNuclei),]),0)
```

```
##  24  41 140 146 159 165 236 250 276 293 295 298 316 322 412 618
##   7   3   1   2   1   1   2   1   2   6   1   1   2   1   1   1
```

```
#final imputation using linear regression
regData$BareNuclei[is.na(regData$BareNuclei)] <- round(predict(lmMod, regData[is.na(regData$BareNuclei)

#confirming imputation was successful, with no major differences in summary stats
Amelia::missmap(regData)
```




```
summary(regData$BareNuclei)
```

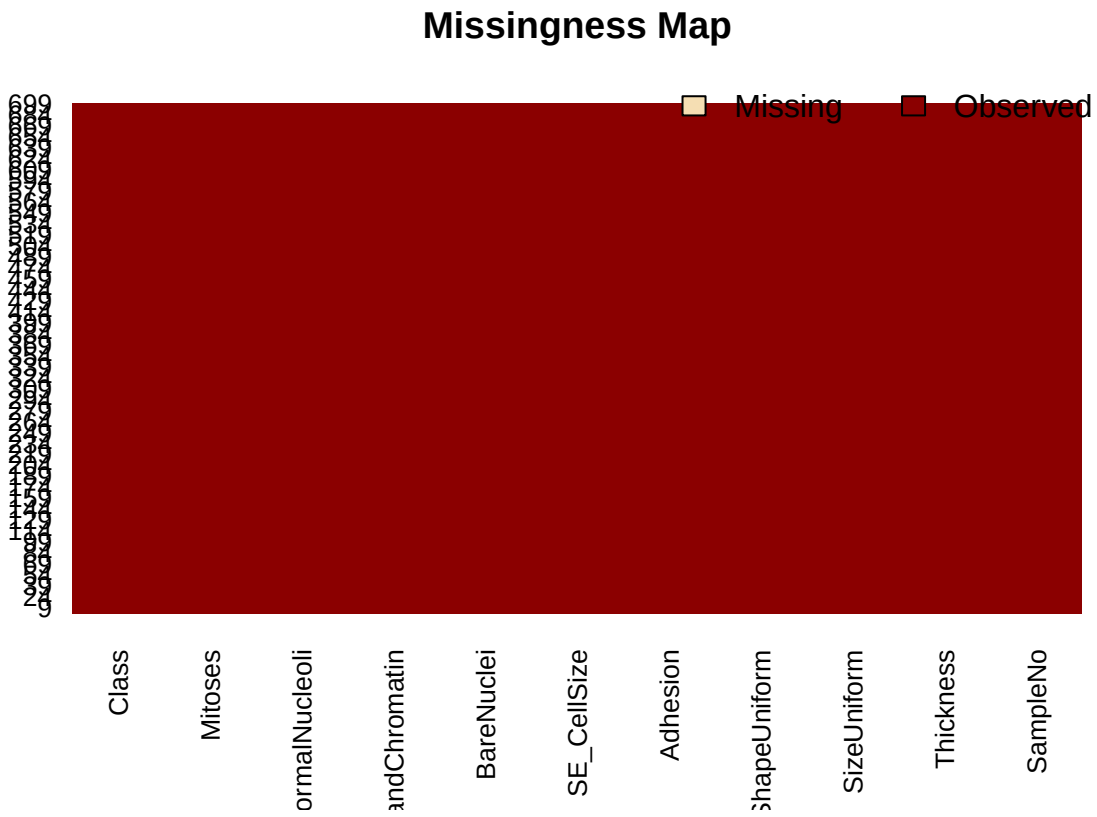
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   1.000   3.511   6.000  10.000
```

Question 2.3

Use regression with perturbation to impute values for the missing data. I used a new dataframe pertData but recycled the linear model I created in 2.2. Rather than just using the linear model prediction results I used the jitter function to add noise to the data. By default the jitter function uses a uniform distribution. I am not a fan or perturbing the data in this way as it creates data that is not similar to the rest of the dataset, non-whole numbers.

```
#using the jitter function to add noise to the predictions based on a uniform distribution of the vecto
pertData$BareNuclei[is.na(pertData$BareNuclei)] <- jitter(predict(lmMod
                                                                  , pertData[is.na(pertData$BareNuclei)

#visualizing the completeness of the dataset to confirm imputation was successful
Amelia::missmap(pertData)
```




```
#summary statistics show that mean is very close to overall original dataset
summary(pertData$BareNuclei)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.9997  1.0000  1.0000  3.5140  6.0000 10.0000
```

Question 2.4

(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

```
#building datasets to run model against
originalData <- cancerData
datasets <- list(mmData,pertData,regData)
names(datasets) <- c('mmData','pertData','regData')

#building SVM Model and Predictions for questions 1-3. Using basic linear kernal for comparisons
```

```
# the model results tell me that each model is identical, the imputations made little difference.
# The reason I think this is the case is due to only 12 values being missing in the dataset and the
# imputations are not really changing the summary statistics in any significant way.
ModelResults <- list()
PredResults <- list()
j = 1

#loop to create a model for the first 3 datasets and evaluate them
for (i in datasets){
  name <- names(datasets)[j]
  ModelResults[[name]] <- ksvm(as.matrix(i[,2:10]),as.matrix(i[,11]), type = "C-svc", kernal = "vanilla
                               , C=.1, scaled = TRUE, cross = 5, na.action = na.omit)
  PredResults[[name]] <- predict(ModelResults[[name]],i[,2:10])
  originalData[,name] <- PredResults[[name]]
  cat('SVM accuracy for:',name,sum(PredResults[[name]] == originalData[,11]) / nrow(originalData),'\n')
  j = j+1
}
```

```
## SVM accuracy for: mmData 0.9399142
## SVM accuracy for: pertData 0.9384835
## SVM accuracy for: regData 0.9399142
```

```
#Creating SVM model with simply ignoring the NA's
SVM.NAIgnore <- ksvm(as.matrix(cancerData[,2:10]), as.matrix(cancerData[,11]),
                     type = "C-svc", kernal = "vanilladot", C=.1, scaled = TRUE, cross = 5, na.action =
cancerData[complete.cases(cancerData),12] <- predict(SVM.NAIgnore,cancerData[complete.cases(cancerData)

#accuracy with just ignoring the NA is actually slightly higher than my imputed models
cat('SVM accuracy for SVM.NAIgnore:',sum(cancerData$V12 == cancerData$Class) /
      nrow(cancerData[complete.cases(cancerData),]),'\n')
```

```
## SVM accuracy for SVM.NAIgnore: NA
```

```
#SVM accuracy for SVM.NAIgnore: 0.9414348 - I have no idea why this is not showing in the PDF output.
#This is the result line 171 gives me. Please do not penalize me because knitr is not evaluating proper

#creating final dataset for comparison
flaggingNAData <- cancerData
flaggingNAData$V12 <- NULL
flaggingNAData$isNA <- ifelse(is.na(flaggingNAData$BareNuclei),1,0)

#now that NA rows are flagged will impute with regression model from previous
# question before placing into model to see how is.na column affects the model
flaggingNAData$BareNuclei <- regData$BareNuclei

#ensuring NA's were imputed
summary(flaggingNAData$BareNuclei)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   1.000   3.511   6.000  10.000
```

```
#building SVM model with new is.na feature
SVM.NAFlagged <- ksvm(as.matrix(flaggingNAData[,2:10,12]), as.matrix(flaggingNAData[,11]), type = "C-sv
                      , kernal = "vanilladot", C=.1, scaled = TRUE, cross = 5)
flaggingNAData$pred <- predict(SVM.NAFlagged,flaggingNAData[,2:10,12])
```

```r
#model accuracy - is nearly identical to the imputed models. This tells me that the new feature
# is not adding a lot of value. However I believe having a is.imputed column is very important
# for future analysis and maintaining a intuitive dataset.
cat('SVM accuracy:',sum(flaggingNAData$pred == flaggingNAData[,11]) / nrow(flaggingNAData),'\n')
```

```
## SVM accuracy: 0.9399142
```

Question 3

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

Answer: A problem from a previous job I had would be optimizing routing of customers to call center agents. I understand this is something that was mentioned in the videos however the use case which would be beneficial in this situation is not simply just routing customers in a timely manner. It would also be taking into account the customers problem they are calling in for as well as the skill level of each agent by problem subject. The idea would be to create a system will optimizes routing by customer need to the agent most likely to be able to solve their issue most effectively.

Data that would be needed is a metric to determine how each agent at the call center performs on specific customer issues. Another data point would be the rate in which customers call in at, duration of phone calls, and a tag which indicates the problem the customer is having. I think there is a lot more to this, however this would be a good starting point to start optimizing.