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.

The following is the setup I used for each part of this question:

```
set.seed(123)
library(tidyverse)

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.
0.0 ──
## ✓ dplyr     1.1.4     ✓ readr     2.1.5
## ✓ forcats   1.0.0     ✓ stringr   1.5.1
## ✓ ggplot2   3.5.1     ✓ tibble    3.2.1
## ✓ lubridate 1.9.3     ✓ tidyr     1.3.1
## ✓ purrr     1.0.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflict
s() ──
## ✕ dplyr::filter() masks stats::filter()
## ✕ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all
conflicts to become errors

library(caret)

## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
data <- read.table("breast-cancer-wisconsin.data.txt",
                  header = FALSE, sep = ",", stringsAsFactors = FALSE,
                  col.names = c("ID", "ClumpThickness", "CellSize",
                                "CellShape", "Adhesion", "EpithelialSize",
                                "BareNuclei", "BlandChromatin",
                                "NormalNucleoli", "Mitoses", "Class"))

# Convert 'BareNuclei' to numeric, replacing '?' with NA
data$BareNuclei <- as.numeric(replace(data$BareNuclei, data$BareNuclei == "?"
, NA))

# Drop the 'ID' column
data <- data %>% select(-ID)


# Check the dataset
summary(data)
##  ClumpThickness      CellSize         CellShape         Adhesion
##  Min.   : 1.000   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
##  1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 1.000   1st Qu.: 1.000
##  Median : 4.000   Median : 1.000   Median : 1.000   Median : 1.000
##  Mean   : 4.418   Mean   : 3.134   Mean   : 3.207   Mean   : 2.807
##  3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.: 5.000   3rd Qu.: 4.000
##  Max.   :10.000   Max.   :10.000   Max.   :10.000   Max.   :10.000
##
##  EpithelialSize     BareNuclei     BlandChromatin   NormalNucleoli
##  Min.   : 1.000   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
##  1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 2.000   1st Qu.: 1.000
##  Median : 2.000   Median : 1.000   Median : 3.000   Median : 1.000
##  Mean   : 3.216   Mean   : 3.545   Mean   : 3.438   Mean   : 2.867
##  3rd Qu.: 4.000   3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.: 4.000
##  Max.   :10.000   Max.   :10.000   Max.   :10.000   Max.   :10.000
##                   NA's   :16
##     Mitoses          Class
##  Min.   : 1.000   Min.   :2.00
##  1st Qu.: 1.000   1st Qu.:2.00
##  Median : 1.000   Median :2.00
##  Mean   : 1.589   Mean   :2.69
##  3rd Qu.: 1.000   3rd Qu.:4.00
##  Max.   :10.000   Max.   :4.00
##
```

To get the data ready to use the imputations in the questions, I saw that the data was all in just a single column separated by commas. I looked at the documentation given by the hyperlink in the question to get the variable information. Using the col.names function while loading in the data, I named each column and assigned each value to each column. I did this by using sep = "," (meaning the first value in the original data set would go into the first column name I specified, and when the first comma was seen, it would then put the next value into the second column I specified, etc.). Having a clearer data set, I looked at it. I knew that there were going to be missing values from the question, I just did not know exactly where yet. I saw that in the “BareNuclei” column there were question marks where the missing data was supposed to be. To work with these later, I needed to convert them to numeric factors, so I switched the question marks to the value NA (which in R is counted as a numeric). Knowing the ID column had nothing to do with any analysis, I just got rid of it to make things simpler. With all of this, I took a quick look at what the numbers looked like and was now ready to run the 3 imputations the question asked to run.

1. Mean/Mode Imputation Method

   Even though the only missing values occurred in the “BareNuclei” column of the data set, I decided to find the means of each column using the mutate function. Calling a summary of the data set (shown above) would do the same thing, but I wanted to see if any values would change (they did not). I made sure to only include numeric features when finding the means of each column. I then told the mutate function that if the value of a data point is NA, to make it the mean of that column. Below is the code for all of this:

```
# Mean imputation for numeric features (including BareNuclei)
data_mean_imputed <- data %>%
  mutate(across(everything(), ~ ifelse(is.na(.), mean(., na.rm = TRUE), .)))

# Check the imputed dataset
summary(data_mean_imputed)

##  ClumpThickness     CellSize         CellShape          Adhesion
##  Min.   : 1.000   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
##  1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 1.000   1st Qu.: 1.000
##  Median : 4.000   Median : 1.000   Median : 1.000   Median : 1.000
##  Mean   : 4.418   Mean   : 3.134   Mean   : 3.207   Mean   : 2.807
##  3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.: 5.000   3rd Qu.: 4.000
##  Max.   :10.000   Max.   :10.000   Max.   :10.000   Max.   :10.000
##  EpithelialSize    BareNuclei      BlandChromatin   NormalNucleoli
##  Min.   : 1.000   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
##  1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 2.000   1st Qu.: 1.000
##  Median : 2.000   Median : 1.000   Median : 3.000   Median : 1.000
##  Mean   : 3.216   Mean   : 3.545   Mean   : 3.438   Mean   : 2.867
##  3rd Qu.: 4.000   3rd Qu.: 5.000   3rd Qu.: 5.000   3rd Qu.: 4.000
##  Max.   :10.000   Max.   :10.000   Max.   :10.000   Max.   :10.000
##     Mitoses          Class
##  Min.   : 1.000   Min.   :2.00
##  1st Qu.: 1.000   1st Qu.:2.00
##  Median : 1.000   Median :2.00
##  Mean   : 1.589   Mean   :2.69  
##  3rd Qu.: 1.000   3rd Qu.:4.00  
##  Max.   :10.000   Max.   :4.00  
```

23	1
24	*NA*
25	1
23	1.000000
24	3.544656
25	1.000000

To put a visual to what happened, above is a sample of the data set's "BareNuclei" column before and after the mean/mode imputation. Before the imputation, row number 24 had a value of NA. Looking at the summary in the code (highlighted in yellow), it shows that the mean for the non-NA values in the "BareNuclei" column is 3.545 (rounded). What the mean/mode imputation does is take that mean value and assign it to each NA value in the column. So, each NA value in the "BareNuclei" column is now 3.544656.

2. Regression Imputation

I found this imputation to be much more difficult from the last one because there was more than just finding the mean and assigning the value. Below is the code for this imputation:

```r
# Linear regression model to predict 'BareNuclei'
lm_model <- lm(BareNuclei ~ ., data = data, na.action = na.exclude)

# Predict the missing values
predicted_values <- predict(lm_model, newdata = data[is.na(data$BareNuclei),
])

# Impute the missing values
data_reg_imputed <- data
data_reg_imputed$BareNuclei[is.na(data$BareNuclei)] <- predicted_values
```

```
# Check the imputed dataset
summary(data_reg_imputed)

##  ClumpThickness      CellSize         CellShape          Adhesion     
##  Min.   : 1.000   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000  
##  1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 1.000   1st Qu.: 1.000  
##  Median : 4.000   Median : 1.000   Median : 1.000   Median : 1.000  
##  Mean   : 4.418   Mean   : 3.134   Mean   : 3.207   Mean   : 2.807  
##  3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.: 5.000   3rd Qu.: 4.000  
##  Max.   :10.000   Max.   :10.000   Max.   :10.000   Max.   :10.000  
##  EpithelialSize     BareNuclei      BlandChromatin   NormalNucleoli  
##  Min.   : 1.000   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000  
##  1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 2.000   1st Qu.: 1.000  
##  Median : 2.000   Median : 1.000   Median : 3.000   Median : 1.000  
##  Mean   : 3.216   Mean   : 3.514   Mean   : 3.438   Mean   : 2.867  
##  3rd Qu.: 4.000   3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.: 4.000  
##  Max.   :10.000   Max.   :10.000   Max.   :10.000   Max.   :10.000  
##     Mitoses           Class     
##  Min.   : 1.000   Min.   :2.00  
##  1st Qu.: 1.000   1st Qu.:2.00  
##  Median : 1.000   Median :2.00  
##  Mean   : 1.589   Mean   :2.69  
##  3rd Qu.: 1.000   3rd Qu.:4.00  
##  Max.   :10.000   Max.   :4.00
```

I first created a linear regression model using the "BareNuclei" column as the response, all the other columns as the predictors, and I excluded all NA values from the model because we are supposed to be finding the values of them, not using them. Using the predict function, I told it to predict the NA values of "BareNuclei" using the linear regression model. The following is the predicted values for the respective rows:

24	7.201509
41	3.412194
140	1.200127
146	1.588095
159	1.271663
165	1.444743

236	1.960806
250	1.407689
276	1.625150
293	6.343076
295	1.219350
298	1.000995
316	2.005965
322	1.407689
412	1.200127
618	1.048844

I then told it to fill each of the NA values in the data set with the predicted values. So now, instead of the data set showing NA, it now shows the values above in the "BareNuclei" column in their respective rows. It is interesting to see that the mean for this column is still around 3.5 (highlighted in yellow).

3. Regression with Perturbation Imputation

   Below is the code for this imputation:

```
# Add noise to the predicted values
perturbation <- rnorm(length(predicted_values), mean = 0,
                      sd = sd(data$BareNuclei, na.rm = TRUE) * 0.1)

data_reg_perturbed <- data_reg_imputed
data_reg_perturbed$BareNuclei[is.na(data$BareNuclei)] <- predicted_values +
perturbation

# Check the imputed dataset
summary(data_reg_perturbed)
```

```
##  ClumpThickness      CellSize         CellShape         Adhesion     
##  Min.   : 1.000   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000  
##  1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 1.000   1st Qu.: 1.000  
##  Median : 4.000   Median : 1.000   Median : 1.000   Median : 1.000  
##  Mean   : 4.418   Mean   : 3.134   Mean   : 3.207   Mean   : 2.807  
##  3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.: 5.000   3rd Qu.: 4.000  
##  Max.   :10.000   Max.   :10.000   Max.   :10.000   Max.   :10.000  
##  EpithelialSize     BareNuclei      BlandChromatin   NormalNucleoli  
##  Min.   : 1.000   Min.   : 0.9467   Min.   : 1.000   Min.   : 1.000  
##  1st Qu.: 2.000   1st Qu.: 1.0000   1st Qu.: 2.000   1st Qu.: 1.000  
##  Median : 2.000   Median : 1.0000   Median : 3.000   Median : 1.000  
##  Mean   : 3.216   Mean   : 3.5162   Mean   : 3.438   Mean   : 2.867  
##  3rd Qu.: 4.000   3rd Qu.: 6.0000   3rd Qu.: 5.000   3rd Qu.: 4.000  
##  Max.   :10.000   Max.   :10.0000   Max.   :10.000   Max.   :10.000  
##     Mitoses           Class     
##  Min.   : 1.000   Min.   :2.00  
##  1st Qu.: 1.000   1st Qu.:2.00  
##  Median : 1.000   Median :2.00  
##  Mean   : 1.589   Mean   :2.69  
##  3rd Qu.: 1.000   3rd Qu.:4.00  
##  Max.   :10.000   Max.   :4.00
```

I used the regression imputation chart as the length of the rnorm function, and then filled in the rest of what I needed to get the perturbation of what the NA values will be. Using the data set with NA' s in it, I replaced the NA values with the values found by the linear regression imputation and added the perturbation values found for each row missing a value in the "BareNuclei" column as well. The perturbation values came out to be:

24	-0.20422932
41	-0.08387339
140	0.56797105
146	0.02569225
159	0.04711060
165	0.62494518
236	0.16795128
250	-0.46097024

276	-0.25027937
293	-0.16239286
295	0.44603792
298	0.13111102
316	0.14603539
322	0.04033120
412	-0.20254057
618	0.65112562

So, taking the values from the regression imputation and adding them to the numbers above from the same rows (Ex. For row 24: 7.201509 + (-0.20422932) = 6.997279) gives the values of each rows value for the "BareNuclei" column for this imputation. After doing all three tests, it was surprising to see that the mean values between all three of them really did not differ all that much (highlighted in yellow). I would be surprised to see if something like an outlier would make one of the imputations change much more than the others.

Using the SVM Model to check quality:

I thought it would be interesting to see the quality of the SVM model with removing the NA values of the original data set. I did not do all 3 of the different data sets to test on because I was kind of just interested to see what it would end up looking like if the NAs were just removed, so I just did the second part of the question. The code is as follows:

```
# Check for missing values
colSums(is.na(data))

## ClumpThickness       CellSize      CellShape       Adhesion EpithelialSize
##              0              0              0              0              0
##     BareNuclei BlandChromatin NormalNucleoli        Mitoses          Class
##             16              0              0              0              0
```

```r
# Remove rows with missing values
data_clean <- na.omit(data)

# Use the cleaned data for training
final_data <- data_clean

# Ensure the target variable 'Class' is a factor
final_data$Class <- factor(final_data$Class, levels = c(2, 4), labels = c("be
nign", "malignant"))

# Verify the structure of the data
str(final_data)
```

```
## 'data.frame':    683 obs. of  10 variables:
##  $ ClumpThickness: int  5 5 3 6 4 8 1 2 2 4 ...
##  $ CellSize      : int  1 4 1 8 1 10 1 1 1 2 ...
##  $ CellShape     : int  1 4 1 8 1 10 1 2 1 1 ...
##  $ Adhesion      : int  1 5 1 1 3 8 1 1 1 1 ...
##  $ EpithelialSize: int  2 7 2 3 2 7 2 2 2 2 ...
##  $ BareNuclei    : num  1 10 2 4 1 10 10 1 1 1 ...
##  $ BlandChromatin: int  3 3 3 3 3 9 3 3 1 2 ...
##  $ NormalNucleoli: int  1 2 1 7 1 7 1 1 1 1 ...
##  $ Mitoses       : int  1 1 1 1 1 1 1 1 5 1 ...
##  $ Class         : Factor w/ 2 levels "benign","malignant": 1 1 1 1 1 2 1
1 1 1 ...
##  - attr(*, "na.action")= 'omit' Named int [1:16] 24 41 140 146 159 165 236
250 276 293 ...
##   ..- attr(*, "names")= chr [1:16] "24" "41" "140" "146" ...
```

```r
train_index <- createDataPartition(final_data$Class, p = 0.7, list = FALSE)

train_data <- final_data[train_index, ]
test_data <- final_data[-train_index, ]

# Train the SVM model with cross-validation
svm_model <- train(Class ~ ., data = train_data, method = "svmLinear",
                   trControl = trainControl(method = "cv", number = 10))

# View the model details
print(svm_model)
```

```
## Support Vector Machines with Linear Kernel
##
## 479 samples
##   9 predictor
##   2 classes: 'benign', 'malignant'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 430, 431, 432, 431, 431, 431, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9708279  0.9360651
##
## Tuning parameter 'C' was held constant at a value of 1
```

```
# Predict on the test data
svm_predictions <- predict(svm_model, test_data)

# Evaluate the model performance
confusionMatrix(svm_predictions, test_data$Class)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction  benign malignant
##   benign       131         7
##   malignant      2        64
##
##                Accuracy : 0.9559
##                  95% CI : (0.9179, 0.9796)
##     No Information Rate : 0.652
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9012
##
##  Mcnemar's Test P-Value : 0.1824
##
##             Sensitivity : 0.9850
##             Specificity : 0.9014
##          Pos Pred Value : 0.9493
##          Neg Pred Value : 0.9697
##              Prevalence : 0.6520
##          Detection Rate : 0.6422
##    Detection Prevalence : 0.6765
##       Balanced Accuracy : 0.9432
##
##        'Positive' Class : benign
```

I removed all of the NA values using the na.omit function. As I had to do classification, I had to change the values of the “Class” column from numeric values to string values. The “Class” column contained values (2,4) which denoted benign or malignant. I made 2 = benign and 4 = malignant. I trained the data with the “Class” column as the predictor and all the other columns (still except for “ID” and now except for the rows with NA

“BareNuclei”). After running the SVM model on the testing and training data, I found that the model predicted a correct classification on a 95.59% accuracy which is super high. I do not think that the elimination of 10 out of almost 700 rows would be the cause of an accuracy score that high. I believe that with the other tests that the optional question says to run would have resulted in a high accuracy score as well.

Question 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?

As a financial advisor, optimization would be very beneficial in portfolio management to maximize client returns while minimizing risk of their portfolio. Some data I would need would be:

- The client’s preferences
  - Their risk tolerance – how much risk they are willing to take
  - Their investment goals – ex. how much money they want to retire with, if they are planning on buying anything in x amount of years, etc
- Asset information
  - Data on past returns for each investment option I would consider
  - Data on each investment options volatility to know its risk
- Market conditions
  - Current prices of assets
  - How any economic conditions would affect asset performances (ex. inflation rates)
- Any constraints
  - The amount the client has available to invest
  - Regulatory requirements affecting investment options
  - The client’s need for cash/liquid assets at any given time