PubH 6886 Fall 2025 Assignment One

Annie Allred

Question One

(50 pts.) The dataset bdiag.csv contains quantitative information from digitized images of a diagnostic test (fine needle aspirate (FNA) test on breast mass) for the diagnosis of breast cancer. The variables describe characteristics of the cell nuclei present in the image.

- `id`
- `diagnosis` (`M` = malignant, `B` = benign)

and ten real-valued features are computed for each cell nucleus:

- `radius` (mean of distances from center to points on the perimeter)
- `texture` (standard deviation of gray-scale values)
- `perimeter`
- `area`
- `smoothness` (local variation in radius lengths)
- `compactness` (perimeter^2 / area - 1.0)
- `concavity` (severity of concave portions of the contour)
- `concave points` (number of concave portions of the contour)
- `symmetry`
- `fractal dimension` ("coastline approximation" - 1)

The mean, standard error (se), and "worst" or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features. For instance, the third column is `radius_mean`, the thirteenth column is `radius_se`, and the twenty third column is `radius_worst`.

Part One

(a) (8 pts.) Conduct a PCA on the 30 features corresponding to mean, se, and worst for each measure. Do this for the original unscaled data. Suppose that we want to retain the smallest number of PCs that explain at least 75% of the variance in the data. How many PCs should be retained (provide justification)? Explain why the first PC accounts for so much of the variance in the data set. (Constructing a biplot of PC 1 vs. PC 2 might be useful but is not necessary.)

Based on the unscaled variables, only the first principal component should be retained, as it explains 98% of the variance in the data. The reason that the first PC explains so much of the data is because a) the first PC is created in such a way that it explains the most amount of variance in the data and b) the area variables are on a much larger scale than the other variables, so it has greater variance to reflect the greater scale.

```r
bdiag <- read.csv("bdiag.csv")
lesserBdiag <- bdiag %>%  # reading in the data
  select(-id, -diagnosis) # dropping the id and diagnosis variables

breastPCS_unscaled <- prcomp(lesserBdiag, scale = F) # unscaled PCs
# finds the amount of variance in the data each PC explains
```

(b) (5 pts.) Conduct a PCA on the 30 features, but this time, scale them to each have a variance of 1. Suppose that we want to retain the smallest number of PCs that explain at least 75% of the variance in the data. How many PCs should be retained (provide justification)?

Based on the scaled variables, the first four PCs should be retained as they explain 79% of the scaled variance in the data.

```r
breastPCS_scaled <- prcomp(lesserBdiag, scale = T) # scaled PCs
# finds the amount of variance in the data each PC explains
round(pcVar_Xscaled <-
        breastPCS_scaled$sdev^2/sum(breastPCS_scaled$sdev^2), 2)
```

```
##  [1] 0.44 0.19 0.09 0.07 0.05 0.04 0.02 0.02 0.01 0.01 0.01 0.01 0.01 0.01 0.00
## [16] 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
```

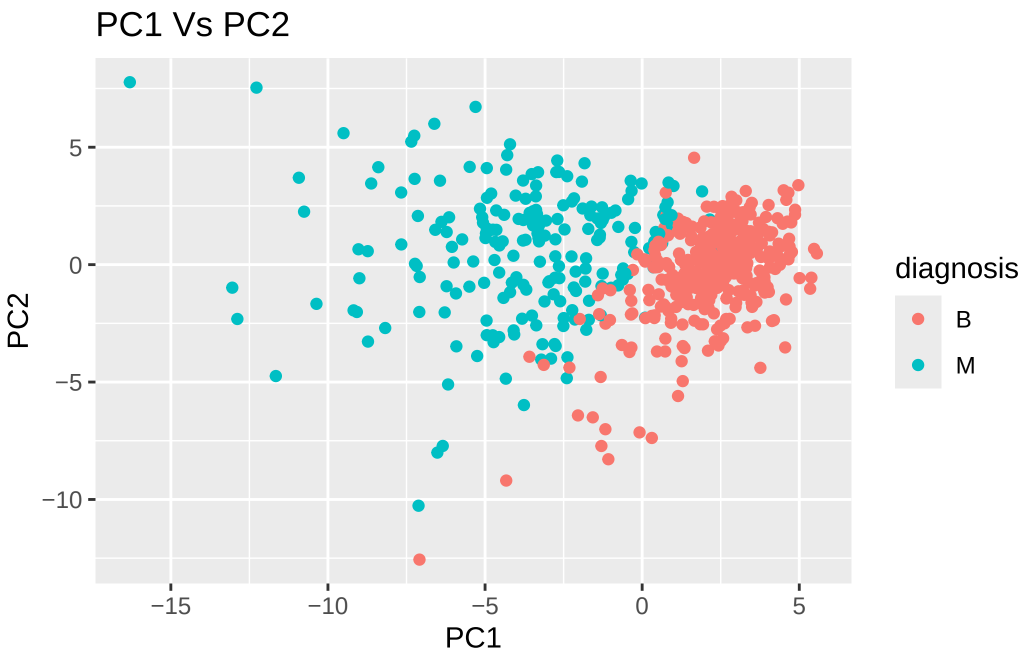
(c) (12 pts.) Using the first 3 PCs derived from the scaled data, is there one pair of these PCs that appears to be predictive of `diagnosis` (`M` = malignant, `B` = benign)? Make some scatterplots to justify your response. (You do not have to do any predictive modelling here, you are simply asked to select the pair and justify the selection based on graphical evidence.) Do your best to characterize/interpret the two PCs that appear to predict `diagnosis`.

The pair of PCs that best predict `diagnosis` are PC1 and PC2. This is shown as the scatter plot of the two points have the least amount of overlap between `M` and `B` points. We would expect PC1 and PC2 to be the PCs that best predict `diagnosis` as they are the PCs that explain the most amount of variance in the data and thus would have the greatest predicting power.

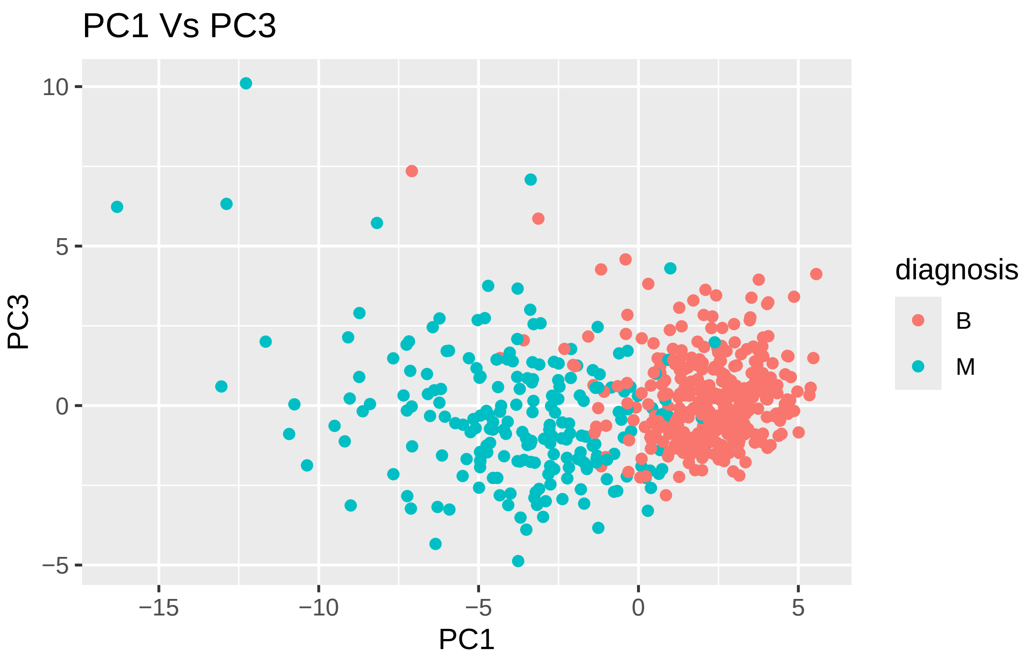
```r
# matrix of diagnosis and the first three PCs
diagMat <- cbind(bdiag[2], breastPCS_scaled$x[,1:3])
diagMat <- diagMat %>%
  mutate(diagF = factor(diagnosis,
                        levels = c("B", "M"),
                        labels = c(1,2)))

str(diagMat)
```

```
'data.frame':   569 obs. of  5 variables:
 $ diagnosis: chr  "M" "M" "M" "M" ...
 $ PC1      : num  -9.18 -2.39 -5.73 -7.12 -3.93 ...
 $ PC2      : num  -1.95 3.76 1.07 -10.27 1.95 ...
 $ PC3      : num  -1.122 -0.529 -0.551 -3.23 1.389 ...
 $ diagF    : Factor w/ 2 levels "1","2": 2 2 2 2 2 2 2 2 2 2 ...
```

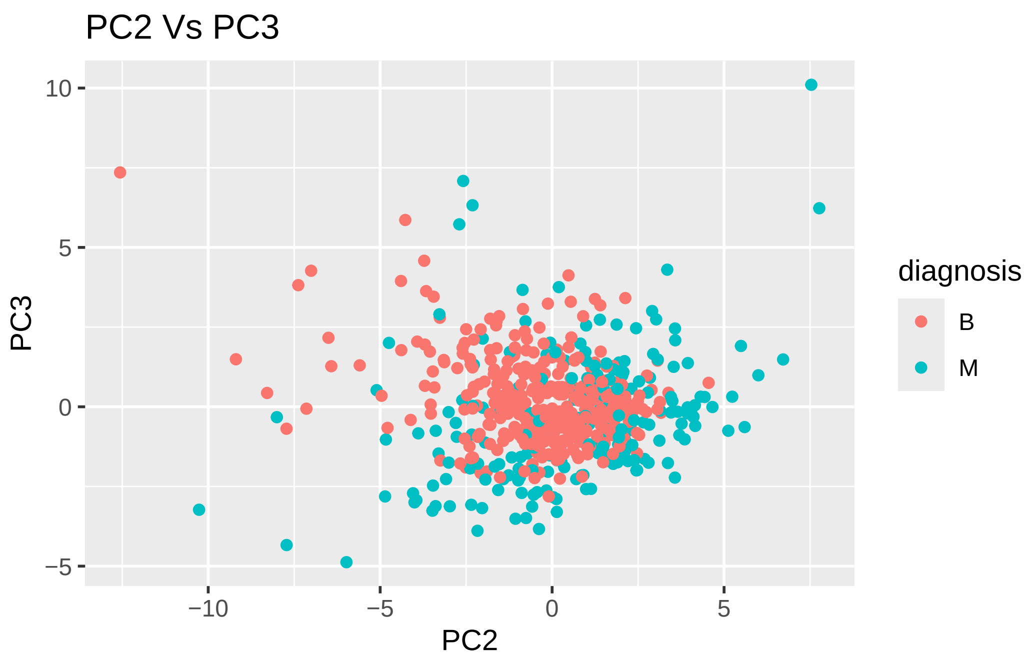
```r
# scatterplots  pairwise comparisons of the first three PCs
ggplot(diagMat, aes(PC1, PC2, colour = diagnosis)) +
  geom_point() +
  ggtitle("PC1 Vs PC2")
```



```r
ggplot(diagMat, aes(PC1, PC3, colour = diagnosis)) +
  geom_point() +
  ggtitle("PC1 Vs PC3")
```



```r
ggplot(diagMat, aes(PC2, PC3, colour = diagnosis)) +
  geom_point() +
  ggtitle("PC2 Vs PC3")
```



(d) (0 pts. but try it anyways) Provide the R code to compute the `x` object output by the `prcomp` function on the scaled data using (1) the scaled data object and (2) the `rotation` output by the `prcomp` function.

Part Two

(e) (14 pts.) Use K-means clustering with squared Euclidean distance to cluster the observations based on the 30 features corresponding to mean, se, and worst (scale them to each have variance 1). Consider $K = 2$ to 10 cluster solutions and select the best solution based on a plot of the total within cluster sum of squares vs. the number of clusters. Use 100 random starts and set the seed to 1 (`set.seed(1)`) prior to running the code for obtaining all 9 clusterings.

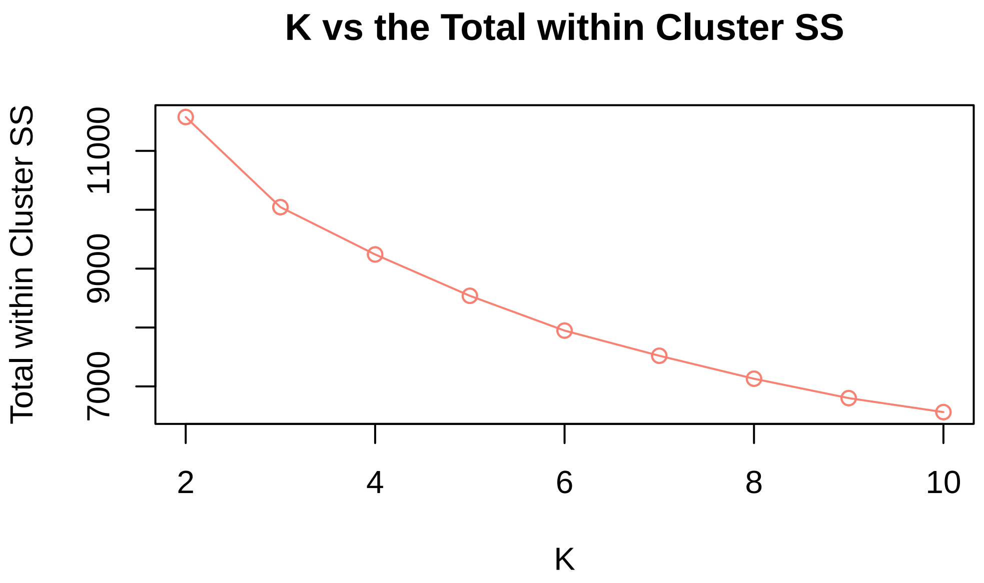
Based on the graph, the best amount of clusters to consider it $K = 3$, as that is about when the graph has a "strong" elbow.

```
set.seed(1) # sets the seed
scaledBdiag <- scale(lesserBdiag)# scales the data
# K-means clustering for K = 2 through K = 10
km_k2 <- kmeans(x = scaledBdiag, centers = 2, nstart = 100)
km_k3 <- kmeans(x = scaledBdiag, centers = 3, nstart = 100)
km_k4 <- kmeans(x = scaledBdiag, centers = 4, nstart = 100)
km_k5 <- kmeans(x = scaledBdiag, centers = 5, nstart = 100)
km_k6 <- kmeans(x = scaledBdiag, centers = 6, nstart = 100)
```

```r
km_k7 <- kmeans(x = scaledBdiag, centers = 7, nstart = 100)
km_k8 <- kmeans(x = scaledBdiag, centers = 8, nstart = 100)
km_k9 <- kmeans(x = scaledBdiag, centers = 9, nstart = 100)
```

```
Warning: did not converge in 10 iterations
```

```r
km_k10 <- kmeans(x = scaledBdiag, centers = 10, nstart = 100)

tot_withinVec <- c(km_k2$tot.withinss, km_k3$tot.withinss, km_k4$tot.withinss,
                   km_k5$tot.withinss, km_k6$tot.withinss, km_k7$tot.withinss,
                   km_k8$tot.withinss, km_k9$tot.withinss, km_k10$tot.withinss)
plot(2:10, tot_withinVec, col = "salmon",
     main = "K vs the Total within Cluster SS",
     xlab = "K", ylab = "Total within Cluster SS", type = "o")
points(2:10, tot_withinVec, col = "salmon")
```



(f) (3 pts.) Does the 2-cluster solution from part (e) correspond to the `diagnosis` (`M` = malignant, `B` = benign)? Justify your response.

Yes, the 2-cluster solution corresponds to the `diagnosis`. The outcome being predicted, `diagnosis`, by the response variables is dichotomous and the two clusters represent the two options. Like the in-class example, the 3-cluster best performed better than the other k-clusters when the outcome was multinominal with 3 levels. The 2-cluster solution sorted around 91.04% of the outcomes into the correct bin, so to speak, so it seems to be a good model.

```
# finds the proportion correctly predicted
mean(km_k2$cluster == diagMat$diagF)
```

```
[1] 0.9103691
```

(g) (8 pts.) Use agglomerative clustering based on Euclidean distance with complete linkage to cluster the observations based on the 30 features corresponding to mean, se, and worst (scale them to each have variance 1). Does the 2-cluster solution correspond to the `diagnosis` (`M` = malignant, `B` = benign)? Justify your response.

Yes, the 2-cluster solution corresponds to the `diagnosis`. The outcome being predicted, `diagnosis`, by the response variables is dichotomous and the two clusters represent the two options. The 2-cluster solution sorted around 63.09% of the outcomes into the correct bin, so to speak. This doesn't seem to be as good of a predictive model as the K-means clustering model.

```
# agglomerative cluster
bdiag_complete <- hclust(dist(scaledBdiag), method = "complete")
# finds the proportion correctly predicted
mean(cutree(bdiag_complete, k = 2) == diagMat$diagF)
```

```
[1] 0.6309315
```

Question Two

2. (50 pts.) In this problem, you will simulate some data and fit a set of linear regression models to assess their predictive performance. You will also construct a set of KNN predictions and compare them with the linear model predictions.

(a) (2 pts.) Using the `rnorm()` function, create a vector, `x`, containing 1100 observations drawn from a N(0,1) distribution. This represents a feature, X. Set the seed to 1 (`set.seed(1)`) prior to creating `x`.

```
set.seed(1)
x <- rnorm(1100, mean = 0, sd = 1)
```

(b) (2 pts.) Using the `rnorm()` function, create a vector, `eps`, containing 1100 observations drawn from a N(0,0.25) distribution—a normal distribution with mean zero and variance 0.25. This represents the error term ε. Set the seed to 2 (`set.seed(2)`) prior to creating `eps`.

```
set.seed(2)
eps <- rnorm(1100, mean = 0, sd = 0.5)
```

(c) (2 pts.) Using x and eps, generate a vector y according to the model: $Y = -1 + 0.5X + \varepsilon$.

```
y <- -1 + 0.5*x + eps
```

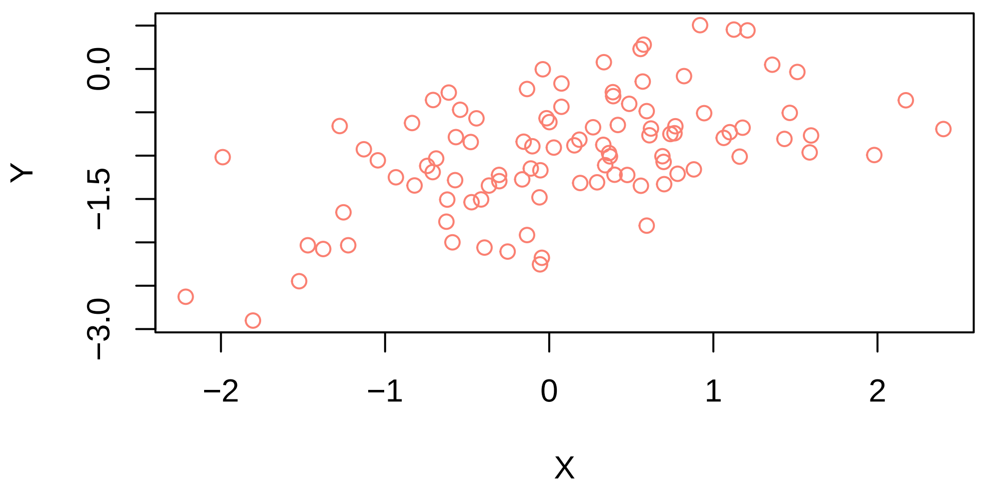
(d) (3 pts.) Create a training data set containing the first 100 x and y observations and a test data set containing the remaining 1000 x and y observations.

```
# creating the training data set
trainingData <- as.data.frame(cbind(c(x[1:100]), y[1:100]))
# naming the columns
colnames(trainingData) <- c("x", "y")
# creating the test data set
testData <- as.data.frame(cbind(c(x[101:1100]), y[101:1100]))
# naming the columns
colnames(testData) <- c("x", "y")
```

(e) (2 pts.) Create a scatterplot displaying the relationship between x and y in the training data set. Comment on what you observe.

The relationship looks fairly linear, which we would expect given it was created from a linear equation.

```
plot(trainingData, col = "salmon", xlab = "X", ylab = "Y")
```



(f) (4 pts.) Fit a least squares linear model to predict y using x. How do the estimated intercept and slope compare to the true values?

The intercept and slopes are extremely similar, -1.0023 compared to -1 and 0.45 compared to 0.5.

```r
# linear regression
summary(xy <- lm(y ~ x, data = trainingData))
```

```

Call:
lm(formula = y ~ x, data = trainingData)

Residuals:
     Min       1Q   Median       3Q      Max 
-1.22689 -0.40393 -0.04575  0.41574  1.14118 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -1.00454    0.05804 -17.308  < 2e-16 ***
x            0.40072    0.06446   6.216 1.25e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5761 on 98 degrees of freedom
Multiple R-squared:  0.2828,	Adjusted R-squared:  0.2755 
F-statistic: 38.64 on 1 and 98 DF,  p-value: 1.247e-08
```

(g) (5 pts.) Compute the training MSE (using the training set) and test MSE (using the test data set). Is it reasonable to expect the training MSE to be smaller than the test MSE? Justify your response.

You would expect the training MSE, 0.325 to be smaller than the test MSE, 0.265 as that is what we are trying to minimize. However, given that the test data is an entire factor of ten larger than the training dataset, the training dataset isn't large enough to be as robust as one might want, so the test MSE is smaller in this case.

```r
# predictions from the training data
xy_training_predict <- predict(xy )
mean((trainingData$y - xy_training_predict)^2) # training MSE
```

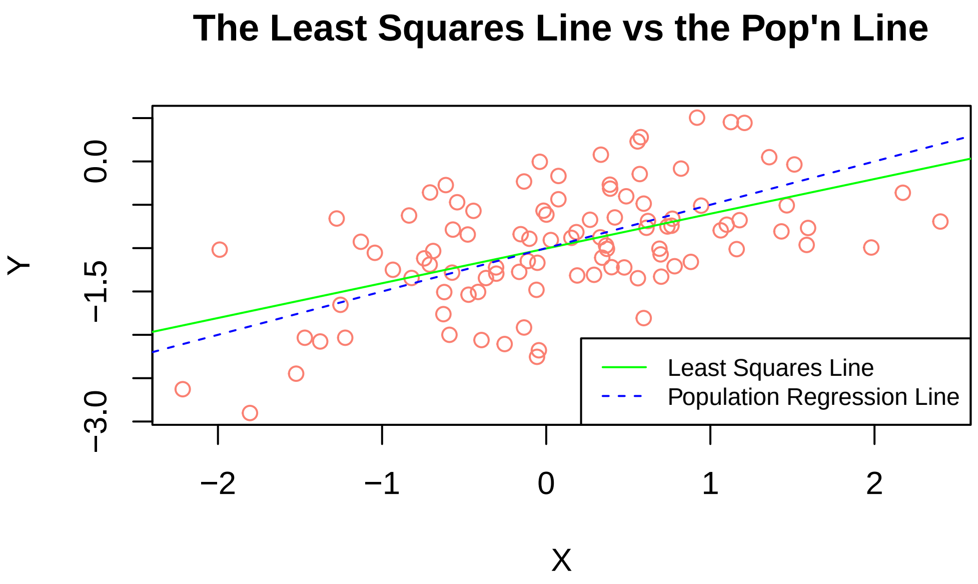
```
[1] 0.3252725
```

```r
# predictions from the testing data
xy_test_predict <- predict(xy, newdata = testData)
mean((testData$y - xy_test_predict)^2) # testing MSE
```

```
[1] 0.2652799
```

(h) (3 pts.) Display the least squares line on the scatterplot obtained in (e). Draw the population regression line on the plot, in a different color. Use the `legend()` command to create an appropriate legend.

```r
plot(trainingData, main = "The Least Squares Line vs the Pop'n Line",
     col = "salmon", xlab = "X", ylab = "Y")
abline(xy, col = "green")
abline(a = -1, b = 0.5, col = "blue", lty = 2)

legend("bottomright",
       legend = c("Least Squares Line", "Population Regression Line"),
       col = c("green", "blue"),
       lty = c(1:2),
       cex = 0.75)
```



(i) (7 pts.) Now fit a polynomial regression model that predicts y using x and the square of x (use `I(x^2)` in your code). Compute the training MSE (using the training set) and test MSE (using the test data set). Is it reasonable to expect the test MSE for the linear model to be smaller than the test MSE for the polynomial model? Justify your response.

The training MSE is 0.312 and the test MSE is 0.282

Yes, we would expect the test MSE for the linear model to be smaller than the test MSE for the polynomial model because the true relationship is linear. With the polynomial, we are adding unnecessary flexibility into the model at the cost of interpretability, which we don't care about in this situation, and bias.

```r
# polynomial regression
summary(x2y <- lm(y ~ x + I(x^2), data = trainingData))
```

```
Call:
lm(formula = y ~ x + I(x^2), data = trainingData)

Residuals:
     Min       1Q   Median       3Q      Max 
-1.30604 -0.38957 -0.06695  0.40921  1.13539 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -0.92420    0.06967 -13.265  < 2e-16 ***
x            0.41623    0.06394   6.509 3.33e-09 ***
I(x^2)      -0.10121    0.05020  -2.016   0.0465 *  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5673 on 97 degrees of freedom
Multiple R-squared:  0.3116,	Adjusted R-squared:  0.2974 
F-statistic: 21.96 on 2 and 97 DF,  p-value: 1.362e-08
```

```r
# predictions from the training data
x2y_training_predict <- predict(x2y)
mean((trainingData$y - x2y_training_predict)^2) # training MSE
```

```
[1] 0.312189
```

```r
# predictions from the testing data
x2y_test_predict <- predict(x2y, newdata = testData)
mean((testData$y - x2y_test_predict)^2) # testing MSE
```

```
[1] 0.2819879
```

(j) (8 pts.) Using the training data, fit KNN regressions of y on x for K = 1, 3, 5, and 10. Based on the test MSE, which value of K is optimal?

Based on the test MSE, having $K = 10$ is optimal as that has the minimum test MSE, 0.306 of the four trials.

```
# KNN for K = 1, 3, 5, and 10
knn_xy_k1 <- knnreg(y ~ x, k = 1, data = trainingData)
knn_xy_k3 <- knnreg(y ~ x, k = 3, data = trainingData)
knn_xy_k5 <- knnreg(y ~ x, k = 5, data = trainingData)
knn_xy_k10 <- knnreg(y ~ x, k = 10, data = trainingData)

# test predictions
knn_xy_k1_test <- predict(knn_xy_k1, newdata = testData)
knn_xy_k3_test <- predict(knn_xy_k3, newdata = testData)
knn_xy_k5_test <- predict(knn_xy_k5, newdata = testData)
knn_xy_k10_test <- predict(knn_xy_k10, newdata = testData)

# test MSEs
mean((testData$y - knn_xy_k1_test)^2)
```

```
[1] 0.5840194
```

```
mean((testData$y - knn_xy_k3_test)^2)
```

```
[1] 0.3781989
```

```
mean((testData$y - knn_xy_k5_test)^2)
```

```
[1] 0.3446176
```

```
mean((testData$y - knn_xy_k10_test)^2)
```

```
[1] 0.3063052
```

(k) (12 pts.) Using the `rnorm()` function repeatedly, create vectors `N1, N2, N3, N4, N5, N6, N7, N8, N9` and `N10`, each containing 1100 observations independently drawn from a N(0,1) distribution. Set the seed to 3 (`set.seed(3)`) prior to running the code to create all 10 vectors. These will serve as noise variables that are independent of the response, Y (and therefore are not predictive of Y). Append the first 100 observations from each of these noise variables to your training set that already contains `x` and `y`. Also append the next 1000 observations from each of these noise variables to your test set that already contains `x` and `y`. Fit a linear regression model on the new training data set with `y` regressed on `x` and the 10 noise variables simultaneously. Compute the test MSE for the fitted model. Compare this test MSE to the test MSE from part (g). Then fit a KNN regression on the new training data set with `y` regressed on `x` and the 10 noise variables simultaneously using the optimal K from part (j). Compute the test MSE for the fitted model. Compare this test MSE to the test MSE from part (j).

```r
set.seed(3) # setting seed
# creating the noise vectors
N1 <- rnorm(1100, mean = 0, sd = 1)
N2 <- rnorm(1100, mean = 0, sd = 1)
N3 <- rnorm(1100, mean = 0, sd = 1)
N4 <- rnorm(1100, mean = 0, sd = 1)
N5 <- rnorm(1100, mean = 0, sd = 1)
N6 <- rnorm(1100, mean = 0, sd = 1)
N7 <- rnorm(1100, mean = 0, sd = 1)
N8 <- rnorm(1100, mean = 0, sd = 1)
N9 <- rnorm(1100, mean = 0, sd = 1)
N10 <- rnorm(1100, mean = 0, sd = 1)

# creating the new training dataset
newTrainingData <- cbind(trainingData, N1[1:100], N2[1:100], N3[1:100],
                     N4[1:100], N5[1:100], N6[1:100], N7[1:100],
                     N8[1:100], N9[1:100], N10[1:100])
colnames(newTrainingData) <- c("x", "y", "N1", "N2", "N3", "N4", "N5", "N6",
                               "N7", "N8", "N9", "N10")

# creating the new test dataset
newTestData <- cbind(testData, N1[101:1100], N2[101:1100], N3[101:1100],
                     N4[101:1100], N5[101:1100], N6[101:1100], N7[101:1100],
                     N8[101:1100], N9[101:1100], N10[101:1100])
colnames(newTestData) <- c("x", "y", "N1", "N2", "N3", "N4", "N5", "N6",
                               "N7", "N8", "N9", "N10")
```

The test MSE for the new fitted linear model is 0.302, which is somewhat larger than the test MSE from part (g), 0.265. It's expected for the new test MSE to be larger than the original test MSE as the new noise variables add noise to the model, which then affects the test MSE.

```r
# the new lm
newXY <- lm(y ~ ., data = newTrainingData)

# predictions for the new testing data
newXY_test_predict <- predict(newXY, newdata = newTestData)
mean((newTestData$y - newXY_test_predict)^2) # testing MSE
```

```
[1] 0.3016816
```

The test MSE for the new KNN regression where $K = 10$ is 0.463, which is larger than the test MSE from part (j), 0.306. Similar to the fitted linear model, it is expected for the new test MSE for $K = 10$

to be larger than the original test MSE as the new noise variables add noise to the model, which then affects the test MSE.

```r
# KNN for K = 10
knn_newXY_k10 <- knnreg(y ~ ., k = 10, data = newTrainingData)
# test predictions
knn_newXY_k10_test <- predict(knn_newXY_k10, newdata = newTestData)
mean((newTestData$y - knn_newXY_k10_test)^2) # test MSE
```

```
[1] 0.4629107
```