Solution to Quiz 2

Question 1

Load the Alzheimer's disease data using the commands:

```
library(AppliedPredictiveModeling)
data(AlzheimerDisease)
```

A: Which of the following commands will create non-overlapping training and test sets with about 50% of the observations assigned to each?

Q:

```
adData = data.frame(diagnosis,predictors)
testIndex = createDataPartition(diagnosis, p = 0.50, list = FALSE)
training = adData[-testIndex,]
testing = adData[testIndex,]
```

Question 2

Load the cement data using the commands:

```
library(AppliedPredictiveModeling)
data(concrete)
library(caret)
set.seed(1000)
inTrain = createDataPartition(mixtures$CompressiveStrength, p = 3/4)[[1]]
training = mixtures[ inTrain,]
testing = mixtures[-inTrain,]
```

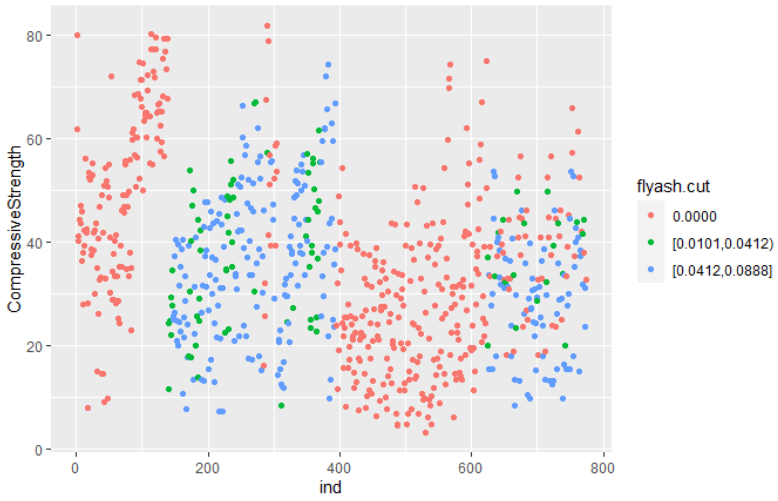
Q: Make a plot of the outcome `CompressiveStrength` versus the index of the samples. Color by each of the variables in the data set (you may find the `cut2()` function in the `Hmisc` package useful for turning continuous covariates into factors). What do you notice in these plots?

A: There is a non-random pattern in the plot of the outcome versus index that is perfectly explained by the `FlyAsh` variable.

Codes:

```
library(Hmisc)
flyash.cut <- cut2(training$FlyAsh, g = 3)

library(ggplot2)
df <- cbind(training, flyash.cut, ind = 1:dim(training)[1])
ggplot(data = df, aes(x = ind, y = CompressiveStrength, col = flyash.cut)) +
  geom_point()
```



Question 3

Load the cement data using the commands:

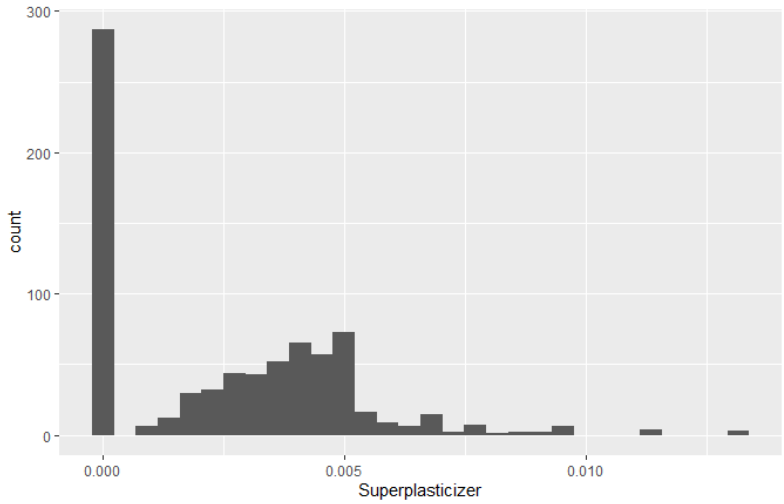
```
library(AppliedPredictiveModeling)
data(concrete)
library(caret)
set.seed(1000)
inTrain = createDataPartition(mixtures$CompressiveStrength, p = 3/4)[[1]]
training = mixtures[ inTrain,]
testing = mixtures[-inTrain,]
```

Q: Make a histogram and confirm the `SuperPlasticizer` variable is skewed. Normally you might use the log transform to try to make the data more symmetric. Why would that be a poor choice for this variable?

A: There are values of zero so when you take the `log()` transform those values will be -Inf.

Codes:

```
ggplot(data = training, aes(x = Superplasticizer)) +
  geom_histogram()
```



Question 4

Load the Alzheimer's disease data using the commands:

```
library(caret)
library(AppliedPredictiveModeling)
set.seed(3433)
data(AlzheimerDisease)
adData = data.frame(diagnosis,predictors)
inTrain = createDataPartition(adData$diagnosis, p = 3/4)[[1]]
training = adData[ inTrain,]
testing = adData[-inTrain,]
```

Q: Find all the predictor variables in the training set that begin with IL. Perform principal components on these variables with the `preProcess()` function from the `caret` package. Calculate the number of principal components needed to capture 90% of the variance. How many are there?

A: 9

Codes:

```
df <- training[, 58:69]
pca.md <- preProcess(df, method = "pca", thresh = 0.9)
pca.md$numComp
```

Question 5

Load the Alzheimer's disease data using the commands:

```
library(caret)
library(AppliedPredictiveModeling)
set.seed(3433)
data(AlzheimerDisease)
adData = data.frame(diagnosis,predictors)
inTrain = createDataPartition(adData$diagnosis, p = 3/4)[[1]]
training = adData[ inTrain,]
testing = adData[-inTrain,]
```

Q: Create a training data set consisting of only the predictors with variable names beginning with IL and the diagnosis. Build two predictive models, one using the predictors as they are and one using PCA with principal components explaining 80% of the variance in the predictors. Use `method="glm"` in the `train` function.

What is the accuracy of each method in the test set? Which is more accurate?

A: Non-PCA Accuracy: 0.75; PCA Accuracy: 0.71

Codes:

```
# for PCA
df.train <- training[, 58:69]
df.test <- testing[, 58:69]

preProc <- preProcess(df.train, method = "pca", thresh = 0.8)
```

```r
preProc.train <- predict(preProc, df.train)
preProc.train <- cbind(preProc.train, diagnosis = training$diagnosis)
preProc.test <- predict(preProc, df.test)
preProc.test <- cbind(preProc.test, diagnosis = testing$diagnosis)

model1 <- train(diagnosis ~., method = "glm", data = preProc.train)
model1.fit <- predict(model1, preProc.test)

confusionMatrix(preProc.test$diagnosis, model1.fit)

# for non-PCA
df.train <- training[, 58:69]
df.train <- cbind(df.train, diagnosis = training$diagnosis)
df.test <- testing[, 58:69]
df.test <- cbind(df.test, diagnosis = testing$diagnosis)

model2 <- train(diagnosis ~., method = "glm", data = df.train)
model2.fit <- predict(model2, df.test)
confusionMatrix(df.test$diagnosis, model2.fit)
```