Week 2

Question 1

Load the Alzheimer's disease data using the commands:

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

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

- Option 1

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

- Option 2

```
adData = data.frame(diagnosis,predictors)
trainIndex = createDataPartition(diagnosis,p=0.5,list=FALSE)
training = adData[trainIndex,]
testing = adData[trainIndex,]
```

- Option 3

```
adData = data.frame(diagnosis,predictors)
train = createDataPartition(diagnosis, p = 0.50,list=FALSE)
test = createDataPartition(diagnosis, p = 0.50,list=FALSE)
```

- Option 4

```
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(975)
inTrain = createDataPartition(mixtures$CompressiveStrength, p = 3/4)[[1]]
training = mixtures[ inTrain,]
testing = mixtures[-inTrain,]
```

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?

(i) The data show a step like pattern that is perfectly explained by the Age variable so there may be a variable missing.

(ii) The data show a step like pattern that is perfectly explained by the Age variable.

(iii) The data show a step like pattern that is perfectly explained by the FlyAsh variable so there may be a variable missing.

(iv) There is a step-like pattern in the plot of outcome versus index in the training set that isn't explained by any of the predictor variables so there may be a variable missing.

Question 3

Load the cement data using the commands:

```
library(AppliedPredictiveModeling)
data(concrete)
library(caret)
set.seed(975)
inTrain = createDataPartition(mixtures$CompressiveStrength, p = 3/4)[[1]]
training = mixtures[ inTrain,]
testing = mixtures[-inTrain,]
```

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?

- The log transform is not a monotone transformation of the data.
- The SuperPlasticizer data include negative values so the log transform can not be performed.
- There are a large number of values that are the same and even if you took the log(SuperPlasticizer + 1) they would still all be identical so the distribution would not be symmetric.
- The log transform produces negative values which can not be used by some classifiers.

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,]
```

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?

- 8
- 11
- 7
- 9

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,]
```

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?

(i) Non-PCA Accuracy: 0.65 PCA Accuracy: 0.72

(ii) Non-PCA Accuracy: 0.72 PCA Accuracy: 0.65

(iii) Non-PCA Accuracy: 0.91 PCA Accuracy: 0.93

(iv) Non-PCA Accuracy: 0.75 PCA Accuracy: 0.71