Quiz2

Xi Fang

6/25/2020

Q1

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

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

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

Q2

- Load the cement data
- 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?

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

library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```r
library(Hmisc)
```

```
## Loading required package: survival
```

```
##
## Attaching package: 'survival'
```

```
## The following object is masked from 'package:caret':
##
##     cluster
```

```
## Loading required package: Formula
```

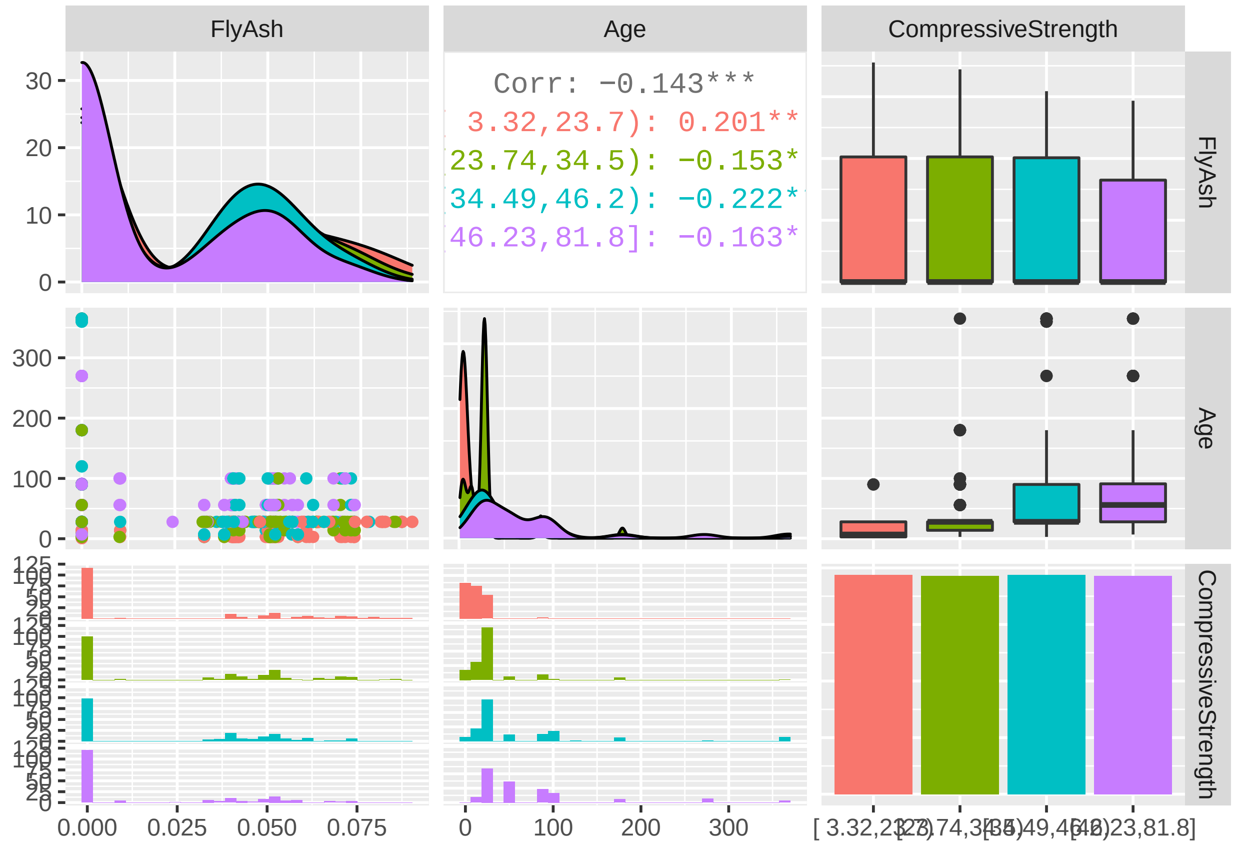
```
##
## Attaching package: 'Hmisc'
```

```
## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```r
t2 <- training
t2$CompressiveStrength <-cut2(t2$CompressiveStrength, g=4)
ggpairs(data=t2, columns = c("FlyAsh","Age","CompressiveStrength"),
        mapping = ggplot2::aes(colour=CompressiveStrength))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

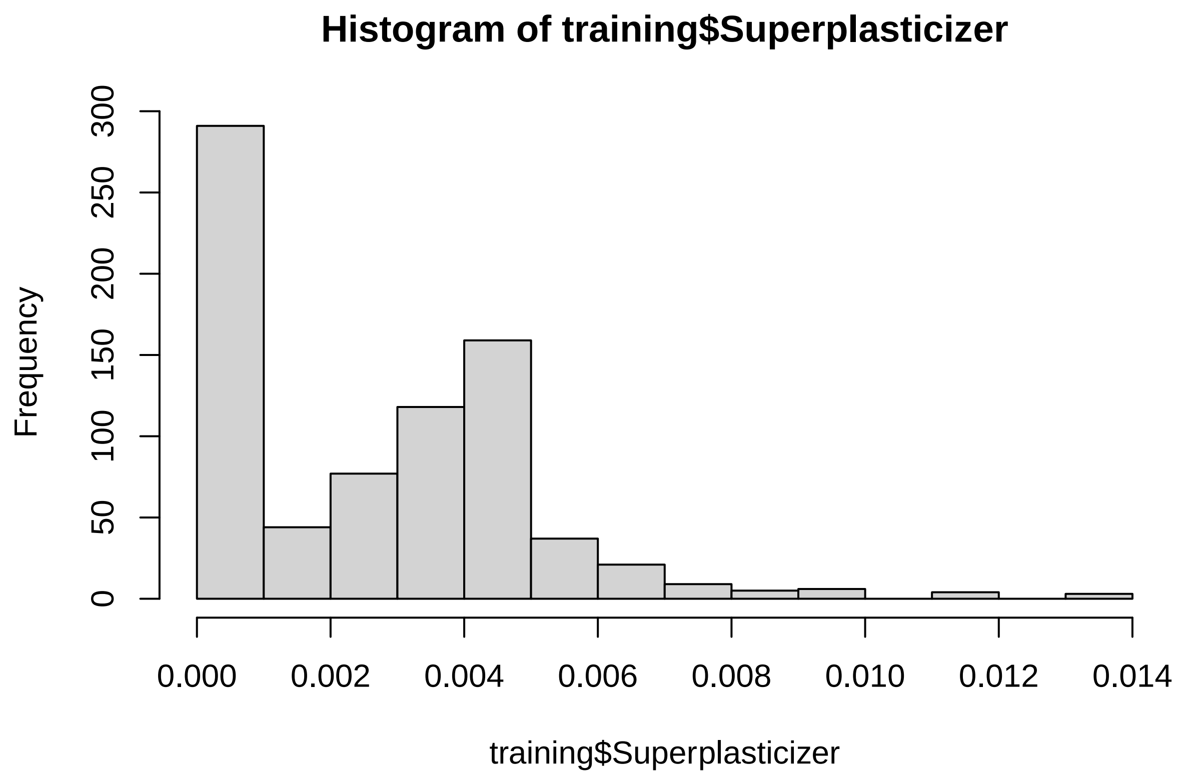
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



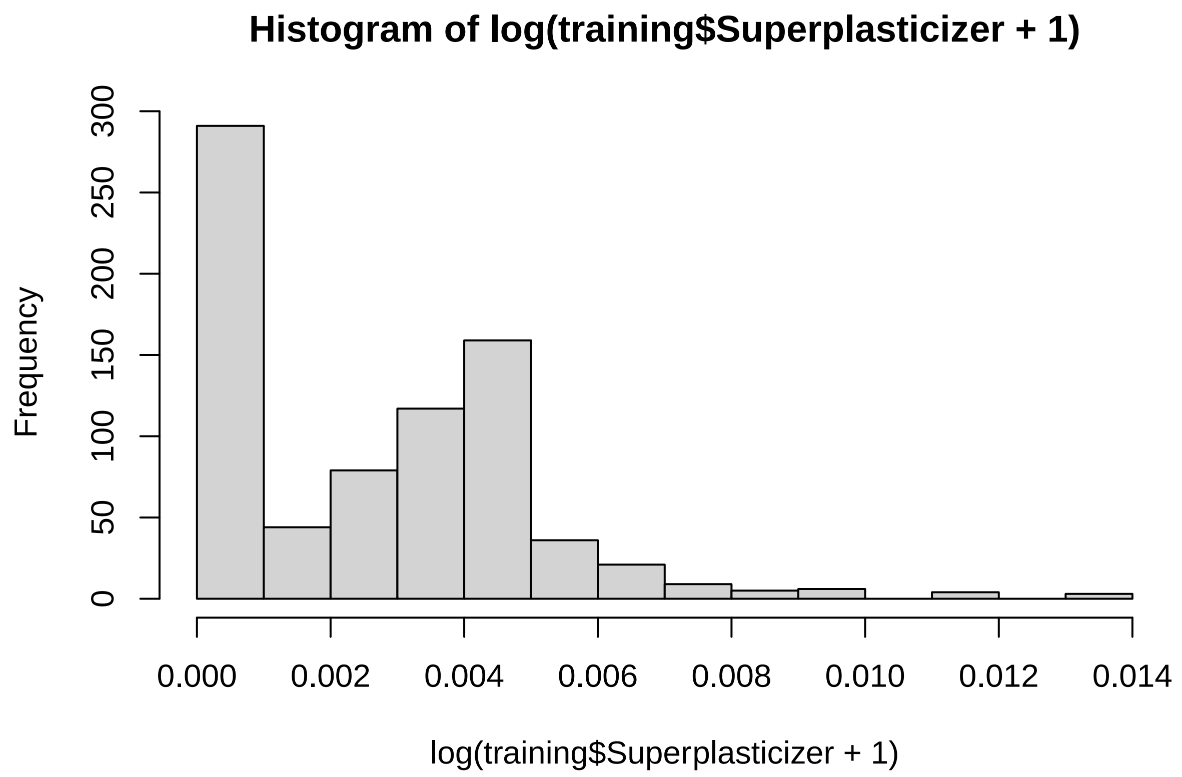
Q3

- Load the cement data
- 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?

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



```
hist(log(training$Superplasticizer+1))
```



Q4

- Load the Alzheimer's disease data
- 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?

```
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,]
# select out columes start with "IL"
b <- training[,grep("^IL", colnames((training)))]

preProcess(b, method="pca", thresh = 0.9)
```

```
## Created from 251 samples and 12 variables
##
## Pre-processing:
##   - centered (12)
##   - ignored (0)
##   - principal component signal extraction (12)
##   - scaled (12)
##
## PCA needed 9 components to capture 90 percent of the variance
```

```r
# below are additional code for exploring the data
c <- as.vector(names(b))
d <- as.formula(paste("diagnosis ~ ", paste(c, collapse="+")))
model <- train(d, method="glm",
               preProcess="pca", data=training)
confusionMatrix(testing$diagnosis, predict(model, testing))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Impaired Control
##   Impaired        2      20
##   Control         2      58
##
##                Accuracy : 0.7317
##                  95% CI : (0.6224, 0.8236)
##     No Information Rate : 0.9512
##     P-Value [Acc > NIR] : 1.0000000
##
##                   Kappa : 0.0777
##
##  Mcnemar's Test P-Value : 0.0002896
##
##             Sensitivity : 0.50000
##             Specificity : 0.74359
##          Pos Pred Value : 0.09091
##          Neg Pred Value : 0.96667
##              Prevalence : 0.04878
##          Detection Rate : 0.02439
##    Detection Prevalence : 0.26829
##       Balanced Accuracy : 0.62179
##
##        'Positive' Class : Impaired
##
```

Q5

- Load the Alzheimer's disease data
- 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?

```r
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,]

# select all columns start with "IL" in the training and testing set, and add diagnosis
trainingIL <- training[,grep("^IL|diagnosis", colnames(training))]
testingIL <- testing[,grep("^IL|diagnosis", colnames(testing))]

## model1 -- using all variables
model1 <- train(diagnosis~., method="glm",
               data=trainingIL)
predict_model1 <- predict(model1, testingIL)
confusionMatrix(predict_model1, testingIL$diagnosis)$overall[1]
```

```
##  Accuracy
## 0.7560976
```

```r
## model2 --- using PCA 80%

model2 <- train(diagnosis ~., method="glm",
                data=trainingIL, preProcess="pca",
                trControl=trainControl(preProcOptions=list(thresh=0.8)))
confusionMatrix(testingIL$diagnosis, predict(model2, testingIL))$overall[1]
```

```
##  Accuracy
## 0.7195122
```