Quiz 2

Aaron Brown

December 16, 2015

Question 1

```
library(AppliedPredictiveModeling); require(caret)
```

```
## Warning: package 'AppliedPredictiveModeling' was built under R version
## 3.2.4
```

```
## Loading required package: caret
```

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

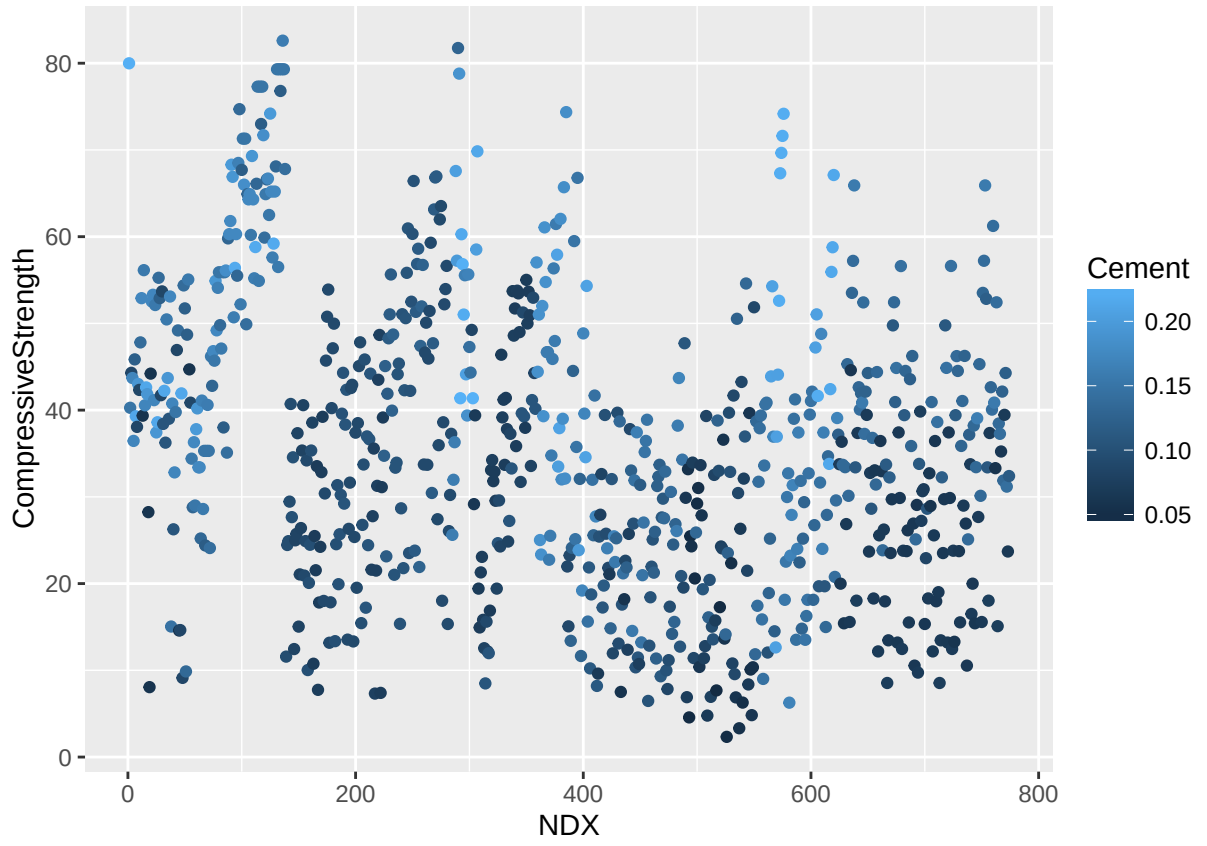
```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 3.2.4
```

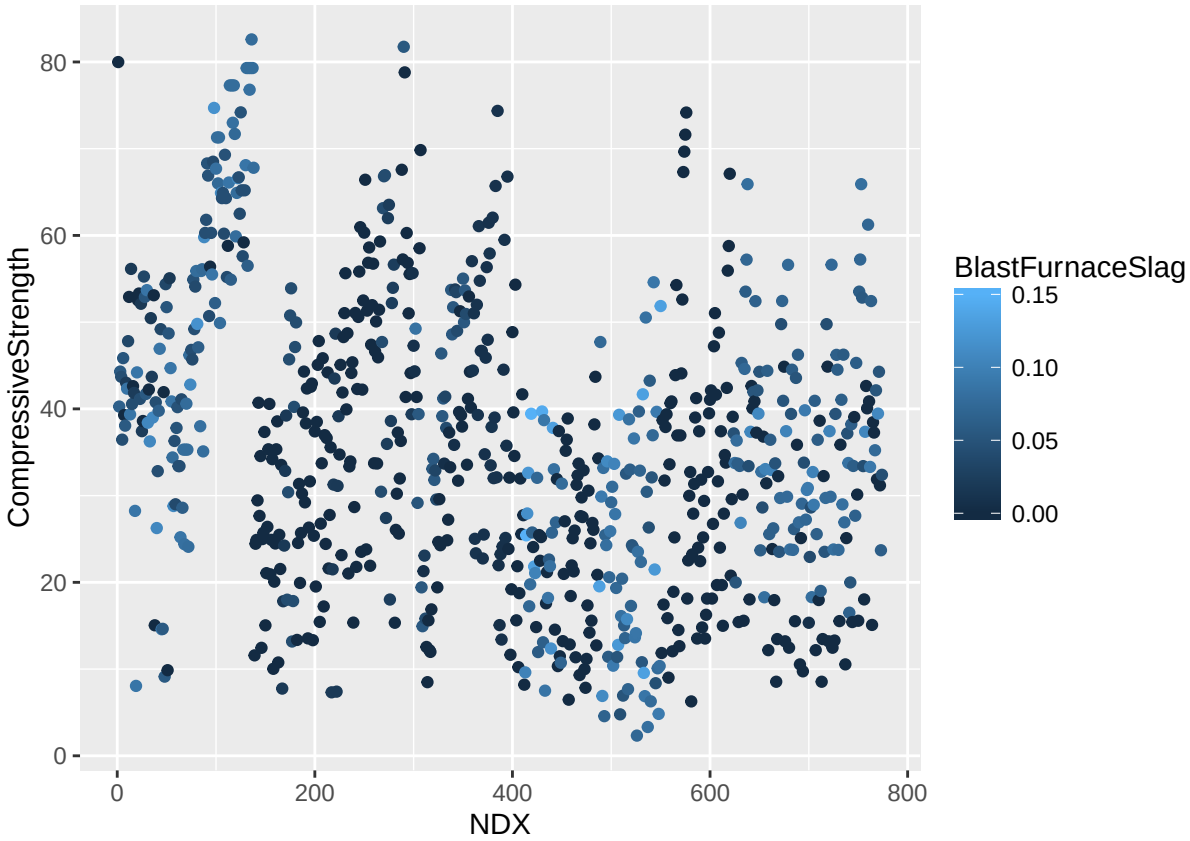
```
data(AlzheimerDisease)

#1
adData = data.frame(diagnosis,predictors)
testIndex = createDataPartition(diagnosis, p = 0.50,list=FALSE)
training = adData[-testIndex,]
testing = adData[testIndex,]

#2
adData = data.frame(diagnosis,predictors)
trainIndex = createDataPartition(diagnosis,p=0.5,list=FALSE)
training = adData[trainIndex,]
testing = adData[-trainIndex,]

#3 - won't work because it randomly samples the training and test set independently
adData = data.frame(diagnosis,predictors)
train = createDataPartition(diagnosis, p = 0.50,list=FALSE)
test = createDataPartition(diagnosis, p = 0.50,list=FALSE)

#4
adData = data.frame(diagnosis,predictors)
trainIndex = createDataPartition(diagnosis,p=0.5,list=FALSE)
training = adData[trainIndex,]
testing = adData[trainIndex,]  #won't work bc it's taking the trainIndex for testing
```

Question 2

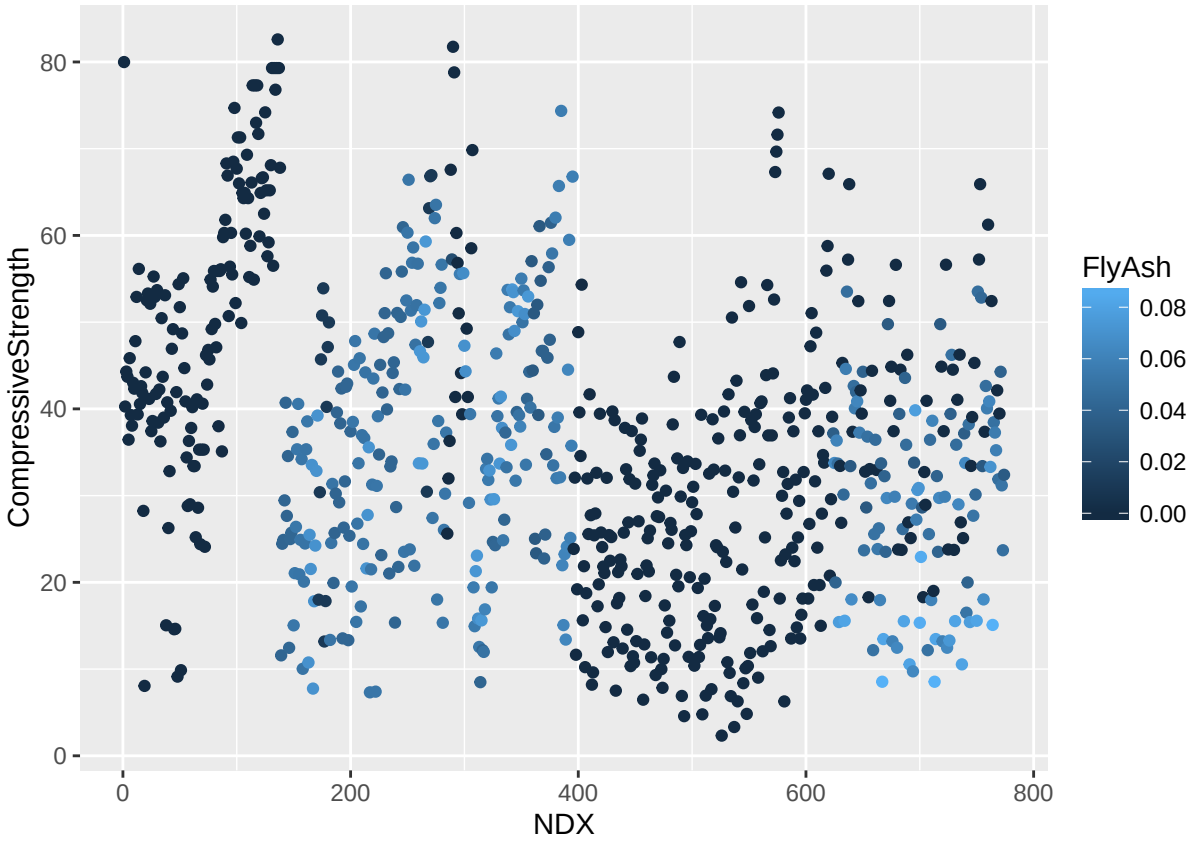
```
library(AppliedPredictiveModeling)
data(concrete)
library(caret)
set.seed(1000)
inTrain = createDataPartition(mixtures$CompressiveStrength, p = 3/4)[[1]]
training = mixtures[ inTrain,]
testing = mixtures[-inTrain,]

training$NDX <- seq(1,nrow(training), 1)

ggplot(data = training) + aes(x = NDX, y = CompressiveStrength,
                              colour = Cement) + geom_point()
```



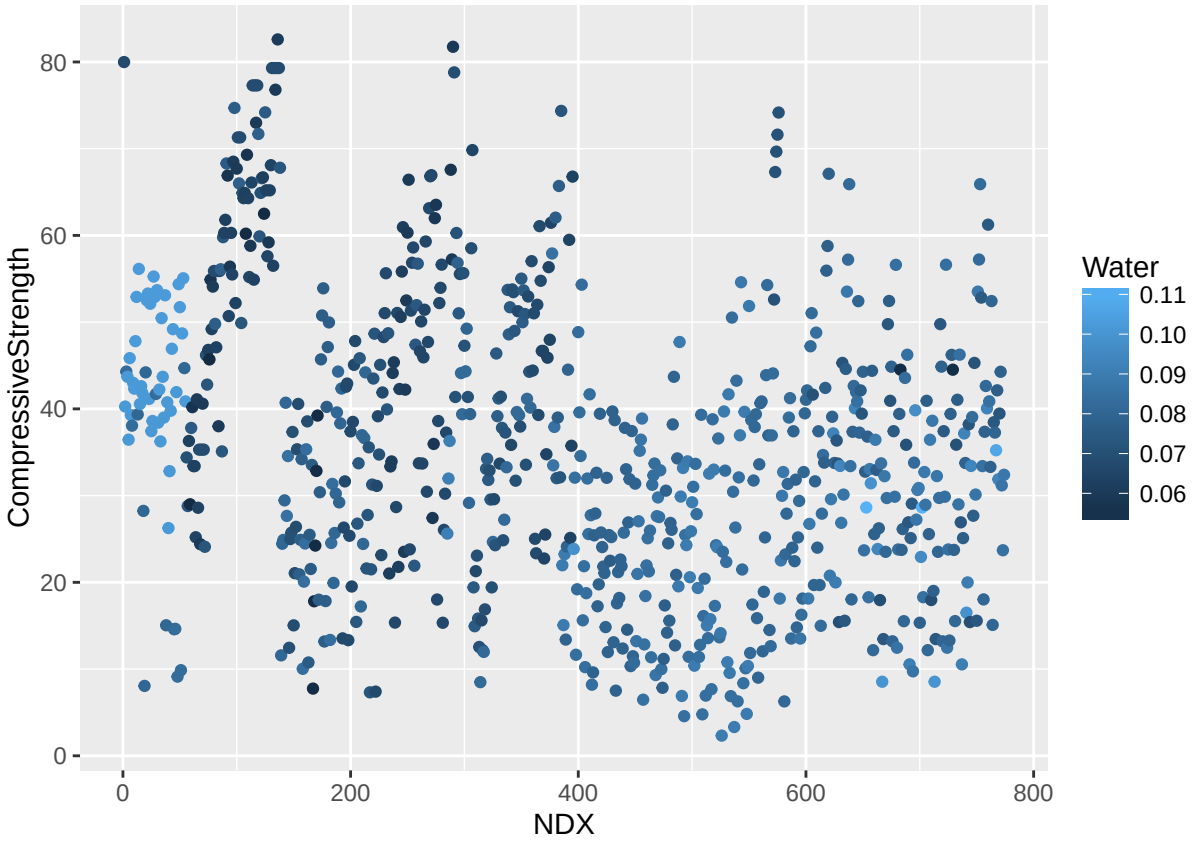

```
ggplot(data = training) + aes(x = NDX, y = CompressiveStrength,
                              colour = BlastFurnaceSlag) + geom_point()
```



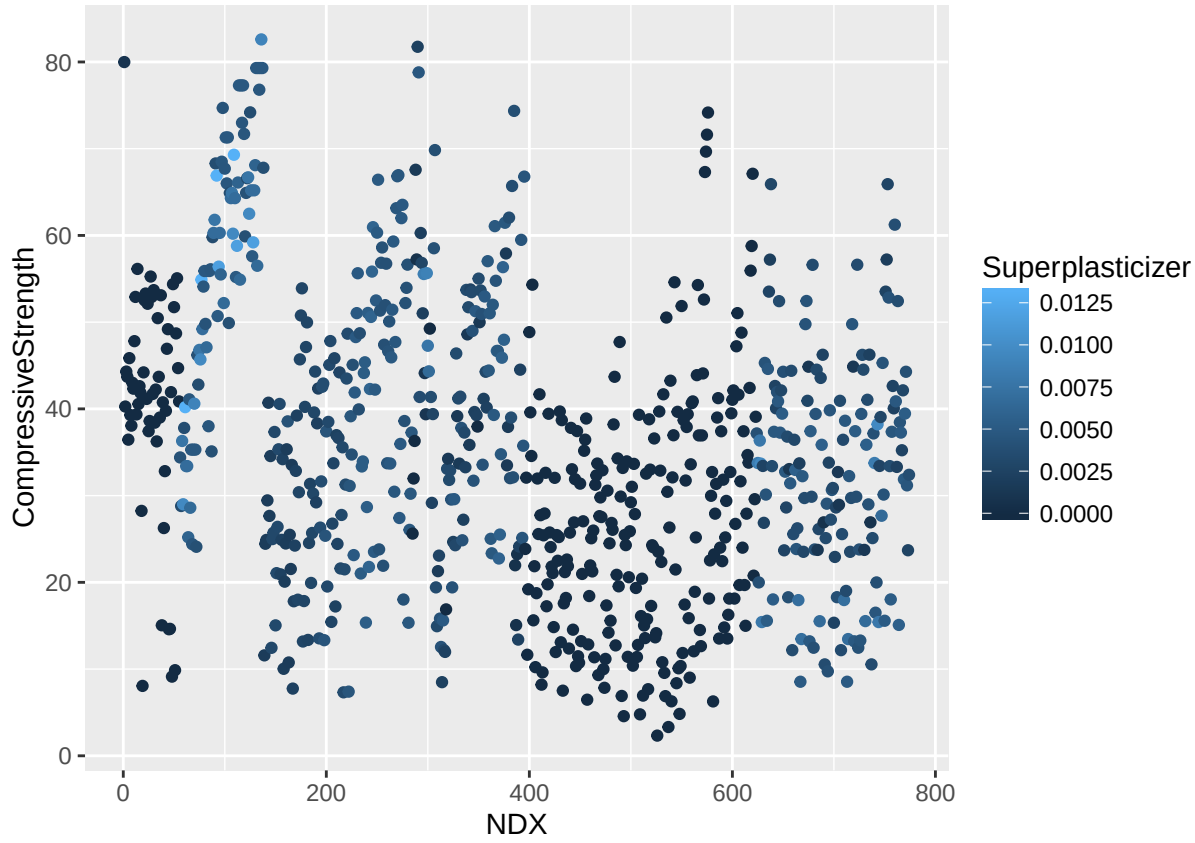

```
ggplot(data = training) + aes(x = NDX, y = CompressiveStrength,
                              colour = FlyAsh) + geom_point()
```



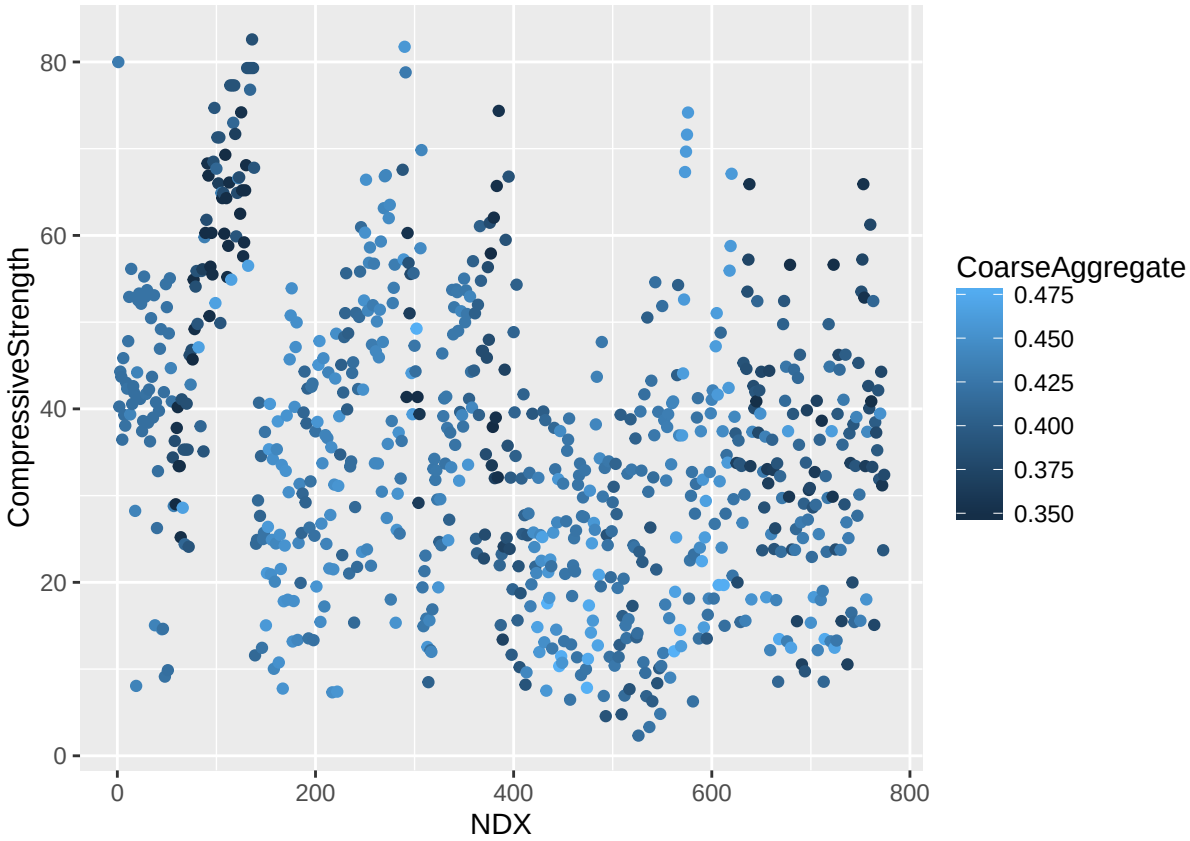

```
ggplot(data = training) + aes(x = NDX, y = CompressiveStrength,
                              colour = Water) + geom_point()
```



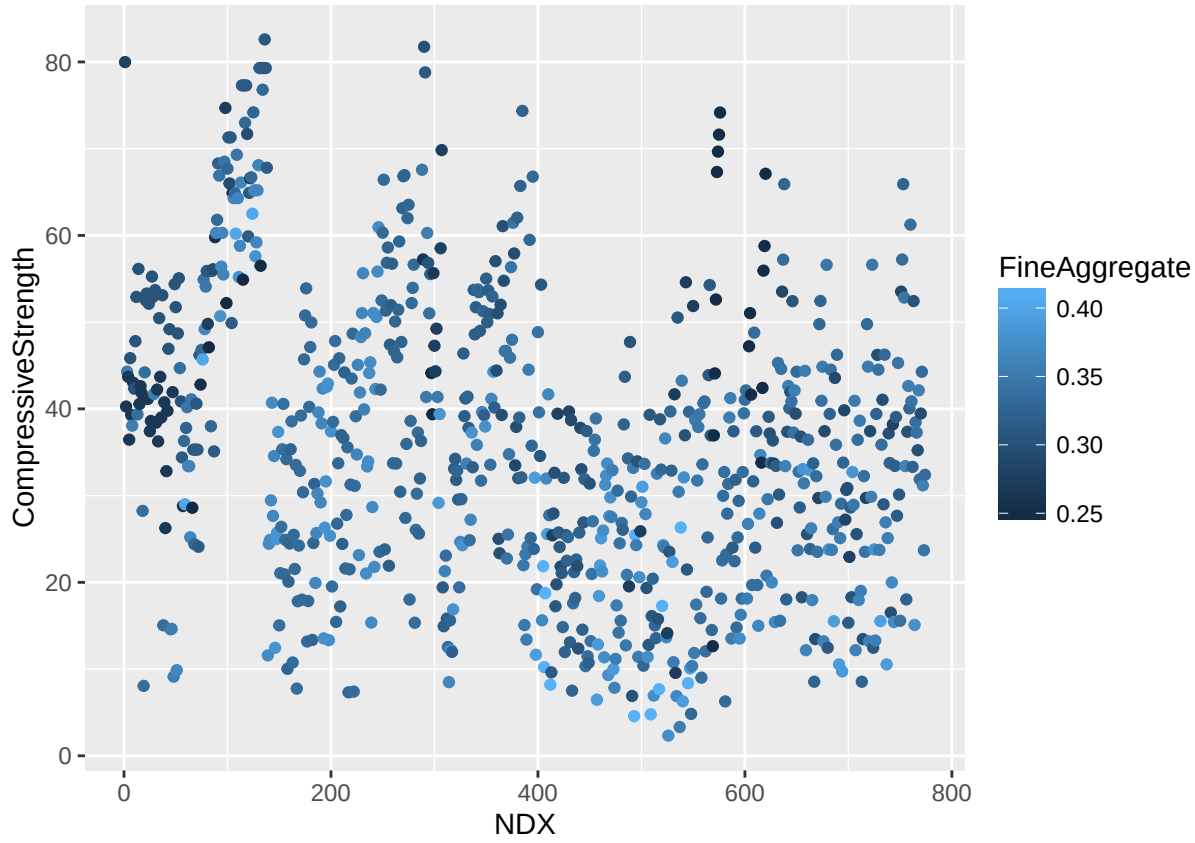

```
ggplot(data = training) + aes(x = NDX, y = CompressiveStrength,
                              colour = Superplasticizer) + geom_point()
```



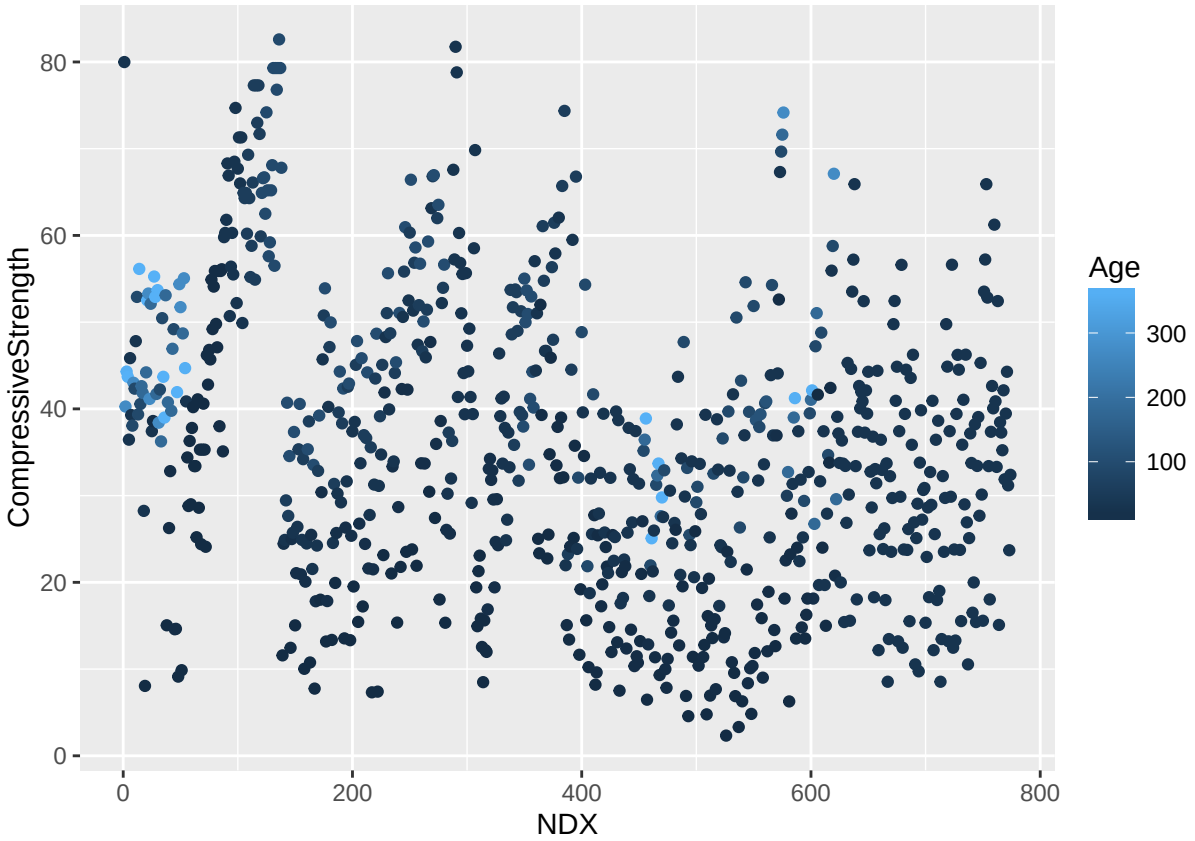

```
ggplot(data = training) + aes(x = NDX, y = CompressiveStrength,
                              colour = CoarseAggregate) + geom_point()
```




```
ggplot(data = training) + aes(x = NDX, y = CompressiveStrength,
                              colour = FineAggregate) + geom_point()
```



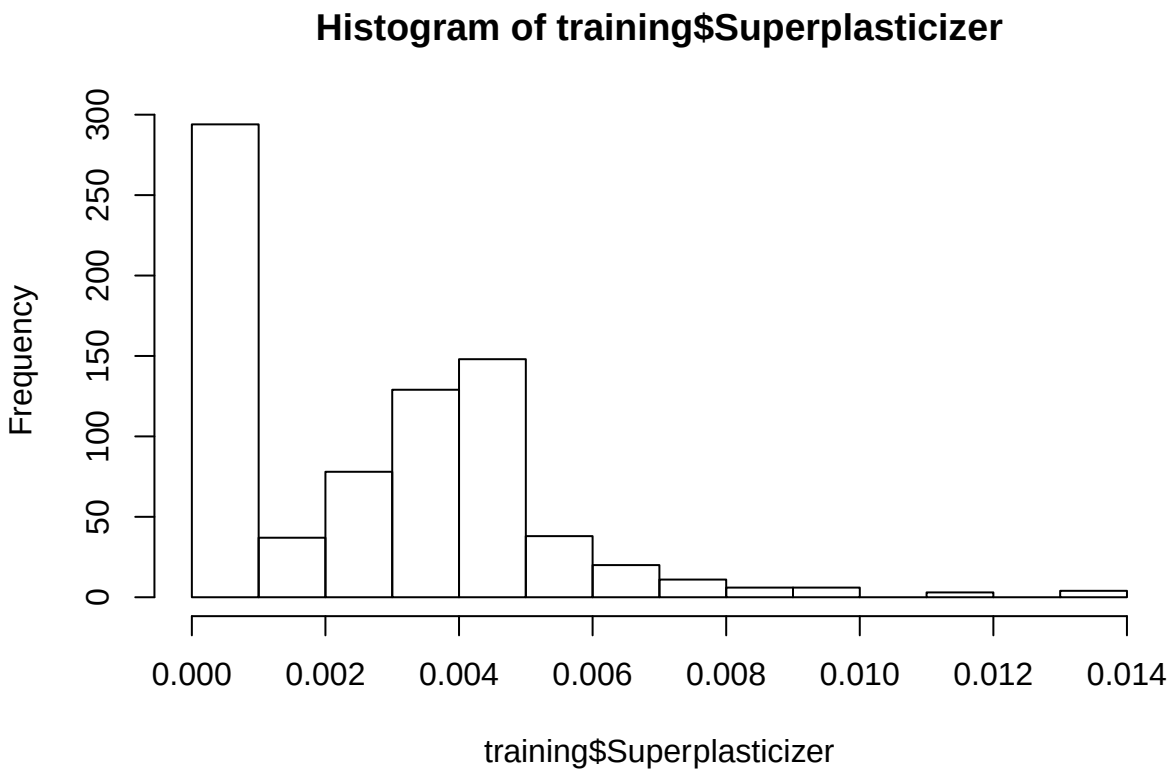

```
ggplot(data = training) + aes(x = NDX, y = CompressiveStrength,
                              colour = Age) + geom_point()
```



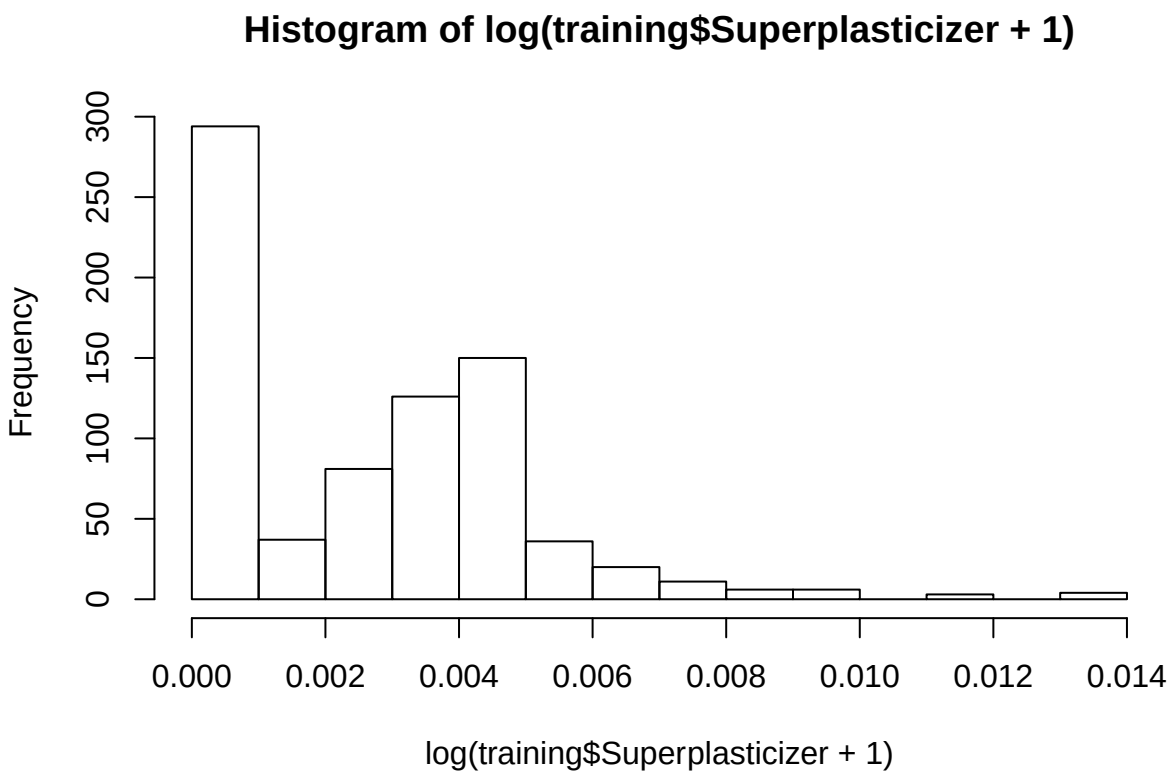
Answer: There is a non-random pattern in the plot of the outcome versus index.

Question 3

```
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))
```



Answer: 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.

– or –

The log transform does not reduce the skewness of the non-zero values of SuperPlasticizer

Question 4

```
##this is the principal component analysis techniques where we identify the number of
##principal components in order to capture 80% of the variance
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,]

ILcols <- training[, grep("^IL", colnames(training))] #creates new DF with only IL cols

a4 <- preProcess(ILcols, method = 'pca', thresh = 0.8)
```

Answer: 7

Question 5

```
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,]

trainingILcols <- training[, c(1,grep("^IL", colnames(training)))] #creates new DF with only IL cols
testingILcols  <- testing[, c(1,grep("^IL", colnames(testing)))] #creates new DF with only IL cols

##Pricipal Components Analysis
preProc <- preProcess(trainingILcols, method = "pca", thresh = 0.8)
trainPC <- predict(preProc, trainingILcols[,-1])
modelFitPC <- train(trainingILcols[,1] ~ ., method = "glm", data = trainPC)

testPC <- predict(preProc, testingILcols[,-1])
PCArslt <- confusionMatrix(testing[,1], predict(modelFitPC, testPC))
PCA.accuracy <- PCArslt[[3]][[1]]

ILcols <- training[, c(1,grep("^IL", colnames(testing)))] #creates new DF with only IL cols

modelFit <- train(ILcols[,1] ~ ., method = "glm", data = ILcols[,-1])
no.PCArslt <- confusionMatrix(ILcols[,1], predict(modelFit, ILcols[,-1]))
no.PCA.accuracy <- no.PCArslt[[3]][[1]]
```

Answer: Non-PCA accuracy: 0.7370518
PCA accuracy: 0.7195122