Practical Machine Learning Quiz 2

Jason Liu

Oct 6, 2016

This is a R Markdown file that serves the purpose of the solution of the second quiz in the course of 'Practical Machine Learning', taught by JHU.

Load Packages

```r
library(caret)
library(AppliedPredictiveModeling)
library(Hmisc)
library(ggplot2)
library(Metrics)
```

Question 1

Load the data based on the instruction

```r
data(AlzheimerDisease)
```

Spliting the data into a training set and a test set, with the ration of 0.5.

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

Key to answer: Never forget to add 'list=FALSE', otherwise the data partition will create a list other than a series of index.

Question 2

Load the data based on the instruction

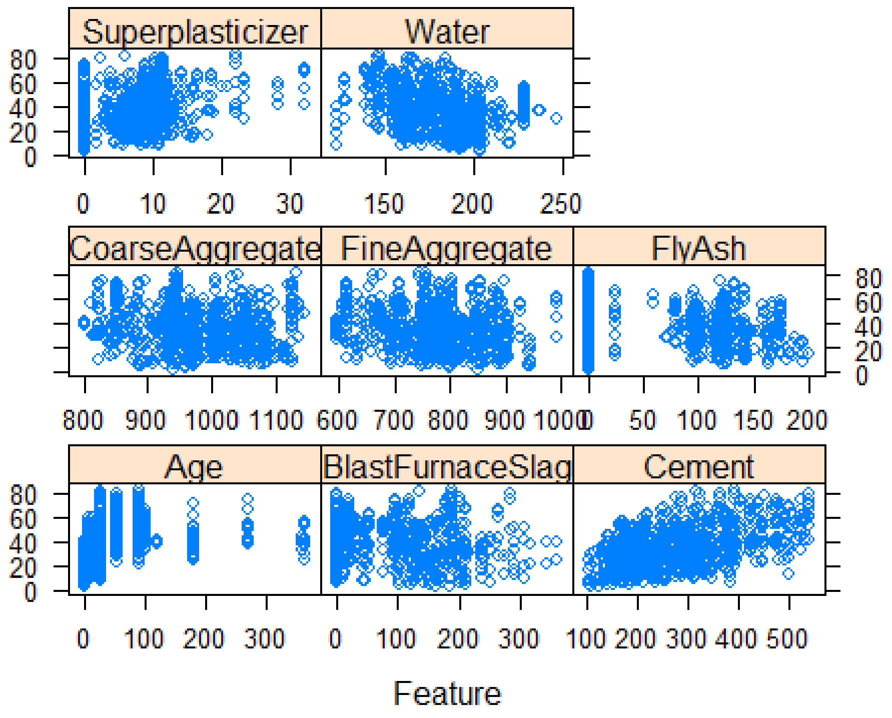
```r
library(AppliedPredictiveModeling)
data(concrete)
library(caret)
set.seed(1000)
inTrain = createDataPartition(mixtures$CompressiveStrength, p =
3/4)[[1]]
training = mixtures[ inTrain,]
testing = mixtures[-inTrain,]
```

Note: If you do not add 'list = FALSE' in data slicing function, you can use the above way to create data partition.

Instruction: 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?

We first make a feature plot to check whether there is a correlation between any of two variables.

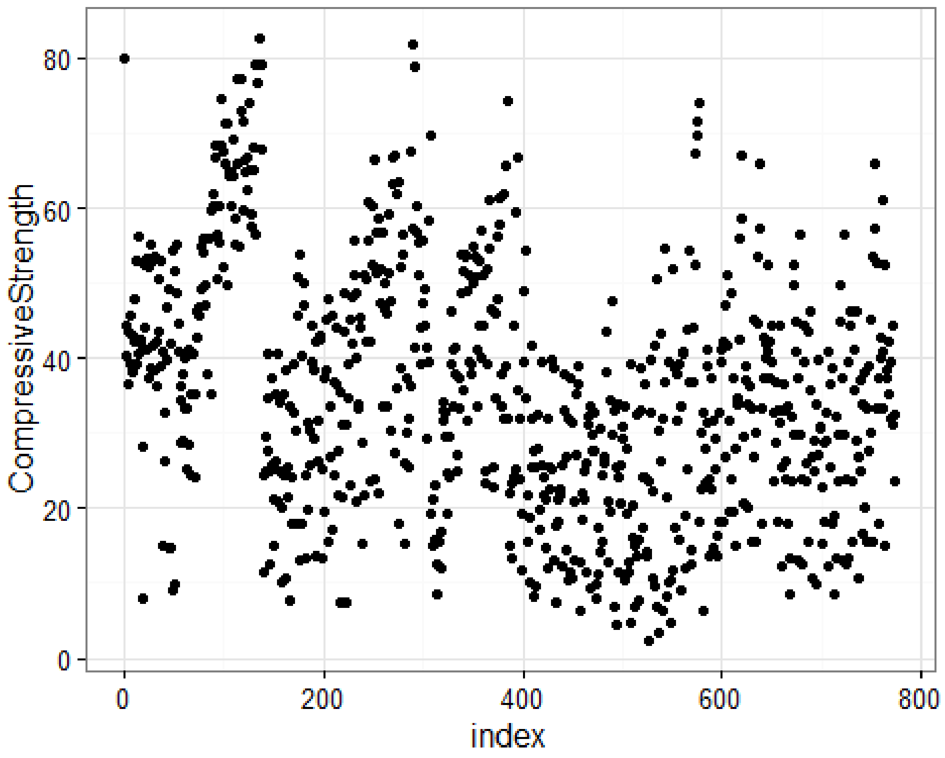
```
featurePlot(concrete[,-9],concrete$CompressiveStrength)
```



It is obvious that only cement has a 'good' correlation with the response variable, judged by the correlation plots.

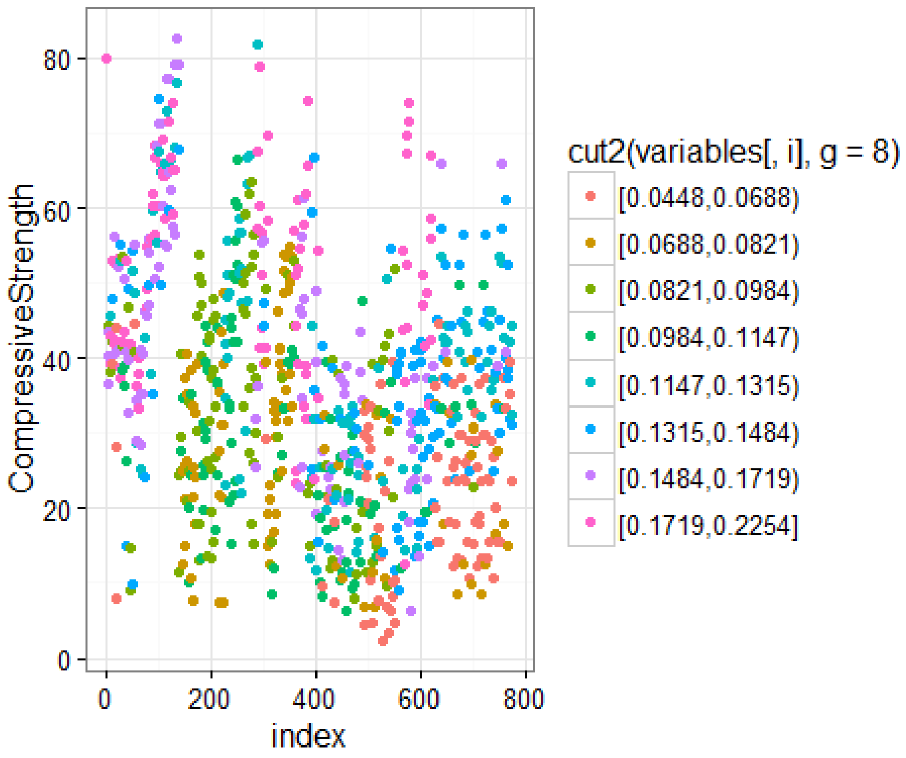
Then we make a plot of the index and the response variables.

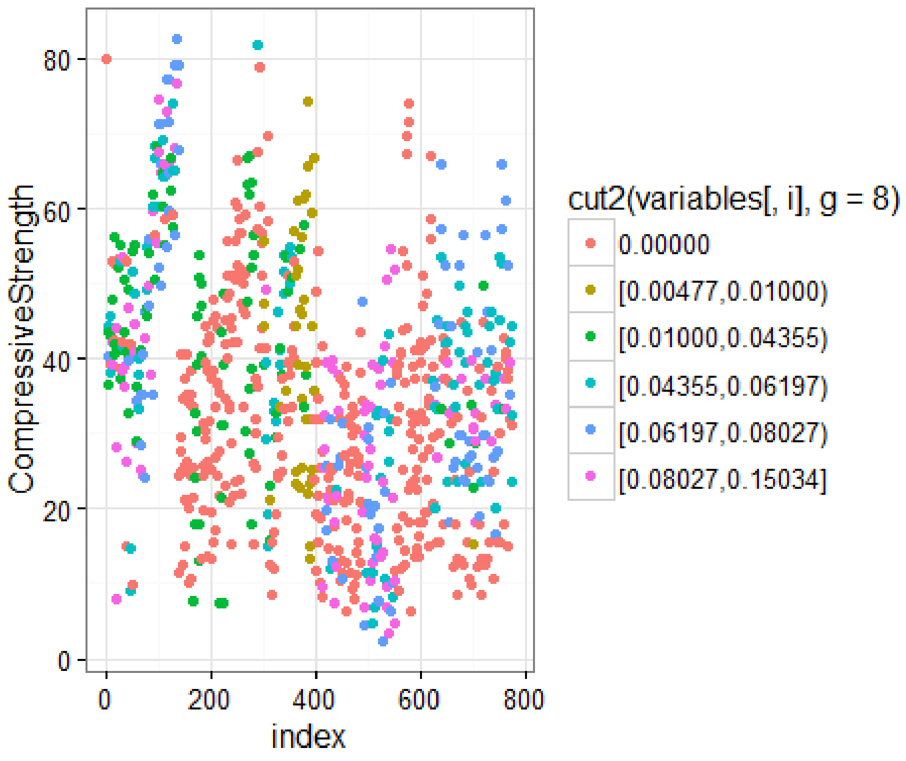
```
index <- seq_along(1:nrow(training))
ggplot(data = training, aes(x = index, y = CompressiveStrength)) +
geom_point() +
    theme_bw()
```

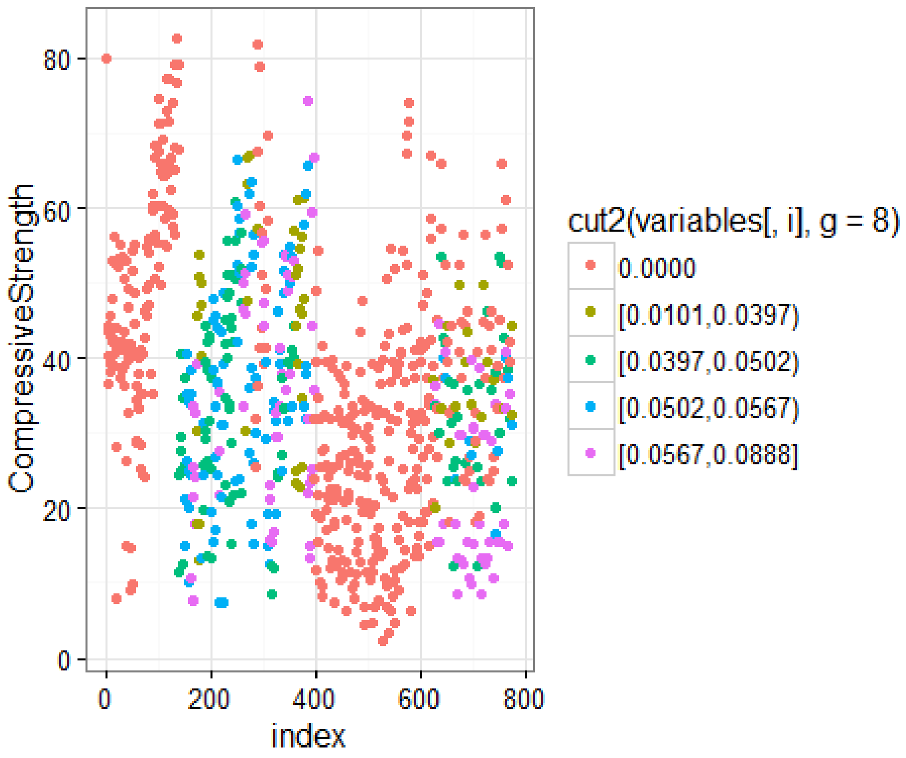


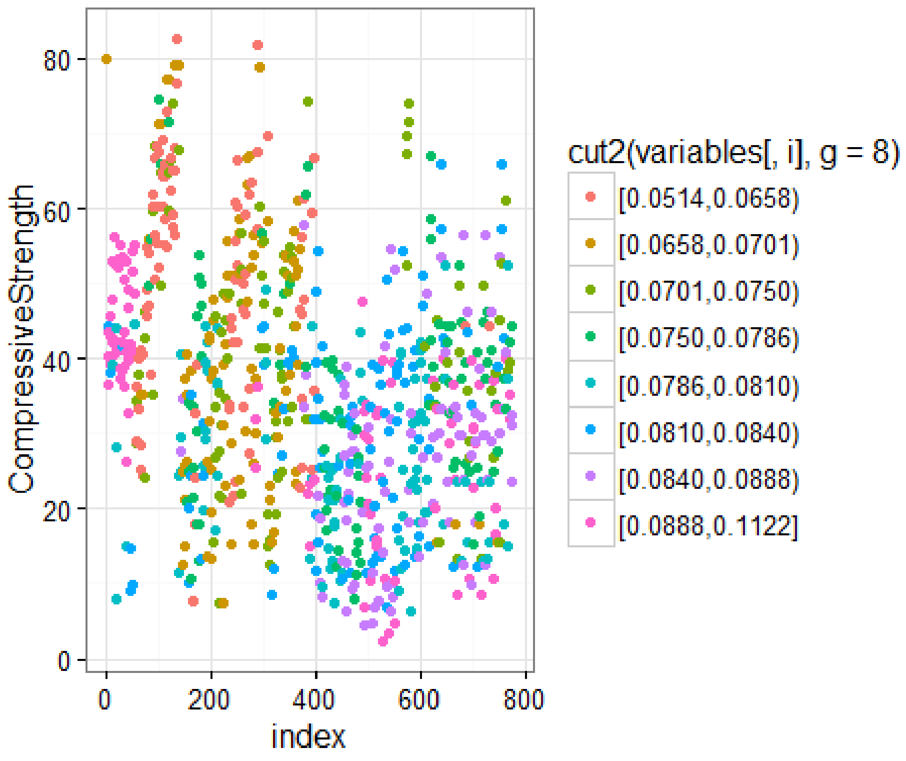
Seems that the points are not randomly distributed as they show the pattern of several 'steps'. In order to further explore the relationship between th pattern and the other variables, we create several plots to check that.

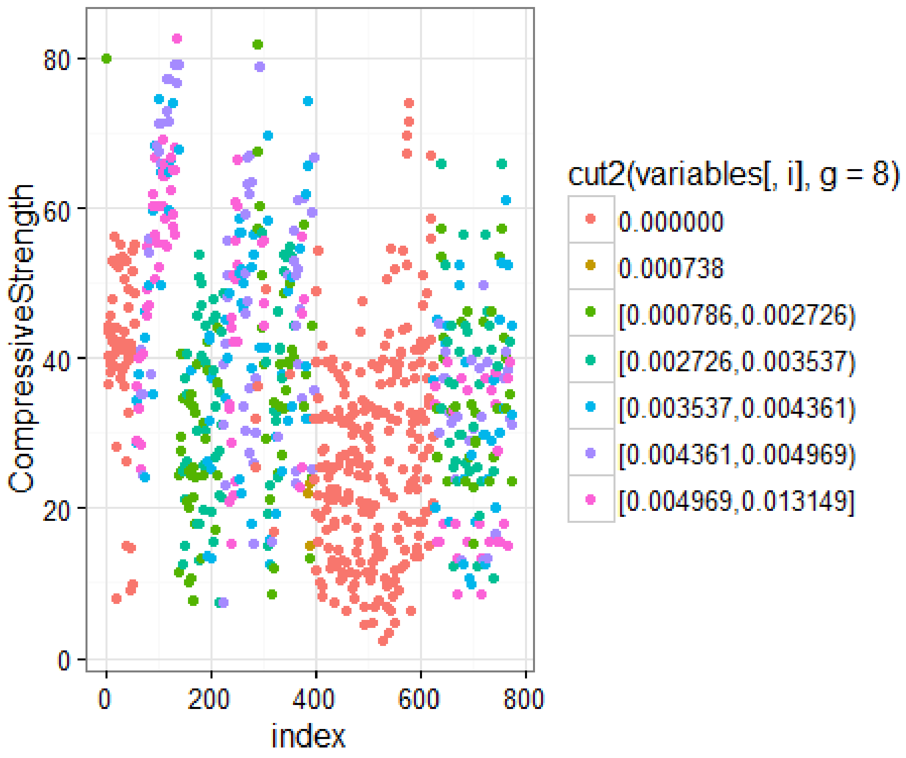
```
variables<- training[,1:8]
for(i in 1:8){
  g<-qplot(index,CompressiveStrength,data =
training,color=cut2(variables[,i],g = 8))+theme_bw()
  print(g)
  }
```


CompressiveStrength
index
cut2(variables[, i], g = 8)
[0.0448,0.0688)
[0.0688,0.0821)
[0.0821,0.0984)
[0.0984,0.1147)
[0.1147,0.1315)
[0.1315,0.1484)
[0.1484,0.1719)
[0.1719,0.2254]

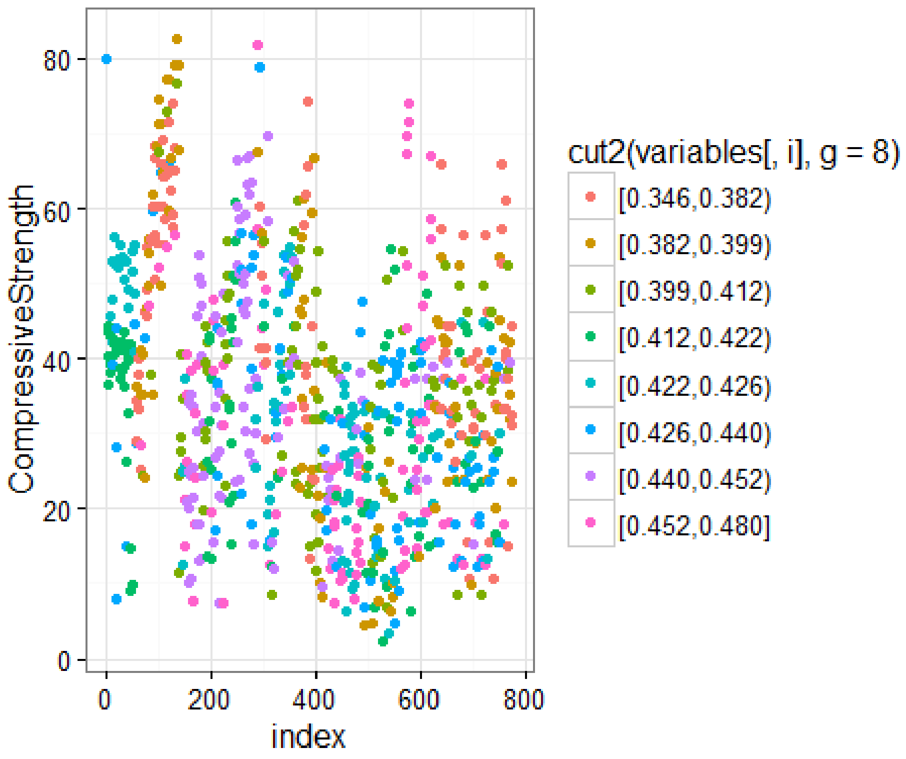

CompressiveStrength
index
cut2(variables[, i], g = 8)
0.00000
[0.00477,0.01000)
[0.01000,0.04355)
[0.04355,0.06197)
[0.06197,0.08027)
[0.08027,0.15034]


CompressiveStrength
index
cut2(variables[, i], g = 8)
0.0000
[0.0101,0.0397)
[0.0397,0.0502)
[0.0502,0.0567)
[0.0567,0.0888]

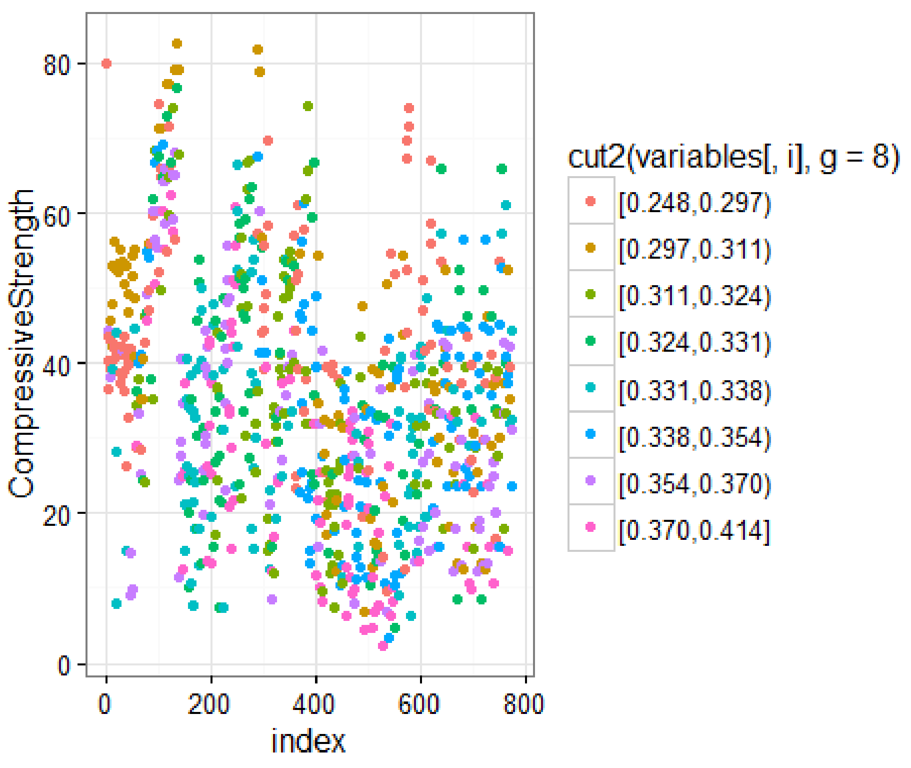

CompressiveStrength
index
cut2(variables[, i], g = 8)
[0.0514,0.0658)
[0.0658,0.0701)
[0.0701,0.0750)
[0.0750,0.0786)
[0.0786,0.0810)
[0.0810,0.0840)
[0.0840,0.0888)
[0.0888,0.1122]

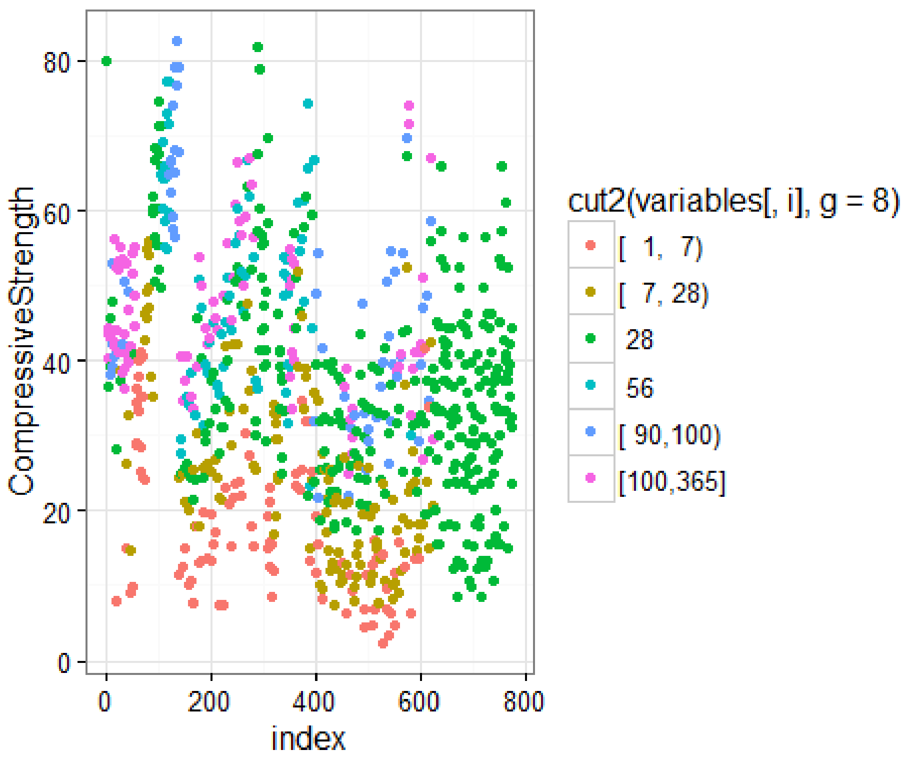


CompressiveStrength
index
0
20
40
60
80
200
400
600
800
cut2(variables[, i], g = 8)
● 0.000000
● 0.000738
● [0.000786,0.002726)
● [0.002726,0.003537)
● [0.003537,0.004361)
● [0.004361,0.004969)
● [0.004969,0.013149]



CompressiveStrength
index
0
20
40
60
80
200
400
600
800
cut2(variables[, i], g = 8)
● [0.346,0.382)
● [0.382,0.399)
● [0.399,0.412)
● [0.412,0.422)
● [0.422,0.426)
● [0.426,0.440)
● [0.440,0.452)
● [0.452,0.480]





Seems that the 'step-like' pattern of the data is not the result of any variables. Also, it is not reasonalbe enough to guess that the pattern is the result of missing variables.

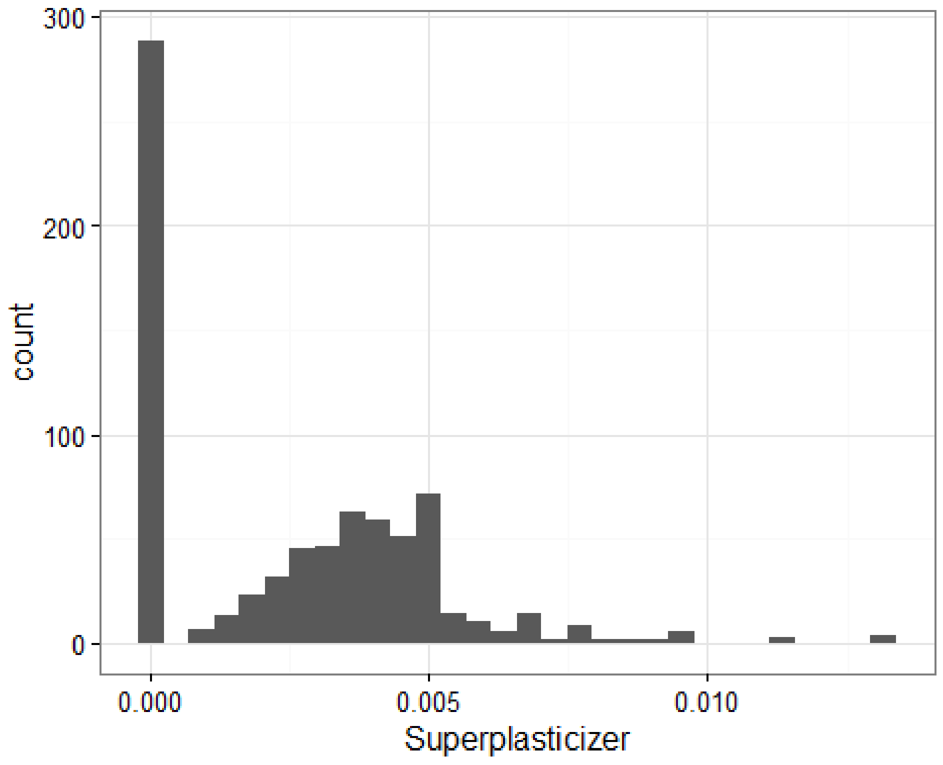
Question 3

We load the data based on the instruction in quiz.

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

Instruction: 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?

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



We can find that a majority of data is centered around zero, which will produce errors when trying to transform them into logrithm form.

Question 4

We load the data based on the instruction in quiz.

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

Instruction: 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?

```
Vars<-grep("^IL", colnames(training), value = TRUE)
Prin<- preProcess(training[, Vars], method = "pca", thresh = 0.9)
Prin
```

```
## 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
```

It is obvious that 9 components are required to capture 90% of variance.

Question 5

We load the data based on the instruction in quiz.

```
set.seed(3433)
library(AppliedPredictiveModeling)
data(AlzheimerDisease)
adData = data.frame(diagnosis, predictors)
inTrain = createDataPartition(adData$diagnosis, p = 3/4)[[1]]
training = adData[inTrain, ]
testing = adData[-inTrain, ]
Vars<-grep("^IL", colnames(training), value = TRUE)
train<- training[,Vars]
```

Instruction: 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.

1. glm with non-pca preprocessing

```r
fit1<- train(train,training$diagnosis,method='glm')
predict1<- predict(fit1,newdata=testing[,Vars])
confusionMatrix(predict1,testing$diagnosis)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Impaired Control
##   Impaired        2       9
##   Control        20      51
##
##                Accuracy : 0.6463
##                  95% CI : (0.533, 0.7488)
##     No Information Rate : 0.7317
##     P-Value [Acc > NIR] : 0.96637
##
##                   Kappa : -0.0702
##  Mcnemar's Test P-Value : 0.06332
##
##             Sensitivity : 0.09091
##             Specificity : 0.85000
##          Pos Pred Value : 0.18182
##          Neg Pred Value : 0.71831
##              Prevalence : 0.26829
##          Detection Rate : 0.02439
##    Detection Prevalence : 0.13415
##       Balanced Accuracy : 0.47045
##
##        'Positive' Class : Impaired
##
```

2. glm with PCA: 80% Variance Captured

```r
PCA<- preProcess(training[,Vars],method = 'pca',thresh = 0.8)
new_train<- predict(PCA,training[,Vars])
new_test <- predict(PCA,testing[,Vars])
fit2<- train(new_train,training$diagnosis,method='glm')
predict2<- predict(fit2,newdata = new_test)
confusionMatrix(predict2,testing$diagnosis)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Impaired Control
##   Impaired        3       4
##   Control        19      56
##
##                Accuracy : 0.7195
##                  95% CI : (0.6094, 0.8132)
##     No Information Rate : 0.7317
##     P-Value [Acc > NIR] : 0.651780
```

```
##
##                  Kappa : 0.0889
##  Mcnemar's Test P-Value : 0.003509
##
##            Sensitivity : 0.13636
##            Specificity : 0.93333
##         Pos Pred Value : 0.42857
##         Neg Pred Value : 0.74667
##             Prevalence : 0.26829
##         Detection Rate : 0.03659
##   Detection Prevalence : 0.08537
##      Balanced Accuracy : 0.53485
##
##       'Positive' Class : Impaired
##
```

Therefore, the answer should be: 0.65 and 0.72.