Practical Machine Learning - Week 2

Saul Lugo

January 11, 2016

Splitting Data, Plotting Predictors and Training Models

The following are examples of how to split the data set in training and testing sets, how to train the model and how to plot the predictors to analyze the relationship between the predictors and the outcome.

Loading the Data

Is this example, the ISLR packages is used. This package has a dataset of Wages in the US.

```r
require(ISLR); require(ggplot2); require(caret);
```

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

## Warning: package 'ggplot2' was built under R version 3.1.3

## Loading required package: caret

## Warning: package 'caret' was built under R version 3.1.3

## Loading required package: lattice

## Warning: package 'lattice' was built under R version 3.1.3
```

```r
data(Wage)
head(Wage)
```

```
##        year age     sex           maritl     race       education
## 231655 2006  18 1. Male 1. Never Married 1. White    1. < HS Grad
## 86582  2004  24 1. Male 1. Never Married 1. White 4. College Grad
## 161300 2003  45 1. Male       2. Married 1. White 3. Some College
## 155159 2003  43 1. Male       2. Married 3. Asian 4. College Grad
## 11443  2005  50 1. Male      4. Divorced 1. White      2. HS Grad
## 376662 2008  54 1. Male       2. Married 1. White 4. College Grad
##                    region       jobclass         health health_ins
## 231655 2. Middle Atlantic  1. Industrial      1. <=Good      2. No
## 86582  2. Middle Atlantic 2. Information 2. >=Very Good      2. No
## 161300 2. Middle Atlantic  1. Industrial      1. <=Good     1. Yes
## 155159 2. Middle Atlantic 2. Information 2. >=Very Good     1. Yes
## 11443  2. Middle Atlantic 2. Information      1. <=Good     1. Yes
## 376662 2. Middle Atlantic 2. Information 2. >=Very Good     1. Yes
##         logwage      wage
```

```
## 231655 4.318063  75.04315
## 86582  4.255273  70.47602
## 161300 4.875061 130.98218
## 155159 5.041393 154.68529
## 11443  4.318063  75.04315
## 376662 4.845098 127.11574
```

```r
summary(Wage)
```

```
##       year           age              sex                    maritl
##  Min.   :2003   Min.   :18.00   1. Male  :3000   1. Never Married: 648
##  1st Qu.:2004   1st Qu.:33.75   2. Female:   0   2. Married      :2074
##  Median :2006   Median :42.00                    3. Widowed      :  19
##  Mean   :2006   Mean   :42.41                    4. Divorced     : 204
##  3rd Qu.:2008   3rd Qu.:51.00                    5. Separated    :  55
##  Max.   :2009   Max.   :80.00
##
##        race                   education                     region
##  1. White:2480   1. < HS Grad      :268   2. Middle Atlantic   :3000
##  2. Black: 293   2. HS Grad        :971   1. New England       :   0
##  3. Asian: 190   3. Some College   :650   3. East North Central:   0
##  4. Other:  37   4. College Grad   :685   4. West North Central:   0
##                  5. Advanced Degree:426   5. South Atlantic    :   0
##                                           6. East South Central:   0
##                                           (Other)              :   0
##            jobclass               health      health_ins      logwage
##  1. Industrial :1544   1. <=Good     : 858   1. Yes:2083   Min.   :3.000
##  2. Information:1456   2. >=Very Good:2142   2. No : 917   1st Qu.:4.447
##                                                            Median :4.653
##                                                            Mean   :4.654
##                                                            3rd Qu.:4.857
##                                                            Max.   :5.763
##
##       wage
##  Min.   : 20.09
##  1st Qu.: 85.38
##  Median :104.92
##  Mean   :111.70
##  3rd Qu.:128.68
##  Max.   :318.34
##
```

Splitting the Data into Training and Test set

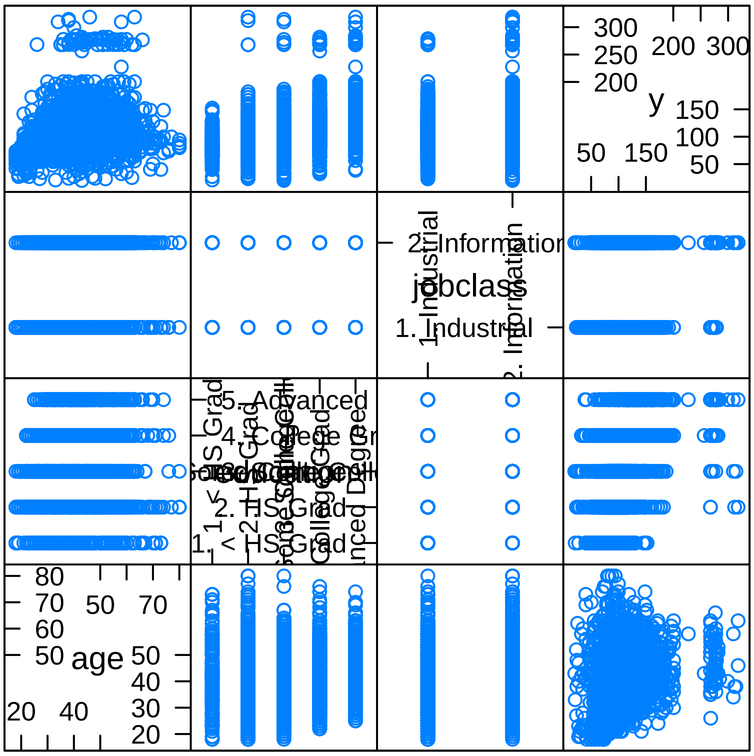
```r
inTrain <- createDataPartition(y = Wage$wage, p = 0.7, list = FALSE)
training <- Wage[inTrain,]
testing <- Wage[-inTrain,]
dim(training); dim(testing)
```

```
## [1] 2102   12
```

```
## [1] 898  12
```

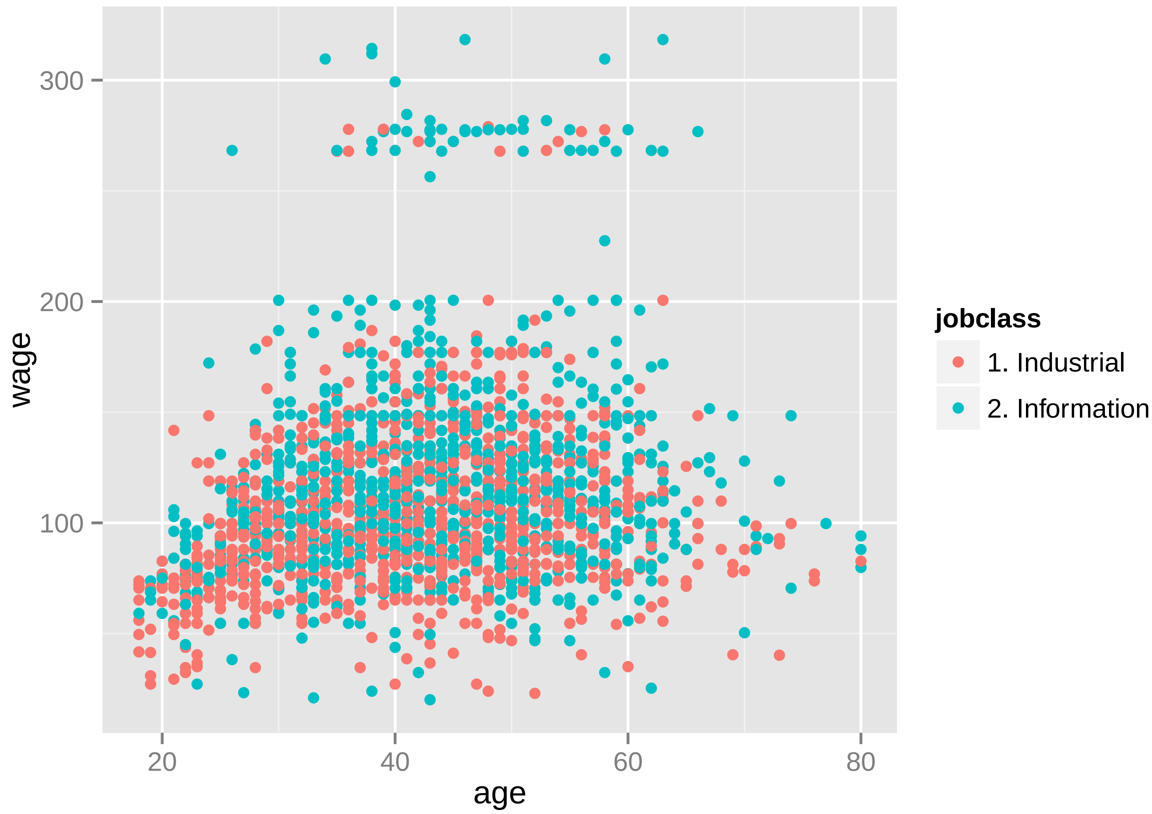
Plotting Predictors vs Outcome

```r
#Plotting several predictors vs the outcome
featurePlot(x = training[,c("age", "education", "jobclass")], y = training$wage, plot="pairs")
```

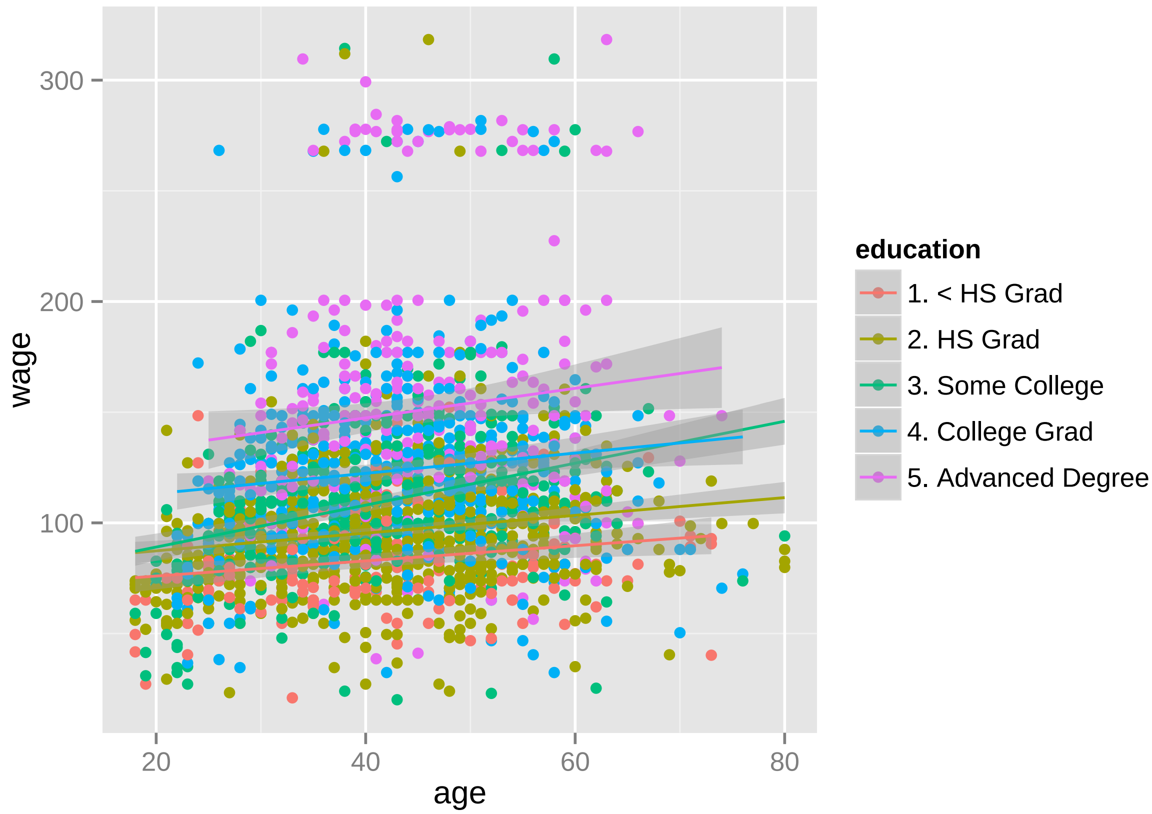


Scatter Plot Matrix

```r
#Plotting one variable vs outcome and adding a second variable in the colour
qplot(age, wage, colour = jobclass,data=training)
```




```
#Add regression smoothers
qq <- qplot(age, wage, colour=education, data=training)
qq + geom_smooth(method="lm", formula = y ~ x)
```




```r
#cut2, making factors (Hmisc package)
require(Hmisc)
```

```
## Loading required package: Hmisc
```

```
## Warning: package 'Hmisc' was built under R version 3.1.3
```

```
## Loading required package: grid
## Loading required package: survival
##
## Attaching package: 'survival'
##
## The following object is masked from 'package:caret':
##
##     cluster
##
## Loading required package: Formula
```

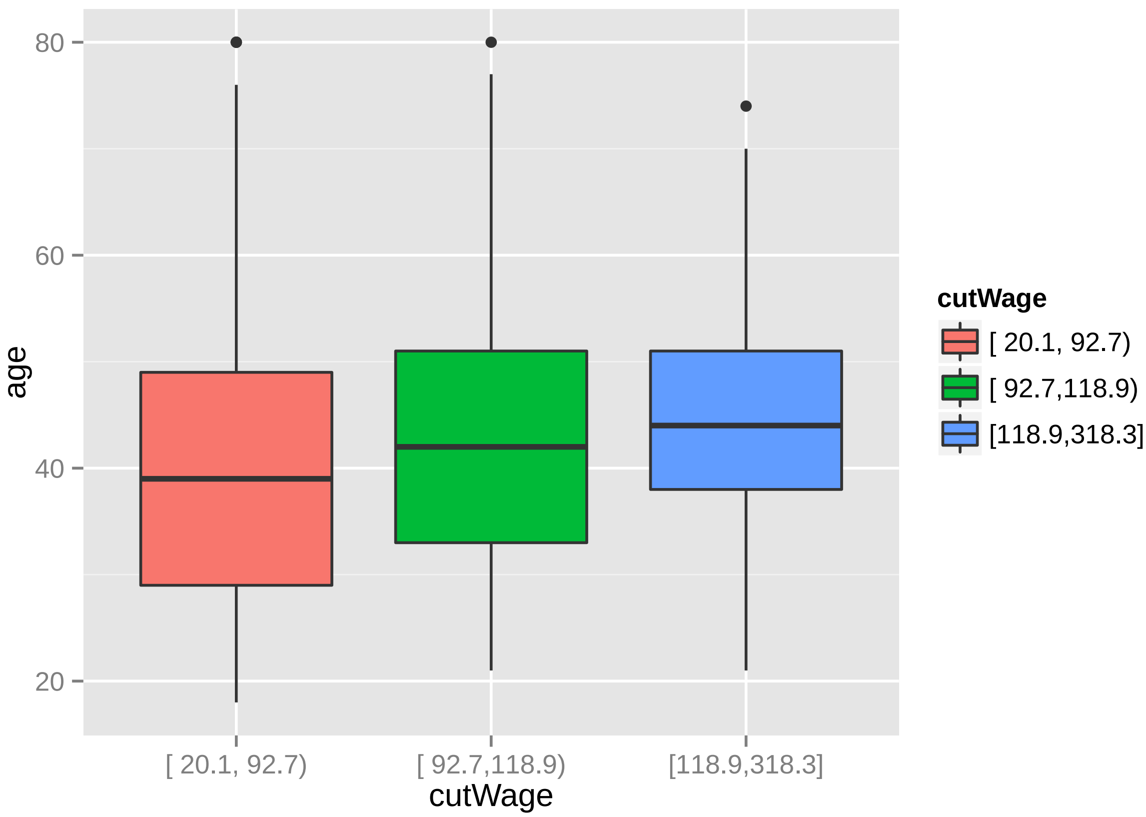
```
## Warning: package 'Formula' was built under R version 3.1.3
```

```
##
## Attaching package: 'Hmisc'
##
## The following objects are masked from 'package:base':
##
##     format.pval, round.POSIXt, trunc.POSIXt, units
```

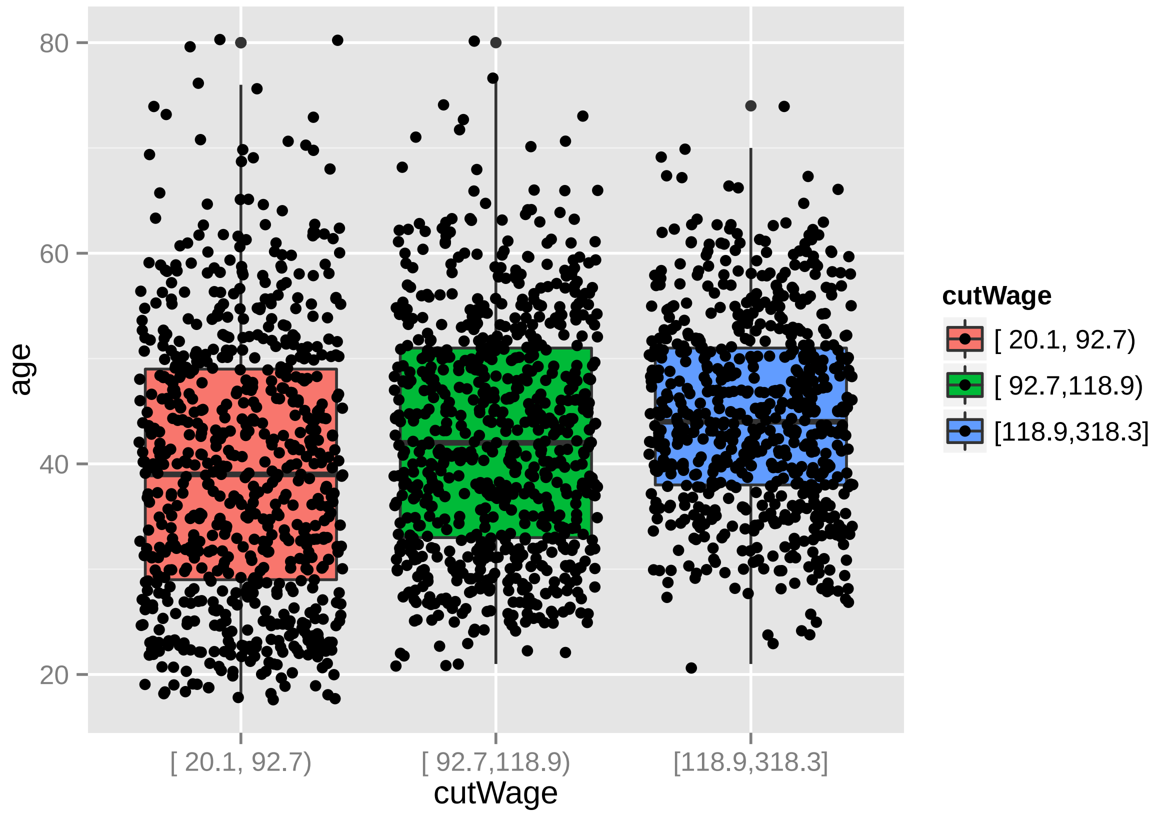
```r
#Splitting the wage variable into groups of quantiles
cutWage <- cut2(training$wage, g=3)
table(cutWage)
```

```
## cutWage
## [ 20.1, 92.7) [ 92.7,118.9) [118.9,318.3]
##           702           729           671
```

```r
#Making a boxplot to see the three different wage groups we created before
p1 <- qplot(cutWage, age, data=training, fill=cutWage, geom = c("boxplot"))
p1
```




```r
#Boxplots with points overlayed
#If the jitter plot shows a lot of the points inside the boxplots that mean that the boxplots are
#actually representative of the data, so any trend one might observes might be true.
#On the contrary if only a few points are shown inside the boxplots, the trend might not be that repres
p2 <- qplot(cutWage, age, data = training, fill = cutWage, geom = c("boxplot","jitter"))
#grid.arrange(p1, p2, ncol=2)
p2
```



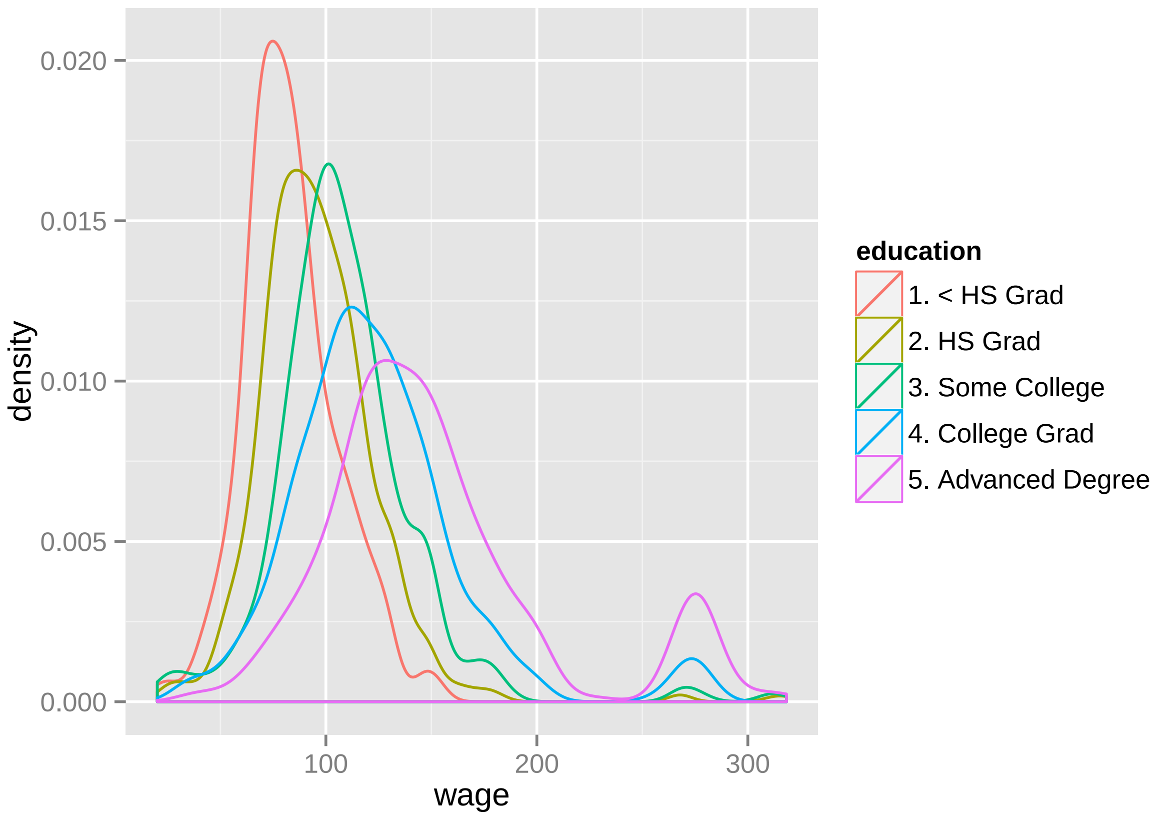

```
#One can make also tables
t1 <- table(cutWage,training$jobclass)
t2 <- table(cutWage,training$race)
t3 <- table(cutWage,training$education)
t1; t2; t3
```

```
##
## cutWage         1. Industrial 2. Information
##   [ 20.1, 92.7)           438            264
##   [ 92.7,118.9)           370            359
##   [118.9,318.3]           275            396

##
## cutWage         1. White 2. Black 3. Asian 4. Other
##   [ 20.1, 92.7)      558       90       39       15
##   [ 92.7,118.9)      617       70       36        6
##   [118.9,318.3]      565       42       59        5

##
## cutWage         1. < HS Grad 2. HS Grad 3. Some College 4. College Grad
##   [ 20.1, 92.7)          135        323             127              93
##   [ 92.7,118.9)           43        256             207             157
##   [118.9,318.3]           12         96             124             222
##
## cutWage         5. Advanced Degree
##   [ 20.1, 92.7)                 24
##   [ 92.7,118.9)                 66
##   [118.9,318.3]                217
```

```r
#One can also use prop.table to get the proportion on each group
prop.table(t2,1)
```

```
##
## cutWage              1. White    2. Black    3. Asian    4. Other
##   [ 20.1, 92.7) 0.794871795 0.128205128 0.055555556 0.021367521
##   [ 92.7,118.9) 0.846364883 0.096021948 0.049382716 0.008230453
##   [118.9,318.3] 0.842026826 0.062593145 0.087928465 0.007451565
```

```r
#Also, one can do Density Plots
qplot(wage, colour=education, data=training, geom="density")
```

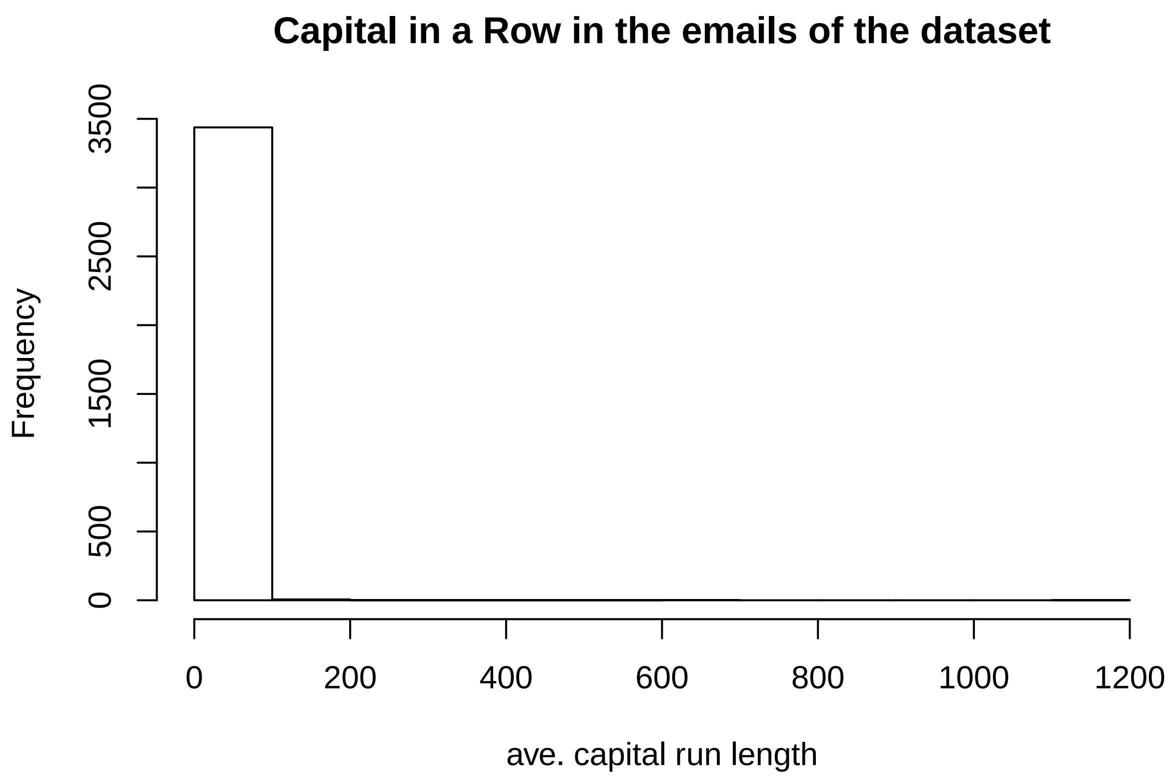


Preprocessing Predictor Values

```r
library(caret)
library(kernlab)
```

```
## Warning: package 'kernlab' was built under R version 3.1.3
```

```r
data(spam)
inTrain <- createDataPartition(y=spam$type, p=0.75, list=FALSE)
training <- spam[inTrain,]
testing <- spam[-inTrain,]
hist(training$capitalAve,main="Capital in a Row in the emails of the dataset",xlab="ave. capital run le
```




```
mean(training$capitalAve)
```

```
## [1] 4.930825
```

```
sd(training$capitalAve)
```

```
## [1] 29.42468
```

It can be observed that this variable is highly skewed. So it can be improved by preprocessing.

Preprocessing by Normalization (Standarization)

To standarize a variable one must substract the mean and divide the result by the SD of the variable:

```
trainCapAve <- training$capitalAve
trainCapAveS <- (trainCapAve - mean(trainCapAve))/sd(trainCapAve)
round(mean(trainCapAveS),4)
```

```
## [1] 0
```

```
round(sd(trainCapAveS),4)
```

```
## [1] 1
```

Also, the function **preProcess** can be used for standarization:

```
preObj <- preProcess(training[,-58],method=c("center","scale"))
trainCapAveS <- predict(preObj,training[,-58])$capitalAve
mean(trainCapAveS)
```

```
## [1] 5.552177e-19
```

```
sd(trainCapAveS)
```

```
## [1] 1
```

If the standarization is done in the test set, the mean and the SD of the training set must be use still. However, after standarized the test set variable the mean of the standarized variable will not be exactly zero neither the SD will be exactly one:

```
testCapAveS <- predict(preObj,testing[,-58])$capitalAve
mean(testCapAveS)
```

```
## [1] 0.03544596
```

```
sd(testCapAveS)
```

```
## [1] 1.285168
```

The **preProcess** function can be passed directly to the **train** function:

```
set.seed(32343)
modelFit <- train(type ~ ., data=training, preProcess=c("center","scale"),method="glm")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

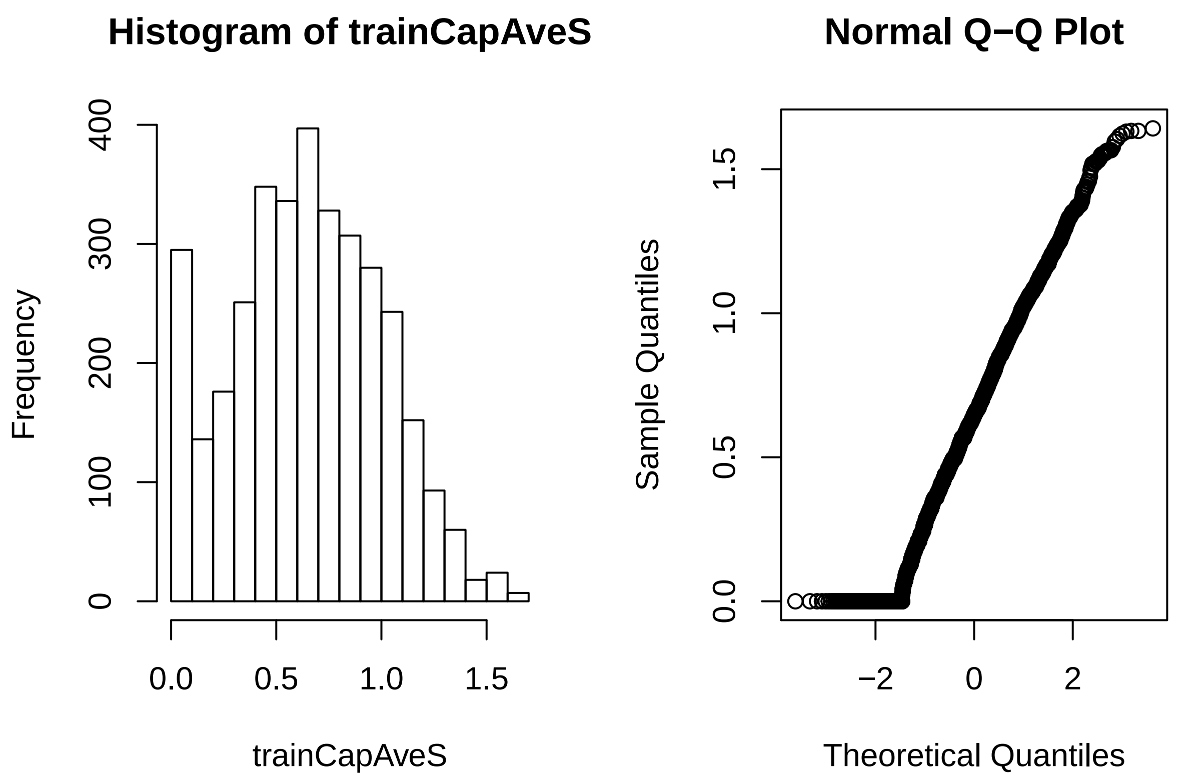
```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
modelFit
```

```
## Generalized Linear Model
##
## 3451 samples
##   57 predictor
##    2 classes: 'nonspam', 'spam'
##
## Pre-processing: centered (57), scaled (57)
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 3451, 3451, 3451, 3451, 3451, 3451, ...
## Resampling results
##
##   Accuracy   Kappa      Accuracy SD  Kappa SD
##   0.9256093  0.8437215  0.009910842  0.0195274
##
##
```

Other transformation available is the **BoxCox** transformation:

```r
preObj <- preProcess(training[,-58],method=c("BoxCox"))
trainCapAveS <- predict(preObj, training[,-58])$capitalAve
par(mfrow=c(1,2)); hist(trainCapAveS); qqnorm(trainCapAveS)
```



Preprocessing Imputing Missing Values

If the dataset has missing values, those can be imputing using **K-nearest neighbor's imputation** algorithm:

```r
#Make some values NA
training$capAve <- training$capitalAve
selectNA <- rbinom(dim(training)[1], size=1, prob=0.05)==1
training$capAve[selectNA] <- NA

#Impute and Standarize
preObj <- preProcess(training[,-58],method="knnImpute")
capAve <- predict(preObj, training[,-58])$capAve

#Standarize true values
capAveTruth <- training$capitalAve
capAveTruth <- (capAveTruth-mean(capAveTruth))/sd(capAveTruth)
```

Creating Covariates (or Features)

In case that one of the predictors is a factor variable, it is better to transform that variable into dummy variables. Prediction algoritms work better with dummy variables than with factor variables:

```r
library(ISLR); library(caret); data(Wage);
inTRain <- createDataPartition(y=Wage$wage, p=0.7, list=FALSE)
training <- Wage[inTrain,]; testing <- Wage[-inTrain,];

#converting the jobclass variable from a qualitative variable to a quantivative variable
#using dummyVars function from the caret package
table(training$jobclass)
```

```
##
##  1. Industrial 2. Information
##           1169           1069
```

```r
dummies <- dummyVars(wage ~ jobclass, data=training)
head(predict(dummies, newdata=training))
```

```
##        jobclass.1. Industrial jobclass.2. Information
## 231655                      1                       0
## 86582                       0                       1
## 161300                      1                       0
## 155159                      0                       1
## 11443                       0                       1
## 450601                      1                       0
```

Removing zero covariates

In order to detect those variables that has close to none variability, and therefore are not useful for prediction, one can use the **nearZerVar** function:

```r
nsv <- nearZeroVar(training, saveMetrics=TRUE)
nsv
```

```
##            freqRatio percentUnique zeroVar   nzv
## year        1.013441    0.20283976   FALSE FALSE
## age         1.101266    1.76760359   FALSE FALSE
## sex         0.000000    0.02897711    TRUE  TRUE
## maritl      3.134694    0.14488554   FALSE FALSE
## race        8.668224    0.11590843   FALSE FALSE
## education   1.328872    0.14488554   FALSE FALSE
## region      0.000000    0.02897711    TRUE  TRUE
## jobclass    1.093545    0.05795422   FALSE FALSE
## health      2.615509    0.05795422   FALSE FALSE
## health_ins  2.243478    0.05795422   FALSE FALSE
## logwage     1.189873   12.17038540   FALSE FALSE
## wage        1.189873   12.17038540   FALSE FALSE
```