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);
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

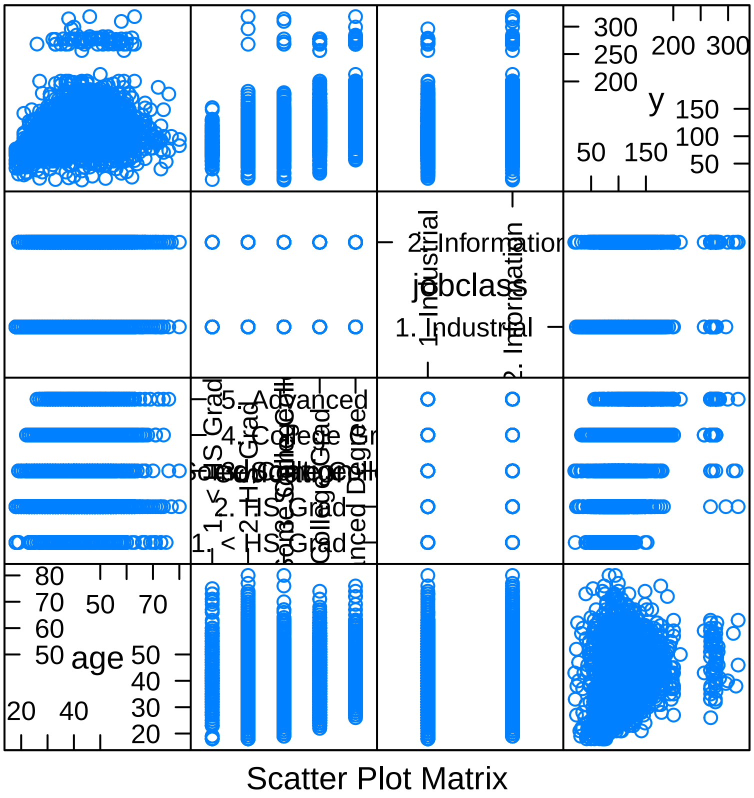
```r
inTrain <- createDataPartition(y = Wage$wage, p = 0.7, list = FALSE)
training <- Wage[inTrain,]
testing <- Wage[-inTrain,]
dim(training); dim(testing)
```

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

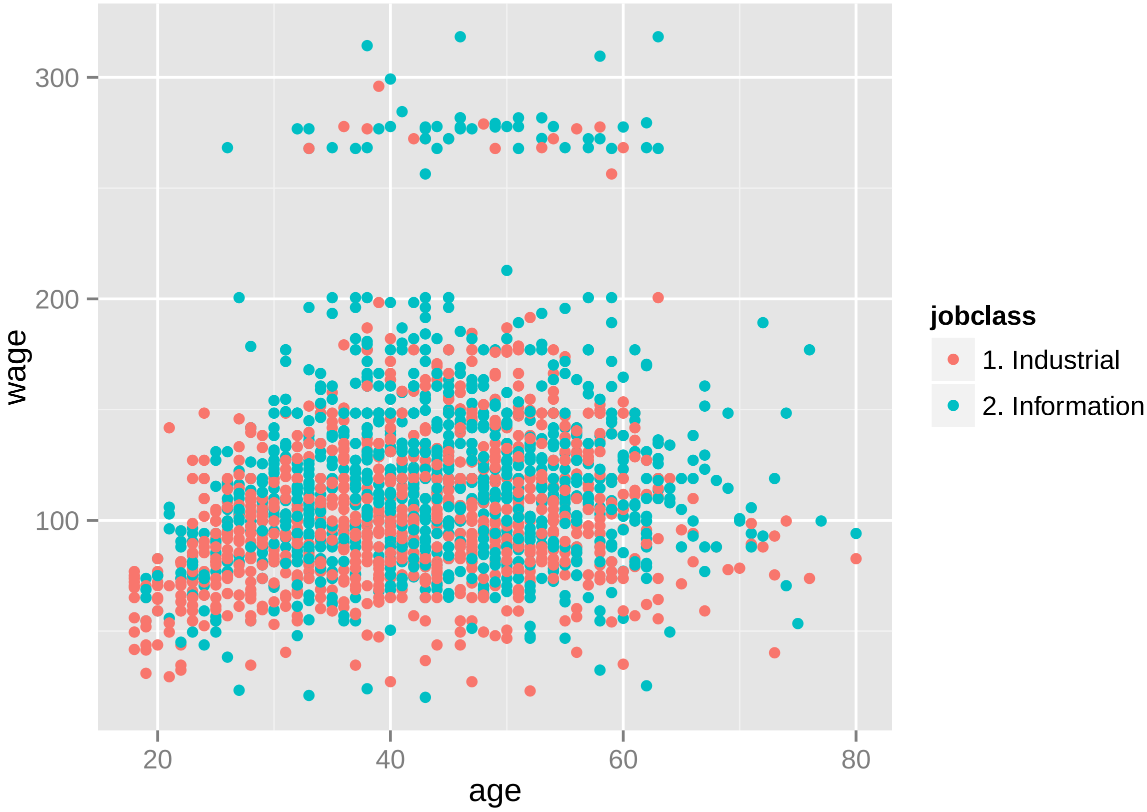
```
## [1] 898  12
```

Plotting Predictors vs Outcome

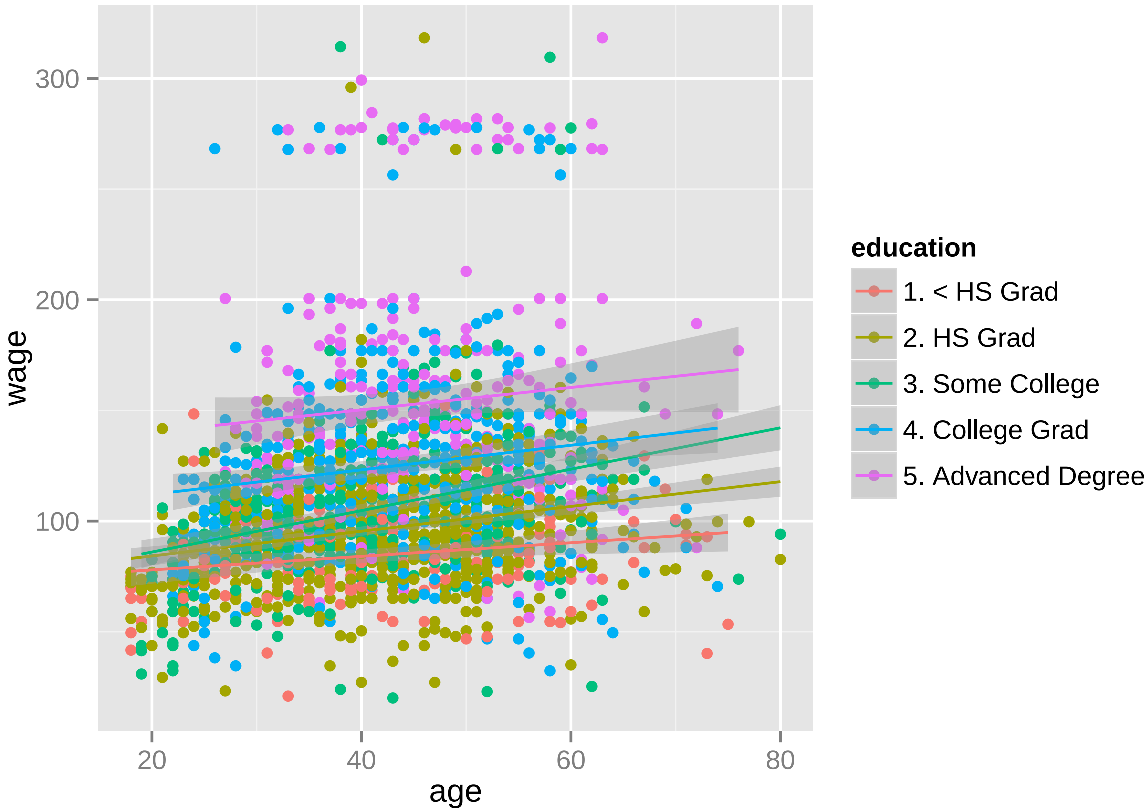
```r
#Plotting several predictors vs the outcome
featurePlot(x = training[,c("age", "education", "jobclass")], y = training$wage, plot="pairs")
```




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



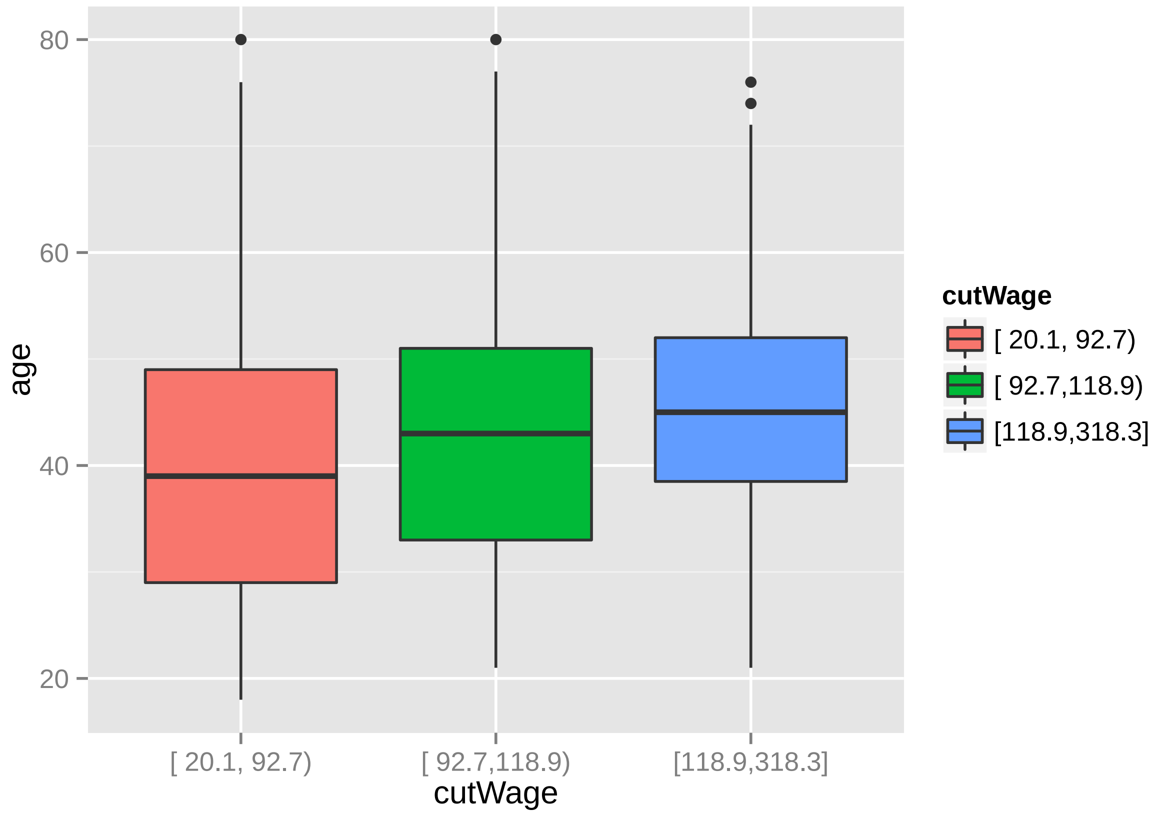
```r
#Add regression smoothers
qq <- qplot(age, wage, colour=education, data=training)
qq + geom_smooth(method="lm", formula = y ~ x)
```



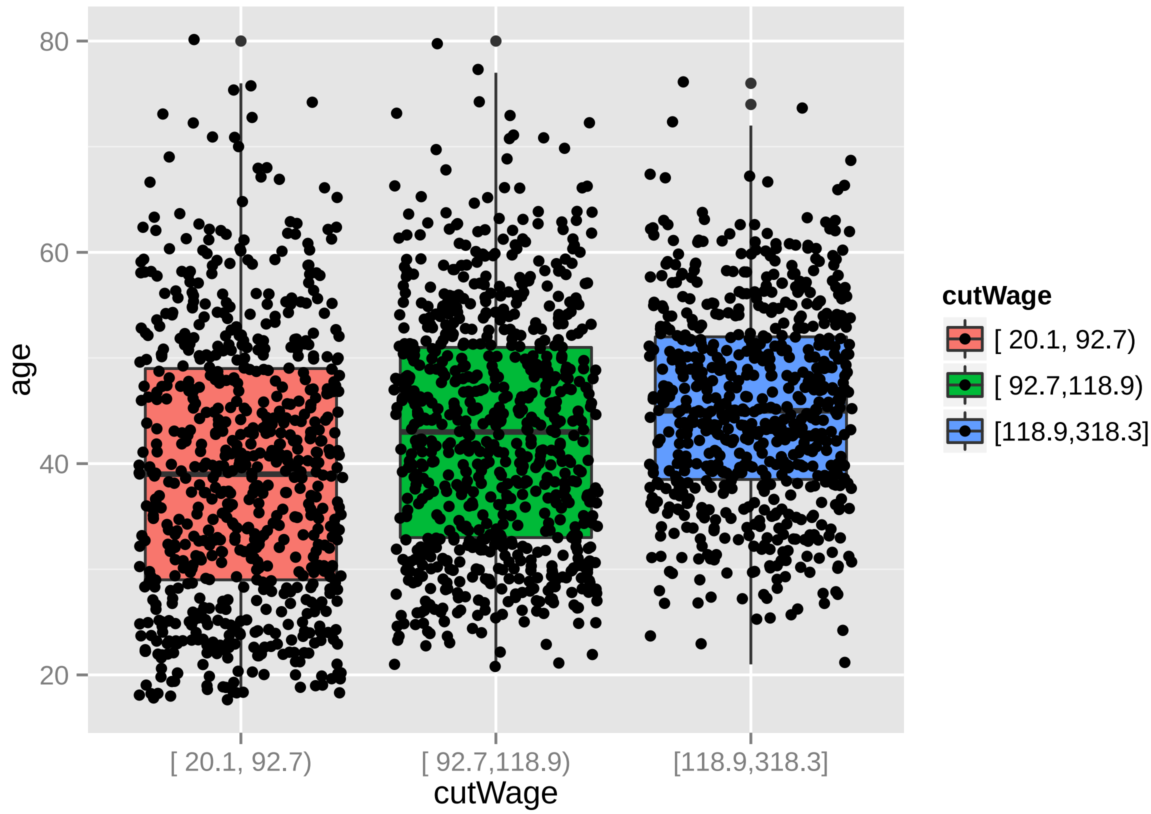

```r
#cut2, making factors (Hmisc package)
require(Hmisc)
#Splitting the wage variable into groups of quantiles
cutWage <- cut2(training$wage, g=3)
table(cutWage)
```

```
## cutWage
## [ 20.1, 92.7) [ 92.7,118.9) [118.9,318.3]
##           701           734           667
```

```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
```



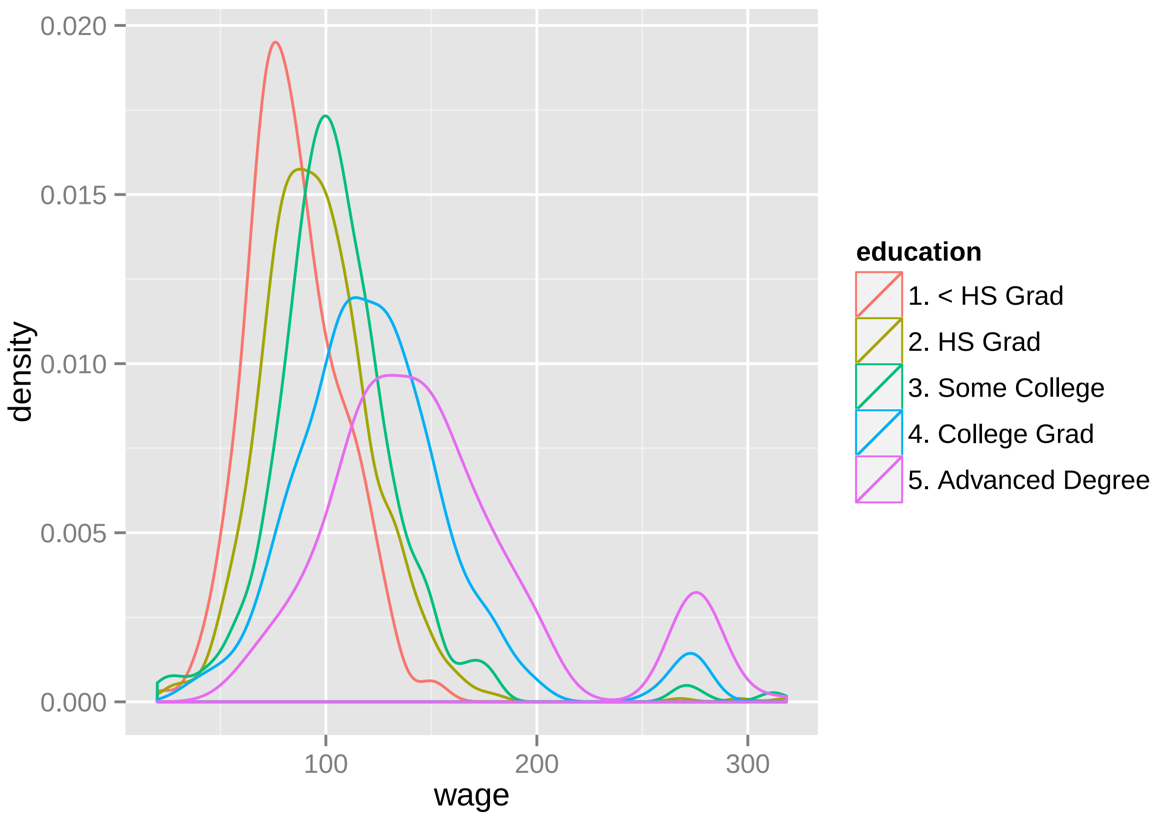

```
#One can make also tables
t1 <- table(cutWage,training$jobclass)
t2 <- table(cutWage,training$race)
t3 <- table(cutWage,training$education)
t1; t2; t3
```

```
##
## cutWage         1. Industrial 2. Information
##   [ 20.1, 92.7)           436            265
##   [ 92.7,118.9)           372            362
##   [118.9,318.3]           256            411

##
## cutWage         1. White 2. Black 3. Asian 4. Other
##   [ 20.1, 92.7)      551       95       38       17
##   [ 92.7,118.9)      613       78       38        5
##   [118.9,318.3]      564       39       59        5

##
## cutWage         1. < HS Grad 2. HS Grad 3. Some College 4. College Grad
##   [ 20.1, 92.7)          124        319             140              93
##   [ 92.7,118.9)           51        264             202             154
##   [118.9,318.3]           11        106             103             238
##
## cutWage         5. Advanced Degree
##   [ 20.1, 92.7)                 25
##   [ 92.7,118.9)                 63
##   [118.9,318.3]                209
```

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

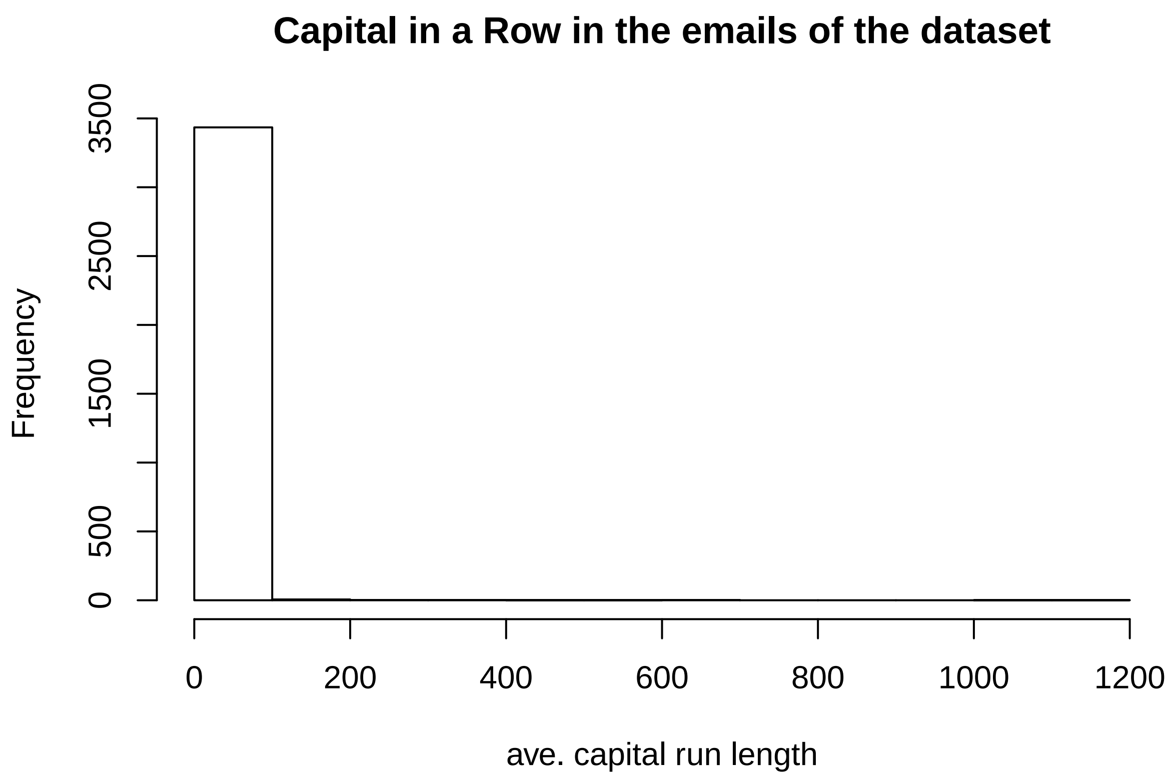
```
##
## cutWage              1. White    2. Black    3. Asian    4. Other
##   [ 20.1, 92.7) 0.786019971 0.135520685 0.054208274 0.024251070
##   [ 92.7,118.9) 0.835149864 0.106267030 0.051771117 0.006811989
##   [118.9,318.3] 0.845577211 0.058470765 0.088455772 0.007496252
```

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



Preprocessing Predictor Values

```
library(caret)
library(kernlab)
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 len
```




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

```
## [1] 5.306681
```

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

```
## [1] 34.87303
```

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:

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

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

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

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

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

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

```
## [1] 7.035144e-18
```

```r
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:

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

```
## [1] -0.0132126
```

```r
sd(testCapAveS)
```

```
## [1] 0.5581167
```

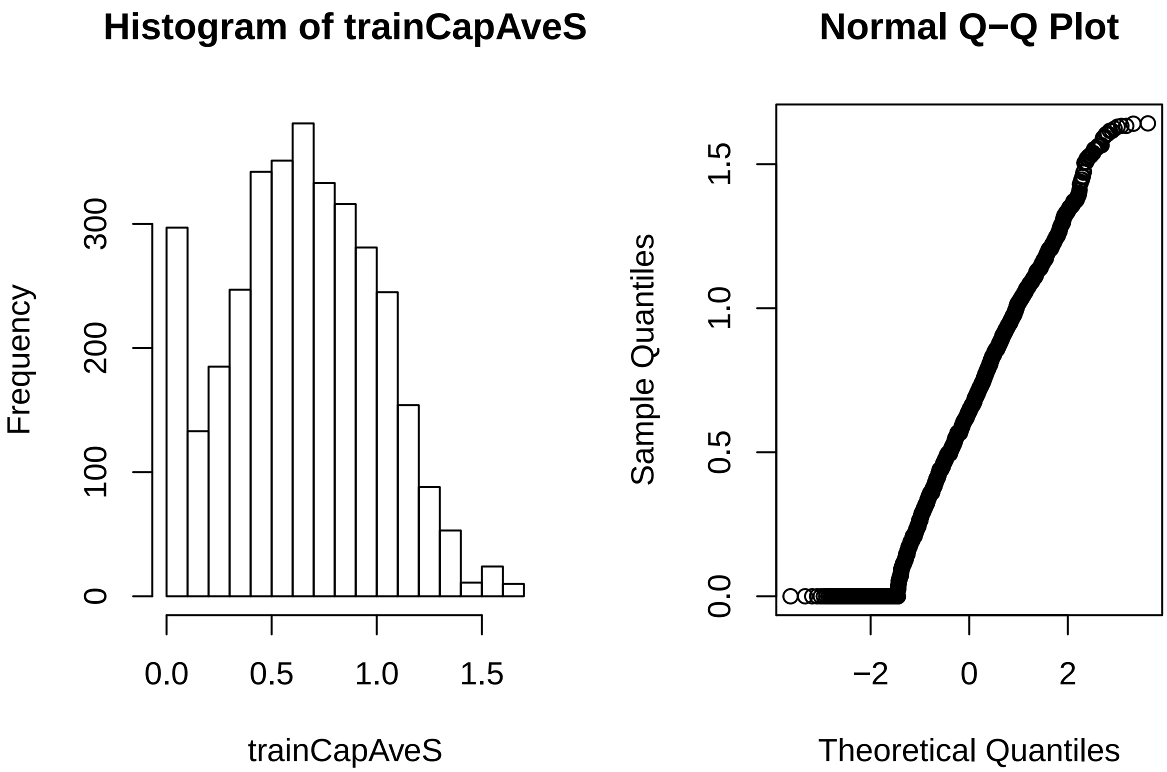
The **preProcess** function can be passed directly to the **train** function:

```r
set.seed(32343)
modelFit <- train(type ~ ., data=training, preProcess=c("center","scale"),method="glm")
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
##   0.916788  0.8246351  0.008329611  0.01718861
##
##
```

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

```
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:

```
#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
##           1128           1103
```

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

```
##        jobclass.1. Industrial jobclass.2. Information
## 231655                      1                       0
## 86582                       0                       1
## 155159                      0                       1
## 11443                       0                       1
## 376662                      0                       1
## 450601                      1                       0
```

Removing zero covariates

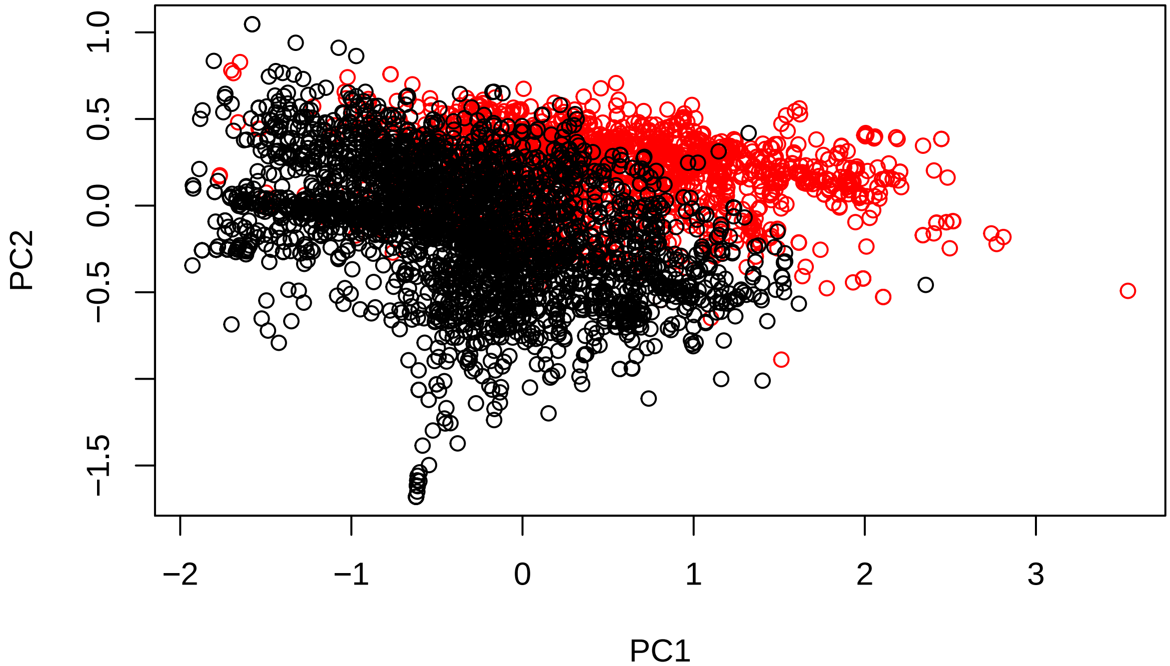
In order to detect those variables that has close to none variability, and therefore are not useful for prediction, one can use the **nearZerVar** function:

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

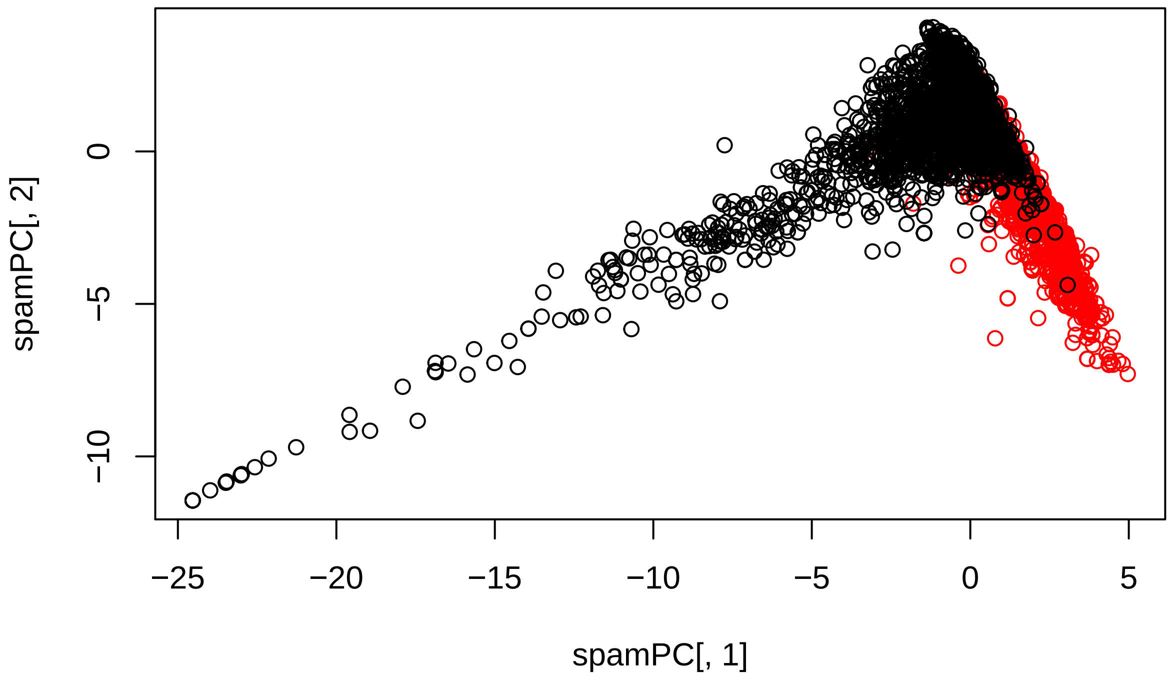
```
##            freqRatio percentUnique zeroVar   nzv
## year        1.066298    0.20283976   FALSE FALSE
## age         1.000000    1.76760359   FALSE FALSE
## sex         0.000000    0.02897711    TRUE  TRUE
## maritl      3.232704    0.14488554   FALSE FALSE
## race        8.578704    0.11590843   FALSE FALSE
## education   1.436508    0.14488554   FALSE FALSE
## region      0.000000    0.02897711    TRUE  TRUE
## jobclass    1.022665    0.05795422   FALSE FALSE
## health      2.502355    0.05795422   FALSE FALSE
## health_ins  2.200861    0.05795422   FALSE FALSE
## logwage     1.022472   11.99652275   FALSE FALSE
## wage        1.022472   11.99652275   FALSE FALSE
```

Principal Componen Analysis (PCA)

```r
library(caret)
library(kernlab)
data(spam)
inTrain <- createDataPartition(y=spam$type, p=0.75, list=FALSE)
training <- spam[inTrain,]
testing <- spam[-inTrain,]

#PCA using the basic pacakge funtion prcomp
typeColor <- ((spam$type=="spam")*1 + 1) #clasifies each point for coloring as spam or ham
prComp <- prcomp(log10(spam[,-58]+1)) #the log10 is to make the variables look more "normal"
plot(prComp$x[,1],prComp$x[,2],col=typeColor,xlab="PC1",ylab="PC2")
```



```r
#PCA using the caret package
preProc <- preProcess(log10(spam[,-58]+1),method="pca",pcaComp=2)
spamPC <- predict(preProc,log10(spam[,-58]+1))
plot(spamPC[,1],spamPC[,2],col=typeColor)
```




```
#One can fit a model with the Training set and the principal componets
preProc <- preProcess(log10(training[,-58]+1),method="pca",pcaComp=2)
trainPC <- predict(preProc,log10(training[,-58]+1))
modelFit <- train(training$type ~ .,method="glm",data=trainPC)
modelFit
```

```
## Generalized Linear Model
##
## 3451 samples
##    1 predictor
##    2 classes: 'nonspam', 'spam'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 3451, 3451, 3451, 3451, 3451, 3451, ...
## Resampling results
##
##   Accuracy   Kappa      Accuracy SD  Kappa SD
##   0.9011037  0.7912827  0.008097626  0.0169247
##
##
```

```
confusionMatrix(training$type,predict(modelFit,trainPC))
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction nonspam spam
##    nonspam    1958  133
##    spam        205 1155
##
```

```
##                Accuracy : 0.9021
##                  95% CI : (0.8917, 0.9118)
##     No Information Rate : 0.6268
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.793
##  Mcnemar's Test P-Value : 0.0001125
##
##             Sensitivity : 0.9052
##             Specificity : 0.8967
##          Pos Pred Value : 0.9364
##          Neg Pred Value : 0.8493
##              Prevalence : 0.6268
##          Detection Rate : 0.5674
##    Detection Prevalence : 0.6059
##       Balanced Accuracy : 0.9010
##
##        'Positive' Class : nonspam
##
```

```
#Now in the test dataset
testPC <- predict(preProc,log10(testing[,-58]+1)) #one must use the same preProc obj calculated for the
confusionMatrix(testing$type,predict(modelFit,testPC)) #one must also use the same model fitted for the
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction nonspam spam
##    nonspam     649   48
##    spam         67  386
##
##                Accuracy : 0.9
##                  95% CI : (0.8812, 0.9167)
##     No Information Rate : 0.6226
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.789
##  Mcnemar's Test P-Value : 0.09325
##
##             Sensitivity : 0.9064
##             Specificity : 0.8894
##          Pos Pred Value : 0.9311
##          Neg Pred Value : 0.8521
##              Prevalence : 0.6226
##          Detection Rate : 0.5643
##    Detection Prevalence : 0.6061
##       Balanced Accuracy : 0.8979
##
##        'Positive' Class : nonspam
##
```

```
#Another way of train the model and use the PCA in the preprocessing at the same time:
modelFit <- train(training$type ~ .,method="glm",preProcess="pca",data=training)
confusionMatrix(training$type,predict(modelFit,training))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction nonspam spam
##    nonspam    1988  103
##    spam        147 1213
##
##                Accuracy : 0.9276
##                  95% CI : (0.9184, 0.936)
##     No Information Rate : 0.6187
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8474
##  Mcnemar's Test P-Value : 0.006537
##
##             Sensitivity : 0.9311
##             Specificity : 0.9217
##          Pos Pred Value : 0.9507
##          Neg Pred Value : 0.8919
##              Prevalence : 0.6187
##          Detection Rate : 0.5761
##    Detection Prevalence : 0.6059
##       Balanced Accuracy : 0.9264
##
##        'Positive' Class : nonspam
##
```

```r
confusionMatrix(testing$type,predict(modelFit,testing))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction nonspam spam
##    nonspam     661   36
##    spam         62  391
##
##                Accuracy : 0.9148
##                  95% CI : (0.8971, 0.9303)
##     No Information Rate : 0.6287
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.8197
##  Mcnemar's Test P-Value : 0.01156
##
##             Sensitivity : 0.9142
##             Specificity : 0.9157
##          Pos Pred Value : 0.9484
##          Neg Pred Value : 0.8631
##              Prevalence : 0.6287
##          Detection Rate : 0.5748
##    Detection Prevalence : 0.6061
##       Balanced Accuracy : 0.9150
##
##        'Positive' Class : nonspam
```

```
##
```

Predicting with Regression Models

Linear Models using the caret package:

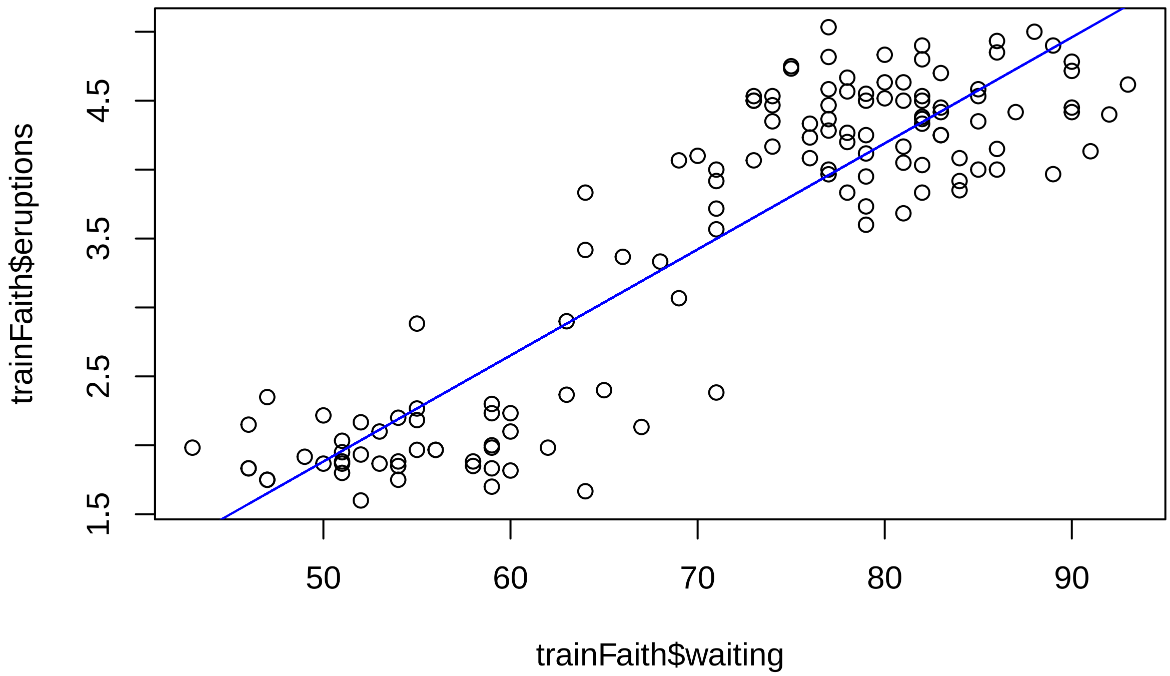
```
library(MASS)
data(faithful); set.seed(333)
inTrain <- createDataPartition(y=faithful$eruptions, p=0.5, list=FALSE)
trainFaith <- faithful[inTrain,]; testFaith <- faithful[-inTrain,]
head(trainFaith)
```

```
##    eruptions waiting
## 1      3.600      79
## 5      4.533      85
## 6      2.883      55
## 9      1.950      51
## 10     4.350      85
## 12     3.917      84
```

```
modFit <- train(eruptions ~ waiting,data=trainFaith,method="lm")
summary(modFit$finalModel)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.29250 -0.37706  0.00889  0.37091  1.07341
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.964020   0.233116  -8.425 4.93e-14 ***
## waiting      0.076930   0.003232  23.806  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5004 on 134 degrees of freedom
## Multiple R-squared:  0.8088, Adjusted R-squared:  0.8073
## F-statistic: 566.7 on 1 and 134 DF,  p-value: < 2.2e-16
```

```
plot(trainFaith$waiting,trainFaith$eruptions)
lines(trainFaith$waiting,modFit$finalModel$fitted,col="blue")
```



Predicting with Regression Models with Multiple variable

```
library(ISLR); library(ggplot2); library(caret)
data(Wage); Wage <- subset(Wage,select=-c(logwage))
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
##  1. Industrial :1544   1. <=Good     : 858   1. Yes:2083
##  2. Information:1456   2. >=Very Good:2142   2. No : 917
##
##
##
##
```

```
##
##       wage
##  Min.   : 20.09
##  1st Qu.: 85.38
##  Median :104.92
##  Mean   :111.70
##  3rd Qu.:128.68
##  Max.   :318.34
##
```

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

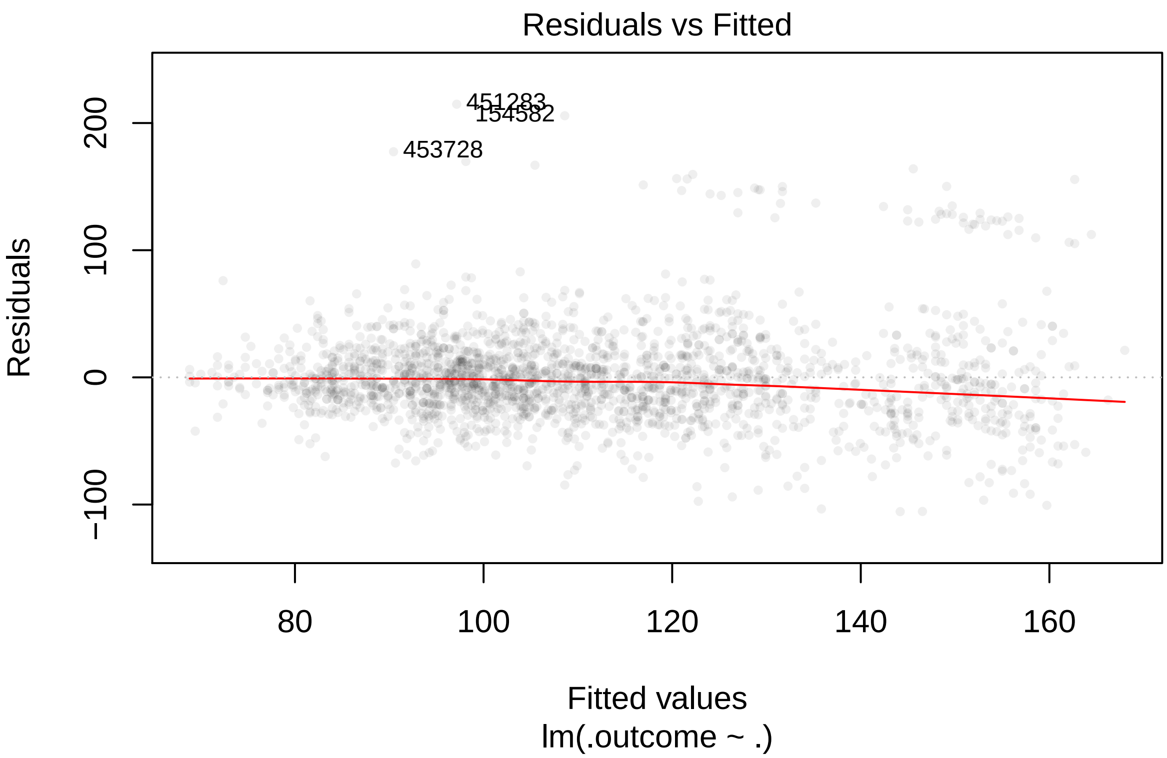
```
## [1] 2102   11
```

```
## [1] 898  11
```

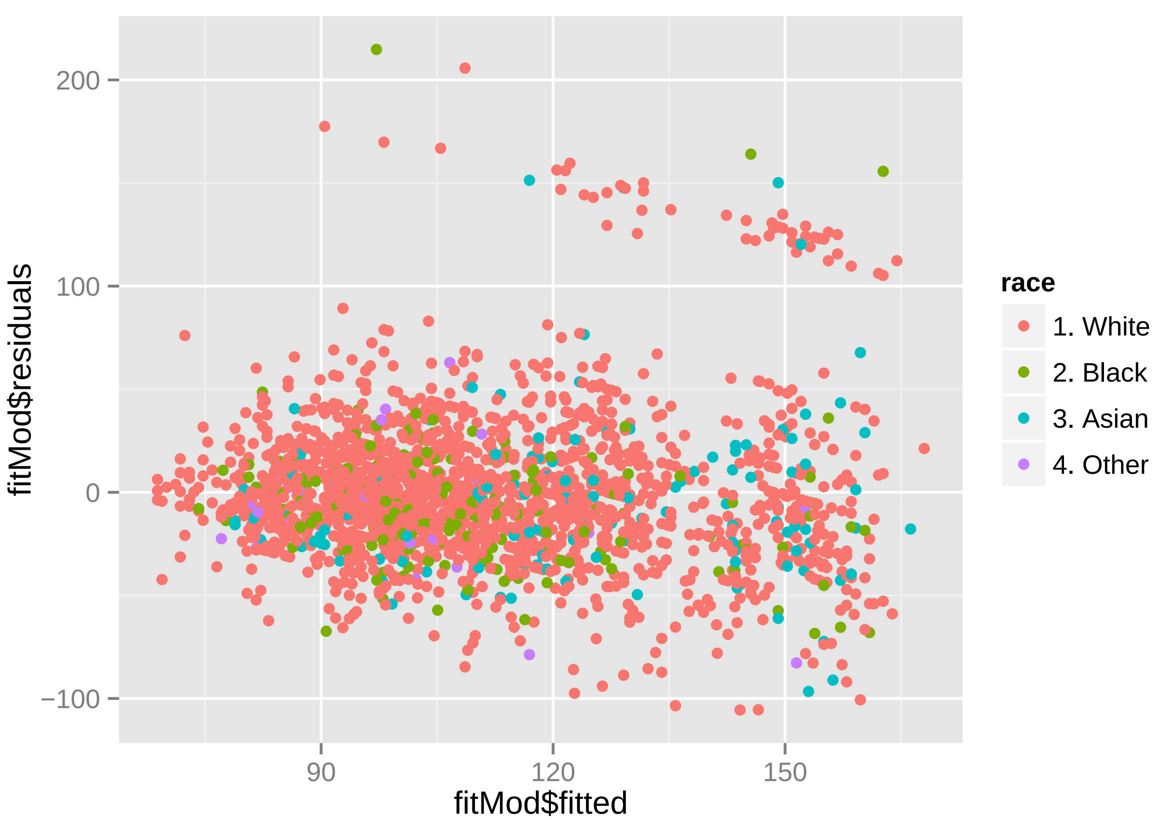
```r
#fitting a linear model
modFit <- train(wage ~ age + jobclass + education, method="lm", data=training) #we fit the model on the

fitMod <- modFit$finalModel
print(modFit)
```

```
## Linear Regression
##
## 2102 samples
##   10 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 2102, 2102, 2102, 2102, 2102, 2102, ...
## Resampling results
##
##   RMSE      Rsquared   RMSE SD  Rsquared SD
##   35.60168  0.2679452  1.19091  0.01751045
##
##
```

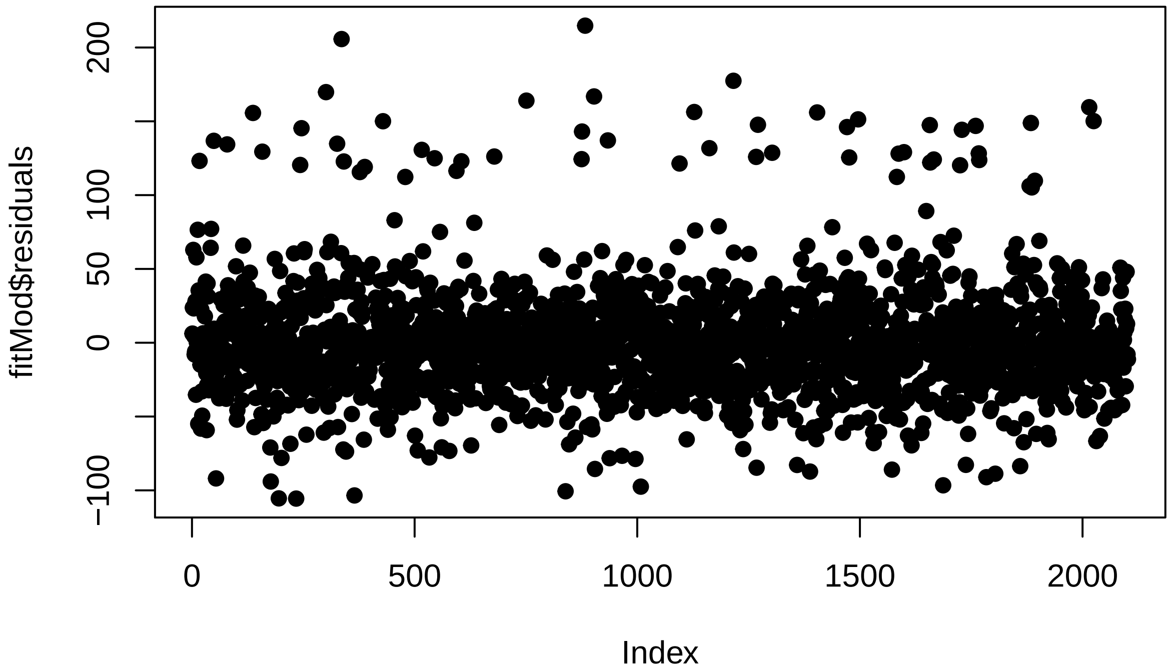
```r
#Plotting the fitted values vs the residuals
plot(fitMod,1,pch=19,cex=0.5,col="#00000010")
```




```r
#You can see there are some outliers in the residuals
#Let't plot fitted vs residuals and add the race variable and see if that variable explains the outliers
qplot(fitMod$fitted, fitMod$residuals, colour=race, data=training)
```



```r
#You see that the outliers might be explained by the race variable
#also plot the residuals accross the dataset and see if there is a patron. The residual should be rando
plot(fitMod$residuals,pch=19)
```




```r
#Predicted versus true values in the test set
pred <- predict(modFit, testing)
qplot(wage, pred, colour=year,data=testing)
```


150
120
90
pred
100
200
300
wage
year
2009
2008
2007
2006
2005
2004
2003