Practical Machine Learning - Course Project

Data

First of all, the main data are read from the C drive and the first column with the number of observation is deleted.

```
RawData<-read.csv("C:/=Raw Data=/Practical Machine Learning Course Project/pml-training.csv",heade
r=TRUE)
RawData<-RawData[,-1] #deleting number of observation (first column)
FinalTestData<-read.csv("C:/=Raw Data=/Practical Machine Learning Course Project/pml-testing.cs
v",header=TRUE)
FinalTestData<-FinalTestData[,-1] #deleting number of observation (first column)
n<-dim(RawData)[1]
```

The raw data set has dimensions 19622, 159. The raw data set should be incomplete, have missing values or not numerical values. The following code simply creates data set only with numerical values.

```
#numeric values
  isNumber<-sapply(RawData,function(x) {sum(is.numeric(x) & !is.na(x))})
  isNumber<-as.logical(isNumber==n)

#data with numeric values, displaysing their means and names
  CompleteData<-RawData[,isNumber]
  outputVariable<-RawData$classes
  apply(CompleteData,2,mean)
  apply(CompleteData,2, var)
  names(CompleteData)

#the same procedure with data for final testing
  FinalTestCompleteData<-FinalTestData[,isNumber]
  apply(FinalTestCompleteData,2,mean)
  apply(FinalTestCompleteData,2, var)
  names(FinalTestCompleteData)

  sum(names(FinalTestCompleteData)!=names(CompleteData))

#output variable for CompleteData
  y<-RawData$classe
```

The dimension of complete data set is 19622, 55. This is a signifficant decrease of attributes.

Basic Approach

The “fast and tidy” approach as a first step of the data analysis is usually recommended. This means try some fast, simple and straightforward way to analyze the data, evaluate it and find out how well/bad this approach is. Next, the improvements can be done based on those findings.

The PCA with knn classification and random forest were choosen as the “fast and tidy” approach on the complete data. The reasons are following.

- The number of attributes 55 is still hight. PCA can reduce it.
- There is a high number of observations 19622. Even if we perform cross validation, the number will be still hight. That is why knn-classification can perform well.
- Random forests are different kind of classifier, simple and effective.

Data Splitting

First of all, it is necessary to split the data into training set and testing set.

```
library(caret)
```

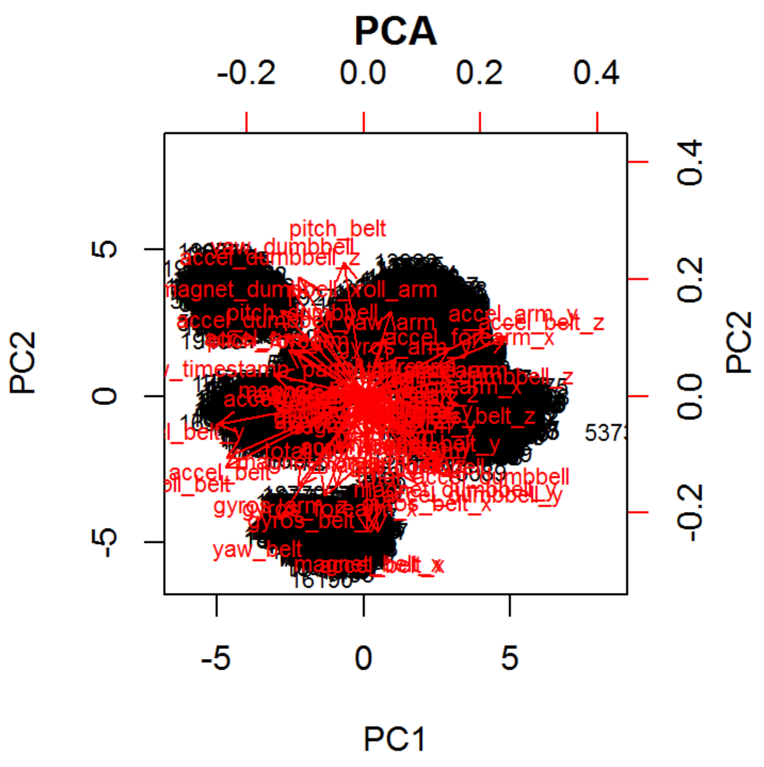
```
## Loading required package: lattice
## Loading required package: ggplot2
```

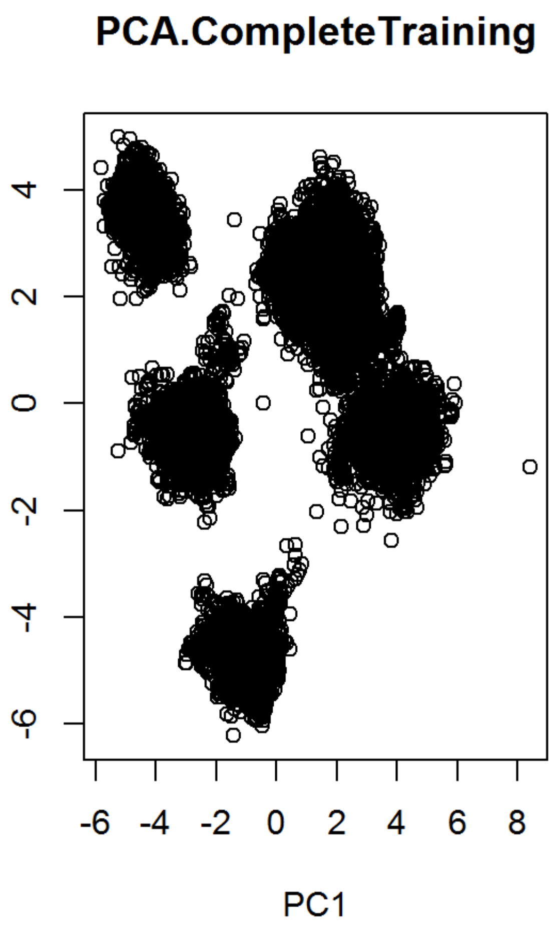
```
library(stats)
set.seed(10201)
inTrain<-createDataPartition(y=y,p=0.6,list=FALSE)
#data splitting
  CompleteTrainingData<-CompleteData[inTrain,]
  CompleteTestingData<-CompleteData[-inTrain,]
  yTraining<-y[inTrain]
  yTesting<-y[-inTrain]
#data standardisation
  CompleteTrainingData<-scale(CompleteTrainingData, center = TRUE, scale = TRUE)
  apply(CompleteTrainingData,2,mean)
  apply(CompleteTrainingData,2,sd)

  CompleteTestingData<-scale(CompleteTestingData, center = TRUE, scale = TRUE)

  FinalTestCompleteData<-scale(FinalTestCompleteData, center = TRUE, scale = TRUE)
```

PCA

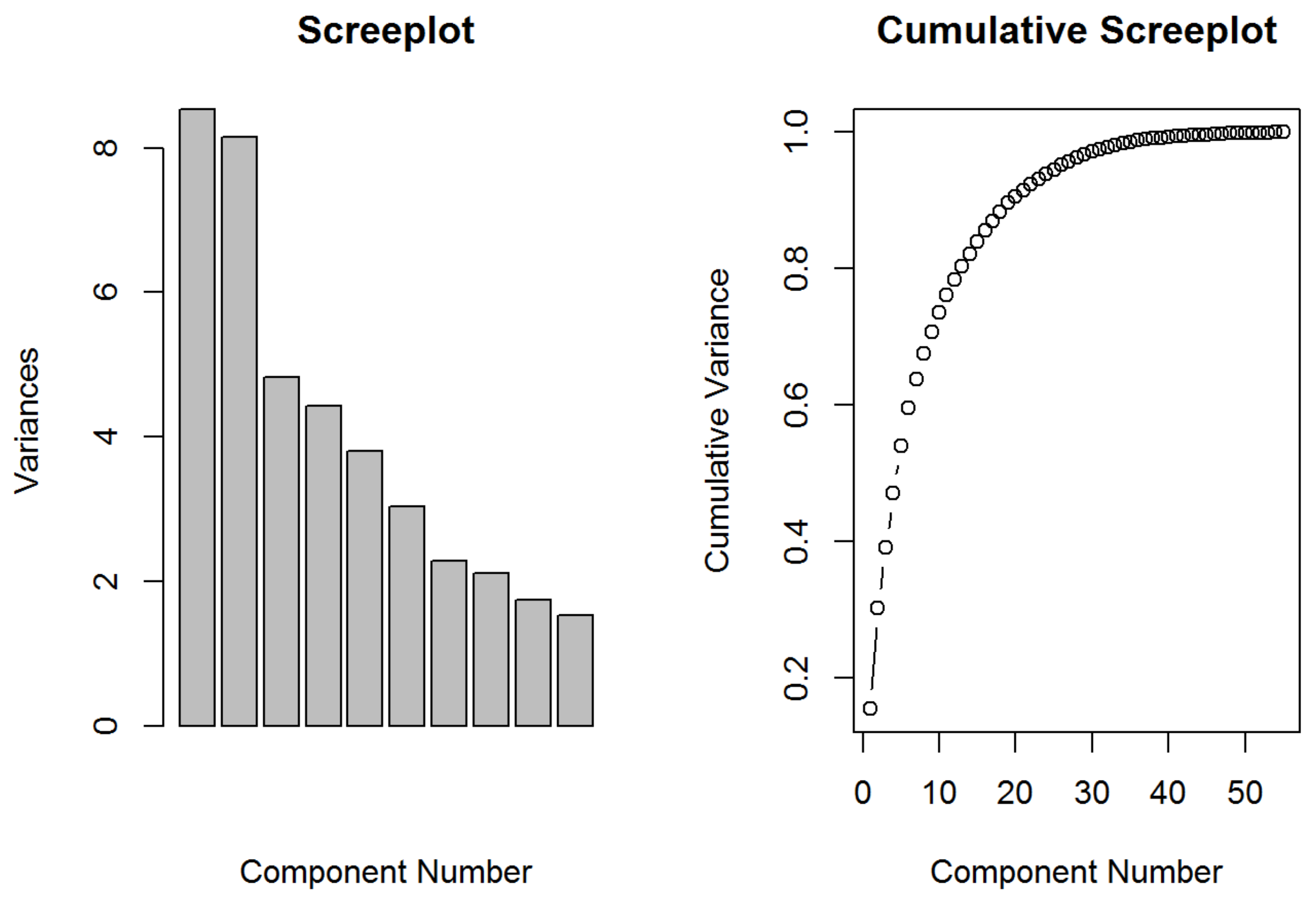
Then, PCA is performed on the trining set and biplot and screeplot are created.

```
PCA.CompleteTraining=prcomp(CompleteTrainingData, scale=FALSE)

par(mfrow=c(1,2))
biplot(PCA.CompleteTraining, scale=0,cex=.7,main="PCA") #zmÄ›na fontu
plot(PCA.CompleteTraining$x[,1],PCA.CompleteTraining$x[,2],main="PCA.CompleteTraining",xlab="PC
1",ylab="PC2")
```





```
vars<-apply(PCA.CompleteTraining$x,2,var)
props<-vars/sum(vars)

par(mfrow=c(1,2))
plot(PCA.CompleteTraining,main="Screeplot",xlab="Component Number")
plot(cumsum(props),type="b",main="Cumulative Screeplot",xlab="Component Number",ylab="Cumulative V
ariance")
```



```
par(mfrow=c(1,1))
```

K nearest neighbour

The K nearest neighbour classifier is performed on the data with two values of k, 5 and 7. The code for second one is hidden.For each value of k, the out-of-bag error on training data and classes for final data set are calculated.

```
#preparing data
  nOfComp<-25
  Dtrain<-PCA.CompleteTraining$x[,1:nOfComp]
  Dtest<-CompleteTestingData%*%PCA.CompleteTraining$rotation
  Dtest<-Dtest[,1:nOfComp]

  DFinalTest<-FinalTestCompleteData%*%PCA.CompleteTraining$rotation
  DFinalTest<-DFinalTest[,1:nOfComp]

#learning
  require(class)
```

```
## Loading required package: class
```

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

```
  library(class)
  k=5
  PCA.knn<-knn(Dtrain,Dtest,yTraining,k=k)
#out-of-bag error - test data
  KnnErr5<-1-sum(diag(table(PCA.knn,yTesting)))/sum(sum(table(PCA.knn,yTesting)))

#Final Test Prediction
  FinalKnn5<-knn(Dtrain,DFinalTest,yTraining,k=k)
```

Random Forests

Finally, random forests are performed.

```
#preparing data
  nOfComp<-25
  Dtrain<-PCA.CompleteTraining$x[,1:nOfComp]
  Dtest<-CompleteTestingData%*%PCA.CompleteTraining$rotation
  Dtest<-Dtest[,1:nOfComp]

  DFinalTest<-FinalTestCompleteData%*%PCA.CompleteTraining$rotation
  DFinalTest<-DFinalTest[,1:nOfComp]

#learning
  require(randomForest)
```

```
## Loading required package: randomForest
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
  PCA.randomForest=randomForest(x=Dtrain,y=yTraining)

#out-of-bag error - test data
  randomForestErr<-1-sum(diag(table(predict(PCA.randomForest,Dtest),yTesting)))/sum(sum(table(pred
ict(PCA.randomForest,Dtest),yTesting)))

#Final Test Prediction
  FinalRandomForest<-predict(PCA.randomForest,DFinalTest)
```

Results

The out-of-bag error was 0.0453734 for knn with k=5, 0.0579913 for knn with k=7 and 0.0307163 for random forests. The predictions are following.

```
t(matrix(c(paste(FinalKnn5),paste(FinalKnn7),paste(FinalRandomForest)),ncol=3,nrow=20))
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
## [1,] "B"  "A"  "C"  "A"  "A"  "E"  "D"  "B"  "A"  "A"   "B"   "C"   "B"  
## [2,] "B"  "A"  "A"  "A"  "A"  "E"  "D"  "D"  "A"  "A"   "B"   "C"   "B"  
## [3,] "B"  "A"  "C"  "A"  "A"  "D"  "D"  "D"  "A"  "A"   "B"   "C"   "B"  
##      [,14] [,15] [,16] [,17] [,18] [,19] [,20]
## [1,] "A"   "E"   "E"   "A"   "B"   "B"   "B"  
## [2,] "A"   "E"   "E"   "A"   "B"   "B"   "B"  
## [3,] "A"   "E"   "E"   "A"   "B"   "B"   "B"
```

The true values are B A B A A E D B A A B C B A E E A B B B.

The majority vote works well for 18 of 20 observation, it fails only for observations number 3 and 8. In this cases, there is needed some improvement.

This analysis shows, that even very simple approach can lead to reasonable results.