Activity Analysis

Sameer

Sunday, August 24, 2014

Executive Summary

The analysis is about classifying activities based on a number of factors. For the classification problem we will compare three algorithms: 1. Random Forest 2. Random Forest with Bagging 3. Random Forest after Dimension Rediuction

Dimension Reduction with Principal Component Analysis, dramatically reduces the complexity and reduces the execution time. However; the algorith compromises on the accuracy and interpretability.

1. Data used for training and testing - Training
2. Data for Prediction - Validation

Loading the Libraries

First, we will load all the required libraries needed. We would use **caret** and **party** package for the analysis.

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.1.1
```

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

```r
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.1.1
```

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

```r
library(party)
```

```
## Warning: package 'party' was built under R version 3.1.1
```

```
## Loading required package: grid
## Loading required package: zoo
##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
##
## Loading required package: sandwich
```

```
## Warning: package 'sandwich' was built under R version 3.1.1
```

```
## Loading required package: strucchange
```

```
## Warning: package 'strucchange' was built under R version 3.1.1
```

```
## Loading required package: modeltools
```

```
## Warning: package 'modeltools' was built under R version 3.1.1
```

```
## Loading required package: stats4
```

Importing the Datasets

```
PML_Training<-read.csv("pml_training.csv",header=T)

PML_Testing<-read.csv("pml_testing.csv",header=T)
```

Data Preprocessing

Through some preliminary analysis, we would first select the columns which are expected to influence the prediction.

We would store the index of these columns in a new variable **columns**

```
columns<-which(names(PML_Training) %in% c("classe",
                                          "num_window",
                                          "roll_belt",
                                          "pitch_belt",
                                          "yaw_belt",
                                          "total_accel_belt",
                                          "gyros_belt_x",
                                          "gyros_belt_y",
                                          "gyros_belt_z",
                                          "accel_belt_x",
                                          "accel_belt_y",
                                          "accel_belt_z",
                                          "magnet_belt_x",
                                          "magnet_belt_y",
                                          "magnet_belt_z",
                                          "roll_arm",
                                          "pitch_arm",
                                          "yaw_arm",
                                          "total_accel_arm",
                                          "gyros_arm_x",
                                          "gyros_arm_y",
                                          "gyros_arm_z",
                                          "accel_arm_x",
                                          "accel_arm_y",
                                          "accel_arm_z",
```

```
                                                  "magnet_arm_x",
                                                  "magnet_arm_y",
                                                  "magnet_arm_z",
                                                  "roll_dumbbell",
                                                  "pitch_dumbbell",
                                                  "yaw_dumbbell"))
```

The columns use for the algorith are as follows: 7, 8, 9, 10, 11, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 60, 61, 62, 63, 64, 65, 66, 67, 68, 84, 85, 86, 160

We will then subset the training set and partition it into training and testing sets.

```
PML_Training_Subset<-PML_Training[,columns]
```

Training set will contain 75% of the records and testing set will contain 25% of the records.

```
inTrain<-createDataPartition(y=PML_Training_Subset$classe,p=0.75,list=FALSE)
training<-PML_Training_Subset[inTrain,]
testing<-PML_Training_Subset[-inTrain,]
```

Simple Random Forrest

Step 1: Training The Model We will first train the model on the training set using all the predictors. The output variable is **classe**

```
r2 = randomForest(classe ~., data=training, importance=TRUE, do.trace=100)
```

```
## ntree      OOB      1      2      3      4      5
##   100:   0.37%  0.12%  0.21%  0.93%  0.41%  0.33%
##   200:   0.31%  0.14%  0.14%  0.70%  0.41%  0.30%
##   300:   0.30%  0.14%  0.18%  0.55%  0.50%  0.26%
##   400:   0.28%  0.14%  0.11%  0.55%  0.46%  0.26%
##   500:   0.28%  0.14%  0.07%  0.55%  0.46%  0.30%
```

Step 2: FOt the Model on the Training Set:

```
pred2<-predict(r2,testing[,-31])
```

Step 3: Building the Confusion Matrix to asses the accuracy of the Model

```
r2
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = training, importance = TRUE,      do.trace = 100)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 0.28%
```

```
## Confusion matrix:
##      A    B    C    D    E class.error
## A 4179    0    1    5    0   0.0014337
## B    0 2846    1    1    0   0.0007022
## C    0   11 2553    3    0   0.0054538
## D    0    0   11 2401    0   0.0045605
## E    0    1    1    6 2698   0.0029564
```

```r
confusionMatrix(testing$classe,pred2)
```

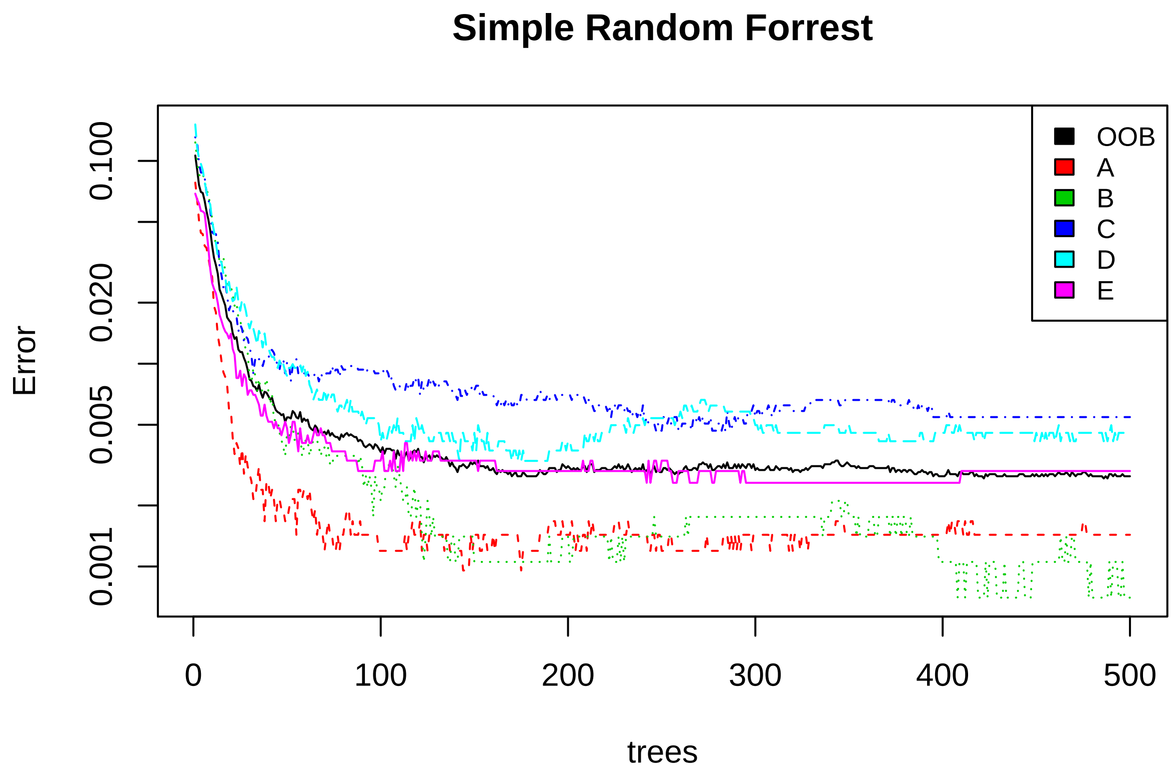
```
## Warning: package 'e1071' was built under R version 3.1.1
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1392    0    0    3    0
##          B    1  947    0    1    0
##          C    0    9  844    1    1
##          D    0    1    0  802    1
##          E    0    2    0    0  899
##
## Overall Statistics
##
##                Accuracy : 0.996
##                  95% CI : (0.994, 0.998)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.995
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.999    0.987    1.000    0.994    0.998
## Specificity             0.999    0.999    0.997    1.000    1.000
## Pos Pred Value          0.998    0.998    0.987    0.998    0.998
## Neg Pred Value          1.000    0.997    1.000    0.999    1.000
## Prevalence              0.284    0.196    0.172    0.165    0.184
## Detection Rate          0.284    0.193    0.172    0.164    0.183
## Detection Prevalence    0.284    0.194    0.174    0.164    0.184
## Balanced Accuracy       0.999    0.993    0.999    0.997    0.999
```

OOB estimate of error rate = 0.26%. Accuracy of the model = 99.76%

Plot of the Error Rate vs Number of Trees used

```r
plot(r2, log="y",main="Simple Random Forrest")
legend("topright", colnames(r2$err.rate),col=1:6,cex=0.8,fill=1:6)
```



The above plot shows that classifying **Activity Type D** has highest error rate.

Importance of Variables

```
VariableUsed<-varUsed(r2, by.tree=FALSE, count=TRUE)
Max_Imp=names(PML_Training_Subset)[which(VariableUsed==max(VariableUsed))]
Min_Imp=names(PML_Training_Subset)[which(VariableUsed==min(VariableUsed))]
```

Variable Used the most **num_window** Variable used the least **total_accel_belt**

Random Forrest with Bagging

```
predictors<-PML_Training_Subset[,-31]
Classe<-PML_Training_Subset[,31]
treeBag<-bag(predictors,Classe,B=10,
             bagControl=bagControl(fit=ctreeBag$fit,
                                   predict=ctreeBag$pred,
                                   aggregate=ctreeBag$aggregate))


pred3<-predict(treeBag,testing[,-31])

confusionMatrix(testing$classe,pred3)
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1380    5    4    3    3
##          B    5  926   12    4    2
##          C    6   17  829    2    1
##          D    3    3   25  770    3
##          E    1    1    5    6  888
##
## Overall Statistics
##
##                Accuracy : 0.977
##                  95% CI : (0.973, 0.981)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.971
##  Mcnemar's Test P-Value : 0.00374
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.989    0.973    0.947    0.981    0.990
## Specificity             0.996    0.994    0.994    0.992    0.997
## Pos Pred Value          0.989    0.976    0.970    0.958    0.986
## Neg Pred Value          0.996    0.993    0.989    0.996    0.998
## Prevalence              0.284    0.194    0.178    0.160    0.183
## Detection Rate          0.281    0.189    0.169    0.157    0.181
## Detection Prevalence    0.284    0.194    0.174    0.164    0.184
## Balanced Accuracy       0.992    0.983    0.970    0.986    0.993
```

Confusion Matrix of the bagged model shows accuracy of 98.06%

Random Forrest with PCA

Step 1: Create Principal Components and determine the number of components to be use

```
prComp<-prcomp(training[,-31],center=T,scale=T)
```

We will now calculate Eigen Values and components with eigen value greater than one will be used.

```
Eigen_Values=prComp$sdev^2
Eigen_Values
```

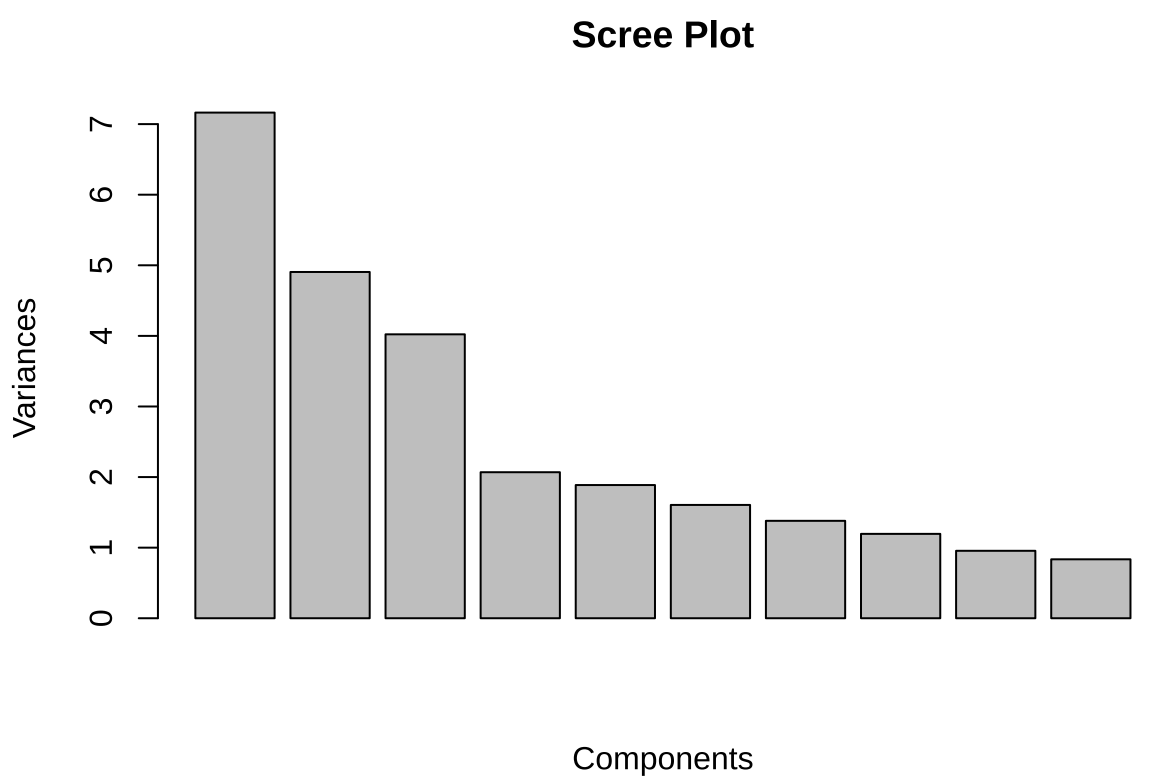
```
##  [1] 7.163154 4.905260 4.021911 2.068437 1.887075 1.604979 1.380029
##  [8] 1.195208 0.955182 0.834903 0.709864 0.641570 0.483029 0.366883
## [15] 0.328718 0.299530 0.271532 0.225821 0.175272 0.130238 0.086590
## [22] 0.060965 0.046186 0.037941 0.035091 0.031928 0.024001 0.019624
## [29] 0.006772 0.002307
```

```
No_of_comp=sum(Eigen_Values>1)
```

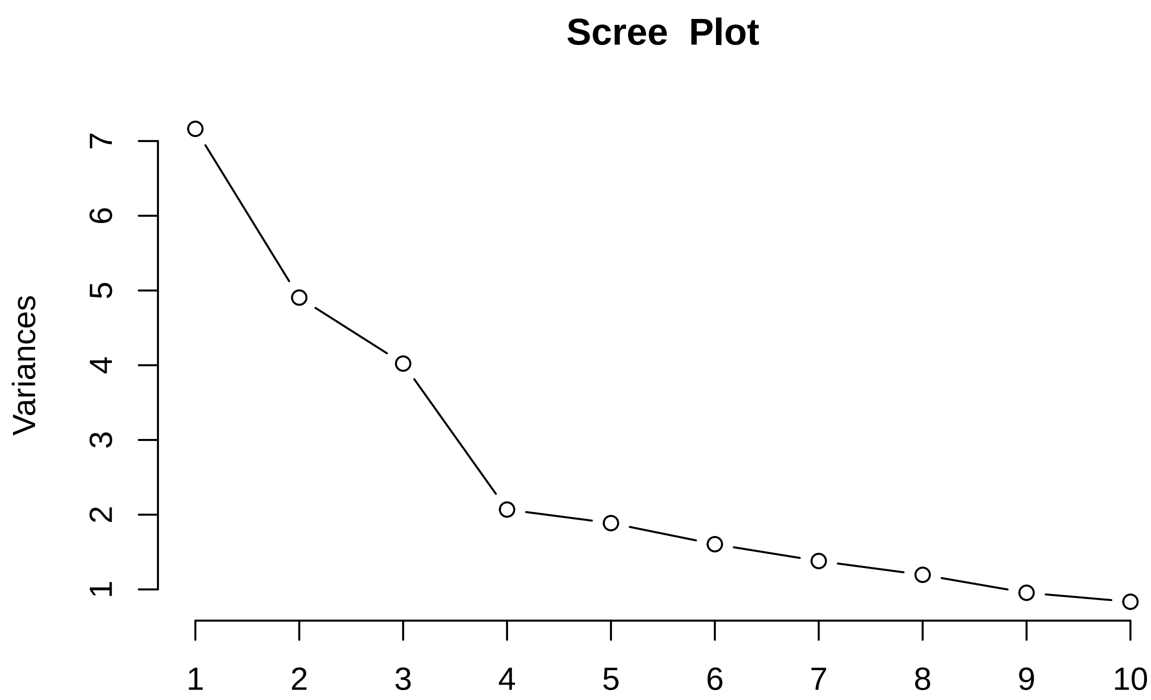
Eigen Value calculation shows that we should use **8** components.

We will also create screeplot to see after how many components variance plot flattens

```
screeplot(prComp,main="Scree Plot",xlab="Components")
```




```
screeplot(prComp,type="line",main="Scree  Plot")
```



The Plot shows that after 10 components, variance flattens.

We will take the middle path and go with 9 components.

Varimax Rotation shows loading of different variables on these components

```
summary(prComp)
```

```
## Importance of components:
##                          PC1   PC2   PC3    PC4    PC5    PC6   PC7    PC8
## Standard deviation     2.676 2.215 2.005 1.4382 1.3737 1.2669 1.175 1.0933
## Proportion of Variance 0.239 0.164 0.134 0.0689 0.0629 0.0535 0.046 0.0398
## Cumulative Proportion  0.239 0.402 0.536 0.6053 0.6682 0.7217 0.768 0.8075
##                           PC9   PC10   PC11   PC12   PC13   PC14  PC15
## Standard deviation     0.9773 0.9137 0.8425 0.8010 0.6950 0.6057 0.573
## Proportion of Variance 0.0318 0.0278 0.0237 0.0214 0.0161 0.0122 0.011
## Cumulative Proportion  0.8394 0.8672 0.8909 0.9123 0.9284 0.9406 0.952
##                           PC16    PC17    PC18    PC19    PC20    PC21
## Standard deviation     0.54729 0.52109 0.47521 0.41865 0.36089 0.29426
## Proportion of Variance 0.00998 0.00905 0.00753 0.00584 0.00434 0.00289
## Cumulative Proportion  0.96152 0.97058 0.97810 0.98395 0.98829 0.99117
##                           PC22    PC23    PC24    PC25    PC26   PC27
## Standard deviation     0.24691 0.21491 0.19478 0.18732 0.17868 0.1549
## Proportion of Variance 0.00203 0.00154 0.00126 0.00117 0.00106 0.0008
## Cumulative Proportion  0.99321 0.99474 0.99601 0.99718 0.99824 0.9990
##                           PC28    PC29    PC30
## Standard deviation     0.14009 0.08229 0.04803
```

```
## Proportion of Variance 0.00065 0.00023 0.00008
## Cumulative Proportion  0.99970 0.99992 1.00000
```

```r
load = prComp$rotation
my.var=varimax(load)
my.var
```

```
## $loadings
##
## Loadings:
##                   PC1 PC2 PC3 PC4 PC5 PC6 PC7 PC8 PC9 PC10 PC11 PC12 PC13
## num_window                                       -1
## roll_belt
## pitch_belt
## yaw_belt
## total_accel_belt
## gyros_belt_x                                          1
## gyros_belt_z                             -1
## accel_belt_x          1
## accel_belt_y
## accel_belt_z
## magnet_belt_x
## magnet_belt_y                    -1
## magnet_belt_z
## roll_arm
## pitch_arm                                                           -1
## yaw_arm                                                    1
## total_accel_arm                      -1
## gyros_arm_x                      1
## gyros_arm_y
## gyros_arm_z
## accel_arm_x
## accel_arm_y        1
## accel_arm_z
## magnet_arm_x
## magnet_arm_y
## magnet_arm_z              1
## roll_dumbbell                                                   1
## pitch_dumbbell                               -1
## yaw_dumbbell
##                   PC14 PC15 PC16 PC17 PC18 PC19 PC20 PC21 PC22 PC23 PC24
## num_window
## roll_belt
## pitch_belt
## yaw_belt
## total_accel_belt
## gyros_belt_x
## gyros_belt_y                               1
## gyros_belt_z
## accel_belt_x
## accel_belt_y                                                   1
## accel_belt_z
## magnet_belt_x                                        1
```

```
## magnet_belt_y
## magnet_belt_z                         1
## roll_arm               1
## pitch_arm
## yaw_arm
## total_accel_arm
## gyros_arm_x
## gyros_arm_y                                              -1
## gyros_arm_z      -1
## accel_arm_x                                                         1
## accel_arm_y
## accel_arm_z
## magnet_arm_x               -1
## magnet_arm_y                                    1
## magnet_arm_z
## roll_dumbbell
## pitch_dumbbell
## yaw_dumbbell                     1
##                  PC25 PC26 PC27 PC28 PC29 PC30
## num_window
## roll_belt                                  1
## pitch_belt                 -1
## yaw_belt               1
## total_accel_belt                -1
## gyros_belt_x
## gyros_belt_y
## gyros_belt_z
## accel_belt_x
## accel_belt_y
## accel_belt_z                          1
## magnet_belt_x
## magnet_belt_y
## magnet_belt_z
## roll_arm
## pitch_arm
## yaw_arm
## total_accel_arm
## gyros_arm_x
## gyros_arm_y
## gyros_arm_z
## accel_arm_x
## accel_arm_y
## accel_arm_z       1
## magnet_arm_x
## magnet_arm_y
## magnet_arm_z
## roll_dumbbell
## pitch_dumbbell
## yaw_dumbbell
##
##                  PC1   PC2   PC3   PC4   PC5   PC6   PC7   PC8   PC9  PC10
## SS loadings    1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
## Proportion Var 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
## Cumulative Var 0.033 0.067 0.100 0.133 0.167 0.200 0.233 0.267 0.300 0.333
```

```
##                 PC11  PC12  PC13  PC14  PC15  PC16  PC17  PC18  PC19  PC20
## SS loadings    1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
## Proportion Var 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
## Cumulative Var 0.367 0.400 0.433 0.467 0.500 0.533 0.567 0.600 0.633 0.667
##                 PC21  PC22  PC23  PC24  PC25  PC26  PC27  PC28  PC29  PC30
## SS loadings    1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
## Proportion Var 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
## Cumulative Var 0.700 0.733 0.767 0.800 0.833 0.867 0.900 0.933 0.967 1.000
##
## $rotmat
##            [,1]       [,2]      [,3]      [,4]     [,5]      [,6]
##  [1,]  0.305035 -0.2050669  0.010866 -0.047551 -0.04060 -0.111858
##  [2,]  0.132762  0.2939648  0.128973  0.041037 -0.17029 -0.043720
##  [3,]  0.164746 -0.1030894  0.437960 -0.015938  0.05666  0.113829
##  [4,] -0.045038 -0.2336950  0.005774  0.490290 -0.30020 -0.075725
##  [5,]  0.093004  0.1525602  0.009478  0.458310  0.41407  0.008319
##  [6,]  0.143259  0.0519818  0.200946  0.089032 -0.20650  0.590779
##  [7,]  0.009198  0.0732994 -0.109918  0.098695 -0.20663 -0.260350
##  [8,]  0.048780  0.0327790 -0.070733  0.041760 -0.20121  0.003975
##  [9,]  0.042774 -0.0314983  0.038084  0.061684  0.10882  0.144194
## [10,]  0.093683  0.0002283  0.017352  0.057739  0.19850  0.124565
## [11,] -0.009372 -0.0554777  0.042603 -0.045853  0.05126  0.087847
## [12,]  0.121882 -0.0256287 -0.075051  0.058186 -0.03063 -0.169591
## [13,]  0.068962  0.1689595  0.039734  0.045529 -0.07858 -0.054535
## [14,] -0.056390  0.0476895 -0.037856  0.176587  0.01246 -0.079140
## [15,] -0.035596 -0.2120225 -0.065628  0.050416  0.15870  0.041621
## [16,] -0.246811  0.0088887 -0.085998  0.094756 -0.23359  0.450045
## [17,] -0.142767  0.0549696 -0.118271  0.015330  0.14046  0.408804
## [18,] -0.015699 -0.1195012 -0.139926 -0.038965  0.38209  0.195322
## [19,]  0.081334 -0.0426249  0.020626  0.041518  0.35181 -0.046962
## [20,]  0.372827 -0.0770488 -0.325391 -0.010972 -0.24617  0.226813
## [21,] -0.287119 -0.1323574  0.386876  0.102855 -0.18815 -0.061704
## [22,]  0.030562 -0.0689884 -0.092387 -0.581779 -0.10996  0.021033
## [23,] -0.039826  0.5306194 -0.055238 -0.049003  0.11604  0.003026
## [24,] -0.151189  0.0304060  0.171588 -0.329585  0.07956  0.004529
## [25,]  0.329309 -0.1678111 -0.369098  0.033233  0.02446  0.011037
## [26,]  0.044441 -0.2356613  0.334692 -0.028478  0.07503  0.005831
## [27,] -0.087388 -0.4843594  0.001720 -0.029161  0.09232 -0.010078
## [28,] -0.319754 -0.2086880 -0.264145  0.040951 -0.02070 -0.002998
## [29,] -0.488249  0.0401495 -0.247296  0.019152  0.01121 -0.015257
## [30,]  0.010389 -0.0028593  0.013430 -0.008577  0.07193  0.001674
##             [,7]      [,8]      [,9]      [,10]      [,11]     [,12]
##  [1,] -0.1170119 -0.018841  0.053922 -5.015e-02  0.1228903 -0.017015
##  [2,] -0.2186458  0.214825 -0.083585  2.832e-01 -0.0623043  0.174915
##  [3,]  0.1068364  0.006220  0.028210 -1.496e-01  0.0372491 -0.110934
##  [4,] -0.0868007  0.078248 -0.122341  1.657e-02 -0.1534241 -0.054883
##  [5,] -0.0712059 -0.105334  0.106718  9.344e-03  0.2592926 -0.057517
##  [6,] -0.0140886 -0.160181 -0.051530 -1.586e-02  0.1284538  0.245891
##  [7,]  0.4398149 -0.211226  0.311958 -2.203e-01 -0.0512565  0.358028
##  [8,]  0.1082001  0.610326 -0.037654 -2.589e-01  0.2466259 -0.478267
##  [9,] -0.1793602  0.195034  0.708997 -8.049e-02 -0.4525481 -0.056831
## [10,]  0.3904656  0.191409 -0.451309  9.573e-02 -0.3423946  0.107496
## [11,]  0.2590606  0.063561  0.352859  3.670e-01  0.4762876 -0.054017
## [12,] -0.1986140  0.065158 -0.089572 -4.164e-01  0.3216238  0.417125
```

```
## [13,]  0.0055147 -0.144378 -0.020221 -1.631e-01 -0.3402870 -0.231089
## [14,] -0.0395912 -0.131074 -0.035442  3.511e-01  0.1246839 -0.196080
## [15,] -0.0200226  0.314430  0.050020  3.227e-01 -0.1078583  0.390402
## [16,]  0.2220551 -0.191683  0.033394 -6.497e-02  0.0541792 -0.073553
## [17,] -0.0914038  0.350508  0.007507 -2.507e-01  0.0421777  0.203851
## [18,] -0.2989202 -0.310336 -0.118892 -1.816e-01 -0.0184273 -0.173866
## [19,]  0.5069131  0.029438 -0.001938 -1.927e-01 -0.0151208 -0.045243
## [20,]  0.0824208 -0.104950  0.033427  1.481e-01 -0.0843776 -0.120245
## [21,] -0.0364921 -0.027194  0.005182 -9.650e-02  0.0135007  0.023171
## [22,] -0.0329699 -0.022988  0.023718 -7.743e-02  0.0248527 -0.005049
## [23,] -0.0134351  0.048665  0.020386 -8.615e-03  0.0060765  0.021528
## [24,]  0.0405211  0.033938 -0.025815  1.476e-01 -0.0134395  0.018110
## [25,] -0.0059176  0.017202 -0.011566  2.696e-02  0.0080362  0.038025
## [26,] -0.0156863 -0.004365 -0.008045 -4.042e-02 -0.0097664  0.010494
## [27,]  0.0159256  0.012977 -0.007610  4.361e-02  0.0038306 -0.008907
## [28,] -0.0185367  0.022939 -0.012158 -9.627e-05  0.0025189  0.016312
## [29,]  0.0076418 -0.004725 -0.001457 -1.950e-02  0.0019265 -0.012811
## [30,] -0.0002282  0.002551 -0.001333 -7.175e-03  0.0006441  0.004816
##           [,13]     [,14]     [,15]      [,16]      [,17]    [,18]
##  [1,]  0.021679  0.257694  0.171959  0.0631814  0.0882060 -0.02145
##  [2,] -0.151277 -0.048236 -0.106237  0.1493408 -0.3436841  0.16671
##  [3,] -0.300074  0.057024 -0.030822  0.3534527  0.0753391 -0.05120
##  [4,]  0.031649 -0.239754 -0.173559  0.0477303  0.0768460  0.29654
##  [5,] -0.027291 -0.099899  0.284528  0.0436390  0.0068100 -0.39699
##  [6,]  0.124141  0.062552  0.118645 -0.3671134  0.0159011  0.20538
##  [7,] -0.193072  0.050102  0.091639  0.0895266 -0.0357652  0.23310
##  [8,] -0.032897  0.117089  0.181351 -0.1008931 -0.2335394  0.09336
##  [9,] -0.043935  0.097051 -0.270044 -0.1403487 -0.1488914 -0.12720
## [10,]  0.066220  0.031614 -0.308349 -0.0858043 -0.1324522 -0.29250
## [11,]  0.430445 -0.216829 -0.312745  0.0620944 -0.0636158  0.11108
## [12,]  0.225098  0.220538 -0.381734  0.0005013 -0.2060584 -0.21430
## [13,]  0.720754  0.222846  0.137540  0.2031297  0.1029027  0.05043
## [14,] -0.176418  0.719679 -0.313464 -0.0517579  0.2203138  0.12324
## [15,]  0.134468  0.293352  0.485850  0.0745233 -0.1726182  0.10269
## [16,] -0.078956  0.230379  0.040122  0.2493922 -0.3687589 -0.20956
## [17,] -0.028798  0.009623 -0.107284  0.2792663  0.5564344  0.14047
## [18,] -0.033998 -0.035697 -0.083733  0.0930727 -0.4022388  0.41813
## [19,] -0.012305  0.081687 -0.042714 -0.1155643 -0.0321198  0.29162
## [20,] -0.061826 -0.102817 -0.007658  0.0225204  0.1080592 -0.21341
## [21,]  0.074151  0.019787 -0.012628  0.1630160 -0.0748177 -0.15112
## [22,] -0.026202  0.081245 -0.001605 -0.1335288  0.0151053 -0.07932
## [23,] -0.008118  0.012399  0.014337 -0.0792349 -0.0077494  0.12073
## [24,]  0.029818 -0.001248 -0.017043  0.3944569 -0.0398471  0.01552
## [25,] -0.001744 -0.029094 -0.060325  0.2925867 -0.0511551  0.01199
## [26,]  0.009722  0.031572  0.019145 -0.3601240  0.0286835 -0.02215
## [27,]  0.025806 -0.001606 -0.002830  0.0300552 -0.0019613  0.03046
## [28,] -0.001840 -0.002644  0.015509 -0.1186971  0.0001711 -0.05091
## [29,]  0.004402 -0.002948  0.004901 -0.1092492  0.0150566 -0.11108
## [30,]  0.002663  0.001399  0.006479 -0.0018925  0.0025480  0.02403
##            [,19]     [,20]     [,21]     [,22]     [,23]     [,24]
##  [1,] -0.2259616  0.033431 -0.225326 -0.121368 -0.284923 -0.057244
##  [2,]  0.1337968  0.137936  0.262251  0.008086 -0.232004 -0.308106
##  [3,] -0.0537342  0.441033 -0.065953 -0.026252  0.129593 -0.275441
##  [4,]  0.0747365  0.068619 -0.221073  0.473912  0.064324 -0.039189
```

```
##  [5,] -0.0460292  0.002040  0.144039  0.428729 -0.090060 -0.008093
##  [6,] -0.0285946 -0.035898  0.019962  0.101299 -0.064294  0.190458
##  [7,] -0.3870476  0.092845  0.124607  0.113126 -0.009163  0.094207
##  [8,] -0.1616778  0.013437  0.080668  0.050292  0.018094  0.185151
##  [9,]  0.0108860 -0.046493 -0.043867  0.047679 -0.048850  0.096507
## [10,] -0.4108709 -0.014149  0.001661  0.068082 -0.061081  0.017658
## [11,] -0.1828038  0.069630  0.012947 -0.033109 -0.001516 -0.150407
## [12,]  0.1891664  0.198499 -0.035399  0.028011  0.048388  0.116457
## [13,] -0.0018658  0.149996  0.076256  0.044675 -0.029363 -0.143146
## [14,] -0.0391588 -0.036556  0.117463  0.102593  0.046689  0.066077
## [15,]  0.0343317  0.094254  0.044091  0.025929  0.154804 -0.043534
## [16,]  0.1635861 -0.109038 -0.305358  0.001038  0.001852 -0.188485
## [17,] -0.0808505 -0.057777  0.136021 -0.021865 -0.070004 -0.087736
## [18,] -0.2735373  0.095758  0.198695 -0.011652  0.002972  0.054785
## [19,]  0.5825418 -0.063223  0.173846  0.025561 -0.125626 -0.077860
## [20,]  0.1775930  0.393359  0.402908 -0.068194  0.152620  0.227285
## [21,] -0.0530550 -0.318685  0.562715 -0.089806  0.255922  0.069179
## [22,] -0.0373375 -0.101818  0.170509  0.638928  0.014115 -0.350490
## [23,] -0.0345456  0.158450 -0.196316  0.036006  0.618574  0.003462
## [24,]  0.1205023  0.130006 -0.117432  0.318929 -0.096615  0.620133
## [25,]  0.0123899 -0.384071 -0.034918  0.001522  0.360685 -0.105116
## [26,] -0.0221549  0.163625 -0.022533  0.012101  0.302983 -0.119470
## [27,]  0.0471742  0.084807 -0.056582  0.027829  0.198175  0.080320
## [28,] -0.0318085  0.307194  0.081437 -0.046656 -0.178294 -0.090346
## [29,] -0.0004065  0.279754  0.060945 -0.012820  0.034872 -0.127375
## [30,] -0.0039868 -0.006773 -0.045140  0.007284  0.010771 -0.004485
##           [,25]     [,26]     [,27]     [,28]     [,29]     [,30]
##  [1,] -0.097746 -0.345780 -0.191225  0.327922  0.333561 -0.343810
##  [2,] -0.061264  0.073578  0.302667  0.157988  0.165553 -0.136789
##  [3,]  0.377398  0.009079 -0.101356 -0.106125 -0.101906  0.093216
##  [4,] -0.071093 -0.132147 -0.231966 -0.003789  0.030372 -0.014768
##  [5,] -0.026064  0.022566  0.125366  0.059994  0.010269 -0.039153
##  [6,]  0.386207  0.015757  0.079792  0.059546  0.069870 -0.055702
##  [7,] -0.090341  0.111769  0.136898 -0.014765  0.019629  0.003084
##  [8,]  0.028851  0.090006  0.070864 -0.038292  0.003316  0.023391
##  [9,]  0.089410 -0.034377 -0.016808  0.027045  0.021549 -0.027287
## [10,]  0.010604 -0.091709 -0.020514  0.031265  0.034441 -0.047021
## [11,] -0.040894 -0.031747 -0.083693 -0.014851 -0.005996  0.003798
## [12,] -0.005889 -0.006067 -0.031917 -0.098510 -0.047929  0.059868
## [13,]  0.033350  0.118229  0.164363  0.011385 -0.013634  0.016751
## [14,]  0.016050  0.065216  0.036196 -0.034992 -0.037976  0.047953
## [15,] -0.026387 -0.050247 -0.244606 -0.196445 -0.112374  0.105589
## [16,] -0.303775 -0.007025  0.028866  0.046794  0.001933  0.001148
## [17,] -0.260250  0.034669  0.071980  0.023738  0.023467 -0.040000
## [18,] -0.116620 -0.028508 -0.125303 -0.070970 -0.018432 -0.007370
## [19,]  0.025846 -0.120984 -0.101698  0.129896  0.079069 -0.115250
## [20,] -0.210141 -0.114057 -0.106776 -0.033656  0.066906  0.004636
## [21,]  0.034743 -0.232810 -0.164319  0.191190  0.081304 -0.049558
## [22,]  0.001789 -0.128083 -0.042156 -0.036418 -0.036326  0.037085
## [23,] -0.024766 -0.359366  0.007858  0.295646 -0.008548 -0.083499
## [24,]  0.009866  0.196995 -0.057202  0.209061  0.095239 -0.108019
## [25,]  0.349606  0.375052  0.006832  0.232999  0.106815 -0.105438
## [26,] -0.444792  0.534845 -0.008665  0.209873  0.111300 -0.122343
## [27,]  0.023767 -0.293750  0.755490 -0.086966  0.078714 -0.113127
```

```
## [28,]  0.204363  0.037961  0.020473  0.656692 -0.322037  0.203421
## [29,]  0.292487  0.143404 -0.131097 -0.230795  0.517203 -0.359570
## [30,] -0.014159 -0.038060  0.030829  0.103075  0.629685  0.762703
```

Cumulative variance equals 1, if we use al the 30 components. However, we decide to use 9 components.

Step 2: Pre-processing to Compute Components

```
preProc<-preProcess(training[,-31],method="pca",pcaComp=9)
```

Step 3: Train the PCA Model on Train set and apply on Test set

```
trainPC<-predict(preProc,training[,-31])
testPC<-predict(preProc,testing[,-31])
```

Step 4: Fit Random Forest on trainPC

```
r = randomForest(training$classe ~., data=trainPC, importance=TRUE, do.trace=100)
```

```
## ntree      OOB      1      2      3      4      5
##   100:   8.21%  6.79% 10.74%  8.84% 10.82%  4.84%
##   200:   7.73%  6.28%  9.76%  8.34% 10.95%  4.40%
##   300:   7.68%  6.21%  9.97%  8.18% 10.90%  4.18%
##   400:   7.53%  6.16%  9.52%  7.99% 10.57%  4.43%
##   500:   7.46%  6.16%  9.52%  7.91% 10.36%  4.29%
```

Step 5: Fit the trained model on training set

```
pred<-predict(r,testPC)
```

We will now construct the Confusion Matrix to assess the accuracy of the Model

```
confusionMatrix(testing$classe,pred)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1316   34   21   22    2
##          B   42  857   34   10    6
##          C   23   27  779   22    4
##          D   11   28   49  703   13
##          E    3   13    6   12  867
##
## Overall Statistics
##
##                Accuracy : 0.922
##                  95% CI : (0.914, 0.929)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : < 2e-16
##
```

```
##                  Kappa : 0.901
##  Mcnemar's Test P-Value : 0.00224
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.943    0.894    0.876    0.914    0.972
## Specificity             0.977    0.977    0.981    0.976    0.992
## Pos Pred Value          0.943    0.903    0.911    0.874    0.962
## Neg Pred Value          0.977    0.974    0.973    0.984    0.994
## Prevalence              0.284    0.196    0.181    0.157    0.182
## Detection Rate          0.268    0.175    0.159    0.143    0.177
## Detection Prevalence    0.284    0.194    0.174    0.164    0.184
## Balanced Accuracy       0.960    0.935    0.929    0.945    0.982
```

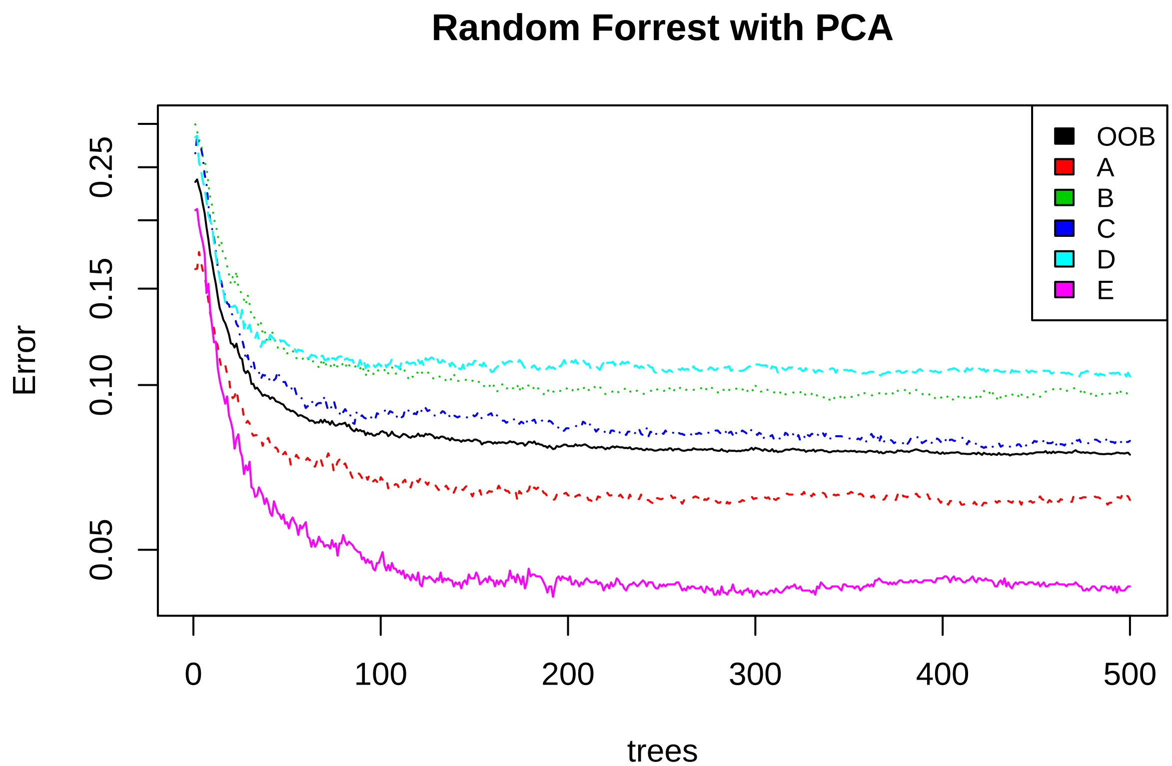
```r
r
```

```
##
## Call:
##  randomForest(formula = training$classe ~ ., data = trainPC, importance = TRUE,      do.trace = 100)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 7.46%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3927  109   62   66   21     0.06165
## B  124 2577  103   29   15     0.09515
## C   58   85 2364   52    8     0.07908
## D   37   60  129 2162   24     0.10365
## E    8   44   22   42 2590     0.04287
```

Accuracy of the Model = 92.31% OOB estimate of error rate = 7.37%

Plot of the Error Rate vs Number of Trees used

```r
plot(r, log="y",main="Random Forrest with PCA")
legend("topright", colnames(r$err.rate),col=1:6,cex=0.8,fill=1:6)
```



The above plot shows that classifying **Activity Type D** has highest error rate.

Importance of Variables

```
VariableUsedPCA<-varUsed(r, by.tree=FALSE, count=TRUE)
Max_ImpPCA=names(testPC)[which(VariableUsedPCA==max(VariableUsedPCA))]
Min_ImpPCA=names(testPC)[which(VariableUsedPCA==min(VariableUsedPCA))]
```

Component Used the most **PC9** Component used the least **PC6**

Model Selection and Prediction

Since, Simple Random Forest gives us the highest accuracy, we will use this model to classify the activities in the Second Dataset.

Step 1: Subset the Dataset to be predicted

```
columns_PML_TEST<-which(names(PML_Testing) %in% c("num_window",
                                          "roll_belt",
                                          "pitch_belt",
                                          "yaw_belt",
                                          "total_accel_belt",
                                          "gyros_belt_x",
                                          "gyros_belt_y",
                                          "gyros_belt_z",
                                          "accel_belt_x",
```

```
                                                    "accel_belt_y",
                                                    "accel_belt_z",
                                                    "magnet_belt_x",
                                                    "magnet_belt_y",
                                                    "magnet_belt_z",
                                                    "roll_arm",
                                                    "pitch_arm",
                                                    "yaw_arm",
                                                    "total_accel_arm",
                                                    "gyros_arm_x",
                                                    "gyros_arm_y",
                                                    "gyros_arm_z",
                                                    "accel_arm_x",
                                                    "accel_arm_y",
                                                    "accel_arm_z",
                                                    "magnet_arm_x",
                                                    "magnet_arm_y",
                                                    "magnet_arm_z",
                                                    "roll_dumbbell",
                                                    "pitch_dumbbell",
                                                    "yaw_dumbbell"))
```

Step 2: Predict Activities based on Simple Random Forest Model

```
pred_rf<-predict(r2,PML_Testing[,columns_PML_TEST])
```

Just for comparison, we will compare predictions of all the models

```
predictPC<-predict(preProc,PML_Testing[,columns_PML_TEST])

pred_rf_pca<-predict(r,predictPC)

pred_treeBag<-predict(treeBag,PML_Testing[,columns_PML_TEST])

Combined_Prediction_DS=data.frame(RandomForest=pred_rf,RF_PCA=pred_rf_pca,TreeBag=pred_treeBag)

View(Combined_Prediction_DS)
```

Conclusion

Random Forest with PCA gives a different value for only Third Prediction.