Let's start by making sure the workspace is clear

```
rm(list = ls())
```

Loading the dataset and necessary packages

```
trainSet = read.csv("/home//filipe/PracticalMachineLearning/pml-training.csv")
testSet = read.csv("/home//filipe/PracticalMachineLearning/pml-testing.csv")
library("caret")
```

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

```
library("ggplot2")
library("rattle")
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 3.0.2 r169 Copyright (c) 2006-2013 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

take 30% of the trainSet out for future model evaluation

```
learningInd = createDataPartition(y = trainSet$classe, p = 0.7, list = FALSE)
learnSet = trainSet[learningInd, ]
validateSet = trainSet[-learningInd, ]
```

Uppon inspection, there are several NAs and we need to remove them.

```
i = 1
W = c()
for (c in 1:dim(learnSet)[2]) {
    vec = learnSet[, c]
    numNa = sum(is.na(vec))
    if (numNa > 0) {
        W[i] = c
        i = i + 1
    }
    rm(numNa)
}

learnSet = learnSet[, -W]
validateSet = validateSet[, -W]
testSet = testSet[, -W]
```

Also, there seems to be a strange formating arround every “new windon == yes”. So let's remove those rows

```
w = which(learnSet$new_window == "yes")
learnSet = learnSet[-w, ]
w = which(validateSet$new_window == "yes")
validateSet = validateSet[-w, ]
```

A series of columns are empty. Let's remove them.

```
i = 1
W = c()
for (c in 1:dim(testSet)[2]) {
    vec = testSet[, c]
    if (length(unique(vec)) == 1) {
        W[i] = c
        i = i + 1
    }
}
```

```r
learnSet = learnSet[, -W]
validateSet = validateSet[, -W]
testSet = testSet[, -W]
```

And finally a series of identifiers of data colection are not relevant predictors (timestamps, etc).

```r
w = c(1, 3, 4, 5, 6)
learnSet = learnSet[, -w]
validateSet = validateSet[, -w]
testSet = testSet[, -w]
```

Make everything that should be numeric, numeric and center and scale the training set, and do the same transformation on the validation set

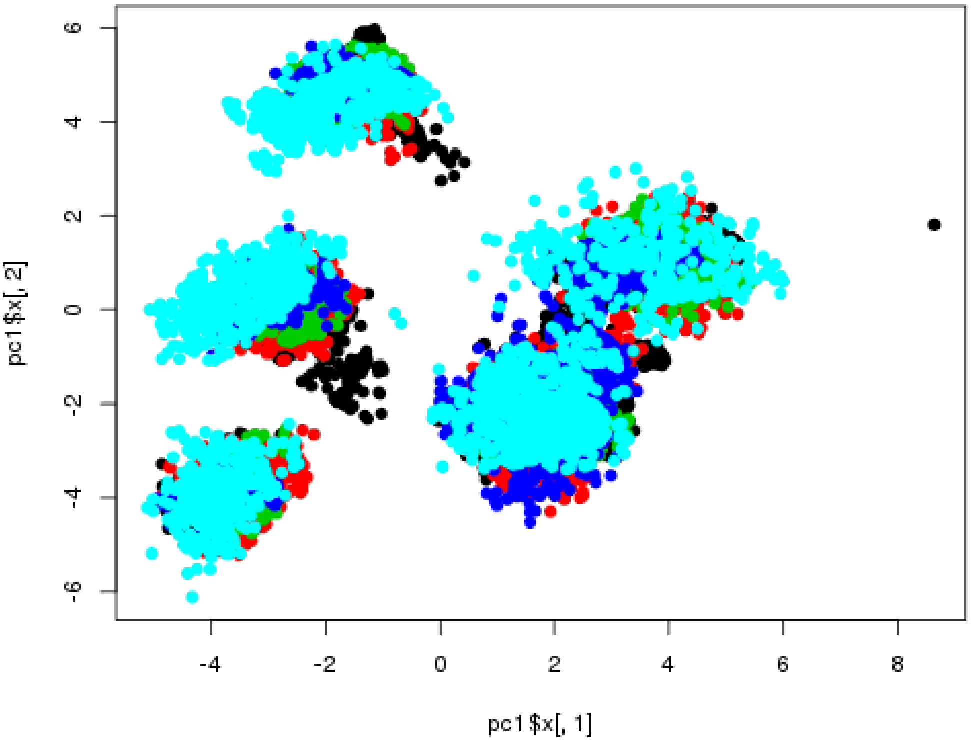
```r
wClasse = which(names(learnSet) == "classe")
wName = 1
means = colMeans(learnSet[, -c(wName, wClasse)])
sds = (sapply(X = learnSet[, -c(wName, wClasse)], FUN = sd))
learnSetNorm = learnSet
validateSetNorm = validateSet
testSetNorm = testSet

for (c in 2:(dim(testSet)[2] - 1)) {
    vec = learnSet[, c]
    learnSetNorm[, c] <- (as.numeric(vec) - means[c - 1])/sds[c - 1]

    vec = validateSet[, c]
    validateSetNorm[, c] <- (as.numeric(vec) - means[c - 1])/sds[c - 1]

    vec = testSet[, c]
    testSetNorm[, c] <- (as.numeric(vec) - means[c - 1])/sds[c - 1]
}
```

An exploratory data analysis by using PCA

```r
pc1 = prcomp(x = (learnSetNorm[, -c(wName, wClasse)]), scale = FALSE, center = FALSE)
```

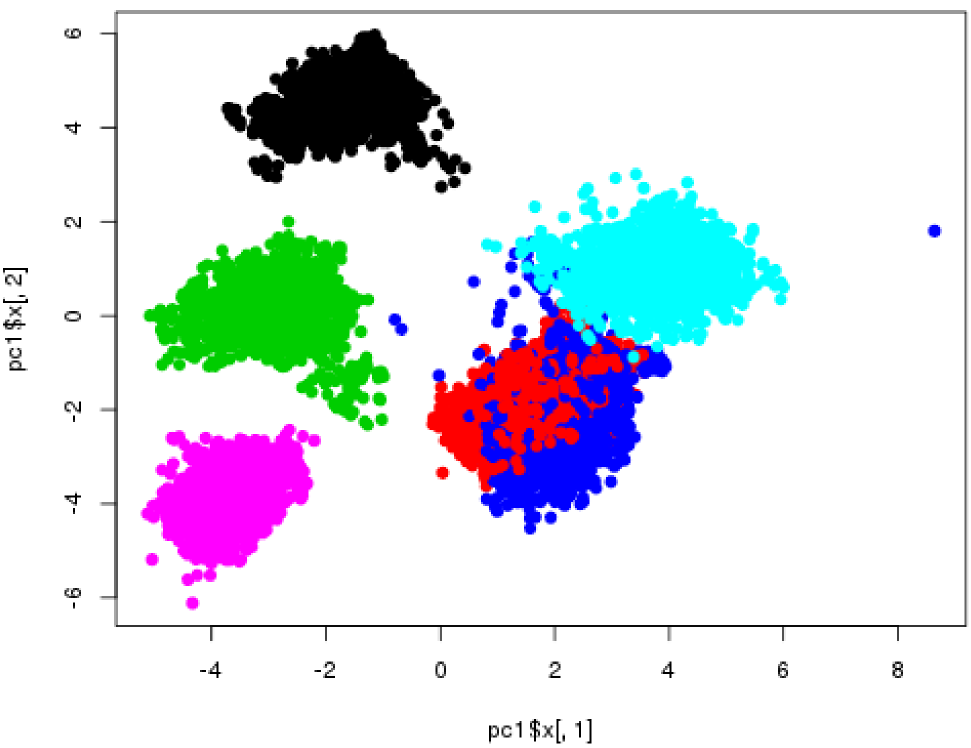
Plot and colour by class

```r
plot(pc1$x[, 1], pc1$x[, 2], pch = 19, col = as.numeric(learnSet$classe))
```



Plot and coulour by user

```
plot(pc1$x[, 1], pc1$x[, 2], pch = 19, col = as.numeric(learnSet$user_name))
```

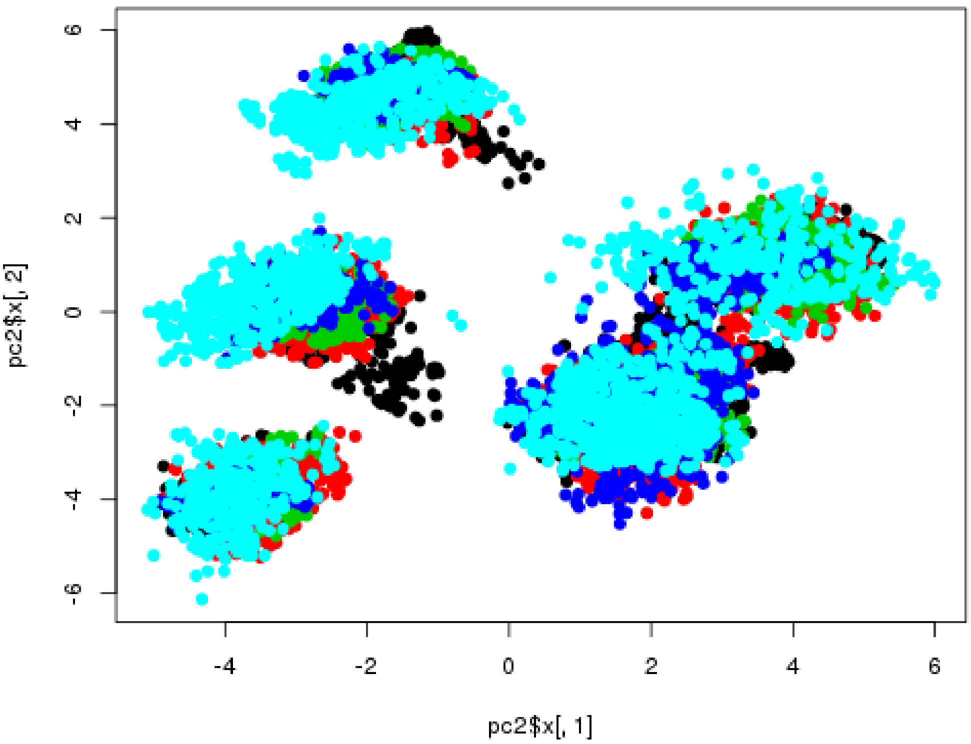


This last plot shows how “user-specific” the measures are. Therefore it's a good idea to have user_name as one of the variables that the models will be able to use.

Since there is one point that seems to be an outlier, let's remove it.

```
w = which.max(pc1$x[, 1])
learnSetNorm = learnSetNorm[-w, ]
learnSet = learnSet[-w, ]

pc2 = prcomp(x = (learnSetNorm[, -c(wName, wClasse)]), scale = FALSE, center = FALSE)
```

```
plot(pc2$x[, 1], pc2$x[, 2], pch = 19, col = as.numeric(learnSet$classe))
```



Time to train the model. I chose random forests, as they have a history of being highly powerful models and, by using bootstraping to create an emsenble to diferent predictive trees, they intrisically include cross validation in their method.

```
treeModelFitNorm = train(classe ~ ., data = learnSetNorm, method = "rf")
```

```
## Loading required package: randomForest
## randomForest 4.6-7
## Type rfNews() to see new features/changes/bug fixes.
```

Now we evalute the model on the validation data set (also the normalized version)

```
p1 = predict(treeModelFitNorm, validateSetNorm)
confusionMatrix(p1, validateSetNorm$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1641   23    0    0    0
##          B    1 1096    6    0    1
##          C    0    0  995    8    0
##          D    0    0    3  923    3
##          E    0    0    0    2 1050
##
```

```
## Overall Statistics
## 
##                Accuracy : 0.992          
##                  95% CI : (0.989, 0.994)  
##     No Information Rate : 0.285          
##     P-Value [Acc > NIR] : <2e-16         
## 
##                   Kappa : 0.99           
##  Mcnemar's Test P-Value : NA             
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.999    0.979    0.991    0.989    0.996
## Specificity             0.994    0.998    0.998    0.999    1.000
## Pos Pred Value          0.986    0.993    0.992    0.994    0.998
## Neg Pred Value          1.000    0.995    0.998    0.998    0.999
## Prevalence              0.285    0.195    0.175    0.162    0.183
## Detection Rate          0.285    0.191    0.173    0.160    0.183
## Detection Prevalence    0.289    0.192    0.174    0.162    0.183
## Balanced Accuracy       0.997    0.989    0.995    0.994    0.998
```

These are excelent results for the vlidation set with an out of bag error of less than 1%.