Project_writeup

Initialization and data loading:

```
library(caret)
```

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

```
library(ggplot2)
library(doMC)
```

```
## Loading required package: foreach
## Loading required package: iterators
## Loading required package: parallel
```

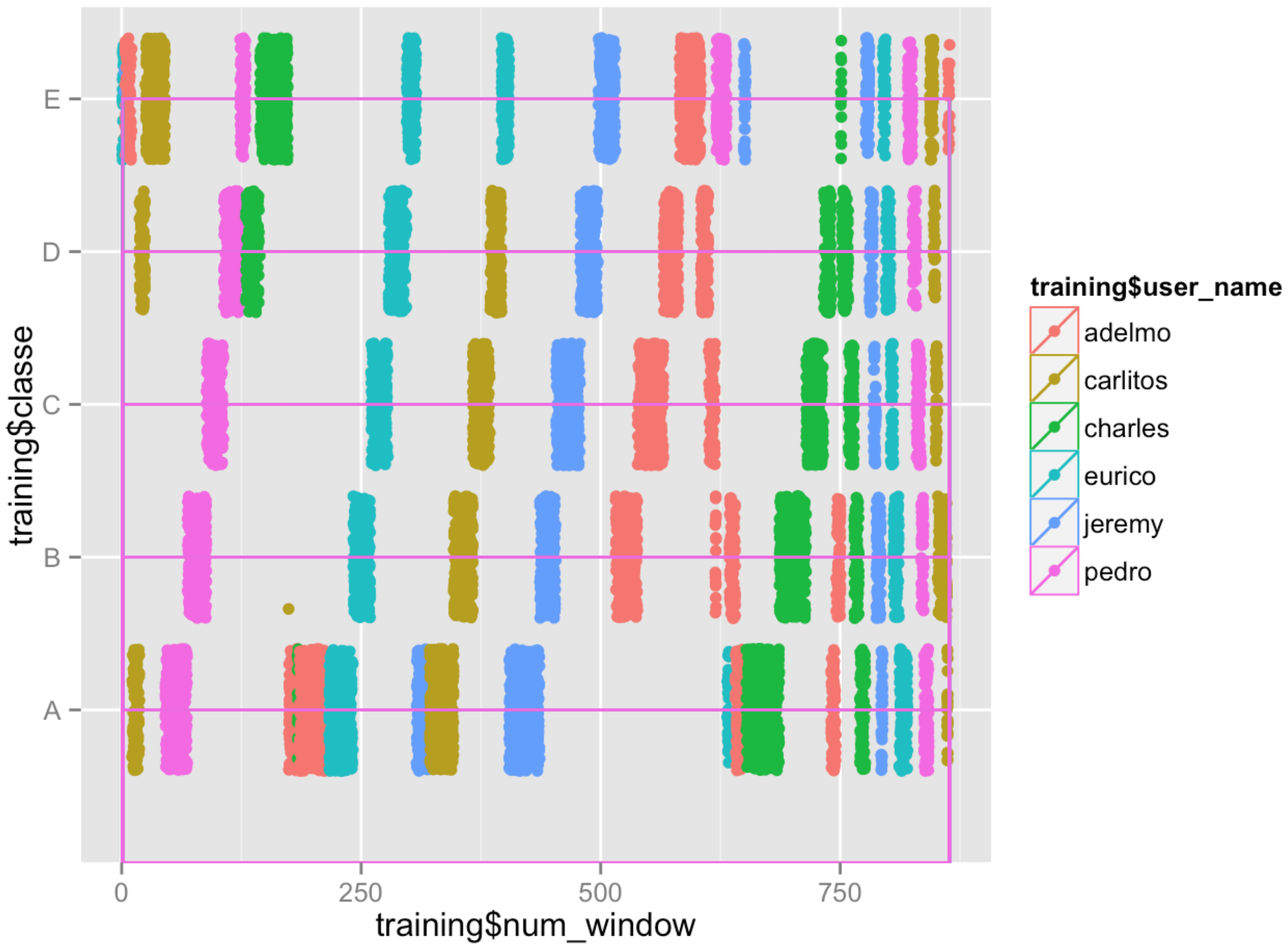
```
registerDoMC(cores = 8)
set.seed(12345)
data=read.csv2(file = "data/pml-training.csv", header = T, sep=",", dec=".", na.strings=c("#DIV/0!", "NA"), strip.white=T)
testing=read.csv2(file = "data/pml-testing.csv", header = T, sep=",", dec=".", na.strings=c("#DIV/0!", "NA"), strip.white=T)
dim(data)
```

```
## [1] 19622   160
```

There are 159 potential predictor, some of them are present only when the window change (new_window==‘yes’), because they are aggregate of other predictors in the time window. We create a validation set and a training set splitting the data (70/30)

```
data_idx=createDataPartition(data$classe, p=.70, list = F)
training=data[data_idx,]
validation=data[-data_idx,]
```

Plotting num_window against classe with user_name as color it is clear that the num_window is more than enough to model the data



In fact, training a random forest using only num_window achieves nearly 100% on the validation set.

```
simple_model=train(classe ~ num_window, data = training, tuneGrid=data.frame(mtry=5), method
="parRF")
```

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

```
m=confusionMatrix(validation$classe, predict(simple_model, validation))
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##         1.0000         1.0000         0.9994         1.0000         0.2845
## AccuracyPValue  McnemarPValue
##         0.0000            NaN
```

However, the problem states that we need to use only sensor data, Therefore we remove all predictors that do not contain valid values in the test set and other predictors that do not contain sensor data, such ad timestamps, user name and so forth.

```
predictor_names=names(testing[which(colSums(sapply(testing,is.na))==0)])
predictor_names=c("classe", predictor_names[9:length(predictor_names)-1])
length(predictor_names)-1
```

```
## [1] 52
```

```
new_training=training[,predictor_names]
new_validation=validation[,predictor_names]
```

We choose to use a random forest using the whole set of predictors, we run the parallel version of the algorithm, using special libraries to speed up processing.

```
rf_model=train(classe ~ ., data=new_training, tuneGrid=data.frame(mtry=5), method="parRF")
rf_model$results
```

```
##   mtry Accuracy  Kappa AccuracySD  KappaSD
## 1    5   0.9907 0.9883     0.0018 0.002273
```

To measure the out-of-sample performance of the obtained method we apply it on the validation set containing 30% of the data

```
validation_cf=confusionMatrix(new_validation$classe, predict(rf_model,new_validation))
validation_cf
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    1    0    0    0
##          B   11 1123    5    0    0
##          C    0   11 1014    1    0
##          D    0    0   21  943    0
##          E    0    0    0    5 1077
## 
## Overall Statistics
##                                           
##                Accuracy : 0.991           
##                  95% CI : (0.988, 0.993)  
##     No Information Rate : 0.286           
##     P-Value [Acc > NIR] : <2e-16          
##                                           
##                   Kappa : 0.988           
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.993    0.989    0.975    0.994    1.000
## Specificity             1.000    0.997    0.998    0.996    0.999
## Pos Pred Value          0.999    0.986    0.988    0.978    0.995
## Neg Pred Value          0.997    0.997    0.995    0.999    1.000
## Prevalence              0.286    0.193    0.177    0.161    0.183
## Detection Rate          0.284    0.191    0.172    0.160    0.183
## Detection Prevalence    0.284    0.194    0.174    0.164    0.184
## Balanced Accuracy       0.997    0.993    0.986    0.995    0.999
```

We aobtain very high values of accuracy and kappa in the validation set, thus we can infer that the model performances are satisfactory.