Introduction

Introduction

Human Activity Recognition - HAR - has emerged as a key research area in the last years and is gaining increasing attention by the pervasive computing research community , especially for the development of context-aware systems. There are many potential applications for HAR, like: elderly monitoring, life log systems for monitoring energy expenditure and for supporting weight-loss programs, and digital assistants for weight lifting exercises.

The human activity recognition research has traditionally focused on discriminating between different activities, i.e. to predict "which" activity was performed at a specific point in time. This Weight Lifting Exercises dataset is to investigate "how (well)" an activity was performed by the wearer. The "how (well)" investigation has only received little attention so far, even though it potentially provides useful information for a large variety of applications,such as sports training.

Backgroud

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement â€" a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, 6 participants were asked to perform barbell lifts correctly and incorrectly in 5 different ways, the goal was to use data from accelerometers on the belt, forearm, arm, and dumbell of these 6 participants and predict the manner of their exercise

1. Read the data

```
wle = read.csv("pml-training.csv", stringsAsFactors = FALSE)
dim(wle)

## [1] 19622   160
```

2. load library

```
library(caret)
library(ggplot2)
library(foreach)
library(randomForest)
```

3.Pre-Processing

3.1 Deal with Missing values

Look at the proportion of NA or blank values of each feature

```
features = colnames(wle)
mv = rep(NA,ncol(wle))
for (i in 1:ncol(wle)) {
  nan = sum(is.na(wle[,i]) | wle[,i]=='' | wle[,i]==' ')
  mv[i] = nan/nrow(wle)
}
```

Drop the variables with missing / blank values more than 90%

```
names(mv) = features  # match the header with the missing value percentage
to_drop = names(mv[mv>0.9]) # will remove the features with more than 90% missing value
ind = match(to_drop, features)
wle_new = wle[,-ind]
dim(wle_new)
```

```
## [1] 19622    60
```

After removing the missing and blank values, There are 60 features left

3.2 Feature Engineering

3.2.1 Categorical features

Only two categorical variables: User_name and new_window

```
table(wle_new$user_name, wle_new$classe)
```

```
##
##               A    B    C    D    E
##   adelmo   1165  776  750  515  686
##   carlitos  834  690  493  486  609
##   charles   899  745  539  642  711
##   eurico    865  592  489  582  542
##   jeremy   1177  489  652  522  562
##   pedro     640  505  499  469  497
```

```
table(wle_new$new_window, wle_new$classe)
```

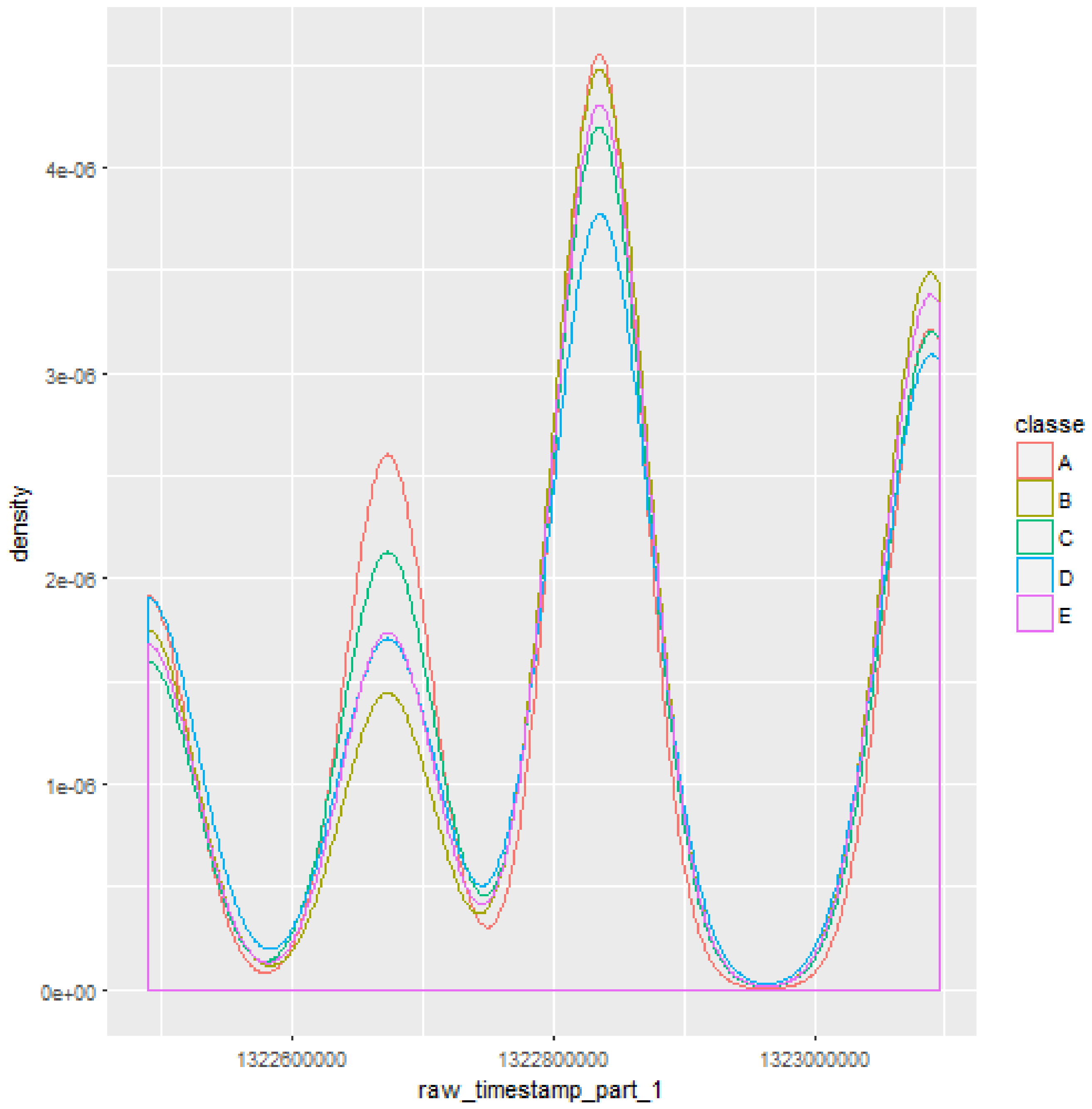
```
##
##          A    B    C    D    E
##   no  5471 3718 3352 3147 3528
##   yes  109   79   70   69   79
```

From the tables above, there are no pattern between these two categorical variable and the response variable (classe) Therefore we are going to drop User_name and new_window

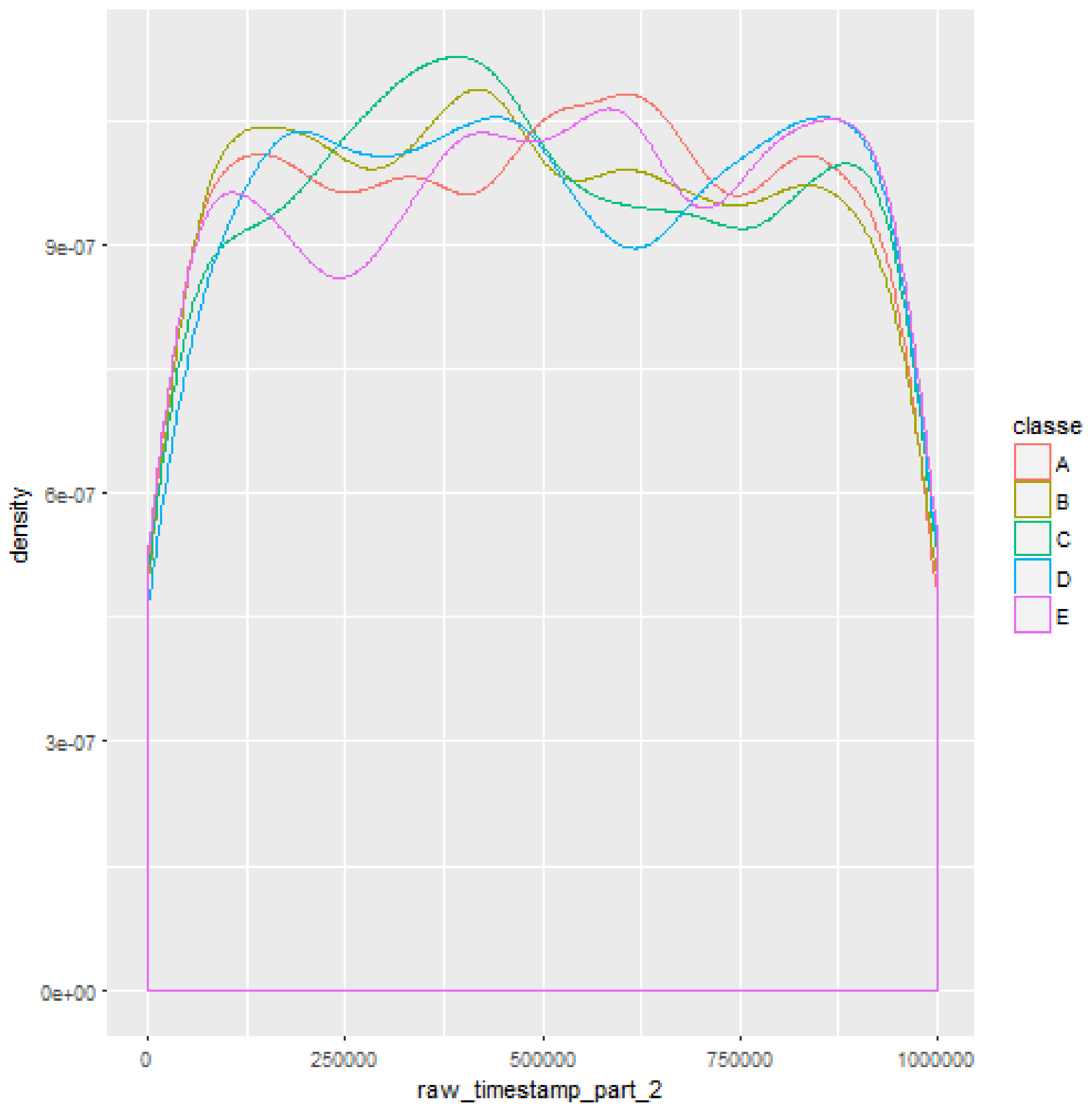
3.2.2 Time features

Intuitively, there should not be any correlation between the date/time of the experiment conducted and the result of the experimebnt But let's take a look at it

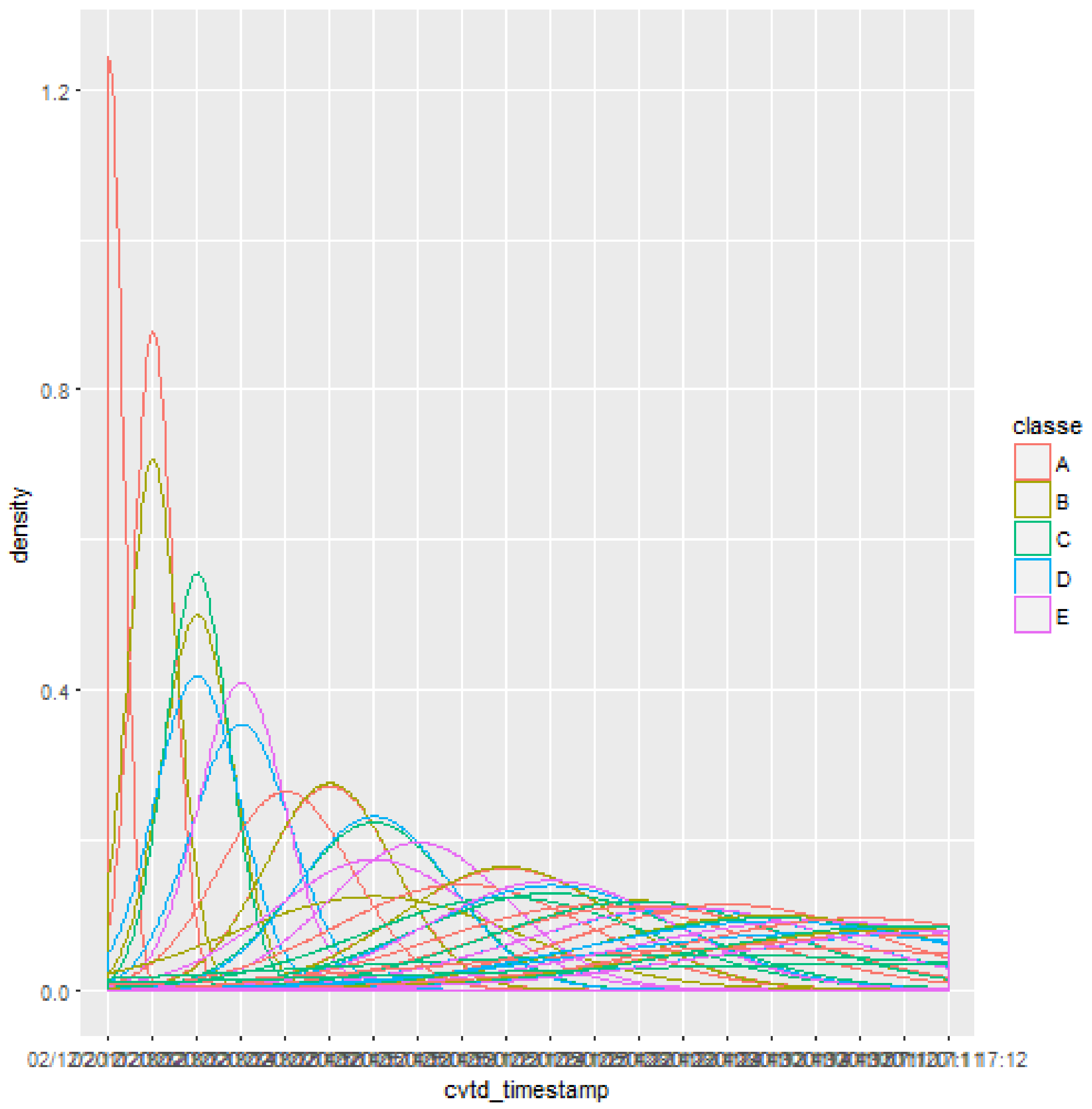
```
g11 <- ggplot(wle_new, aes(raw_timestamp_part_1, color = classe)) + geom_density()
g11
```



```
g12 <- ggplot(wle_new, aes(raw_timestamp_part_2, color = classe)) + geom_density()
g12
```



```
g13 <- ggplot(wle_new, aes(cvtd_timestamp, color = classe)) + geom_density()
g13
```



From the above density plots, we can clearly see that there is no connection between the 3 time variables and the outcome of the exercise, So we will drop them as well

Remove the unnecessary features

```
wle_new = wle_new[,7:60]
```

3.2.3 Numerical features

correlation analasis

Build a correlation matrix

```
corr.matrix = cor(wle_new[, 1:53], use = "pairwise.complete.obs")
corr.matrix[is.na(corr.matrix)] = 0
```

Figure out which features are highly correlated ##

```
corr_features = foreach(i = 1:nrow(corr.matrix))  %do% {
  rownames(corr.matrix[corr.matrix[,i] > 0.9,])
}

corr_features = corr_features[sapply(corr_features, function(x) length(x) > 0 )] ## remove empty rows
corr_features = unique(corr_features) ## remove duplicates
corr_features
```

```
## [[1]]
## [1] "roll_belt"        "total_accel_belt" "accel_belt_y"
##
## [[2]]
## [1] "gyros_dumbbell_z" "gyros_forearm_z"
```

There are 2 sets of correlated features and we will create new features to replace the correlated features

```
new_gyro = wle_new$gyros_dumbbell_z * wle_new$gyros_forearm_z
cor(wle_new$gyros_dumbbell_z, new_gyro)
```

```
## [1] 0.9901482
```

```
cor(wle_new$gyros_forearm_z, new_gyro)
```

```
## [1] 0.939303
```

```
new_belt = wle_new$total_accel_belt - wle_new$accel_belt_y - wle_new$roll_belt
cor(new_belt, wle_new$roll_belt)
```

```
## [1] -0.9917479
```

```
cor(new_belt, wle_new$accel_belt_y)
```

```
## [1] -0.9655505
```

Now we will replace the correlated features as indicated above with the new features

```
wle_new$new_gyro = new_gyro
wle_new$new_belt = new_belt
```

Remove the correlated features:

```
cor_f = c(corr_features[[1]], corr_features[[2]])
cor_ind = match(cor_f, colnames(wle_new))
## remove them
wle_new = wle_new[,-cor_ind]
```

So far we have total of 19622 observations and 51 features that we will use for modeling

```
dim(wle_new)
```

```
## [1] 19622    51
```

check missing value one more time

```
sum(is.na(wle_new))
```

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

3.2.4 Split the data to Train and CV sets

```
training = wle_new
training$classe = as.factor(training$classe)
```

Will split the data into 80% Training and 20% validation sets

```
set.seed(12345)
inTrain = createDataPartition(y = training$classe, p=0.8, list = FALSE)
training_train = training[inTrain,]
training_cv = training[-inTrain,]
dim(training_train);dim(training_cv)
```

```
## [1] 15699    51
```

```
## [1] 3923   51
```

4 Model Training

Random Forest Classifier for a Multi-Class Classification problem

For this problem, I chose Random Forest because of its robustness to sparsity and multi-colinearity, as well as its generalization to the test data.

```
rf_fit = randomForest(classe ~., data = training_train, ntree=500, mtry=10)
pred_rf = predict(rf_fit, newdata = training_cv, type = 'response')
table(prediction = pred_rf, actual = training_cv$classe)
```

```
##           actual
## prediction    A    B    C    D    E
##          A 1116    0    0    0    0
##          B    0  759    8    0    0
##          C    0    0  676    5    0
##          D    0    0    0  638    2
##          E    0    0    0    0  719
```

```
Pred_Acuracy = sum(diag(table(prediction = pred_rf, actual = training_cv$classe)))/length(training_cv$classe)
paste("Prediction Accuracy on the CV set is ", Pred_Acuracy)
```

```
## [1] "Prediction Accuracy on the CV set is  0.9961763956156"
```

Since a 99.6% prediction accuracy is achieved , therefore I decided that there is no need to tune parameters in order to improve the model performance

5, Test the model with Testing data

5.1 apply the same Pre-Procesing method to test data

```
test = read.csv("pml-testing.csv", stringsAsFactors = FALSE)
test_new = test[,-ind]
test_new = test_new[,7:60]
test_new$new_gyro = test_new$gyros_dumbbell_z * test_new$gyros_forearm_z
test_new$new_belt = test_new$total_accel_belt - test_new$accel_belt_y - test_new$roll_belt
test_new = test_new[,-cor_ind]
dim(test_new) ### make sure we have the same feasure space as training data
```

```
## [1] 20 51
```

5.2 Predict with trained Random Forest classifier

```
test_pred = predict(rf_fit, newdata = test_new, type = 'response')
test_pred
```

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

Reference

Data Source - http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har