CS 498 HW5, UIUC

March 4th, 2019

Team:

Apoorva Srinivasa (apoorva6@illinois.edu) , Julia Tcholakova (idt2@illinois.edu)

Experiment Table

h2o.kmeans used for the clustering, which yielded best accuracy for slice size = 30 and K = 40

	Run params	Slice size	Num of K clusters	Accuracy
1	S30_C40	30	40	0.9916963
3	S20_C60	20	60	0.9863238
4	S15_C20	15	20	0.9734456
5	S32_C20	32	20	0.9561091
2	S20_C20	20	20	0.9098457

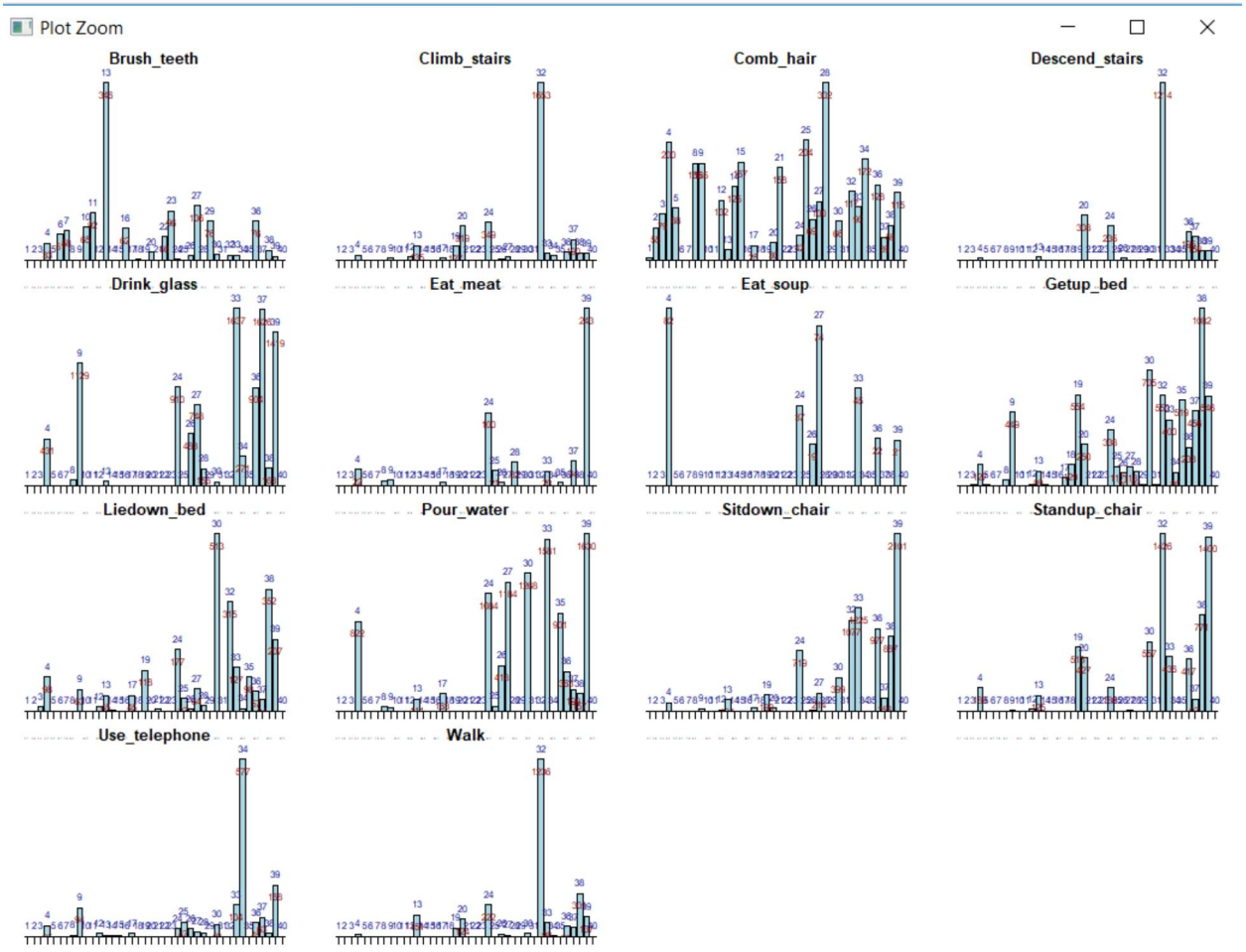
Histograms

Histograms of the mean quantized vector for each activity with K = 40 and slice = 30

K cluster in blue

Allocation per cluster in red

(Individual activity graphs included for reference at the end of all code section)



Confusion Matrix

CM based on the *h2o RandomForest* classifier with quantized data from K = 40 (slice = 30) clustering

```
Confusion Matrix and Statistics

          Reference
Prediction  1  2  3  4  5  6  7  8  9 10 11 12 13 14
        1   4  0  0  0  0  0  0  0  0  0  0  0  0  0
        2   0 34  0  0  0  0  0  0  0  0  0  0  0  0
        3   0  0 11  0  0  0  0  0  0  0  0  0  0  0
        4   0  0  0 14  0  0  0  0  0  0  0  0  0  0
        5   0  0  0  0 33  2  0  0  0  0  0  0  0  0
        6   0  0  0  0  0  0  0  0  0  0  0  0  0  0
        7   0  0  0  0  0  0  1  0  0  0  0  0  0  0
        8   0  0  0  0  0  0  0 34  0  0  0  0  0  0
        9   0  0  0  0  0  0  0  0 10  0  0  0  0  0
        10  0  0  0  0  0  0  0  0  0 33  0  0  0  0
        11  0  0  0  0  0  0  0  0  0  0 33  0  0  0
        12  0  0  0  0  0  0  0  0  0  0  0 34  0  0
        13  0  0  0  0  0  0  0  0  0  0  0  0  5  0
        14  0  0  0  0  0  0  0  0  0  0  0  0  0 33
```

```
Overall Statistics

               Accuracy : 0.9929
                 95% CI : (0.9745, 0.9991)
    No Information Rate : 0.121
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.992
 Mcnemar's Test P-Value : NA
```

Snippets of code:

```
#Segmentation
slice_signal = function(input_df, slice){
  segment_list = list()
  num = nrow(input_df)
  by_num = slice
    for (i in seq(1, num, by_num)){
    j = ifelse(i == 1, i, j + 1)
    step = i + by_num -1
    segment_df = input_df[i:step,]
    #check if there is a segment with no values
    num_back = nrow(segment_df[rowSums(is.na(segment_df)) > 0,])
    #back to overlap
    if(num_back > 0){
       a = i-num_back
       b = step-num_back
       segment_df = input_df[a:b,]
    }
       segment_list[[j]] = segment_df
  }
   random_sample = sample(segment_list, 100, replace = TRUE)
   return(random_sample)
 }

#K-means
all_segments_df = as.data.frame(all_segments_mat)
numCol = ncol(all_segments_df) - 2
clust_all = h2o.kmeans(training_frame = train_all_hex, k = clust_num)

#Generating the histogram
calcFeatures = function(seg_m, dict_m, clust_n){
  feature_v = seq(1:clust_num)
  seg_hex = as.h2o(seg_m, destination_frame = "seg.hex")
  pred_m = h2o.predict(dict_m, seg_hex)
    for (i in 1:length(feature_v)){
    feature_v[i] = sum(pred_m == i)
  }
  return(feature_v)
  }

#Classification
train_hex = as.h2o(train_all_df, destination_frame = "t.hex")
test_hex = as.h2o(test_all_df, destination_frame = "t_test.hex")
mod_cm = h2oModel(train_hex,1:clust_num, label, 30, 16)
acc_cm = calcAccuracy(mod_cm, test_hex, label)

library(caret)
test_hex_cm = as.h2o(test_all_df[, -label], destination_frame = "t_test_cm.hex")
test_pred = h2o.predict(object = mod_cm, newdata = test_hex_cm)
test_pred_c = as.data.frame(test_pred)
```

All code:

```
#Homework 5

#Julia and Apoorva

#data reference:
(https://archive.ics.uci.edu/ml/datasets/Dataset+for+ADL+Recognition+with+Wrist-
worn+Accelerometer,

#data provided by Barbara Bruno, Fulvio Mastrogiovanni and Antonio Sgorbissa).

############################################ Helper Functions
#########################################################################

library(h2o)

#function for slicing the signal

slice_signal = function(input_df, slice){

  segment_list = list()

  num = nrow(input_df)

  by_num = slice

  for (i in seq(1, num, by_num)){

    j = ifelse(i == 1, i, j + 1)

    step = i + by_num -1

    segment_df = input_df[i:step,]

    #check if there is a segment with no values

    num_back = nrow(segment_df[rowSums(is.na(segment_df)) > 0,])

    #back to overlap

    if(num_back > 0){

      a = i-num_back

      b = step-num_back
```

```
      segment_df = input_df[a:b,]
    }
     segment_list[[j]] = segment_df
  }
  random_sample = sample(segment_list, 100, replace = TRUE)
  return(random_sample)
}
#function to flatten a list into vectors
flatListMat = function(input_ls){
  segments_mat = matrix(nrow = length(input_ls), ncol = slice_num*3)
  for(i in 1:length(input_ls)){
    segments_mat[i,] = as.vector(rbind(input_ls[[i]]$V1, input_ls[[i]]$V2, input_ls[[i]]$V3 ))
  }
  return(segments_mat)
}
h2o.init(ip = 'localhost', port = 54321, max_mem_size = '2650m')
#create the feature vector for a doc
calcFeatures = function(seg_m, dict_m, clust_n){
  feature_v = seq(1:clust_num)
  seg_hex = as.h2o(seg_m, destination_frame = "seg.hex")
  pred_m = h2o.predict(dict_m, seg_hex)
  for (i in 1:length(feature_v)){
    feature_v[i] = sum(pred_m == i)
  }
```

```
  return(feature_v)
}
#functions to use for the classifier
#function to create the model with different parameters
h2oModel = function(tframe,a,b,nTree,depth){
  mod = h2o.randomForest(training_frame = tframe,
                         x = a,
                         y = b,
                         ntrees = nTree,
                         max_depth = depth)
  return(mod)
}
#function to calc the accuracies
calcAccuracy = function(mod, test_data, y){
  test_pred = h2o.predict(object = mod, newdata = test_data[, -y])
  acc_test = mean(test_pred[, 1] == test_data[, y])
  return(acc_test)
}


############################################################ VQ - Dictionary
#read all folders each file and store all activities in a list
base = "HMP_Dataset/"
act_folder = c(paste0(base,"Brush_teeth/"), paste0(base,"Climb_stairs/"),
paste0(base,"Comb_hair/"),
```

```
            paste0(base,"Descend_stairs/"),paste0(base,"Drink_glass/"),
paste0(base,"Eat_meat/"),

            paste0(base,"Eat_soup/"), paste0(base,"Getup_bed/"), paste0(base,"Liedown_bed/"),

            paste0(base,"Pour_water/"), paste0(base,"Sitdown_chair/"),
paste0(base,"Standup_chair/"),

            paste0(base,"Use_telephone/"), paste0(base,"Walk/"))



#global to reset

slice_num = 30

clust_num = 40

label = clust_num + 1

#store all data and make segments for all files

all_data_ls  = list()

all_segments_ls = list()

all_len = c()

list_idx = 1

for(i in 1:length(act_folder)){

  act_data_ls = list()

  files = list.files(act_folder[i],pattern = '*.txt', full.names = TRUE)

  for (j in 1:length(files)){

    file_ls = list()

    act_df = read.table(files[j], quote="\"", comment.char="")

    #store the file with the label

    act_data_ls[[j]] = cbind(act_df, y = i, f = j)

    #file segmented
```

```
    file_ls = slice_signal(act_df, slice_num)
    #flatten the file list
    file_mat = flatListMat(file_ls)
    file_mat = cbind(file_mat, y = i, f = j)
    #all_segments_ls = append(all_segments_ls,(slice_signal(act_df, slice_num)))
    all_segments_ls[[list_idx]] = file_mat
    #sanity check
    all_len = c(all_len, nrow(act_df))
    list_idx = list_idx + 1
  }
  #all_data_ls = append(act_data_ls,list(act_data_ls))
  all_data_ls[[i]] = act_data_ls
}
#make a matrix for init cluster input
all_segments_mat = do.call(rbind, all_segments_ls)

#build a dictionary
set.seed(20)
#clust_all = kmeans(all_segments_mat, clust_num, nstart = 10, iter.max = 20)
#dict_mat = clust_all$centers
all_segments_df = as.data.frame(all_segments_mat)
numCol = ncol(all_segments_df) - 2
train_all_hex = as.h2o(all_segments_df[, 1:numCol], destination_frame = "train_all.hex")
clust_all = h2o.kmeans(training_frame = train_all_hex, k = clust_num)
```

```
############################################### Prepare data for Classifier

#featurize all data

feature_vec_ls = list()

n_vec = 1

#loop by activities and files to make feature vectors

for (a in 1:14){

  act_df = subset(all_segments_df, all_segments_df$y == a)

  file_vec = sort(unique(act_df$f))

  for(f in 1:length(file_vec)){

    doc_n = subset(act_df, act_df$f == file_vec[f])

    pass = doc_n[, 1:numCol]

    feature_vec = calcFeatures(pass, clust_all, clust_num)

    feature_vec = c(feature_vec, y = a)

    feature_vec_ls[[n_vec]] = feature_vec

    n_vec = n_vec + 1

  }

}

 #flatten the list with doc features to a dataframe

feature_all_df = as.data.frame(do.call(rbind, feature_vec_ls))

#save to file

write.table(feature_all_df, paste0("features_", slice_num, "_" ,clust_num, ".csv"),
row.names=FALSE, col.names = TRUE, sep = ',')



#for each class 3 fold cross validation, divide into 3 - 2 for training, 1 for testing

train_run_ls = list()
```

```
test_run_ls = list()
for(k in 1:3){
  train_all_ls = list()
  test_all_ls = list()
  for(i in 1:14){
    #get all class data
    class_data = subset(feature_all_df, feature_all_df$y == i)
    train_idx = sample(seq_len(nrow(class_data)), size = floor(0.67 * nrow(class_data)))
    train_df = class_data[train_idx, ]
    test_df = class_data[-train_idx,]

    train_all_ls[[i]] = train_df
    test_all_ls[[i]] = test_df

  }#end of class loop
  #combine class data selected
  train_all_df = do.call(rbind, train_all_ls)
  test_all_df = do.call(rbind, test_all_ls)

  #store data for each run
  train_run_ls[[k]] = train_all_df
  test_run_ls[[k]] = test_all_df
}#end of run loop
#write to files for each run
```

```
for(k in 1:length(train_run_ls)){
  file_train = paste0("run_", k, "_train_", slice_num, "_", clust_num, ".csv")
  file_test = paste0("run_", k, "_test_", slice_num, "_", clust_num, ".csv")
  write.table(train_run_ls[[k]], file_train, row.names=FALSE, col.names = TRUE, sep = ',')
  write.table(test_run_ls[[k]], file_test, row.names=FALSE, col.names = TRUE, sep = ',')
}
acc_avg = rep(0, 3)
for(i in 1:length(acc_avg)){
  train_hex = as.h2o(train_run_ls[[i]], destination_frame = "t.hex")
  test_hex = as.h2o(test_run_ls[[i]], destination_frame = "t_test.hex")
  mod_test = h2oModel(train_hex,1:clust_num ,label ,30,16)
  acc_avg[i] = calcAccuracy(mod_test, test_hex, label)
}

tbl_vec = matrix(nrow = 1, ncol = 3)
tbl_vec[1,] = c(slice_num, clust_num, mean(acc_avg))
rownames(tbl_vec) = paste0("S", slice_num, "_C", clust_num)

############data for the table Page 1 ############
tbl_ls = list()
###add for all testing with different slice and cluster
tbl_ls = append(tbl_ls, list(tbl_vec))

####
```

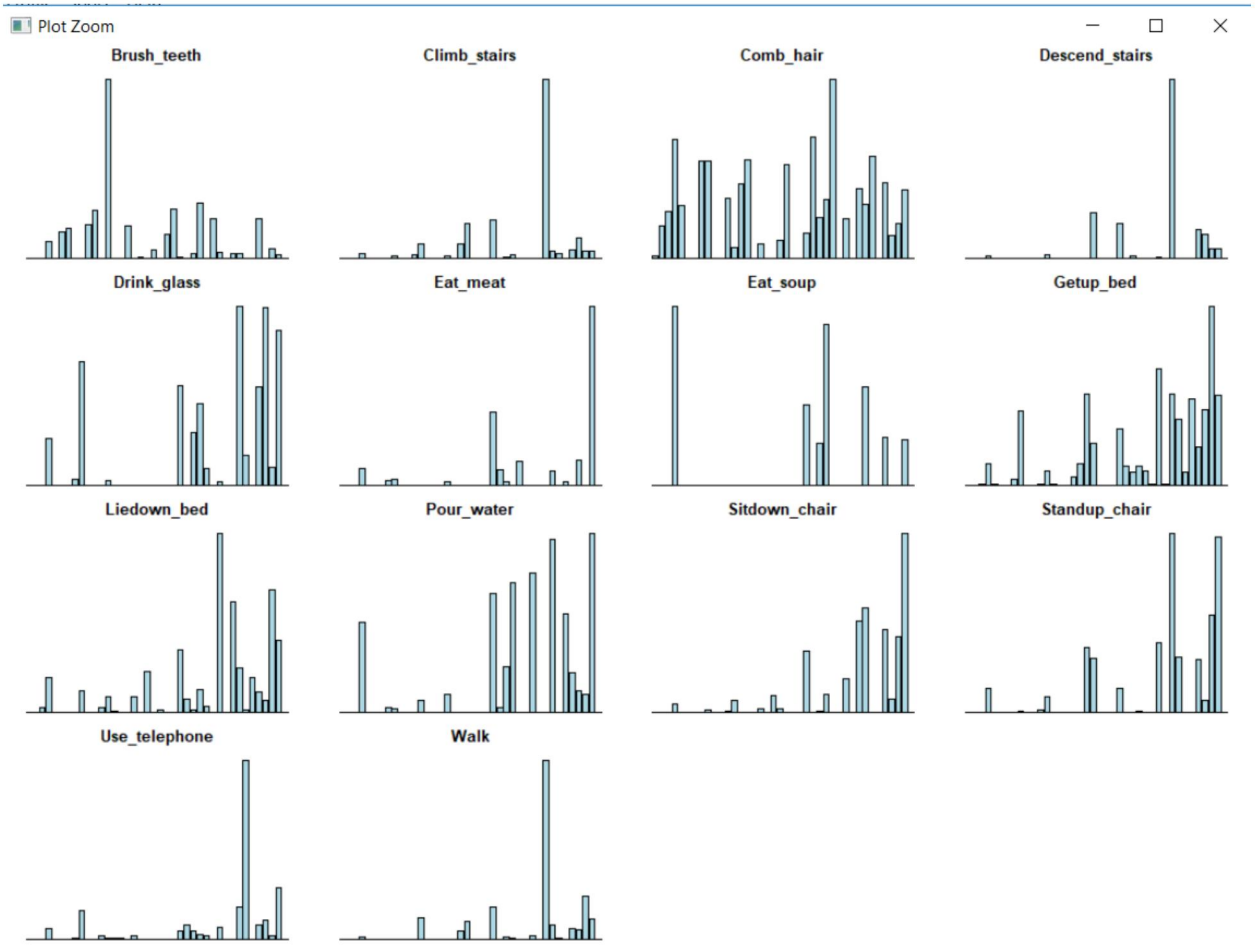
```
tbl_df = as.data.frame(do.call(rbind, tbl_ls))


write.table(tbl_df, "classifier tbl.csv", row.names=TRUE, col.names = FALSE, sep = ',')

#####  submission part 1  ##########

part_1 = read.csv("classifier tbl.csv", header = FALSE)

rnames = c("Run params", "Slice size", "Num of K clusters", "Accuracy")

colnames(part_1) = rnames

#order by accuracy desc

part_1_ordered = part_1[order(part_1$Accuracy, decreasing = TRUE),, drop = FALSE]

knitr::kable(part_1_ordered)


#write for histograms

#write.table(feature_all_df, "features_best_a.csv", row.names=FALSE, col.names = TRUE, sep
= ',')

#read the file back

#feature_all_df = read.csv("features_best_a.csv", header = TRUE)


feature_mean_df = as.data.frame(matrix(nrow = 14, ncol = clust_num + 1))

for(i in 1:nrow(feature_all_df)){

  act_sub = subset(feature_all_df, feature_all_df$y == i)

  act_mean = apply(act_sub, 2, mean)

  feature_mean_df[i, ] = act_mean

}

#mean histograms
```

```
act_names = c("Brush_teeth","Climb_stairs", "Comb_hair", "Descend_stairs", "Drink_glass",
"Eat_meat",

            "Eat_soup", "Getup_bed", "Liedown_bed", "Pour_water", "Sitdown_chair",
"Standup_chair",

            "Use_telephone", "Walk")




graphics.off()

#par("mar")

#par(mar=c(1,1,1,1))

par(mfrow=c(4,4))

for(i in 1:14)

{

  activity = feature_mean_df[i, 1:clust_num]

  ylim = c(0, 1.1*max(activity))

  #hist(as.numeric(activity), main=act_names[i], breaks = 40)

  bp = barplot(as.numeric(activity), axes = FALSE, main=act_names[i], col = "lightblue",

               ylim = ylim, xlab = paste0("Clusters - ", clust_num), ylab = "Frequency")

  text(x = bp, y = as.numeric(activity), labels = as.numeric(activity), pos = 1, cex = 0.8, col =
"darkred")

  text(x = bp, y = as.numeric(activity), pos = 3, cex = 0.8, col = "darkblue")

  axis(side = 1, at = bp, tick = TRUE, cex.axis = 0.01)

}



dev.off()
```

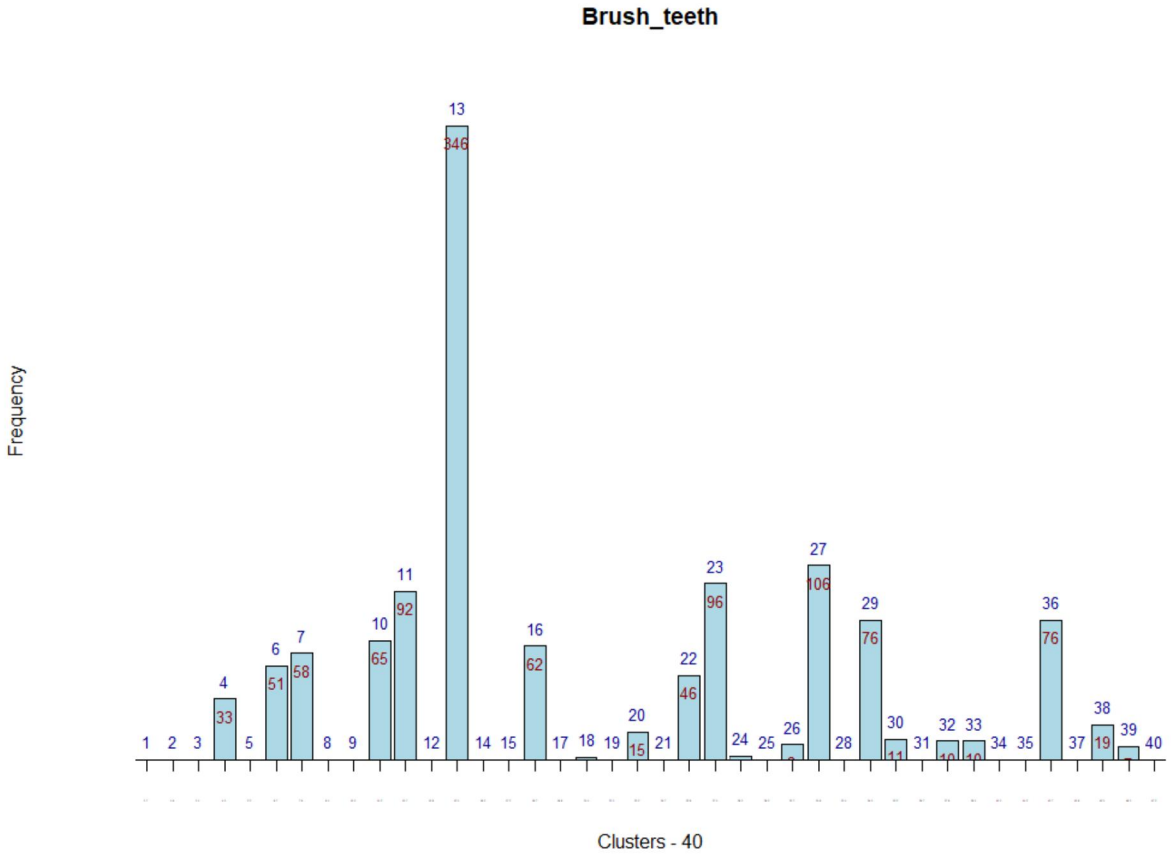
```
#CM for best accuracy

train_hex = as.h2o(train_all_df, destination_frame = "t.hex")

test_hex = as.h2o(test_all_df, destination_frame = "t_test.hex")

mod_cm = h2oModel(train_hex,1:clust_num, label, 30, 16)

acc_cm = calcAccuracy(mod_cm, test_hex, label)


library(caret)

test_hex_cm = as.h2o(test_all_df[, -label], destination_frame = "t_test_cm.hex")

test_pred = h2o.predict(object = mod_cm, newdata = test_hex_cm)

test_pred_c = as.data.frame(test_pred)

pred_y = as.integer(test_pred_c$predict)

pred_y = as.factor(pred_y)

true_y = as.factor(test_all_df$y)

conf_mat = confusionMatrix(pred_y, true_y)

h2o.removeAll()

########################################## Testing

#train

run_1_train = as.data.frame(do.call(rbind, train_run_ls[[1]]))

run_2_train = as.data.frame(do.call(rbind, train_run_ls[[2]]))

run_3_train = as.data.frame(do.call(rbind, train_run_ls[[3]]))

#test

run_1_test = as.data.frame(do.call(rbind, test_run_ls[[1]]))

run_2_test = as.data.frame(do.call(rbind, test_run_ls[[2]]))

run_3_test = as.data.frame(do.call(rbind, test_run_ls[[3]]))
```

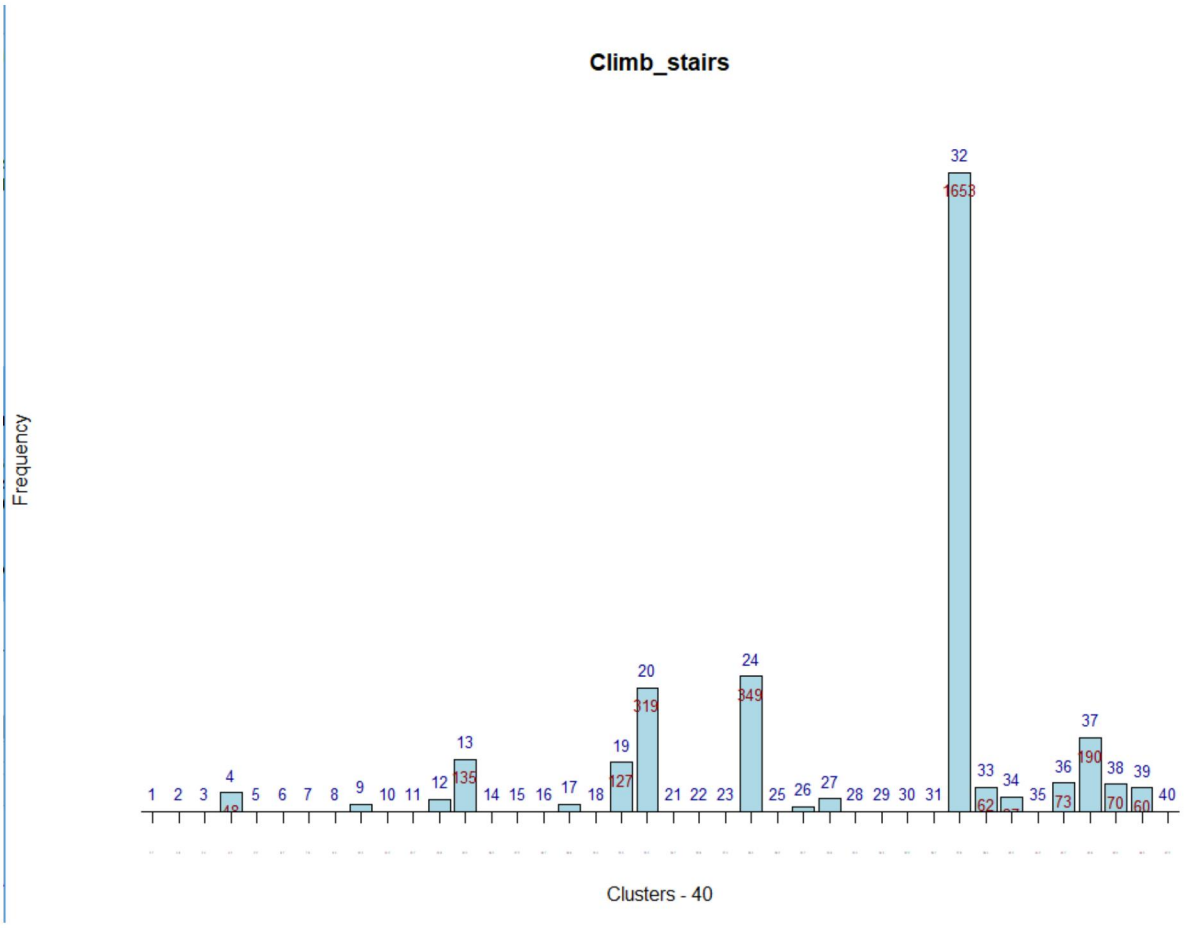
More graphs

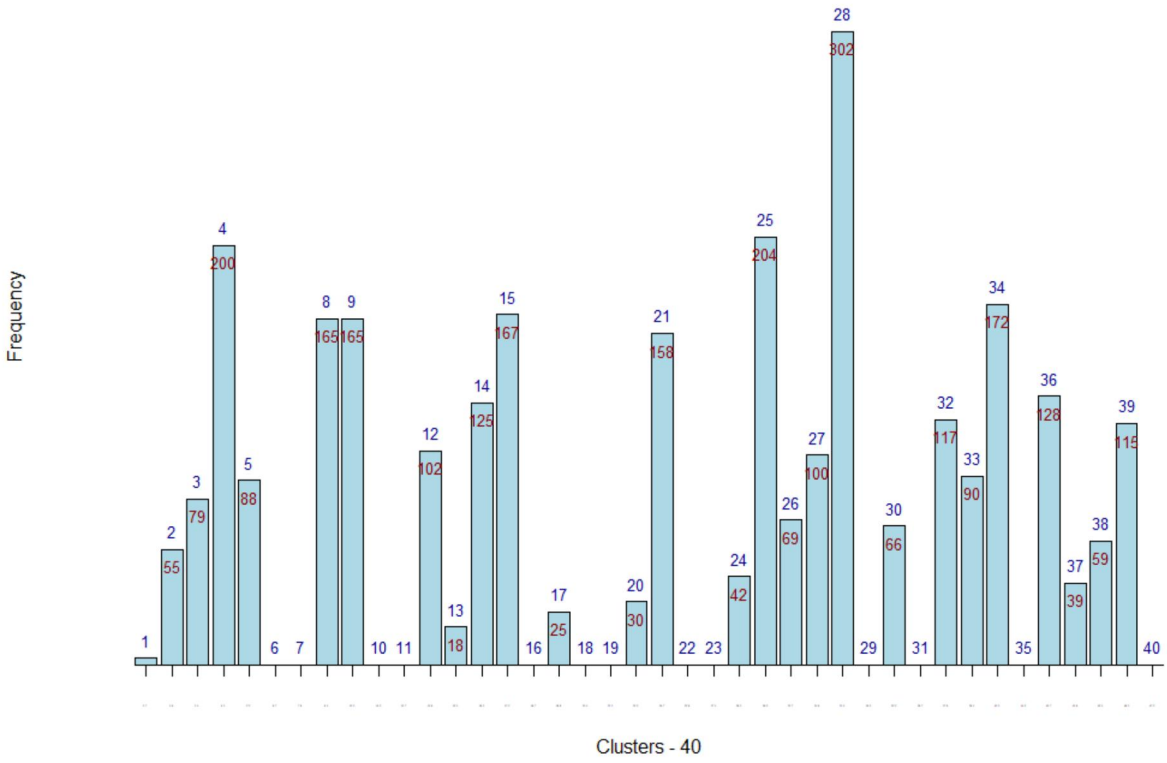
Cleaner look



Individual per activity:




Climb_stairs
Frequency
Clusters - 40
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
135
127
319
349
1653
62
73
190
70
60


Comb_hair
Frequency
1
2
55
3
79
4
200
5
88
6
7
8
165
9
165
10
11
12
102
13
18
14
125
15
167
16
17
25
18
19
20
30
21
158
22
23
24
42
25
204
26
69
27
100
28
302
29
30
66
31
32
117
33
90
34
172
35
36
128
37
39
38
59
39
115
40
Clusters - 40