Multivariate Statistics

Assignment 2

Tom Leppens r0716859
Wenhan Cu r0973145
Zeynep Deniz Guvenol r0768138
Thien Nguyen r0774196

Task 1

We first defined functions to calculate the error rate of the fitted models.

```
> #Calculate error rate from observed and predicted
> errorrate <- function(observed, predicted) {
+   tab<-table(observed, predicted)
+   errorrate<-1-sum(diag(tab))/sum(tab)
+   return(errorrate)
+ }
>
> #Calculate error rate from model ouput, training and test data
> model <- function(model.out, train.data, train.target, test.data, test.target) {
+   pred.train <- predict(model.out, train.data)
+   pred.test <- predict(model.out, test.data)
+   train <- errorrate(train.target, pred.train$class)
+   test <- errorrate(test.target, pred.test$class)
+   return(list(train = train, test = test))
+ }
>
> #Tune k for KNN
> tuneknn <- function(train.data, train.target, test.data, test.target, kmax) {
+   knn<-matrix(rep(0,kmax*2),nrow=kmax)
+   for (j in 1:kmax){
+     predknn.train<- knn(train.data, train.data, train.target, k=j)
+     knn[j,1]<-errorrate(train.target,predknn.train)
+
+     predknn.test<- knn(train.data, test.data, train.target, k=j)
+     knn[j,2]<-errorrate(test.target,predknn.test)
+   }
+   return(knn)
+ }
>
> #Plot KNN as a function of k
> plotknn <- function(knn, kmax) {
+   plot(-10,-10,xlim=c(1,kmax),ylim=c(0,0.15),col="red",type="b",xlab="K",ylab="error
")
+   lines(c(1:kmax),knn[,1],col="red")
+   lines(c(1:kmax),knn[,2],col="blue")
+   legend("topright",c("training error", "test error"),col=c("red","blue"),lty=c(1,1)
)
+ }
>
> #Find the best k for knn, not taking k=1 into account
> knnbest <- function(knn) {
+   best <- which(knn1[,2] == sort(unique(knn1[,2]))[2])
+   return(best = best)
+ }
>
> #Calculate error rate for random forest
> rferror <- function(train.data, train.target, test.data, test.target, mtry, ntree) {
+   rfdata<-data.frame(train.target=factor(train.target),train.data)
+   bag.mod=randomForest(train.target~.,data=rfdata,mtry=mtry,ntree=ntree,importance=T
RUE)
+   predrf.train<-predict(bag.mod,newdata=rfdata)
+   train<-errorrate(rfdata$train.target,predrf.train)
+   predrf.test<-predict(bag.mod,newdata=test.data)
+   test<-errorrate(test.target,predrf.test)
+   return(list(train = train, test = test))
+ }
>
> #Calculate error rate for HDDA
> hddaerror <- function(train.data, train.target, test.data, test.target, model, d_sel
ect, threshold) {
+   hdda.out <- hdda(train.data, train.target, model=model, d_select = d_select, thres
hold = threshold)
+   predhdda.train <- predict(hdda.out, train.data, train.target)
+   train<-errorrate(train.target,predhdda.train$class)
+   predhdda.test <- predict(hdda.out, test.data, test.target)
+   test<-errorrate(test.target,predhdda.test$class)
+   return(list(train = train, test = test))
```

```
+ }
```

a)

The principal components were extracted from training data and were used to transform the train and test data.

```
> pctraindata <- prcomp(traindata)
> pctestdata <- prcomp(testdata)
>
> totvar<- sum(apply(traindata,2,var))
> eigenvalues<-pctraindata$sdev^2
> propvar<-eigenvalues/totvar
> cumpropvar<-cumsum(propvar)

> #scenario 1: components account for 80% of the variance in the training data
> scen1 <- 43
> pcatrain1<-traindata%*%pctraindata$rotation[,1:scen1]
> pcatest1<-testdata%*%pctraindata$rotation[,1:scen1]
> #scenario 2: components account for 90% of the variance in the training data
> scen2 <- 86
> pcatrain2<-traindata%*%pctraindata$rotation[,1:scen2]
> pcatest2<-testdata%*%pctraindata$rotation[,1:scen2]
```

b)

```
> #LDA PCA scenario 1
> lda1.out<-lda(pcatrain1,train.target)
> lda1 <- model(lda1.out, pcatrain1, train.target, pcatest1, test.target)
>
> #LDA PCA scenario 2
> lda2.out<-lda(pcatrain2,train.target)
> lda2 <- model(lda2.out, pcatrain2, train.target, pcatest2, test.target)
>
> #QDA PCA scenario 1
> qda1.out<-qda(pcatrain1,train.target)
> qda1 <- model(qda1.out, pcatrain1, train.target, pcatest1, test.target)
>
> #QDA PCA scenario 2
> qda2.out<-qda(pcatrain2,train.target)
> qda2 <- model(qda2.out, pcatrain2, train.target, pcatest2, test.target)
>
> #KNN PCA scenario 1 with tuning
> knn1 <- tuneknn(pcatrain1, train.target, pcatest1, test.target, 20)
> plotknn(knn1, 20)
>
> #KNN PCA scenario 2 with tuning
> knn2 <- tuneknn(pcatrain2, train.target, pcatest2, test.target, 20)
> plotknn(knn2, 20)
>
> #Random Forest PCA scenario 1 (p = 43)
> rf1<-rferror(pcatrain1, train.target, pcatest1, test.target, 5, 500)
>
> #Random Forest PCA scenario 2 (p = 86)
> rf2<-rferror(pcatrain2, train.target, pcatest2, test.target, 5, 500)
>
> # HDDA AKJBKQKD (raw data)
> hdda1<-hddaerror(traindata, train.target, testdata, test.target, model="AKJBKQKD", d
_select = "Cattell", threshold = 0.05)
>
> # HDDA AKJBQKD (raw data)
> hdda2<-hddaerror(traindata, train.target, testdata, test.target, model="AKJBQKD", d_
select = "Cattell", threshold = 0.05)
```

The two KNN models for scenarios 1 and 2 were tuned for the best number of k. Training and test error were plotted as a function of k, as shown in the figures below. We do not take k=1 as the best model since its perfect fit is expected. Hence, the second-best models were chosen, here both k are equal to 3.

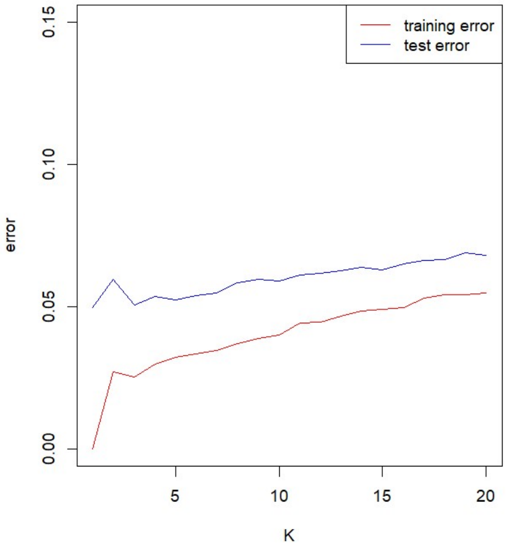

Figure 1.1. KNN1 performance

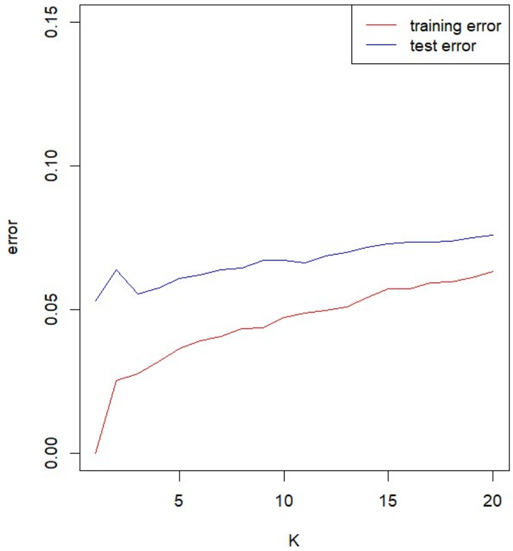

Figure 1.2. KNN2 performance

We tuned the number of variables to be considered, *mtry*, in the random forest models, resulting in mtry=5 for both. We selected 500 as the number of trees, since this is reasonably large with acceptable running time.

c)

```
> results.out<-results()
        train   test
LDA1  0.1300 0.1350
LDA2  0.1196 0.1274
QDA1  0.0230 0.0462
QDA2  0.0130 0.0564
KNN1  0.0254 0.0506
KNN2  0.0278 0.0556
RF1   0.0000 0.0690
RF2   0.0000 0.0702
HDDA1 0.0262 0.0526
HDDA2 0.0256 0.0508
[1] "The best KNN1 has k = "
[1] 3
[1] "The best KNN2 has k = "
[1] 3
[1] "The best model is: "
[1] "QDA1"
```

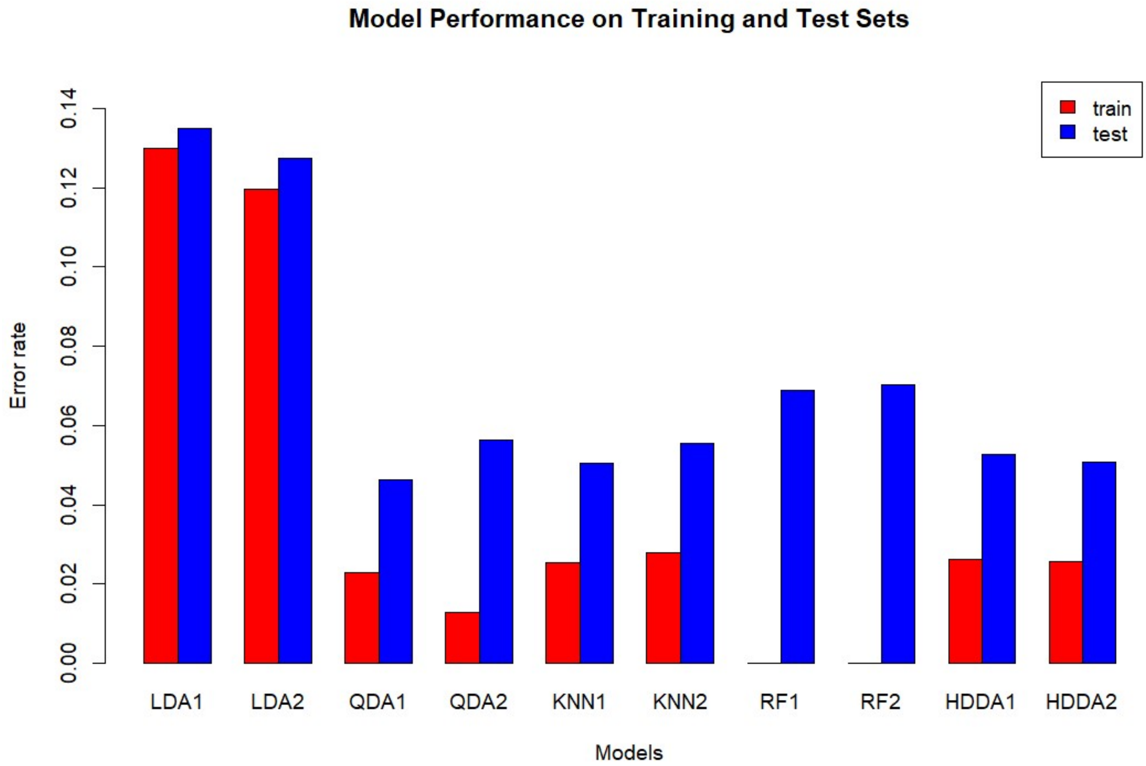

Figure 1.3. Model Performance

Overall, QDA, KNN, and HDDA are the best-performing models based on the error rate, with Random Forest performing perfectly on the training set but exhibiting slightly higher error rates on the test set compared to other models. LDA performs the worst, with a high error rate in both train and test set.

Task 2

The objective of this task was to investigate to what extent we can recover the true class labels using unsupervised clustering techniques with 4 clusters. This will be done using the Adjusted Rand Index (ARI). First, the data was loaded, then centered, and the true labels were extracted.

The asked analyses were executed:

- Hierarchical clustering on squared Euclidean distances using the method of Ward.
- K-means clustering.
- HDDC with model AkjBkQkD and initialized with hierarchical and K-means clustering results.
- HDDC with model AkjBQkD and initialized with hierarchical and K-means clustering results.

For the Hierarchical clustering, we first calculated the Euclidean distances and use the Ward.D2 to create the clusters. After this, we selected 4 clusters

```
> #Hierarchical clustering on squared Euclidean distances using the method of Ward
> dist_data <- dist(data, method = "euclidean",diag = TRUE, upper = TRUE)
> hier_clust <- hclust(dist_data,"ward.D2")
>
> #Cutting the tree to 4 clusters
> clust_hier_4 <- cutree(hier_clust, k = 4)
```

For K means clustering we used 500 different starting points with a max iteration of 2000 steps. Again 4 clusters centroids were used:

```
> #K-means clustering
> kmeans_result <-kmeans(data,4,nstart=500,iter.max=2000)
```

For the HDDC, 4 different models were created using the following code:

```
> #HDDC clustering AkjBkQkD with hierarchical clustering
> hddc_AkjBkQkD_hier <- hddc(data, K = 4, model = "AkjBkQkD",d_select = "Cattell" ,ini
t.vector = clust_hier_4, threshold = 0.05)
>
> #HDDC clustering AkjBkQkD with kmeans
> hddc_AkjBkQkD_means <- hddc(data, K = 4, model = "AkjBkQkD",d_select = "Cattell" ,in
it.vector = kmeans_result, threshold = 0.05)
>
> #HDDC clustering AkjBQkD with hierarchical clustering
> hddc_AkjBQkD_hier<-hddc(data, K = 4, model = "AkjBQkD",d_select = "Cattell" ,init.ve
ctor = clust_hier_4, threshold = 0.05)
>
> #HDDC clustering AkjBkQkD with kmeans
> hddc_AkjBQkD_means<-hddc(data, K = 4, model = "AkjBQkD",d_select = "Cattell" ,init.v
ector = kmeans_result, threshold = 0.05)
```

After creating all the models, the ARI was calculated for each method to evaluate performance, and the results are shown in the table below.

Method	*ARI*
Hierarchical	0.9076422
K-means	0.6843655
HDDC AkjBkQkD Hierarchical	0.7778358
HDDC AkjBQkD Hierarchical	0.7632849
HDDC AkjBkQkD K-means	0.7778358
HDDC AkjBQkD K-means	0.7785905

With a value of 0.91, the hierarchical clustering yielded an excellent recovery. All the other models have a moderate recovery. Using the first two principal components, we can visualize the observed and predicted class labels with the hierarchical model.

```
> # Visualizing the higest ARI with 2 PCA on centerd data
> prcomp<- prcomp(data, center = TRUE, scale. = FALSE)
> pc2_data <- prcomp$x[, 1:2]
> plot(pc2_data, col = clust_hier_4, main = "Hierarchical clustering", xlab = "PC1", y
lab = "PC2")
```

The results indicate that hierarchical clustering outperforms the other methods in recovering the actual class labels. This could be attributed to the dataset's nature and the images' inherent structure. K-means clustering showed lower performance, possibly due to its sensitivity to the initial choice of centroids and its tendency to find spherical clusters. The HDDC models, while not outperforming hierarchical clustering, still showed reasonable performance.

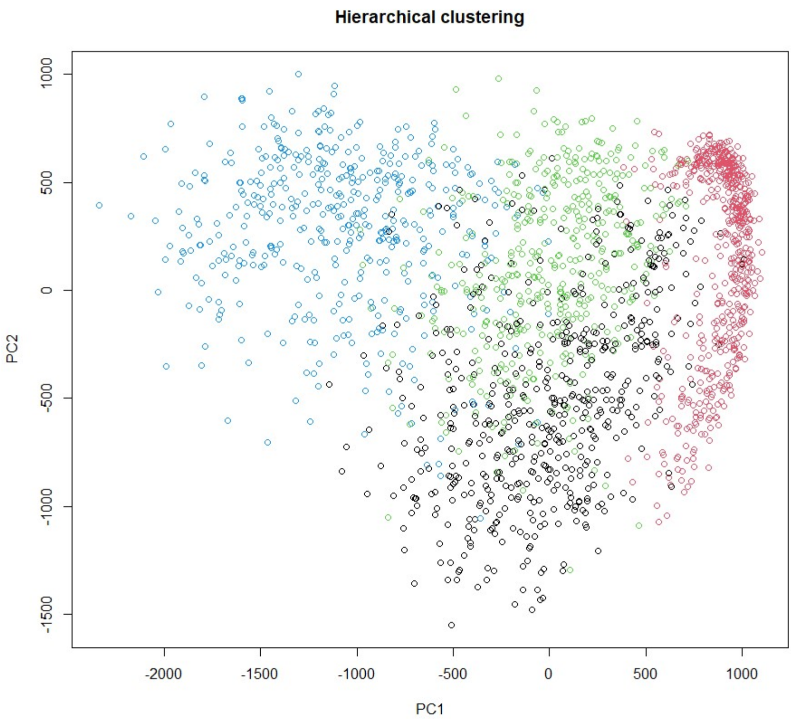

Figure 2. Hierarchical clustering

Task 3

a)

```
> # Aggregate across situations (the second dimension is for situations)
> person_behavior_aggregated <- apply(anger$data, MARGIN =3, FUN = rowSums)
> # compute squared Euclidean distances
> EuclideanDistance <- dist(person_behavior_aggregated, method = "euclidean", diag = T
RUE, upper = TRUE)
> # cluster on squared Euclidean distance
> hiclust_ward<- hclust(EuclideanDistance, "ward.D2")
> par(pty="s")
> plot(hiclust_ward,hang=-1)
```

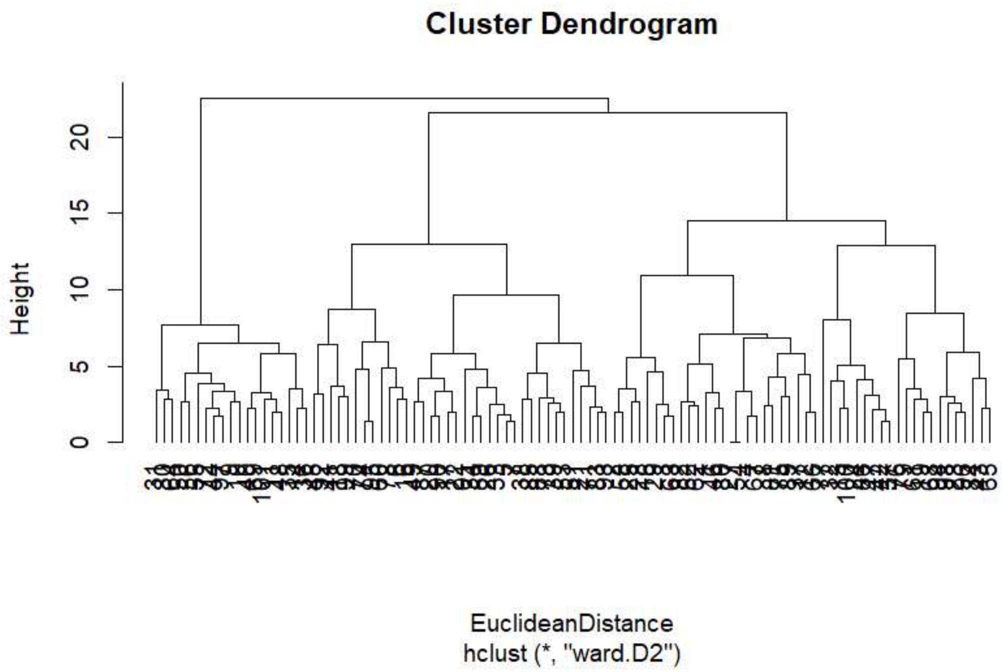

Figure 3.1. Cluster Dendogram

```
> #Save the cluster membership variable of the 2-cluster solution
> clusters <- cutree(hiclust_ward, k = 2)
> clusters
> #centroid
> stat<-describeBy(person_behavior_aggregated, clusters, mat=TRUE)
> hcenter <- matrix(stat[,5],nrow=nclust)
> rownames(hcenter) <- paste("c_",rep(1:nclust),sep="")
> colnames(hcenter) <- c(colnames(anger$freq2))
> round(hcenter,2)
    [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
c_1 2.41 2.28 1.63 2.09 3.13 3.62 2.76 2.55
c_2 1.21 0.68 3.84 4.05 4.21 4.26 1.68 1.58
```

From the dendrogram, we conclude that Ward's method fails to capture the difference in the true modality of the two clusters. Maybe 3 clusters would be a better idea based on the output.

Cluster 2 seems to have higher frequencies for behaviors related to leaving, avoiding, emotional sharing compared to Cluster 1. On the other hand, Cluster 1 generally has lower frequencies for these behaviors and higher frequencies for behaviors related to fighting and making up.

b)

```
> # Combine the aggregated matrix and cluster assignments
> data_with_clusters <- data.frame(person_behavior_aggregated, cluster = clusters)
> profile_vectors <- aggregate(. ~ cluster, data = data_with_clusters, sum)
> profile_vectors <- profile_vectors[, -1
> # Define the new column names
> new_column_names <- c(
+   "fly off the handle",
+   "quarrel",
+   "leave",
+   "avoid",
+   "pour out one's hart",
+   "tell one's story",
+   "make up",
+   "clear up the matter"
+ )
>
> # Assign the new column names to 'profile_vectors'
> colnames(profile_vectors) <- new_column_names
>
> final_freq1 <- rbind(anger$freq1, profile_vectors)
```

	fly off the handle	quarrel	leave	avoid	pour out one's hart	tell one's story	make up	clear up the matter
like	47	61	32	37	54	53	87	78
dislike	31	26	40	52	67	69	21	18
unfamiliar	32	18	46	46	36	54	8	7
higher status	18	15	40	54	77	83	18	21
lower status	62	45	15	18	33	44	69	66
equal status	31	35	34	41	70	75	55	49
1	198	187	134	171	257	297	226	209
2	23	13	73	77	80	81	32	30

c)

```
> #H0: bahabior and situations are statistically independent
> #if the Pearson-Chi square test indicates that Xand Y are statistically
> #dependent, it is meaningful to use CA to further study the nature of the
> #relation between Xand Y.
>
> chisq.test(final_freq1)

        Pearson's Chi-squared test

data:  final_freq1
X-squared = 431.48, df = 49, p-value < 2.2e-16
```

The p-value is small enough to reject null. It is meaningful to use CA to further study the nature of the relation between Xand Y.

```
> ca.out <- ca(final_freq1)
> summary(ca.out)
Principal inertias (eigenvalues):

 dim    value      %   cum%   scree plot
 1      0.087910  85.1  85.1  *********************
 2      0.008238   8.0  93.1  **
 3      0.005522   5.3  98.4  *
 4      0.001318   1.3  99.7
 5      0.000271   0.3  99.9
 6      6.6e-050   0.1 100.0
 7      00000000   0.0 100.0
        -------- -----
 Total: 0.103325 100.0

Rows:
    name   mass  qlt  inr    k=1 cor ctr    k=2 cor ctr
1 | like |  108  892  144 |  348 875 148 |   48  17  30 |
2 | dslk |   78  911   76 | -301 902  80 |  -30   9   8 |
3 | unfm |   59  992  153 | -435 707 127 | -276 285 546 |
4 | hghr |   78  952  150 | -417 875 154 |  124  77 145 |
5 | lwrs |   84  972  215 |  498 942 237 |  -89  30  81 |
6 | eqls |   93  935   15 |   24  35   1 |  121 900 166 |
7 |    1 |  402  872   49 |  104 865  49 |  -10   8   5 |
8 |    2 |   98  872  199 | -427 865 203 |   40   8  19 |

Columns:
    name   mass  qlt  inr    k=1 cor ctr    k=2 cor ctr
1 | flyf |  106  940   94 |  213 499  55 | -201 441 517 |
2 | qrrl |   96  891  102 |  307 863 103 |  -56  29  37 |
3 | leav |   99  881  163 | -379 844 162 |  -79  37  76 |
4 | avod |  119  974  139 | -343 970 159 |  -22   4   7 |
5 | prtn |  161  926   81 | -192 713  68 |  105 213 216 |
6 | tlln |  181  814   71 | -180 804  67 |   20  10   9 |
7 | makp |  124  968  186 |  382 939 205 |   67  29  68 |
8 | clrp |  114  970  165 |  373 936 181 |   71  34  70 |
```

The first three dimensions contribute to 98.4% of total inertia. This means that a three dimensions solution can well represent the dependencies between the rows and the columns of the table.

The squared correlations indicate that the first dimension in row explains most of the inertia for all situations except eqls, which was largely explained by the second dimension. We can see that the second dimension also explains some of the Unfm

We can also see that both dimensions contribute equally to the inertia of flyf. While first dimension explains most of inertia in all other behaviors.

All total quality values are above 800, which means that for all row and column points, inertia is well explained by the two dimensions. This is no surprise because the first two dimensions account for 93.1% of total inertia.

```
> par(pty="s", cex=0.9)
> plot(ca.out, mass = TRUE,arrows = c(TRUE, FALSE),)
```

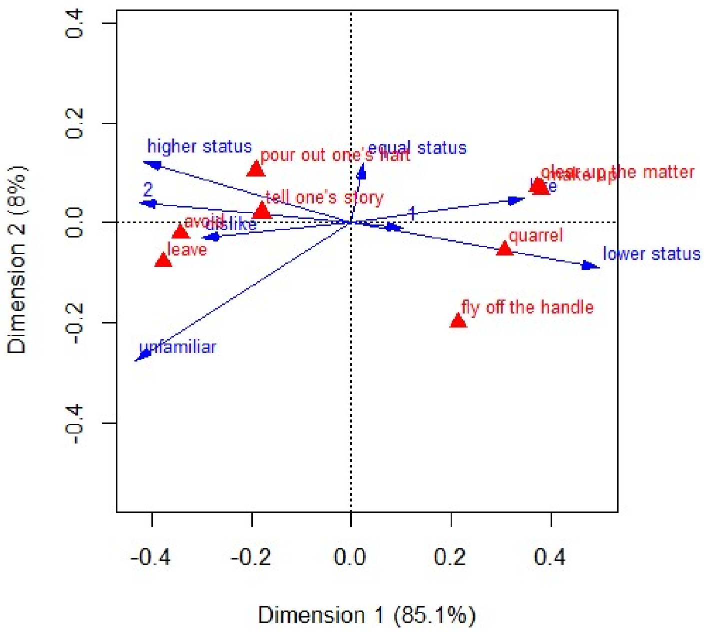

Figure 3.2. Two-dimensional biplot of the situations, behaviors, and clusters.

The plot shows that quarrels and fly off the handle are selected more than average in lower status. Make up and clean up the matter are more selected if they like the person. Avoid and leave are selected more than average if dislike the person. People tend to put out their heart and tell the story if the person has higher status.