KU LEUVEN

Multivariate Statistics

Assignment 2

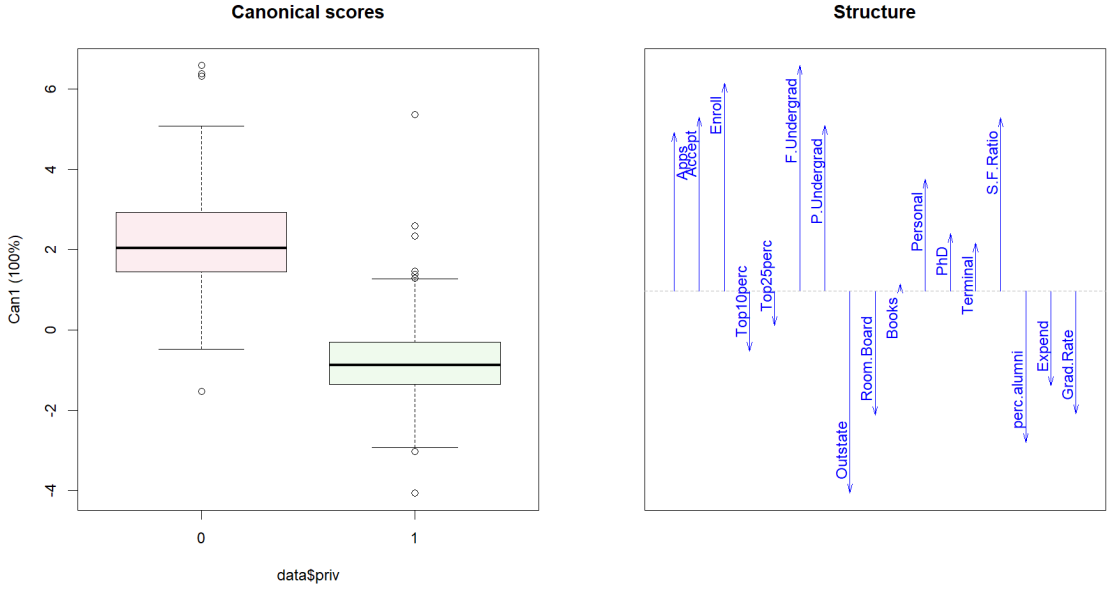
Ana Sofia Mendes - r0925549
Ishika Jain - r0915387
Shreekar Araveti - r0919044
Sounak Ghosh - r0914328

Team_08

MSc Statistics and Data Science

January 2023

Task 1



The function `candisc()` is used to conduct canonical discriminant analysis with private colleges as discriminant function. There are 2 categories of colleges and hence we get one canonical variate. 100% of the variance in college is explained by the only variate (referred to as *Can1*). The null hypothesis of the 2 groups having the same mean is rejected owing to the very small p-value realized from the LR test. It can be seen from the canonical loadings that as the number of students from outstate increases so does the chance of the university not being private owing to its negative canonical p-value. The number of full time students increases the higher the value of the college or, more chances of the college being private owing to its more positive canonical loading. The Figure above maps the influence of data variables on the full-time variable.

```
> round(candisc.out$structure,3)
              Can1
Apps         0.542
Accept       0.596
Enroll       0.712
Top10perc   -0.206
Top25perc   -0.120
F.Undergrad  0.772
P.Undergrad  0.567
Outstate    -0.693
Room.Board  -0.427
Books        0.023
Personal     0.382
PhD          0.197
Terminal     0.163
S.F.Ratio    0.592
perc.alumni -0.520
Expend      -0.324
Grad.Rate   -0.422
```

LDA

```
> lm.out<-lm(cbind(Apps, Accept, Enroll, Top10perc, Top25perc, F.Undergrad, P.Undergrad, Outstate, Room.Board, Books, Persona
l, PhD, Terminal, S.F.Ratio, perc.alumni, Expend, Grad.Rate)~ as.factor(College$Private),data=College)
> summary(manova(lm.out),test=c("Wilks"))
                          Df  Wilks approx F num Df den Df    Pr(>F)
as.factor(College$Private)  1 0.36421   77.937     17    759 < 2.2e-16 ***
Residuals                 775
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We conduct a test based on Wilk's Lambda with the null hypothesis that the centroids of private and non-private colleges are the same. We can clearly see that the null hypothesis is rejected, meaning that the centroids of private and non-private colleges differ significantly, resulting in meaningful discriminant analysis.

Thus, we perform LDA on the untransformed data and obtained the following results, as indicated by prior probabilities, the proportion of private colleges in the sample equals 72.72%. The raw discriminant function coefficients are seen in the figure below (under Coefficients of linear discriminants).

```
Call:
lda(College[, 2:18], College$Private)

Prior probabilities of groups:
       No       Yes
0.2728443 0.7271557

Group means:
        Apps   Accept    Enroll Top10perc Top25perc F.Undergrad P.Undergrad Outstate Room.Board    Books Personal
No  5729.920 3919.288 1640.8726  22.83491  52.70283    8571.005   1978.1887  6813.41   3748.241 554.3774 1676.981
Yes 1977.929 1305.703  456.9451  29.33097  56.95752    1872.168    433.9664 11801.69   4586.143 547.5062 1214.441
         PhD Terminal S.F.Ratio perc.alumni    Expend Grad.Rate
No  76.83491 82.81604  17.13915    14.35849  7458.316  56.04245
Yes 71.09381 78.53451  12.94549    25.89027 10486.354  68.99823

Coefficients of linear discriminants:
                    LD1
Apps        -1.570662e-04
Accept       1.984383e-04
Enroll      -4.930994e-05
Top10perc    1.014785e-02
Top25perc   -9.324913e-04
F.Undergrad -1.359972e-04
P.Undergrad -4.354011e-05
Outstate     2.036160e-04
Room.Board   1.713442e-04
Books        2.821373e-04
Personal     1.490414e-06
PhD         -1.888176e-02
Terminal    -1.869346e-02
S.F.Ratio   -6.858416e-02
perc.alumni  1.252984e-02
Expend      -2.542696e-05
Grad.Rate    7.174711e-03
```

We then perform LDA, on the below-mentioned transformations, and note their performance using the following R code(The structure of the code is the same for all other transformations, just the data set changes to the one with the respective transformation.) Here, we are showing the code for the log-transformed data set followed by standardizing of variable:

```
##########LDA Log-transformation of skewed variables followed by standardizing of all variables
lda.log.std.out<-lda(df_skstd, College$Private)
print(lda.log.std.out)

#hit rate training data
predlda.df_skstd<-predict(lda.log.std.out, df_skstd)
tab<-table(College$Private, predlda.df_skstd$class)
sum(diag(tab))/sum(tab)
df_skstd_hitrate<-sum(diag(tab))/sum(tab)

#loocv hit rate, with prior prob
pred.loocv<-lda(df_skstd, College$Private, CV=TRUE)
tab<-table(College$Private, pred.loocv$class)
sum(diag(tab))/sum(tab)
LOOCVhitrate_log_std<-sum(diag(tab))/sum(tab)
#Sensitivity
LOOCV_sensitivity_log_std<-tab[2,2]/sum(tab[2,])
#Specificity
LOOCV_specificity_log_std<-tab[1,1]/sum(tab[1,])
```

The table below includes the test classification performance (hit rate, sensitivity, specificity) of LDA for the following 4 variable transformation scenarios: (a) Centering of all variables (b) Log-transformation of skewed variables followed by centering of all variables (c) Standardization of all variables (d) Log-transformation of skewed variables followed by standardization of all variables. We use LOOCV predictions to estimate the test performance.

We can note that the test performance is not affected by centering or standardizing the variables, whereas using a (natural) log transformation on skewed variables (indicated in the table provided in the

Model	*LOOCV HitRate*	*Sensitivity*	*Specificity*
*N*ormal	0.9343629344	0.9699115044	0.8396226415
*C*entered	0.9343629344	0.9699115044	0.8396226415
*L*og centered	0.9510939511	0.971681419	0.8962264151
*S*tandardized	0.9343629344	0.9699115044	0.8396226415
*L*og Standardized	0.9510939511	0.971681419	0.8962264151

Table 1

task) followed by standardization/centering of variables improves the predictive performance of the classifiers to some extent, as the LOOCV hit rate increases slightly from 0.934 to 0.951, sensitivity from 0.9699 to 0.9717, and specificity from 0.8396 to 0.8962. Thus, we can conclude that performing log transformation on skewed variables (followed by centering/standardizing) gives the best LOOCV test performance while using LDA (influence of the skewed values decreases significantly, and we beget a better model).

QDA

We perform Box's M test, the test of Box indicates that the null hypothesis of equal covariance matrices across groups is not supported by the data as seen in the figure below. As an alternative. we could use QDA model in which the assumption of equal covariance matrices is relaxed, however QDA does not always have a better out-of-sample performance than LDA.

```
> ##BOX Test
> boxM(College[, 2:18], College[, "Private"])

        Box's M-test for Homogeneity of Covariance Matrices

data:  College[, 2:18]
Chi-Sq (approx.) = 2753.3, df = 153, p-value < 2.2e-16
```

We perform QDA on untransformed data and get the following group means.

```
> #QDA
> qda.out<-qda(College[,2:18],College$Private)
> print(qda.out)
Call:
qda(College[, 2:18], College$Private)

Prior probabilities of groups:
       No       Yes
0.2728443 0.7271557

Group means:
        Apps   Accept    Enroll Top10perc Top25perc F.Undergrad P.Undergrad Outstate Room.Board    Books Personal
No  5729.920 3919.288 1640.8726  22.83491  52.70283    8571.005   1978.1887  6813.41   3748.241 554.3774 1676.981
Yes 1977.929 1305.703  456.9451  29.33097  56.95752    1872.168    433.9664 11801.69   4586.143 547.5062 1214.441
         PhD Terminal S.F.Ratio perc.alumni    Expend Grad.Rate
No  76.83491 82.81604  17.13915    14.35849  7458.316  56.04245
Yes 71.09381 78.53451  12.94549    25.89027 10486.354  68.99823
```

We then perform QDA, on all the transformations, and note their performance using the following R code(The structure of the code is the same for all other transformations, just the data set changes to the one with the respective transformation.) Here, we are showing the code for the log-transformed data set followed by standardizing of variable:

```
#####Log transformed and standardized
qda.out<-qda(df_skstd,College$Private)
print(qda.out)
#hit rate training data
predqda.train<-predict(qda.out,df_skstd)
tab<-table(College$Private,predqda.train$class)
sum(diag(tab))/sum(tab)
qda_4HR<-sum(diag(tab))/sum(tab)
```

```r
#loocv hit rate, with prior prob
pred.loocv<-qda(df_skstd, College$Private, CV=TRUE)
tab<-table(College$Private, pred.loocv$class)
sum(diag(tab))/sum(tab)
LOOCV_HR4<-sum(diag(tab))/sum(tab)
#Sensitivity
LOOCV_sensitivity4<-tab[2,2]/sum(tab[2,])
#specificity
LOOCV_specificity4<-tab[1,1]/sum(tab[1,])
```

The table below includes the test classification performance (hit rate, sensitivity, specificity) of QDA for the above 4 variable transformation scenarios, we use LOOCV predictions to evaluate the performance:

Model	*LOOCV HitRate*	*Sensitivity*	*Specificity*
*N*ormal	0.9009009009	0.9699115044	0.7169811321
*C*entered	0.9009009009	0.9699115044	0.7169811321
*L*og centered	0.9395109395	0.9628318584	0.8773584906
*S*tandardized	0.9009009009	0.9699115044	0.7169811321
*L*og Standardized	0.9395109395	0.9628318584	0.8773584906

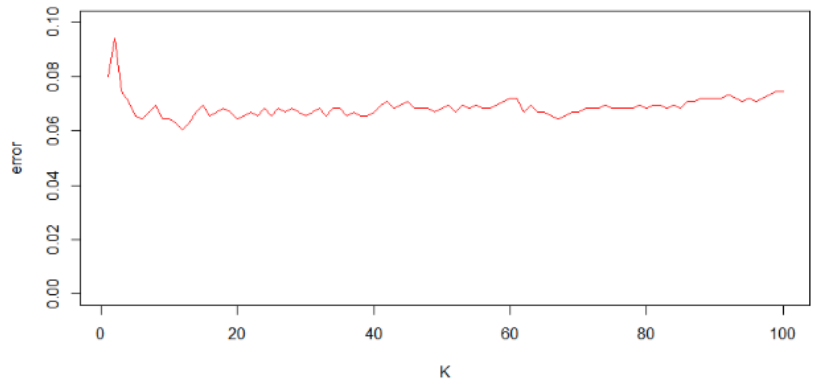
Table 2

We note that, similar to LDA the performance is not affected by centering or standardizing variables. We see that the hit rate and the specificity increased on the log transforming the skewed variables, but the sensitivity is slightly reduced. But overall, the model with a (natural) log transformation on skewed variables seems the one with the best LOOCV test performance.

We can also conclude that the performance of LDA was better than that of QDA after all the transformations.

KNN

We vary k from 1 to 100, to get the k with the least LOOCV error.



For the untransformed model, we get the least LOOCV error for $k = 12$, then we use $k = 12$ to evaluate the performance of the classifier.

We then perform KNN, on all the transformations, and note their performance using the following R code(The structure of the code is the same for all other transformations, just the data set changes to the one with the respective transformation.) Here, we are showing the code for the log-transformed data set followed by standardizing of variable:

```
set.seed(1)
knnmax<-100

err<-matrix(rep(0,knnmax),nrow=knnmax)
for (j in 1:knnmax){
  predknn.loocv<- knn.cv(df_skstd, College$Private, k=j)
  err[j,1]<-1-hitratknn(College$Private,predknn.loocv)
}
which.min(err) #test error is minimum for obs no. 10
plot(-10,-10,xlim=c(1,knnmax),ylim=c(0,0.1),col="red",type="b",xlab="K",ylab="error")
lines(c(1:knnmax),err[,1],col="red")

## K=10
predknn.loocv<- knn.cv(df_skstd, College$Private, k=10)
loocv_err<-1-hitratknn(College$Private,predknn.loocv)
tab<-table(College$Private,predknn.loocv)
knn_hitrate_log_std<-sum(diag(tab))/sum(tab)
#Sensitivity
sensitivity_knn_log_std<-tab[2,2]/sum(tab[2,])
#specificity
specificity_knn_log_std<-tab[1,1]/sum(tab[1,])
```

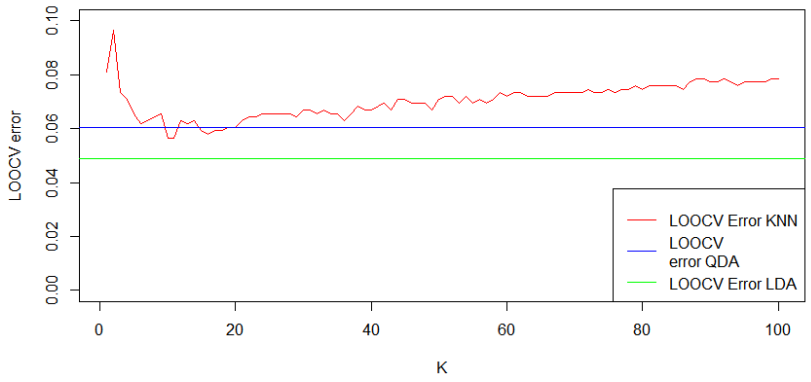
The table below includes the test classification performance (hit rate, sensitivity, specificity) of KNN for the above 4 variable transformation scenarios, we use LOOCV predictions to evaluate the performance (Note: We compute the following by using the value of k for which the LOOCV error is least after transformation):

Model	*LOOCV HitRate*	*Sensitivity*	*Specificity*	*k*
*N*ormal	0.9356499356	0.9699115044	0.8443396226	12
*C*entered	0.9305019305	0.9663716814	0.8349056604	6
*L*og centered	0.9407979408	0.9769911504	0.8443396226	10
*S*tandardized	0.9305019305	0.9663716814	0.8349056604	6
*L*og Standardized	0.9407979408	0.9769911504	0.8443396226	10

Table 3

We can see that unlike LDA, and QDA, the test performance of KNN is affected by the centering and standardizing of variables. The test performance worsened slightly, whereas the log transformation of skewed variables increased the predictive test performance. We can again conclude that the LOOCV test performance of KNN is best for the model with log-transformed skewed variables with $k = 10$.

Comparison of classifiers after using a (natural) log transformation on skewed variables



We can see from the graph that the LOOCV error is the least for LDA, followed by KNN with $k = 10$, and then QDA. The sensitivity is highest for KNN with k=10, followed by LDA, and QDA. Specificity is also the highest for LDA, then QDA, and the least for KNN.

Thus according to the LOOCV test performance, LDA can be called the best classifier among LDA, QDA, and KNN after the log transformation of the skewed variables.

Classifier	*Sensitivity*	*Specificity*
LDA	0.971681419	0.8962264151
QDA	0.9628318584	0.8773584906
KNN	0.9769911504	0.8443396226

Table 4

Bagging and Random forests

To perform bagging and random forest the data is first split into training and validation set, the library `randomForest` has to be installed, to run the function `randomforest`. Unlike a normal tree, bagging takes a bootstrapped data set (data is randomly chosen, with replacement from the data set and this process, some of the data is left behind roughly 1/3rd). Bagging is an iterative algorithm and chooses bootstrapped data time and again, the out of bag error is estimated on the *ith* sample that is not in the bootstrapped data set and the difference is squared and summed over the number of samples that aren't in the bootstrapped data set.

Bagging and Random forest are similar techniques, however, in random forests only a number of regressors are considered (trees modeled on the subset of samples and variables). This report has considered 8 samples - this is because the tree no longer subsets data based on lesser influential data, reducing overfitting, and thereby decreasing its OOB. This can be seen in the tables of random forests and bagging, for the same kind of data (centered, standardized, log-transformed of skewed variable followed by centering and standardizing). Random forests classification has lesser OOB than that of bagging.

Model	*OOBErrorEstimate*	*HitRate*	*Sensitivity*	*Specificity*	*error*
Normal	7.22%	0.933162	0.9618056	0.8514851	0.06683805
Centered	6.96%	0.933162	0.9583333	0.8613861	0.06683805
Log centered	6.96%	0.933162	0.9513889	0.8811881	0.06683805
Standardized	7.73%	0.9280206	0.9513889	0.8613861	0.07197943
Log Standardized	6.96%	0.933162	0.9513889	0.8811881	0.06683805

Table 5: Bagging

```
df_sktrc<-scale(df_sktr[,1:17],center=TRUE, scale= TRUE)
df_sktstc<-scale(df_sktst[,1:17],center=TRUE, scale=TRUE)

set.seed(1)
rf.mod=randomForest(as.factor(target.train)~.,data=df_sktrc,mtry=17,ntree=389,importance=TRUE)
rf.mod

#predictions training data
pred.train<-predict(rf.mod,newdata=df_sktrc,type="prob")

#Bayes classifier
class.train<-ifelse(pred.train>0.5,1,0)
tab<-table(target.train,class.train[,2])
rf.train.equal<-performance(tab)

#hitrate test data
(Bagging_HR0<-sum(diag(tab))/sum(tab))
#Sensitivity
(Bagging_sensitivity<-tab[2,2]/sum(tab[2,]))
#specificity
(Bagging_specificity<-tab[1,1]/sum(tab[1,]))

#predictions test data
(pred.test<-predict(rf.mod,newdata=df_sktstc,type="prob"))
#Bayes classifier
class.test<-ifelse(pred.test>0.5,1,0)
(tab<-table(target.test,class.test[,2]))
rf.test.equal<-performance(tab)
```

```
#hitrate test data
(Bagging_HR0<-sum(diag(tab))/sum(tab))
#Sensitivity
(Bagging_sensitivity<-tab[2,2]/sum(tab[2,]))
#specificity
(Bagging_specificity<-tab[1,1]/sum(tab[1,]))
```

The code above is performed on log standardized data set(bagging) , by changing the value mtry(=8 in this case) random forests can be run.

Realized results on performing Bagging, imply that the out of bag error for unaltered data is higher than that for the centered one, the OOB error is highest for the standardized data. However, the centered, log-transformed values of skewed centered and standardized OOB errors are the same. It is also noted that for validation data, specificity(1-[type 1 error]) values are on an average lesser than sensitivity(1- [type 2 errors]). This implies that proportion of true positives (in this case is a private college) that are correctly classified is higher than that of the proportion of data that is true negative (Is not a private college). Hit rate on testing data is lowest for the standardized model, the other models have a marginally better hit rate and is observed to be the same across all other transformations. The error rate is higher for standardized data, other transformations (and untransformed data) have the same error rate (1- hit rate), observed to be better. It is also observed that sensitivity is marginally better for centered data when compared to other transformations and is negligibly smaller than untransformed data, this will imply that the best transformation would probably be centered transformation. However it is to be noted that there is not a lot of variation in the observed values, other than maybe some betterment in OOB estimate of error rate, on transformations of data by centering and log transformed data, however the standardized data has higher OOB. Suggested transformations would be to center data , log transform (skewed variables) and center or standardize data. Owing to a marginally better hit rate and sensitivity, centered data might have a slight edge over the other transformations.

Model	*OOBErrorEstimate*	*HitRate*	*Sensitivity*	*Specificity*	*error*
*N*ormal	6.70%	0.9280206	0.96180556	0.8316832	0.07197943
*C*entered	6.70%	0.9357326	0.9583333	0.8712871	0.06426735
*L*og centered	6.44%	0.933162	0.9513889	0.8811881	0.06683805
*S*tandardized	6.96%	0.933162	0.9548611	0.8712871	0.06683805
*L*og Standardized	6.44%	0.933162	0.9513889	0.8811881	0.06683805

Table 6: Random forest (mtry=8)

Random forests method of classifications yields a better OOB estimate of error (as seen in the table), this is because unnecessary branches (regressors with lesser influence) are not used to classify data (if features/variables are correlated an unnecessary trees may be added, as may in bagging). The out of bag error estimation is the same for both untransformed and centered data, the standardized data has a higher OOB estimate of error, however the OOB estimate of error for log transformed data (both centered and standardized) is the lowest, this may be because of the reduction of extreme values of skewed variables, it is also observed that there is no difference in the log transformed center and standardized data with regards to hit rate, sensitivity and specificity, nullifying the effect of scaling on log transformed data. It is noted that centered data has the best hit rate followed by standardized data and log-transformed data (the log-transformed and centered data having similar outcomes), followed by the untransformed data. Error rate is observed to be highest for untransformed data, followed by the log transformed data and standardized data (these three transformations have equal error rates), with the best error rate observed in centered data. There is a huge difference between centered and standardized data, with standardized data performing worse than its untransformed counterpart. Implying that skewness of variables has an effect in the model, this is confirmed by the significant log-transformed of OOB estimate error (in case of log transformed data sets).

Owing to the significant betterment in OOB error estimate, it is recommended to make log transformations on the skewed variables and standardizing/ centering it, to get the best random forest model.

Conclusion: Based on the test performance of each classifier, and the discussion of the results done before, we conclude that the LDA classifier has the best hit rate, and specificity, whereas KNN with k=10 has the best sensitivity on the test set. Besides, log transformation (followed by centering and standardizing variables) increases the predictive performance on all classifiers except for bagging. Also, we note that the centering, and standardizing of the variables did not affect the test performance of LDA, and QDA, and have different effects on the rest classifiers.

Task 2

a.

HddClassif

We start by centering the training data (*train.data*). We use the function `hddc()` to estimate cluster solutions when fitting the models *AkjBkQkD* and *AkjBQkD*, with common dimensions from 2 to 6. Since we have the true class labels available (*train.target*), we can use it to evaluate the performance of the unsupervised training procedure.

We know that *AkjBkQkDk* is the most general model, while *ABQD* is the most constrained, and that we can build more parsimonious models by constraining the parameters to be common within or between classes. In our case, the models only differ in the `Bk` parameter (noise of the classes subspaces), where if Bk: each class has its proper noise; if B: all classes have the same noise. Therefore, we see that *AkjBQkD* is a more parsimonious model than *AkjBkQkD* since it has more constraints.

```
set.seed(1)

### using HDClassif
dim <- c(2,3,4,5,6)

### with model "AkjBkQkD"
ari_model1 <- rep(0,5)

for (i in dim){
    hddc_model1.out<-hddc(X,K=3,model="AkjBkQkD", com_dim=i)
    hddc_model1.out
    #compute classification table
    tab1<-table(hddc_model1.out$class,class)
    tab1
    #adjusted rand index
    j <- i-1
    ari_model1[j] <- adjustedRandIndex(hddc_model1.out$class,class)}

print(ari_model1)

### same process with model "AkjBQkD"
ari_model2 <- rep(0,5)

for (i in dim){
hddc_model2.out<-hddc(X,K=3,model="AkjBQkD", com_dim=i)
hddc_model2.out
tab6<-table(hddc_model2.out$class,class)
hddc_model2.out
j <- i-1
ari_model2[j] <- adjustedRandIndex(hddc_model2.out$class,class)}
print(ari_model2)
```

Table 7 gathers the ARI values obtained with the different dimensions, for both models:

ARI	$n = 2$	$n = 3$	$n = 4$	$n = 5$	$n = 6$
AkjBkQkD	0.8201029	0.8534184	0.8776449	0.8694625	0.8730921
AkjBQkD	0.9214278	0.9285283	**0.9367286**	0.9141156	0.9215654

Table 7

We conclude that the model *AkjBQkD* is more stable since ARI values are higher compared to the other model, for each common dimension. Nevertheless, all models show an ARI value higher than 0.8, meaning that they can recover the true class labels well.

The best recovery was obtained with common dimension 4 and model *AkjBQkD*, with an ARI value of 93.67%. We can do further analysis on it:

```
> hddc_best.out<-hddc(X,K=3,model="AkjBQkD", com_dim=4)
> hddc_best.out

HIGH DIMENSIONAL DATA CLUSTERING
MODEL: AKJBQKD
  Posterior probabilities of groups
      1     2     3
  0.333 0.333 0.334
     Intrinsic dimensions of the classes:
      1 2 3
  dim: 4 4 4

Class      a1     a2     a3     a4
    1  365389 262104 158474 133007
    2 1033086 223477 188282 121671
    3  394536 224492 103867  79935

B: 1109
BIC:  -139519143
> table_hddc<-table(hddc_best.out$class,class)
> mapClass(hddc_best.out$class,class)
> table_hddc<-table_hddc[c(2,1,3),]
> table_hddc
   class
       0    1    7
  2 5847  236    3
  1  147 5764    0
  3    6    0 5997
> #percentage of correct classification
> sum(diag(table_hddc))/sum(table_hddc)
[1] 0.9782222

> hddc_best.out$loglik
[1] -69702027
> hddc_best.out$complexity
[1] 11746
```

We confirm the common dimensions of the model, as it uses 4, 4 and 4 dimensions to model the covariance matrices of the three clusters. The percentage of correct classification equals 97.82%, which is very high. The model includes 11746 parameters and has a loglikelihood = -69702027. The BIC of the model equals -139519143.

MClust

When using the function `cumsum()` we can see the percentage of variance the first k principal components account for. We obtain that the first 2 unstandardized PC account for 59% of the variance, the first 3 account for 65%, the first 4 account for 68%, the first 5 account for 71%, and the first 6 account for 72%.

All the models being analyzed (*VVE*, *VEV*, *EVV* and *VVV*) have an ellipsoidal distribution, going from the least to most complex model.

```
indices <- c(2,3,4,5,6)
```

```
#3-cluster model for VVE model
ari_vve <- rep(0,5)
for (i in indices){
mclust_vve.out<-Mclust(comp[,1:i],G=3, modelNames="VVE")
summary(mclust_vve.out)
tab1m<-table(class,mclust_vve.out$classification)
tab1m
mapClass(mclust_vve.out$class,class)
j <- i-1
ari_vve[j] <- adjustedRandIndex(mclust_vve.out$class,class)}
print(ari_vve)

#for VEV model
ari_vev <- rep(0,5)
for (i in indices){
mclust_vev.out<-Mclust(comp[,1:i],G=3, modelNames="VEV")
summary(mclust_vev.out)
tab2m<-table(class,mclust_vev.out$classification)
tab2m
mapClass(mclust_vev.out$class,class)
j <- i-1
ari_vev[j] <- adjustedRandIndex(mclust_vev.out$class,class)}
print(ari_vev)

#for EVV model
ari_evv <- rep(0,5)
for (i in indices){
mclust_evv.out<-Mclust(comp[,1:i],G=3, modelNames="EVV")
summary(mclust_evv.out)
tab3m<-table(class,mclust_evv.out$classification)
tab3m
mapClass(mclust_evv.out$class,class)
j <- i-1
ari_evv[j] <- adjustedRandIndex(mclust_evv.out$class,class)}
print(ari_evv)

#for VVV model
ari_vvv <- rep(0,5)
for (i in indices){
mclust_vvv.out<-Mclust(comp[,1:i],G=3, modelNames="VVV")
summary(mclust_vvv.out)
tab3m<-table(class,mclust_vvv.out$classification)
tab3m
mapClass(mclust_vvv.out$class,class)
j <- i-1
ari_vvv[j] <- adjustedRandIndex(mclust_vvv.out$class,class)}
print(ari_vvv)
```

The results are summarized in Table 8:

ARI	*VVE*	*VEV*	*EVV*	*VVV*
First 2 PCA	0.7553523	0.7989124	0.8067490	0.8121480
First 3 PCA	0.8860274	0.8320214	0.7545344	0.8631980
First 4 PCA	0.8793628	0.8552251	0.8465507	0.8720528
First 5 PCA	0.8929202	0.8937315	0.9247805	0.9285782
First 6 PCA	0.9302037	0.9233363	**0.9409102**	0.9192532

Table 8

In general, using more unstandardized PC leads to a more stable cluster solution (in general, the ARI increases when more PC are being used, for all four models). Comparing the models, for the same number of PC used, there's no clear conclusion about a model being better or worse.

The best cluster solution was obtained when using the first 6 PC and the model *EVV*, with an ARI value of 94.09%. As done for the HDDClassification, we can further analyse this model:

```
> #best MClust model
> mclust_best.out<-Mclust(comp[,1:6],G=3, modelNames="EVV")
> summary(mclust_best.out)
```

```
Gaussian finite mixture model fitted by EM algorithm
```

```
Mclust EVV (ellipsoidal, equal volume) model with 3 components:

 log-likelihood      n df       BIC       ICL
      -750946.8 18000 81 -1502687 -1502901

Clustering table:
   1    2    3
5870 6115 6015
> table_mclust<-table(class,mclust_best.out$classification)
> mapClass(mclust_best.out$class,class)
> table_mclust<-table_mclust[c(2,1,3),]
> table_mclust

class    1    2    3
    1 5756  244    0
    0  114 5869   17
    7    0    2 5998
> #percentage of correct classification
> sum(diag(table_mclust))/sum(table_mclust)
[1] 0.9790556
```

The model includes 81 parameters and has a loglikelihood=-750946.8. The BIC of the model equals -1502687.

b.

We plot the observed clusters and the derived clusters with the best model obtained previously (using `mclust()` with the first 6 unstandardized PC and the model EVV) in the space of the first two principal components. Recall that the first two unstandardized components account for 41.5% and 17.3% of the variance, respectively.

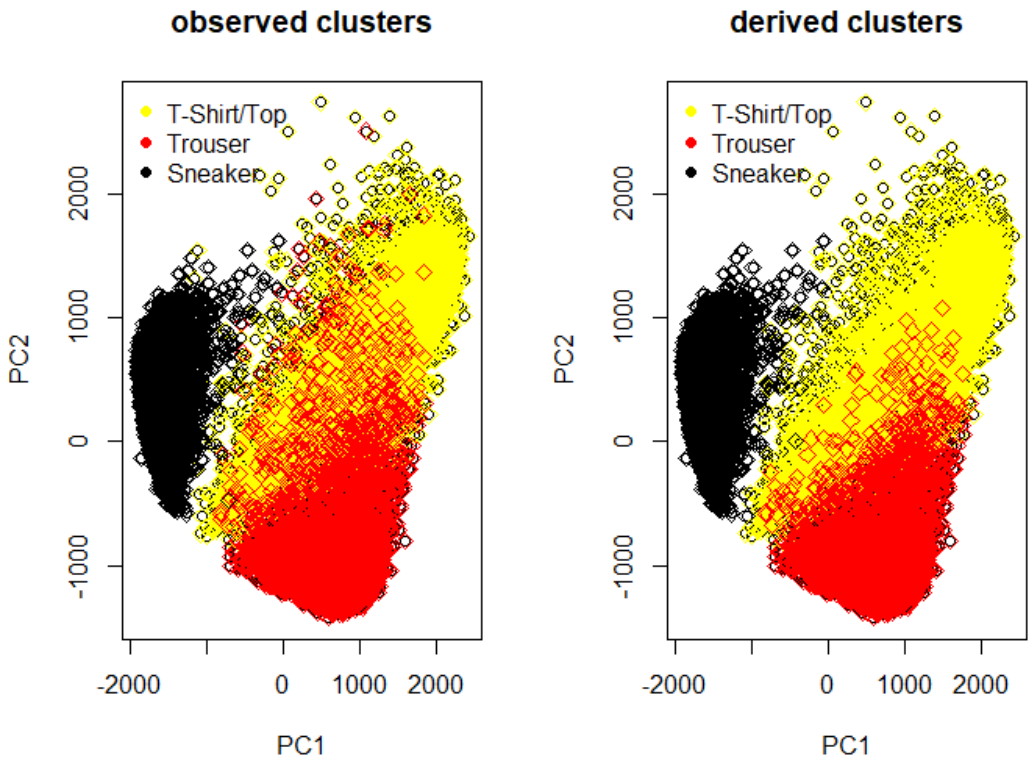
```
### PCA plot
#conduct PCA on the data - done before (not necessary)
prcomp.out<-prcomp(X)
comp<-as.matrix(X)%*%prcomp.out$rotation

#best model
mclust_best.out<-Mclust(comp[,1:6],G=3, modelNames="EVV")
mapClass(mclust_best.out$class,class)

par(mfrow=c(1,2))
#plot observed clusters
plot(comp,main="observed clusters")
points(comp[class==0,],col="yellow",pch=5)  #put the order when we change it with ntab
points(comp[class==1,],col="red",pch=5)
points(comp[class==7,],col="black",pch=5)
legend("topleft",c("T-Shirt/Top","Trouser","Sneaker"),col=c("yellow","red","black"),pch=c(19,19,19),bty="n")

#plot clusters extracted with HDDC in space of first two principal components
#use the clustering model with the best value of ARI
plot(comp,main="derived clusters")
points(comp[mclust_best.out$class==3,],col="yellow",pch=5)
points(comp[mclust_best.out$class==1,],col="red",pch=5)
points(comp[mclust_best.out$class==2,],col="black",pch=5)
legend("topleft",c("T-Shirt/Top","Trouser","Sneaker"),col=c("yellow","red","black"),pch=c(19,19,19),bty="n")
```

In the graphics below we present the observed and the derived clusters:

The graphs confirm that the selected model can recover the true cluster structure rather well. However, we observe that a considerable amount of data that was observed in the "Trouser" cluster was labeled as belonging to the "T-shirt/Top" cluster. This can be due to the fact that, in the space of the first 2 PC, observations from these two clusters show similar characteristics (overlap between the yellow and red clusters). Therefore, it is more difficult for the model to recover the true labels.

By this logic, it makes sense that the cluster "Sneakers" was the one that was better recovered with the model (characteristics are more clear and distinct).

Additionally, we conduct principal components analysis on the standardized variables, and compute the VAF of the components:




```
#standardize training data
X_std <-scale(train.data,center=TRUE,scale=TRUE)
prcomp.out_std<-prcomp(X_std)

#VAF components
round(head(prcomp.out_std$sd^2/sum(prcomp.out_std$sd^2)),3)
[1] 0.273 0.136 0.058 0.047 0.032 0.021
```

The first two principal components account for 27.3% and 13.6% of the variance in the data. Together, they account for 40.9% of the variance.

Task 3

a.

We load the data set `rectangles` and the data set `rect_constr` which consists of the width and height of the 16 rectangles using grid, and we standardize the `rect_constr`.

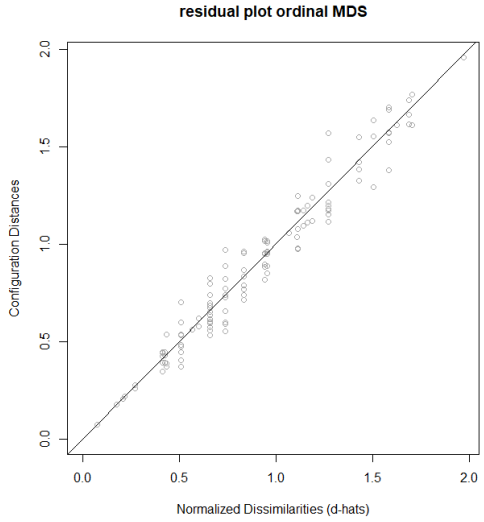
```
library(smacof)
data(rectangles)
data(rect_constr)
zrect_constr<-scale(rect_constr,center=TRUE,scale=TRUE)
```

We use `smacofSym()` to fit two-dimensional models using ratio, interval, ordinal, and mspline measurement levels. The *Torgerson* solution is used as rational starting point.

```
#ratio
m1<-smacofSym(delta=rectangles, ndim=2, type="ratio", init="torgerson")
```

```
#interval
m2<-smacofSym(delta=rectangles, ndim=2, type="interval", init="torgerson")
#ordinal
m3<-smacofSym(delta=rectangles, ndim=2, type="ordinal", init="torgerson")
#mspline
m4<-smacofSym(delta=rectangles, ndim=2 ,type="mspline",spline.degree =4 , spline.intKnots = 4, init="torgerson")
#stress-1 values
round(c(m1$stress ,m2$stress ,m3$stress ,m4$stress),3)
0.153  0.127  0.089  0.107
```

The results show that stress-1 is lowest using ordinal MDS. A residual plot of optimally scaled dissimilarities versus configuration distances for the ordinal solution shows that there is very low variation in transformed dissimilarities with the same configuration distance. The correlation between transformed dissimilarities and configuration distances equals 0.98.

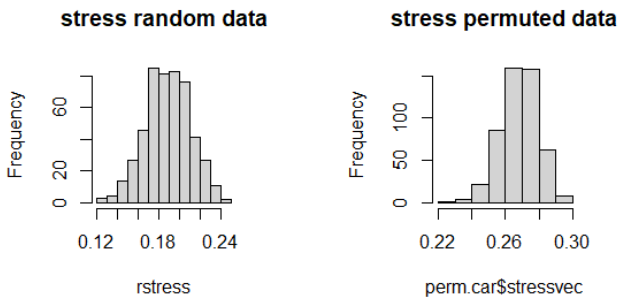


b.

We use stress norms and a permutation test to investigate if the ordinal MDS solution has satisfactory fit.

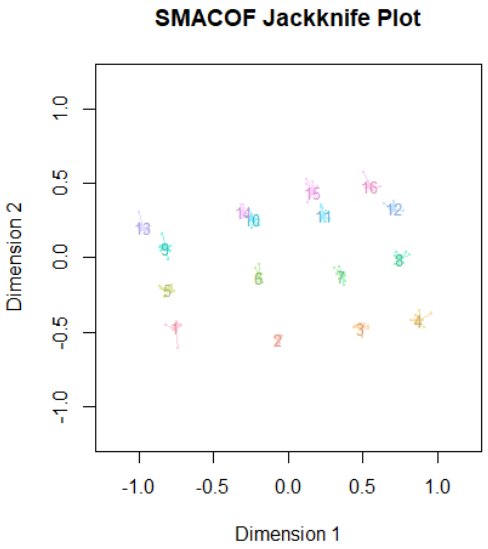
```
#ordinal MDS
#stress norm
set.seed(1)
rstress<-randomstress(n=10,ndim=2,nrep=500,type="ordinal")
#distribution of stress for random data
mean(rstress)-2*sd(rstress)
0.1450965

#permutation test
set.seed(1)
perm.car<-permtest(m3, nrep=500)

#plot distribution stress
par(mfrow=c(1,2), pty="s")
hist(rstress ,main="stress random data")
hist(perm.car$stressvec ,main="stress permuted data")
```

The observed stress-1 value of the ordinal MDS solution (0.089) is smaller than the cutoff of $\overline{x}_r - 2\hat{\sigma}_r = 0.246$ obtained from the distribution of stress-1 using random data. This means that the ordinal MDS solution has a satisfactory fit. The results for the other MDS solutions are less relevant, since stress-1 is lowest for the ordinal MDS solution.



Next we assess the stability of the ordinal MDS solution using the jackknife:

```
#stability of solution using jackknife
jack.car<-jackmds(m3)
plot(jack.car,xlim=c(-1.2,1.2),ylim=c(-1,1))
```

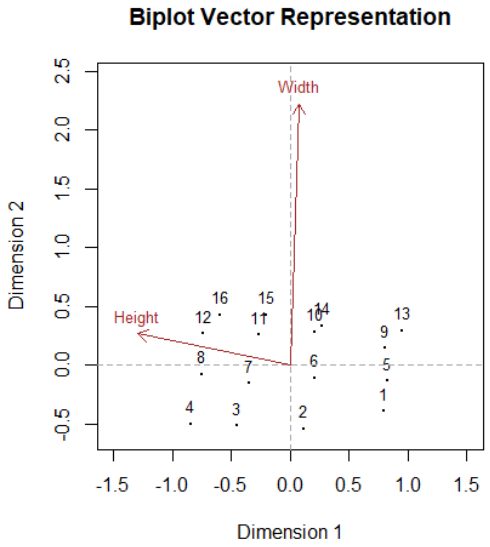


The Jackknife plot indicates that the points in the configuration have a stable position The stability measure reported is 0.998, which also indicates stability.

c.

We use the function biplotmds() to run multivariate linear regression of the standardized external variables on the object coordinates obtained with model m3. Next, we make an MDS biplot that projects the external variables in the configuration of the signals.

```
biRect <- biplotmds(m3, zrect_constr)
coef(biRect)
plot(biRect, main = "Biplot Vector Representation", vecscale = 0.8,
     xlim = c(-1.5, 1.5), vec.conf = list(col = "brown"), pch = 20, cex = 0.5)

#print R-square of regressions
round(biRect$R2vec,3)
 Width Height
 0.945  0.987

#print regression coefficients
round(biRect$coefficients,3)
    Width Height
D1  0.096 -1.623
D2  2.778  0.334
```




```
#correlation of rect_constr and dimensions
round(cor(m3$conf,zrect_constr),3)
    Width Height
D1  0.117 -0.987
D2  0.970  0.056
```

The MDS biplot indicates that the first dimension separates rectangles by height and the second dimension by width (in lowest and highest values). The first dimension has a strong negative correlation (-0.987) with *height*, while the second dimension has a strong positive correlation (0.970) with *width*.

The multivariate regression of the variables *width* and *height* on the configuration coordinates indicates that the *width* variable can be very well explained by the configuration of the rectangles. Linear regression of *width* on the configuration coordinates yields an R^2 of 0.945. The predicted *width* increases in the direction indicated by the vector of the variable *width*. In particular, if D1 increases 1 SD the predicted Width increases by 0.096 and if D2 increases 1 SD the predicted Width increases by 2.778. The variable *height* is represented in the configuration as well. Linear regression of *height* on the configuration coordinates yields an R^2 of 98.7%. The predicted *height* increases in the direction indicated by the vector of the variable *height*. In particular, if D1 increases 1 SD the predicted *width* decreases by 1.623 and if D2 increases 1 SD the predicted *height* increases by 0.334.

```
> #regressing width on D1 and D2
summary(lm(zrect_constr[,1]~m3$conf))

Call:
lm(formula = zrect_constr[, 1] ~ m3$conf)

Residuals:
      Min        1Q    Median        3Q       Max
-0.36586 -0.17280 -0.00103  0.16890  0.39448

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 7.117e-17  6.299e-02   0.000    1.000
m3$confD1   9.624e-02  1.064e-01   0.904    0.382
m3$confD2   2.778e+00  1.873e-01  14.834 1.58e-09 ***
---
Signif. codes:  0    ***    0.001    **    0.01    *    0.05    .    0.1         1

Residual standard error: 0.2519 on 13 degrees of freedom
Multiple R-squared:  0.945,     Adjusted R-squared:  0.9365
F-statistic: 111.7 on 2 and 13 DF,  p-value: 6.501e-09
```

Regression of width on D1 and D2 using the `lm()` function shows that only the regression coefficient D2 is significant. On the other hand, when regressing height on D1 and D2 we conclude that both regressors are significant.