**Solution assignment 2 Multivariate Statistics 2022-2023**

**Task 1**

We first load the necessary packages and compute data sets for the four cases,

data.raw.center: centering of all variables
data.raw.std: standardization of all variables
data.trans.center: log-transformation of skewed variables followed by centering of all variables
data.trans.std: log-transformation of skewed variables followed by standardization of all variables.

```
library(ISLR2)
library(randomForest)
library(MASS)
library(class) ##knn()
library(car)

data(College)
target<-College[,1]
data.raw.center<-scale(College[,2:18],center=TRUE,scale=FALSE)
data.raw.std<-scale(College[,2:18],center=TRUE,scale=TRUE)

data.trans<-College[,2:18]
data.trans[,c(1,2,3,4,6,7,10,11,16)]<-
log(data.trans[,c(1,2,3,4,6,7,10,11,16)])
data.trans.center<-scale(data.trans,center=TRUE,scale=FALSE)
data.trans.std<-scale(data.trans,center=TRUE,scale=TRUE)
```

Next, we write a general function “classif” that can compute the requested out-of-sample performance measures for LDA, QDA, Bagging, Random Forest (RF) and KNN given a certain input data set and the target variable. That is, for LDA, QDA, and KNN we use LOOCV predictions and for Bagging and RF we use OOB predictions. Furthermore, for KNN we use K=1,..100 and select the value of k for which the LOOCV hit rate is highest. For both Bagging and RF we use 5000 trees. For Bagging we use all 17 variables as candidates in each split (mtry=17), for RF we use a random sample of 4 variables in each split (mtry=4). Note that we selected mtry=4 as the square root of the total number of variables is the recommended value. Finally we note the we always use the Bayes classifier to compute predicted classifications as this will lead to the highest out-of-sample hitrate.

```
classif<-function(data,target){
#function to compute performance measures
performance<-function(tab){
hitrate<-sum(diag(tab))/sum(tab)
sensitivity<-tab[2,2]/(tab[2,1]+tab[2,2])
specificity<-tab[1,1]/(tab[1,1]+tab[1,2])
performance<-c(hitrate=hitrate,
sensitivity=sensitivity,specificity=specificity)
}

#LOOCV error bayes classifier for LDA
pred.test<-lda(data,target,CV=TRUE)
lda.test<-performance(table(target,pred.test$class))

#LOOCV error bayes classifier for QDA
pred.test<-qda(data,target,CV=TRUE)
qda.test<-performance(table(target,pred.test$class))

#Bagging
set.seed(1)
bag.mod=randomForest(target~.,data=data.frame(target,data),mtry=17,
ntree=5000,importance=TRUE)
#OOB error bayes classifier
class.test<-ifelse(bag.mod$votes[,2]>0.5,1,0)
bag.test<-performance(table(target,class.test))

#Random Forest
set.seed(1)
rf.mod=randomForest(target~.,data=data.frame(target,data),mtry=4,
ntree=5000,importance=TRUE)
#OOB error bayes classifier
class.test<-ifelse(rf.mod$votes[,2]>0.5,1,0)
rf.test<-performance(table(target,class.test))

#knn
knnmax<-100
outknn<-matrix(rep(0,knnmax*3),nrow=knnmax)
set.seed(1)
```

```
for (j in 1:knnmax){
predknn<- knn.cv(data, target, k=j)
outknn[j,]<-performance(table(target,predknn))
}
ksel<-which.min(1-outknn[,1])
knn.test<-outknn[ksel,]
classif<-rbind(lda.test,qda.test,bag.test,rf.test,knn.test)
}
```

Next, we use classif() to compute the performance measures for each of the four data sets and we write a function to analyze visualize the results for each performance measure (see Figures 1,2,3).

```
#apply "classif" function to each case
raw.center.out<-classif(data.raw.center,target)
raw.std.out<-classif(data.raw.std,target)
trans.center.out<-classif(data.trans.center,target)
trans.std.out<-classif(data.trans.std,target)

#visualize results
plotresult<-function(column,statistic,lower,upper){
plot(-1,-1,xlim=c(1,5),ylim=c(lower,upper),axes=FALSE,xlab="method",
ylab=statistic, main=statistic)
axis(1,at=c(1:5),labels=c("LDA","QDA","Bagging","RF","KNN"))
axis(2)
points(c(1:5),raw.center.out[,column],col="red",pch=1)
lines(c(1:5),raw.center.out[,column],col="red",lty=1)
points(c(1:5),raw.std.out[,column],col="red",lty=2,pch=2)
lines(c(1:5),raw.std.out[,column],col="red",lty=2)
points(c(1:5),trans.center.out[,column],col="blue",pch=1)
lines(c(1:5),trans.center.out[,column],col="blue",lty=1)
points(c(1:5),trans.std.out[,column],col="blue",pch=2)
lines(c(1:5),trans.std.out[,column],col="blue",lty=2)
legend("bottomleft",c("raw.center","raw.std","trans.center",
"trans.std"),col=c("red","red","blue","blue"),pch=c(1,2,1,2),lty=c(1
,2,1,2))}


plotresult(1,"Hitrate",0.8,1)
plotresult(2,"sensitivity",0.85,1)
plotresult(3,"specificity",0.5,1)
```

Figure 1

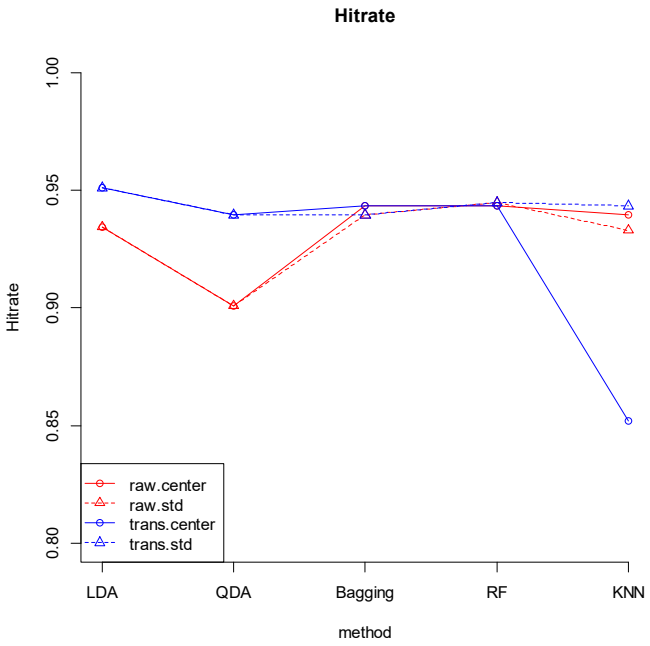



Figure 2

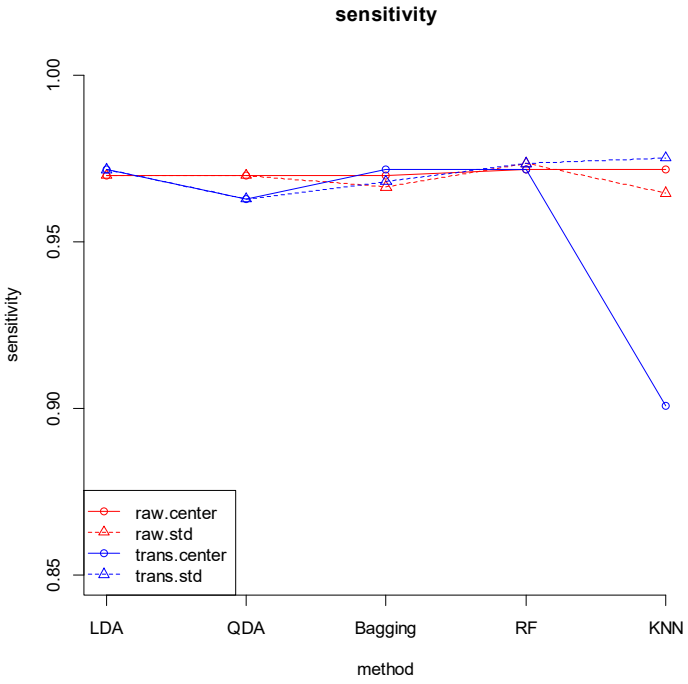
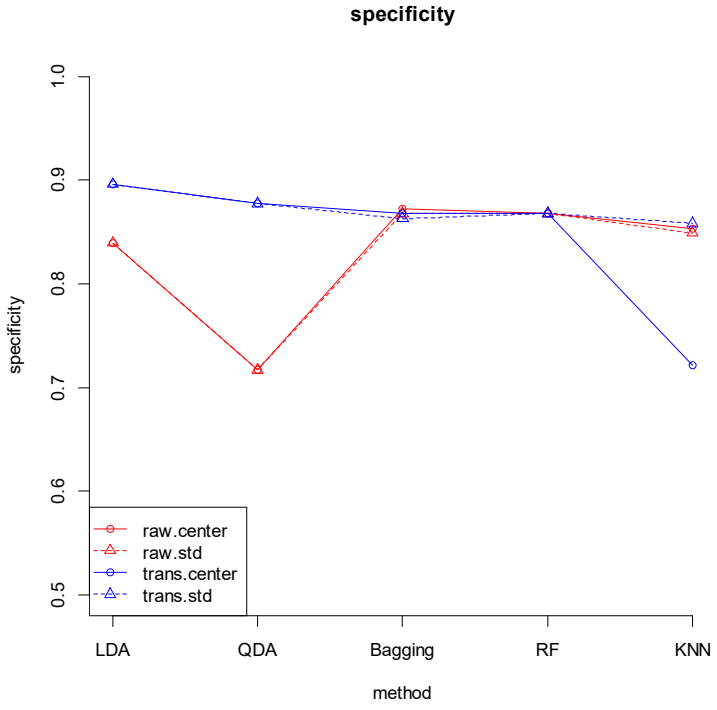



Figure 3



From the graphs we can draw the following conclusions about the out-of sample performance of the methods on the different perormance measures, and how transforming the variables affects this out-of-sample performance:

- For the **hitrate and the specificity** we see that the different variable transformations have a similar effect on the performance of the methods.
    - Using a log transformation of skewed variables increases the hitrate and specificity for LDA and QDA. For QDA the increase is larger than for LDA. In contrast using a log-transformation on skewed variables + centering decreases the hitrate and the specificity for KNN (compared to analysis on raw centered data), whereas using a log-transformation on skewed variables + standardization does not have this negative effect on the hitrate and specificity for KNN. Finally, the performance of Bagging and RF are not affected by using a log-transformation of skewed variables.
- For the **sensitivity**, we see that all methods have a rather similar performance, except for KNN which has a lower sensitivity when using a log-transformation on skewed variables + centering.
- **Centering versus standardizing** the data yields exactly the same performance measures for LDA and QDA, which means these methods are scale-invariant. Also for the Bagging and RF centering versus standardizing does not affect the performance.
- **LDA after using a log transformation on skewed variables has the highest out-of-sample hitrate**, but the difference with the other methods is rather small.

**Task 2 (a)**

We load the necessary packages and the data.

```
library(HDclassif)
library(mclust)
load("fashion.Rdata")
```

Next we center the variables in the training set, and we use the prcomp() function to conduct principal components analysis on the training data.

```
#center training and test data
ctrain.data<-scale(train.data,center=TRUE,scale=FALSE)

#PCA
prcomp.out<-prcomp(ctrain.data)
totvar<- sum(apply(ctrain.data,2,var))
propvar<-prcomp.out$sdev^2/totvar
cumpropvar<-cumsum(propvar)
> cumpropvar[6]
[1] 0.7241621
cumpropvar[50]
[1] 0.8985033
```

It turns out that the first 6 principal components account for 72.4% of the variance in the training data set. The first 50 principal components account for almost 89.9% of the variance in the training data set. We compute the first 6 unstandardized principal components.

```
#compute first 6 unstandardized principal components
ncomp<-6
train.comp<-ctrain.data%*%prcomp.out$rotation[,1:ncomp]
```

Next we estimate all the requested models and compute the adjusted rand index for each model. The computation is done using a loop where we take the input parameters of each model as input in different iterations. Note that all models extract 3 clusters from the training data (which is the true number of clusters in the data).

As in model-based clustering the solution is generally not unique (i.e, the likelihood surface can have different local maxima), we should be careful when choosing the starting point of the algorithm. For hddc() models we first compute the k-means solution and always use this as a rational starting point for all hddc models.

For Mclust, the default is to start from the hierarchical clustering solution computed on a sample of 2000 observations. We keep this option because conducting hierarchical clustering on the full data set is not feasible. As the procedure involves taking a sample of 2000 observations to compute the starting point, different runs can lead to different results. To obtain stable results we conduct for each model 20 runs, compute the ARI for each run, and then compute the highest ARI value across the 20 runs.

```
#conduct k-means to obtain starting point of hddc
km.init<-kmeans(ctrain.data, centers=3, iter.max = 100, nstart = 20,
algorithm = c("Hartigan-Wong"), trace=FALSE)

for (i in 1:10){
hddc.out<-
hddc(ctrain.data,K=3,com_dim=resulthddc[i,2],model=resulthddc[i,1],
init.vector=km.init$cluster)
resulthddc[i,3]<-round(adjustedRandIndex(hddc.out$class,train.target),3)}

#make matrix with mclust models that need to be estimated
mclust.options(subset=2000)
ari.runs<-matrix(rep(NA,20*20),ncol=20)
resultmclust<-
data.frame(model=c(rep("VVE",5),rep("VEV",5),rep("EVV",5),rep("VVV",5)),
ncomp=c(2:6,2:6,2:6,2:6),ari=ari.runs)
for (i in 1:20){
for (j in 1:20){
mclust.out<-
Mclust(train.comp[,1:resultmclust[i,2]],G=3,modelNames=resultmclust[i,1])
resultmclust[i,2+j]<-
round(adjustedRandIndex(train.target,mclust.out$classification),3)}}
resultmclust$best<-apply(resultmclust[,3:22],1,max)
```

Next we visualize the computed ARI of all models in one graph (see Figure 4). As can be seen in the graph, the models than can capture the observed cluster structure best are HDDC model AKjBQkD with com_dim=4 (ARI=.937) and MCLUST model EVV fitted on the first 6 principal components (ARI=.942).

```
plot(-1,-1,xlim=c(1,5),ylim=c(0.7,1),axes=FALSE,xlab="com_dim",ylab="ARI")
axis(1,at=c(1:5),labels=c(2:6))
axis(2)
points(c(1:5),resulthddc[1:5,3],col="red",pch=1)
lines(c(1:5),resulthddc[1:5,3],col="red",lty=1)
points(c(1:5),resulthddc[6:10,3],col="red",pch=2)
lines(c(1:5),resulthddc[6:10,3],col="red",lty=2)
points(c(1:5),resultmclust[1:5,"best"],col="blue",pch=1)
lines(c(1:5),resultmclust[1:5,"best"],col="blue",lty=1)
points(c(1:5),resultmclust[6:10,"best"],col="blue",pch=2)
lines(c(1:5),resultmclust[6:10,"best"],col="blue",lty=2)
points(c(1:5),resultmclust[11:15,"best"],col="blue",pch=3)
lines(c(1:5),resultmclust[11:15,"best"],col="blue",lty=3)
points(c(1:5),resultmclust[16:20,"best"],col="blue",pch=4)
lines(c(1:5),resultmclust[16:20,"best"],col="blue",lty=4)
legend("bottomright",c("HDDC AkjBkQkD","HDDC AkjBQkD","MCLUST VVE","MCLUST
VEV","MCLUST EVV", "MCLUST VVV"),
col=c("red","red","blue","blue","blue","blue"),
pch=c(1,2,1,2,3,4),lty=c(1,2,1,2,3,4))
```

**Task 2 (b)**

We visualize the observed class labels and the predicted class labels for the best model (i.e. with the highest ARI) in the space of the first two principal components. (see Figure 4)

```
#best model is Mclust model EVV on fitted on the first 6 principal
components
set.seed(5) #choose seed so that ARI=.924
mclust.options(subset=2000)
mclust.out<-Mclust(train.comp[,1:6],G=3,modelNames="EVV")
print(round(adjustedRandIndex(train.target,mclust.out$classification),3))
#plot observed and predicted class labels in the space of the first two
principal components
plot(train.comp[,1:2],pch='',main="observed class labels")
points(train.comp[train.target==0,1:2],col="red",pch='.')
points(train.comp[train.target==1,1:2],col="blue",pch='.')
points(train.comp[train.target==7,1:2],col="black",pch='.')
plot(train.comp[,1:2],pch='',main="predicted class labels")
points(train.comp[mclust.out$classification==1,1:2],col="blue",pch='.')
points(train.comp[mclust.out$classification==2,1:2],col="black",pch='.')
points(train.comp[mclust.out$classification==3,1:2],col="red",pch='.')
```

Figure 4

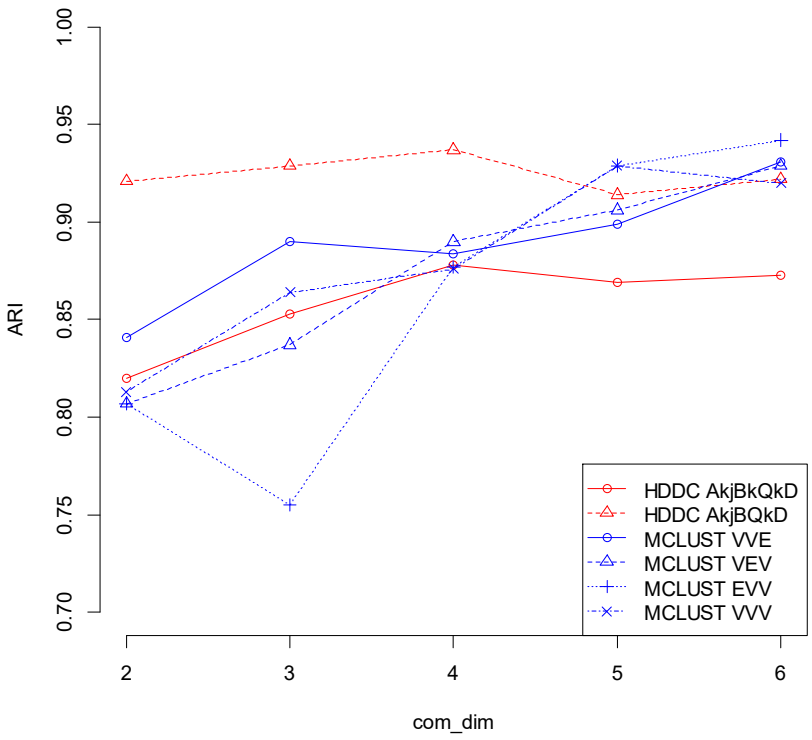



Figure 5

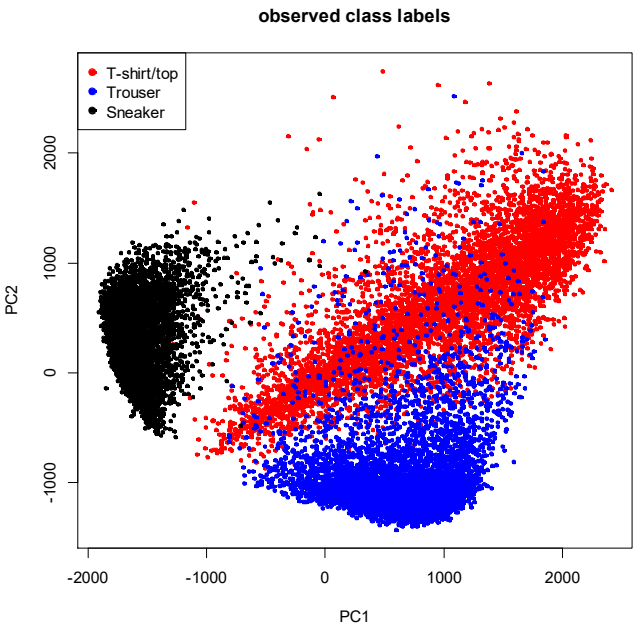
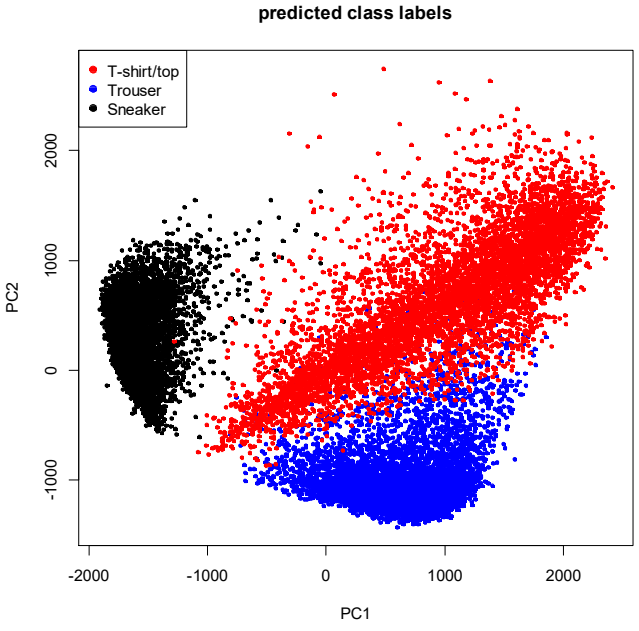



Figure 6



As we can see in Figures 5 the cluster with Sneakers is well-separated from the other two clusters. On the other hand, Trousers and T-shirt/Top can are not separated well as certain Trousers are in the area of the cluster T-shirt/Top. Figure 6 shows that the clusters derived by Mclust can recover the observed cluster structure well, except for the fact that Trousers in the area of the cluster T-shirt/Top are incorrectly classified as T-shirt/Top.

**Task 3 (a)**

We load the smacof package which also contains the data set.

```
library(smacof)
```

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.

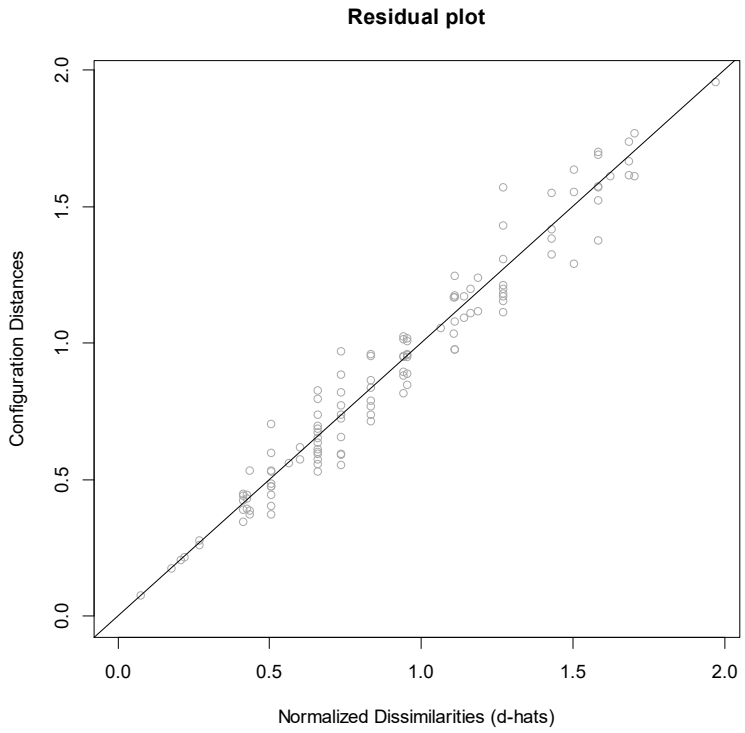
```
#ratio
m1<-smacofSym(delta=rectangles, ndim=2, type="ratio", init="torgerson")
#interval
m2<-smacofSym(delta=rectangles, ndim=2, type="interval", init="torgerson")
#ordinal
m3<-smacofSym(delta=rectangles, ndim=2, type="ordinal", init="torgerson")
#mspline
m4<-smacofSym(delta=rectangles, ndim=2, type="mspline",spline.degree =4 ,
spline.intKnots = 4, init="torgerson")


#stress-1 values
round(c(m1$stress,m2$stress,m3$stress,m4$stress),3)
[1] 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 (i.e. disparities) versus configuration distances for the ordinal solution shows that the configuration distances can predict the disparities very well (see Figure 7). The correlation between transformed dissimilarities and configuration distances equals .977.

```
plot(m3,plot.type="resplot")
```

Figure 7




**Task 3 (b)**

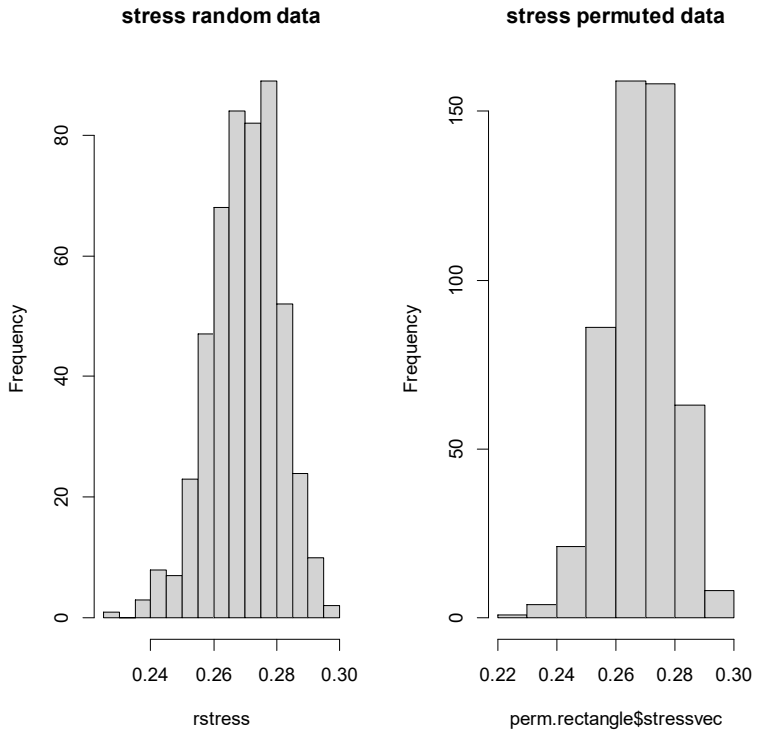
We use stress norms and a permutation test to investigate whether the ordinal MDS solution has satisfactory fit. Next we plot the stress-1 distributions for each test (see Figure 8)

```
#stress norm
set.seed(1)
rstress<-randomstress(n=16,ndim=2,nrep=500,type="ordinal")
#distribution of stress for random data
mean(rstress)-2*sd(rstress)
[1] 0.247801
#permutation test
set.seed(1)
perm.rectangle<-permtest(m3,nrep=500)
mean(perm.rectangle$stressvec)-2*sd(perm.rectangle$stressvec)
[1] 0.2459418

#plot distribution stress
par(mfrow=c(1,2),pty="s")
hist(rstress,main="stress random data")
```

```
hist(perm.rectangle$stressvec,main="stress permuted data")
```

Figure 8




The observed stress-1 value of the ordinal MDS solution (.089) is smaller than the cutoff obtained from the distribution of stress-1 using random data $(\bar{x}_r - 2\hat{\sigma}_r = .248)$, and than the cutoff obtained from the stress-1 distribution using permuted data $(\bar{x}_r - 2\hat{\sigma}_r = .246)$. Therefore we can conclude that the ordinal MDS solution has a satisfactory fit.

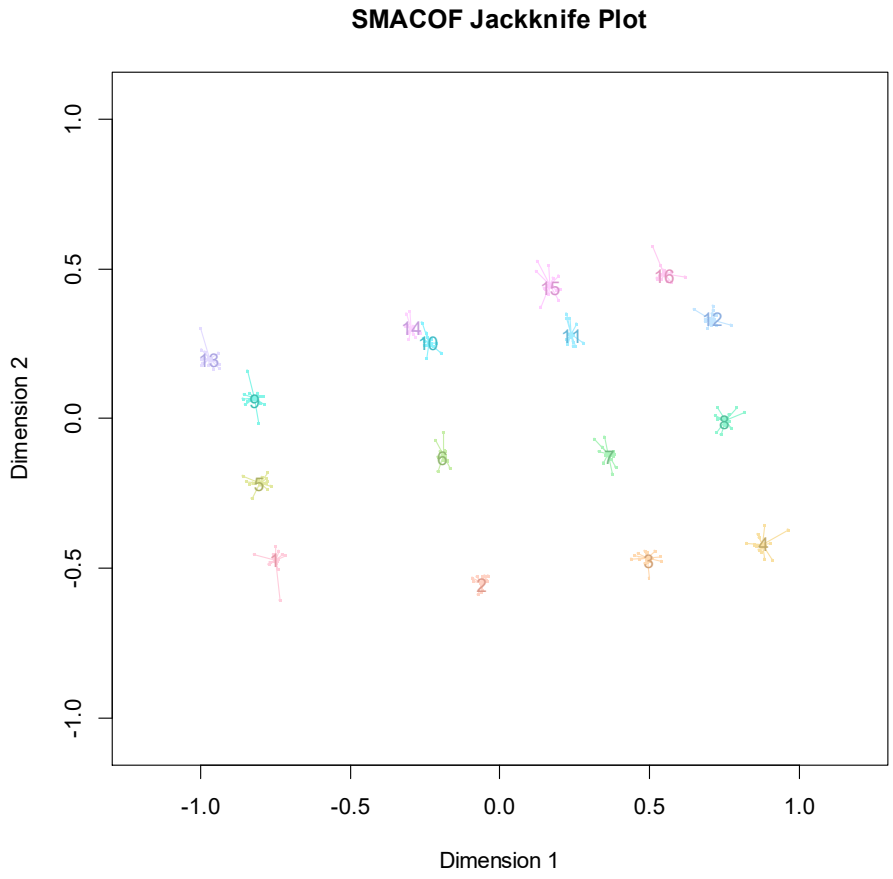
The results for the other MDS solutions are less relevant, since stress-1 is lowest for the ordinal MDS solution.

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 (see Figure 9) indicates that the points in the configuration have a very stable position.

Figure 9




**Task 3 (b)**

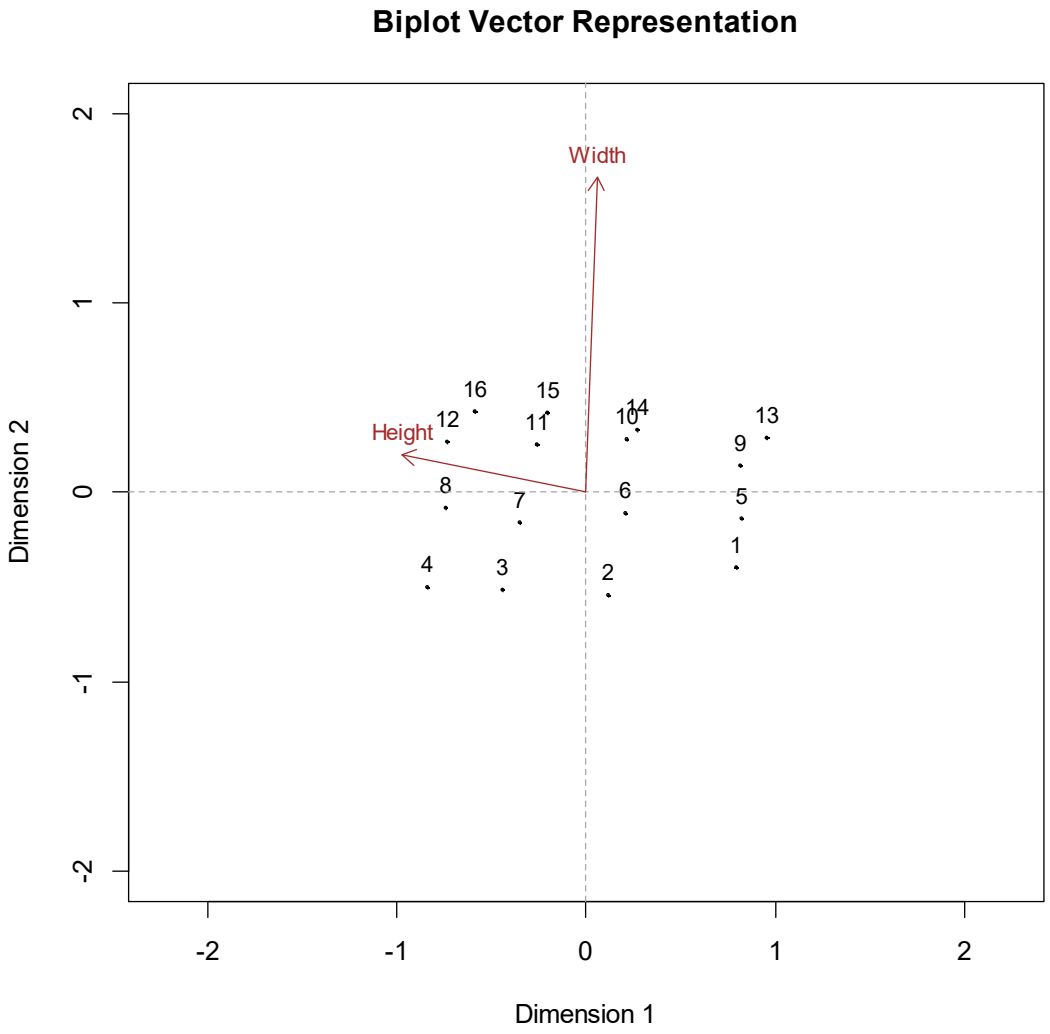
We use biplotmds() to project the variables “width” and “height” into the configuration plot of the ordinal two-dimensional solution (see Figure 10). We also compute the correlations between the dimensions in the configuration plot and the external variables.

```
birect <- biplotmds(m3, rect_constr)

plot(birect, main = "Biplot Vector Representation", vecscale = 0.6,
xlim = c(-2, 2), ylim=c(-2,2), vec.conf = list(col = "brown"),
pch = 20, cex = 0.5)

round(cor(rect_constr,m3$conf),3)
           D1    D2
Width   0.117 0.970
Height -0.987 0.056
```

Figure 10




The biplot shows that D1 strongly correlates with “Height” (R=-.987) and that D2 strongly correlates with “Width” (R=.97). The configuration of the points in the biplot reflects the true width and height of the stimuli used in the experiment only to some extent: In the experimental design the width and the height of the rectangles increases with equal steps. More specifically, width increases with steps of 1.25 (i.e. width values are 3, 4.25, 5.5, 6.75) and height is increases with steps of 0.75 (i.e., height values are 0.5, 1.25, 2, 2.75). In contrast ,in the configuration plot we see that the width and the height of the stimuli does not increase with equal steps, i.e. the width of stimuli 13,14,15,16 and the height of stimuli 4,8,12,16 is underestimated. This could be evidence that “ psychophysical rescalings follow the Weber‐Fechner law, which states a logarithmic correspondence of psychological and physical units”. (see Borg & Groenen, 2005, p. 374)

**reference**

Borg, I., & Groenen, P. J. F. (2005). *Modern multidimensional scaling: Theory and applications* (second Edition). New York: Springer Science + Business Media, Inc.