

MDS Analysis

Danielle Daidone

3/31/26

This script runs multidimensional scaling (MDS) analyses on matrix outputs from `create_FC_similarity_matrices.Rmd`. There are chunks for running MDS on all contexts combined as well as up to 4 contexts separately.

It outputs 1,2,3,4 and 5 dimensional solutions as text files and produces a stress plot to help you decide on the most appropriate number of dimensions. It also produces basic plots of the solutions, but note that the solutions are unrotated and thus may not be easily interpretable.

This script also computes Euclidean distances between points for 2 and 3 dimensional solutions (i.e. the number of dimensions most likely to be interpretable) with all speakers combined so that MDS distances between sounds can be compared to discrimination accuracy. It outputs these distance matrices as csv files. The `smacof` package automatically calculates distances as part of the output for every matrix, so you can easily get distances for any other MDS solution as desired.

Because of the length of the script, I recommend collapsing all the code chunks for easy of readability. Go to Edit -> Folding -> Collapse All. Alternately, the short cut is Alt + O (the letter).

Download and load relevant packages See documentation: `smacof` (for MDS): <https://rdrr.io/cran/smacof/man/smacofSym.html>

```
if(!require(rmarkdown)){install.packages("rmarkdown")}
```

```
## Loading required package: rmarkdown
```

```
## Warning: package 'rmarkdown' was built under R version 4.5.3
```

```
library(rmarkdown)
if(!require(smacof)){install.packages("smacof")}
```

```
## Loading required package: smacof
```

```
## Loading required package: plotrix
```

```
## Loading required package: colorspace
```

```
## Loading required package: e1071
```

```
##
```

```
## Attaching package: 'smacof'
```

```
## The following object is masked from 'package:base':
```

```
##
```

```
## transform
```

```
library(smaccf)
```

Set working directory - *Change to match your filepath*

SECTION 1: CALCULATE MDS SOLUTIONS

PART 1: GET MDS SOLUTIONS FOR 1-5 DIMENSIONS FOR CONTEXTS COMBINED

Open dataset - Contexts combined *dissimilarity matrix*

```
# Note that the dissimilarity text files are tab delimited files with headers  
# If you've changed the default file names from the previous script, you'll need to adjust this  
FCmatrixAll=read.table("Output_Matrix_AllContexts_percent_dis.txt",sep="\t",header=TRUE)
```

Contexts combined: Look at the first few rows of the dataset

```
head(FCmatrixAll)
```

```
##           cVccV_L  cVccV_M  cVccV_N  cVccVV_L  cVccVV_M  cVccVV_N  cVcV_L  
## cVccV_L  0.0000000  0.7564103  0.6666667  0.8461538  0.8589744  0.8205128  0.7564103  
## cVccV_M  0.7564103  0.0000000  0.6666667  0.9615385  0.6025641  0.9487179  0.3974359  
## cVccV_N  0.6666667  0.6666667  0.0000000  0.8717949  0.8846154  0.8076923  0.7692308  
## cVccVV_L 0.8461538  0.9615385  0.8717949  0.0000000  0.8589744  0.5897436  0.9615385  
## cVccVV_M 0.8589744  0.6025641  0.8846154  0.8589744  0.0000000  0.8333333  0.7307692  
## cVccVV_N 0.8205128  0.9487179  0.8076923  0.5897436  0.8333333  0.0000000  0.9102564  
##           cVcV_M  cVcV_N  cVcVV_L  cVcVV_M  cVcVV_N  cVVccV_L  cVVccV_M  
## cVccV_L  0.8076923  0.7692308  0.8461538  0.9102564  0.9102564  0.7692308  0.8589744  
## cVccV_M  0.4102564  0.5256410  0.8205128  0.6666667  0.9230769  0.9871795  0.8589744  
## cVccV_N  0.7435897  0.7307692  0.9102564  0.9230769  0.9615385  0.7820513  0.7564103  
## cVccVV_L 0.9743590  0.9871795  0.6410256  0.8333333  0.6666667  0.8974359  0.8846154  
## cVccVV_M 0.7692308  0.8333333  0.7564103  0.5384615  0.8333333  0.9230769  0.7564103  
## cVccVV_N 0.9615385  0.9230769  0.6410256  0.8205128  0.5641026  0.8846154  0.9102564  
##           cVVccV_N  cVVccVV_L  cVVccVV_M  cVVccVV_N  cVVcV_L  cVVcV_M  cVVcV_N  
## cVccV_L  0.7820513  0.8076923  0.8461538  0.8717949  0.8717949  0.8461538  0.8461538  
## cVccV_M  0.9487179  0.9102564  0.8589744  0.9102564  0.9743590  0.9102564  0.9487179  
## cVccV_N  0.7179487  0.8846154  0.8846154  0.8461538  0.8717949  0.9230769  0.8205128  
## cVccVV_L 0.9102564  0.7179487  0.8333333  0.7948718  0.8717949  0.8717949  0.8461538  
## cVccVV_M 0.9358974  0.8974359  0.7820513  0.8846154  0.9102564  0.8461538  0.9230769  
## cVccVV_N 0.8461538  0.8717949  0.8974359  0.7820513  0.8717949  0.9230769  0.7692308  
##           cVVcV_L  cVVcV_M  cVVcV_N  
## cVccV_L  0.9487179  0.8974359  0.8717949  
## cVccV_M  0.8974359  0.8205128  0.9230769  
## cVccV_N  0.8974359  0.7948718  0.8846154  
## cVccVV_L 0.7435897  0.8974359  0.8846154  
## cVccVV_M 0.9358974  0.8461538  0.8589744  
## cVccVV_N 0.9743590  0.9230769  0.7435897
```

Contexts combined: Run ordinal multi-dimensional scaling (MDS) - 1 dimensional solution

```
# ndim is the number of dimensions  
MDS1Dim_all = mds(FCmatrixAll, ndim = 1, type = "ordinal")  
  
# view Kruskal stress  
MDS1Dim_all
```

```
##  
## Call:  
## mds(delta = FCmatrixAll, ndim = 1, type = "ordinal")  
##  
## Model: Symmetric SMACOF  
## Number of objects: 24  
## Stress-1 value: 0.373  
## Number of iterations: 8
```

```
# get dim scores  
dim1scores_all = MDS1Dim_all$conf  
  
# write dim scores out to a text file in your working directory  
write.table(dim1scores_all,file="MDS_1D_AllContexts.txt",sep="\t",quote=FALSE)
```

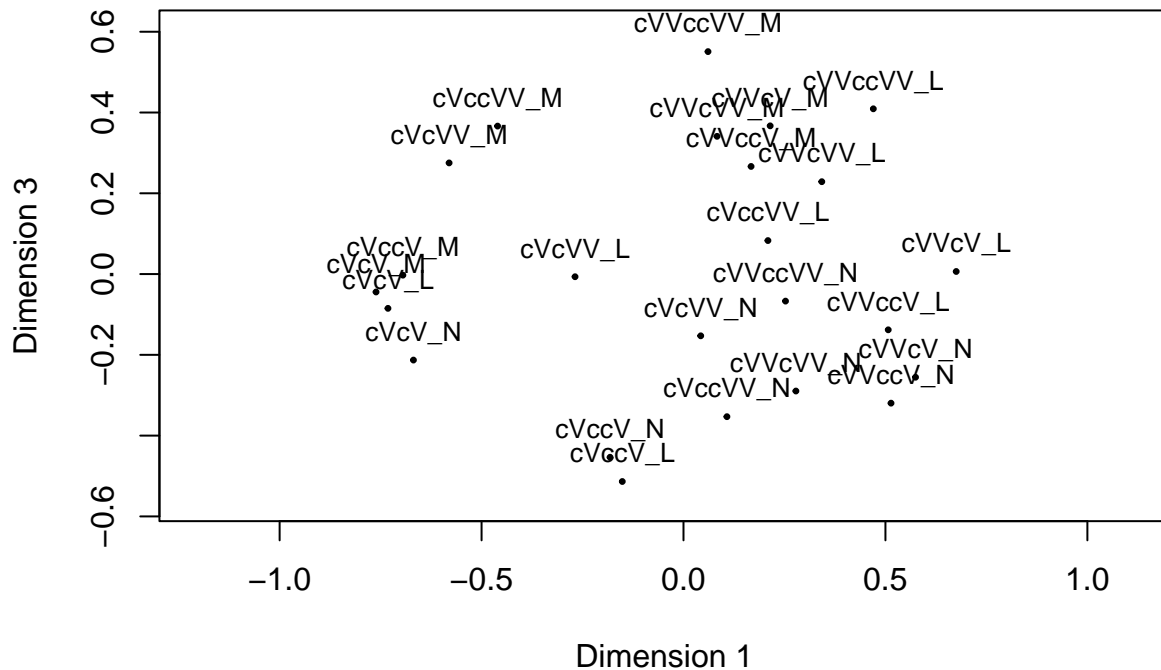
Contexts combined: Run ordinal multi-dimensional scaling (MDS) - 2 dimensional solution

```
# ndim is the number of dimensions  
MDS2Dim_all = mds(FCmatrixAll, ndim = 2, type = "ordinal")  
  
# view Kruskal stress  
MDS2Dim_all
```

```
##  
## Call:  
## mds(delta = FCmatrixAll, ndim = 2, type = "ordinal")  
##  
## Model: Symmetric SMACOF  
## Number of objects: 24  
## Stress-1 value: 0.183  
## Number of iterations: 30
```

```
# get dim scores  
dim2scores_all = MDS2Dim_all$conf  
  
# write dim scores out to a text file in your working directory  
write.table(dim2scores_all,file="MDS_2D_AllContexts.txt",sep="\t",quote=FALSE)  
  
#plot dim 1 by dim 2  
plot(MDS2Dim_all, plot.type = "confplot", plot.dim = c(1,2))
```


Configuration Plot



Contexts combined: Run ordinal multi-dimensional scaling (MDS) - 5 dimensional solution

```
# ndim is the number of dimensions
MDS5Dim_all = mds(FCmatrixAll, ndim = 5, type = "ordinal")

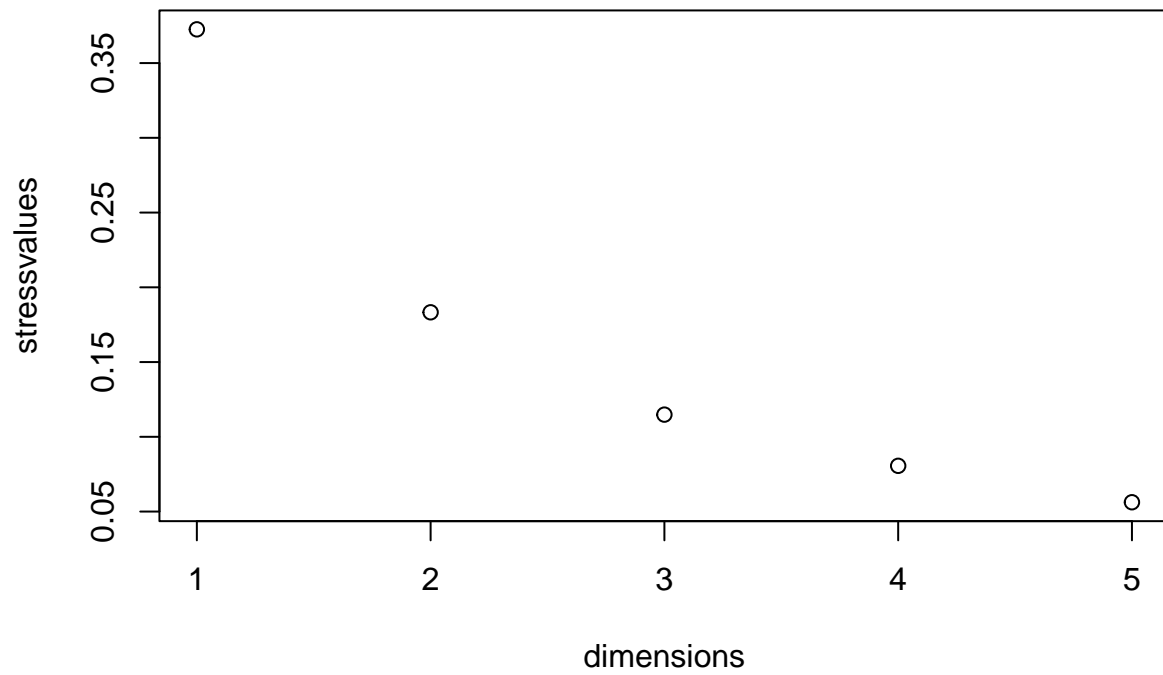
# view Kruskal stress
MDS5Dim_all
```

```
##
## Call:
## mds(delta = FCmatrixAll, ndim = 5, type = "ordinal")
##
## Model: Symmetric SMACOF
## Number of objects: 24
## Stress-1 value: 0.056
## Number of iterations: 73
```

```
# get dim scores
dim5scores_all = MDS5Dim_all$conf

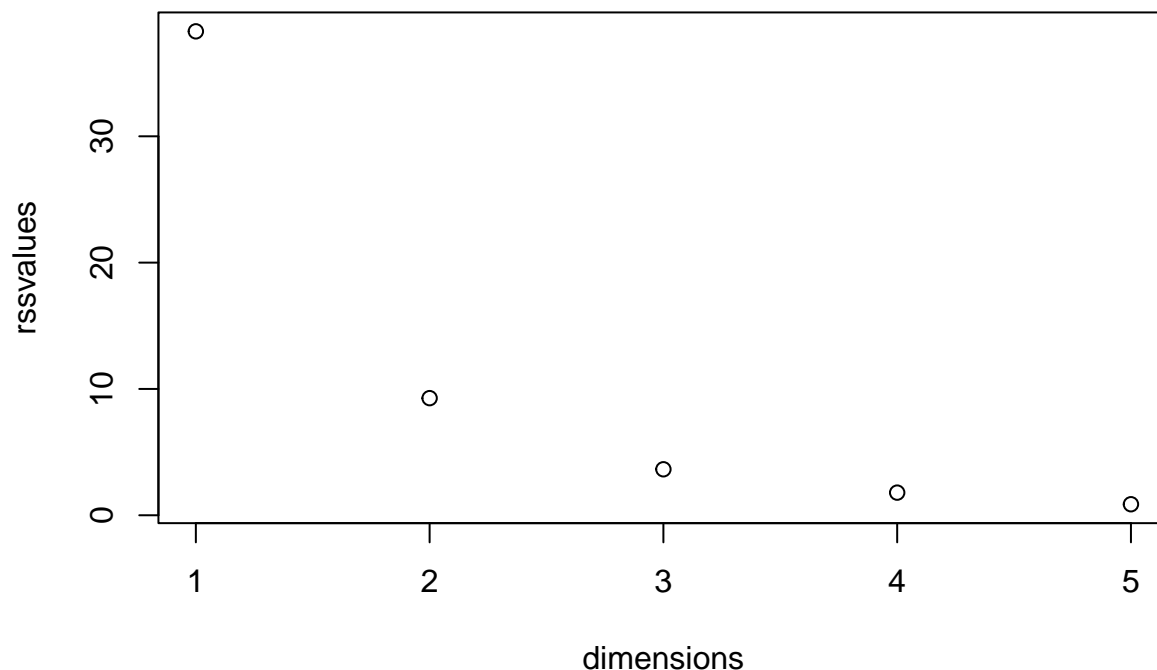
# write dim scores out to a text file in your working directory
write.table(dim5scores_all, file="MDS_5D_AllContexts.txt", sep="\t", quote=FALSE)

#plot dim 1 by dim 2
plot(MDS5Dim_all, plot.type = "confplot", plot.dim = c(1,2))
```

Contexts combined: Create plot of residual sum of squares

```
rssvalues = c(MDS1Dim_all$rss,MDS2Dim_all$rss,MDS3Dim_all$rss,MDS4Dim_all$rss,MDS5Dim_all$rss)
dimensions = c(1,2,3,4,5)
plot(dimensions,rssvalues)
```



PART 2: GET MDS SOLUTIONS FOR 1-5 DIMENSIONS FOR CONTEXT 1

Open dataset - Context 1 *dissimilarity* matrix

```
# Note that the dissimilarity text files are tab delimited files with headers
# If you've changed the default file names from the previous script, you'll need to adjust this
FCmatrixcontext1=read.table("Output_Matrix_Context1_percent_dis.txt",sep="\t",header=TRUE)
```

Context 1: Look at the first few rows of the dataset

```
head(FCmatrixcontext1)
```

```
##          cVccV_upu_L cVccV_upu_M cVccV_upu_N cVccVV_upu_L cVccVV_upu_M
## cVccV_upu_L  0.0000000  0.8076923  0.7307692  0.7692308  0.9615385
## cVccV_upu_M  0.8076923  0.0000000  0.5769231  0.9615385  0.7307692
## cVccV_upu_N  0.7307692  0.5769231  0.0000000  0.9230769  0.8846154
## cVccVV_upu_L 0.7692308  0.9615385  0.9230769  0.0000000  0.8076923
## cVccVV_upu_M 0.9615385  0.7307692  0.8846154  0.8076923  0.0000000
## cVccVV_upu_N 0.8846154  0.8846154  0.8461538  0.5769231  0.7692308
##          cVccVV_upu_N cVcV_upu_L cVcV_upu_M cVcV_upu_N cVcVV_upu_L
## cVccV_upu_L  0.8846154  0.6923077  0.8461538  0.7692308  0.8461538
## cVccV_upu_M  0.8846154  0.5000000  0.3461538  0.5384615  0.7307692
## cVccV_upu_N  0.8461538  0.7307692  0.6153846  0.7307692  0.9230769
## cVccVV_upu_L 0.5769231  0.9615385  1.0000000  0.9615385  0.5384615
## cVccVV_upu_M 0.7692308  0.8846154  0.8076923  0.8461538  0.6538462
## cVccVV_upu_N 0.0000000  0.8846154  0.9615385  0.8846154  0.6923077
```

```

##          cVcVV_upu_M cVcVV_upu_N cVVccV_upu_L cVVccV_upu_M cVVccV_upu_N
## cVccV_upu_L    0.8846154  0.8846154  0.6538462  0.8076923  0.6923077
## cVccV_upu_M    0.5769231  0.9615385  0.9615385  0.8461538  0.8846154
## cVccV_upu_N    0.8846154  0.9230769  0.8076923  0.7692308  0.8076923
## cVccVV_upu_L   0.8076923  0.7307692  0.8846154  0.8461538  0.9230769
## cVccVV_upu_M   0.4615385  0.8076923  0.9230769  0.8076923  0.9230769
## cVccVV_upu_N   0.8076923  0.4615385  0.8846154  0.9230769  0.8461538
##          cVVccVV_upu_L cVVccVV_upu_M cVVccVV_upu_N cVVcV_upu_L cVVcV_upu_M
## cVccV_upu_L    0.8461538  0.8461538  0.8076923  0.8461538  0.7307692
## cVccV_upu_M    0.9230769  0.8846154  0.9230769  0.9615385  0.8846154
## cVccV_upu_N    0.8461538  0.8461538  0.7692308  0.8846154  0.8846154
## cVccVV_upu_L   0.7307692  0.8461538  0.8076923  0.8461538  0.9230769
## cVccVV_upu_M   0.8461538  0.6923077  0.8846154  0.9615385  0.8461538
## cVccVV_upu_N   0.8076923  0.8846154  0.7307692  0.8076923  1.0000000
##          cVVcV_upu_N cVVcVV_upu_L cVVcVV_upu_M cVVcVV_upu_N
## cVccV_upu_L    0.8846154  0.9230769  0.8846154  0.8846154
## cVccV_upu_M    0.9230769  0.8846154  0.8846154  0.8846154
## cVccV_upu_N    0.8846154  0.8461538  0.8461538  0.8461538
## cVccVV_upu_L   0.9230769  0.8461538  0.8846154  0.8846154
## cVccVV_upu_M   0.8846154  0.8846154  0.8461538  0.7307692
## cVccVV_upu_N   0.8461538  1.0000000  0.8846154  0.7692308

```

Context 1: Run ordinal multi-dimensional scaling (MDS) - 1 dimensional solution

```

# ndim is the number of dimensions
MDS1Dim_context1 = mds(FCmatrixcontext1, ndim = 1, type = "ordinal")

# view Kruskal stress
MDS1Dim_context1

##
## Call:
## mds(delta = FCmatrixcontext1, ndim = 1, type = "ordinal")
##
## Model: Symmetric SMACOF
## Number of objects: 24
## Stress-1 value: 0.379
## Number of iterations: 7

# get dim scores
dim1scores_context1 = MDS1Dim_context1$conf

# write dim scores out to a text file in your working directory
write.table(dim1scores_context1,file="MDS_1D_Context1.txt",sep="\t",quote=FALSE)

```

Context 1: Run ordinal multi-dimensional scaling (MDS) - 2 dimensional solution

```

# ndim is the number of dimensions
MDS2Dim_context1 = mds(FCmatrixcontext1, ndim = 2, type = "ordinal")

# view Kruskal stress
MDS2Dim_context1

```

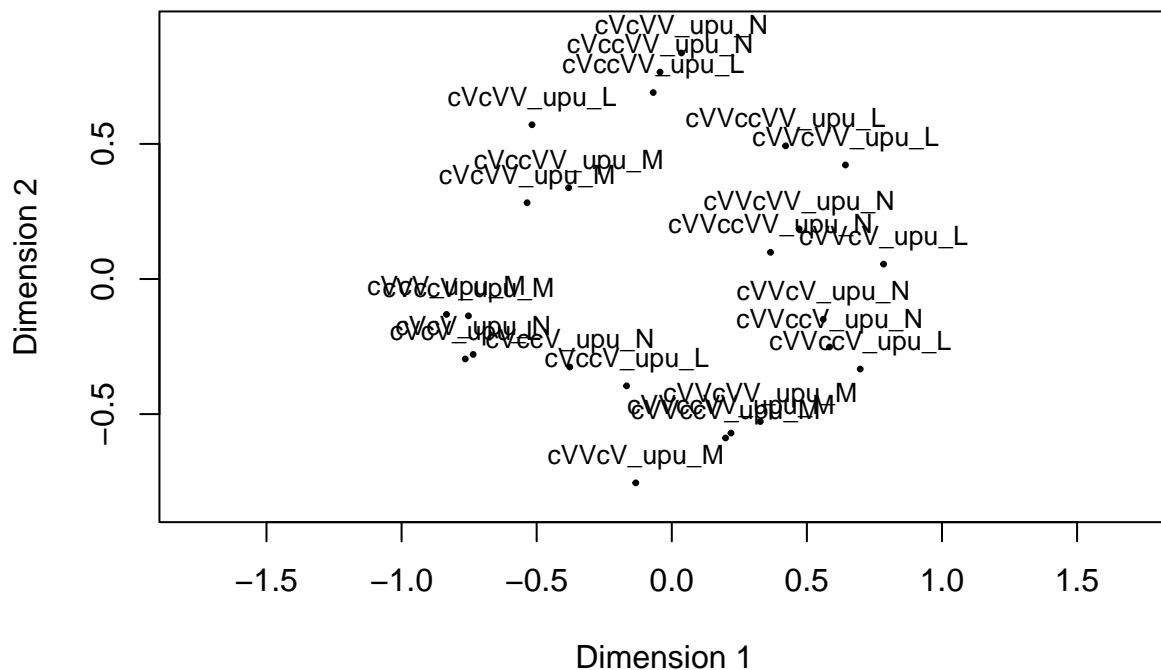
```
##
## Call:
## mds(delta = FCmatrixcontext1, ndim = 2, type = "ordinal")
##
## Model: Symmetric SMACOF
## Number of objects: 24
## Stress-1 value: 0.185
## Number of iterations: 34
```

```
# get dim scores
dim2scores_context1 = MDS2Dim_context1$conf

# write dim scores out to a text file in your working directory
write.table(dim2scores_context1,file="MDS_2D_Context1.txt",sep="\t",quote=FALSE)

#plot dim 1 by dim 2
plot(MDS2Dim_context1, plot.type = "confplot", plot.dim = c(1,2))
```

Configuration Plot



Contexts combined: Run ordinal multi-dimensional scaling (MDS) - 3 dimensional solution

```
# ndim is the number of dimensions
MDS3Dim_context1 = mds(FCmatrixcontext1, ndim = 3, type = "ordinal")

# view Kruskal stress
MDS3Dim_context1
```

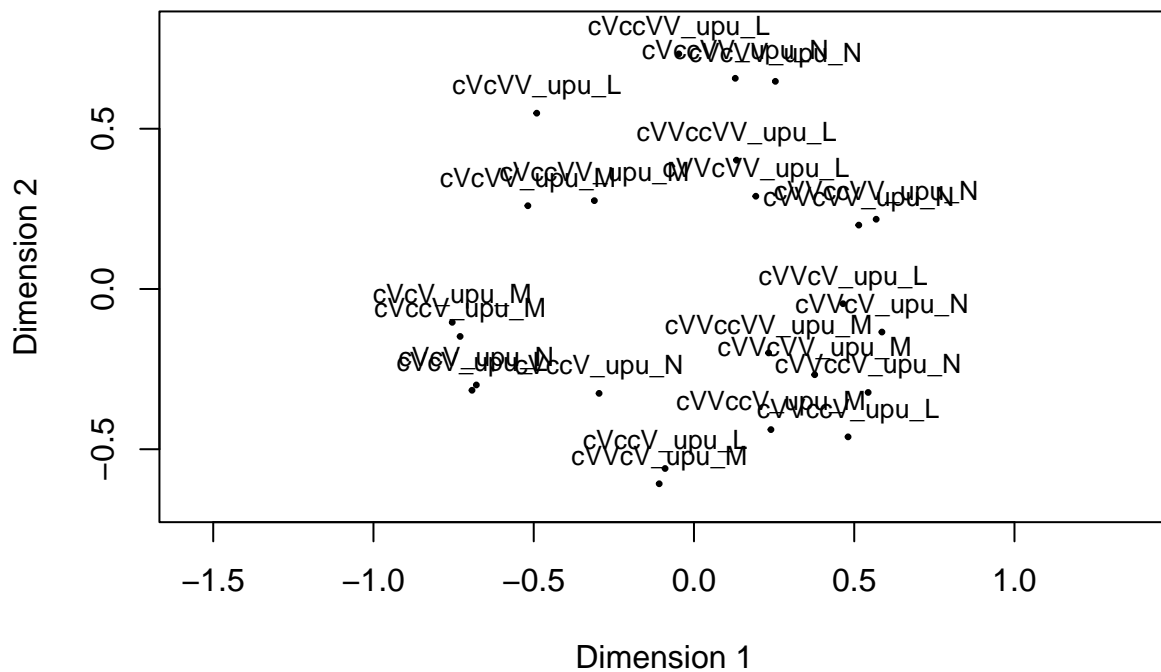
```
##
## Call:
## mds(delta = FCmatrixcontext1, ndim = 3, type = "ordinal")
##
## Model: Symmetric SMACOF
## Number of objects: 24
## Stress-1 value: 0.109
## Number of iterations: 67

# get dim scores
dim3scores_context1 = MDS3Dim_context1$conf

# write dim scores out to a text file in your working directory
write.table(dim3scores_context1,file="MDS_3D_Context1.txt",sep="\t",quote=FALSE)

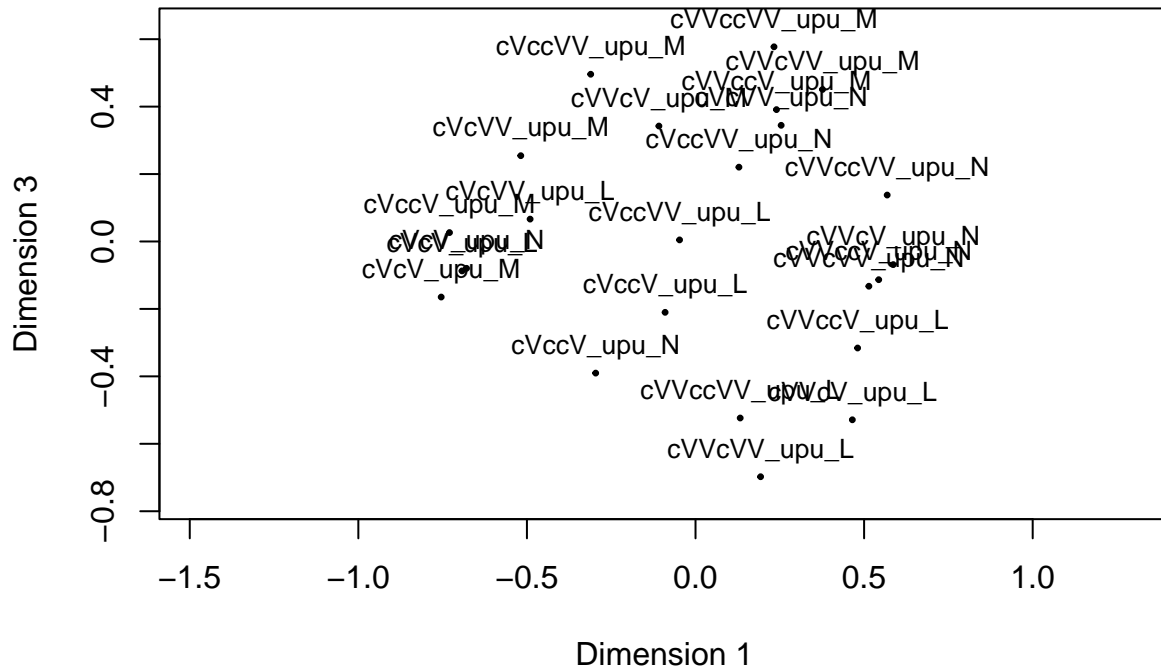
#plot dim 1 by dim 2
plot(MDS3Dim_context1, plot.type = "confplot", plot.dim = c(1,2))
```

Configuration Plot



```
#plot dim 1 by dim 3 (as if looking into previous plot from the top)
plot(MDS3Dim_context1, plot.type = "confplot", plot.dim = c(1,3))
```

Configuration Plot



Context 1: Run ordinal multi-dimensional scaling (MDS) - 4 dimensional solution

```
# ndim is the number of dimensions
MDS4Dim_context1 = mds(FCmatrixcontext1, ndim = 4, type = "ordinal")

# view Kruskal stress
MDS4Dim_context1

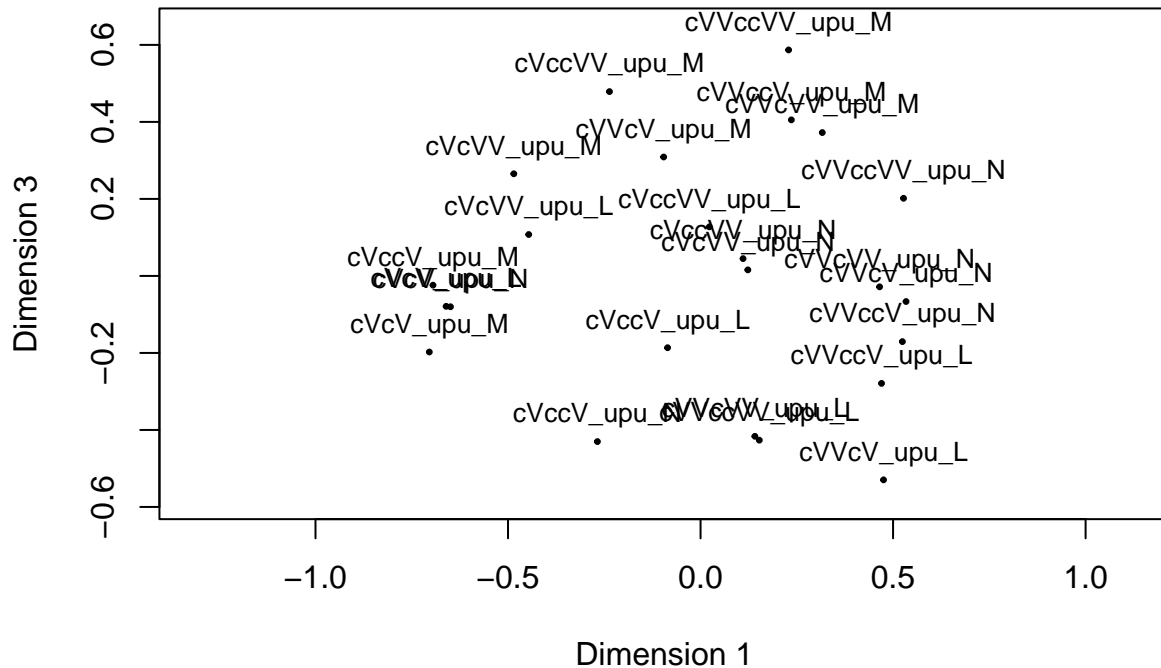
##
## Call:
## mds(delta = FCmatrixcontext1, ndim = 4, type = "ordinal")
##
## Model: Symmetric SMACOF
## Number of objects: 24
## Stress-1 value: 0.072
## Number of iterations: 65

# get dim scores
dim4scores_context1 = MDS4Dim_context1$conf

# write dim scores out to a text file in your working directory
write.table(dim4scores_context1, file="MDS_4D_Context1.txt", sep="\t", quote=FALSE)

#plot dim 1 by dim 2
plot(MDS4Dim_context1, plot.type = "confplot", plot.dim = c(1,2))
```


Configuration Plot



Context 1: Run ordinal multi-dimensional scaling (MDS) - 5 dimensional solution

```
# ndim is the number of dimensions
MDS5Dim_context1 = mds(FCmatrixcontext1, ndim = 5, type = "ordinal")

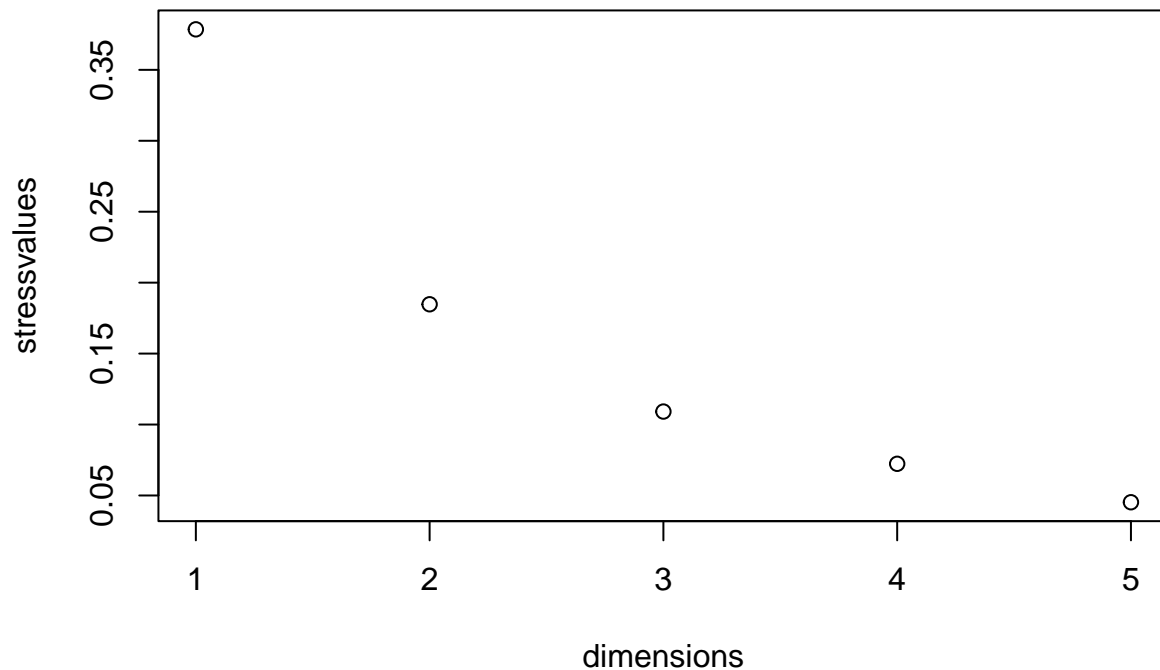
# view Kruskal stress
MDS5Dim_context1

##
## Call:
## mds(delta = FCmatrixcontext1, ndim = 5, type = "ordinal")
##
## Model: Symmetric SMACOF
## Number of objects: 24
## Stress-1 value: 0.045
## Number of iterations: 61

# get dim scores
dim5scores_context1 = MDS5Dim_context1$conf

# write dim scores out to a text file in your working directory
write.table(dim5scores_context1, file="MDS_5D_Context1.txt", sep="\t", quote=FALSE)

#plot dim 1 by dim 2
plot(MDS5Dim_context1, plot.type = "confplot", plot.dim = c(1,2))
```

PART 3: GET MDS SOLUTIONS FOR 1-5 DIMENSIONS FOR CONTEXT 2

Open dataset - Context 2 *dissimilarity* matrix

```
# Note that the dissimilarity text files are tab delimited files with headers
# If you've changed the default file names from the previous script, you'll need to adjust this
FCmatrixcontext2=read.table("Output_Matrix_Context2_percent_dis.txt",sep="\t",header=TRUE)
```

Context 2: Look at the first few rows of the dataset

```
head(FCmatrixcontext2)
```

```
##          cVccV_iki_L cVccV_iki_M cVccV_iki_N cVccVV_iki_L cVccVV_iki_M
## cVccV_iki_L  0.0000000  0.7307692  0.6153846  0.8846154  0.7692308
## cVccV_iki_M  0.7307692  0.0000000  0.7692308  0.9615385  0.6153846
## cVccV_iki_N  0.6153846  0.7692308  0.0000000  0.9230769  0.8461538
## cVccVV_iki_L 0.8846154  0.9615385  0.9230769  0.0000000  0.8846154
## cVccVV_iki_M 0.7692308  0.6153846  0.8461538  0.8846154  0.0000000
## cVccVV_iki_N 0.6923077  0.9615385  0.7307692  0.7307692  0.8461538
##          cVccVV_iki_N cVcV_iki_L cVcV_iki_M cVcV_iki_N cVcVV_iki_L
## cVccV_iki_L  0.6923077  0.8076923  0.8076923  0.8076923  0.7692308
## cVccV_iki_M  0.9615385  0.4615385  0.5000000  0.6538462  0.8461538
## cVccV_iki_N  0.7307692  0.8846154  0.9230769  0.8076923  1.0000000
## cVccVV_iki_L 0.7307692  1.0000000  1.0000000  1.0000000  0.8076923
## cVccVV_iki_M 0.8461538  0.7692308  0.7307692  0.8461538  0.8076923
## cVccVV_iki_N 0.0000000  0.9230769  1.0000000  0.9230769  0.6538462
```

```

##          cVcVV_iki_M cVcVV_iki_N cVVccV_iki_L cVVccV_iki_M cVVccV_iki_N
## cVccV_iki_L    0.8846154  0.8846154  0.9230769  0.9615385  0.8076923
## cVccV_iki_M    0.6538462  0.8461538  1.0000000  0.8846154  0.9615385
## cVccV_iki_N    0.9615385  1.0000000  0.7692308  0.6923077  0.4615385
## cVccVV_iki_L  0.9615385  0.6153846  0.9230769  0.9230769  0.9615385
## cVccVV_iki_M  0.5384615  0.8461538  0.9230769  0.7307692  0.9230769
## cVccVV_iki_N  0.9615385  0.7692308  0.8461538  0.9230769  0.8076923
##          cVVccVV_iki_L cVVccVV_iki_M cVVccVV_iki_N cVVcV_iki_L cVVcV_iki_M
## cVccV_iki_L    0.8846154  0.8461538  0.9615385  0.9615385  0.9615385
## cVccV_iki_M    0.8846154  0.7692308  0.8846154  0.9615385  0.9230769
## cVccV_iki_N    0.9615385  0.9230769  0.9230769  0.8846154  0.9615385
## cVccVV_iki_L  0.6538462  0.8846154  0.8461538  0.8076923  0.8076923
## cVccVV_iki_M  0.9230769  0.8846154  0.8846154  0.9230769  0.8846154
## cVccVV_iki_N  0.9615385  0.8461538  0.7307692  0.8846154  0.9230769
##          cVVcV_iki_N cVVcVV_iki_L cVVcVV_iki_M cVVcVV_iki_N
## cVccV_iki_L    0.8846154  1.0000000  0.8461538  0.8461538
## cVccV_iki_M    0.9230769  0.8846154  0.7307692  0.9230769
## cVccV_iki_N    0.7307692  1.0000000  0.8076923  0.9615385
## cVccVV_iki_L  0.8076923  0.5769231  0.8846154  0.8846154
## cVccVV_iki_M  0.9615385  1.0000000  0.8461538  0.8461538
## cVccVV_iki_N  0.7692308  0.9615385  0.9615385  0.6923077

```

Context 2: Run ordinal multi-dimensional scaling (MDS) - 1 dimensional solution

```

# ndim is the number of dimensions
MDS1Dim_context2 = mds(FCmatrixcontext2, ndim = 1, type = "ordinal")

# view Kruskal stress
MDS1Dim_context2

##
## Call:
## mds(delta = FCmatrixcontext2, ndim = 1, type = "ordinal")
##
## Model: Symmetric SMACOF
## Number of objects: 24
## Stress-1 value: 0.343
## Number of iterations: 7

# get dim scores
dim1scores_context2 = MDS1Dim_context2$conf

# write dim scores out to a text file in your working directory
write.table(dim1scores_context2,file="MDS_1D_Context2.txt",sep="\t",quote=FALSE)

```

Context 2: Run ordinal multi-dimensional scaling (MDS) - 2 dimensional solution

```

# ndim is the number of dimensions
MDS2Dim_context2 = mds(FCmatrixcontext2, ndim = 2, type = "ordinal")

# view Kruskal stress
MDS2Dim_context2

```

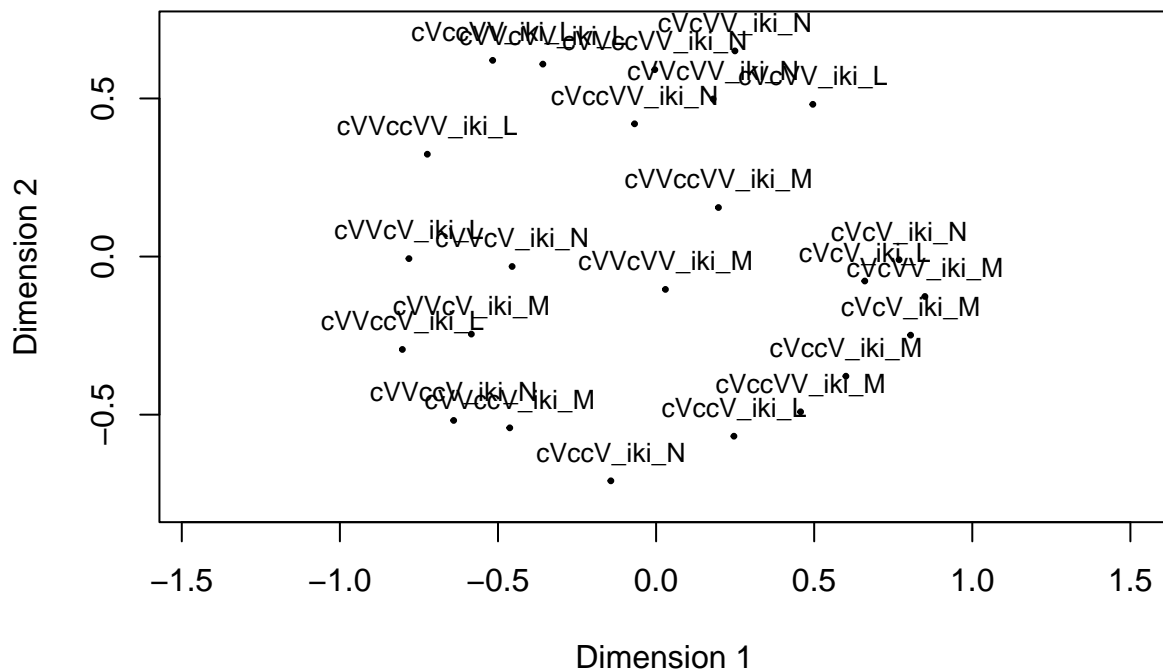
```
##
## Call:
## mds(delta = FCmatrixcontext2, ndim = 2, type = "ordinal")
##
## Model: Symmetric SMACOF
## Number of objects: 24
## Stress-1 value: 0.185
## Number of iterations: 78

# get dim scores
dim2scores_context2 = MDS2Dim_context2$conf

# write dim scores out to a text file in your working directory
write.table(dim2scores_context2,file="MDS_2D_Context2.txt",sep="\t",quote=FALSE)

#plot dim 1 by dim 2
plot(MDS2Dim_context2, plot.type = "confplot", plot.dim = c(1,2))
```

Configuration Plot



Contexts combined: Run ordinal multi-dimensional scaling (MDS) - 3 dimensional solution

```
# ndim is the number of dimensions
MDS3Dim_context2 = mds(FCmatrixcontext2, ndim = 3, type = "ordinal")

# view Kruskal stress
MDS3Dim_context2
```

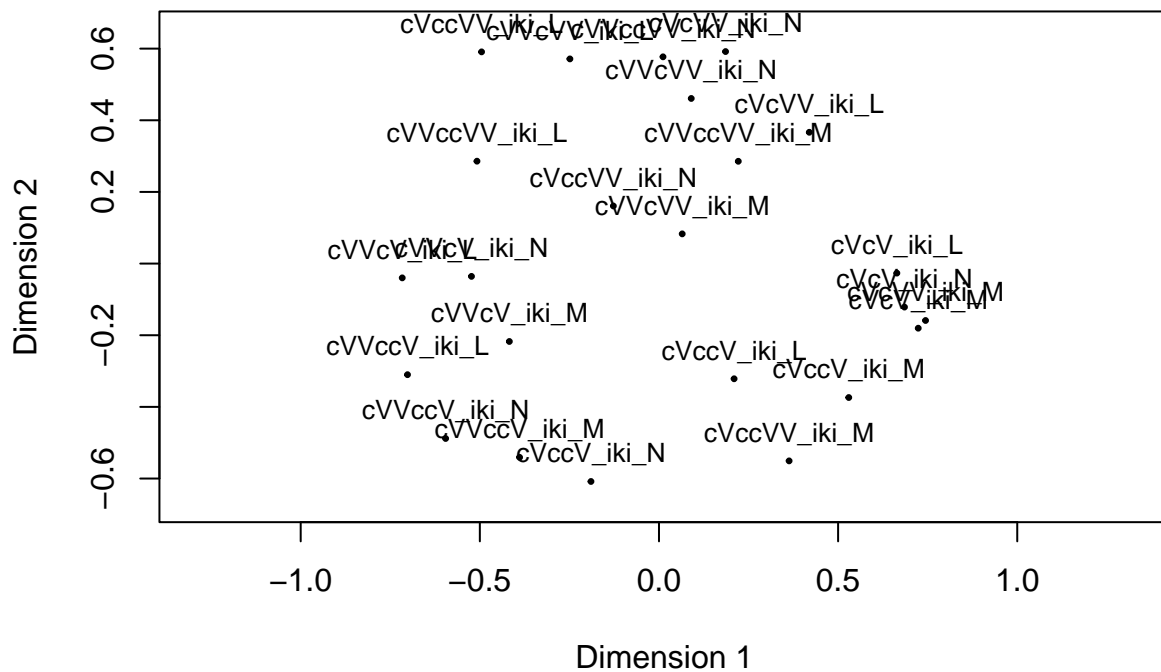
```
##
## Call:
## mds(delta = FCmatrixcontext2, ndim = 3, type = "ordinal")
##
## Model: Symmetric SMACOF
## Number of objects: 24
## Stress-1 value: 0.104
## Number of iterations: 40

# get dim scores
dim3scores_context2 = MDS3Dim_context2$conf

# write dim scores out to a text file in your working directory
write.table(dim3scores_context2,file="MDS_3D_Context2.txt",sep="\t",quote=FALSE)

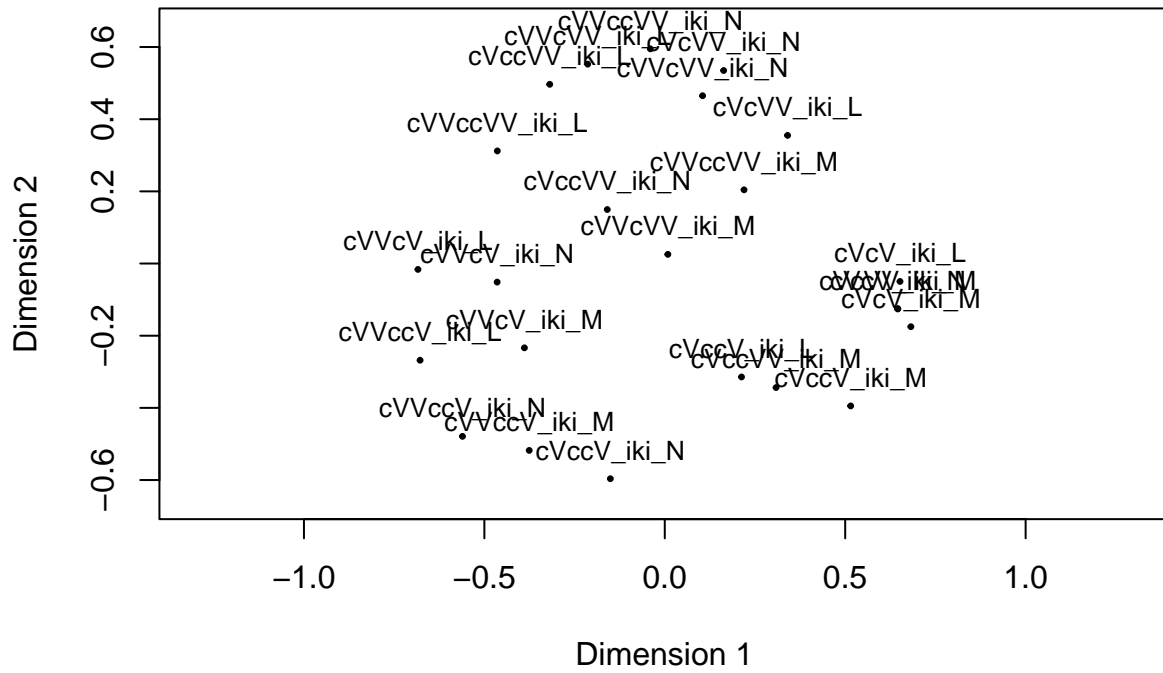
#plot dim 1 by dim 2
plot(MDS3Dim_context2, plot.type = "confplot", plot.dim = c(1,2))
```

Configuration Plot



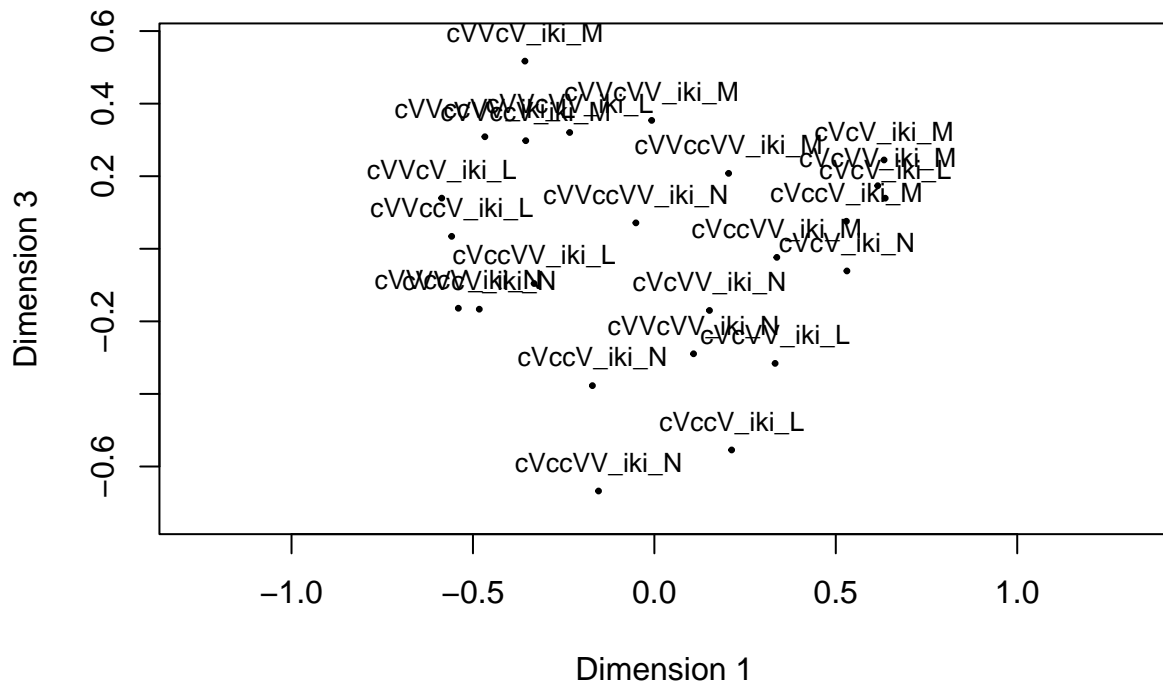
```
#plot dim 1 by dim 3 (as if looking into previous plot from the top)
plot(MDS3Dim_context2, plot.type = "confplot", plot.dim = c(1,3))
```


Configuration Plot



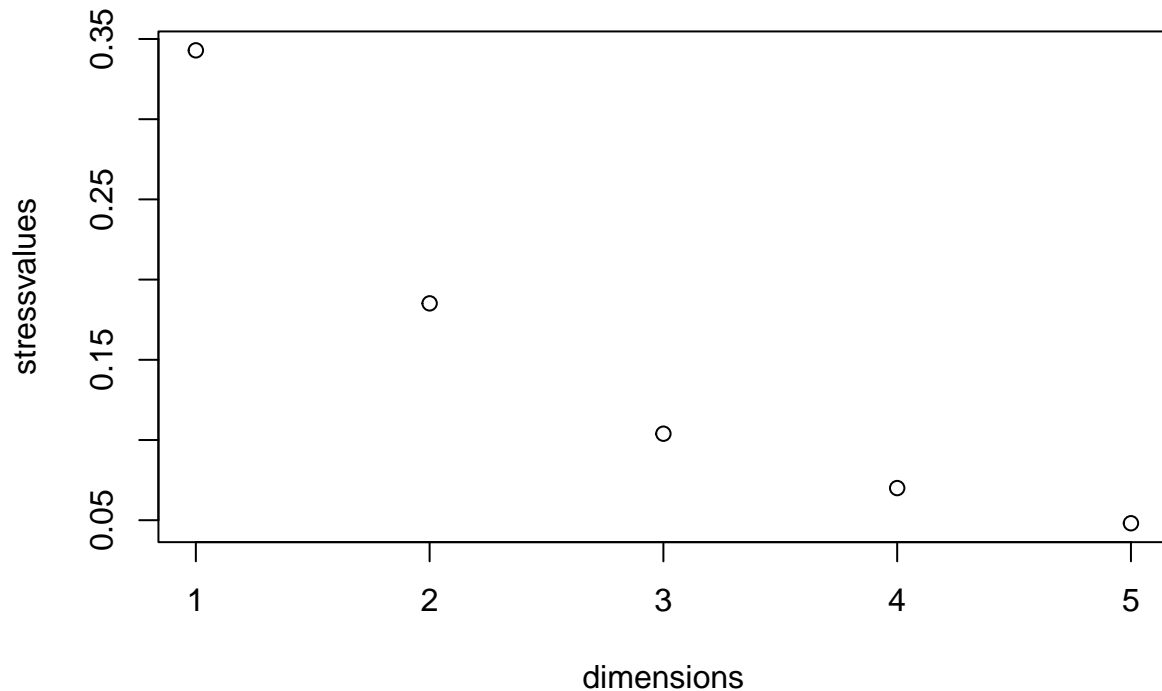
```
#plot dim 1 by dim 4 (as if looking into previous plot from the top)  
plot(MDS4Dim_context2, plot.type = "confplot", plot.dim = c(1,3))
```


Configuration Plot



Context 2: Create stress plot

```
stressvalues = c(MDS1Dim_context2$stress,MDS2Dim_context2$stress,MDS3Dim_context2$stress,MDS4Dim_context2$stress)
dimensions = c(1,2,3,4,5)
plot(dimensions,stressvalues)
```



PART 4: GET MDS SOLUTIONS FOR 1-5 DIMENSIONS FOR CONTEXT 3

Open dataset - Context 3 dissimilarity matrix

```
# Note that the dissimilarity text files are tab delimited files with headers
# If you've changed the default file names from the previous script, you'll need to adjust this
FCmatrixcontext3=read.table("Output_Matrix_Context3_percent_dis.txt",sep="\t",header=TRUE)
```

Context 3: Look at the first few rows of the dataset

```
head(FCmatrixcontext3)
```

```
##          cVccV_ata_L cVccV_ata_M cVccV_ata_N cVccVV_ata_L cVccVV_ata_M
## cVccV_ata_L  0.0000000  0.7307692  0.6538462  0.8846154  0.8461538
## cVccV_ata_M  0.7307692  0.0000000  0.6538462  0.9615385  0.4615385
## cVccV_ata_N  0.6538462  0.6538462  0.0000000  0.7692308  0.9230769
## cVccVV_ata_L 0.8846154  0.9615385  0.7692308  0.0000000  0.8846154
## cVccVV_ata_M 0.8461538  0.4615385  0.9230769  0.8846154  0.0000000
## cVccVV_ata_N 0.8846154  1.0000000  0.8461538  0.4615385  0.8846154
##          cVccVV_ata_N cVcV_ata_L cVcV_ata_M cVcV_ata_N cVcVV_ata_L
## cVccV_ata_L  0.8846154  0.7692308  0.7692308  0.7307692  0.9230769
## cVccV_ata_M  1.0000000  0.2307692  0.3846154  0.3846154  0.8846154
## cVccV_ata_N  0.8461538  0.6923077  0.6923077  0.6538462  0.8076923
## cVccVV_ata_L 0.4615385  0.9230769  0.9230769  1.0000000  0.5769231
## cVccVV_ata_M 0.8846154  0.5384615  0.7692308  0.8076923  0.8076923
## cVccVV_ata_N 0.0000000  0.9230769  0.9230769  0.9615385  0.5769231
```

```

##          cVcVV_ata_M cVcVV_ata_N cVVccV_ata_L cVVccV_ata_M cVVccV_ata_N
## cVccV_ata_L    0.9615385  0.9615385  0.7307692  0.8076923  0.8461538
## cVccV_ata_M    0.7692308  0.9615385  1.0000000  0.8461538  1.0000000
## cVccV_ata_N    0.9230769  0.9615385  0.7692308  0.8076923  0.8846154
## cVccVV_ata_L  0.7307692  0.6538462  0.8846154  0.8846154  0.8461538
## cVccVV_ata_M  0.6153846  0.8461538  0.9230769  0.7307692  0.9615385
## cVccVV_ata_N  0.6923077  0.4615385  0.9230769  0.8846154  0.8846154
##          cVVccVV_ata_L cVVccVV_ata_M cVVccVV_ata_N cVVcV_ata_L cVVcV_ata_M
## cVccV_ata_L    0.6923077  0.8461538  0.8461538  0.8076923  0.8461538
## cVccV_ata_M    0.9230769  0.9230769  0.9230769  1.0000000  0.9230769
## cVccV_ata_N    0.8461538  0.8846154  0.8461538  0.8461538  0.9230769
## cVccVV_ata_L  0.7692308  0.7692308  0.7307692  0.9615385  0.8846154
## cVccVV_ata_M  0.9230769  0.7692308  0.8846154  0.8461538  0.8076923
## cVccVV_ata_N  0.8461538  0.9615385  0.8846154  0.9230769  0.8461538
##          cVVcV_ata_N cVVcVV_ata_L cVVcVV_ata_M cVVcVV_ata_N
## cVccV_ata_L    0.7692308  0.9230769  0.9615385  0.8846154
## cVccV_ata_M    1.0000000  0.9230769  0.8461538  0.9615385
## cVccV_ata_N    0.8461538  0.8461538  0.7307692  0.8461538
## cVccVV_ata_L  0.8076923  0.8076923  0.9230769  0.8846154
## cVccVV_ata_M  0.9230769  0.9230769  0.8461538  1.0000000
## cVccVV_ata_N  0.6923077  0.9615385  0.9230769  0.7692308

```

Context 3: Run ordinal multi-dimensional scaling (MDS) - 1 dimensional solution

```

# ndim is the number of dimensions
MDS1Dim_context3 = mds(FCmatrixcontext3, ndim = 1, type = "ordinal")

# view Kruskal stress
MDS1Dim_context3

##
## Call:
## mds(delta = FCmatrixcontext3, ndim = 1, type = "ordinal")
##
## Model: Symmetric SMACOF
## Number of objects: 24
## Stress-1 value: 0.33
## Number of iterations: 11

# get dim scores
dim1scores_context3 = MDS1Dim_context3$conf

# write dim scores out to a text file in your working directory
write.table(dim1scores_context3,file="MDS_1D_Context3.txt",sep="\t",quote=FALSE)

```

Context 3: Run ordinal multi-dimensional scaling (MDS) - 2 dimensional solution

```

# ndim is the number of dimensions
MDS2Dim_context3 = mds(FCmatrixcontext3, ndim = 2, type = "ordinal")

# view Kruskal stress
MDS2Dim_context3

```

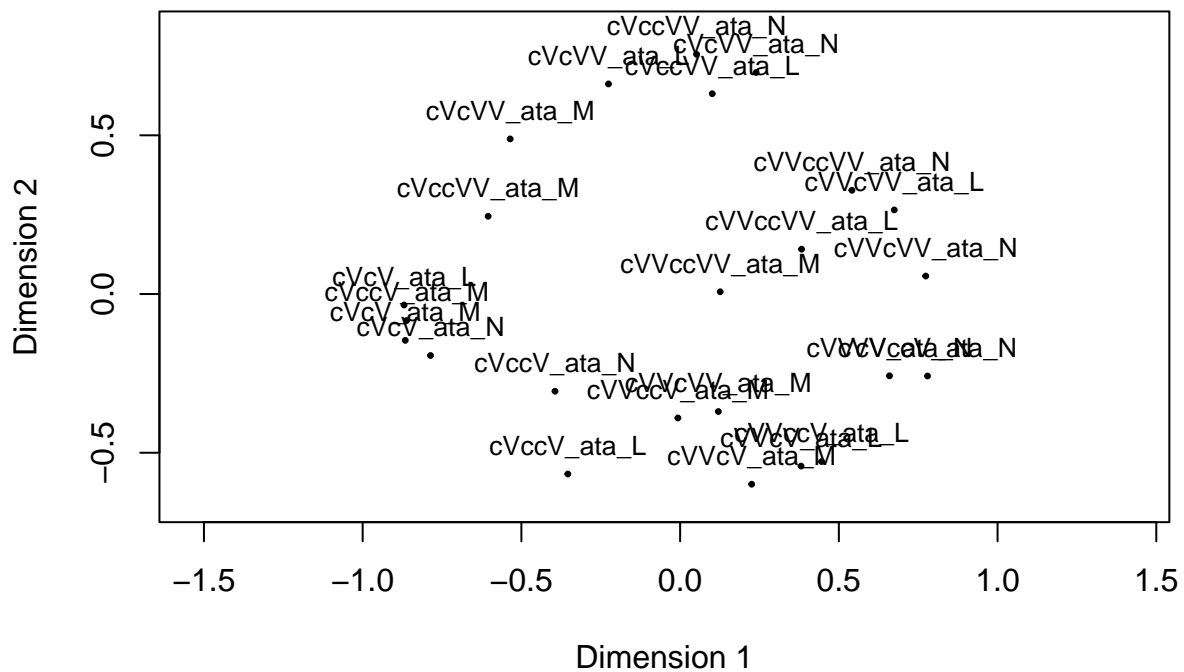
```
##
## Call:
## mds(delta = FCmatrixcontext3, ndim = 2, type = "ordinal")
##
## Model: Symmetric SMACOF
## Number of objects: 24
## Stress-1 value: 0.185
## Number of iterations: 29

# get dim scores
dim2scores_context3 = MDS2Dim_context3$conf

# write dim scores out to a text file in your working directory
write.table(dim2scores_context3,file="MDS_2D_Context3.txt",sep="\t",quote=FALSE)

#plot dim 1 by dim 2
plot(MDS2Dim_context3, plot.type = "confplot", plot.dim = c(1,2))
```

Configuration Plot



Contexts combined: Run ordinal multi-dimensional scaling (MDS) - 3 dimensional solution

```
# ndim is the number of dimensions
MDS3Dim_context3 = mds(FCmatrixcontext3, ndim = 3, type = "ordinal")

# view Kruskal stress
MDS3Dim_context3
```

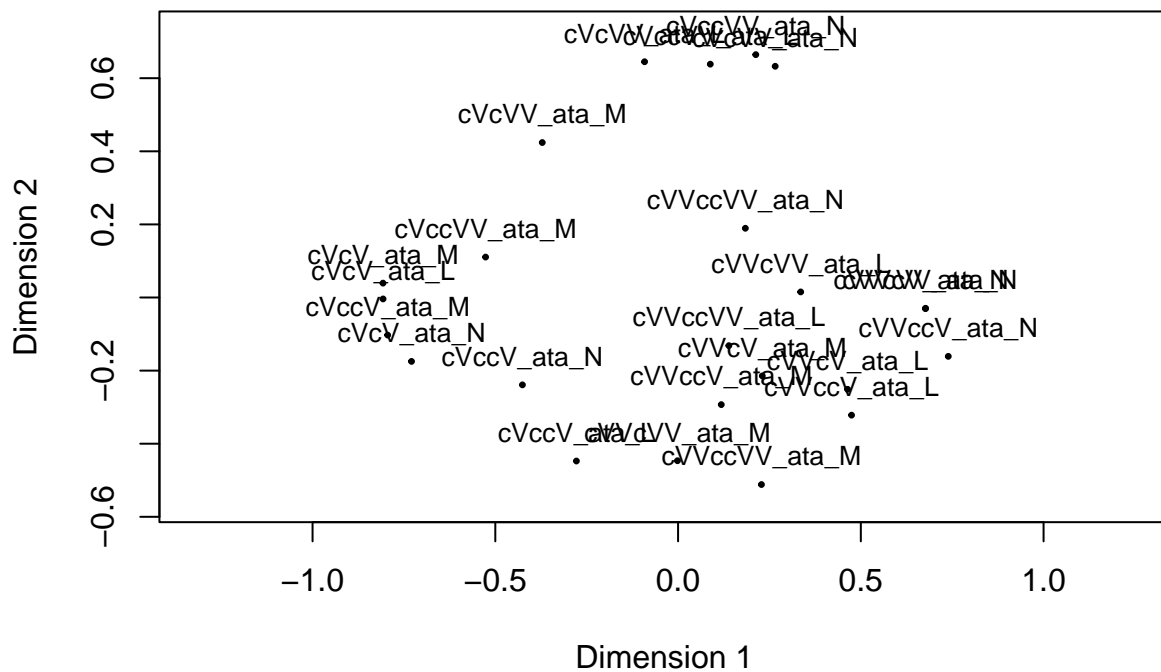
```
##
## Call:
## mds(delta = FCmatrixcontext3, ndim = 3, type = "ordinal")
##
## Model: Symmetric SMACOF
## Number of objects: 24
## Stress-1 value: 0.117
## Number of iterations: 136
```

```
# get dim scores
dim3scores_context3 = MDS3Dim_context3$conf

# write dim scores out to a text file in your working directory
write.table(dim3scores_context3,file="MDS_3D_Context3.txt",sep="\t",quote=FALSE)

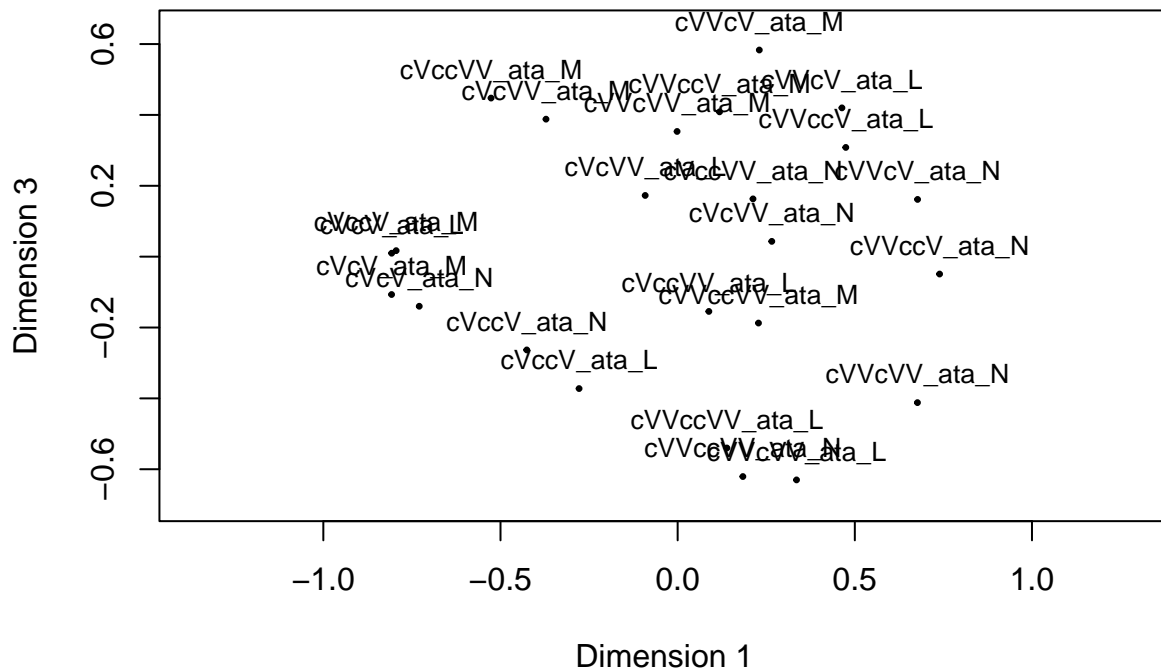
#plot dim 1 by dim 2
plot(MDS3Dim_context3, plot.type = "confplot", plot.dim = c(1,2))
```

Configuration Plot



```
#plot dim 1 by dim 3 (as if looking into previous plot from the top)
plot(MDS3Dim_context3, plot.type = "confplot", plot.dim = c(1,3))
```

Configuration Plot



Context 3: Run ordinal multi-dimensional scaling (MDS) - 4 dimensional solution

```
# ndim is the number of dimensions
MDS4Dim_context3 = mds(FCmatrixcontext3, ndim = 4, type = "ordinal")

# view Kruskal stress
MDS4Dim_context3

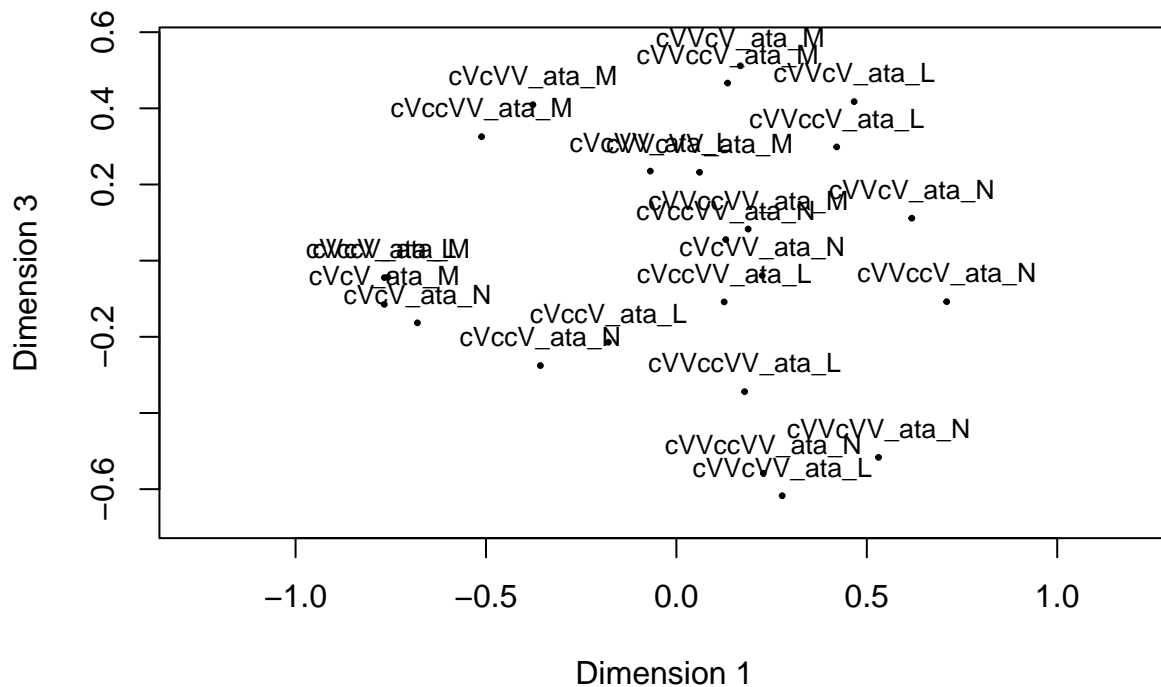
##
## Call:
## mds(delta = FCmatrixcontext3, ndim = 4, type = "ordinal")
##
## Model: Symmetric SMACOF
## Number of objects: 24
## Stress-1 value: 0.074
## Number of iterations: 84

# get dim scores
dim4scores_context3 = MDS4Dim_context3$conf

# write dim scores out to a text file in your working directory
write.table(dim4scores_context3, file="MDS_4D_Context3.txt", sep="\t", quote=FALSE)

#plot dim 1 by dim 2
plot(MDS4Dim_context3, plot.type = "confplot", plot.dim = c(1,2))
```


Configuration Plot



Context 3: Run ordinal multi-dimensional scaling (MDS) - 5 dimensional solution

```
# ndim is the number of dimensions
MDS5Dim_context3 = mds(FCmatrixcontext3, ndim = 5, type = "ordinal")

# view Kruskal stress
MDS5Dim_context3

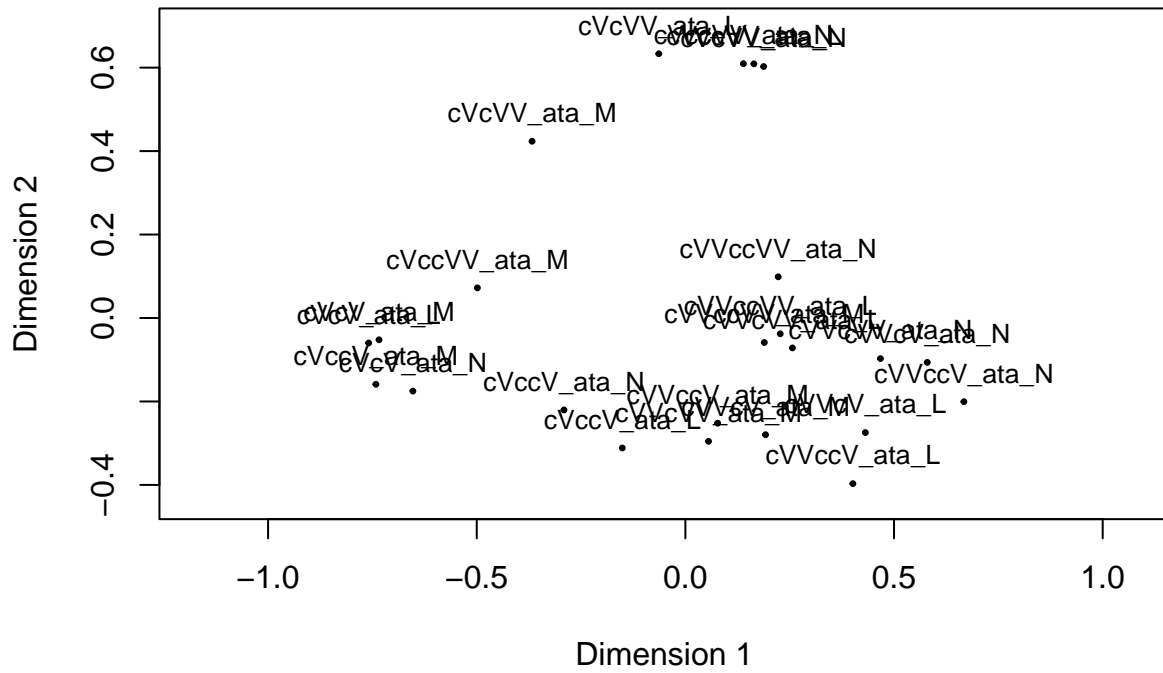
##
## Call:
## mds(delta = FCmatrixcontext3, ndim = 5, type = "ordinal")
##
## Model: Symmetric SMACOF
## Number of objects: 24
## Stress-1 value: 0.049
## Number of iterations: 120

# get dim scores
dim5scores_context3 = MDS5Dim_context3$conf

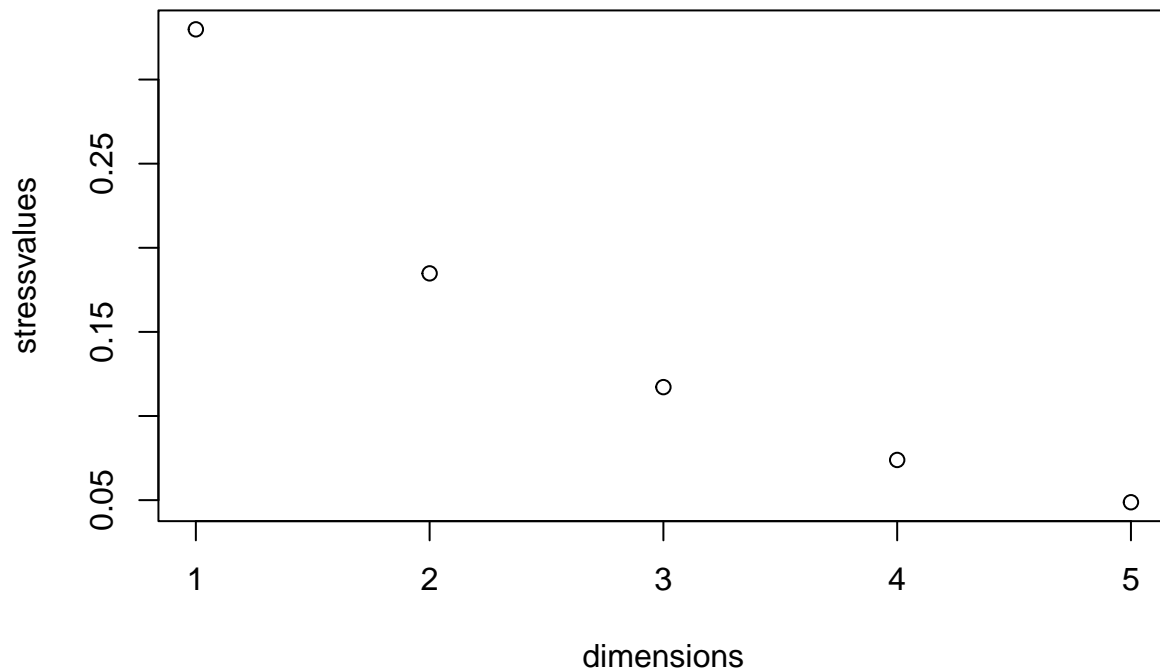
# write dim scores out to a text file in your working directory
write.table(dim5scores_context3, file="MDS_5D_Context3.txt", sep="\t", quote=FALSE)

#plot dim 1 by dim 2
plot(MDS5Dim_context3, plot.type = "confplot", plot.dim = c(1,2))
```

Configuration Plot



```
#plot dim 1 by dim 5 (as if looking into previous plot from the top)  
plot(MDS5Dim_context3, plot.type = "confplot", plot.dim = c(1,3))
```

SECTION 2: CALCULATE EUCLIDEAN DISTANCES BETWEEN DIMENSION SCORES

Note that all speakers are combined in these analyses in order to get an average distance between categories.

Part 1: CALCULATE EUCLIDEAN DISTANCES FOR 2D & 3D SOLUTIONS FOR CONTEXTS COMBINED

Contexts Combined: Calculate Euclidean distances between dimension scores for 2-dimensional solution

```
# read in file for all contexts and all speakers combined
FCmatrixAll_all = read.table("Output_Matrix_AllContexts_AllSpeakers_percent_dis.txt", sep="\t", header=TRUE)

# run MDS analysis
MDS2Dim_all_all = mds(FCmatrixAll_all, ndim = 2, type = "ordinal")

# save distances calculated with mds() and convert to a symmetrical matrix for saving to file
dist2D_all_all = as.matrix(MDS2Dim_all_all$confdist)

# write distance matrix to a file in your working directory
write.csv(dist2D_all_all, "Dist2D_AllContexts.csv", row.names = TRUE)
```

Contexts Combined: Calculate Euclidean distances between dimension scores for 3-dimensional solution

```
# read in file for all contexts and all speakers combined
FCmatrixAll_all = read.table("Output_Matrix_AllContexts_AllSpeakers_percent_dis.txt", sep="\t", header=TRUE)

# run MDS analysis
```

```
MDS3Dim_all_all = mds(FCmatrixAll_all, ndim = 3, type = "ordinal")

# save distances calculated with mds() and convert to a symmetrical matrix for saving to file
dist3D_all_all = as.matrix(MDS2Dim_all_all$confdist)

# write distance matrix to a file in your working directory
write.csv(dist3D_all_all, "Dist3D_AllContexts.csv", row.names = TRUE)
```

Part 2: CALCULATE EUCLIDEAN DISTANCES FOR 2D & 3D SOLUTIONS FOR CONTEXT 1

Context 1: Calculate Euclidean distances between dimension scores for 2-dimensional solution

```
# read in file for Context 1 and all speakers combined
FCmatrixcontext1_all = read.table("Output_Matrix_Context1_AllSpeakers_percent_dis.txt", sep="\t", header=

# run MDS analysis
MDS2Dim_context1_all = mds(FCmatrixcontext1_all, ndim = 2, type = "ordinal")

# save distances calculated with mds() and convert to a symmetrical matrix for saving to file
dist2D_context1_all = as.matrix(MDS2Dim_context1_all$confdist)

# write distance matrix to a file in your working directory
write.csv(dist2D_context1_all, "Dist2D_Context1.csv", row.names = TRUE)
```

Context 1: Calculate Euclidean distances between dimension scores for 3-dimensional solution

```
# read in file for Context 1 and all speakers combined
FCmatrixcontext1_all = read.table("Output_Matrix_Context1_AllSpeakers_percent_dis.txt", sep="\t", header=

# run MDS analysis
MDS3Dim_context1_all = mds(FCmatrixcontext1_all, ndim = 3, type = "ordinal")

# save distances calculated with mds() and convert to a symmetrical matrix for saving to file
dist3D_context1_all = as.matrix(MDS3Dim_context1_all$confdist)

# write distance matrix to a file in your working directory
write.csv(dist3D_context1_all, "Dist3D_Context1.csv", row.names = TRUE)
```

Part 3: CALCULATE EUCLIDEAN DISTANCES FOR 2D & 3D SOLUTIONS FOR CONTEXT 2

Context 2: Calculate Euclidean distances between dimension scores for 2-dimensional solution

```
# read in file for Context 2 and all speakers combined
FCmatrixcontext2_all = read.table("Output_Matrix_Context2_AllSpeakers_percent_dis.txt", sep="\t", header=

# run MDS analysis
MDS2Dim_context2_all = mds(FCmatrixcontext2_all, ndim = 2, type = "ordinal")

# save distances calculated with mds() and convert to a symmetrical matrix for saving to file
dist2D_context2_all = as.matrix(MDS2Dim_context2_all$confdist)

# write distance matrix to a file in your working directory
write.csv(dist2D_context2_all, "Dist2D_Context2.csv", row.names = TRUE)
```

Context 2: Calculate Euclidean distances between dimension scores for 3-dimensional solution

```
# read in file for Context 2 and all speakers combined
FCmatrixcontext2_all = read.table("Output_Matrix_Context2_AllSpeakers_percent_dis.txt", sep="\t", header=
# run MDS analysis
MDS3Dim_context2_all = mds(FCmatrixcontext2_all, ndim = 3, type = "ordinal")
# save distances calculated with mds() and convert to a symmetrical matrix for saving to file
dist3D_context2_all = as.matrix(MDS3Dim_context2_all$confdist)
# write distance matrix to a file in your working directory
write.csv(dist3D_context2_all, "Dist3D_Context2.csv", row.names = TRUE)
```

Part 4: CALCULATE EUCLIDEAN DISTANCES FOR 2D & 3D SOLUTIONS FOR CONTEXT 3

Context 3: Calculate Euclidean distances between dimension scores for 2-dimensional solution

```
# read in file for Context 3 and all speakers combined
FCmatrixcontext3_all = read.table("Output_Matrix_Context3_AllSpeakers_percent_dis.txt", sep="\t", header=
# run MDS analysis
MDS2Dim_context3_all = mds(FCmatrixcontext3_all, ndim = 2, type = "ordinal")
# save distances calculated with mds() and convert to a symmetrical matrix for saving to file
dist2D_context3_all = as.matrix(MDS2Dim_context3_all$confdist)
# write distance matrix to a file in your working directory
write.csv(dist2D_context3_all, "Dist2D_Context3.csv", row.names = TRUE)
```

Context 3: Calculate Euclidean distances between dimension scores for 3-dimensional solution

```
# read in file for Context 3 and all speakers combined
FCmatrixcontext3_all = read.table("Output_Matrix_Context3_AllSpeakers_percent_dis.txt", sep="\t", header=
# run MDS analysis
MDS3Dim_context3_all = mds(FCmatrixcontext3_all, ndim = 3, type = "ordinal")
# save distances calculated with mds() and convert to a symmetrical matrix for saving to file
dist3D_context3_all = as.matrix(MDS3Dim_context3_all$confdist)
# write distance matrix to a file in your working directory
write.csv(dist3D_context3_all, "Dist3D_Context3.csv", row.names = TRUE)
```