Multidimensional Scaling

MATH 381 HW 7

Tim Ferrin, 1764309

December 4, 2020

I Introduction

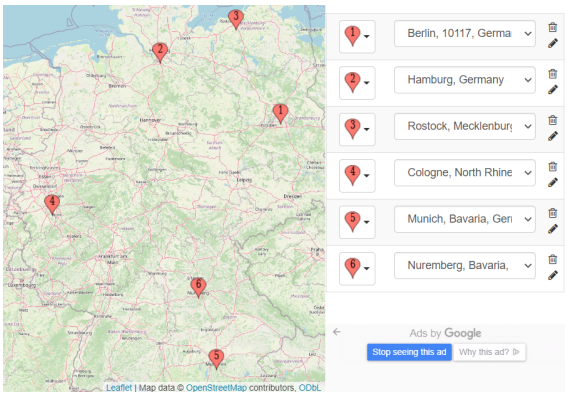
The focus of this paper is to analyze multidimensional scaling (MDS) on two sets of cities. The first set being examined includes Berlin, Hamburg, Rostock, Köln (Cologne), München (Munich), and Nürnberg (Nuremberg), all of which are in Germany. The second set includes Shanghai, Delhi, São Paulo, Cairo, London, and New York City. First, a matrix of distances between the pairs of cities is established for each of the sets, using the DistanceFromTo website (https://www.distancefromto.net) which gives the air travel distances between any two cities. The two matrices are listed below, with the distances being measure in kilometers.

City	Berlin	Hamburg	Rostock	Köln	München	Nürnberg
Berlin	0	255.71	193.82	477.68	504.5	378.6
Hamburg	255.71	0	152.67	356.36	612.29	462.44
Rostock	193.82	152.67	0	494.93	662.85	520.95
Köln	477.68	356.36	494.93	0	456.1	336.81
München	504.5	612.29	662.85	456.1	0	150.34
Nürnberg	378.6	462.44	520.95	336.81	150.34	0

City	Shanghai	Delhi	Sao Paulo	Cairo	London	New York
Shanghai	0	4,242.61	18,562.73	8,352.16	9,196.34	11,860.47
Delhi	4,242.61	0	14,429.87	4,429.78	6,710.47	11,753.2
Sao Paulo	18,562.73	14,429.87	0	10,219.89	9,496.92	7,685.32
Cairo	8,352.16	4,429.78	10,219.89	0	3,510.86	9,022.1
London	9,196.34	6,710.47	9,496.92	3,510.86	0	5,570.64
New York	11,860.47	11,753.2	7,685.32	9,022.1	5,570.64	0

These distances are then plugged into the cmdscale() function in R, using 1, 2, and 3 dimensions. The results of these computations are examined in the next section. As a reference,

two maps with the sets of cities flagged are provided in Figures 1 and 2, obtained from screen shotting the MapCustomizer website (https://www.mapcustomizer.com).



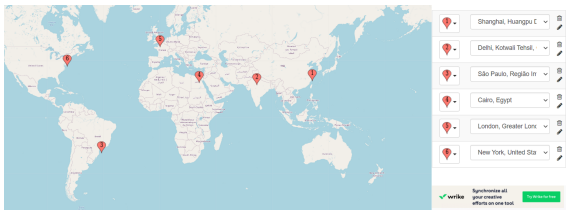
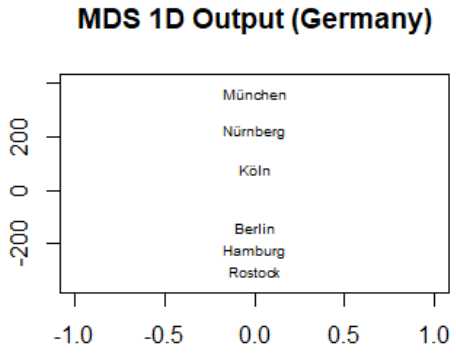


Figure 1: (left) German cities set flagged

Figure 2: (right) World cities set flagged

II MDS Output

The 1D MDS of the cities in Germany result in the orientation shown in Figure 3, where a city's location is specified by the middle of it's word location. Since MDS doesn't know the actual layout of a map of Germany, it produces this output. If we negate the location values for each city, we obtain Figure 4. Note that this roughly shows the cities' relative latitudes, eg. Rostock is the northern-most city and München is the southern-most city.



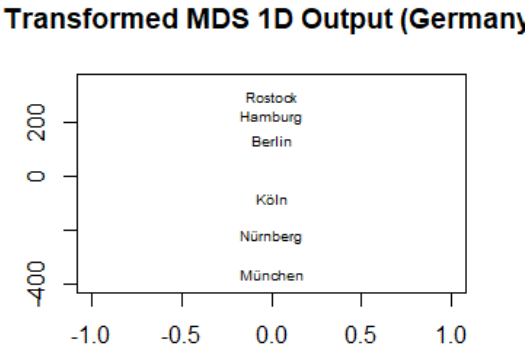
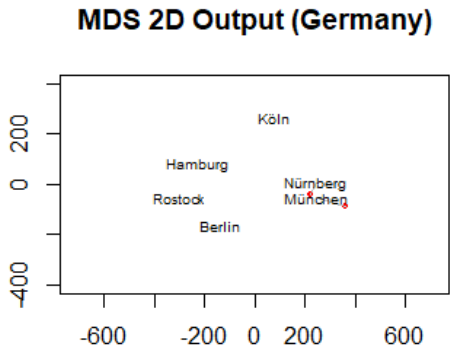


Figure 3: (left) 1D cmdscale() output

Figure 4: (right) Natural orientation

Similarly, the 2D MDS of the cities in Germany result in the orientation shown in Figure 5. By rotating the created map 90° clockwise and reflecting across the y-axis, you can obtain

a reasonable approximation of the actual layout of the German cities, shown in Figure 6.



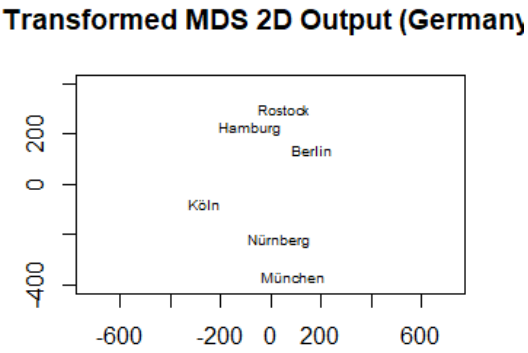
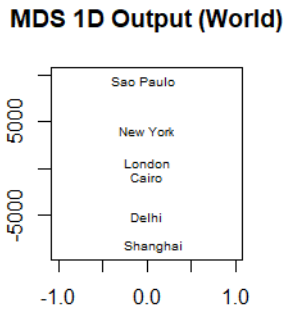
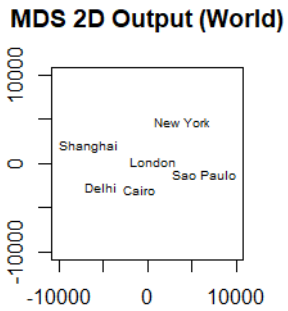


Figure 5: (left) 2D cmdscale() output

Figure 6: (right) Natural orientation

The plot for the 3D reduction is essentially the 2D reduction plane embedded in a 3 dimensional space. specifically, the z values for the cities are all within the interval (-2,2).
The 1D, 2D, and 3D plots for the world cities are respectively shown in Figures 7, 8, and 9 below.





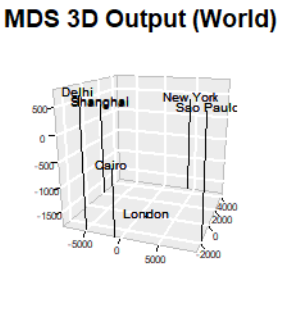


Figure 7: (left) 1D cmdscale() output

Figure 8: (center) 2D cmdscale() output

Figure 9: (right) 3D cmdscale() output

It is difficult to distinguish any sort of meaning from any of these three figures, which is by design. MDS only works well when the distances between objects are Euclidean. Since the world set cities lie on a sphere and their air distances aren't Euclidean, these plots do not at all resemble the map in Figure 2. Conversely, since the German set cities are somewhat

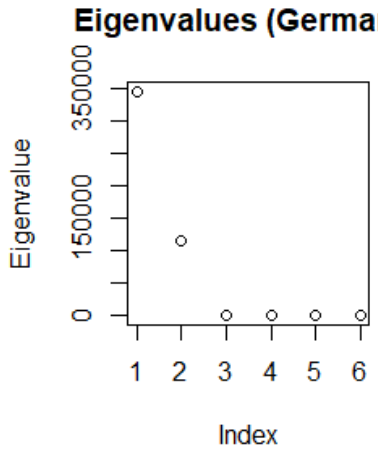
close together, their air distances represent Euclidean geometry well enough for the MDS algorithm to establish some sort of pattern.

III Model Efficacy

Hinted at in the previous section, some of the models are better at approximating the data than others. The table below lists the GOF (a correlational metric) for each model.

City Set	1D	2D	3D
Germany	0.7496325	0.9999088	0.9999201
World	0.7522914	0.9170490	0.9340454

This shows that only two dimensions are needed to accurately model the Germany cities, but at least three dimensions are needed for the world cities. We can also see this in the eigenvalue plots for each city set in Figures 10 and 11. As shown, all eigenvalues after in-



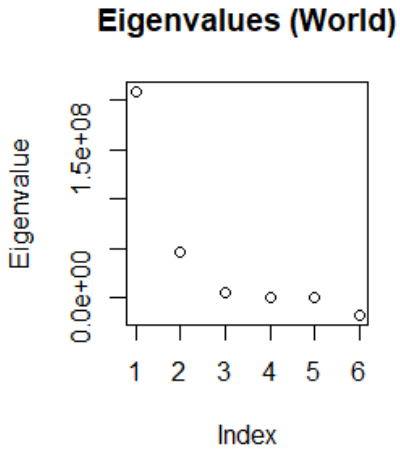


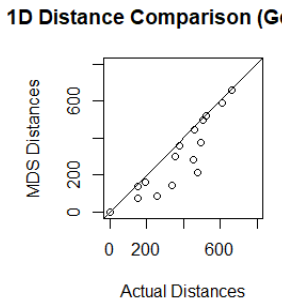
Figure 10: (left) 2D cmdscale() output

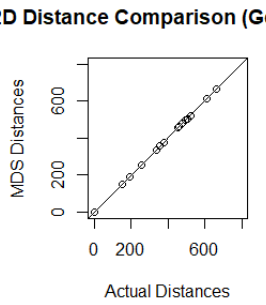
Figure 11: (right) Natural orientation

dex two for the Germany set are essentially 0, but there are four non-zero eigenvalues for the world set. The sixth eigenvalue for the world set is negative, which indicates that the distances between these cities are non-Euclidean, as one would expect.

Next we plot the modeled vs actual distances between the cities for each set, shown below in Figures 12-17.

From these we can draw similar conclusions to those previously discussed. In the ideal model, all points in these plots would be along the $y = x$ line, meaning that the actual and modeled distances are the same. Note that for the German city set, the 2D graph points are all along





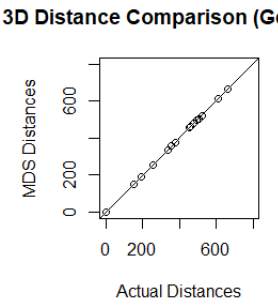
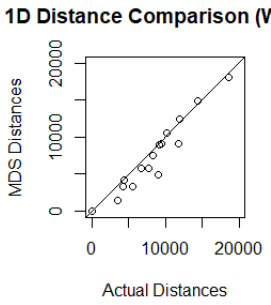
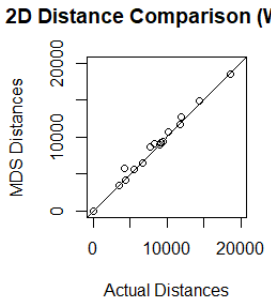


Figure 12: (left) Actual vs 1D MDS distances

Figure 13: (center) Actual vs 2D MDS distances

Figure 14: (right) Actual vs 3D MDS distances





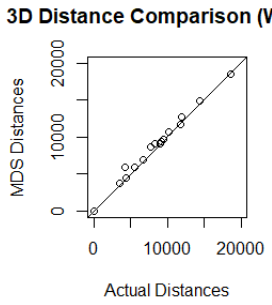


Figure 15: (left) Actual vs 1D MDS distances

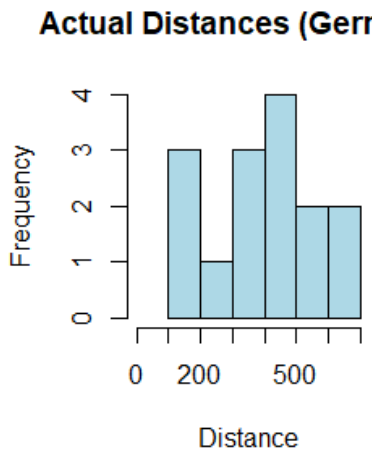
Figure 16: (center) Actual vs 2D MDS distances

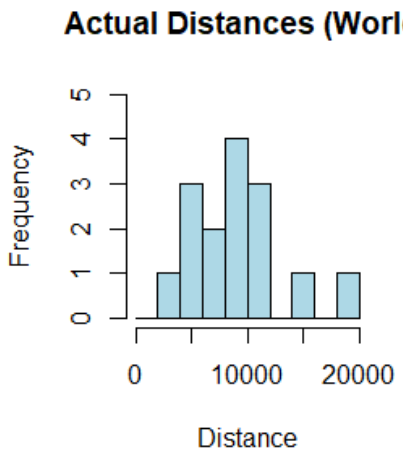
Figure 17: (right) Actual vs 3D MDS distances

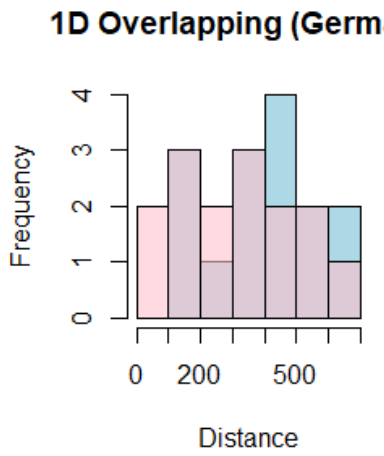
the line, but for the world city set, the 3D graph points are still not quite all lying on the $y = x$ line.

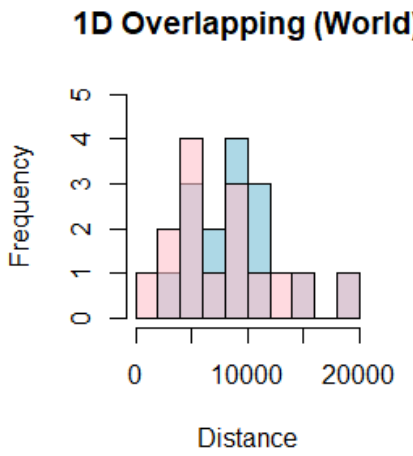
Additionally, histograms of the distances for each model are provided below to compare with the actual distance histograms. Peruse them as you wish.

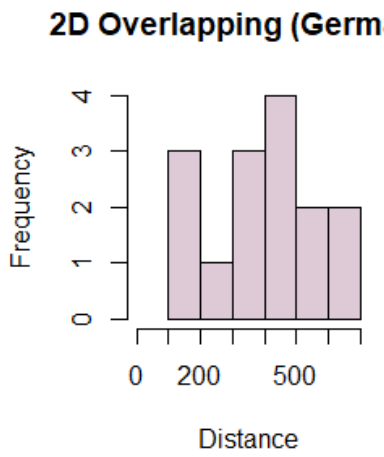
Finally, the maximum and minimum percent differences and the pairs of cities which correspond to these differences for each of the models is given in the table below. Not much, that hasn't previously stated, can be gleaned from this table. Once again, we can see from the drop in max percentage difference that the German cities are well modelled by two dimensions.

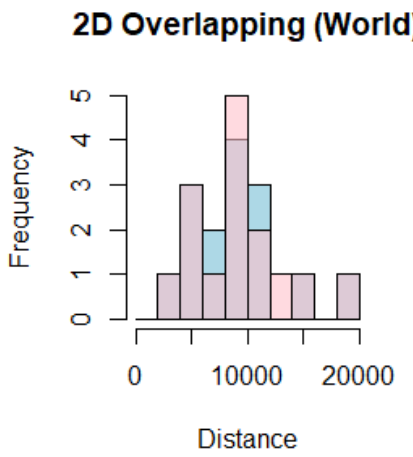

Actual Distances (Germany)
Frequency
0
1
2
3
4
0
200
500
Distance

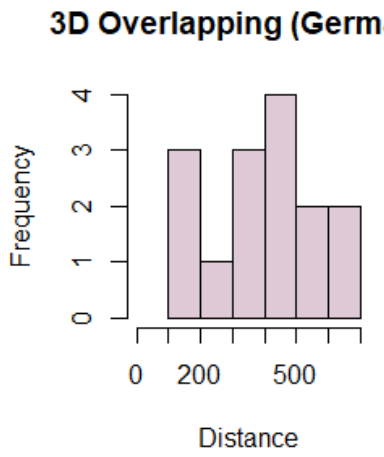

Actual Distances (World)
Frequency
0
1
2
3
4
5
0
10000
20000
Distance

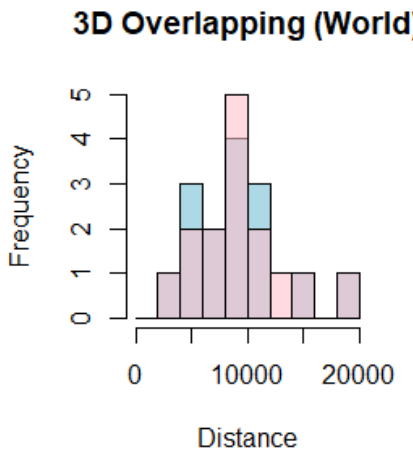

1D Overlapping (Germany)
Frequency
0
1
2
3
4
0
200
500
Distance


1D Overlapping (World)
Frequency
0
1
2
3
4
5
0
10000
20000
Distance


2D Overlapping (Germany)
Frequency
0
1
2
3
4
0
200
500
Distance


2D Overlapping (World)
Frequency
0
1
2
3
4
5
0
10000
20000
Distance





City Set	Dim.	Max % Diff	Max Diff Pair	Min % Diff	Min Diff Pair
Germany	**1D**	65.23229	Berlin-Hamburg	0.04903486	Rostock-Nürnberg
	2D	0.0355409	Berlin-Hamburg	2.543624e-05	Hamburg-München
	3D	0.03580851	Berlin-Hamburg	0.0002687652	Hamburg-München
World	**1D**	58.1873	Cairo-London	1.834423	Shanghai-London
	2D	38.1676	Shanghai-Delhi	0.003456947	Cairo-New York
	3D	38.61258	Shanghai-Delhi	0.03091561	Shanghai-São Paulo

IV Code

The following is the R code used for evaluating the data and generating all the plots/histograms used:

```
# Loading in the data for part 1
dists <- read.csv(file="C:/Users/timfe/Downloads/GermanyDistances.csv",head=FALSE,sep=",
distsplusnames <- read.csv(file="C:/Users/timfe/Downloads/GermanyCities.csv",head=TRUE,s

# Computing the MDS coordinates
model11a <- cmdscale(dists,k=1)
model12 <- cmdscale(dists,k=2)
model13 <- cmdscale(dists,k=3)
library(wordcloud)

# Plotting the various dimension MDS outputs and their natural transforms
model11 <- cbind(0,model11a)
textplot(model11[,1],model11[,2],
   gsub("(^[^\\s]+\\s{1})","",
   distsplusnames$City,perl=TRUE),
```

```
   ylim=c(-350,400),xlim=c(-1,1),xlab="",ylab="",cex=0.6)
title("MDS 1D Output (Germany)")
transLocs11 <- -1*model11
textplot(transLocs11[,1],transLocs11[,2],
   gsub("(^[^\\s]+\\s{1})","",
   distsplusnames$City,perl=TRUE),
   ylim=c(-400,350),xlim=c(-1,1),xlab="",ylab="",cex=0.6)
title("Transformed MDS 1D Output (Germany)")

textplot(model12[,1],model12[,2],asp=1,
   gsub("(^[^\\s]+\\s{1})","",
   distsplusnames$City,perl=TRUE),
   xlim=c(-400,400),ylim=c(-400,400),xlab="",ylab="",cex=0.6)
title("MDS 2D Output (Germany)")
transMat <- cbind(c(0,-1),c(-1,0))
transLocs12 <- t(transMat %*% t(model12))
textplot(transLocs12[,1],transLocs12[,2],asp=1,
         gsub("(^[^\\s]+\\s{1})","",
              distsplusnames$City,perl=TRUE),
         xlim=c(-400,400),ylim=c(-400,400),xlab="",ylab="",cex=0.6)
title("Transformed MDS 2D Output (Germany)")

library("plot3D")
cityNames <- distsplusnames[,1]
text3D(model13[,1],model13[,2],model13[,3],
   cityNames,colvar=NULL,phi=15,theta=20,cex=0.6,adj=0.5,
   bty="g",ticktype="detailed",cex.axis=0.5,d=2,asp=1,
   xlab="",ylab="",zlab="")
scatter3D(model13[,1],model13[,2],model13[,3]-0.15,
   colvar=NULL,type="h",pch=".",add=TRUE)
title("MDS 3D Output (Germany)")

# Computing the MDS coordinates (again, but with eigenvalues and GOF)
cmdscale(dists,k=1,eig=TRUE)$GOF
cmdscale(dists,k=2,eig=TRUE)$GOF
cmdscale(dists,k=3,eig=TRUE)$GOF
plot(cmdscale(dists,k=3,eig=TRUE)$eig,ylab="Eigenvalue",main="Eigenvalues (Germany)")

# Comparing the MDS distances to the actual distances
mdsDist11 <- as.matrix(dist(model11a,diag=TRUE,upper=TRUE))
mdsDist12 <- as.matrix(dist(model12,diag=TRUE,upper=TRUE))
mdsDist13 <- as.matrix(dist(model13,diag=TRUE,upper=TRUE))
par(pty="s")
plot(unlist(dists),as.vector(mdsDist11),asp=1,xlim=c(0,800),ylim=c(0,800),
    xlab="Actual Distances",ylab="MDS Distances")
```

```
title("1D Distance Comparison (Germany)")
abline(0,1)
plot(unlist(dists),as.vector(mdsDist12),asp=1,xlim=c(0,800),ylim=c(0,800),
     xlab="Actual Distances",ylab="MDS Distances")
title("2D Distance Comparison (Germany)")
abline(0,1)
plot(unlist(dists),as.vector(mdsDist13),asp=1,xlim=c(0,800),ylim=c(0,800),
     xlab="Actual Distances",ylab="MDS Distances")
title("3D Distance Comparison (Germany)")
abline(0,1)

# Plotting the distance histograms
c1 <- rgb(173,216,230,max = 255, alpha = 255, names = "lt.blue")
c2 <- rgb(255,192,203,max = 255, alpha = 150, names = "lt.pink")
hgAct <- hist(as.dist(dists),col=c1,xlim=c(0,700),main="Actual Distances (Germany)",xlab
hg11 <- hist(as.dist(mdsDist11),col=c2,ylim=c(0,4),main="MDS 1D Distances (Germany)",xla
plot(hgAct,col=c1,xlim=c(0,700),main="1D Overlapping (Germany)",xlab="Distance")
plot(hg11,col=c2,add=TRUE)
hg12 <- hist(as.dist(mdsDist12),col=c2,xlim=c(0,700),ylim=c(0,4),main="MDS 2D Distances
plot(hgAct,col=c1,xlim=c(0,700),main="2D Overlapping (Germany)",xlab="Distance")
plot(hg12,col=c2,add=TRUE)
hg13 <- hist(as.dist(mdsDist13),col=c2,xlim=c(0,700),ylim=c(0,4),main="MDS 2D Distances
plot(hgAct,col=c1,xlim=c(0,700),main="3D Overlapping (Germany)",xlab="Distance")
plot(hg13,col=c2,add=TRUE)

# Percent differences in distances
percDifs11 <- 100*(dists-mdsDist11)/dists
percDifs12 <- 100*(dists-mdsDist12)/dists
percDifs13 <- 100*(dists-mdsDist13)/dists
max(abs(percDifs11),na.rm=TRUE)
max(abs(percDifs12),na.rm=TRUE)
max(abs(percDifs13),na.rm=TRUE)
min(abs(percDifs11),na.rm=TRUE)
min(abs(percDifs12),na.rm=TRUE)
min(abs(percDifs13),na.rm=TRUE)

#
# Do everything again, but with global cities
#
# Loading in the data for part 1
dists2 <- read.csv(file="C:/Users/timfe/Downloads/WorldDistances.csv",head=FALSE,sep=","
distsplusnames2 <- read.csv(file="C:/Users/timfe/Downloads/WorldCities.csv",head=TRUE,se

# Computing the MDS coordinates
model21a <- cmdscale(dists2,k=1)
```

```
model22 <- cmdscale(dists2,k=2)
model23 <- cmdscale(dists2,k=3)
library(wordcloud)

# Plotting the various dimension MDS outputs and their natural transforms
model21 <- cbind(0,model21a)
textplot(model21[,1],model21[,2],distsplusnames2$City,
   ylim=c(-9000,10000),xlim=c(-1,1),xlab="",ylab="",cex=0.6)
title("MDS 1D Output (World)")
transLocs21 <- -1*model21

textplot(model22[,1],model22[,2],asp=1,distsplusnames2$City,
   xlim=c(-10000,10000),ylim=c(-10000,10000),xlab="",ylab="",cex=0.6)
title("MDS 2D Output (World)")

library("plot3D")
cityNames <- distsplusnames2[,1]
text3D(model23[,1],model23[,2],model23[,3],
   cityNames,colvar=NULL,phi=17,theta=20,cex=0.6,adj=0.5,
   bty="g",ticktype="detailed",cex.axis=0.5,d=2,asp=1,
   xlab="",ylab="",zlab="")
scatter3D(model23[,1],model23[,2],model23[,3]-100,
          colvar=NULL,type="h",pch=".",add=TRUE)
title("MDS 3D Output (World)")

# Computing the MDS coordinates (again, but with eigenvalues and GOF)
cmdscale(dists2,k=1,eig=TRUE)$GOF
cmdscale(dists2,k=2,eig=TRUE)$GOF
cmdscale(dists2,k=3,eig=TRUE)$GOF
plot(cmdscale(dists2,k=3,eig=TRUE)$eig,ylab="Eigenvalue",main="Eigenvalues (World)")

# Comparing the MDS distances to the actual distances
mdsDist21 <- as.matrix(dist(model21a,diag=TRUE,upper=TRUE))
mdsDist22 <- as.matrix(dist(model22,diag=TRUE,upper=TRUE))
mdsDist23 <- as.matrix(dist(model23,diag=TRUE,upper=TRUE))
par(pty="s")
plot(unlist(dists2),as.vector(mdsDist21),asp=1,xlim=c(0,20000),ylim=c(0,20000),
     xlab="Actual Distances",ylab="MDS Distances")
title("1D Distance Comparison (World)")
abline(0,1)
plot(unlist(dists2),as.vector(mdsDist22),asp=1,xlim=c(0,20000),ylim=c(0,20000),
     xlab="Actual Distances",ylab="MDS Distances")
title("2D Distance Comparison (World)")
abline(0,1)
plot(unlist(dists2),as.vector(mdsDist23),asp=1,xlim=c(0,20000),ylim=c(0,20000),
```

```
     xlab="Actual Distances",ylab="MDS Distances")
title("3D Distance Comparison (World)")
abline(0,1)

# Plotting the distance histograms
hgAct <- hist(as.dist(dists2),breaks=0:10*2000,ylim=c(0,5),col=c1,main="Actual Distances
hg21 <- hist(as.dist(mdsDist21),breaks=0:10*2000,ylim=c(0,5),col=c2,main="MDS 1D Distanc
plot(hgAct,ylim=c(0,5),col=c1,main="1D Overlapping (World)",xlab="Distance")
plot(hg21,ylim=c(0,5),col=c2,add=TRUE)
hg22 <- hist(as.dist(mdsDist22),breaks=0:10*2000,ylim=c(0,5),col=c2,main="MDS 2D Distanc
plot(hgAct,col=c1,ylim=c(0,5),main="2D Overlapping (World)",xlab="Distance")
plot(hg22,col=c2,add=TRUE)
hg23 <- hist(as.dist(mdsDist23),breaks=0:10*2000,ylim=c(0,5),col=c2,main="MDS 2D Distanc
plot(hgAct,col=c1,ylim=c(0,5),main="3D Overlapping (World)",xlab="Distance")
plot(hg23,col=c2,add=TRUE)

# Percent differences in distances
percDifs21 <- 100*(dists2-mdsDist21)/dists2
percDifs22 <- 100*(dists2-mdsDist22)/dists2
percDifs23 <- 100*(dists2-mdsDist23)/dists2
max(abs(percDifs21),na.rm=TRUE)
max(abs(percDifs22),na.rm=TRUE)
max(abs(percDifs23),na.rm=TRUE)
min(abs(percDifs21),na.rm=TRUE)
min(abs(percDifs22),na.rm=TRUE)
min(abs(percDifs23),na.rm=TRUE)

# Copies the plots currently open to the specified directory
plots.dir.path <- list.files(tempdir(), pattern="rs-graphics", full.names = TRUE);
plots.png.paths <- list.files(plots.dir.path, pattern=".png", full.names = TRUE);
file.copy(from=plots.png.paths, to="C:/Users/timfe/Desktop/MATH 381/HW6")
```