**3.1 - Multidimensional Scaling**

R Packages: ade4, MASS (comes with base R), rgl, vegan, cba, cba, proxy, cluster, corrplot

The idea is behind multidimensional scaling is quite simple, although the computational aspects required to implement it are not. Generally, we start with an data matrix , where rows represent observations and columns are variables.

Next, we compute pairwise distances or dissimilarities between the *n* observations, using appropriate metrics, variable recoding, and scaling if necessary, to form a distance matrix .

Finally, we then seek to find a *lower dimensional* representation so that the “distances” between observations in the lower dimension (typically 2 or 3) are “close” in some sense to the original distances in



Closeness is measured by different measures of ***stress*** or distortion. If the distances in are true distances then we seek a lower dimensional representation that matches those distances as closely as possible in a least squares sense. This is called **metric** or **classic multidimensional scaling**. If the distance metric is Euclidean distance on scaled or unscaled variables then metric MDS is the same as principal component analysis (PCA) which we will be discussing in a later section in this course. If however, you use a different metric to measure distance between observations (e.g. Manhattan distance/taxi cab metric) then the results of metric MDS are not the same as PCA.



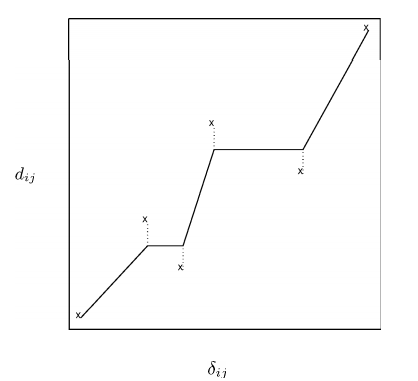
However, if the original “distances” are not true distances and only reflect how dissimilar observations are from each other, then we don’t need to try to match the dissimilarities exactly. Rather we try construct distances in the lower dimensional space that retains the same rank ordering as the original dissimilarities. For example, you might know that objects 1 and 2 are more distant/dissimilar than objects 2 and 3, so you try to quantify that, on top of finding the dimensions and locations in the lower dimensional space. This is called **non-metric** or **ordinal multidimensional scaling**. The stress criterion looks similar to that for classical MDS, but differs in that it is essentially measuring how well the ordering of the original dissimilarities is preserved in the lower dimensional space.

Here is the basic algorithm for **non-metric MDS** (adapted from the presentation in Johnson & Wichern):

1. For observations, obtain the dissimilarities between distinct pairs of items. Order the dissimilarities in a strictly ascending order:

Here denotes the pair of items most dissimilar, i.e. the items with rank 1 in the dissimilarity ordering and consequently denotes the pair of items least dissimilar (i.e. most similar).

1. Using a trial configuration in dimensions, determine the inter-item distances and numbers , where the latter satisfy (1) above and minimize the stress measured using one of the two measures below.





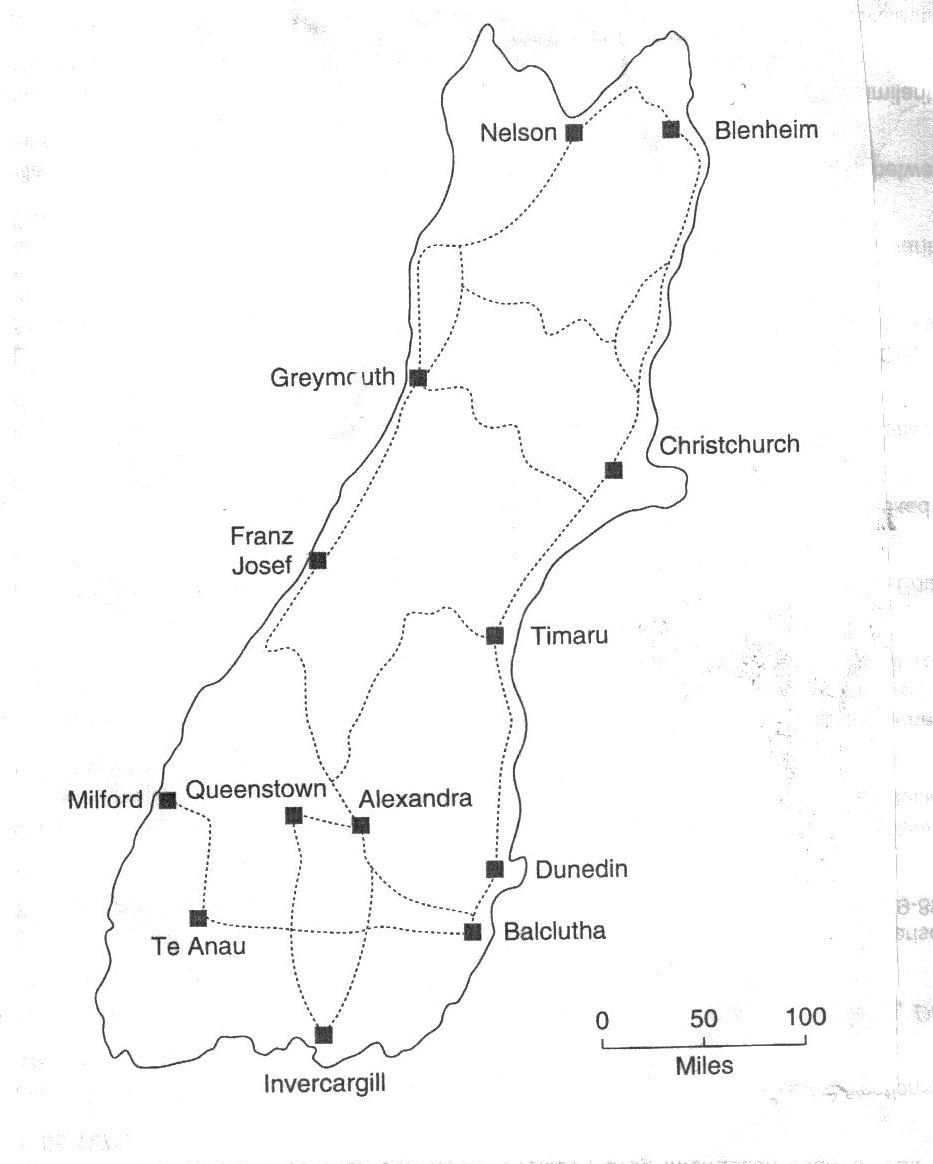
This is done by regressing on using ***monotonic*** regression, an example is shown above. The dashed vertical lines in the plot above are the differences in the numerator of the stress measures.

1. Using the move the points around to obtain an improved configuration as determined by the using one of the stress measures above. In this context, the stress is regarded as a function of the coordinates of the items in the lower dimensional space. A new improved configuration will have new , new , and smaller stress. The process is repeated until the best (minimum stress) representation is obtained.
2. We can try several choices for and choose the “best” number of dimensions.

Example 3.1: New Zealand Road Distances (distance between locations based on road distances)



Map of the South Island of New Zealand  

Metric MDS in R (uses cmdscale from MASS library)  
> library(ade4); data(zealand)  
> distNZ = zealand$road> nz.mds = cmdscale(distNZ,k=2,eig=T)> nz.mds$points

[,1] [,2]

Alexandra -123.42689 35.56878

Balclutha -141.56415 -174.25105

Blenheim 345.53108 -60.16752

Christchurch 145.08123 -58.43562

Dunedin -83.01900 -88.13695

Franz Josef 113.74536 238.89235

Greymouth 225.35514 73.19175

Invercargill -218.47777 -33.86827

Milford -330.60429 59.46636

Nelson 423.01479 -15.82089

Queenstown -138.35332 75.08549

Te Anau -258.64187 35.07239

Timaru 41.35968 -86.59681

$eig

[1] 6.700511e+05 1.281586e+05 2.423273e+04 1.500097e+04 6.346515e+03 3.860712e+03 1.371446e+03 1.455192e-11

[9] -6.650459e+02 -2.772176e+03 -8.323530e+03 -2.233212e+04 -7.772742e+04

$x

NULL

$ac

[1] 0

$GOF

[1] 0.9038571 0.9401519

> dist2 = as.matrix(dist(nz.mds$points)) # distance matrices are lower triangular

> dist2

Alexandra Balclutha Blenheim Christchurch Dunedin Franz Josef Greymouth Invercargill Milford Nelson Queenstown

Alexandra 0.00000 210.6023 478.63036 284.4880 130.1380 312.3958 350.8053 117.71225 208.55112 548.85281 42.24179

Balclutha 210.60228 0.0000 500.27675 309.1582 104.1306 485.6649 442.5581 160.07195 300.59940 586.38681 249.35722

Blenheim 478.63036 500.2767 0.00000 200.4573 429.4618 378.3668 179.5187 564.62167 686.63768 89.27681 502.43158

Christchurch 284.48802 309.1582 200.45734 0.0000 230.0258 298.9747 154.1741 364.38811 490.07916 281.18158 313.30980

Dunedin 130.13803 104.1306 429.46183 230.0258 0.0000 381.6600 348.0252 145.92521 288.24506 511.17492 172.34690

Franz Josef 312.39582 485.6649 378.36679 298.9747 381.6600 0.0000 199.7835 429.84946 479.20798 400.65748 300.64336

Greymouth 350.80535 442.5581 179.51867 154.1741 348.0252 199.7835 0.0000 456.56270 556.12882 216.77774 363.71339

Invercargill 117.71225 160.0719 564.62167 364.3881 145.9252 429.8495 456.5627 0.00000 145.88938 641.74637 135.24367

Milford 208.55112 300.5994 686.63768 490.0792 288.2451 479.2080 556.1288 145.88938 0.00000 757.37037 192.88440

Nelson 548.85281 586.3868 89.27681 281.1816 511.1749 400.6575 216.7777 641.74637 757.37037 0.00000 568.68104

Queenstown 42.24179 249.3572 502.43158 313.3098 172.3469 300.6434 363.7134 135.24367 192.88440 568.68104 0.00000

Te Anau 135.21589 239.8406 611.63355 414.4105 214.5319 424.5172 485.4958 79.78702 75.98458 683.55389 126.76901

Timaru 205.13178 202.8408 305.31746 107.4766 124.3882 333.4410 243.6939 265.13355 399.61437 388.16214 241.73939

Te Anau Timaru

Alexandra 135.21589 205.1318

Balclutha 239.84055 202.8408

Blenheim 611.63355 305.3175

Christchurch 414.41052 107.4766

Dunedin 214.53189 124.3882

Franz Josef 424.51716 333.4410

Greymouth 485.49582 243.6939

Invercargill 79.78702 265.1335

Milford 75.98458 399.6144

Nelson 683.55389 388.1621

Queenstown 126.76901 241.7394

Te Anau 0.00000 323.7350

Timaru 323.73496 0.0000

Notice how far off some of the distances in the metric MDS are! For example, the actual driving distance between Alexandria and Balclutha is 100 km, but MDS solution has the distance at 210.6023 km!!!

> stress2 = sum((distNZ – dist2)^2)/sum(distNZ^2)

> stress2 🡨 stress squared

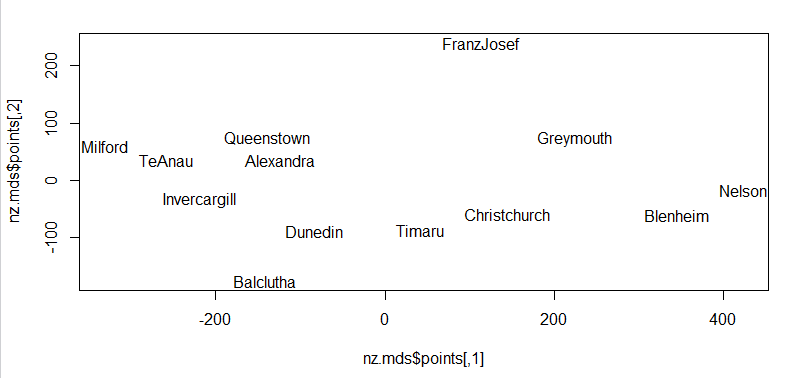
[1] 0.01180292

> sqrt(stress2) 🡨 stress

[1] 0.1086412

> plot(nz.mds$points,type=”n”)

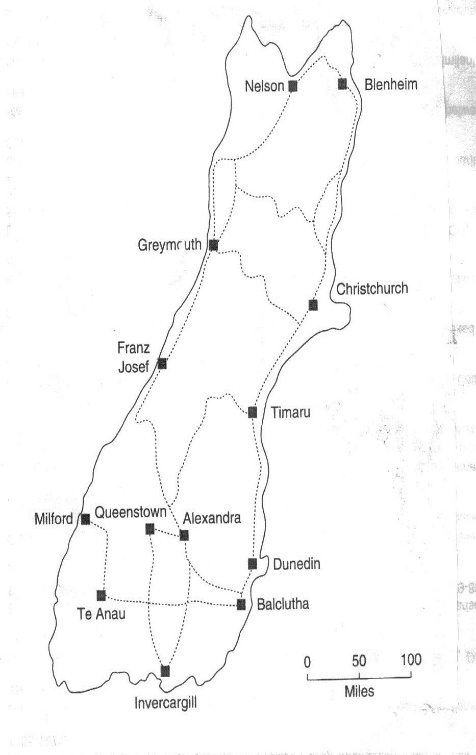
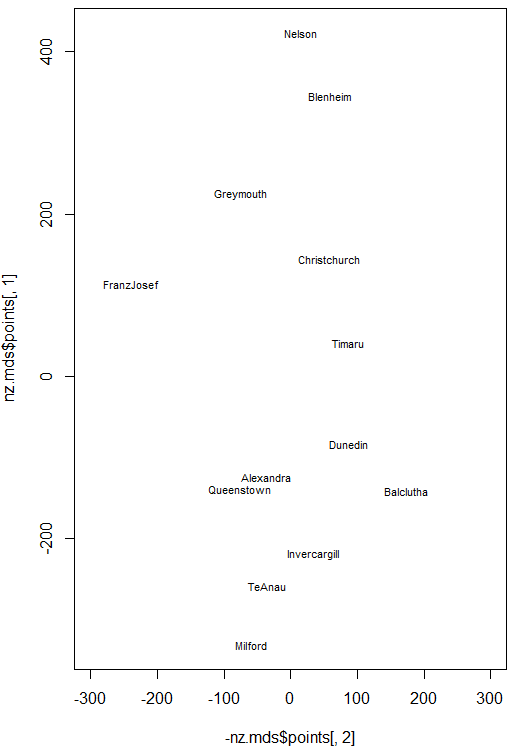
> text(nz.mds$points,row.names(nz.mds$points),cex=.70)



This orientation makes it hard to compare so change it. Any solution to the multidimensional scaling problem can be rotated orthogonally without changing the distances in the lower dimensional space.

> plot(-nz.mds$points[,2],nz.mds$points[,1],type=”n”,xlim=c(-300,300))

> text(-nz.mds$points[,2],nz.mds$points[,1],row.names(distNZ),cex=.7)



This is the same plot that is shown next to the map above.

Non-Metric MDS in R **(**uses sammon or isoMDS from MASS library, but there are several others!**)**> distNZ2 = as.dist(distNZ) 🡨 create a lower-triangular representation of

> nz.samm = sammon(distNZ2,k=2)

Initial stress : 0.02881

stress after 10 iters: 0.00523, magic = 0.500

stress after 20 iters: 0.00522, magic = 0.500

> nz.samm

$points

[,1] [,2]

Alexandra -104.06429 19.63688

Balclutha -140.32204 -88.32090

Blenheim 352.52698 -45.25842

Christchurch 146.10419 -62.49976

Dunedin -88.10541 -102.88833

Franz Josef 107.32850 175.42533

Greymouth 210.33442 94.97742

Invercargill -214.19209 -56.15620

Milford -322.32007 54.35357

Nelson 399.15703 18.38260

Queenstown -138.71795 62.76167

Te Anau -253.30046 21.28809

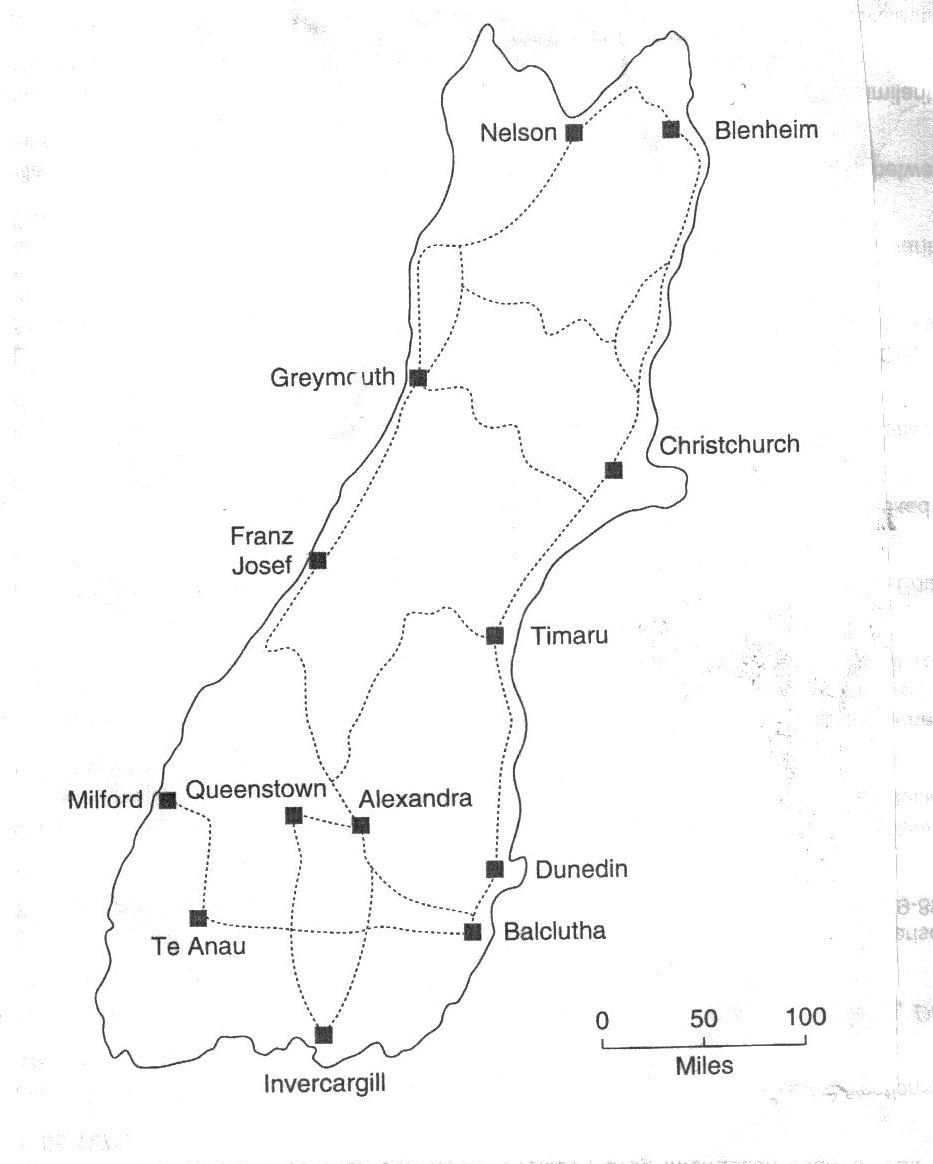
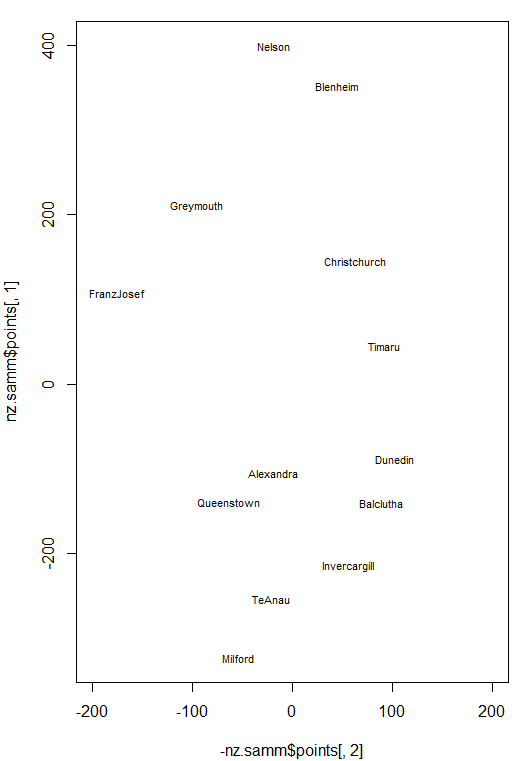
Timaru 45.57120 -91.70197

$stress

[1] 0.005221445

> plot(-nz.samm$points[,2],nz.samm$points[,1],type="n",xlim=c(-200,200))

> text(-nz.samm$points[,2],nz.samm$points[,1],row.names(nz.samm$points),cex=.7)

> distNZ = as.matrix(distNZ)🡨 forms the complete symmetric distance matrix

> nz.iso = isoMDS(distNZ) 🡨 isoMDS will take lower triangular or complete distance matrix

initial value 7.307991

final value 7.304465

converged

> nz.iso

$points

[,1] [,2]

Alexandra -123.41215 35.56509

Balclutha -141.56615 -174.21031

Blenheim 345.54166 -60.17193

Christchurch 145.08852 -58.43829

Dunedin -83.02413 -88.14665

Franz Josef 113.74493 238.87119

Greymouth 225.35707 73.20601

Invercargill -218.48612 -33.88337

Milford -330.61222 59.46502

Nelson 423.00812 -15.81684

Queenstown -138.35550 75.09309

Te Anau -258.65092 35.07015

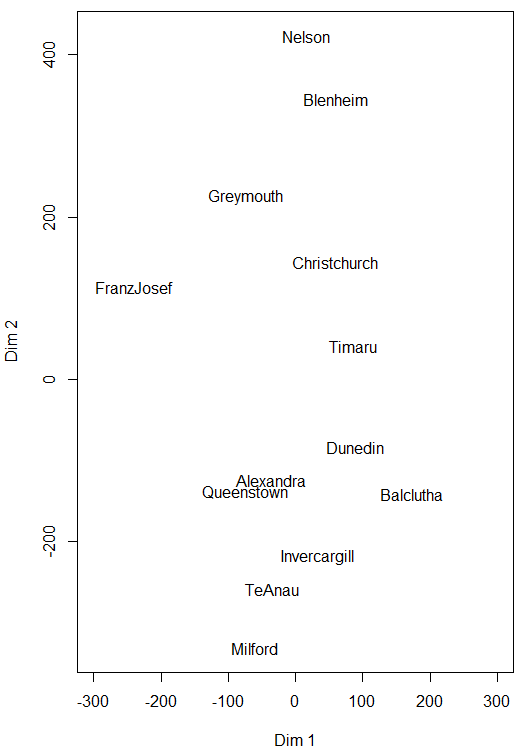
Timaru 41.36689 -86.60314

$stress

[1] 7.304465 🡨 stress expressed a percent (i.e. stress = .073)

> plot(-nz.iso$points[,2],nz.iso$points[,1],type="n",xlab="Dim 1",ylab="Dim 2",  
 xlim=c(-300,300))

> text(-nz.iso$points[,2],nz.iso$points[,1],row.names(nz.iso$points))



Shepard Plots – used to compare all original pair-wise distances to distances in lower-dimensional space

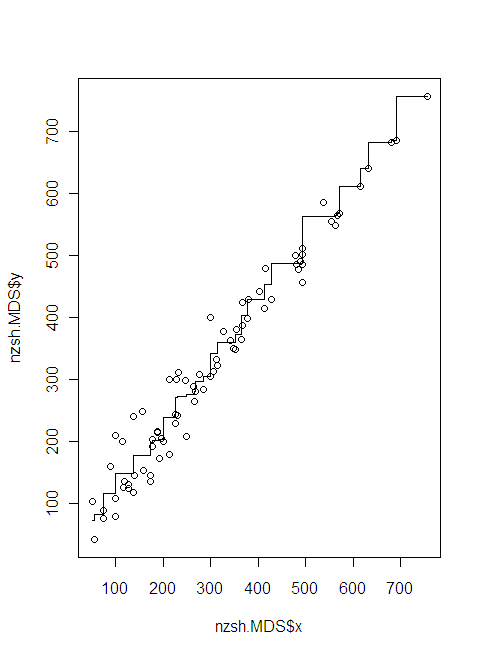
We use the Shepard function in the MASS library to construct this plot.

> distNZ2 = as.dist(distNZ) 🡨 again form the lower-triangular distances

> nzsh.MDS = Shepard(distNZ2,nz.mds$points) 🡨 results from classic MDS

> plot(nzsh.MDS)

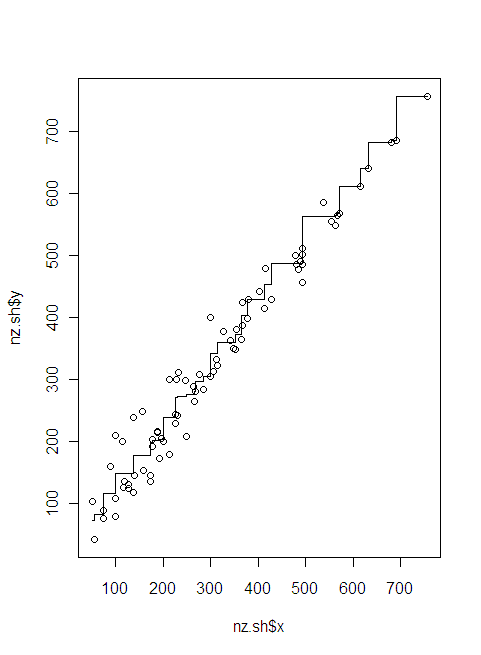
> lines(nzsh.MDS$x,nzsh.MDS$yf,type="S")



> nz.sh = Shepard(distNZ2,nz.iso$points) 🡨 results from non-metric MDS using isomap

> plot(nz.sh)

> lines(nz.sh$x,nz.sh$yf,type="S")



A good rule of thumb: stress < 0.05 provides an excellent representation in reduced dimensions, stress < 0.1 is great, < 0.2 is good/ok, and stress < 0.3 provides a poor representation. To reiterate: high stress is bad, low stress is good! True in life as well!

Example 3.2: Fatty Acid Content in Italian Olive Oils

> names(Olives)

[1] "Region.name" "Area.name" "Region" "Area" "palmitic" "palmitoleic"

[7] "strearic" "oleic" "linoleic" "eicosanoic" "linolenic" "eicosenoic"

> olive.mat = Olives[,5:11] 🡨 Eicosenoic acid level is NOT included!

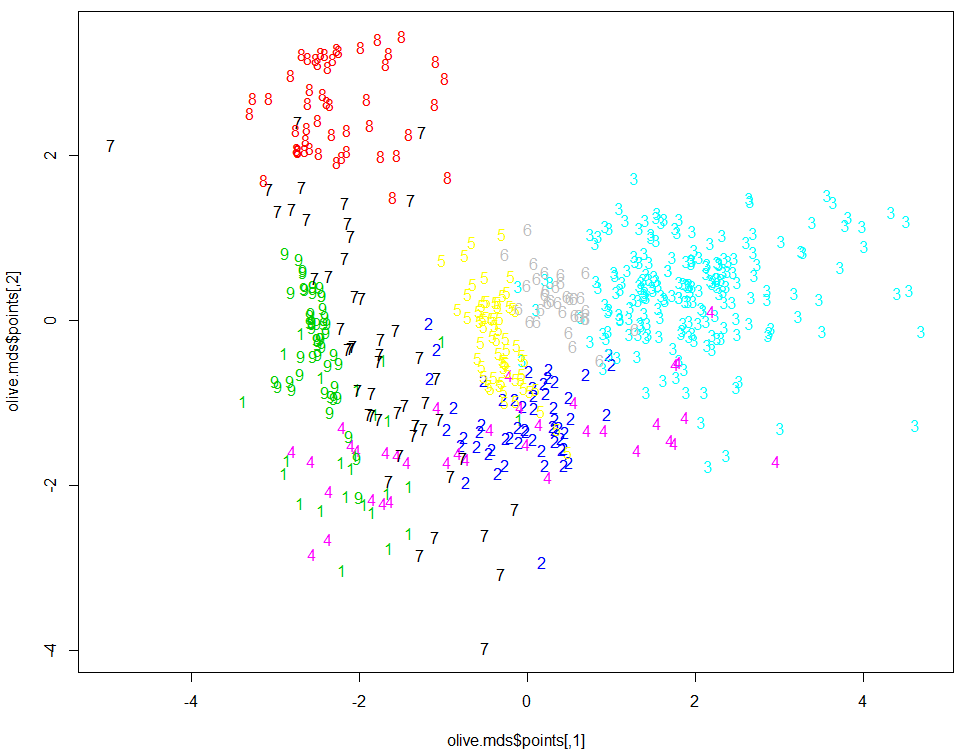
> olive.mat = scale(olive.mat)

> olive.dist = dist(olive.mat)

> olive.MDS = cmdscale(olive.dist,k=2,eig=T)

> plot(olive.MDS$points,type="n")

> text(olive.MDS$points,as.character(Olives$Area),col=as.numeric(Olives$Area)+2)



**Computing the stress measures (appropriate for metric MDS only!) - see pg. 72**

Kruskal-Shepard stress measure

Stress = function(dist,mdspts) {

dist2 = dist(mdspts)

s2 = sum((dist-dist2)^2)/sum(dist^2)

sqrt(s2)

}

> Stress(olive.dist,olive.mds$points)

[1] 0.2432502

Takane’s stress measure

Stress2 = function(dist,mdspts) {

dist2 = dist(mdspts)

s2 = sum((dist^2-dist2^2)^2)/sum(dist^4)

sqrt(s2)

}

> Stress2(olive.dist,olive.mds$points)

[1] 0.309317 🡨 NOT GOOD!

> olive.mds$GOF

[1] 0.7114404 0.7114404

Increasing the dimension of the solution will always decrease the stress. For olive oil data we consider the   
 solution vs. the two-dimensional solution above.

> olive.mds3 = cmdscale(olive.dist,k=3,eig=T)

> Stress(olive.dist,olive.mds3$points)

[1] 0.1449855

> Stress2(olive.dist,olive.mds3$points)

[1] 0.1747458

> olive.mds3$GOF

[1] 0.8469011 0.8469011 🡨 think of these like values in regression

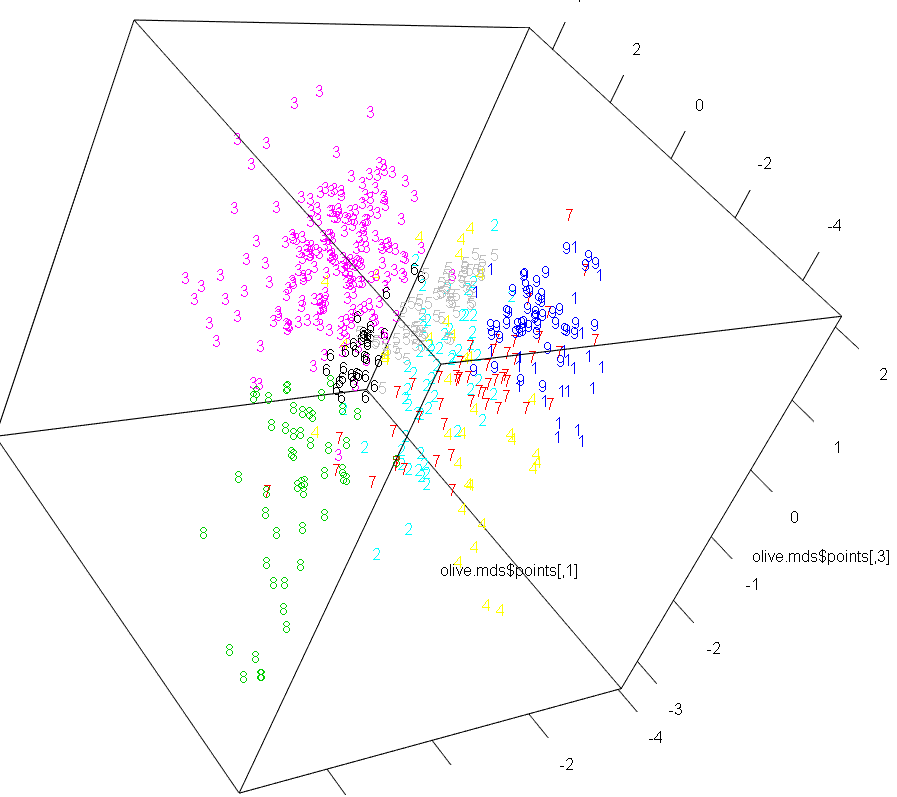
By using the plot3d and text3d functions in the rgl package we can view 3-D multidimensional scaling solutions. For the olive data color coding and labeling by Area or Region can be informative.

> library(rgl)

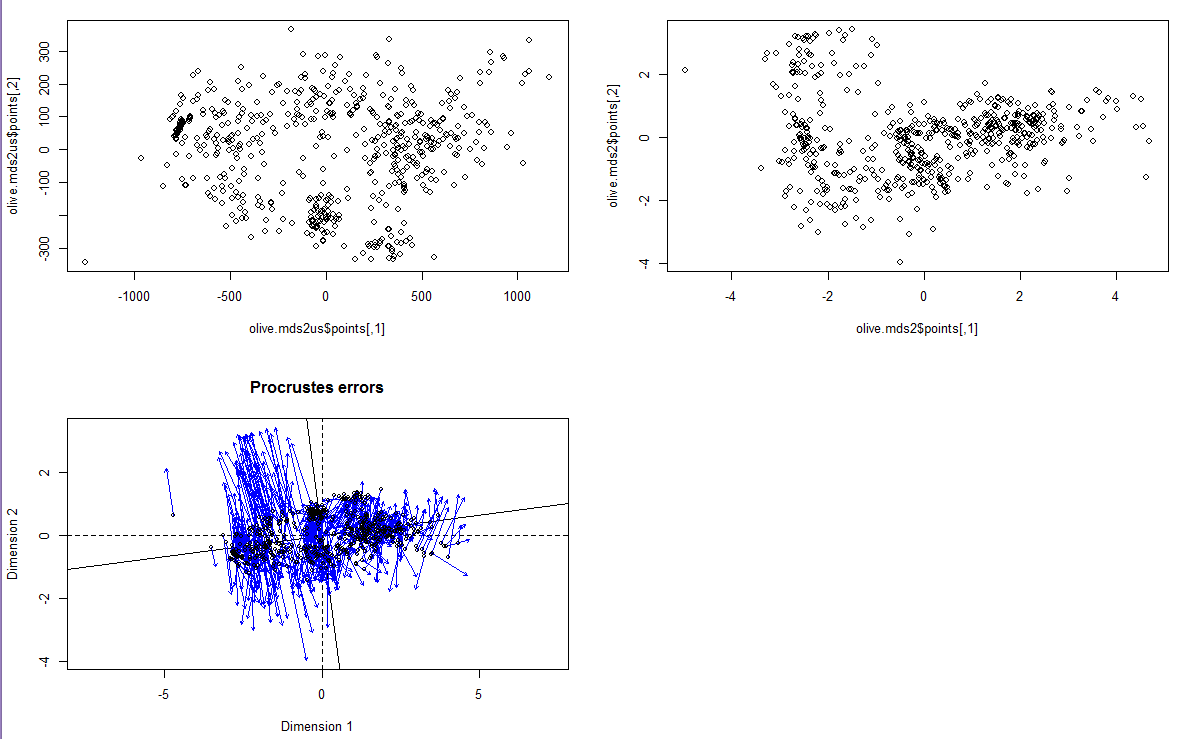
> olive.mds = cmdscale(olive.dist,k=3,eig=T)

> plot3d(olive.mds3$points,type="n")

> text3d(olive.mds3$points,texts=as.character(Olives$Area),col=as.numeric(Olives$Area)+3)



**Footnote: Importance of scaling numeric variables!**



Non-metric Multidimensional Scaling for Olive Data (Kruskal’s and Sammon Mapping)

We now consider non-metric multidimensional scaling methods for the olive oil data. First we use isoMDS in the MASS library to perform Kruskal’s non-metric MDS.

> olive.iso = isoMDS(olive.dist,k=2)

Error in isoMDS(olive.dist, k = 2) :

zero or negative distance between objects 450 and 451

Huh? Non-metric multidimensional scaling cannot deal with 0 distances, which happens because observations 450 and 451 have exactly the same values for the fatty acids.

> olive.mat[450:451,]

palmitic palmitoleic strearic oleic linoleic eicosanoic linolenic

450 -0.9593635 -0.9733313 -1.112137 1.64671 -1.559016 -0.7624599 -0.5945417

451 -0.9593635 -0.9733313 -1.112137 1.64671 -1.559016 -0.7624599 -0.5945417

Since any lower dimensional representation will/should put these points in the same place we can simply delete one of them from the analysis.

> olive.dist2 = dist(olive.mat[-451,])

> olive.iso = isoMDS(olive.dist2,k=2)

initial value 17.534925

iter 5 value 13.879367

iter 10 value 13.298940

final value 13.172985

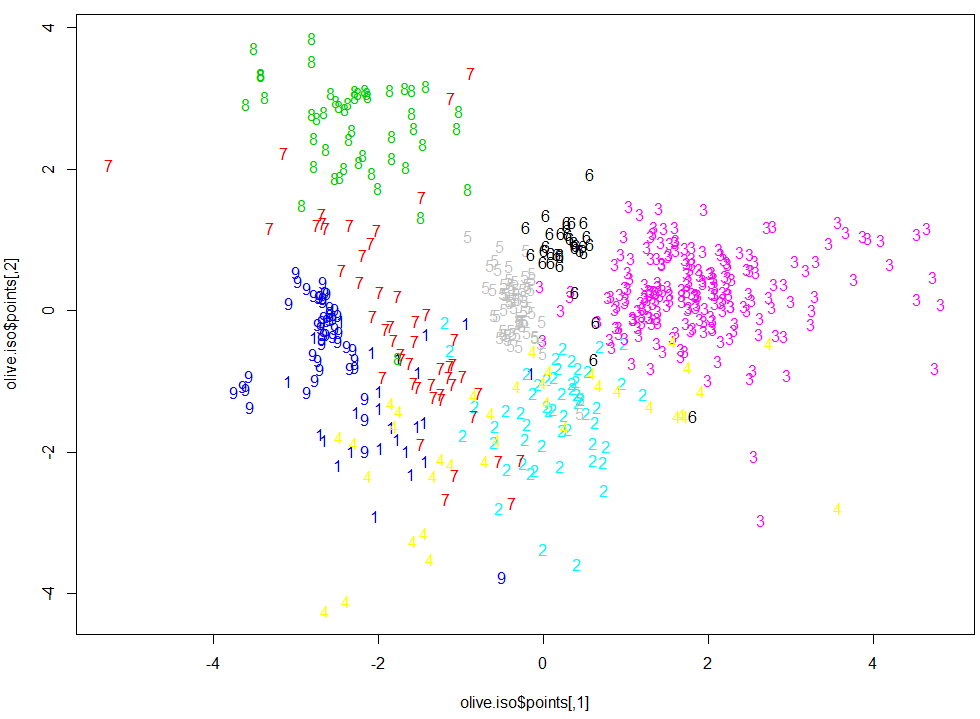
converged

> olive.iso$stress 🡨stress measure for non-metric MDS expressed as a %.

[1] 13.17298

> plot(olive.iso$points,type="n")

> text(olive.iso$points,labels=as.character(Olives$Area[-451),col=as.numeric(Olives$Area[-451)+3)



The plot below shows how the points from the metric MDS would need to be moved to obtain the non-metric MDS results from isoMDS.

> library(vegan)

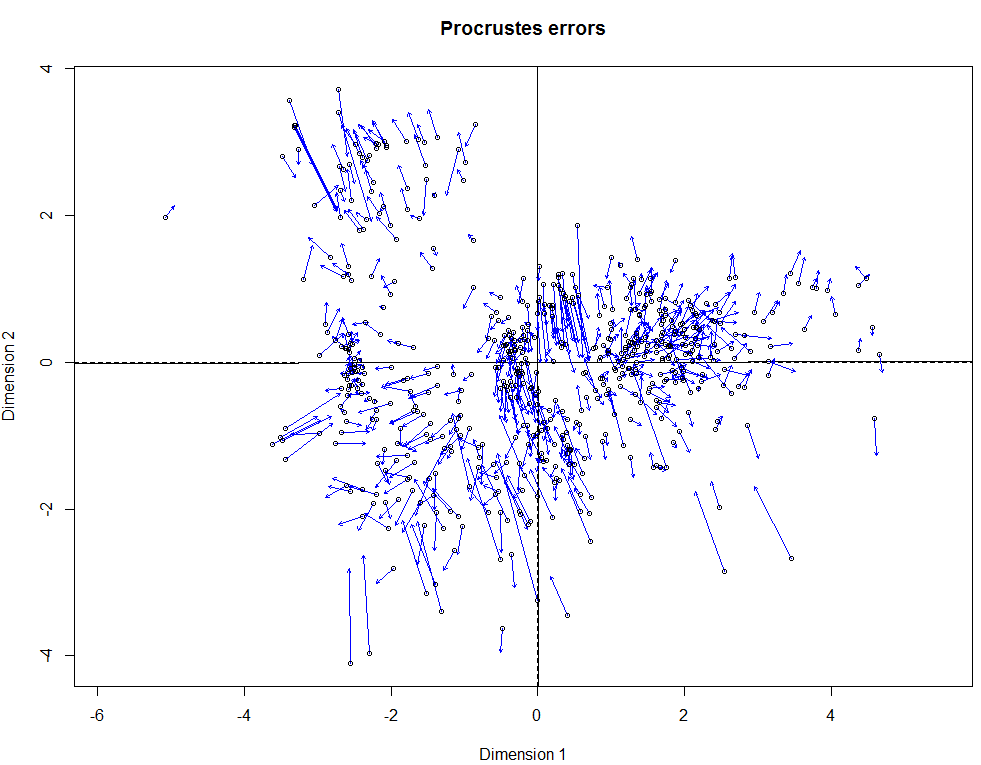
> olive.mds = cmdscale(olive.dist,k=2,eig=T)

> olive.iso = isoMDS(olive.dist,k=2)

> temp = procrustes(olive.mds$points,olive.iso$points)

> plot(temp)

To see how the points in the non-metric representation would need to be moved in order to match the coordinates of the metric MDS reverse the order of the arguments in the procrustes function.



> olive.iso3 = isoMDS(olive.dist2,k=3)

initial value 10.680385

iter 5 value 8.995923

iter 10 value 8.500259

final value 8.422152

converged

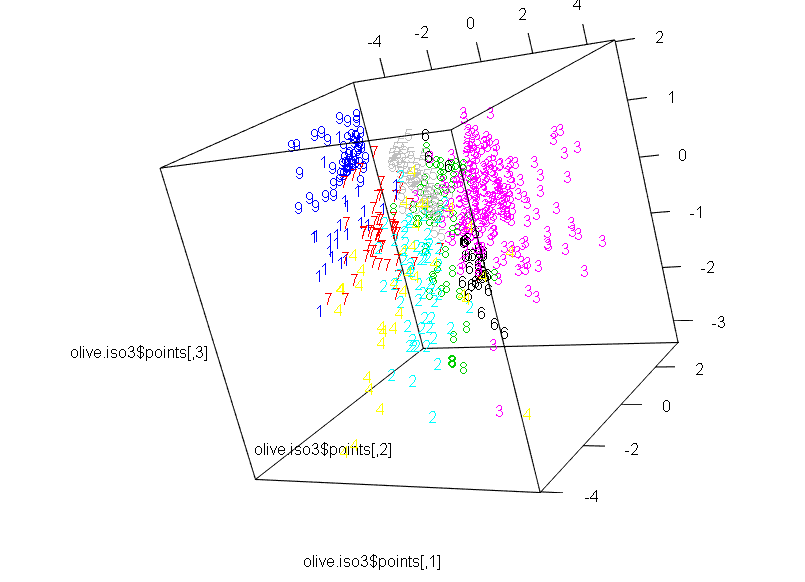
> olive.iso3$stress 🡨 Stress = 8.42% which is pretty good!

[1] 8.422152

> plot3d(olive.iso3$points,type="n")

> text3d(olive.iso3$points,texts=as.character(Olives$Area[-451]),col=as.numeric(Olives$Area[-451])+3)

Note: Because we removed observation 451 from the data matrix, we need to move it from the Area labels as well.



Sammon mapping is **actually another metric MDS** method that uses a different measure of stress (even though in the R documentation it is called a non-metric method). The Sammon stress gives more weight to preserving smaller dissimilarities in the optimization process.

Sammon stress



> olive.samm = sammon(olive.dist,k=2)

Error in sammon(olive.dist, k = 2) :

zero or negative distance between objects 450 and 451 🡨 we again to remove obs. 451

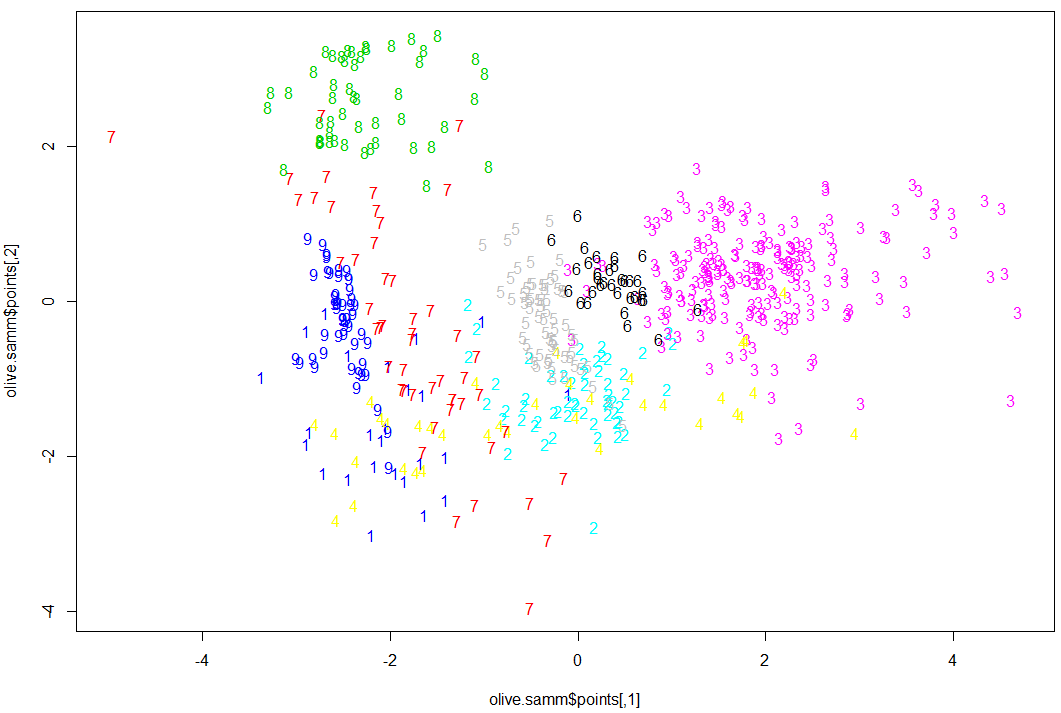
> olive.samm = sammon(olive.dist2,k=2)

Initial stress : 0.07724

stress after 0 iters: 0.07724

> plot(olive.samm$points,type="n")

> text(olive.samm$points,labels=as.character(Olives$Area[-451]),col=as.numeric(Olives$Area[-451])+3)



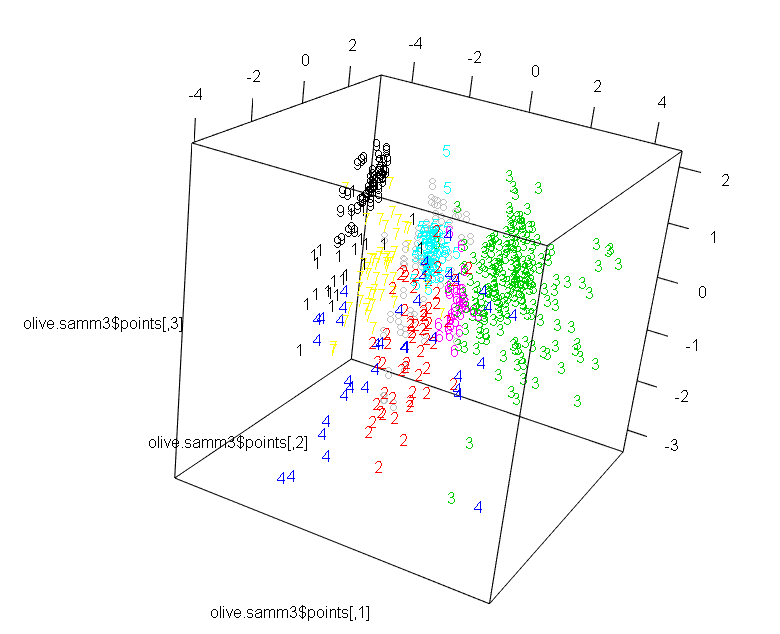
> olive.samm3 = sammon(olive.dist2,k=3)

Initial stress : 0.02933

stress after 1 iters: 0.02918

> plot3d(olive.samm3$points,type="n")

> text3d(olive.samm3$points,texts=as.character(Olives$Area[-451]),col=as.numeric(Olives$Area[-451]))



Example 3.3: Edibility of Mushrooms and their Physical Characteristics

We have used these data previously in Section 2 – Measuring Distances. We will now use multi-dimensional scaling to visualize the structure in these data in a lower dimensional space. As all variables are nominal we will need to think about what metric(s) we will use to measure distances between mushrooms in these data.

The mushroom attributes are defined below:

Variable Levels \*  
     Poisonous edible=e,poisonous=p

X1 = cap-shape: bell=b,conical=c,convex=x,flat=f,  
 knobbed=k,sunken=s  
 X2 = cap-surface: fibrous=f,grooves=g,scaly=y,smooth=s  
 X3 = cap-color: brown=n,buff=b,cinnamon=c,gray=g,green=r,  
 pink=p,purple=u,red=e,white=w,yellow=y  
 X4 = bruises?: true=t,false=f  
 X5 = odor: almond=a,anise=l,creosote=c,fishy=y,foul=f,  
 musty=m,none=n,pungent=p,spicy=s  
 X6 = gill-attachment: attached=a, free=f  
 X7 = gill-spacing: close=c,crowded=w  
 X8 = gill-size: broad=b,narrow=n  
 X9 = gill-color: black=k,brown=n,buff=b,chocolate=h,gray=g,  
 green=r,orange=o,pink=p,purple=u,red=e,  
 white=w,yellow=y  
 X10 = stalk-shape: enlarging=e,tapering=t  
 X11 = stalk-root: bulbous=b,club=c,cup=u,equal=e,  
 rhizomorphs=z,rooted=r,missing=? 🡨 has lots of missing values!  
 X12 = stalk-surface-above-ring: ibrous=f,scaly=y,silky=k,smooth=s  
 X13 = stalk-surface-below-ring: ibrous=f,scaly=y,silky=k,smooth=s 🡨 has lots of missing values!  
 X14 = stalk-color-above-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o,  
 pink=p,red=e,white=w,yellow=y  
 X15 = stalk-color-below-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o,  
 pink=p,red=e,white=w,yellow=y  
 X16 = veil-type: partial=p,universal=u 🡨 All partial (no universal)!  
 X17 = veil-color: brown=n,orange=o,white=w,yellow=y  
 X18 = ring-number: none=n,one=o,two=t  
 X19 = ring-type: cobwebby=c,evanescent=e,flaring=f,large=l,  
 none=n,pendant=p,sheathing=s,zone=z 🡨 some levels not represented  
 X20 = spore-print-color: black=k,brown=n,buff=b,chocolate=h,green=r,  
 orange=o,purple=u,white=w,yellow=y  
 X21 = population: abundant=a,clustered=c,numerous=n,  
 scattered=s,several=v,solitary=y  
 X22 = habitat: grasses=g,leaves=l,meadows=m,paths=p,  
 urban=u,waste=w,woods=d

The dataframe Mushroom.train in the mult.Rdata folder I sent you and contains these data for 4072 mushrooms. We will now conduct a multi-dimensional scaling of these mushrooms on the basis of similarity/dissimilarity measures for nominal variables, many of which have more than two levels, i.e. are not binary/dichotomous.

Again we will be using the following libraries in R thus you will need to install and load the following libraries: cba, proxy, rgl, vegan, MASS 🡨 included with R just load it by checking the box next to it in the list.

Multidimensional Scaling of Mushroom Data

> names(Mushrooms.train)

[1] "Poisonous" "x1" "x2" "x3" "x4" "x5"

[7] "x6" "x7" "x8" "x9" "x10" "x11"

[13] "x12" "x13" "x14" "x15" "x16" "x17"

[19] "x18" "x19" "x20" "x21" "x22"  
  
> summary(Mushrooms.train)

Poisonous x1 x2 x3 x4 x5 x6 x7

e:2107 b: 259 f:1117 n :1120 f:2377 n :1772 a: 122 c:3392

p:1955 c: 2 g: 2 g : 876 t:1685 f :1064 f:3940 w: 670

f:1574 s:1329 e : 747 y : 287

k: 417 y:1614 w : 556 s : 281

s: 10 y : 556 l : 207

x:1800 b : 88 a : 200

(Other): 119 (Other): 251

x8 x9 x10 x11 x12 x13 x14 x15

b:2824 b :838 e:1826 ?:1245 f: 274 f: 297 w :2269 w :2220

n:1238 p :730 t:2236 b:1871 k:1189 k:1136 p : 915 p : 905

w :594 c: 288 s:2583 s:2484 g : 265 g : 273

n :536 e: 562 y: 16 y: 145 n : 221 n : 255

g :394 r: 96 b : 210 b : 215

h :379 o : 113 o : 113

(Other):591 (Other): 69 (Other): 81

x16 x17 x18 x19 x20 x21 x22

p:4062 n: 61 n: 17 e:1370 w :1193 a: 193 d:1533

o: 52 o:3712 f: 23 n : 971 c: 189 g:1104

w:3944 t: 333 l: 646 k : 924 n: 204 l: 411

y: 5 n: 17 h : 817 s: 650 m: 148

p:2006 r : 47 v:1998 p: 574

y : 33 y: 828 u: 190

(Other): 77 w: 102

As you can see there are several nominal variables with multiple levels and also that variable 11 has missing values denoted by “?”. Also Poisonous is an indicator of the mushrooms edibility and will not be used in the multidimensional scaling of these data, rather we will use this information in plotting these data.

> mush.reduced = Mushrooms.train[,-c(1,12)]

> names(mush.reduced)

[1] "x1" "x2" "x3" "x4" "x5" "x6" "x7" "x8" "x9" "x10" "x12"

[12] "x13" "x14" "x15" "x16" "x17" "x18" "x19" "x20" "x21" "x22"

> mush.dummy = as.data.frame(as.dummy(mush.reduced))

> dim(mush.dummy)

[1] 4062 112

> head(mush.dummy)

x1 b x1 c x1 f x1 k x1 s x1 x x2 f x2 g x2 s x2 y x3 b x3 c x3 e x3 g

7251 FALSE FALSE FALSE TRUE FALSE FALSE FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE

724 FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE

2997 FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE FALSE TRUE FALSE FALSE FALSE FALSE

5218 FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE

6242 FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE FALSE TRUE FALSE FALSE TRUE FALSE

1310 FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE FALSE FALSE TRUE

x3 n x3 p x3 r x3 u x3 w x3 y x4 f x4 t x5 a x5 c x5 f x5 l x5 m x5 n

7251 TRUE FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE

724 TRUE FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE FALSE TRUE

2997 TRUE FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE TRUE

5218 FALSE FALSE FALSE FALSE TRUE FALSE FALSE TRUE FALSE FALSE TRUE FALSE FALSE FALSE

6242 FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE FALSE TRUE FALSE FALSE FALSE

1310 FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE FALSE TRUE

x5 p x5 s x5 y x6 a x6 f x7 c x7 w x8 b x8 n x9 b x9 e x9 g x9 h x9 k x9 n

7251 FALSE FALSE TRUE FALSE TRUE TRUE FALSE FALSE TRUE TRUE FALSE FALSE FALSE FALSE FALSE

724 FALSE FALSE FALSE FALSE TRUE FALSE TRUE TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE

2997 FALSE FALSE FALSE FALSE TRUE TRUE FALSE TRUE FALSE FALSE FALSE FALSE FALSE FALSE TRUE

5218 FALSE FALSE FALSE FALSE TRUE TRUE FALSE TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE

6242 FALSE FALSE FALSE FALSE TRUE TRUE FALSE FALSE TRUE TRUE FALSE FALSE FALSE FALSE FALSE

1310 FALSE FALSE FALSE FALSE TRUE FALSE TRUE TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE

x9 o x9 p x9 r x9 u x9 w x9 y x10 e x10 t x12 f x12 k x12 s x12 y x13 f x13 k

7251 FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE TRUE FALSE FALSE FALSE

724 FALSE TRUE FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE TRUE FALSE FALSE FALSE

2997 FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE TRUE FALSE FALSE FALSE

5218 FALSE FALSE FALSE FALSE TRUE FALSE FALSE TRUE FALSE FALSE TRUE FALSE FALSE FALSE

6242 FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE TRUE FALSE FALSE FALSE

1310 FALSE TRUE FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE TRUE FALSE FALSE FALSE

x13 s x13 y x14 b x14 c x14 e x14 g x14 n x14 o x14 p x14 w x14 y x15 b x15 c x15 e

7251 TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE

724 TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE FALSE FALSE

2997 TRUE FALSE FALSE FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE

5218 TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE FALSE FALSE

6242 TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE

1310 TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE FALSE FALSE

x15 g x15 n x15 o x15 p x15 w x15 y x16 p x17 n x17 o x17 w x17 y x18 n x18 o x18 t

7251 FALSE FALSE FALSE TRUE FALSE FALSE TRUE FALSE FALSE TRUE FALSE FALSE TRUE FALSE

724 FALSE FALSE FALSE FALSE TRUE FALSE TRUE FALSE FALSE TRUE FALSE FALSE TRUE FALSE

2997 TRUE FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE TRUE FALSE FALSE TRUE FALSE

5218 FALSE FALSE FALSE FALSE TRUE FALSE TRUE FALSE FALSE TRUE FALSE FALSE TRUE FALSE

6242 FALSE FALSE FALSE TRUE FALSE FALSE TRUE FALSE FALSE TRUE FALSE FALSE TRUE FALSE

1310 FALSE FALSE FALSE FALSE TRUE FALSE TRUE FALSE FALSE TRUE FALSE FALSE TRUE FALSE

x19 e x19 f x19 l x19 n x19 p x20 b x20 h x20 k x20 n x20 o x20 r x20 u x20 w x20 y

7251 TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE

724 TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE

2997 FALSE FALSE FALSE FALSE TRUE FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE FALSE

5218 FALSE FALSE FALSE FALSE TRUE FALSE TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE

6242 TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE

1310 TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE

x21 a x21 c x21 n x21 s x21 v x21 y x22 d x22 g x22 l x22 m x22 p x22 u x22 w

7251 FALSE FALSE FALSE FALSE TRUE FALSE FALSE FALSE TRUE FALSE FALSE FALSE FALSE

724 TRUE FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE

2997 FALSE FALSE FALSE FALSE FALSE TRUE TRUE FALSE FALSE FALSE FALSE FALSE FALSE

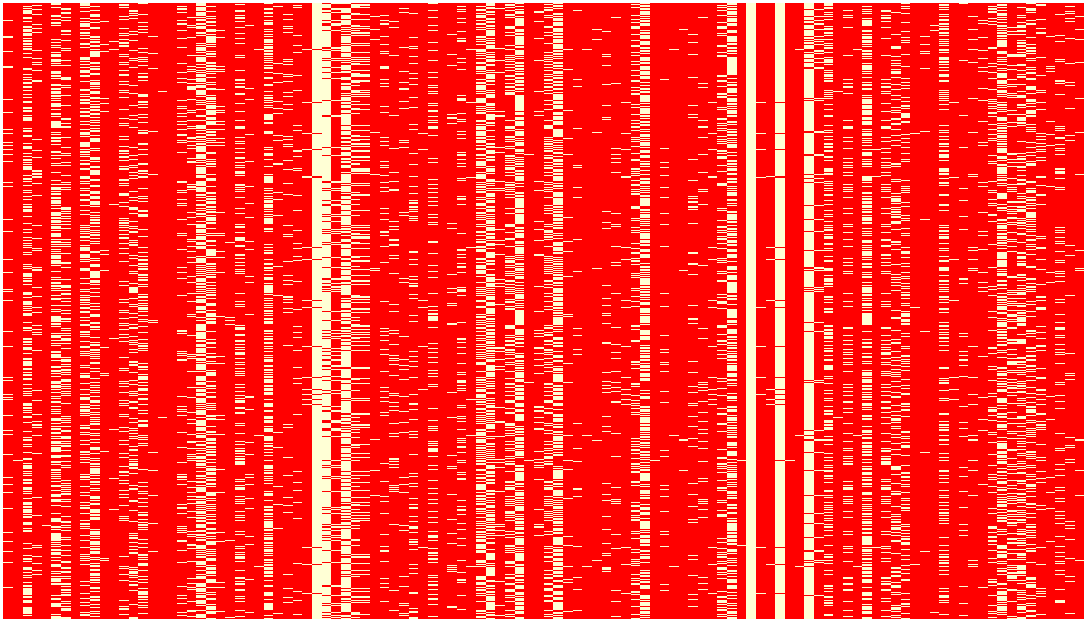
5218 FALSE FALSE FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE

6242 FALSE FALSE FALSE FALSE TRUE FALSE TRUE FALSE FALSE FALSE FALSE FALSE FALSE

1310 FALSE FALSE FALSE TRUE FALSE FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE

We can see that the as.dummy command from the cba library has created a dummy variable (T/F) for level of each of the nominal variables in the mushroom data set. Because of this there will be a lot of (False/False or 0/0) matches thus when measuring the similarity between mushrooms we will want to give more weight to (True/True or 1/1) matches, potentially ignoring the 0-0 matches entirely.

> implot(as.matrix(mush.dummy))



Shaded = False, White = True

We will now compute a distance matrix containing the pairwise dissimilarities between the 4062 mushrooms in the data set. The dist function in the proxy library contain numerous metrics for computing “distances” between observations in a data set. To obtain information about the available metrics run the commands below in R, making sure that the proxy library has been installed and loaded into R.

> summary(pr\_DB,"long")

\* Similarity measures:

Jaccard/binary/Reyssac/Roux (binary) = a / (a + b + c)

Kulczynski1 (binary) = a / (b + c)

Kulczynski2 (binary) = [a / (a + b) + a / (a + c)] / 2

Mountford (binary) = 2a / (ab + ac + 2bc)

Fager/McGowan (binary) = a / sqrt((a + b)(a + c)) - sqrt(a + c) / 2

Russel/Rao (binary) = a / n

simple matching/Sokal/Michener (binary) = (a + d) / n

Hamman (binary) = ([a + d] - [b + c]) / n

Faith (binary) = (a + d/2) / n

Tanimoto/Rogers (binary) = (a + d) / (a + 2b + 2c + d)

Dice/Czekanowski/Sorensen (binary) = 2a / (2a + b + c)

Phi (binary) = (ad - bc) / sqrt[(a + b)(c + d)(a + c)(b + d)]

Stiles (binary) = log(n(|ad-bc| - 0.5n)^2 / [(a + b)(c + d)(a + c)(b + d)])

Michael (binary) = 4(ad - bc) / [(a + d)^2 + (b + c)^2]

Mozley/Margalef (binary) = an / (a + b)(a + c)

Yule (binary) = (ad - bc) / (ad + bc)

Yule2 (binary) = (sqrt(ad) - sqrt(bc)) / (sqrt(ad) + sqrt(bc))

Ochiai (binary) = a / sqrt[(a + b)(a + c)]

Simpson (binary) = a / min{(a + b), (a + c)}

Braun-Blanquet (binary) = a / max{(a + b), (a + c)}

cosine/angular (metric) = xy / sqrt(xx \* yy)

eJaccard/extended\_Jaccard (metric) = xy / (xx + yy - xy)

correlation (metric) = xy / sqrt(xx \* yy) for centered x,y

Chi-squared (nominal) = sum\_ij (o\_i - e\_i)^2 / e\_i

Phi-squared (nominal) = [sum\_ij (o\_i - e\_i)^2 / e\_i] / n

Tschuprow (nominal) = sqrt{[sum\_ij (o\_i - e\_i)^2 / e\_i] / n / sqrt((p - 1)(q - 1))}

Cramer (nominal) = sqrt{[Chi / n)] / min[(p - 1), (q - 1)]}

Pearson/contingency (nominal) = sqrt{Chi / (n + Chi)}

Gower (other) = Sum\_k (s\_ijk \* w\_k) / Sum\_k (d\_ijk \* w\_k)

\* Distance measures:

Euclidean/L2 (metric) = sqrt(sum\_i (x\_i - y\_i)^2))

Mahalanobis (metric) = sqrt((x - y) Sigma^(-1) (x - y))

Bhjattacharyya (metric) = sqrt(sum\_i (sqrt(x\_i) - sqrt(y\_i))^2))

Manhattan/City-Block/L1/taxi (metric) = sum\_i |x\_i - y\_i|

supremum/max/maximum/Tschebyscheff/Chebyshev (metric) = max\_i |x\_i - y\_i|

Minkowski/Lp (metric) = (sum\_i (x\_i - y\_i)^p)^(1/p)

Canberra (metric) = sum\_i |x\_i - y\_i| / |x\_i + y\_i|

Wave/Hedges (metric) = sum\_i (1 - min(x\_i, y\_i) / max(x\_i, y\_i))

divergence (metric) = sum\_i (x\_i - y\_i)^2 / (x\_i + y\_i)^2

Kullback/Leibler (metric) = sum\_i [x\_i \* log((x\_i / sum\_j x\_j) / (y\_i / sum\_j y\_j)) / sum\_j x\_j)]

Bray/Curtis (metric) = sum\_i |x\_i - y\_i| / sum\_i (x\_i + y\_i)

Soergel (metric) = sum\_i |x\_i - y\_i| / sum\_i max{x\_i, y\_i}

Levenshtein (other) = Number of insertions, edits, and deletions between to strings

Podani/discordance (metric) = 1 - 2 \* (a - b + c - d) / (n \* (n - 1))

Chord (metric) = sqrt(2 \* (1 - xy / sqrt(xx \* yy)))

Geodesic (metric) = arccos(xy / sqrt(xx \* yy))

Whittaker (metric) = sum\_i |x\_i / sum\_i x - y\_i / sum\_i y| / 2

Hellinger (metric) = sqrt(sum\_i (sqrt(x\_i / sum\_i x) - sqrt(y\_i / sum\_i y)) ^ 2)

fJaccard/fuzzy\_Jaccard (metric) = sum\_i (min{x\_i, y\_i} / max{x\_i, y\_i})

> summary(pr\_DB,"short")

\* Similarity measures:

Braun-Blanquet, Chi-squared, correlation, cosine, Cramer, Dice, eJaccard, Fager, Faith, Gower,

Hamman, Jaccard, Kulczynski1, Kulczynski2, Michael, Mountford, Mozley, Ochiai, Pearson, Phi,

Phi-squared, Russel, simple matching, Simpson, Stiles, Tanimoto, Tschuprow, Yule, Yule2

\* Distance measures:

Bhjattacharyya, Bray, Canberra, Chord, divergence, Euclidean, fJaccard, Geodesic, Hellinger,

Kullback, Levenshtein, Mahalanobis, Manhattan, Minkowski, Podani, Soergel, supremum, Wave,

Whittaker

We will use Jaccard as our metric as this metric ignores (False/False or 0-0) matches in both the numerator and the denominator, i.e.

where,

a = # of variables where observations i and j are a 1-1 match.

b = # of variables where observation i is 1, but j is 0

c = # of variables where observation i is 0, but j is 1.

> mush.dist = dist(mush.dummy,method="Jaccard")

> summary(as.numeric(mush.dist))

Min. 1st Qu. Median Mean 3rd Qu. Max.

0.09091 0.60000 0.68750 0.66210 0.76470 0.92310

> mush.mds = cmdscale(mush.dist,k=3)

The short function below will construct a scatterplot matrix where the color and label of the points is determined by a grouping variable with a single character denoting the groups (e.g. a number or character (“e” vs. “p” here)).

> pairs.grps2 = function (x, groups, cex = 0.7)

{

pairs(x, panel = function(x, y, ...) {

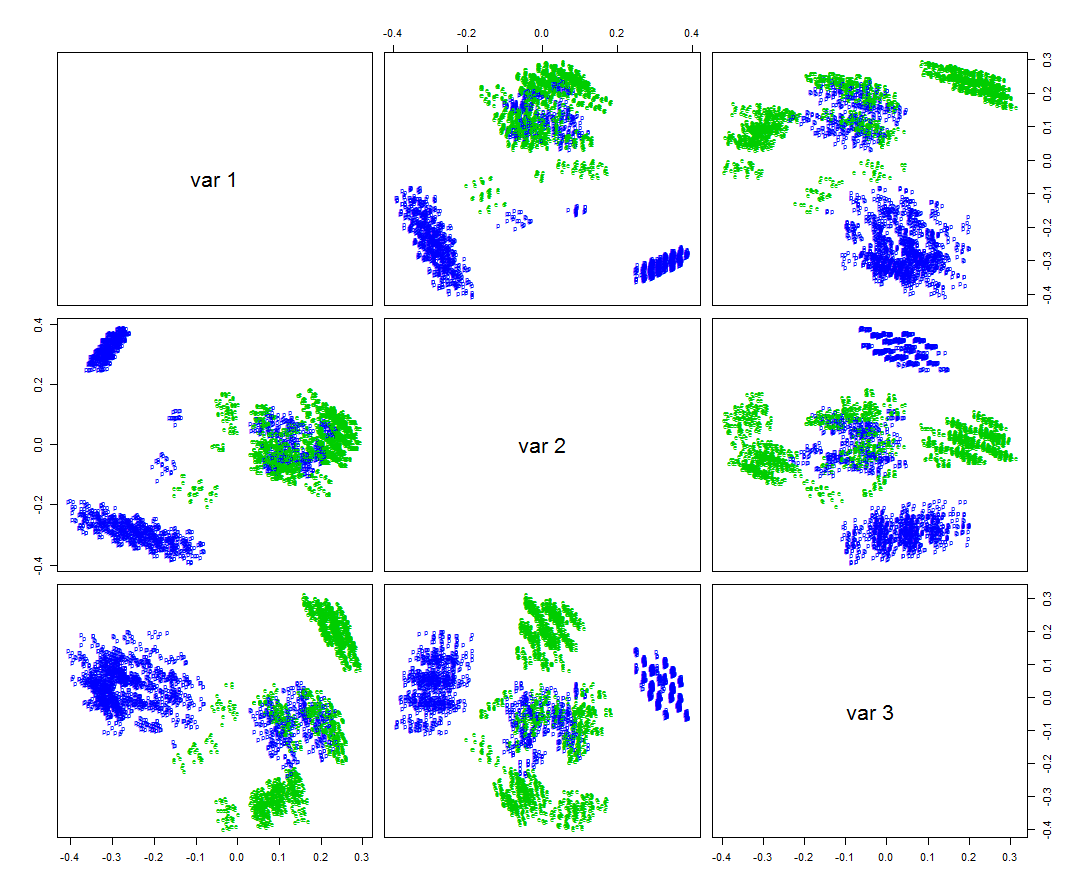
text(x, y, as.character(groups), cex = cex, col = as.numeric(groups) +

2)

})

}

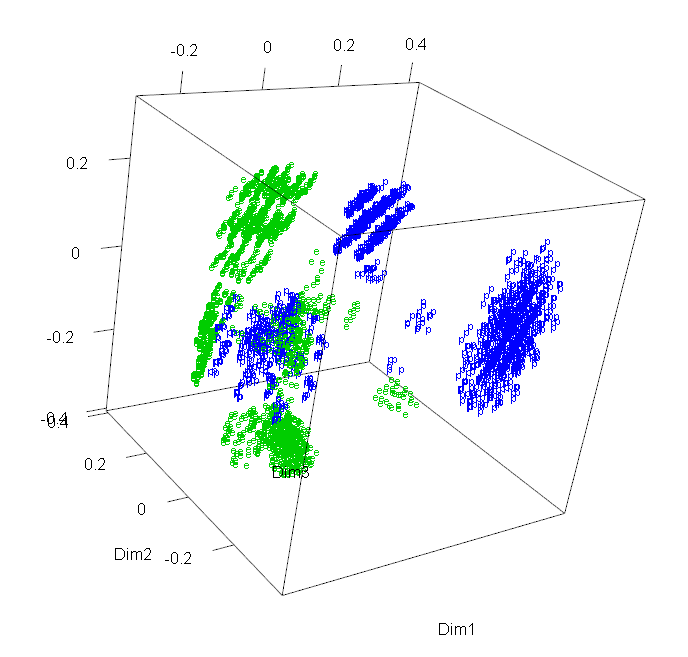
> pairs.grps2(mush.mds,groups=Mushrooms$Poisonous,cex=.7)



We can see the following: (1) the metric multidimensional scaling forms distinct groups of cluster of mushrooms and (2) the mushrooms cluster do convey information about the edibility of the mushrooms, although certainly not perfectly! Perhaps a 3-D view show this better. We again create dynamic 3-D plots as shown above in the olive oil example using functions in the rgl library.

> plot3d(mush.mds,type="n")

> text3d(mush.mds,text=as.character(Mushrooms.train$Poisonous),col=as.numeric(Mushrooms.train$Poisonous)+2)



As the distances/dissimilarities are not true Euclidean distances that need to be matched in the lower dimensional representation, we probably should be using non-metric multidimensional scaling methods for these data. We will use the isoMDS function in the MASS library to do this.

> mush.iso1 = isoMDS(mush.dist,k=1)

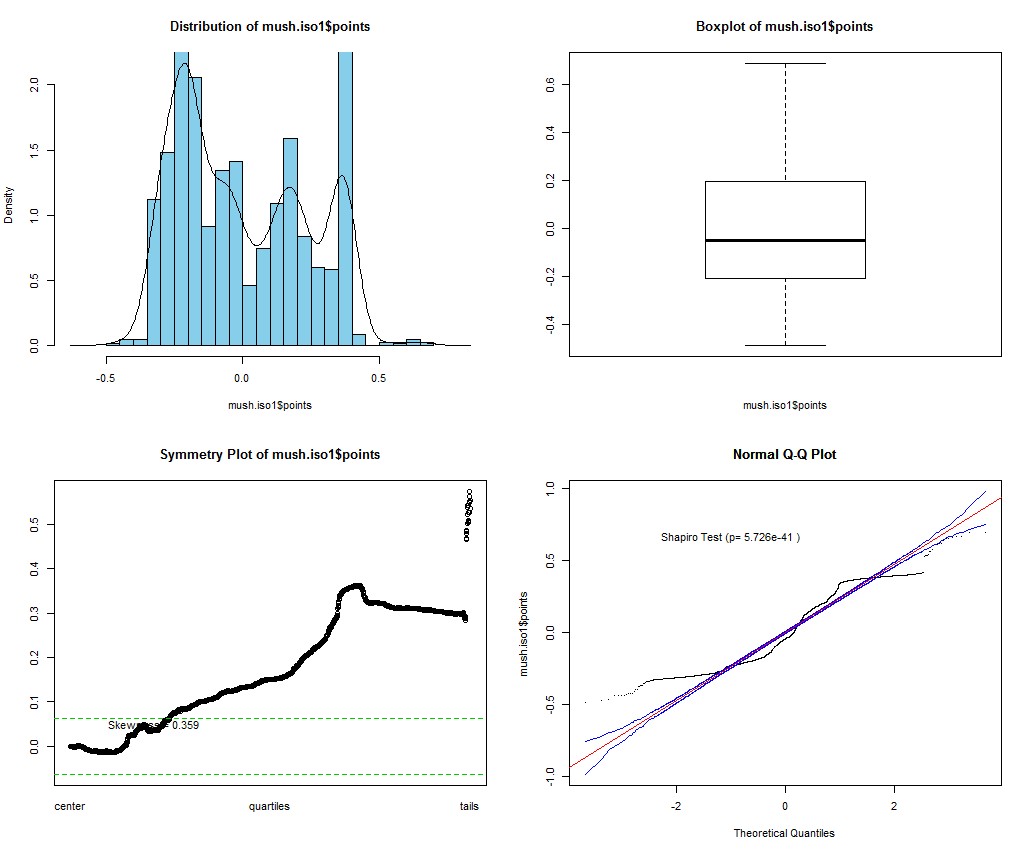
initial value 48.551357

iter 5 value 43.807540

final value 43.458998 🡨 very high stress!

converged

> Statplot(mush.iso$points)



We can see that even the 1-D solution is quite stressed, however it does seems to form some clusters of mushrooms as the histogram is strongly multimodal. We now consider a two-dimensional solution.

> Statplot(mush.iso1$points)

> mush.iso2 = isoMDS(mush.dist,k=2)

initial value 39.114915

iter 5 value 29.039498

iter 10 value 27.388035

iter 15 value 26.263594

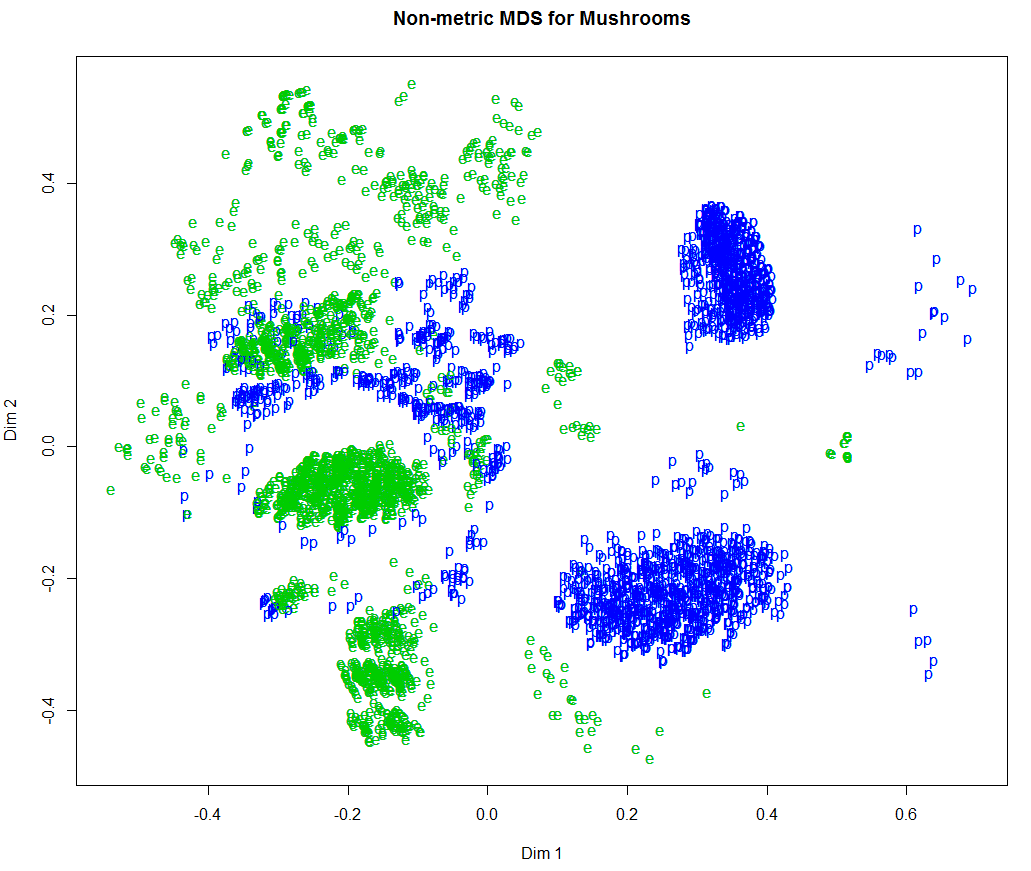
iter 20 value 25.836340

final value 25.691631 🡨 the 2-D solution is still pretty stressed, but definitely better than the 1-D solution!

converged

> plot(mush.iso2$points,xlab="Dim 1",ylab="Dim 2",main="Non-metric MDS for Mushrooms",type="n")

> text(mush.iso2$points,as.character(Mushrooms.train$Poisonous),

col=as.numeric(Mushrooms.train$Poisonous)+2)

> mush.iso3 = isoMDS(mush.dist,k=3)

initial value 26.376961

iter 5 value 19.024057

iter 10 value 17.470162

iter 15 value 17.115541

iter 20 value 16.983762

iter 20 value 16.977573

iter 20 value 16.973459

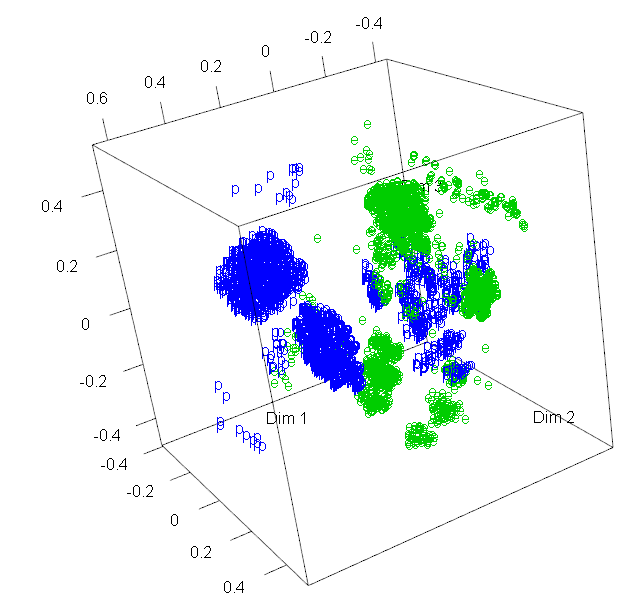
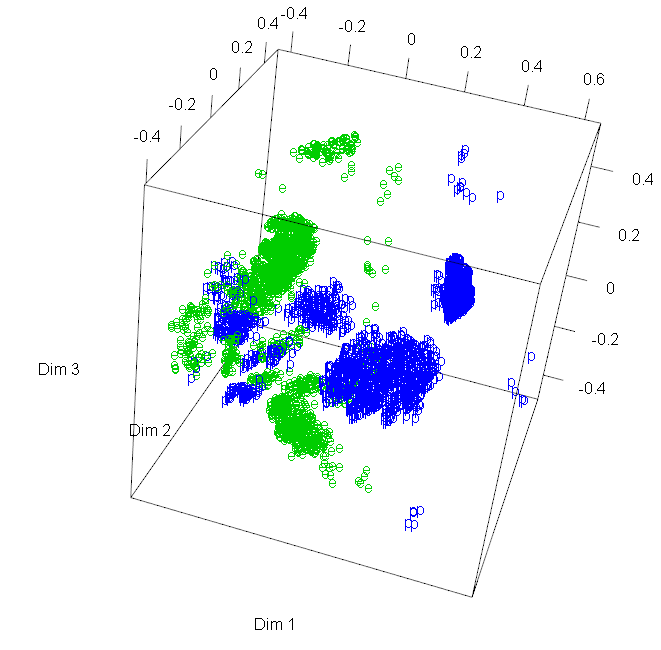
final value 16.973459 🡨 not outstanding, but respectable given the scope of the problem!

Converged

> plot3d(mush.iso3$points,xlab="Dim 1",ylab="Dim 2",zlab="Dim 3",type="n")

> text3d(mush.iso3$points,text=as.character(Mushrooms.train$Poisonous),

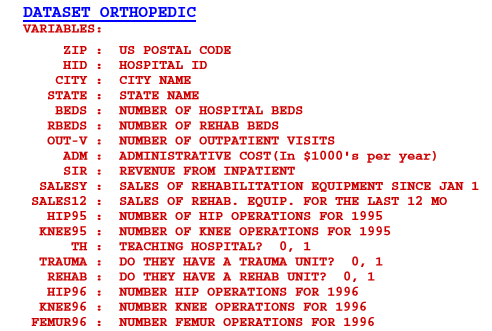
col=as.numeric(Mushrooms.train$Poisonous)+2)



The non-metric MDS of these data produces superior results in my opinion in terms forming distinct groups/clusters and the homogeneity of the clusters/groups in terms of edibility. It is important to realize that this algorithm does not specifically seek to form distinct groups or clusters of similar observations, and in the case of these data, it is also not seeking to classify mushrooms according to their edibility. However, both goals seem to be achieved given the underlying dissimilarities between observations in these data!

Example 3.4: Orthopedic Sales Data (Mixture of Data Types)

The goal in the analysis of these data is to identify hospitals that are underperforming in terms of orthopedic sales given what we know about them, i.e. to identify hospitals with good sales potential. In order to achieve this goal we might want to look for are groups of similar hospitals on the basis of their characteristics. Of course, in order to do this we need to some measure(s) of similarity/dissimilarity between hospitals given their characteristics. The variables available for this purpose are shown below:



You will notice that three of the variables are binary/dichotomous categorical variables: Teaching Hosptial, Trauma Unit, and Rehabilitation Unit. The others are numeric, with most of these having very skewed right distributions with lots of zeroes.

Let’s first compute the distance between hospitals after reading these data into R. Using the .csv file I sent you we can do the following.

> Orthopedic = read.table(file.choose(),header=T,sep=",")

> names(Orthopedic)

[1] "ZIP" "HospID" "City" "State" "Beds" "RBeds" "Outpatients" "Admin"

[9] "Inpatient" "Hip95" "Knee95" "SalesYr" "Sales12" "Teach" "Trauma" "Rehab"

[17] "Hip96" "Knee96" "Femur96"

> head(Orthopedic)

ZIP HospID City State Beds RBeds Outpatients Admin Inpatient Hip95 Knee95 SalesYr Sales12 Teach Trauma

1 919 44604 San Juan PR 386 0 0 12975 4106 2 2 90 101 0 0

2 935 5004 San Juan PR 311 0 118065 11309 21049 0 0 70 26 1 0

3 1060 157014 Northampton MA 175 0 114912 7365 5862 0 1 93 60 0 0

4 1104 194014 Springfield MA 324 0 95702 10406 13648 28 29 122 94 0 0

5 1199 195514 Springfield MA 507 0 108258 22361 15081 0 7 148 97 1 1

6 1420 93014 Fitchburg MA 175 25 26428 5619 5701 6 11 61 48 1 1

Rehab Hip96 Knee96 Femur96

1 0 101 137 100

2 0 66 40 330

3 0 97 64 58

4 0 152 95 116

5 0 166 110 236

6 1 57 44 40

> summary(Orthopedic)

ZIP HospID City State Beds RBeds

Min. : 612 006F61: 1 Chicago : 45 CA : 458 Min. : 0.0 Min. : 0.000

1st Qu.:28552 006G61: 1 Houston : 41 TX : 342 1st Qu.: 69.0 1st Qu.: 0.000

Median :49001 009A74: 1 Philadelphia: 38 NY : 241 Median : 136.0 Median : 0.000

Mean :50595 011A71: 1 Los Angeles : 28 PA : 238 Mean : 191.2 Mean : 7.244

3rd Qu.:75235 011A72: 1 Dallas : 24 FL : 228 3rd Qu.: 262.0 3rd Qu.: 0.000

Max. :99901 015A63: 1 New York : 24 IL : 208 Max. :1476.0 Max. :850.000

(Other):4697 (Other) :4503 (Other):2988

Outpatients Admin Inpatient Hip95 Knee95 SalesYr

Min. : 0 Min. : 0 Min. : 0 Min. : 0.00 Min. : 0.00 Min. : 0.00

1st Qu.: 7510 1st Qu.: 1932 1st Qu.: 1312 1st Qu.: 0.00 1st Qu.: 0.00 1st Qu.: 7.00

Median : 20876 Median : 4508 Median : 3384 Median : 1.00 Median : 2.00 Median : 28.00

Mean : 47354 Mean : 6689 Mean : 4849 Mean : 25.91 Mean : 41.05 Mean : 51.27

3rd Qu.: 47700 3rd Qu.: 9402 3rd Qu.: 6832 3rd Qu.: 23.00 3rd Qu.: 33.00 3rd Qu.: 70.00

Max. :1986530 Max. :66439 Max. :70297 Max. :1209.00 Max. :2770.00 Max. :1421.00

Sales12 Teach Trauma Rehab Hip96 Knee96

Min. : 0.00 Min. :0.0000 Min. :0.0000 Min. :0.0000 Min. : 0.0 Min. : 0.00

1st Qu.: 1.00 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.: 8.0 1st Qu.: 0.00

Median : 18.00 Median :0.0000 Median :0.0000 Median :0.0000 Median : 29.0 Median : 18.00

Mean : 41.73 Mean :0.2737 Mean :0.1225 Mean :0.1839 Mean : 52.6 Mean : 41.91

3rd Qu.: 52.50 3rd Qu.:1.0000 3rd Qu.:0.0000 3rd Qu.:0.0000 3rd Qu.: 71.0 3rd Qu.: 56.00

Max. :868.00 Max. :1.0000 Max. :1.0000 Max. :1.0000 Max. :1373.0 Max. :1081.00

Femur96

Min. : 0.00

1st Qu.: 11.00

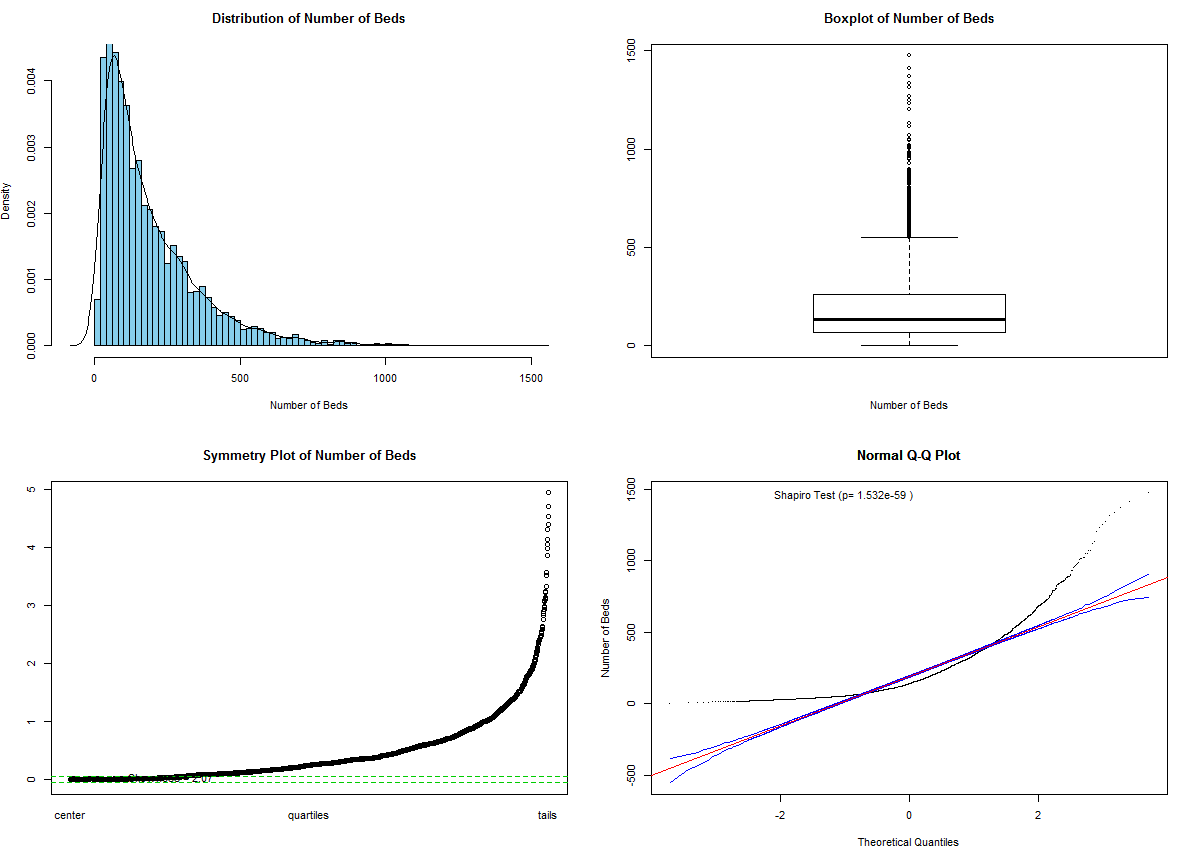
Median : 34.00

Mean : 49.39

3rd Qu.: 74.00

Max. :489.00

Notice that Teach, Trauma, and Rehab are all 0-1 coded (1 = Yes and 0 = No) thus the mean is the proportion of hospital facilities of each type, e.g. 27.37% of the hospitals are considered teaching hospitals and 18.39% have a rehabilitative services.

> Statplot(Orthopedic$Beds,xname=”Number of Beds”)  


> Orthopedic$logBeds = log(Orthopedic$Beds+1) 🡨 log-transform number of beds ()

Notice that the Orthopedic data frame now contains a new variable logBeds.

> names(Orthopedic)

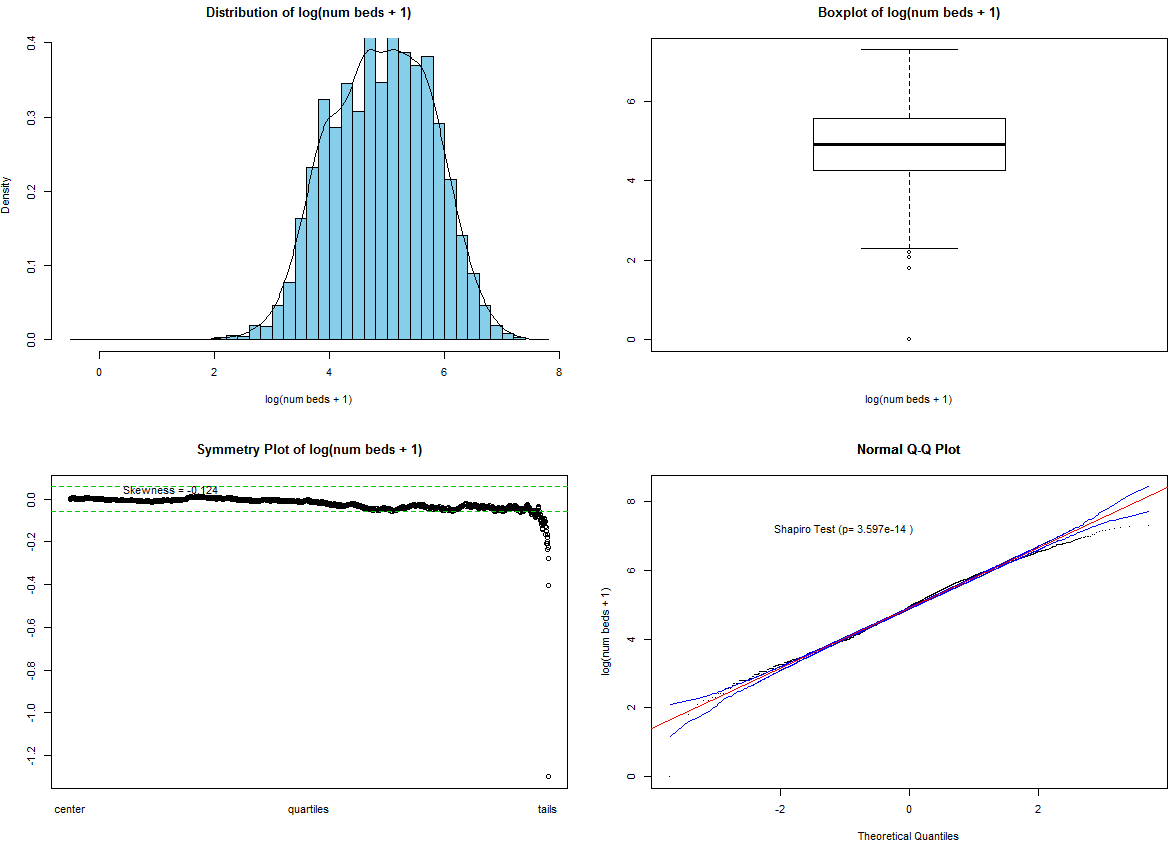
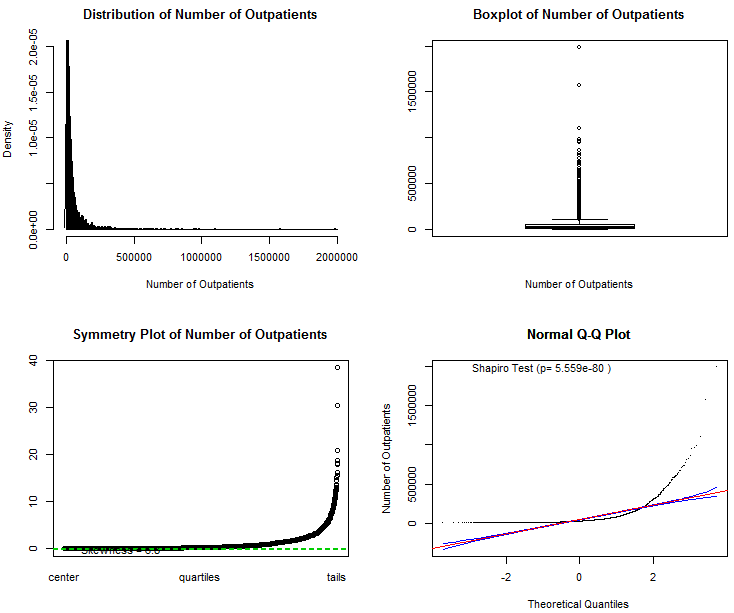
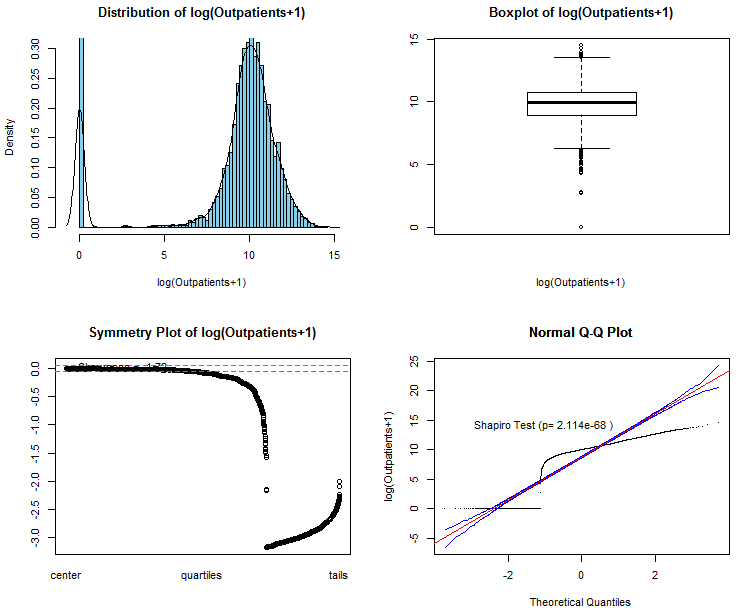
[1] "ZIP" "HospID" "City" "State"

[5] "Beds" "RBeds" "Outpatients" "Admin"

[9] "Inpatient" "Hip95" "Knee95" "SalesYr"

[13] "Sales12" "Teach" "Trauma" "Rehab"

[17] "Hip96" "Knee96" "Femur96" "logBeds"

> Statplot(Orthopedic$logBeds,xname="log(num beds + 1)")  
  
> Statplot(Orthopedic$Outpatients,xname="Number of Outpatients")  
> Statplot(log(Orthopedic$Outpatients+1),xname="log(Outpatients+1)")  
 

> Orthopedic$logOut = log(Orthopedic$Outpatients+1)

> names(Orthopedic)

[1] "ZIP" "HospID" "City" "State"

[5] "Beds" "RBeds" "Outpatients" "Admin"

[9] "Inpatient" "Hip95" "Knee95" "SalesYr"

[13] "Sales12" "Teach" "Trauma" "Rehab"

[17] "Hip96" "Knee96" "Femur96" "logBeds"

[21] "logOut"

Repeating the processes used above for the other numeric variables we can create the following version of the orthopedic sales data frame.

> Orthopedic$logInpat = log(Orthopedic$Inpatient+1)

> Orthopedic$logHip95 = log(Orthopedic$Hip95+1)

> Orthopedic$logKnee95 = log(Orthopedic$Knee95+1)

> Orthopedic$logSales = log(Orthopedic$SalesYr+1)

> Orthopedic$logSales12 = log(Orthopedic$Sales12+1)

> Orthopedic$logHip96 = log(Orthopedic$Hip96+1)

> Orthopedic$logKnee96 = log(Orthopedic$Knee96+1)

> Orthopedic$logFemur96 = log(Orthopedic$Femur96+1)

> names(Orthopedic)

[1] "ZIP" "HospID" "City" "State" "Beds" "RBeds" "Outpatients"

[8] "Admin" "Inpatient" "Hip95" "Knee95" "SalesYr" "Sales12" "Teach"

[15] "Trauma" "Rehab" "Hip96" "Knee96" "Femur96" "logBeds" "logOut"

[22] "logRBeds" "logAdmin" "logInpat" "logHip95" "logKnee95" "logSales" "logSales12"

[29] "logHip96" "logKnee96" "logFemur96"

We begin by forming an appropriate distance matrix that measuring the pairwise dissimilarity between these 4,703 hospitals in this data base, realizing that Teach, Trauma, and Rehab are all dichotomous binary variables and using the log-transformed versions of the other continuous variates.

> library(cluster)

> ortho.mat = Orthopedic[,c(14:16,20:31)]  
> ?daisy 🡨 view the help file for the daisy command  
  
The first three columns (Teach, Trauma, Rehab) are all asymmetric binary variables, so we need identify them as such in the call to daisy command.

> ortho.daisy = daisy(ortho.mat,type=list(asymm=1:3))

> summary(as.numeric(ortho.daisy))

11056753 dissimilarities, summarized :

Min. 1st Qu. Median Mean 3rd Qu. Max.

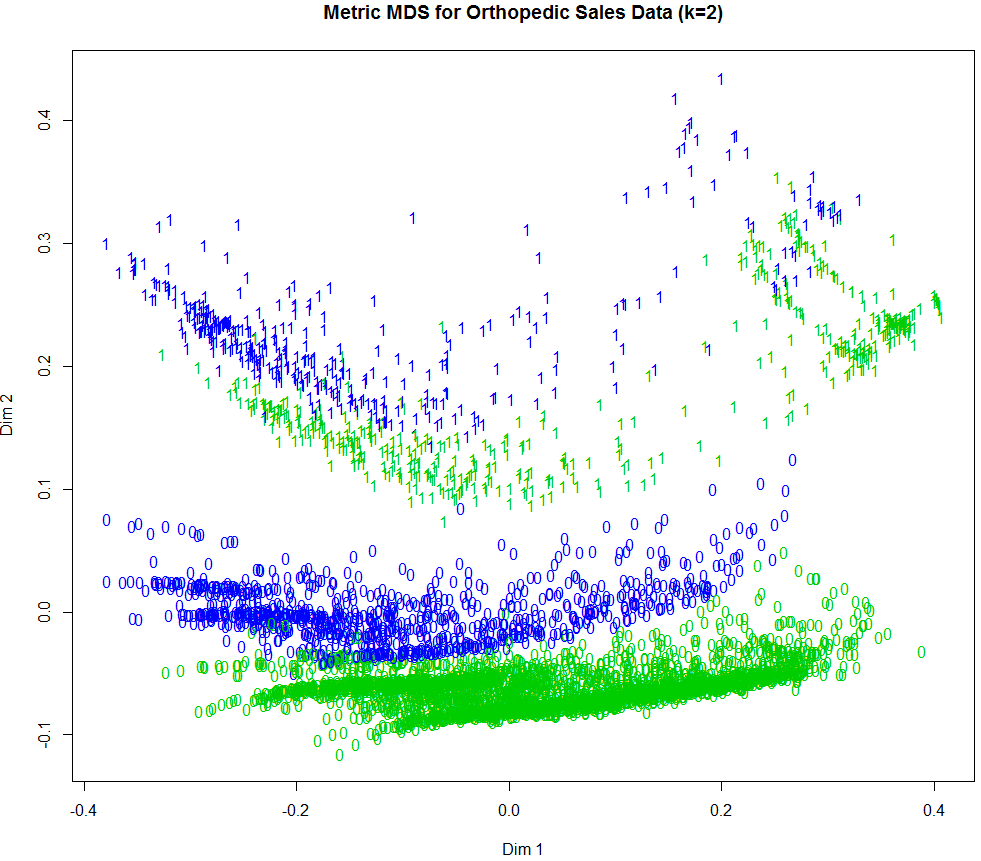
0.00000 0.17338 0.25841 0.27514 0.36062 0.83207

Metric : mixed ; Types = A, A, A, I, I, I, I, I, I, I, I, I, I, I, I

Number of objects : 4703  
  
> ortho.mds = cmdscale(ortho.daisy,k=2)  
> plot(ortho.mds,type="n",xlab="Dim 1",ylab="Dim 2")

> text(ortho.mds,labels=Orthopedic$Rehab,col=as.numeric(Orthopedic$Teach)+3)

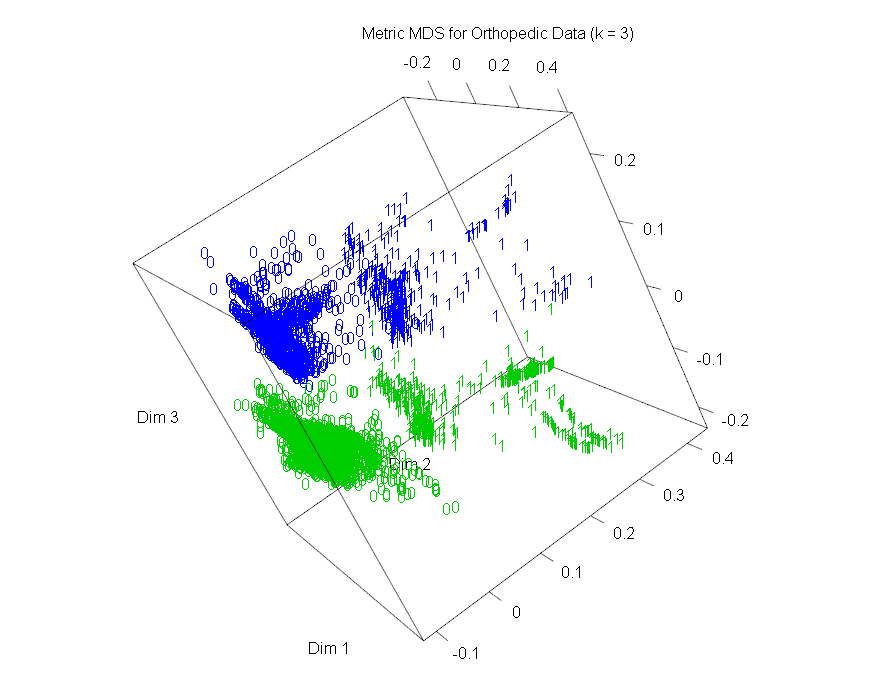
> title(main="Metric MDS for Orthopedic Data (k =2)")



> ortho.mds3 = cmdscale(ortho.daisy,k=3)

> plot3d(ortho.mds3,xlab="Dim 1",ylab="Dim 2",zlab="Dim 3",type="n")

> text3d(ortho.mds3,texts=Orthopedic$Rehab,col=as.numeric(Orthopedic$Teach)+3)  
> title3d(main="Metric MDS for Orthopedic Data (k = 3)")

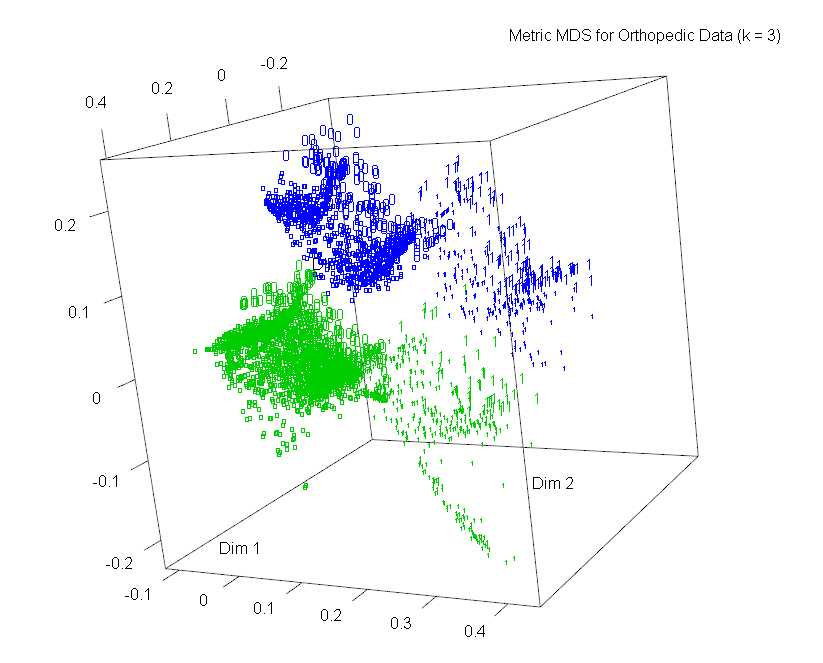


> plot3d(ortho.mds3,xlab="Dim 1",ylab="Dim 2",zlab="Dim 3",type="n")

> text3d(ortho.mds3,texts=Orthopedic$Rehab,col=as.numeric(Orthopedic$Teach)+3,

cex=(as.numeric(Orthopedic$Trauma)+.8)/2)

> title3d(main="Metric MDS for Orthopedic Data (k = 3)")



This plot shows that much of the separation between the groups of hospitals is due to the asymmetric binary variables. The case labels are determined by the rehabilitation status of the hospital, color by the teaching status, and the size of the plotting symbol by trauma status.

**Non-Metric Multidimensional Scaling (isoMDS)**

> library(MASS)

> ortho.iso = isoMDS(ortho.daisy,k=2)

> ortho.iso = isoMDS(ortho.daisy,k=2)

Error in isoMDS(ortho.daisy, k = 2) :

zero or negative distance between objects 3319 and 3841

> ortho.iso = isoMDS(ortho.daisy+.01,k=2)

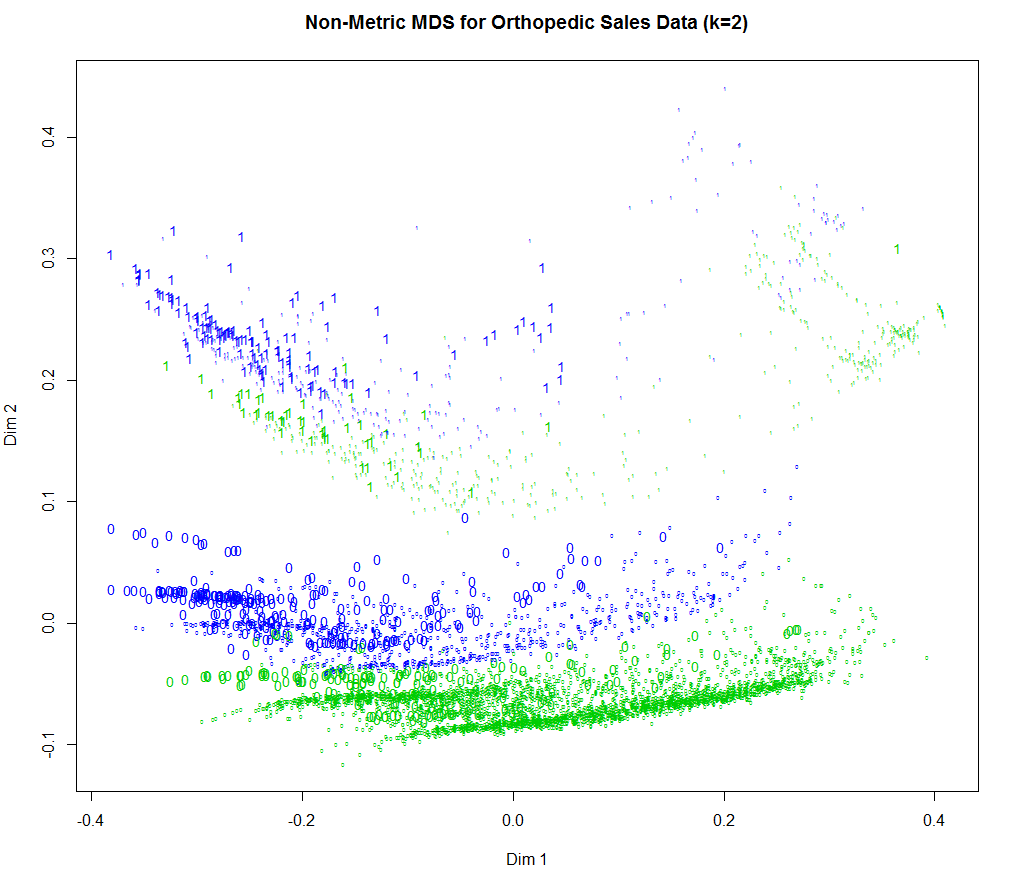
initial value 19.219630

final value 19.219630

converged

> plot(ortho.iso$points,xlab="Dim 1",ylab="Dim 2",type="n")

> text(ortho.iso$points,label=as.character(Orthopedic$Rehab),col=as.numeric(Orthopedic$Teach)+3,  
cex=(as.numeric(Orthopedic$Trauma)+.8)/2)



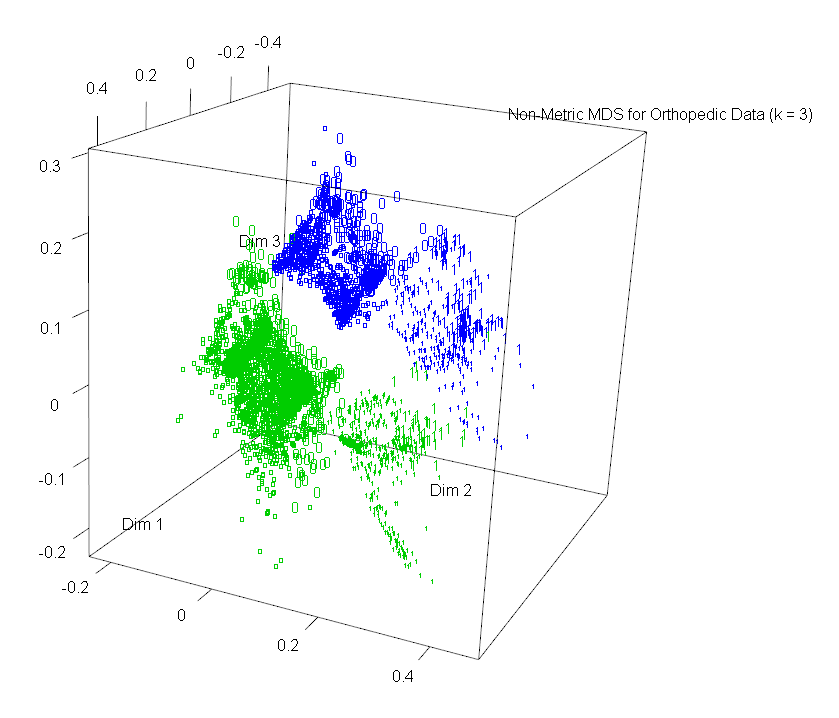
> ortho.iso3 = isoMDS(ortho.daisy+.01,k=3) 🡨 WARNING: this takes a VERY LONG time to run!!

> plot3d(ortho.mds3,xlab="Dim 1",ylab="Dim 2",zlab="Dim 3",type="n")

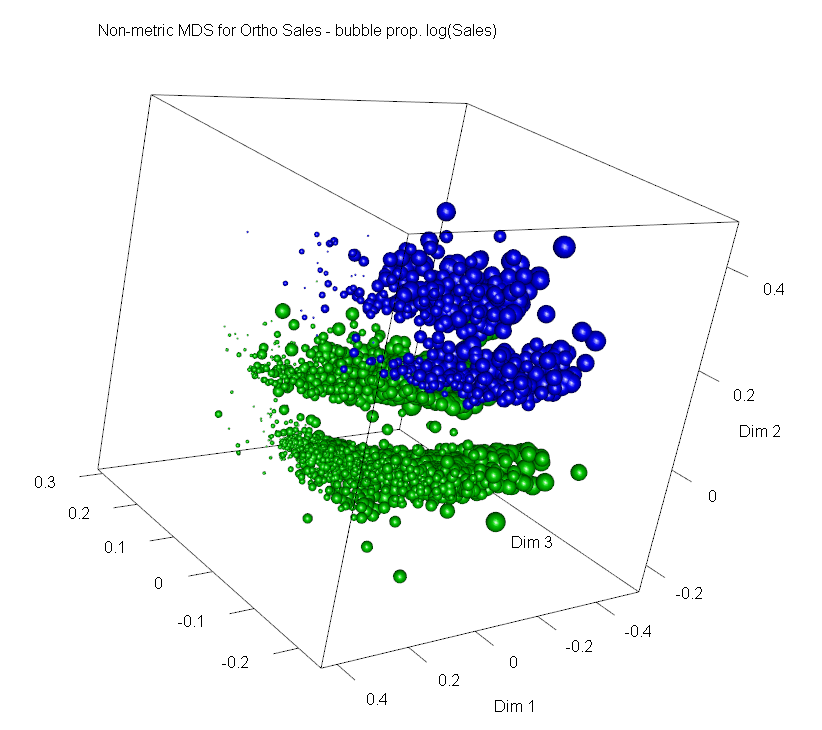
> text3d(ortho.mds3,texts=Orthopedic$Rehab,col=as.numeric(Orthopedic$Teach)+3,

cex=(as.numeric(Orthopedic$Trauma)+.8)/2)

> title3d(main="Metric MDS for Orthopedic Data (k = 3)")



We can use the log of the total sales from last year to control the size of the data points using the spheres3d command.

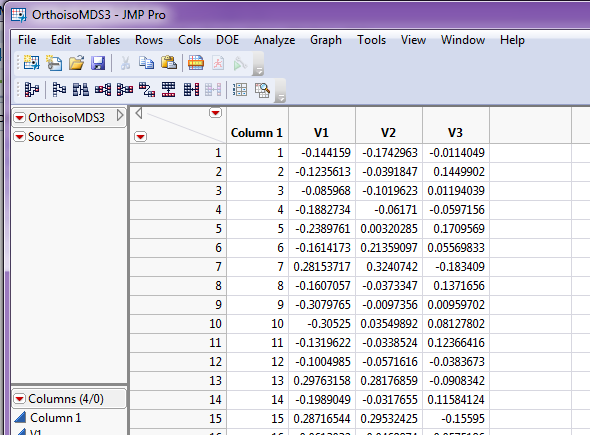
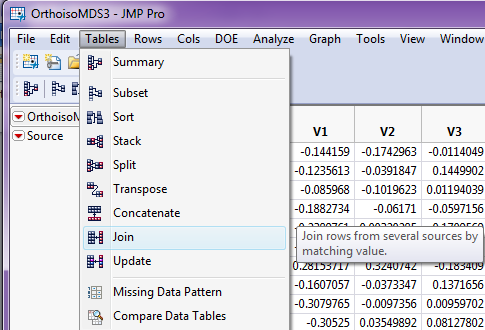


**Exporting Results to JMP**

We can export the non-metric MDS coordinates (k = 3) to JMP and use them in plotting and potentially analyzing these data.

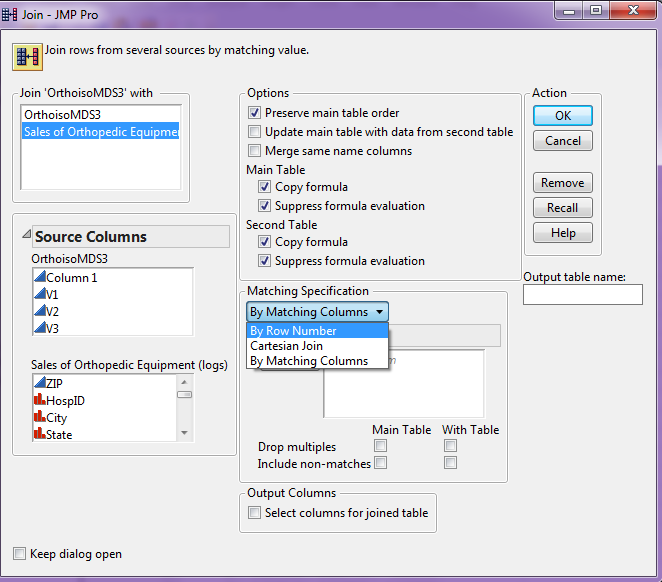
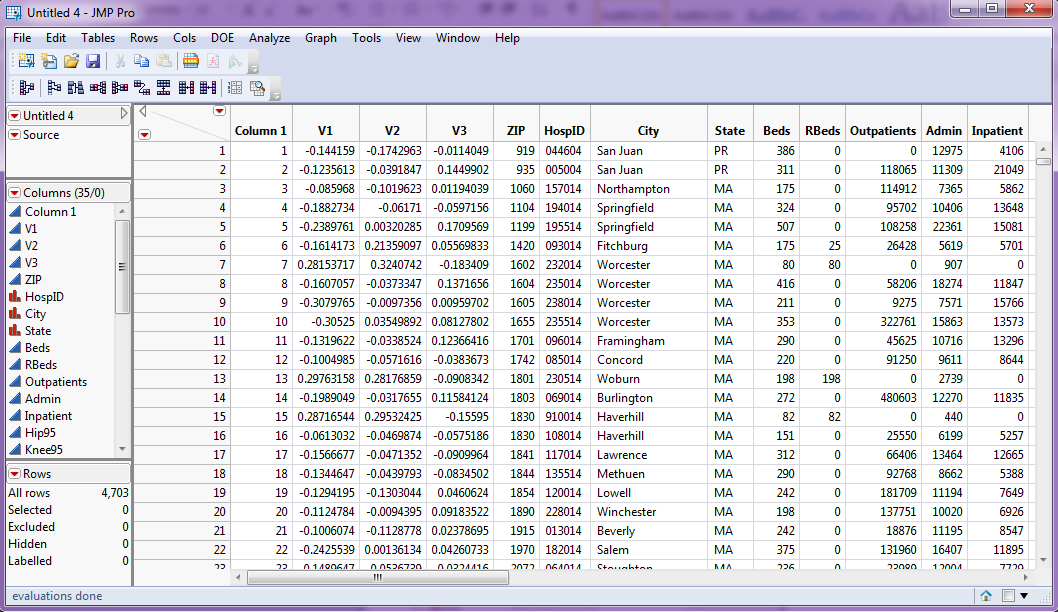
> write.csv(ortho.iso3$points,"OrthoisoMDS3.csv")

Open the OrthoisoMDS3.csv file in JMP

We can select **Tables > Join** to add these columns to the original database in JMP.

We want to match by Row Number here.

The resulting JMP table adds coordinates returned from the non-metric MDS (k = 3) to the original data file.

**MDS for Variables vs. Observations**

Recall that we can also measure dissimilarities/similarities between variables and thus we can perform MDS using variables rather than observations. If the variables are all numeric we can simply transpose the data matrix so rows are variables and columns are observations.

Then we can use standard metrics for measuring distance for numeric data.

Example 3.5: Italian Olive Oils - measuring distance between fatty acids (i.e. variables)

> names(Olives)

[1] "Region.name" "Area.name" "Region" "Area" "palmitic" "palmitoleic" "strearic"

[8] "oleic" "linoleic" "eicosanoic" "linolenic" "eicosenoic"

> olive.mat = Olives[,5:12]

> olive.vars = t(olive.mat)

> row.names(olive.vars)

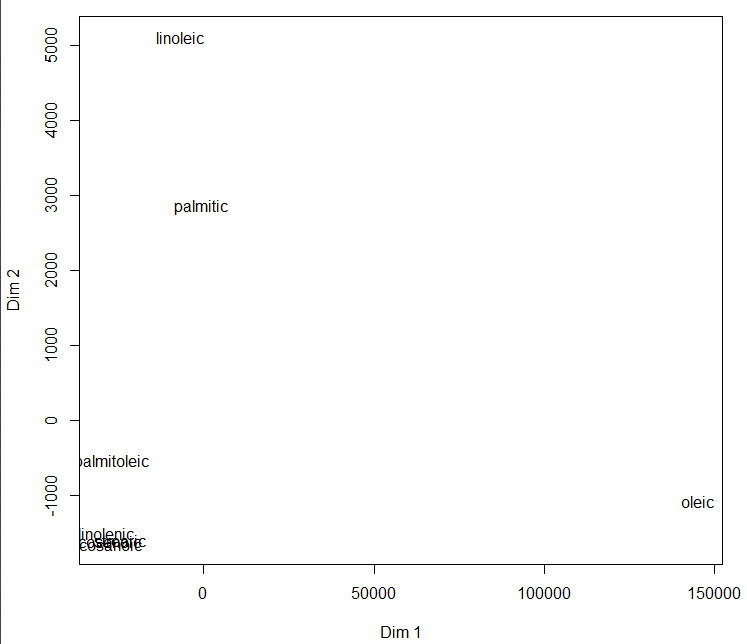
[1] "palmitic" "palmitoleic" "strearic" "oleic" "linoleic" "eicosanoic" "linolenic"

[8] "eicosenoic"

> var.dist = dist(olive.vars)

> olive.mds = cmdscale(var.dist,k=2)

> plot(olive.mds,xlab="Dim 1",ylab="Dim 2",type="n")

> text(olive.mds,labels=row.names(olive.vars))  


Oops, we forgot to scale!!!

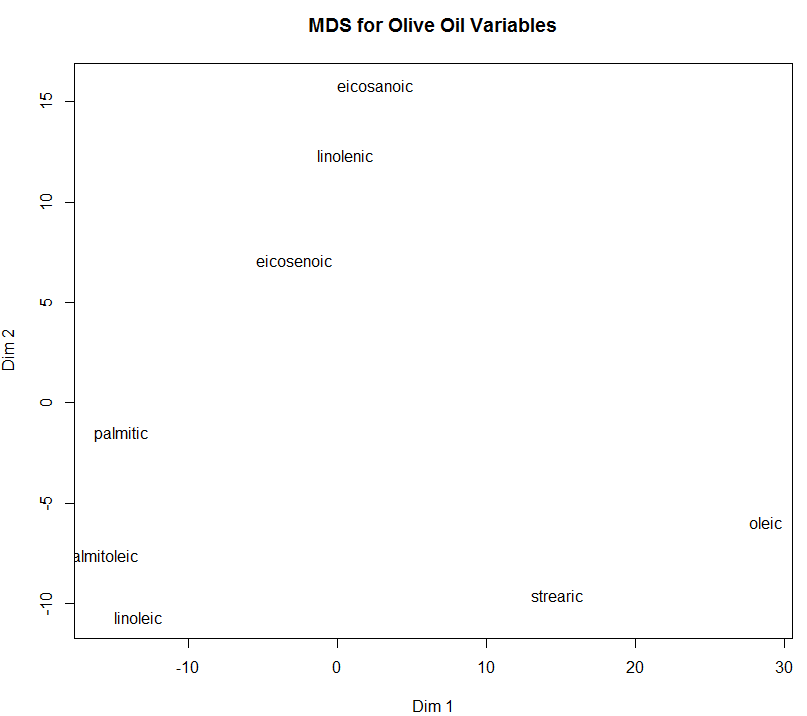
> olive.vars = t(scale(olive.mat))

> var.dist = dist(olive.vars)

> olive.mds = cmdscale(var.dist,k=2)

> plot(olive.mds,xlab="Dim 1",ylab="Dim 2",type="n")

> text(olive.mds,labels=row.names(olive.vars))



We can also use distances between variables based upon correlations.

> olive.cor = cor(olive.mat)

> olive.cor

palmitic palmitoleic strearic oleic linoleic eicosanoic linolenic eicosenoic

palmitic 1.0000000 0.83560497 -0.17039178 -0.8373354 0.46068446 0.31932669 0.22829912 0.50195179

palmitoleic 0.8356050 1.00000000 -0.22218545 -0.8524384 0.62162666 0.09311163 0.08548117 0.41635048

strearic -0.1703918 -0.22218545 1.00000000 0.1135987 -0.19781693 0.01891719 -0.04097892 0.14037748

oleic -0.8373354 -0.85243835 0.11359873 1.0000000 -0.85031837 -0.21817123 -0.31996234 -0.42414586

linoleic 0.4606845 0.62162666 -0.19781693 -0.8503184 1.00000000 -0.05743858 0.21097260 0.08904499

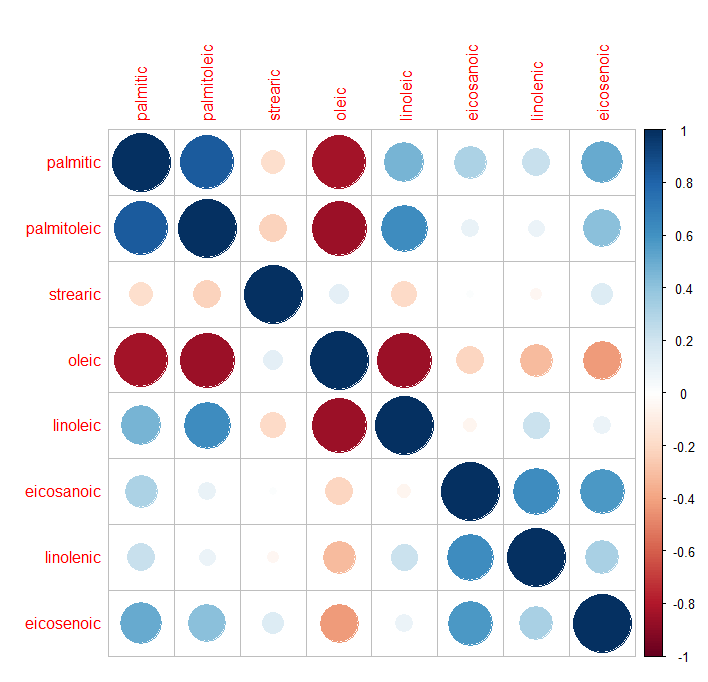
eicosanoic 0.3193267 0.09311163 0.01891719 -0.2181712 -0.05743858 1.00000000 0.62023577 0.57831851

linolenic 0.2282991 0.08548117 -0.04097892 -0.3199623 0.21097260 0.62023577 1.00000000 0.32866349

eicosenoic 0.5019518 0.41635048 0.14037748 -0.4241459 0.08904499 0.57831851 0.32866349 1.00000000

> library(corrplot)

> corrplot(olive.cor)



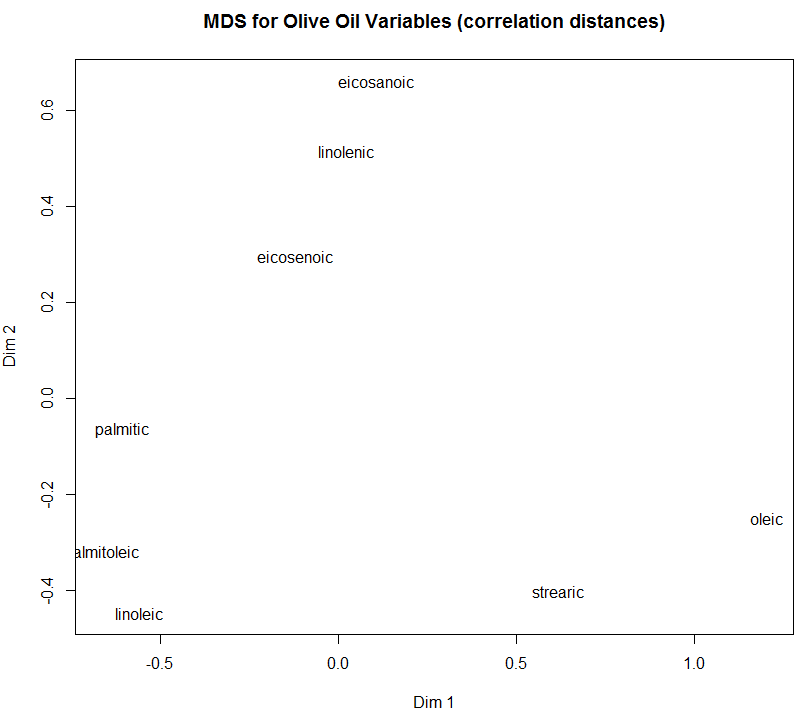
There are different ways to convert correlations to a distance between variables, here are some standard choices:

> var.dist = sqrt(2\*(1-olive.cor))  
> var.mds = cmdscale(var.dist,k=2)

> plot(var.mds,xlab="Dim 1",ylab="Dim 2",type="n")

> text(var.mds,labels=names(olive.mat))

> title(main="MDS for Olive Oil Variables (correlation distances)")



**Non-Metric MDS for Olive Variables**

> var.iso = isoMDS(var.dist,k=2)

initial value 11.394466

iter 5 value 5.348152

iter 10 value 4.699663

iter 15 value 4.034934

iter 20 value 3.621228

iter 25 value 3.300183

iter 30 value 2.630931

iter 35 value 2.297891

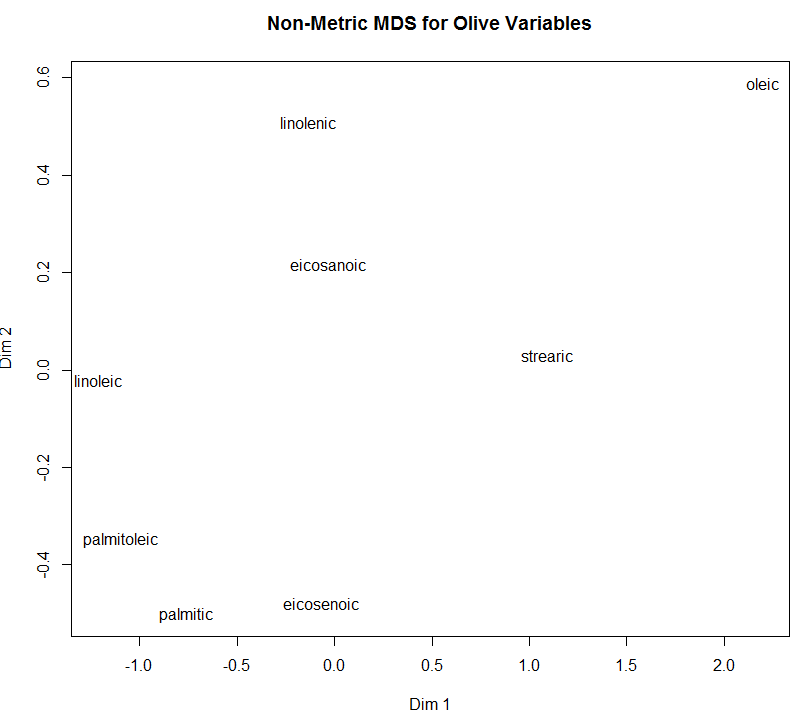
iter 40 value 1.856453

final value 1.820113

converged

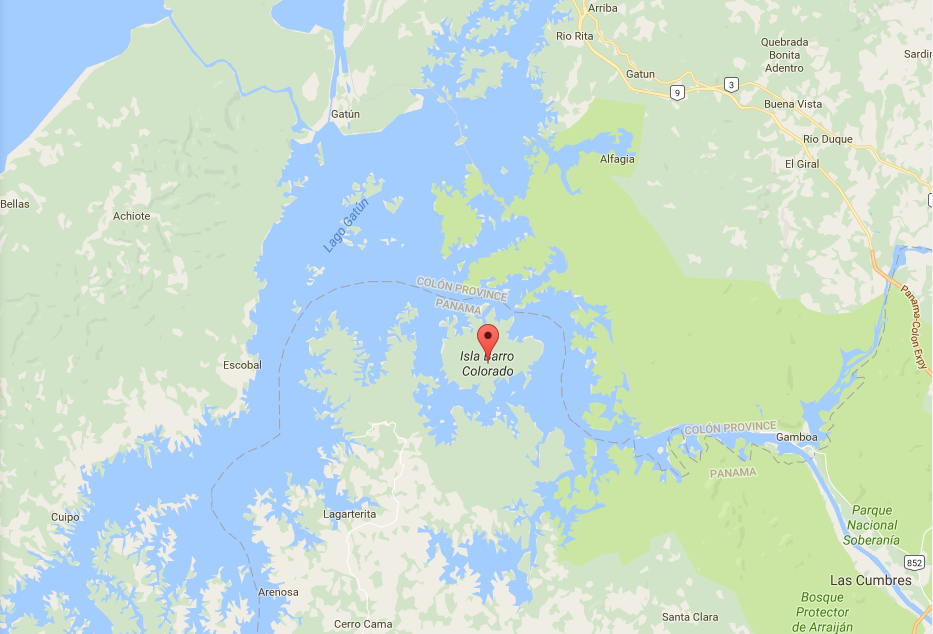
> plot(var.iso$points,xlab="Dim 1",ylab="Dim 2",type="n",main="Non-Metric MDS for Olive Variables")

> text(var.iso$points,labels=names(olive.mat))



Example 3.6: Barro Colorado Island Tree Counts (site-species study)

These data come from a study where tree counts on 1-hectare plots in the Barro Colorado Island and associated site level information were collected. Barro Colorado Island is located in the man-made Gatun Lake in the middle of the Panama Canal. The island was formed when the waters of the Chagres River were dammed to form the lake in 1913. A collected data from 50 plots (rows) of 1 hectare with counts of trees on each plot with total of 225 species (columns) are contained the file **Barro Island Species.csv**. Full Latin names are used for all tree species. The names were updated against <http://www.theplantlist.org> and Kress et al. (2009) which allows matching 207 of species against <http://datadryad.org/resource/doi:10.5061/dryad.63q27> (Zanne et al., 2014).



There are lots of potential analyses we could conduct using these data, however let’s first consider questions that can be answered using the methods discussed above. Some potential research questions might be:

* Are there site differences in the distribution of tree species found?
* If there are differences, what are the nature of the between the sites and the species found?
* What are the most prevalent species found? Least?
* How do the sites differ in terms of number of trees found?

Other questions that we might have that will require additional statistical methods are:

* How diverse are the sites in terms of species found? How similar?
* What sites are most similar in terms of their species distribution? Are there groups of sites with similar species patterns?
* What species are most associated with one another?
* How are the species found related to the environmental measures at each site?   
  (Note: these measures are contained in the file BCI with Environmentals.csv as well)

In this example we will focus on the use of multidimensional scaling to examine both sites (rows) and species (columns).

> Barro = read.csv(file.choose())

> names(Barro)

> names(Barro)

[1] "Site" "UTM.EW"

[3] "UTM.NS" "Precipitation"

[5] "Elevation" "Age.cat"

[7] "Geology" "Habitat"

[9] "Stream" "EnvHet"

[11] "Abarema.macradenia" "Vachellia.melanoceras"

[13] "Acalypha.diversifolia" "Acalypha.macrostachya"

… … …

[231] "Zanthoxylum.ekmanii" "Zanthoxylum.juniperinum"

[233] "Zanthoxylum.panamense" "Zanthoxylum.setulosum"

[235] "Zuelania.guidonia"

> dim(Barro)

[1] 50 235

> spec.mat = Barro[,-c(1:10)]

Calculate total counts for each site and species.

> rowSums(spec.mat)

[1] 448 435 463 508 505 412 416 431 409 483 401 366 409 438 462 437 381 347 433 429 408 418

[23] 340 392 442 407 417 387 364 475 421 459 436 447 601 430 435 447 424 489 402 414 407 409

[45] 444 430 425 415 427 432

> colSums(spec.mat)

Abarema.macradenia Vachellia.melanoceras

1 3

Acalypha.diversifolia Acalypha.macrostachya

2 1

Adelia.triloba Aegiphila.panamensis

92 23

Alchornea.costaricensis Alchornea.latifolia

156 1

Alibertia.edulis Allophylus.psilospermus

1 27

Alseis.blackiana Amaioua.corymbosa

983 3

Anacardium.excelsum Andira.inermis

22 28

… …

Turpinia.occidentalis Unonopsis.pittieri

58 163

Virola.multiflora Virola.sebifera

25 617

Virola.surinamensis Vismia.baccifera

164 1

Vochysia.ferruginea Xylopia.macrantha

12 143

Zanthoxylum.ekmanii Zanthoxylum.juniperinum

149 45

Zanthoxylum.panamense Zanthoxylum.setulosum

67 1

Zuelania.guidonia

10

As we can see some of the species frequencies are quite small! We should probably focus our attention on the most prevalent species across the 50 sites when doing our analysis.

We will focus on species that were at least 100 which is about the 80th percentile of the species frequencies. This will leave about species (turns out it is 44 species).

> spec.counts = colSums(spec.mat)

> quantile(spec.counts,probs=seq(0,1,.01))

> spec100.mat = spec.mat[,colSums(spec.mat)>100]

> dim(spec100.mat)

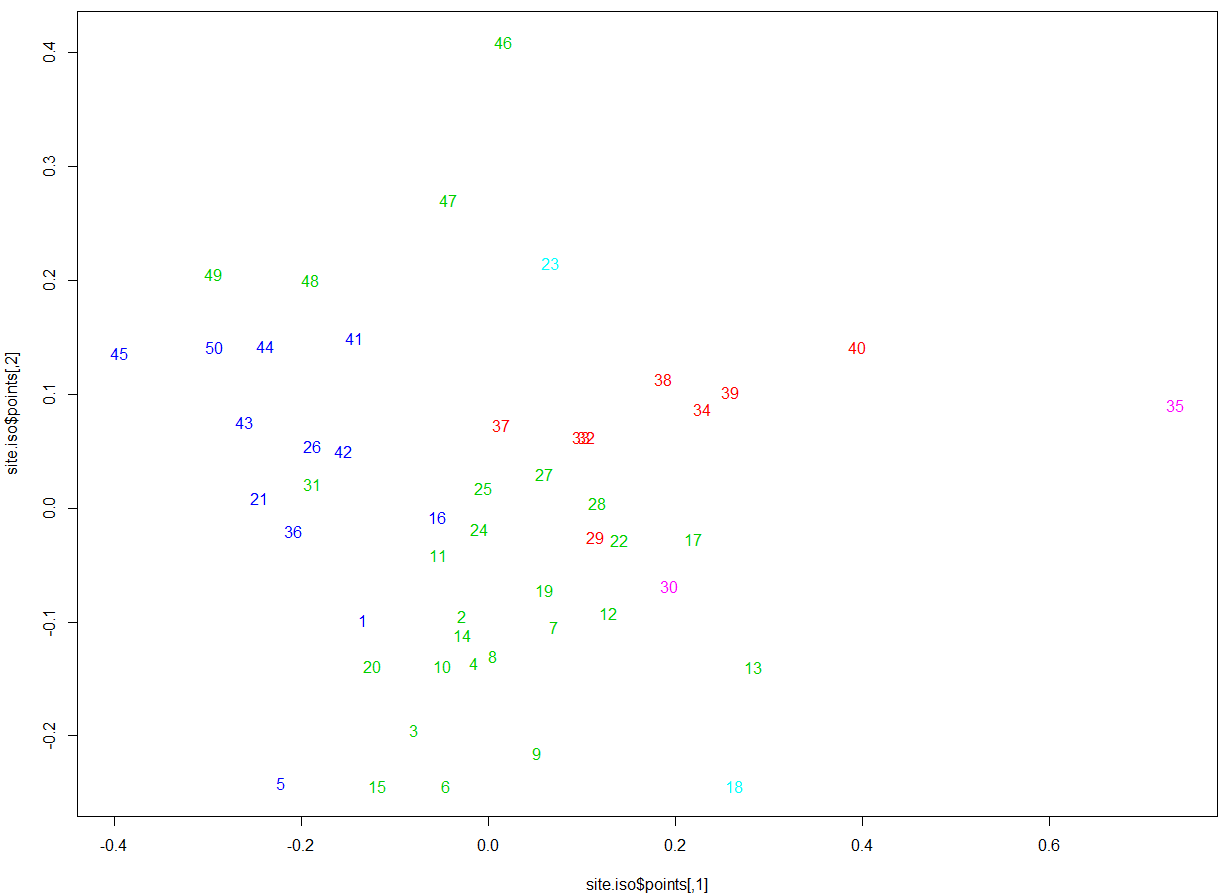
[1] 50 44

> library(proxy)  
> site.dist = dist(spec100.mat,method=”Bray”)

> site.iso = isoMDS(site.dist,k=2)

> plot(site.iso$points,type=”n”)

> text(site.iso$points,as.character(Barro$Site),col= as.numeric(Barro$Habitat)+1)



By transposing the site (rows) and species (columns) count matrix we can explore the relationships between the species counts in a lower dimensional representation as well.

> tspec100.mat = t(spec100.mat)

> spec.dist = dist(tspec100.mat,method=”Bray”)

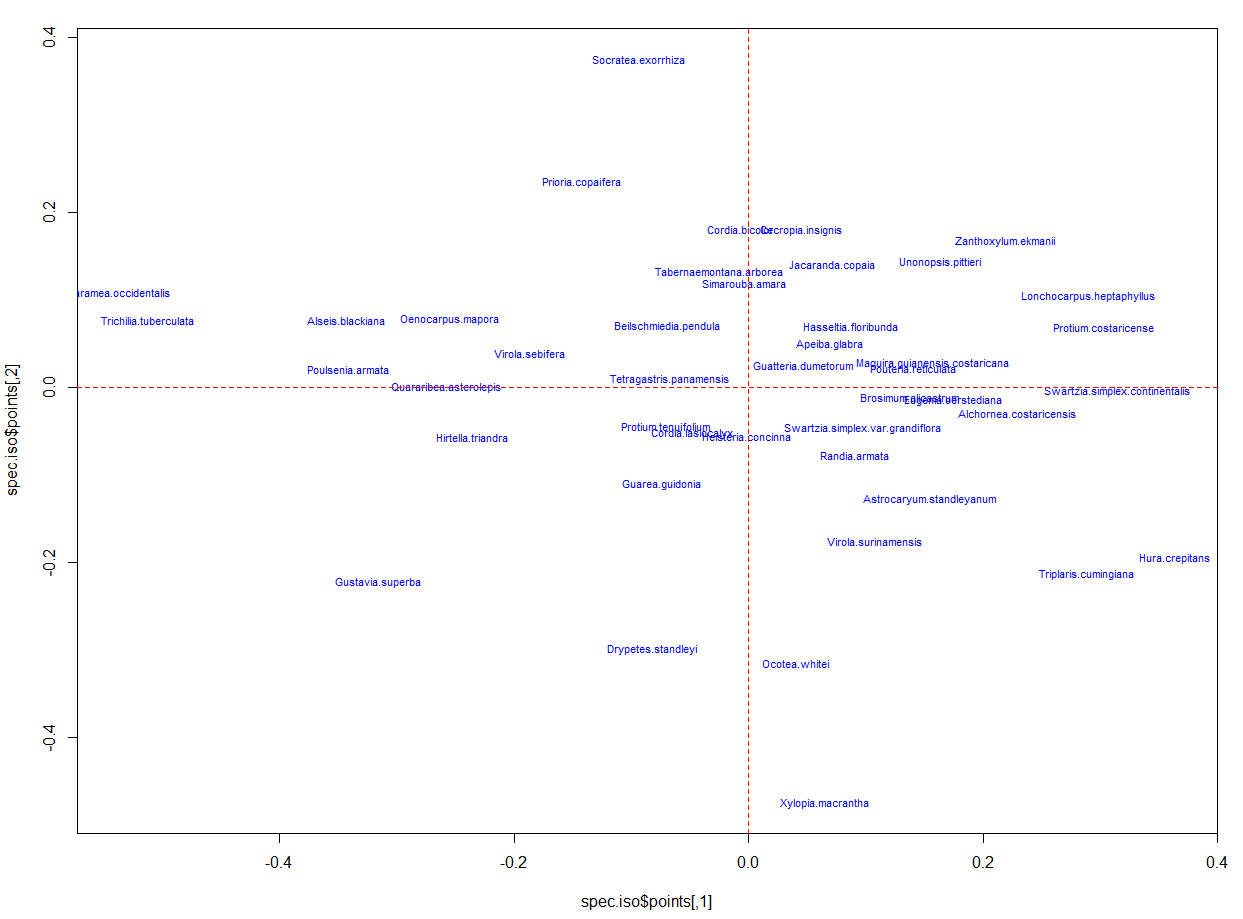
> spec.iso = isoMDS(spec.dist,k=2)

> plot(spec.iso$points,type=”n”)

> text(spec.iso$points,as.character(row.names(tspec100.mat)),cex=.7)

> abline(h=0,v=0,lty=2,col=”red”,lwd=2)

The plot on the following page shows how the species relationships project into two-dimensions. We could also consider dimensional representations for both sites and species.



> library(vegan)

> site.meta = metaMDS(spec100.mat,k=2)

Notice that the function metaMDS takes the table of counts as an argument, not the distances. By default metaMDS will use the Bray-Curtis metric to compute the distance matrix from the supplied counts and then perform non-metric MDS to find the lower dimensional representation.

> site.meta = metaMDS(spec100.mat,k=2)

Square root transformation

Wisconsin double standardization

Run 0 stress 0.1545235

Run 1 stress 0.1545234

... New best solution

Run 16 stress 0.1574869

Run 17 stress 0.1549544

... Procrustes: rmse 0.01247892 max resid 0.07137517

Run 18 stress 0.1704934

Run 19 stress 0.1574863

Run 20 stress 0.1704938

\*\*\* Solution reached

> plot(site.meta,type="n")

> text(site.meta,display="sites",as.character(Barro$Site))

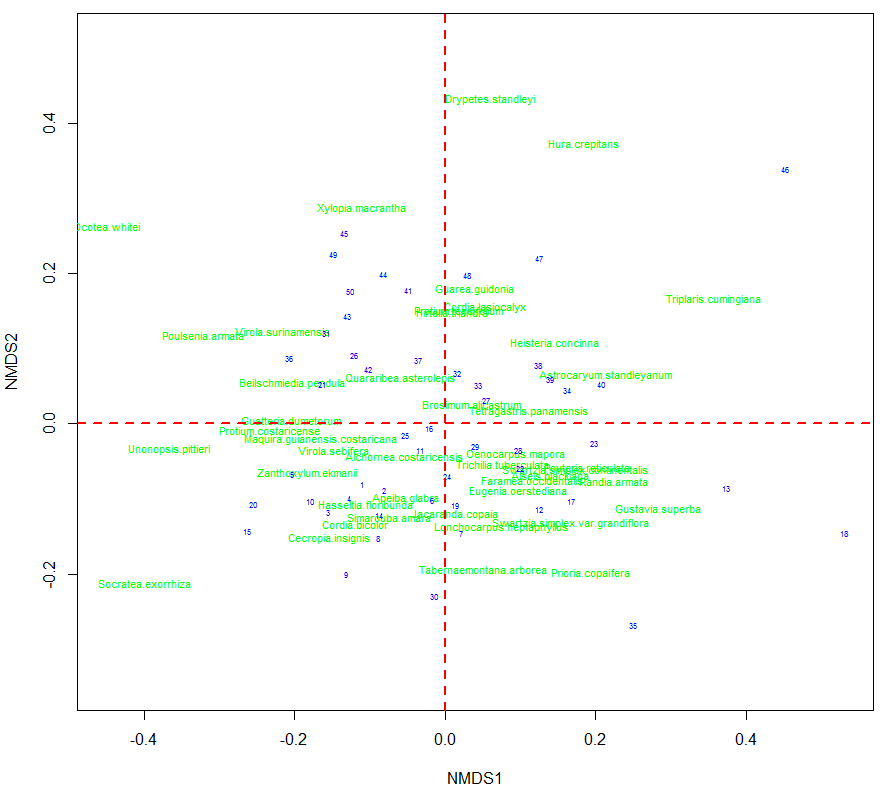
> text(site.meta,display="species",as.character(names(spec100.mat)))

```  
> plot(site.meta,display=c(“sites”,”species”,type="n")

> text(site.meta,display="species",as.character(names(spec100.mat)),cex=.7,col="green")

> text(site.meta,display="sites",as.character(Barro$Site),cex=.5,col="blue")

> abline(h=0,v=0,lty=2,col="red",lwd=2)



This shows both the site and species relationships in one plot!

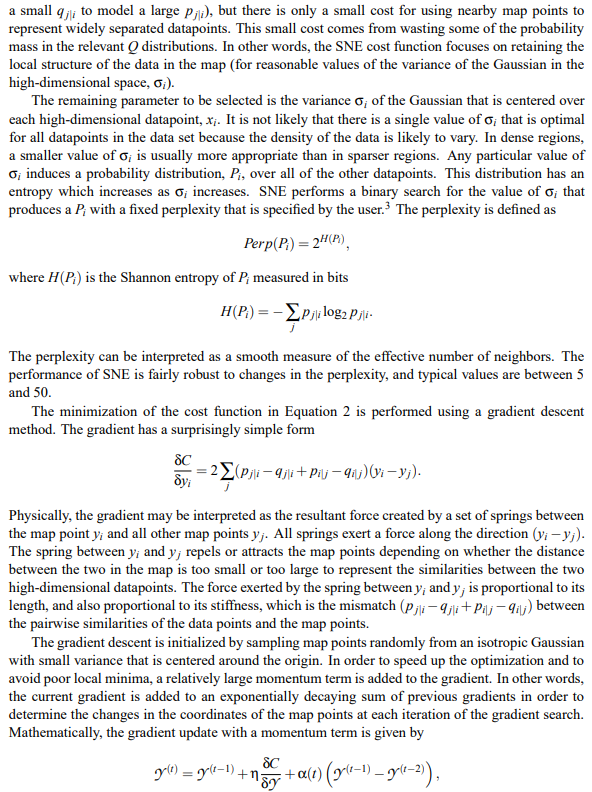
Another method for examining relationships between categorical variables is multiple correspondence analysis, which we will cover in Chapter 6.

**Stochastic Neighborhood Embedding (SNE and t-SNE)**

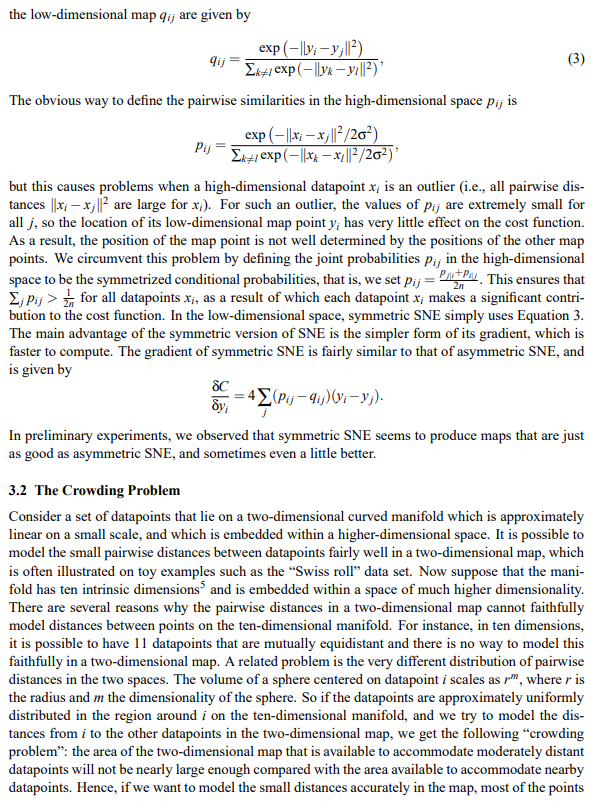
A relatively recent tool for visualizing high dimensional data in a lower dimensional space (k = 2 or 3) is Stochastic Neighbor Embedding or SNE (Hinton and Roweis, 2002) and t-SNE (van der Maaten and Hinton, 2008). Below we will present the basic idea each approach and consider some examples using some the data sets used in earlier examples in this chapter.

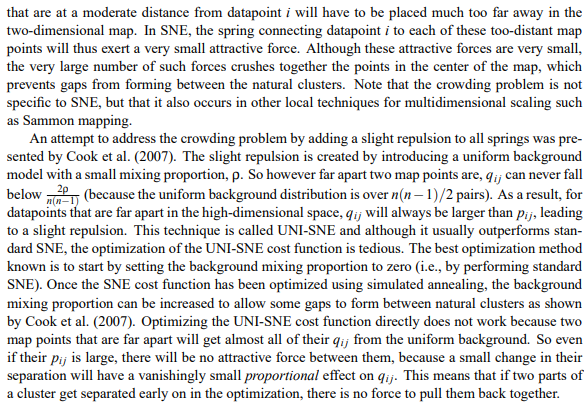
**SNE – taken from Visualizing Data using t-SNE (van der Maaten and Hinton, 2008)**

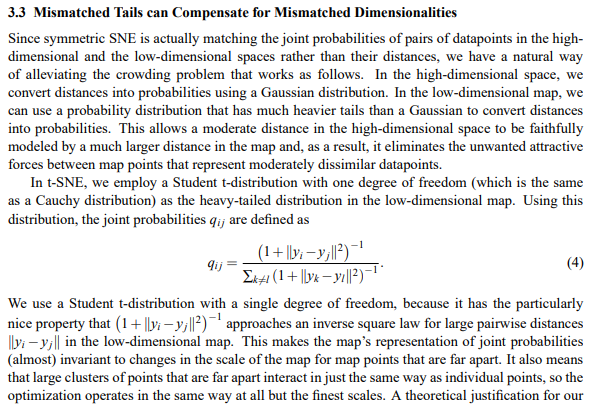
****

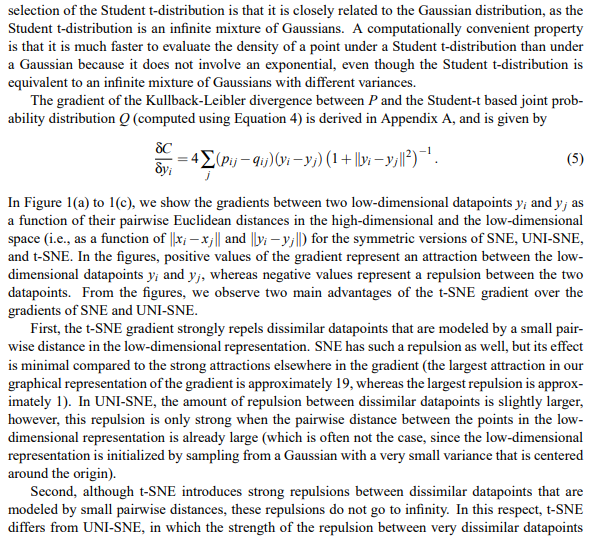


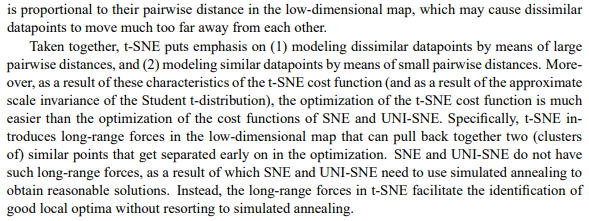




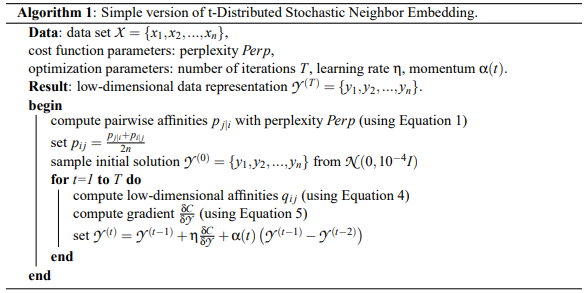








**t-SNE Algorithm**



**Examples of t-SNE**

The R package Rtsne contains functions to perform t-SNE.

**Olives Oils**  
> names(Olives)

[1] "Region.name" "Area.name" "Region" "Area" "palmitic" "palmitoleic"

[7] "strearic" "oleic" "linoleic" "eicosanoic" "linolenic" "eicosenoic"

> Olive.mat = Olives[,5:11]

> sOlive.mat = scale(Olive.mat)

> olives.tsne = Rtsne(sOlive.mat)

Error in Rtsne.default(sOlive.mat) :

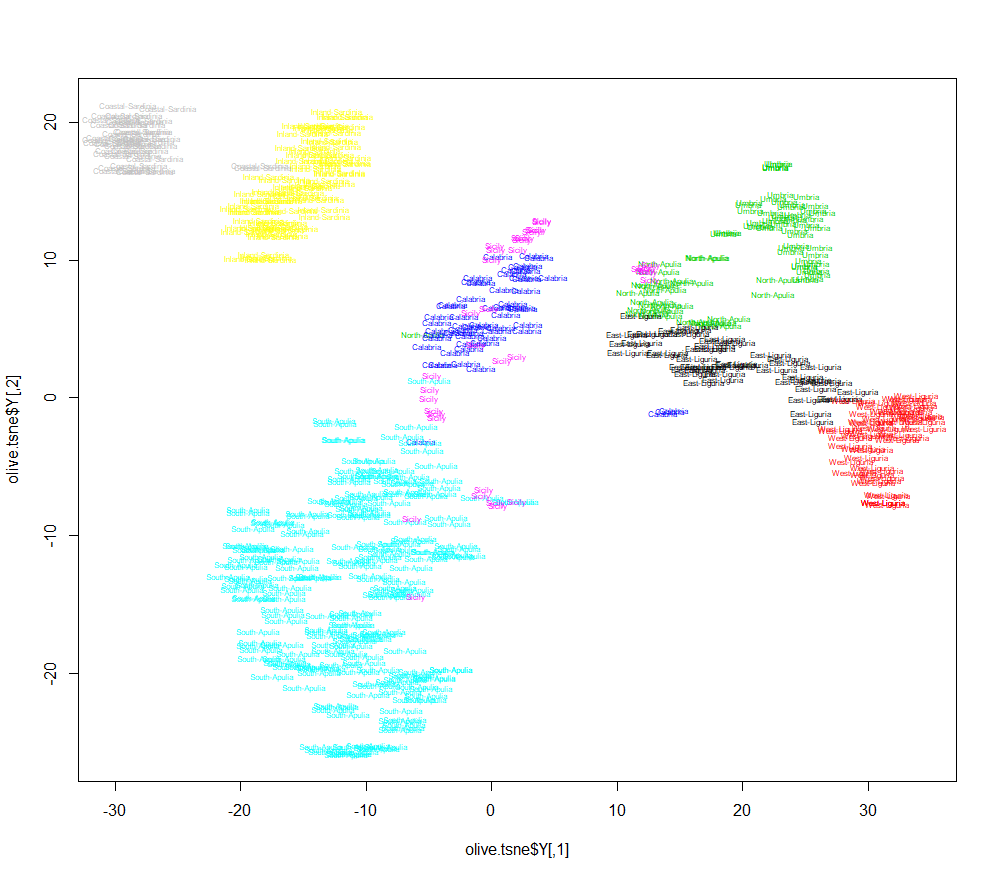
Remove duplicates before running TSNE.

> sOlive.mat = scale(Olive.mat[-451,])

> olive.tsne = Rtsne(sOlive.mat,perplexity=20)

> plot(olive.tsne$Y,type="n")

> text(olive.tsne$Y,labels=as.character(Olives$Area.name[-451]),cex=.5,col=as.numeric(Olives$Area[-451])+2)



**Mushrooms**

Using the same pre-processing as in the MDS examples above, for t-SNE we have.

> mush.dist = dist(mush.dummy,method="Jaccard")

> summary(mush.dist)

Min. 1st Qu. Median Mean 3rd Qu. Max.

0.09091 0.60000 0.68750 0.66458 0.76471 0.92308

> mush.tsne = Rtsne(mush.dist,is\_distance=TRUE)

> plot(mush.tsne$Y,type="n")

> text(mush.tsne$Y,as.character(Mushrooms.train$Poisonous),col=as.numeric(Mushrooms.train$Poisonous),cex=.7)

