Getting the most sig. differentiated genes between tissue

The way I am approaching this is to get each list of sig DE genes between all DE anaylsis between tissue in WT and *tf2*, keep only the unique and see how many genes there are.

Required Libraries

```
library(ggplot2)
library(reshape)
library(kohonen)
library(RColorBrewer)
```

Read in files and format data from raw files

Create a list of all DE analysis:

Count Data

```
#Read in count data
countData <- read.csv("../data/normalized_read_count.csv")
#Melt count data
countData <- melt(countData)
colnames(countData) <- c("gene", "sample", "count")

#set genotype

countData$genotype <- ifelse(grepl("wt", countData$sample, ignore.case = T), "wt",
         ifelse(grepl("tf2", countData$sample, ignore.case = T), "tf2", "unknown"))

#set tissue

countData$tissue <- ifelse(grepl("other", countData$sample, ignore.case = T), "other",
         ifelse(grepl("mbr", countData$sample, ignore.case = T), "mbr", "unknown"))

#Set Region
countData$region <- ifelse(grepl("a", countData$sample, ignore.case = T), "A",
         ifelse(grepl("c", countData$sample, ignore.case = T), "C", "B"))

#Set type

countData$type <- paste(countData$region, countData$tissue,  sep = "")
head(countData)

#Subset by Genotype, since we will not be looking at tf2 at this stage
countData <- subset(countData, genotype == "wt")
```

Most Significantly DE genes

```
#Read in each list of DE expressed genes
wtaother_wtcother  <- read.table("../data/allSigGenes/wtaother_wtcother_DE_sig.txt", header = TRUE, fil
wtambr_wtaother <- read.table("../data/allSigGenes/wtambr_wtaother_DE_sig.txt", header = TRUE, fill = T
wtambr_wtbmbr <- read.table("../data/allSigGenes/wtambr_wtbmbr_DE_sig.txt", header = TRUE, fill = TRUE)
wtambr_wtcmbr <- read.table("../data/allSigGenes/wtambr_wtcmbr_DE_sig.txt", header = TRUE, fill = TRUE)
wtaother_wtbother <- read.table("../data/allSigGenes/wtaother_wtbother_DE_sig.txt", header = TRUE, fill
wtbmbr_wtbother <- read.table("../data/allSigGenes/wtbmbr_wtbother_DE_sig.txt", header = TRUE, fill = T
wtbmbr_wtcmbr <- read.table("../data/allSigGenes/wtbmbr_wtcmbr_DE_sig.txt", header = TRUE, fill = TRUE)
wtbother_wtcother <- read.table("../data/allSigGenes/wtbother_wtcother_DE_sig.txt", header = TRUE, fill
wtcmbr_wtcother <- read.table("../data/allSigGenes/wtcmbr_wtcother_DE_sig.txt", header = TRUE, fill = T

#merge them together
allGenes <- rbind(wtaother_wtcother, wtambr_wtaother, wtambr_wtbmbr, wtambr_wtcmbr, wtaother_wtbother,w

dim(allGenes)
head(allGenes)

#recieve just the list
allGenesITAG <- allGenes[,1]
length(allGenesITAG)
#Remove duplicates
allGenesITAG <- unique(allGenesITAG)

#make an empty table to hold all the genes
allGeneList <- data.frame(t(rep(NA,7)))
colnames(allGeneList) <- c("type", "genotype", "N", "mean", "sd", "se", "gene")
allGeneList <- allGeneList[-1,] #remove first row
head(allGeneList)
```

Loop together all relevent gene information.

```
for(GENE in allGenesITAG) {

  if(length(grep(GENE, countData$gene)) < 1){ #this is just making sure that the list of sig genes
    next;
    }

  geneData <- subset(countData, grepl(GENE, countData$gene))

  sumGraph <- ddply(geneData, c("type", "genotype"), summarise,
                N    = length(count),
                mean = mean(count),
                sd   = sd(count),
                se   = sd / sqrt(N))

  sumGraph$gene <- GENE

allGeneList  <- rbind(allGeneList, sumGraph) #bind together all the new rows per loop.
```

```
}
dim(allGeneList)
head(allGeneList)
write.table(allGeneList, file = "../data/allGeneList.csv", sep = ",")
```

Self Organizing Maps

The goal of this analysis is to find genes that have co-expression patterns of

1. most differentiated genes between tissue.

1.pca.R

First read in file that came from mostSigDEgenes.Rmd. This is a list of genes from all DE analysis in WT. They were all cancatenated, then duplicate genes were removed. In addition the mean was calculated from the replicates of each type.

The first step is to get it into the right format. First column being the genes, while the subsequent columns are the different libraries (type).

```
mostDEgenes <- read.csv("../data/allGeneList.csv")
head(mostDEgenes)
```

```
##     type genotype N    mean      sd      se               gene
## 2   Ambr       wt 3  17.193  21.125 12.1965 Solyc00g005070.1.1
## 3 Aother       wt 5   4.524   1.835  0.8207 Solyc00g005070.1.1
## 4   Bmbr       wt 4  12.646  18.656  9.3278 Solyc00g005070.1.1
## 5 Bother       wt 4   3.462   1.789  0.8947 Solyc00g005070.1.1
## 6   Cmbr       wt 6   4.181   2.819  1.1507 Solyc00g005070.1.1
## 7 Cother       wt 3 105.967 168.648 97.3689 Solyc00g005070.1.1
```

```
mostDEgenes <- mostDEgenes[c(7, 1, 4)] #keep only needed columns (gene, type, mean)

#Change from long to wide data format
mostDEgene.long <- cast(mostDEgenes, gene ~ type, value.var = mean, fun.aggregate = "mean")  #why did I
```

```
## Using mean as value column.  Use the value argument to cast to override this choice
```

```
mostDEgene.long <- as.data.frame(mostDEgene.long) #transformation.
names(mostDEgene.long)
```

```
## [1] "gene"   "Ambr"   "Aother" "Bmbr"   "Bother" "Cmbr"   "Cother"
```

```
scale_data <- as.matrix(t(scale(t(mostDEgene.long[c(2:7)]))))

#Principle Component Analysis
pca <- prcomp(scale_data, scale=TRUE)

summary(pca)
```

```
## Importance of components:
##                          PC1   PC2   PC3   PC4   PC5      PC6
## Standard deviation     1.399 1.116 1.058 0.962 0.868 9.21e-16
## Proportion of Variance 0.326 0.208 0.186 0.154 0.126 0.00e+00
## Cumulative Proportion  0.326 0.534 0.720 0.874 1.000 1.00e+00
```

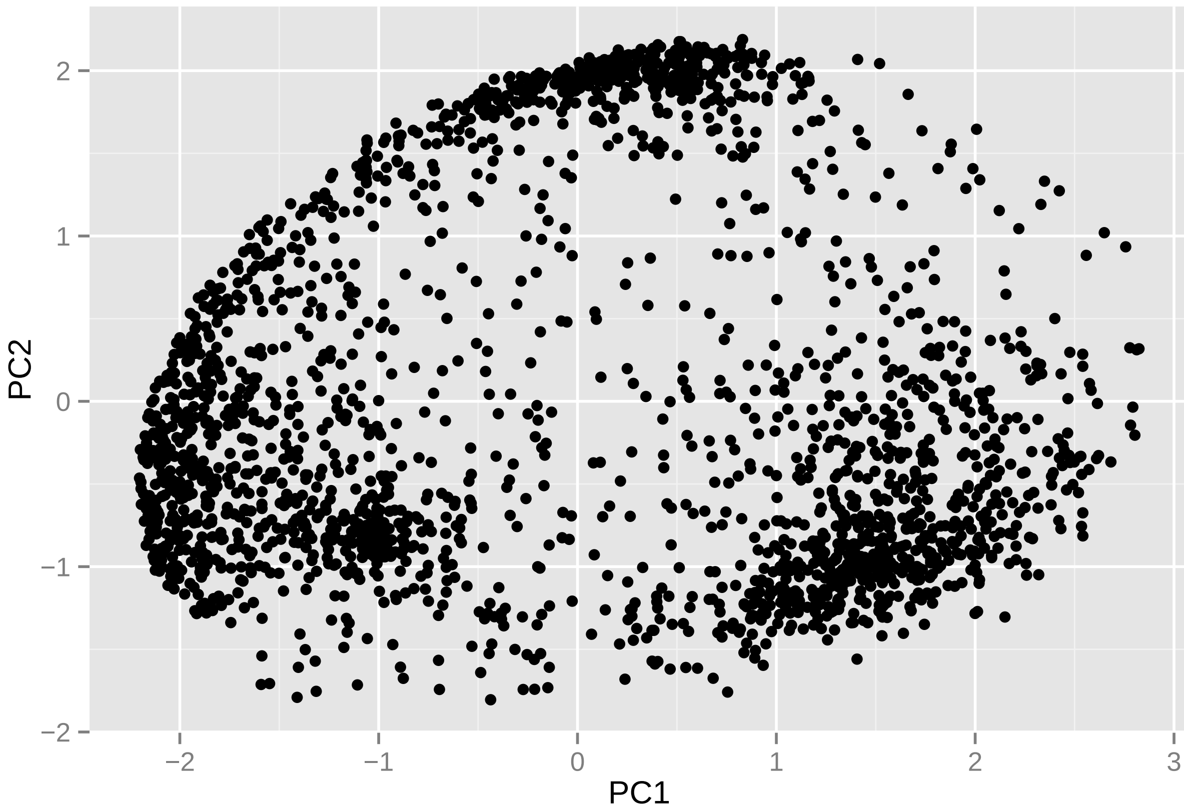
```
pca.scores <- data.frame(pca$x)

data.val <- cbind(mostDEgene.long, scale_data, pca.scores)
head(data.val)
```

```
##                 gene      Ambr   Aother     Bmbr  Bother    Cmbr  Cother
## 1 Solyc00g005070.1.1   17.1934   4.5236  12.6456   3.462   4.181 105.967
## 2 Solyc00g005080.1.1   16.0215  10.1277   7.5622   8.496   8.413  25.029
## 3 Solyc00g005840.2.1   19.7458  14.0743  11.4830  83.513  15.811  13.981
## 4 Solyc00g005870.1.1    0.1336   0.6414   3.1241   2.779   1.780  12.462
## 5 Solyc00g006470.1.1 2455.5111 605.8620 108.0947 360.482 499.448 390.760
## 6 Solyc00g006670.2.1   66.9760   5.9947   0.6023   7.071  11.065   4.678
##      Ambr  Aother     Bmbr  Bother    Cmbr  Cother     PC1     PC2
## 1 -0.1857 -0.5008 -0.29882 -0.5272 -0.5093  2.0219  0.4124  1.9721
## 2  0.5010 -0.3641 -0.74069 -0.6037 -0.6158  1.8232  0.4094  1.4972
## 3 -0.2381 -0.4399 -0.53216  2.0315 -0.3781 -0.4433 -0.9687 -0.9406
## 4 -0.7372 -0.6256 -0.07972 -0.1557 -0.3751  1.9733  0.2002  2.0773
## 5  2.0024 -0.1524 -0.73231 -0.4383 -0.2764 -0.4030  1.1164 -1.0607
## 6  2.0226 -0.4000 -0.61428 -0.3573 -0.1986 -0.4524  1.3269 -1.0424
##        PC3     PC4     PC5        PC6
## 1  0.62321 0.23135  0.1379 -1.110e-15
## 2  1.14022 0.75216 -0.1592 -1.332e-15
## 3 -0.45945 0.47067  1.6559  1.499e-15
## 4  0.08514 0.01114  0.5230 -9.992e-16
## 5  0.54497 0.93811 -0.4748 -5.551e-16
## 6  0.32412 0.97528 -0.2106 -1.665e-16
```

Visualizing the PCA

Looks to be 2 maybe three major clusters.

```
p <- ggplot(data.val, aes(PC1, PC2))
p + geom_point()
```



1. Self Organizing Map - (6,6) Large

The size of the map is something that may cause differences in the genes that are clustered. Using a small map size (3,2), I found they cluster in according to tissue type. This makes the interpretation of the results pretty straight forward. My only worry is that the map might not be large enough, considering [1], suggests that you size of the map based on count distribution, the goal being an even distribution, with no "peak" counts in any one cluster while also having no empty clusters.

The only way to see how this is affects what we see is to compare the clusters of the small (3,2) and large map (6,6).

```
names(data.val)
```

```
##  [1] "gene"    "Ambr"    "Aother" "Bmbr"    "Bother" "Cmbr"    "Cother"
##  [8] "Ambr"    "Aother" "Bmbr"    "Bother" "Cmbr"    "Cother" "PC1"
## [15] "PC2"     "PC3"     "PC4"     "PC5"     "PC6"
```

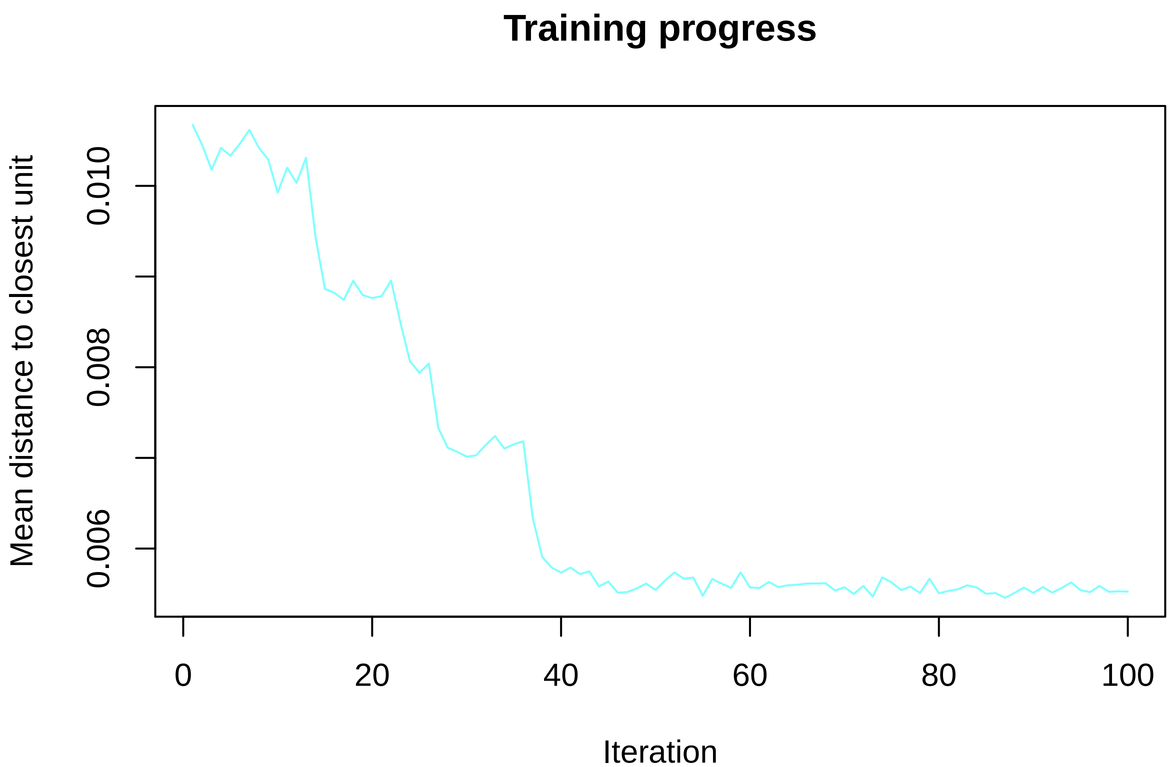
```
som.data <- as.matrix(data.val[,c(8:13)])  #subset only the scaled gene expression values

set.seed(2)

som <- som(data=som.data, somgrid(6,6,"hexagonal")) # This is where you change the size of the map
summary(som)
```

```
## som map of size 6x6 with a hexagonal topology.
## Training data included; dimension is 2249 by 6
## Mean distance to the closest unit in the map: 0.4131
```

Training Plot ("changes") - Large

This shows a hundred iterations. Training decreases with iterations and plateaus at around 40 iterations. Ideally you want the the training to reach a minimum plateau. In the example online, the decrease to this plateau happens slowly with a slow decline to the minimum plateau. I should look into what this sudden drop means.

```
plot(som, type ="changes")
```

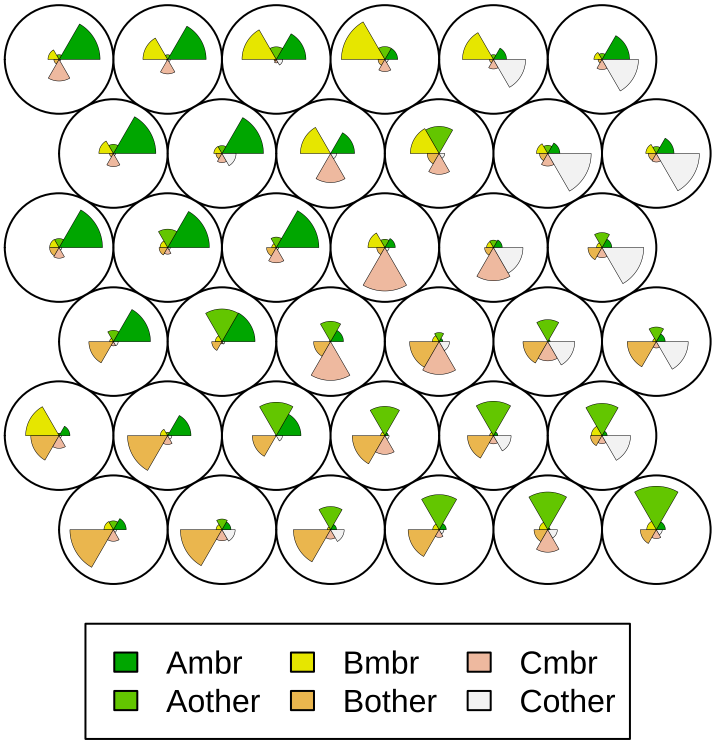


Code Plot - Large

The the code plot shows each cluster and the node wieght vectors or "codes" associated with each node. These are made up of the original normalized values of the original values used to generate the map. You should see patterns of clustering.

The fan chart in the center of the clusters reveals the characteristics that define how the genes were clustered into each particular cluster. For instance if one cluster has only one large fan piece, say for Bother, this is telling us that most of the genes in this cluster were grouped because of similar normalized gene count value of the Bother region. We do not know the degree, it could mean all these genes are up-regulated or down-regulated in the Bother region, but we do not know which at this point.

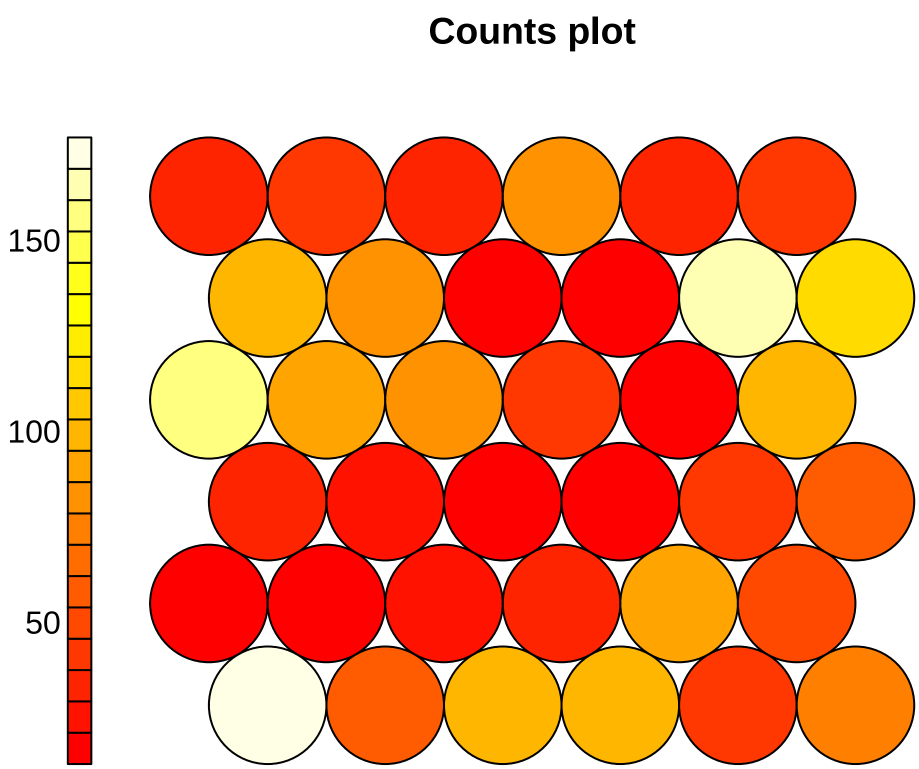
```
plot(som, type = "codes")
```



Count Plot - Large

This tells you how many genes are in each of the clusters. The count plot can be used as a quality check. Ideally you want a uniform distribution. If there are some peaks in certain areas, this means you should likely increase the map size. If you have empty nodes you should decrease the map size [1].

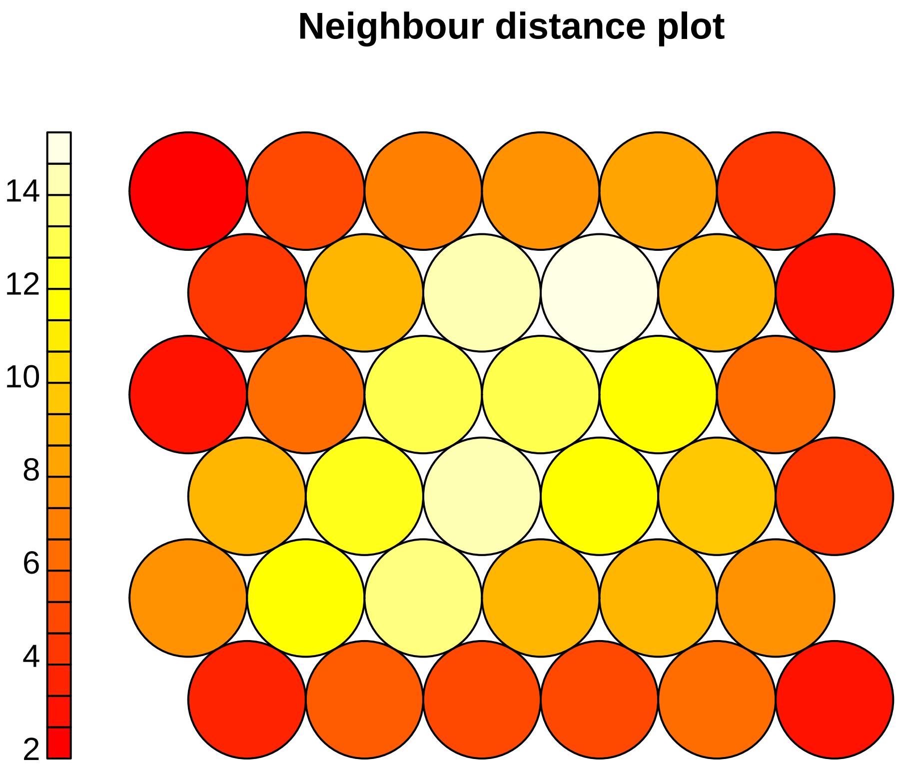
```
plot(som, type = "counts")
```



Distance Neighbour Plot - Large

This is sometimes called the "U-Matrix", it can help identify further clustering. Areas of low neighbour distance indicate groups of nodes that are similar and the further apart nodes indicate natural "borders" in the map.

```
plot(som, type="dist.neighbours")
```



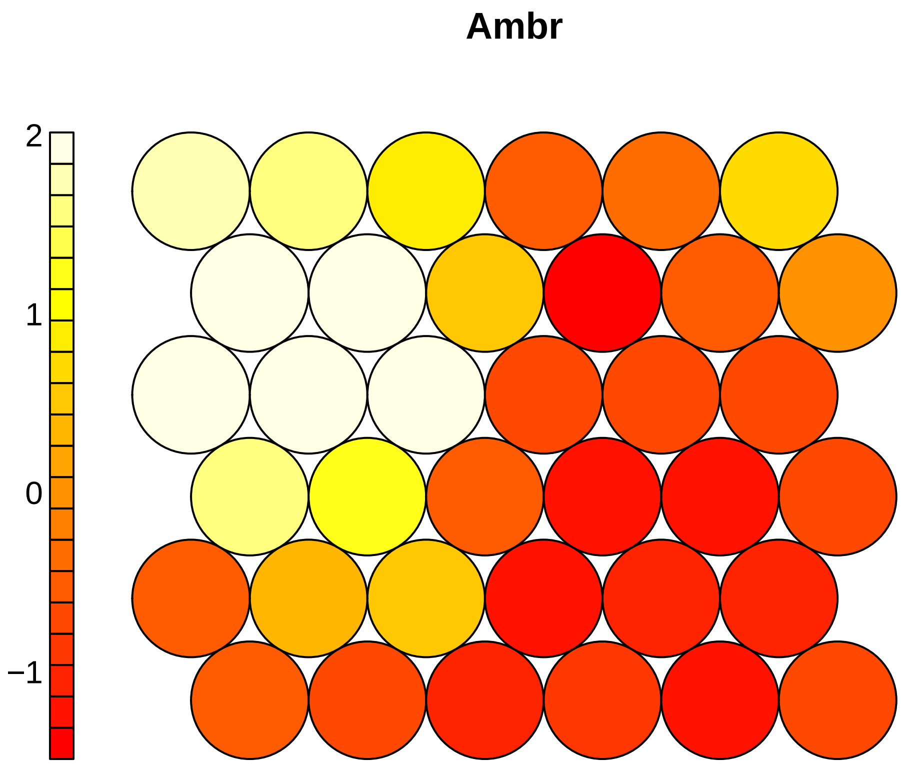
Heatmaps - large

This shows the distribution of each type

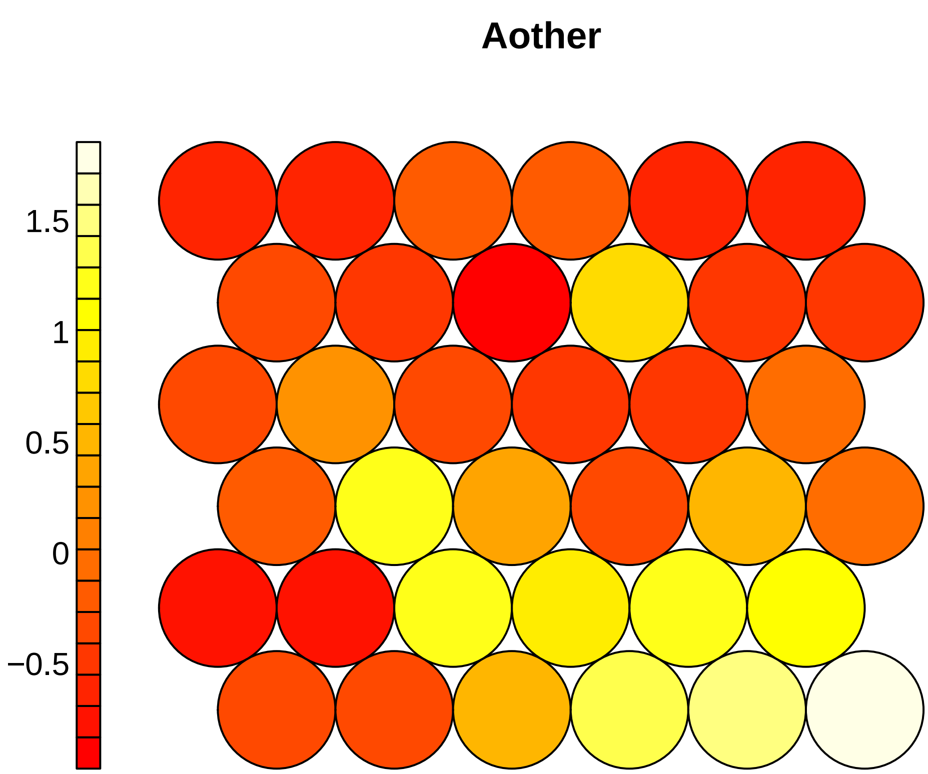
```r
head(som$codes)
```

```
##          Ambr  Aother    Bmbr   Bother    Cmbr   Cother
## [1,] -0.4519 -0.3986 -0.3959  1.99972 -0.3451 -0.40820
## [2,] -0.7647 -0.2780 -0.6316  1.86810 -0.3824  0.18861
## [3,] -1.0013  0.5359 -0.8077  1.55468 -0.4964  0.21486
## [4,] -0.9120  1.3120 -0.5958  1.11797 -0.6098 -0.31229
## [5,] -1.2945  1.4819 -0.4572 -0.10471  0.3999 -0.02544
## [6,] -0.7562  1.8570 -0.3837  0.05502 -0.5277 -0.24439
```

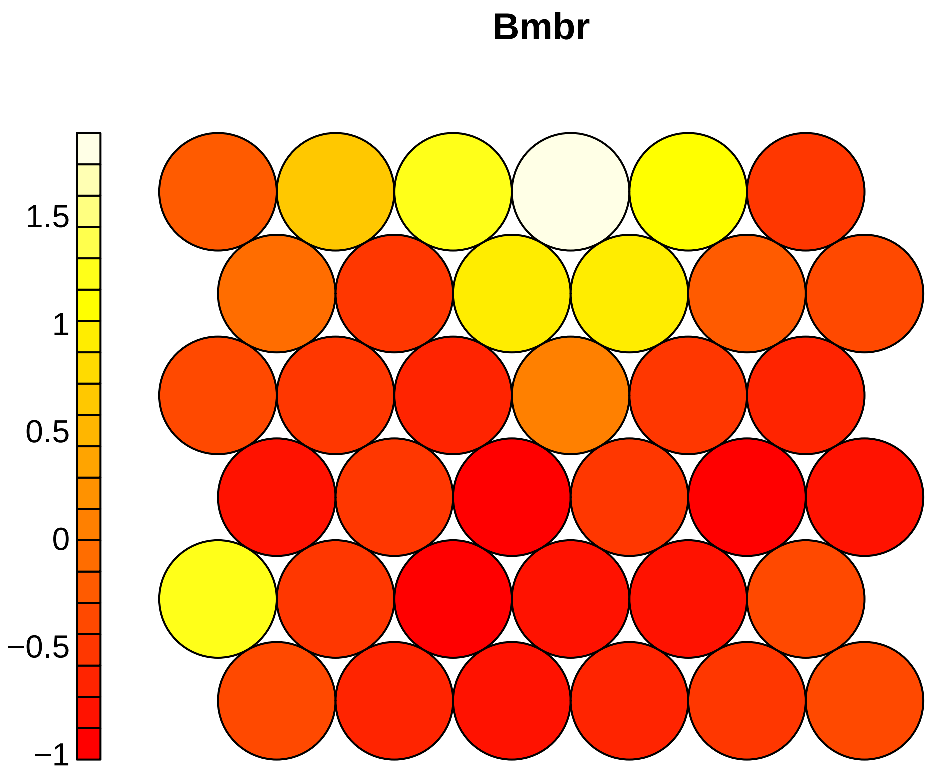
```r
som$data <- data.frame(som$data) #changed to dataframe to extract column names easier.

#This is just a loop that plots the distrinution of each tissue type across the map.
for (i in 1:6){
  plot(som, type = "property", property = som$codes[,i], main=names(som$data)[i])
  print(plot)
  }
```



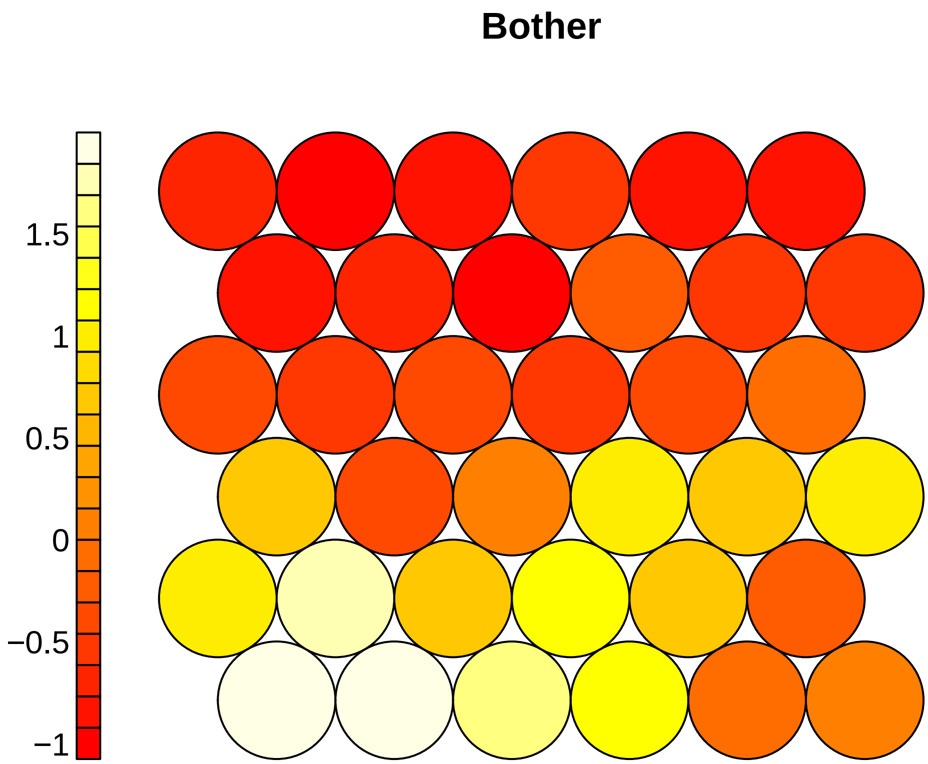

```
## function (x, y, ...)
## UseMethod("plot")
## <bytecode: 0x7fff01192b88>
## <environment: namespace:graphics>
```



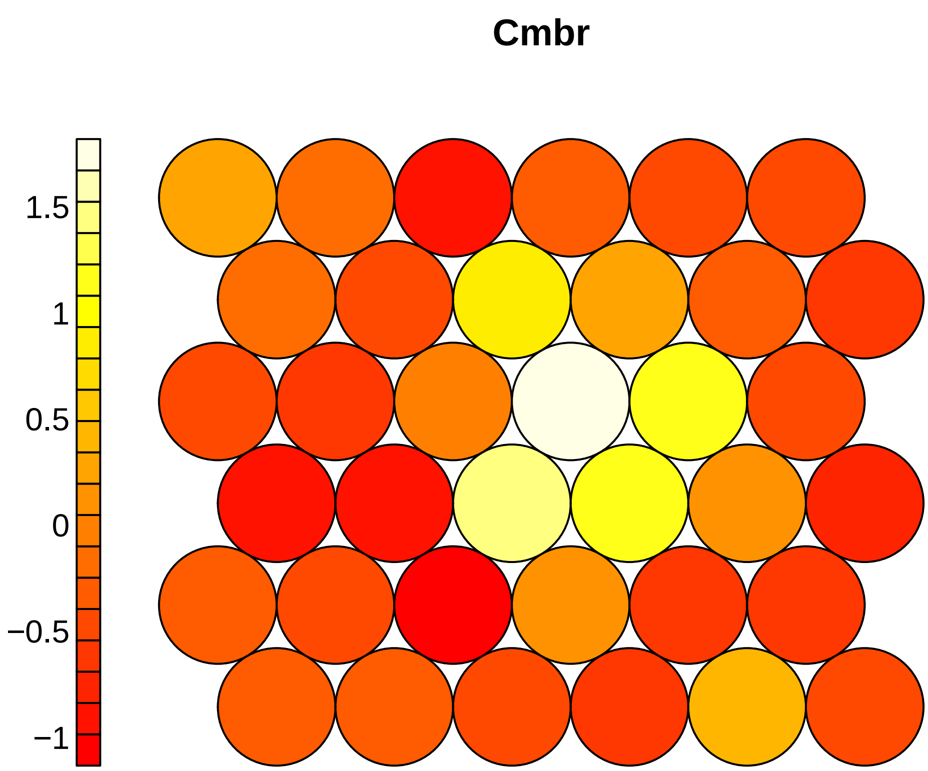
```
## function (x, y, ...)
## UseMethod("plot")
## <bytecode: 0x7fff01192b88>
## <environment: namespace:graphics>
```



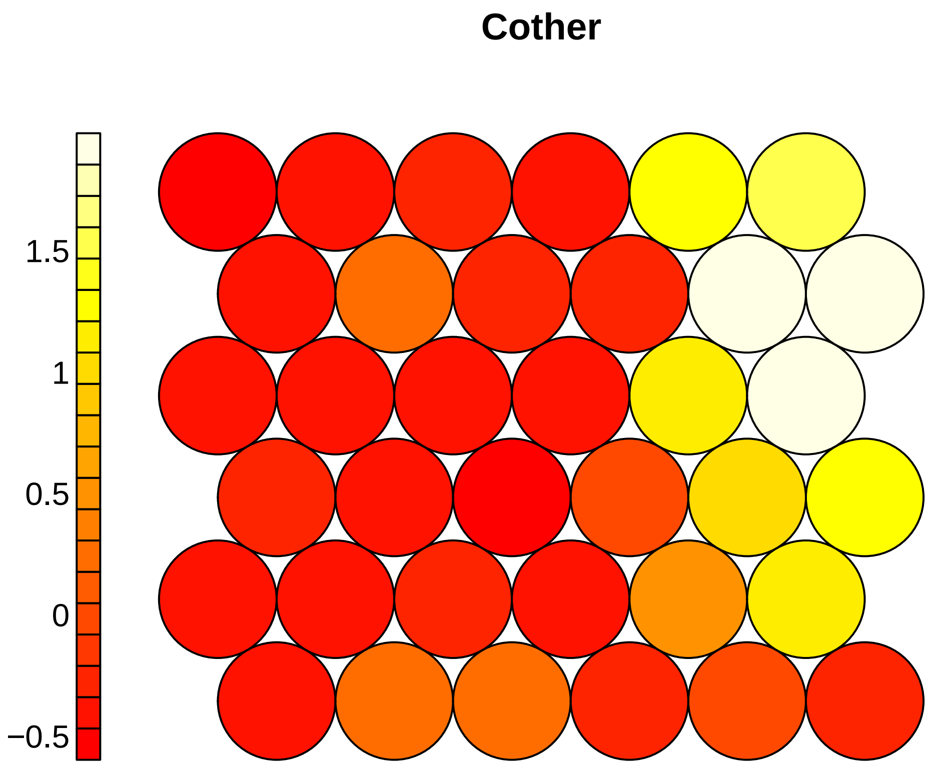

```
## function (x, y, ...)
## UseMethod("plot")
## <bytecode: 0x7fff01192b88>
## <environment: namespace:graphics>
```




```
## function (x, y, ...)
## UseMethod("plot")
## <bytecode: 0x7fff01192b88>
## <environment: namespace:graphics>
```



```
## function (x, y, ...)
## UseMethod("plot")
## <bytecode: 0x7fff01192b88>
## <environment: namespace:graphics>
```




```
## function (x, y, ...)
## UseMethod("plot")
## <bytecode: 0x7fff01192b88>
## <environment: namespace:graphics>
```

Clustering Plot - Large

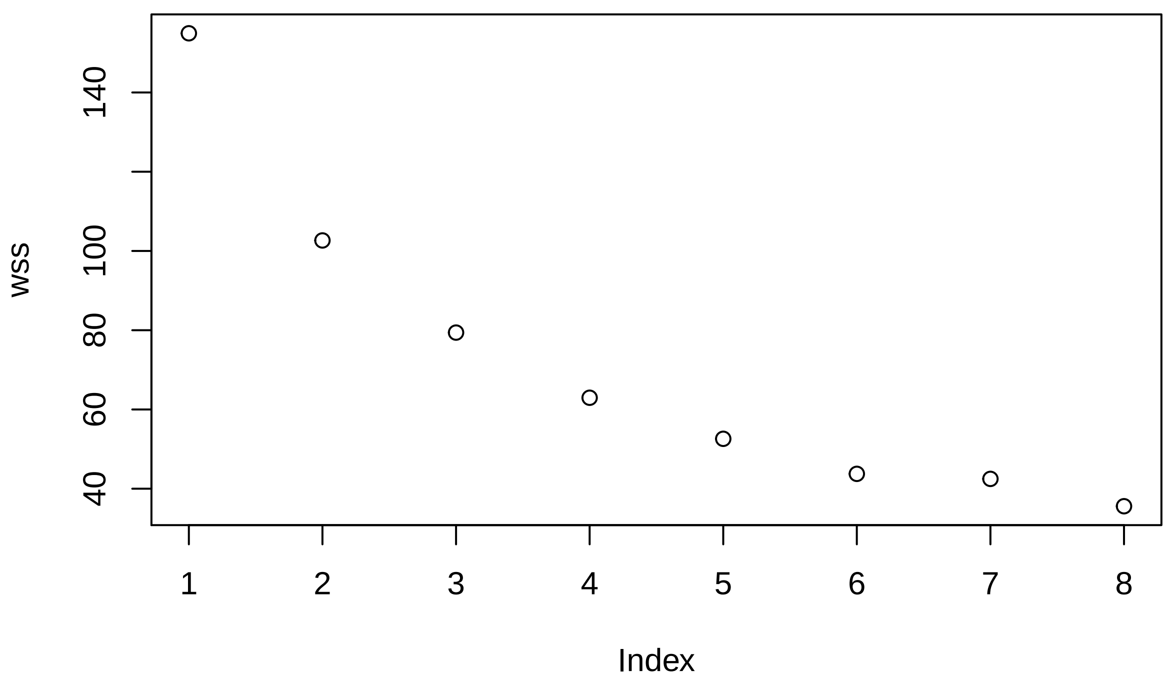
This groups clusters based on similar "metrics". The advice given from [1], suggests that the heatmap should be used to view the overall "story" of the map. Essentiatlly you are taking all the tissue types into account and viewing it all on one heat map.

```
#"estimate of the number of clusters that would be suitable can be ascertained using a kmeans algorithm

#What the hell is going on here?

mydata <- som$codes
head(mydata)
```

```
##         Ambr  Aother    Bmbr   Bother    Cmbr   Cother
## [1,] -0.4519 -0.3986 -0.3959  1.99972 -0.3451 -0.40820
## [2,] -0.7647 -0.2780 -0.6316  1.86810 -0.3824  0.18861
## [3,] -1.0013  0.5359 -0.8077  1.55468 -0.4964  0.21486
```

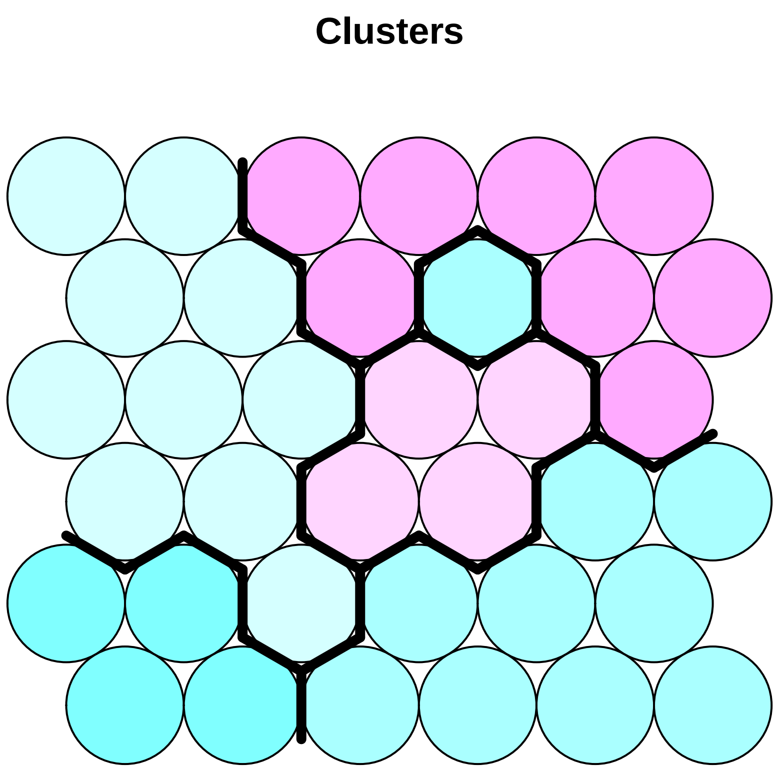
```
## [4,] -0.9120  1.3120 -0.5958  1.11797 -0.6098 -0.31229
## [5,] -1.2945  1.4819 -0.4572 -0.10471  0.3999 -0.02544
## [6,] -0.7562  1.8570 -0.3837  0.05502 -0.5277 -0.24439
```

```
wss <- (nrow(mydata)-1)*sum(apply(mydata,2,var))

for (i in 1:8) {
  wss[i] <- sum(kmeans(mydata, centers=i)$withinss)
}

plot(wss)
```



This is setting the larger clusters that incorporate multiple clusters.

```
## use hierarchical clustering to cluster the codebook vectors
som_cluster <- cutree(hclust(dist(som$codes)), 5)
# plot these results:
plot(som, type="mapping", bgcol = som_cluster, main = "Clusters")
add.cluster.boundaries(som, som_cluster)
```




```
# I want to attach the hierchal cluster to the larger dataset data.val.
som_clusterKey <- data.frame(som_cluster)
som_clusterKey$unit.classif <- c(1:36)

data.val <- cbind(data.val,som$unit.classif,som$distances)

#Merge data.val with som_clusterKey
##change data.val to match som_cluster key
names(data.val)[20] <- "unit.classif"

data.val <- merge(data.val, som_clusterKey, by.x = "unit.classif" ) #ignore warning, this is what you w
```

```
## Warning: column names 'Ambr', 'Aother', 'Bmbr', 'Bother', 'Cmbr', 'Cother'
## are duplicated in the result
```

```
write.table(data.val, file="../data/analysis1.som.data.txt")
#Make sure that there is just one of each value som$unit.classif and distances column.
names(data.val)
```

```
##  [1] "unit.classif"  "gene"          "Ambr"          "Aother"
##  [5] "Bmbr"          "Bother"        "Cmbr"          "Cother"
##  [9] "Ambr"          "Aother"        "Bmbr"          "Bother"
## [13] "Cmbr"          "Cother"        "PC1"           "PC2"
## [17] "PC3"           "PC4"           "PC5"           "PC6"
## [21] "som$distances" "som_cluster"
```

Other Visualization - Large

Visualize by Cluster

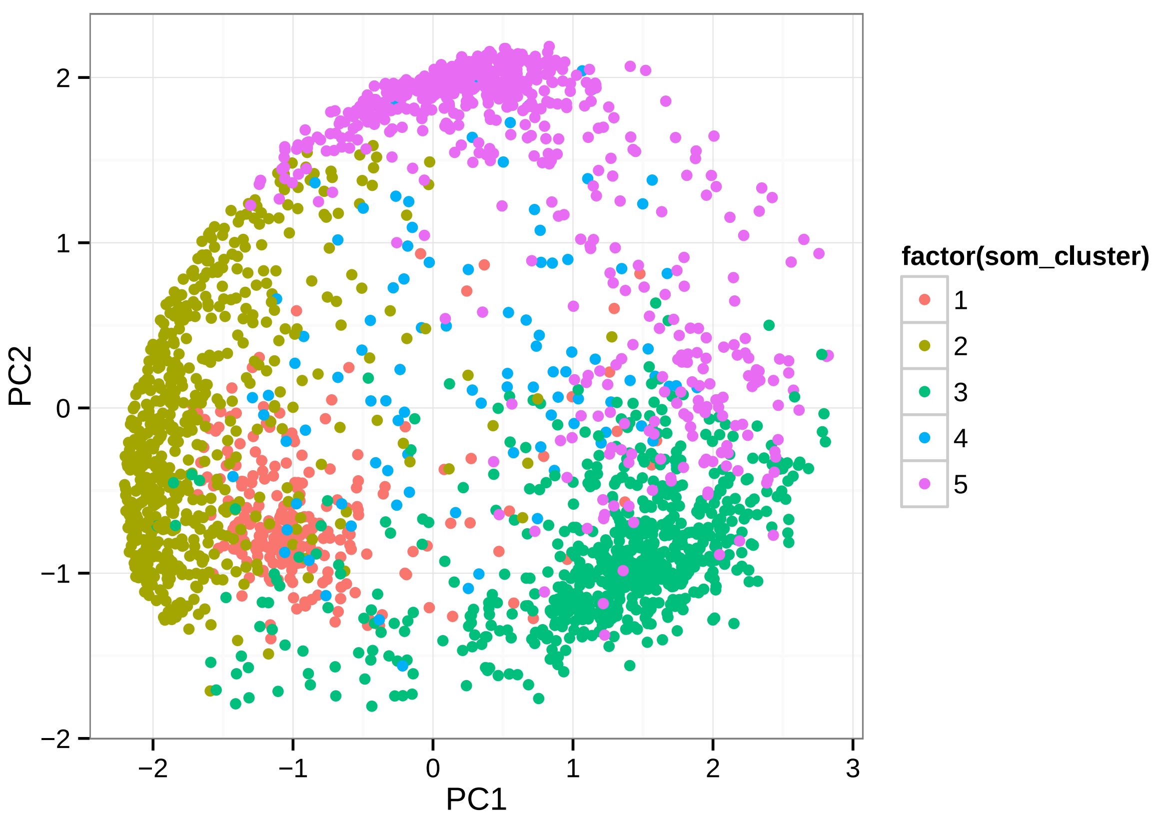
```
plot.data <- read.table("../data/analysis1.som.data.txt",header=TRUE)
names(plot.data)
```

```
##  [1] "unit.classif"  "gene"          "Ambr"          "Aother"
##  [5] "Bmbr"          "Bother"        "Cmbr"          "Cother"
##  [9] "Ambr.1"        "Aother.1"      "Bmbr.1"        "Bother.1"
## [13] "Cmbr.1"        "Cother.1"      "PC1"           "PC2"
## [17] "PC3"           "PC4"           "PC5"           "PC6"
## [21] "som.distances" "som_cluster"
```

```
dim(plot.data)
```

```
## [1] 2249   22
```

```
p <- ggplot(plot.data, aes(PC1, PC2, colour=factor(som_cluster))) #notice I am using som_cluster and no
p + geom_point() + theme_bw()
```



Visualize by individual clusters - Large

What the hell is going on here for the large cluster?

```r
sub_cluster <- subset(plot.data, som_cluster =="5") #Again here I am interested in the clusters made up

sub_data <- sub_cluster[,8:13] # just the sample types

names(sub_data)
```

```
## [1] "Cother"   "Ambr.1"   "Aother.1" "Bmbr.1"   "Bother.1" "Cmbr.1"
```

```r
head(sub_data)
```

```
##      Cother  Ambr.1  Aother.1  Bmbr.1 Bother.1  Cmbr.1
## 1376 53.931 -0.8934  0.354033 -0.5508 -0.11974 -0.6253
## 1377 13.976 -1.0493 -0.001158 -0.6414  0.04355 -0.2073
## 1378 12.109 -0.7117 -0.323637 -0.6846  0.06769 -0.3032
## 1379 31.107 -0.5741  0.092137 -0.6657 -0.28025 -0.5352
## 1380  9.625 -0.6829  0.327024 -0.6916 -0.28933 -0.5505
## 1381 42.308 -0.2991  0.334493 -0.6530  0.06925 -1.1779
```

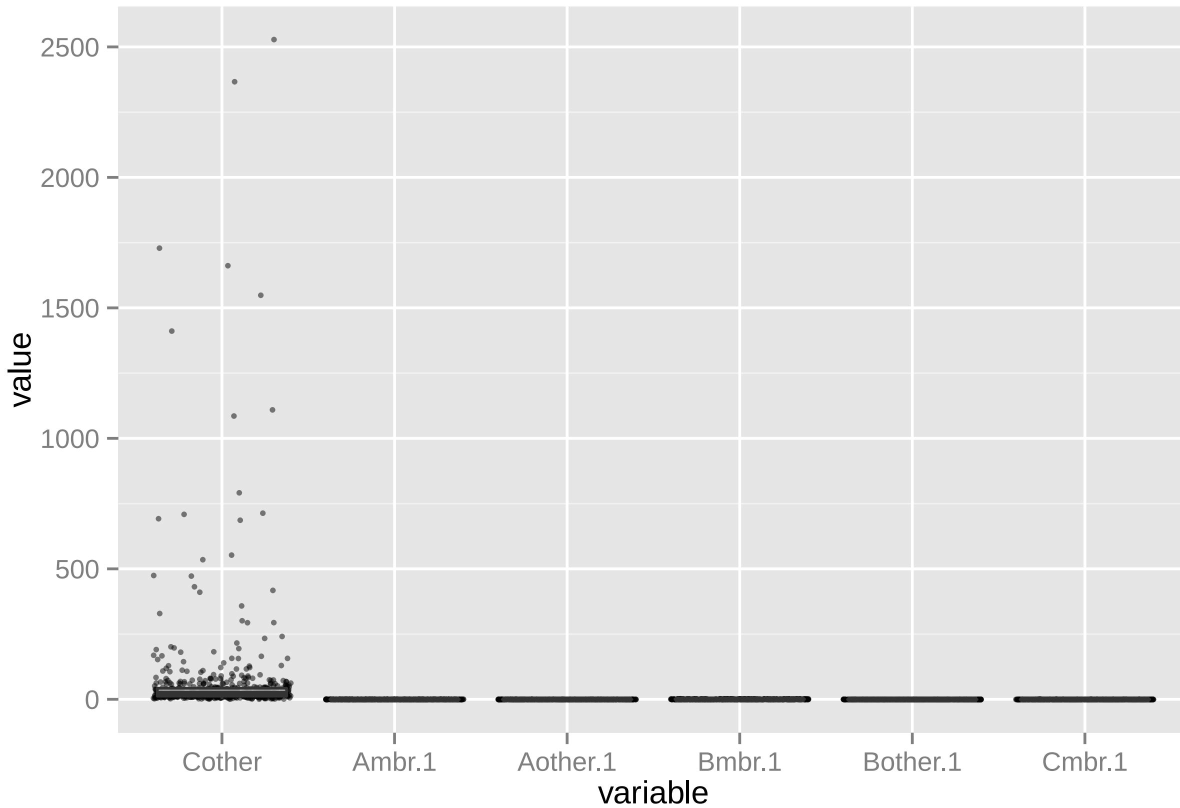
```r
m.data <- melt(sub_data)
```

```
## Using  as id variables
```

```r
head(m.data)
```

```
##   variable  value
## 1   Cother 53.931
## 2   Cother 13.976
## 3   Cother 12.109
## 4   Cother 31.107
## 5   Cother  9.625
## 6   Cother 42.308
```

```r
p <- ggplot(m.data, aes(x=variable, y=value))
p + geom_point(alpha=0.5, position="jitter", size=1) + geom_boxplot(alpha=0.75, outlier.size=0)
```



Conclusion:

It appears as though we get clustering in tissue specific regions. When you set the size of the map to six, you get six clusters that are tissue specific regions. The problem comes when that there is a "peak" in the counts data and ideally you want a more even distribution according to the tutorial [1]. I then made the map bigger and was able down the line to get 5 distinct clusters. The question is are these the same clusters that were in the original 3,3 map? I will have to isolate the each of the clusters in both circumstances and see if I have the same genes. It could be that using more clusters will help identify more interaction between the tissue. For instance, genes in one larger cluster migh be comprised of a group of genes that is upregulated

2. Self Organizing Map- Small (3,2)

The size of the map is something that may cause differences in the genes that are clustered. Using a small map size (3,2), I found they cluster in according to tissue type. See below.

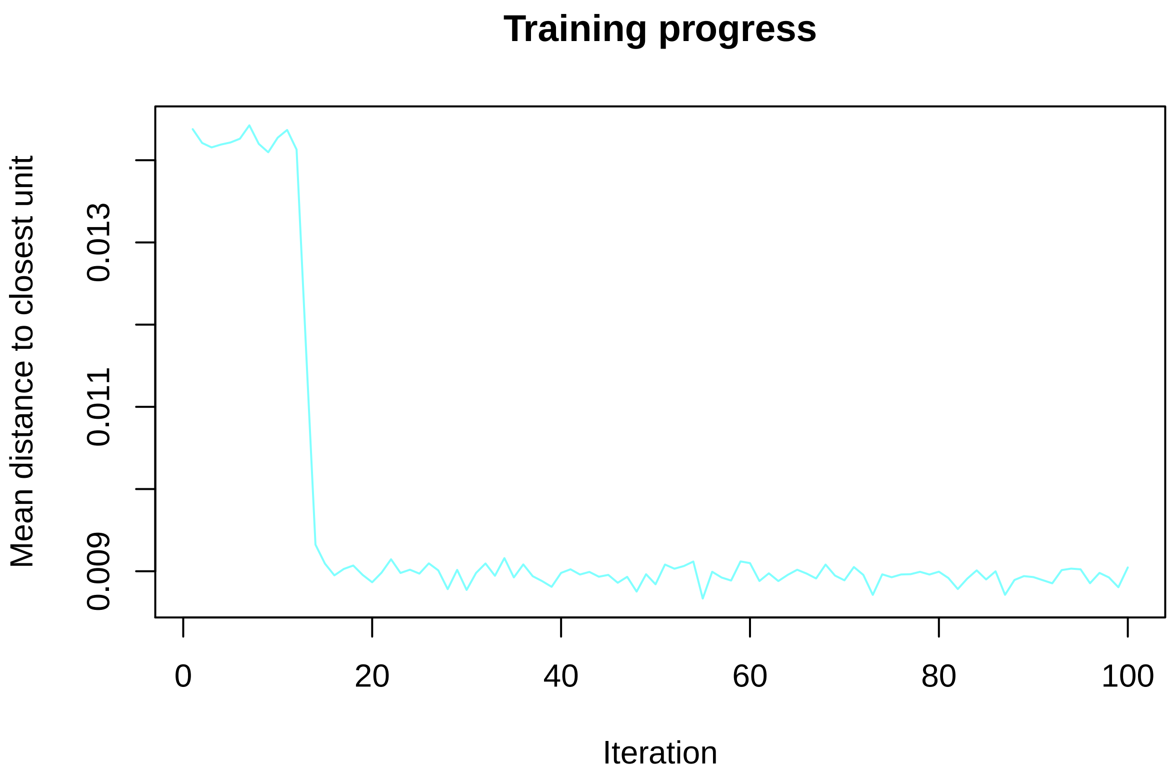
```
data.val.small <- cbind(mostDEgene.long, scale_data, pca.scores) #start with new dataset

som.data <- as.matrix(data.val.small[,c(8:13)])  #subset only the scaled gene expression values

set.seed(2)

som <- som(data=som.data, somgrid(3,2,"hexagonal")) # This is where you change the size of the map
summary(som)
```

```
## som map of size 3x2 with a hexagonal topology.
## Training data included; dimension is 2249 by 6
## Mean distance to the closest unit in the map: 1.076
```

Training Plot ("changes") - Small

This shows a hundred iterations. Training plateaus at around 20 iterations steeply. Is this a problem?

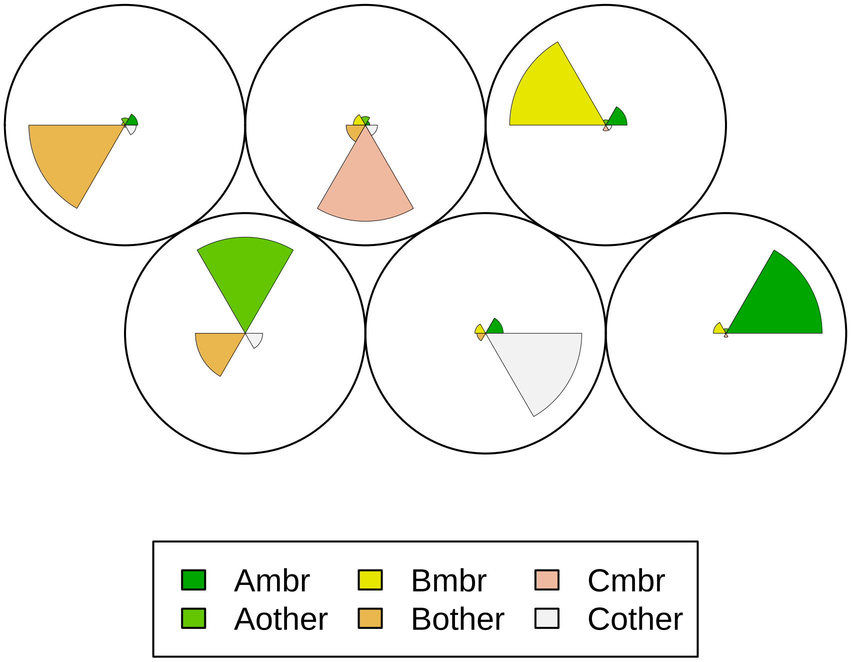
```
plot(som, type ="changes")
```



Code Plot - Small

Here with the small map, each tissue has a tissue specific cluster.

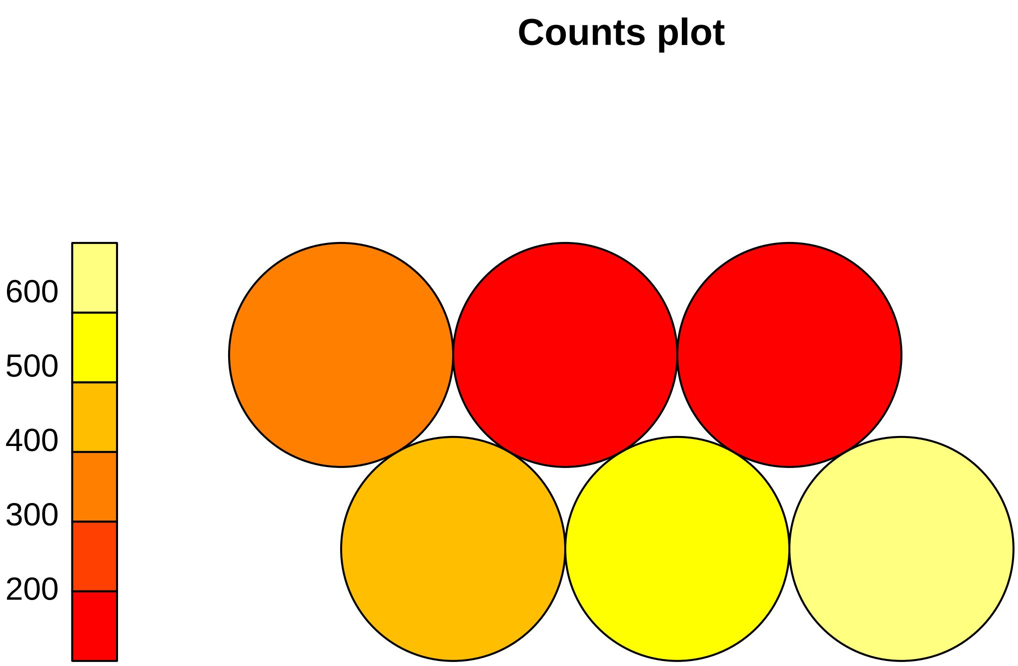
```
plot(som, type = "codes")
```



Count Plot - Small

This tells you how many genes are in each of the clusters. The Aother cluster has around 600 genes, which may be a little high.

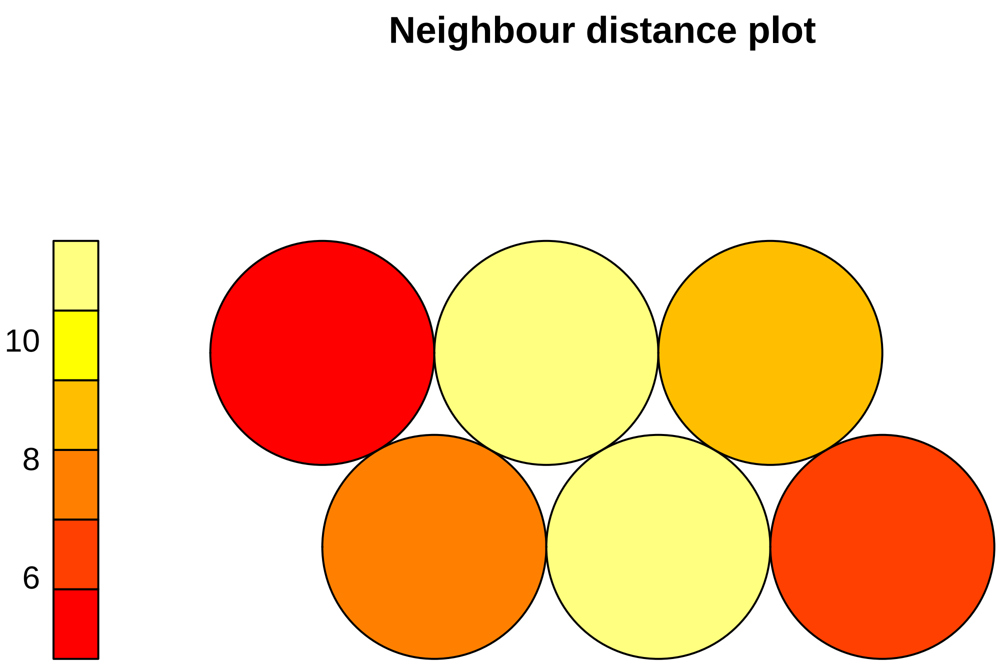
```
plot(som, type = "counts")
```



Distance Neighbour Plot- Small

This is sometimes called the "U-Matrix", it can help identify further clustering. Areas of low neighbour distance indicate groups of nodes that are similar and the further apart nodes indicate natural "borders" in the map.

```
plot(som, type="dist.neighbours")
```



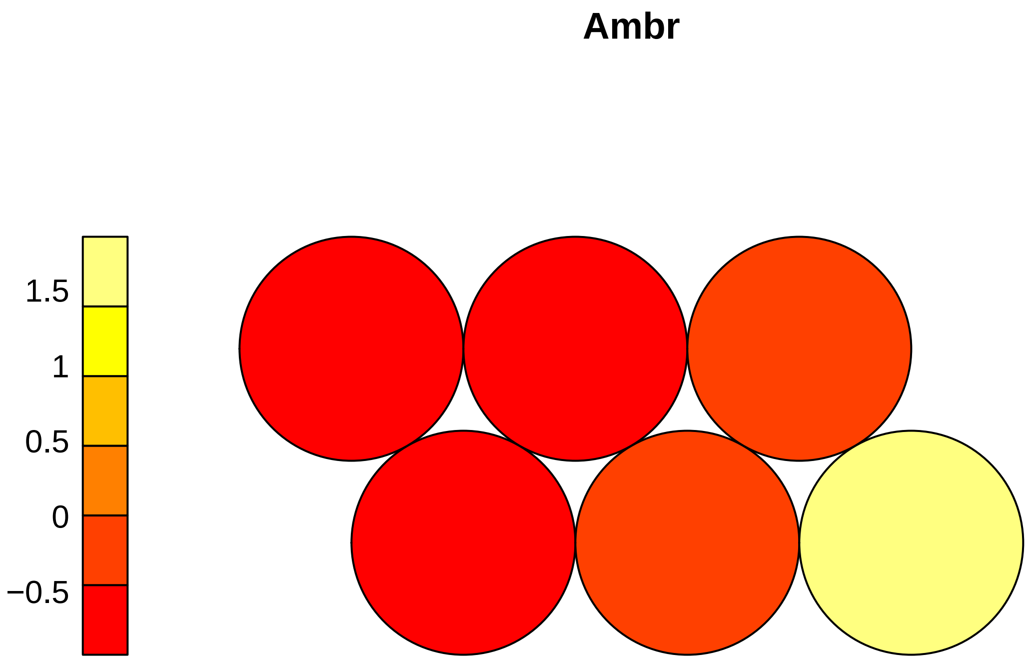
Heatmaps - Small

This shows the distribution of each type of tissue. This doesn't really work too well when the the map is so small. Bother is the only tissue type that contributes to two clusters.

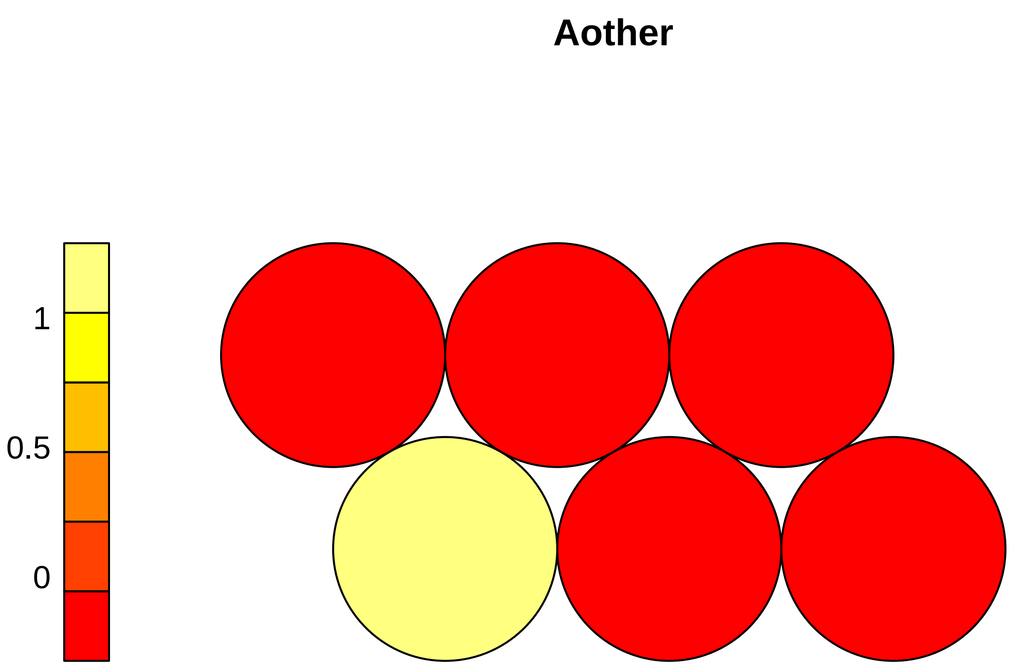
```
head(som$codes)
```

```
##         Ambr  Aother    Bmbr   Bother    Cmbr   Cother
## [1,] -0.9158  1.2883 -0.6326  0.68215 -0.4390  0.01682
## [2,] -0.4084 -0.3225 -0.3759 -0.33660 -0.4250  1.86841
## [3,]  1.8580 -0.2443 -0.3298 -0.52975 -0.3598 -0.39437
## [4,] -0.5490 -0.2023 -0.5542  1.83172 -0.3962 -0.13003
## [5,] -0.7877 -0.1781 -0.3428 -0.07438  1.4922 -0.10915
## [6,] -0.3031 -0.2339  1.6678 -0.55181 -0.3233 -0.25565
```

```
som$data <- data.frame(som$data) #changed to dataframe to extract column names easier.

#This is just a loop that plots the distrinution of each tissue type across the map.
for (i in 1:6){
  plot(som, type = "property", property = som$codes[,i], main=names(som$data)[i])
  print(plot)
  }
```

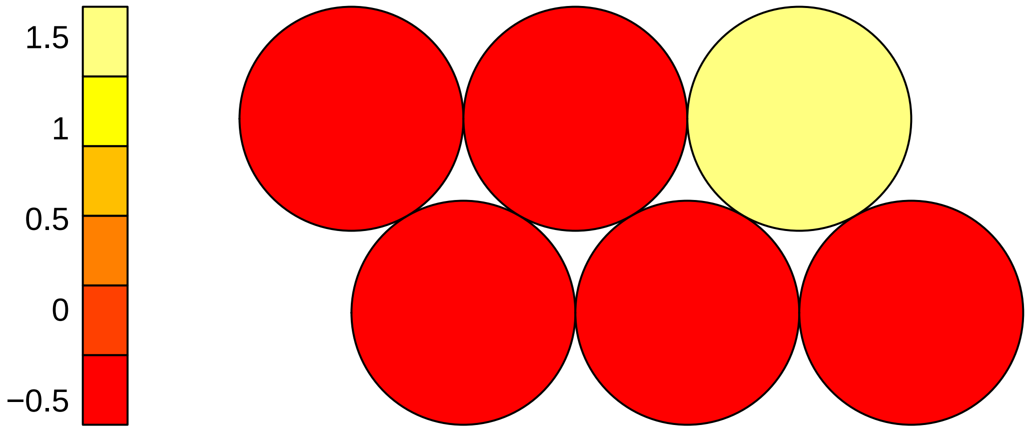



```
## function (x, y, ...)
## UseMethod("plot")
## <bytecode: 0x7fff01192b88>
## <environment: namespace:graphics>
```



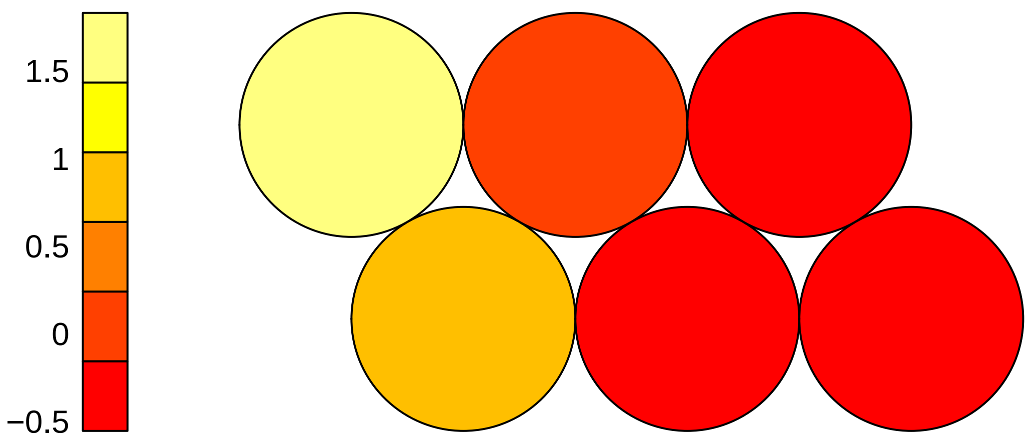

```
## function (x, y, ...)
## UseMethod("plot")
## <bytecode: 0x7fff01192b88>
## <environment: namespace:graphics>
```

Bmbr



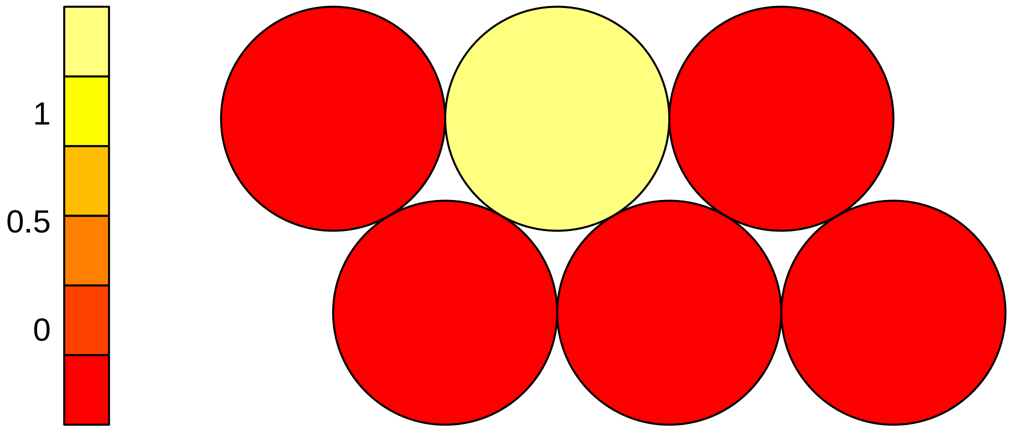

```
## function (x, y, ...)
## UseMethod("plot")
## <bytecode: 0x7fff01192b88>
## <environment: namespace:graphics>
```

Bother



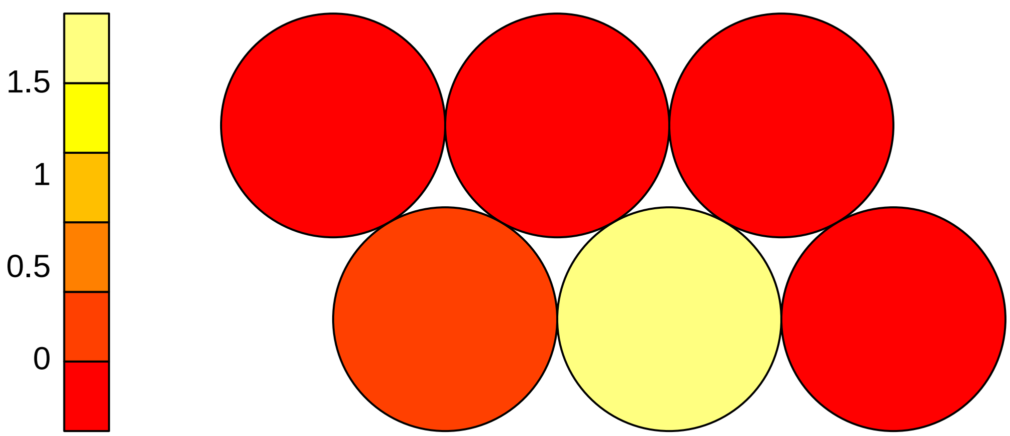

```
## function (x, y, ...)
## UseMethod("plot")
## <bytecode: 0x7fff01192b88>
## <environment: namespace:graphics>
```

Cmbr




```
## function (x, y, ...)
## UseMethod("plot")
## <bytecode: 0x7fff01192b88>
## <environment: namespace:graphics>
```

Cother




```
## function (x, y, ...)
## UseMethod("plot")
## <bytecode: 0x7fff01192b88>
## <environment: namespace:graphics>
```

Output

```
data.val.small <- cbind(data.val.small,som$unit.classif,som$distances)
write.table(data.val.small, file="../data/analysis1.som.data.small.txt")
#Make sure that there is just one of each value som$unit.classif and distances column.
names(data.val.small)
```

```
##  [1] "gene"              "Ambr"              "Aother"
##  [4] "Bmbr"              "Bother"            "Cmbr"
##  [7] "Cother"            "Ambr"              "Aother"
## [10] "Bmbr"              "Bother"            "Cmbr"
## [13] "Cother"            "PC1"               "PC2"
## [16] "PC3"               "PC4"               "PC5"
## [19] "PC6"               "som$unit.classif" "som$distances"
```

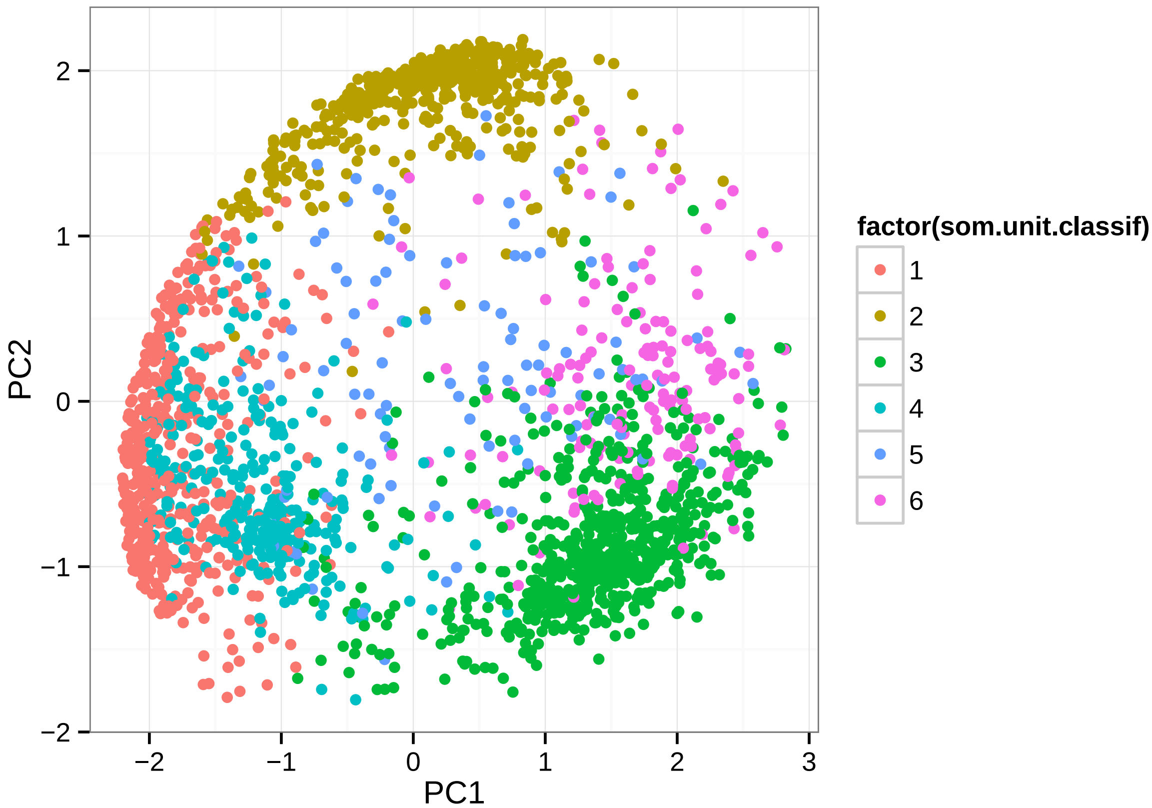
Other Visualization

Visualize by Cluster

```
plot.data <- read.table("../data/analysis1.som.data.small.txt",header=TRUE)
names(plot.data)
```

```
##  [1] "gene"              "Ambr"              "Aother"
##  [4] "Bmbr"              "Bother"            "Cmbr"
##  [7] "Cother"            "Ambr.1"            "Aother.1"
## [10] "Bmbr.1"            "Bother.1"          "Cmbr.1"
## [13] "Cother.1"          "PC1"               "PC2"
## [16] "PC3"               "PC4"               "PC5"
## [19] "PC6"               "som.unit.classif" "som.distances"
```

```
p <- ggplot(plot.data, aes(PC1, PC2, colour=factor(som.unit.classif))) #use unit.classif for smaller da
p + geom_point() + theme_bw()
```



Visualize by individual clusters

```
sub_cluster <- subset(plot.data, som.unit.classif=="3")

sub_data <- sub_cluster[,8:13] # just the sample types

names(sub_data)
```

```
## [1] "Ambr.1"   "Aother.1" "Bmbr.1"   "Bother.1" "Cmbr.1"   "Cother.1"
```

```
head(sub_data)
```

```
##    Ambr.1 Aother.1  Bmbr.1 Bother.1  Cmbr.1 Cother.1
## 5   2.002 -0.15242 -0.7323  -0.4383 -0.2764  -0.4030
## 6   2.023 -0.40005 -0.6143  -0.3573 -0.1986  -0.4524
## 7   1.936 -0.45401 -0.7368  -0.4386  0.2081  -0.5151
## 8   2.007 -0.40461 -0.6867  -0.4799 -0.1407  -0.2949
## 11  1.936  0.09519 -0.8327  -0.3996 -0.2083  -0.5901
## 12  1.928  0.15182 -0.8523  -0.4527 -0.2697  -0.5052
```

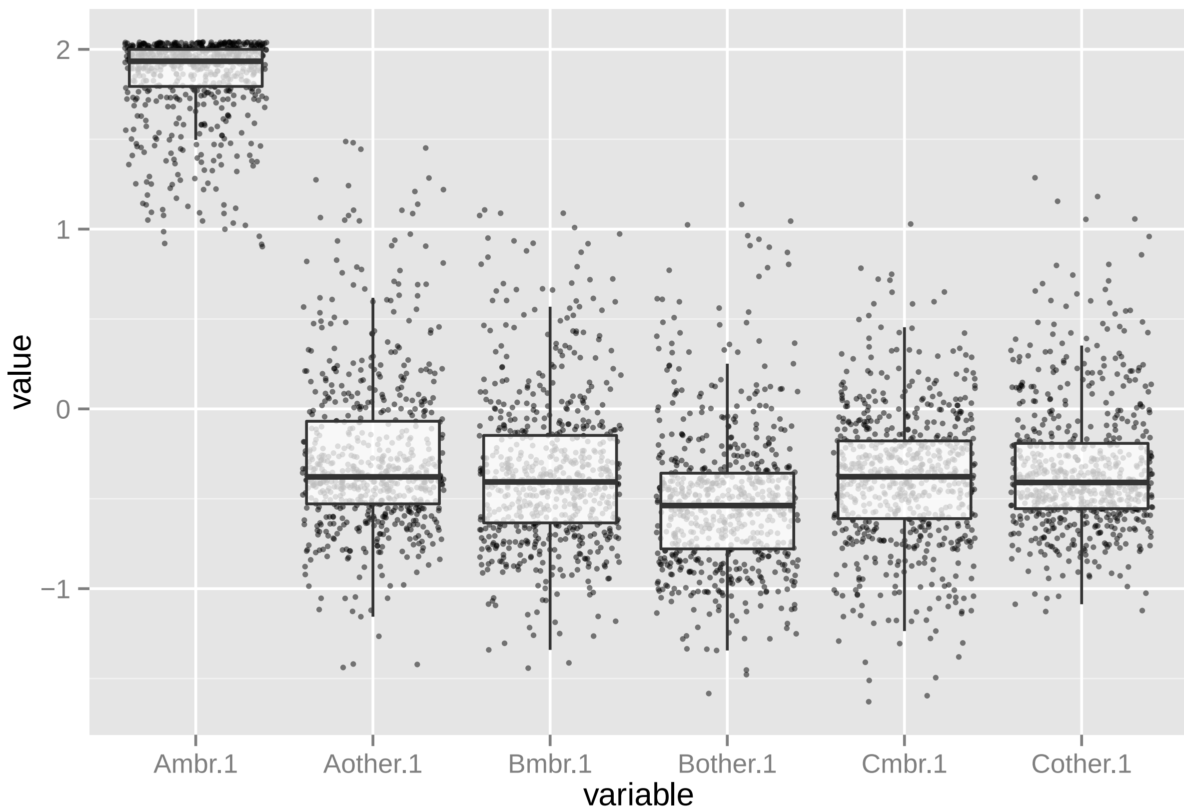
```
m.data <- melt(sub_data)
```

```
## Using  as id variables
```

```r
head(m.data)
```

```
##   variable value
## 1   Ambr.1 2.002
## 2   Ambr.1 2.023
## 3   Ambr.1 1.936
## 4   Ambr.1 2.007
## 5   Ambr.1 1.936
## 6   Ambr.1 1.928
```

```r
p <- ggplot(m.data, aes(x=variable, y=value))
p + geom_point(alpha=0.5, position="jitter", size=1) + geom_boxplot(alpha=0.75, outlier.size=0)
```



Conclusion:

Just looking at the PCA between the two, they show the same clusters no matter if you do the small or large. So in light of this, I am going to go ahead and continue with the small because it is more intuitive to understand and essentially they are the same thing.

References

1. R self Organizing map tutorial