chroGPS: visualizing the epigenome.

Oscar Reina *and David Rossell *

1 Introduction

The chroGPS package provides tools to generate intuitive maps to visualize the association between genetic elements, with emphasis on epigenetics. The approach is based on Multi-Dimensional Scaling. We provide several sensible distance metrics, and adjustment procedures to remove systematic biases typically observed when merging data obtained under different technologies or genetic backgrounds. This manual illustrates the software functionality and highlights some ideas, for a detailed technical description the reader is referred to the supplementary material on (**?**).

Many routines allow performing computations in parallel by specifying an argument `mc.cores`, which uses package `parallel`.

We start by loading the package and a ChIP-chip dataset with genomic distribution of 20 epigenetic elements from the Drosophila melanogaster S2-DRSC cell line, coming from the modEncode project, which we will use for illustration purposes. Even though our study and examples focuses on assessing associations between genetic elements, this methodology can be successfully used with any kind of multivariate data where relative distances between elements of interest can be computed based on a given set of variables.

2 chroGPS^{factors}

```
> options(width=70)
> par(mar=c(2,2,2,2))
> library(chroGPS)
> data(s2) # Loading Dmelanogaster S2 modEncode toy example
> data(toydists) # Loading precomputed distGPS objects
> s2

RangedDataList of length 20
names(20): ASH1-Q4177.S2 CP190-HB.S2 ... Su(var)3-9.S2 mod2.2-VC.S2
```

`s2` is a `RangedDataList` object storing the binding sites for 20 Drosophila melanogaster S2-DRSC sample proteins. Data was retrieved from the modEncode website (www.modencode.org)

*Bioinformatics & Biostatistics Unit, IRB Barcelona

and belongs to the public subset of the Release 29.1 dataset. GFF files were downloaded, read and formatted into individual RangedData objects, stored later into a `RangedDataList` (see functions `getURL` and `gff2RDList` for details.) For shortening computing time for the dynamic generation of this document, some of the distances between epigenetic factors have been precomputed and stored in the `toydists` object.

2.1 Building chroGPS^{factors} maps

The methodology behing chroGPS^{factors} is to generate a distance matrix with all the pairwise distances between elements of interest by means of a chosen metric. After this, a Multidimensional Scaling representation is generated to fit the n-dimensional distances in a lower (usually 2 or 3) k-dimensional space.

```
> # d <- distGPS(s2, metric='avgdist')
> d

Object of class distGPS with avgdist distances between 20 objects

> mds1 <- mds(d,k=2,type='isoMDS')
> mds1

Object of class MDS approximating distances between 20 objects
R-squared= 0.6284 Stress= 0.0795

> mds1.3d <- mds(d,k=3,type='isoMDS')
> mds1.3d

Object of class MDS approximating distances between 20 objects
R-squared= 0.8577 Stress= 0.0287
```

The R^2 coefficient between the original distances and their approximation in the plot can be seen as an analogue to the percentage of explained variability in a PCA analysis. For our sample data R^2=0.628 and stress=0.079 in the 2-dimensional plot, both of which indicate a fairly good fit. A 3-dimensional plot improves these values. We can produce a map by using the `plot` method for `MDS` objects. The result in shown in Figure **??**. For 3D representations the `plot` method opens an interactive window that allows to take full advantage of the additional dimension. Here we commented out the code for the 3D plot and simply show a snapshot in Figure **??**. Short names for modEncode factors as well as colors for each chromatin domain identified (lightgreen=transcriptionally active elements, purple=Polymerase, grey=boundary elements, yellow=Polycomb repression, lightblue=HP1 repression) are provided in the data frame object `s2names`, stored within `s2`.

```
> cols <- as.character(s2names$Color)
> plot(mds1,drawlabels=TRUE,point.pch=20,point.cex=8,text.cex=.7,
+ point.col=cols,text.col='black',labels=s2names$Factor,font=2)
> legend('topleft',legend=sprintf('R2=%.3f / stress=%.3f',getR2(mds1),getStress(mds1)),
+ bty='n',cex=1)
> #plot(mds1.3d,drawlabels=TRUE,type.3d='s',point.pch=20,point.cex=.1,text.cex=.7,
> #point.col=cols,text.col='black',labels=s2names$Factor)
```

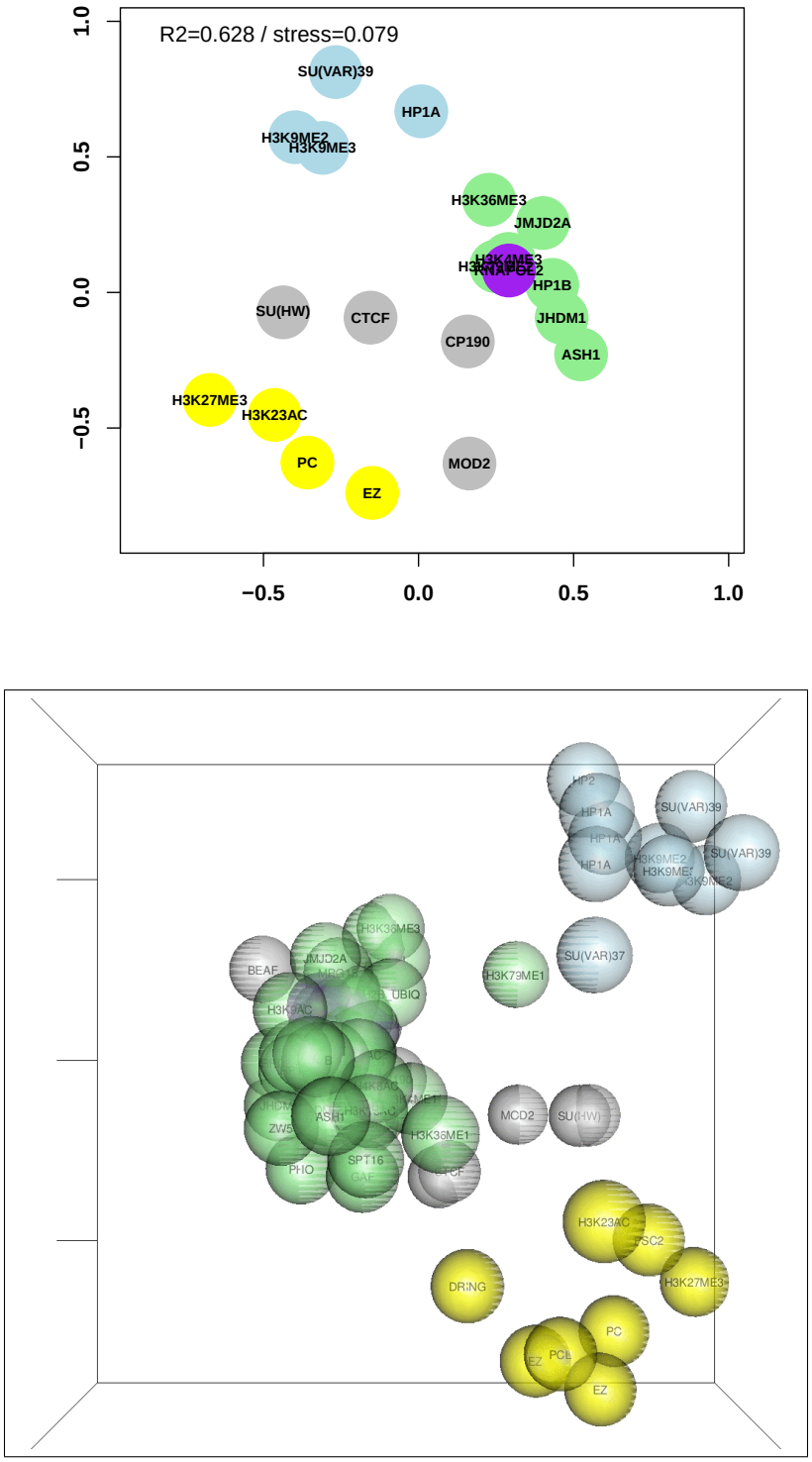

Figure 1: 2D map from the 20 S2 epigenetic factors and example 3D map with 76 S2 factors. Factors with more similar binding site distribution appear closer.

2.2 Integrating data sources: technical background

Currently, genomic profiling of epigenetic factors is being largely determined through high throughput methodologies such as ultra-sequencing (ChIP-Seq), which identifies binding sites with higher accuracy that ChIP-chip experiments. However, there is an extensive knowledge background based on the later. ChroGPS allows integrating different technical sources by adjusting for systematic biases.

We propose two adjustment methods: Procrustes and Peak Width Adjustment. Procrustes finds the optimal superimposition of two sets of points by altering their location, scale and orientation while maintaining their relative distances. It is therefore a general method of adjustment that can take care of several kind of biases. However, its main limitation is that a minimal set of common points (that is, the same factor/protein binding sites mapped in both data sources) is needed to effectively perform a valid adjustment. Due to the spatial nature of Procrustes adjustment, we strongly recommend a minimum number of 3 common points.

We illustrate the adjustments by loading *Drosophila melanogaster* S2 ChIP-seq data obtained from NCBI GEO GSE19325, `http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE19325`. We start by producing a joint map with no adjustment.

```
> data(s2Seq)
> s2Seq

RangedDataList of length 4
names(4): GSM480156_dm3-S2-H3K4me3.bed.rd ...

> # d2 <- distGPS(c(s2,s2Seq),metric='avgdist')
> mds2 <- mds(d2,k=2,type='isoMDS')
> cols <- c(as.character(s2names$Color),as.character(s2SeqNames$Color))
> sampleid <- c(as.character(s2names$Factor),as.character(s2SeqNames$Factor))
> pchs <- rep(c(20,17),c(length(s2),length(s2Seq)))
> point.cex <- rep(c(8,5),c(length(s2),length(s2Seq)))
> par(mar=c(2,2,2,2))
> plot(mds2,drawlabels=TRUE,point.pch=pchs,point.cex=point.cex,text.cex=.7,
+ point.col=cols,text.col='black',labels=sampleid,font=2)
> legend('topleft',legend=sprintf('R2=%.3f / stress=%.3f',getR2(mds2),getStress(mds2)),
+ bty='n',cex=1)
> legend('topright',legend=c('ChIP-Chip','ChIP-Seq'),pch=c(20,17),pt.cex=c(1.5,1))
```

Figure **??** shows the resulting map. While ChIP-seq elements appear close to their ChIP-chip counterparts, they form an external layer. We now apply Procrustes to adjust these systematic biases using function `procrustesAdj`.

```
> adjust <- rep(c('chip','seq'),c(length(s2),length(s2Seq)))
> sampleid <- c(as.character(s2names$Factor),as.character(s2SeqNames$Factor))
> mds3 <- procrustesAdj(mds2,d2,adjust=adjust,sampleid=sampleid)
> par(mar=c(0,0,0,0),xaxt='n',yaxt='n')
> plot(mds3,drawlabels=TRUE,point.pch=pchs,point.cex=point.cex,text.cex=.7,
+ point.col=cols,text.col='black',labels=sampleid,font=2)
> legend('topleft',legend=sprintf('R2=%.3f / stress=%.3f',getR2(mds3),getStress(mds3)),
+ bty='n',cex=1)
> legend('topright',legend=c('ChIP-Chip','ChIP-Seq'),pch=c(20,17),pt.cex=c(1.5,1))
```

```
RangedDataList of length 4
names(4): GSM480156_dm3-S2-H3K4me3.bed.rd ...
```

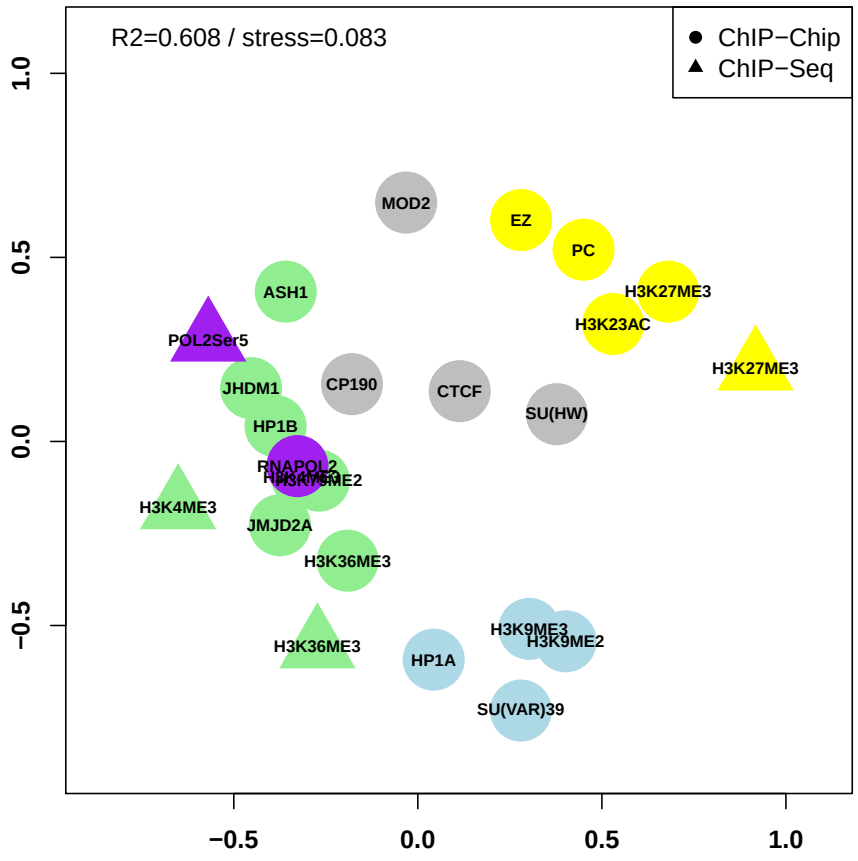

Figure 2: S2 ChIP-chip and ChIP-Seq data, raw integration (no adjustment).

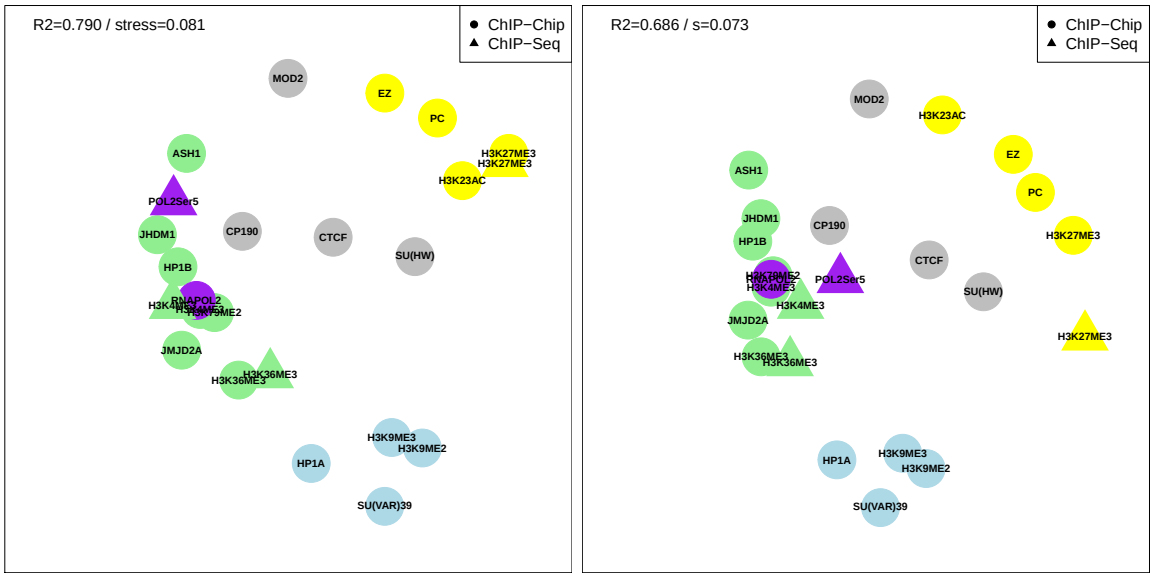

Figure 3: S2 ChIP-chip and ChIP-Seq data. Left: Procrustes adjustment. Right: Peak Width Adjustment.

Peak Width Adjustment relies on the basic difference between the two different sources of information used in our case, that is, the resolution difference between ChIP-Seq and ChIP-chip peaks, which translates basically in the width presented by the regions identified as binding sites, being those peaks usually much wider in ChIP-chip data (poorer resolution).

```
> s2.pAdj <- adjustPeaks(c(s2,s2Seq),adjust=adjust,sampleid=sampleid,logscale=TRUE)
> # d3 <- distGPS(s2.pAdj,metric='avgdist')
> mds4 <- mds(d3,k=2,type='isoMDS')
> par(mar=c(0,0,0,0),xaxt='n',yaxt='n')
> plot(mds4,drawlabels=TRUE,point.pch=pchs,point.cex=point.cex,text.cex=.7,
+ point.col=cols,text.col='black',labels=sampleid,font=2)
> legend('topleft',legend=sprintf('R2=%.3f / s=%.3f',getR2(mds4),getStress(mds4)),
+ bty='n',cex=1)
> legend('topright',legend=c('ChIP-Chip','ChIP-Seq'),pch=c(20,17),pt.cex=c(1.5,1))
```

Figure **??** shows the map after Peak Width Adjustment, where ChIP-chip and ChIP-seq elements have been adequately matched. Whenever possible, we strongly recommend using Procrustes adjustment due to its general nature and lack of mechanistic assumptions. This is even more important if integrating other data sources for binding site discovery, such as DamID, Chiapet, etc, where technical biases are more complex than just peak location resolution and peak size.

3 ChroGPS^{genes}

In addition to assessing relationship between epigenetic factors, chroGPS also provides tools to generate chroGPS^{genes} maps, useful to visualize the relationships between genes based on their epigenetic pattern similarities (the epigenetic marks they share).

3.1 Building chroGPS^{genes} maps

The proceedings are analog to those of chroGPS^{factors}, that is, the definition of a metric to measure similarity between genes and using it to generate MDS representations in k-dimensional space. The data source of chroGPS^{genes} has to be a matrix or data frame of N genes x M factors (rows x cols), where each cell has a value of 1 if a binding site for that protein or factor has been found in the region defined by that gene. This annotation table can be generated by multiple methods, in our case we annotated the genomic distribution on 76 S2 modEncode against the Drosophila melanogaster genome (Ensembl february 2012), accounting for strict overlaps within 1000bp of gene regions, using the `annotatePeakInBatch` function from the `ChIPpeakAnno` package (**?**). After that, 500 random genes were selected randomly and this is the dataset that will be used in all further examples.

```
> s2.tab[1:10,1:4]
```

```
            ASH1-Q4177.S2 BEAF-70.S2 BEAF-HB.S2 Chro(Chriz)BR.S2
FBgn0051778             0          0          0                0
FBgn0028562             0          0          0                0
FBgn0011653             0          0          0                0
FBgn0262889             0          0          0                0
FBgn0030056             1          0          1                1
FBgn0035496             0          0          0                0
FBgn0026149             1          0          1                1
FBgn0030142             0          0          1                1
FBgn0003008             0          1          1                1
FBgn0052703             0          0          0                0
```

```
> d <- distGPS(s2.tab, metric='tanimoto', uniqueRows=TRUE)
> d
```

```
Object of class distGPS with tanimoto distances between 466 objects
```

```
> mds1 <- mds(d,k=2,type='isoMDS')
> mds1
```

```
Object of class MDS approximating distances between 466 objects
R-squared= 0.8217 Stress= 0.1269
```

```
> mds2 <- mds(d,k=3,type='isoMDS')
> mds2
```

```
Object of class MDS approximating distances between 466 objects
R-squared= 0.8884 Stress= 0.0757
```

Increasing k improves the R^2 and stress values. For our examples here we use non-metric isoMDS by indicating `type='isoMDS'`, which calls the `isoMDS` function from the `MASS` package (**?**).

```
> par(mar=c(2,2,2,2))
> plot(mds1,point.cex=1.5,point.col=densCols(getPoints(mds1)))
> #plot(mds2,point.cex=1.5,type.3d='s',point.col=densCols(getPoints(mds2)))
```

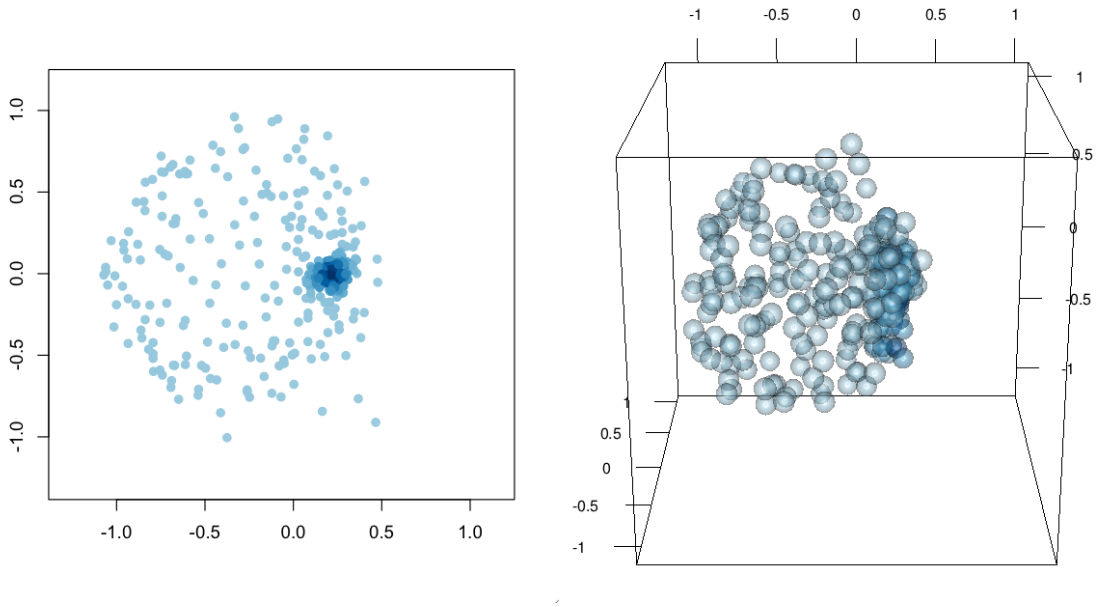

Figure 4: 2 and 3-dimensional chroGPS^{genes}. Genes with more similar epigenetic marks (binding site patterns) appear closer.

3.2 Genome-wide chroGPS^{genes} maps

As mentioned, our example dataset for chroGPS^{genes} maps consists in a combination of 76 protein binding sites for 500 genes. When only unique factor combinations are considered (all genes sharing a specific combination of epigenetic marks are merged into a single 'epigene'), the size of the dataset gets down to 466 genes per 76 factors.

```
> dim(s2.tab)

[1] 500  76

> dim(uniqueCount(s2.tab))

[1] 466  78
```

However, when genome-wide patterns are considered, the number of epigenes can still be very high, in the order of ten thousand unique epigenes. This poses a real challenge for Multidimensional Scaling when trying to find an optimal solution for k-space representation of the pairwise distances both in terms of accuracy and computational cost.

We start by re-running the isoMDS fit and measuring the CPU time.

```
> system.time(mds3 <- mds(d,k=2,type='isoMDS'))

   user  system elapsed
  4.140   0.024   4.167

> mds3

Object of class MDS approximating distances between 466 objects
R-squared= 0.8217 Stress= 0.1269
```

We now apply our BoostMDS algorithm, which is a 2-step procedure (see package help for function `mds` and Supplementary Methods of (**?**) for details). BoostMDS generates maps at much lower time and memory consumption requirements, while improving the R^2 and stress coefficients. The first step is to obtain an initial solution by randomly splitting the original distance matrix in a number of smaller submatrices with a certain number of overlapping elements between them, so that individual MDS representations can be found for each one and later become stitched by using Procrustes with their common points. The second step is to formally maximize the R^2 coefficient by using a gradient descent algorithm using the `boostMDS` function. The second step also ensures that the arbitrary split used in the first step does not have a decisive effect on the final MDS point configuration.

```
> system.time(mds3 <- mds(d,type='isoMDS',splitMDS=TRUE,split=.5,overlap=.05,mc.cores=1))

   user  system elapsed
  1.157   0.035   1.192

> mds3

Object of class MDS approximating distances between 466 objects
R-squared= 0.8002 Stress= 0.1301

> system.time(mds4 <- mds(d,mds3,type='boostMDS',scale=TRUE))

Sampling  100 elements...
   Correl   Step size
0.7651936
0.8059597 0.04414643
0.8189542 0.0216275
0.8212578 0.0184511
0.8224748 0.01060938
0.8233287 0.01373785
   user  system elapsed
  0.626   0.044   0.671

> mds4

Object of class MDS approximating distances between 466 objects
R-squared= 0.8457 Stress= 0.1224
```

Here BoostMDS provided a better solution in terms of R^2 and stress than isoMDS, at a lower computational time. Our experience is that in a real example with tens of thousands of points the advantages become more extreme.

3.3 Annotating chroGPS^{genes} maps with quantitative information

Gene expression, coming from a microarray experiment or from more advanced RNA-Seq techniques is probably one of the first sources of information to be used when studying a given set of genes. Another basic source of information from epigenetic data is the number of epigenetic marks present on a given set of genes. It is known that some genes present more complex regulation programs that make necessary the co-localization of several DNA binding proteins.

ChroGPS^{genes} maps provide a straightforward way of representing such information over a context-rich base. Basically, coloring epigenes according to a color scale using their average gene expression or number of epigenetic marks is sufficient to differentiate possible regions of interest. Thus, our chroGPS^{genes} map turn into a context-rich heatmap where genes relate together due to their epigenetic similarity and at the same time possible correlation with gene expression is clearly visible. Furthermore, if expression data along a timeline is available, for instance on an experiment studying time-dependant gene expression after certain knock-out or gene activation, one can track expression changes on specific map regions.

In our case, we will use expression information coming from a microarray assay involving normal Drosophila S2-DSRC cell lines. The object `s2.wt` has normalized median expression value per gene and epigene (*i.e.*, we compute the median expression of all genes with the same combination of epigenetic marks). The resulting plot is shown in Figure **??**

```
> summary(s2.wt$epigene)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  2.192   4.518   8.934   7.917  10.570  13.260      47

> summary(s2.wt$gene)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  2.136   3.987   8.343   7.454  10.430  13.260      31

> plot(mds1,point.cex=1.5,scalecol=TRUE,scale=s2.wt$epigene,
+      palette=rev(heat.colors(100)))
```

3.4 Annotating chroGPS^{genes} maps: clustering

A natural way to describe chroGPS^{genes} maps is to highlight a set of genes of interest, for instance those possessing an individual epigenetic mark. One can repeat this step for several interesting gene sets but this is cumbersome and doesn't lead to easy interpretation unless very few sets are considered. A more advanced approach is to analyze the whole set of epigene dissimilarities by clustering, allowing us to detect genes with similar epigenetic patterns. Again, using colors to represent genes in a given cluster gives an idea of the underlying structure, even though overlapping areas are difficult to follow, specially as the number of considered clusters increase. We now use hierarchical clustering with average linkage to find gene clusters. We will illustrate an example where we consider a partition with between cluster distances of 0.5.

Clustering algorithms may deliver a large number of small clusters which are difficult to interpret. To overcome this, we developed a `preMerge` step that assigns clusters below a certain size to its closest cluster according to centroid distances. After the pre-merging step, the number of clusters is reduced considerably, and all them have a minimum size which allows easier map interpretation. The function `clusGPS` integrates a clustering result into an existing map. It also computes density estimates for

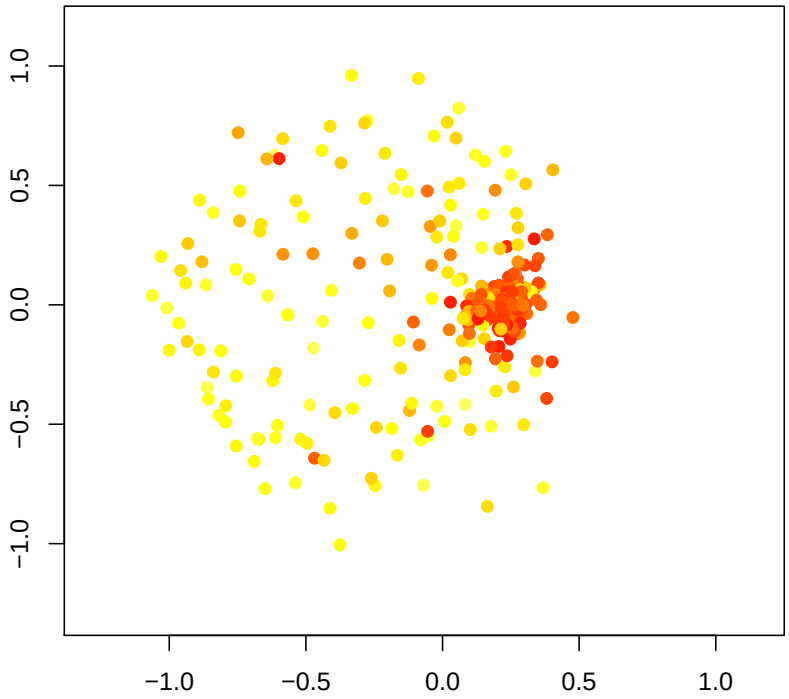

Figure 5: 2-dimensional MDS plot with chroGPS^{genes} map and gene expression information.

each cluster in the map, which can be useful to assess cluster separation and further merge clusters, as we shall see later. We will first perform a hierarchical clustering over the distance matrix, which we can access with the `as.matrix` function.

```
> h <- hclust(as.dist(as.matrix(d)),method='average')
> set.seed(149) # Random seed for the MCMC process within density estimation
> clus <- clusGPS(d,mds1,h,ngrid=1000,densgrid=FALSE,verbose=TRUE,
+ preMerge=TRUE,k=max(cutree(h,h=0.5)),minpoints=20,mc.cores=1)

Precalculating Grid

Pre-merging non-clustered points in nodules of size 20...

Calculating posterior density of mis-classification for cluster: 1


Calculating posterior density of mis-classification for cluster: 2


Calculating posterior density of mis-classification for cluster: 6


Calculating posterior density of mis-classification for cluster: 28


Calculating posterior density of mis-classification for cluster: 56


Calculating posterior density of mis-classification for cluster: 89


Adjusting posterior probabilities...

> clus

Object of class clusGPS with clustering for 466 elements.
1 clustering configuration(s) with name(s) 125
```

We can represent the output of `clusGPS` graphically using the `plot` method. The result in shown in Figure **??**. We appreciate that the the resulting configuration presents a main central cluster (cluster 56, n=293 epigenes, colored in blue) containing more than 50 percent of genes in all map, and is surrounded by smaller ones that distribute along the external sections of the map. Our functions `clusNames` and `tabClusters` provides information about the name and size of the cluster partitions stored within a `clusGPS` object. The function `clusterID` can be used to retrieve the vector of cluster assignments for the elements of a particular clustering configuration.

```
> clus

Object of class clusGPS with clustering for 466 elements.
1 clustering configuration(s) with name(s) 125

> clusNames(clus)

[1] "125"
```

```
> tabClusters(clus,125)

  1   2   6  28  56  89
 39  38  25  48 293  23

> point.col <- rainbow(length(tabClusters(clus,125)))
> names(point.col) <- names(tabClusters(clus,125))
> point.col

          1           2           6          28          56
"#FF0000FF" "#FFFF00FF" "#00FF00FF" "#00FFFFFF" "#0000FFFF"
         89
"#FF00FFFF"

> par(mar=c(0,0,0,0),xaxt='n',yaxt='n')
> plot(mds1,point.col=point.col[as.character(clusterID(clus,125))],
+ point.pch=19)
```

Different clustering algorithms can deliver significantly different results, thus it is important to decide how to approach the clustering step depending on your data. Our example using hclust with average linkage tends to divide smaller and more divergent clusters before, while other methods may first 'attack' the most similar agglomerations. You can use any alternative clustering algorithm by formatting its result as an `hclust` object h and passing it to the `clusGPS` function.

3.5 Cluster visualization with density contours

We achieve this by using a contour representation to indicate the regions in the map where a group of genes (ie genes with a given mark) locate with high probability. The contour representation provides a clearer visualization of the extent of overlap between gene sets, in an analog way to those of the popular Venn diagrams but with the benefit of a context-rich base providing a functional context for interpretation.

```
> par(mar=c(0,0,0,0),xaxt='n',yaxt='n')
> plot(mds1,point.cex=1.5,point.col='grey')
> for (p in c(0.95, 0.50))
+ plot(clus,type='contours',k=max(cutree(h,h=0.5)),lwd=5,probContour=p,
+ drawlabels=TRUE,labcex=2,font=2)
```

The `clusGPS` function computes Bayesian non-parametric density estimates using the `DPdensity` function from the `DPPackage` package, but individual contours can be generated and plotted by just calling the `contour2dDP` function with a given set of points from the MDS object. Keep in mind that computation of density estimates may be imprecise with clusters of very few elements. Check the help of the `clusGPS` function to get more insight on the `minpoints` parameter and how it relates to the `preMerge` step described above.

3.6 Assessing cluster separation in chroGPS^{genes} maps

Deciding the appropiate number of clusters is not an easy question. `chroGPS` provides a method to evaluate cluster separation in the lower dimensional representation. The cluster density estimates can be used to compute the posterior expected

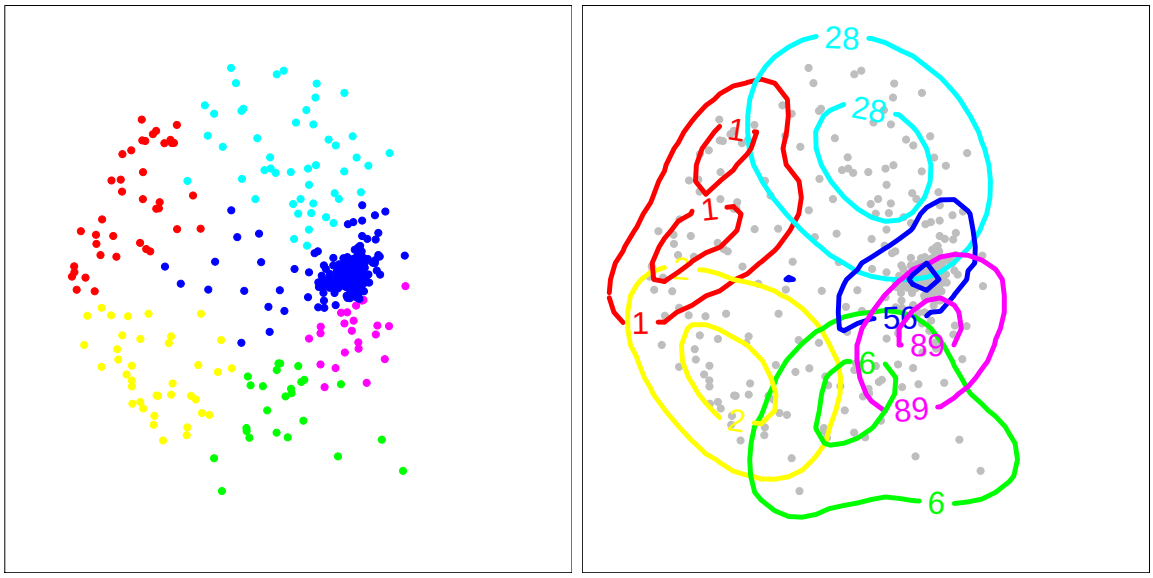

Figure 6: 2-dimensional MDS plot with chroGPS^{genes} map and cluster identities indicated by point colors (left) and probabilistic contours drawn at 50 and 95 percent (right).

correct classification rate (CCR) for each point, cluster and for the whole map, thus not only giving an answer to how many clusters to use, but also to show reproducible are the individual clusters in the chosen solution. Intuitively, when two clusters share a region of high density in the map, their miss-classification rate increases. We can assess the CCR for each cluster using the `plot` function with the argument `type='stats'`. Figure **??** shows the obtained plot. The dashed black line indicates the overall CCR for the map, which is slightly lower than 0.9. All individual clusters have a CCR ≥ 0.8.

```
> plot(clus,type='stats',k=max(cutree(h,h=0.5)),ylim=c(0,1),col=point.col,cex=2,pch=19,
+ lwd=2,ylab='CCR',xlab='Cluster ID',cut=0.75,cut.lty=3,axes=FALSE)
> axis(1,at=1:length(tabClusters(clus,125)),labels=names(tabClusters(clus,125))); axis(2)
> box()
```

3.7 Locating genes and factors on chroGPS^{genes} maps

A natural question is where genes having a given epigenetic mark tend to locate on the map. An easy solution is just to highlight those points on a map, but that may be misleading, especially when multiple factors are considered simultaneously. We offer tools to locate high-probability regions (*i.e.* regions on the map containing a certain proportion of all the genes with a given epigenetic mark or belonging to a specific Gene Ontology term). For instance, we will highlight the genes with the epigenetic factor HP1a. The result is shown in Figure **??** (left). We see that HP1a shows a certain bimodality in its distribution, with a clear presence in the central clusters (56, 89) but also in the upper left region of the map (cluster 1 and to a lesser extent, 28).

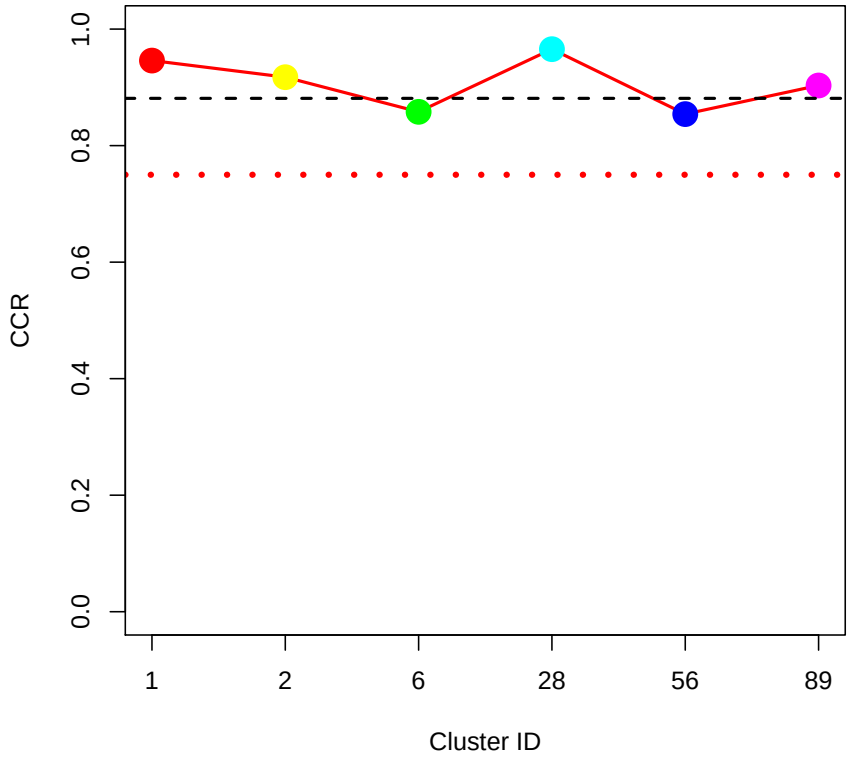

Figure 7: Per-cluster (dots and continuous line) and global (dashed line) Correct Classification Rate. Red pointed line indicates an arbitrary threshold of 0.75 CCR. Higher values indicate more robust clusters which are better separated in space.

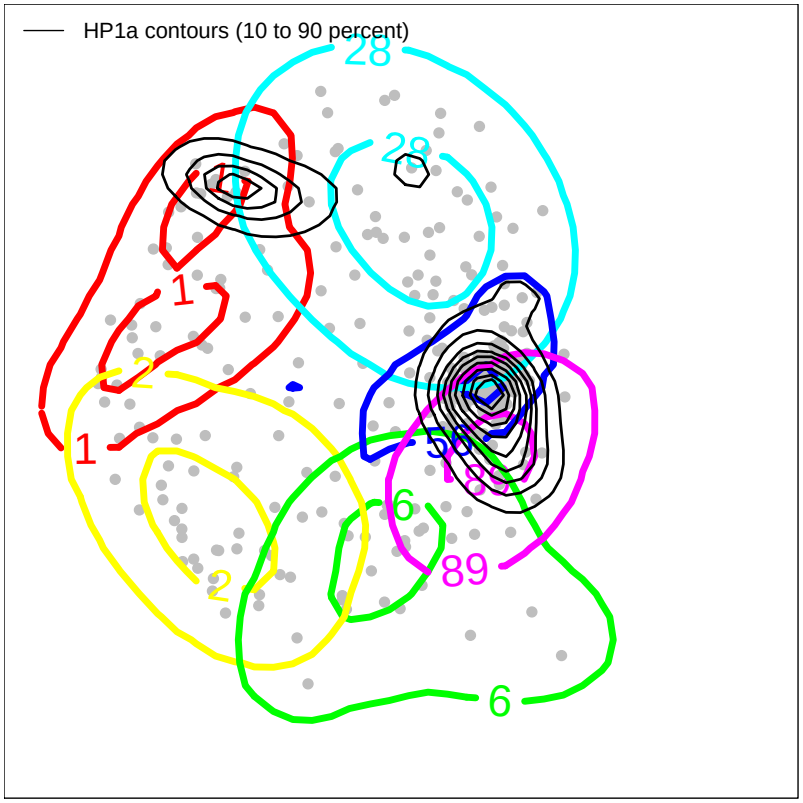

Figure 8: chroGPS^{genes} map with cluster contours at 50 and 95 percent the 5 clusters presented above. In black, probability contour for HP1a factor.

```
> par(mar=c(0,0,0,0),xaxt='n',yaxt='n')
> plot(mds1,point.cex=1.5,point.col='grey')
> for (p in c(0.5,0.95)) plot(clus,type='contours',k=max(cutree(h,h=0.5)),lwd=5,probContour=p,
+ drawlabels=TRUE,labcex=2,font=2)
> fgenes <- uniqueCount(s2.tab)[,'HP1a_wa184.S2']==1
> set.seed(149)
> c1 <- contour2dDP(getPoints(mds1)[fgenes,],ngrid=1000,contour.type='none')
```

```
> for (p in seq(0.1,0.9,0.1)) plotContour(c1,probContour=p,col='black')
> legend('topleft',lwd=1,lty=1,col='black',legend='HP1a contours (10 to 90 percent)',bty='n')
```

Highlighting a small set of genes on the map (*e.g.* canonical pathways) is also possible by using the `geneSetGPS` function. We randomly select 10 genes for illustration purposes. Figure **??** (right) shows the results.

```
> par(mar=c(0,0,0,0),xaxt='n',yaxt='n')
> plot(mds1,point.cex=1.5,point.col='grey')
> for (p in c(0.5,0.95)) plot(clus,type='contours',k=max(cutree(h,h=0.5)),lwd=5,probContour=p,
+ drawlabels=TRUE,labcex=2,font=2)
> set.seed(149) # Random seed for random gene sampling
> geneset <- sample(rownames(s2.tab),10,rep=FALSE)
> mds2 <- geneSetGPS(s2.tab,mds1,geneset,uniqueCount=TRUE)
```

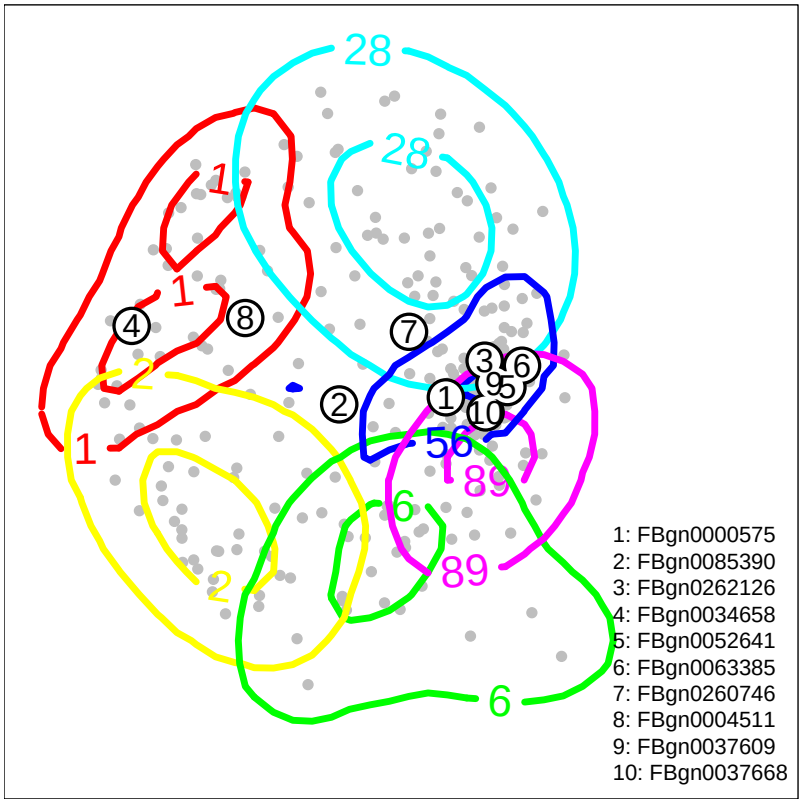

Figure 9: chroGPS^{genes} map with cluster contours at 50 and 95 percent the 5 clusters presented above. Left: In black, probability contour for HP1a factor. Right: random geneset located on the chroGPS^{genes} map.

```
> points(getPoints(mds2),col='black',cex=5,lwd=4,pch=20)
> points(getPoints(mds2),col='white',cex=4,lwd=4,pch=20)
> text(getPoints(mds2)[,1],getPoints(mds2)[,2],1:nrow(getPoints(mds2)),cex=1.5)
> legend('bottomright',col='black',legend=paste(1:nrow(getPoints(mds2)),
+ geneset,sep=': '),cex=1,bty='n')
```

3.8 Merging overlapping clusters

As discussed in Section **??**, for our toy example clusters obtained by setting a between-cluster distance threshold of 0.5 are well-separated and the CCR is high. When the number of points is higher or the threshold is set to a lower value, it is common that some clusters overlap substantially, hampering interpretation. Cluster density estimates offer us an elegant way to detect significant cluster overlap over the space defined by our MDS map, and thus allow us to merge clearly overlapping clusters. Our approach performs this merging in an unsupervised manner, by merging in each step the two clusters having maximum spatial overlap, and stopping when the two next clusters to merge show an overlap substantially lower than that from previous steps. For more details, check help for the function `cpt.mean` in the `changepoint` package. By obtaining clusters which better separate in space, their rate of correct classification also improves, delivering a map configuration which is

robust, intuitive, and easy to interpret, specially with very populated maps where the initial number of clusters may be very high.

To illustrate the usefulness of cluster merging in some conditions, we will use a different cluster cut so that their boundaries overlap more significantly in our 2D map. We then merge clusters using the `mergeClusters` function.

```
> set.seed(149) # Random seed for MCMC within the density estimate process
> clus2 <- clusGPS(d,mds1,h,ngrid=1000,densgrid=FALSE,verbose=TRUE,
+ preMerge=TRUE,k=max(cutree(h,h=0.2)),minpoints=20,mc.cores=1)

Precalculating Grid

Pre-merging non-clustered points in nodules of size 20...

Calculating posterior density of mis-classification for cluster: 1


Calculating posterior density of mis-classification for cluster: 2


Calculating posterior density of mis-classification for cluster: 6


Calculating posterior density of mis-classification for cluster: 42


Calculating posterior density of mis-classification for cluster: 148


Calculating posterior density of mis-classification for cluster: 156


Calculating posterior density of mis-classification for cluster: 200


Calculating posterior density of mis-classification for cluster: 201


Calculating posterior density of mis-classification for cluster: 245


Adjusting posterior probabilities...

> par(mar=c(2,2,2,2))
> clus3 <- mergeClusters(clus2,brake=0,mc.cores=1)
> clus3

Object of class clusGPS with clustering for 466 elements.
1 clustering configuration(s) with name(s) 330

> tabClusters(clus3,330)

  1   2   3   4   5
 45  44 323  30  24
```

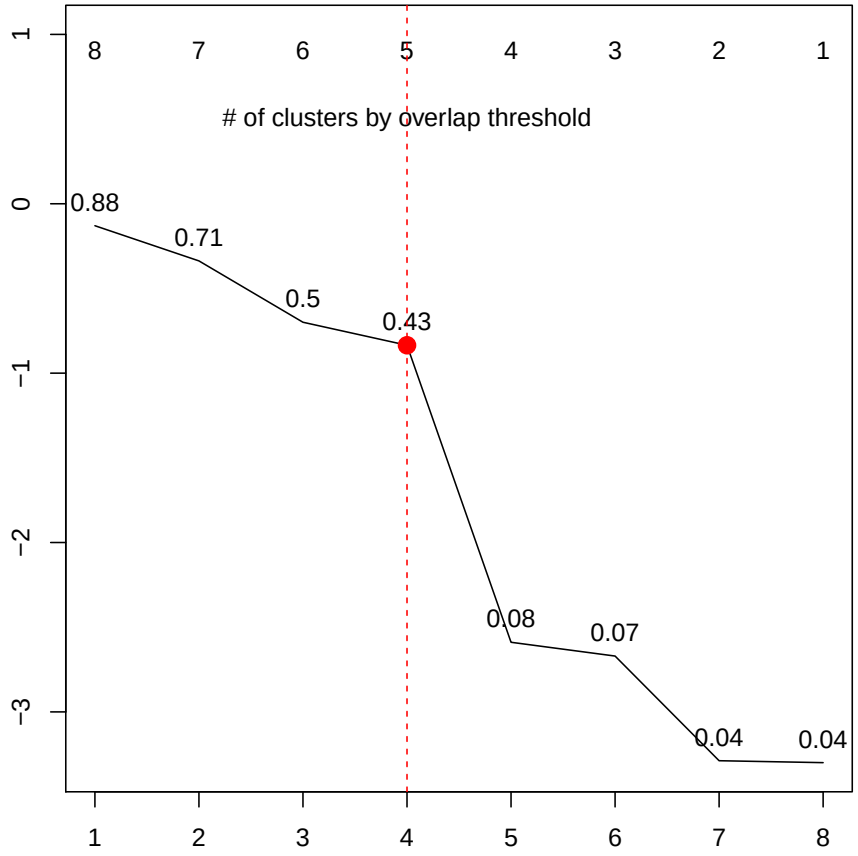

Figure 10: Overview of maximum cluster overlap observed in each merging step. Merging stops at 5 clusters, when the next two clusters to merge show an overlap differing significantly in mean to those from previous steps.

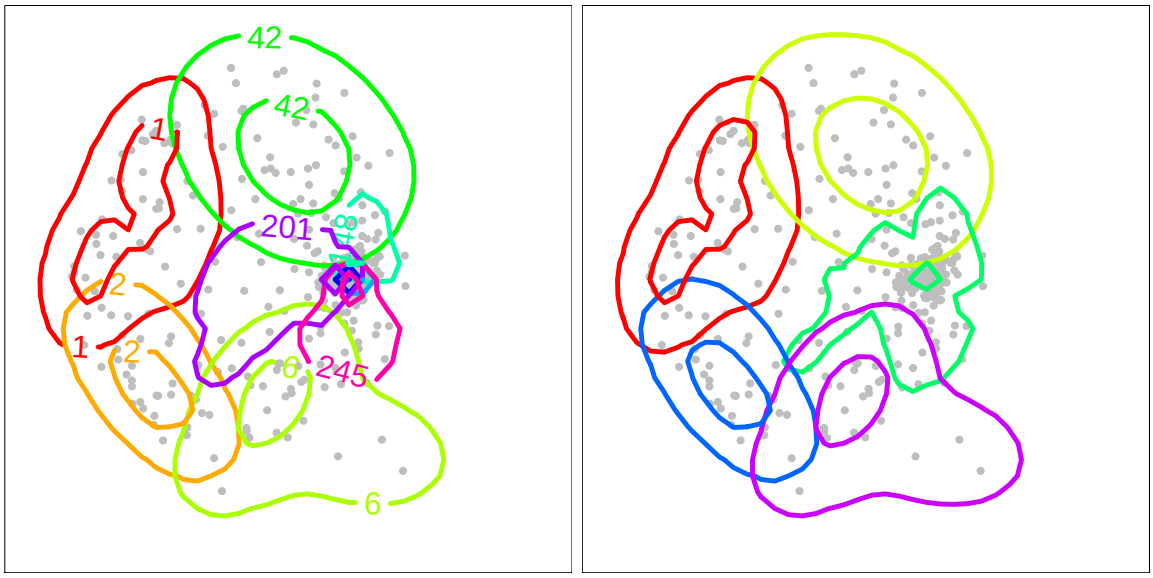

Figure 11: chroGPS^{genes} map with clusters at between-cluster distance of 0.2, and cluster density contours at 50 and 95 percent. Left: Unmerged. Right: Merged.

We plot the cluster contours before and after merging (Figure **??**). The merging step combined clusters from the central dense region of the map. These clusters had a low cluster-specific CCR, as shown in Figure **??**. After merging all cluster-specific CCR values were roughly ≥ 0.9. The code required to produce Figure **??** is provided below.

```
> par(mar=c(0,0,0,0),xaxt='n',yaxt='n')
> plot(mds1,point.cex=1.5,point.col='grey')
> for (p in c(0.95, 0.50)) plot(clus2,type='contours',k=max(cutree(h,h=0.2)),
+ lwd=5,probContour=p,drawlabels=TRUE,labcex=2,font=2)

> par(mar=c(0,0,0,0),xaxt='n',yaxt='n')
> plot(mds1,point.cex=1.5,point.col='grey')
> for (p in c(0.95, 0.50)) plot(clus3,type='contours',k=max(cutree(h,h=0.2)),
+ lwd=5,probContour=p)
```

And as we did before, we can have a look at per-cluster CCR values before and after cluster merging (**??**)

```
> plot(clus2,type='stats',k=max(cutree(h,h=0.2)),ylim=c(0,1),lwd=2,
+ ylab='CCR',xlab='Cluster ID')

> plot(clus3,type='stats',k=max(cutree(h,h=0.2)),ylim=c(0,1),lwd=2,
+ ylab='CCR',xlab='Cluster ID')
```

3.9 Studying the epigenetic profile of selected clusters

A classical way of analyzing a group epigenes is to look at the distribution of their epigenetic marks, that is, looking at their epigenetic profile. A quick look into a heatmap-like plot produced with the `heatmap.2` function from the `gplots` package

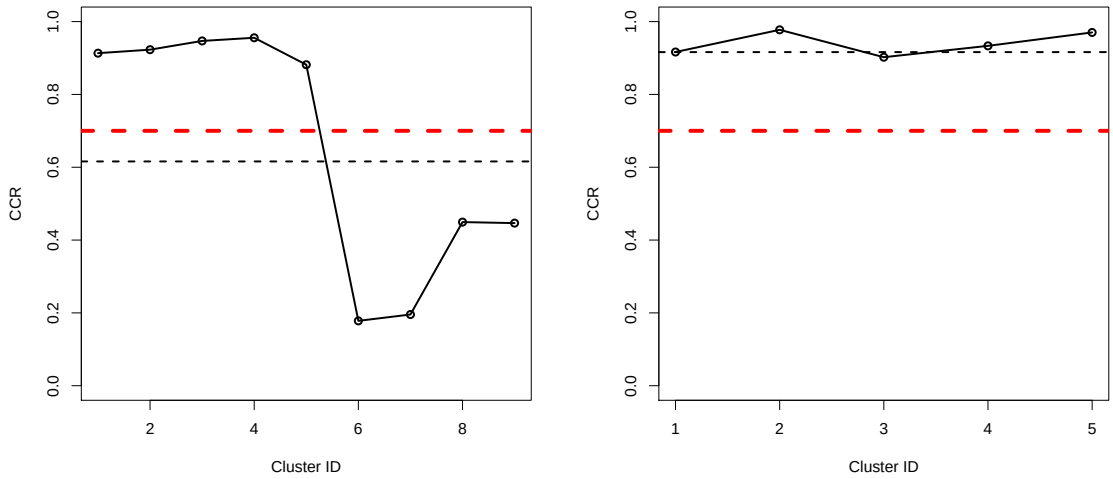

Figure 12: Per-cluster (dots and continuous line) and global (dashed line) misclassification rate for the clusters shown in Figure **??**. Red dashed line indicates an arbitrary threshold of 0.7 CCR. Left: Unmerged. Right: Merged

can highlight specific enrichments or depletions of certain epigenetic factors in a given cluster. As expected, this matches the distribution of epigenetic factors seen in Figure **??**.

```
> p1 <- profileClusters(s2.tab, uniqueCount = TRUE, clus=clus3, i=max(cutree(h,h=0.2)),
+ log2 = TRUE, plt = FALSE, minpoints=0)
> # Requires gplots library
> library(gplots)
> heatmap.2(p1[,1:20],trace='none',col=bluered(100),margins=c(10,12),symbreaks=TRUE,
+ Rowv=FALSE,Colv=FALSE,dendrogram='none')
```

3.10 Beyond R: exporting chroGPS maps to Cytoscape

No doubt R is a wonderful environment, but it has its limitations and it may not be the most direct software to use for biologists. Having that in mind, we developed a function for exporting any of the MDS graphics from our chroGPS maps as an XGMML format network for the widely used Cytoscape software `http://www.cytoscape.org`, (**?**). Network nodes are identified by their factor or epigene name, so that importing external information (i.e. expression values) or expanding the original chroGPS object with for instance external regulation networks, Gene Ontology enrichments, etc, becomes natural for Cytoscape users.
Even if no edges are returned, the exported network keeps the relative distribution of elements as seen in chroGPS, in order to keep the distances between the original elements intact. For three-dimensional maps Cytoscape 3D Renderer is required.

```
> # For instance if mds1 contains a valid chroGPS-factors map.
> # gps2xgmml(mds1, fname='chroGPS_factors.xgmml', fontSize=4,
```

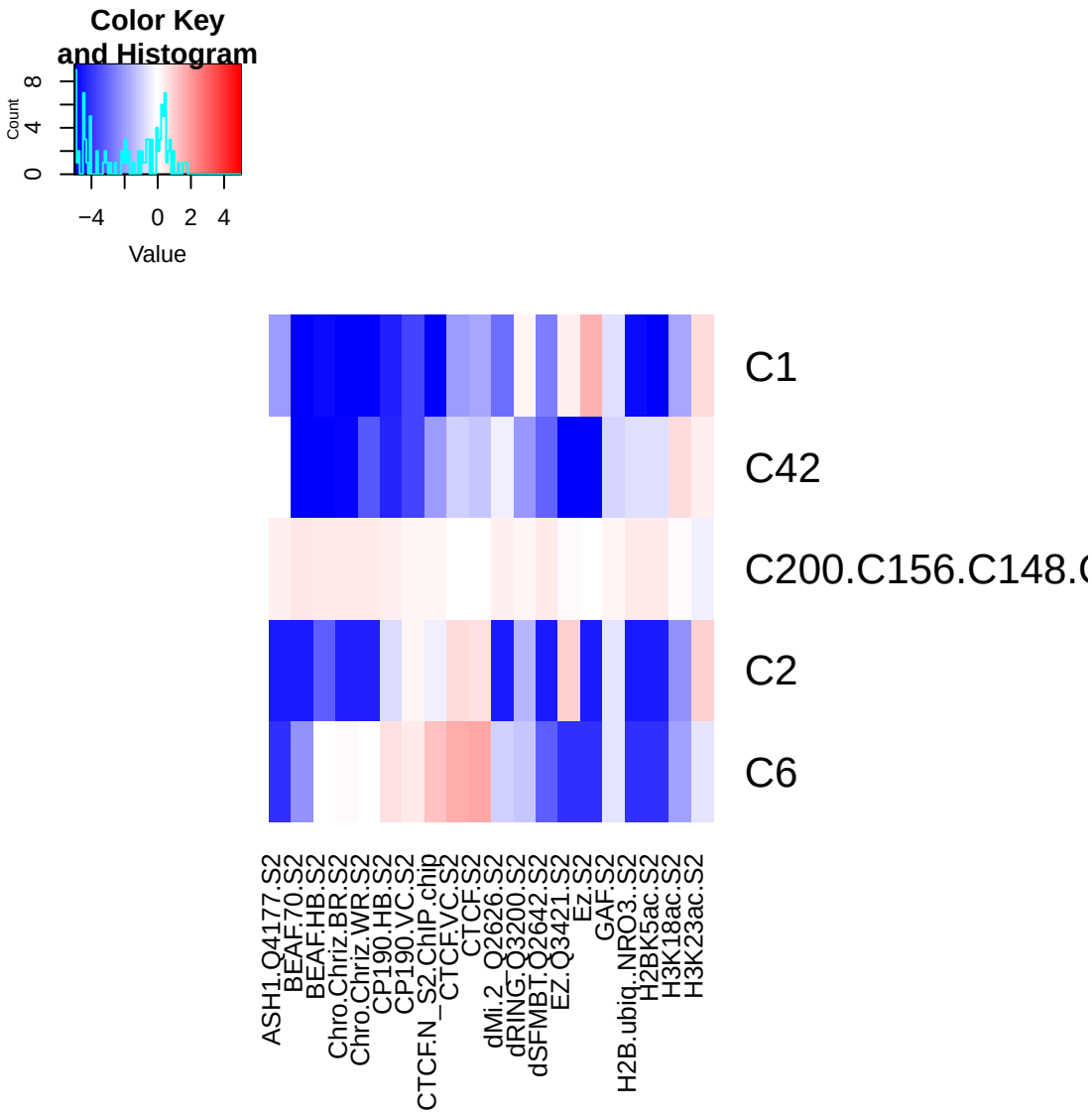

Figure 13: chroGPS^{genes} profile heatmap of the 9 unmerged clusters presented at Figure **??** after unsupervised merging of overlapping clusters (showing 20 first factors for visualization purposes). Merged clusters get concatenated names from the original clusters.

```
> # col=s2names$Color, cex=8)
> # And use Cytoscape -> File -> Import -> Network (Multiple File Types)
> # to load the generated .xgmml file
```

And this is everything, hope you enjoy using chroGPS as much as we did developing it !

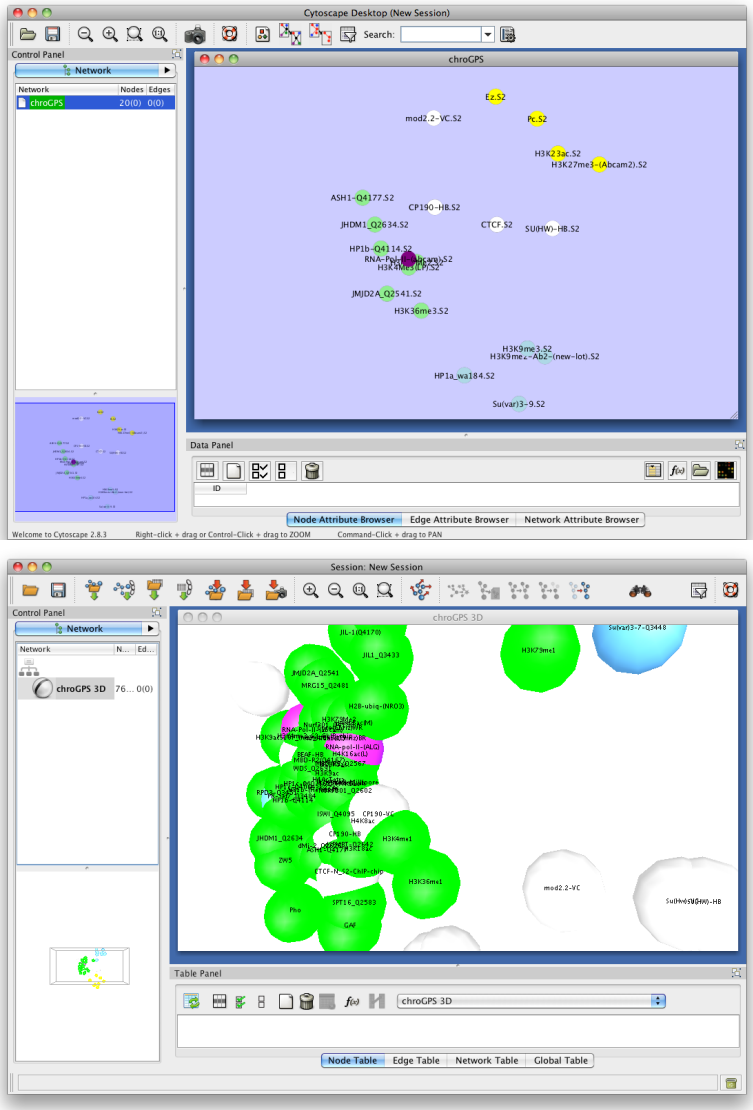

Figure 14: chroGPS^{factors} network exported and visualized in Cytoscape. Top: 2D. Bottom: 3D.